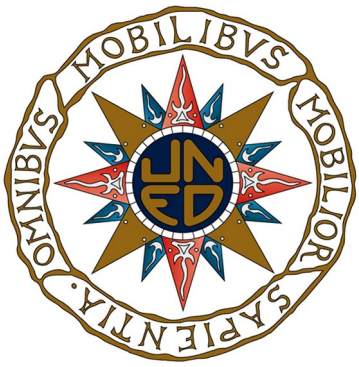


Universidad Nacional de Educación a Distancia

Escuela Técnica Superior de Ingeniería Informática

Proyecto de fin de Grado en Ingeniería Informática

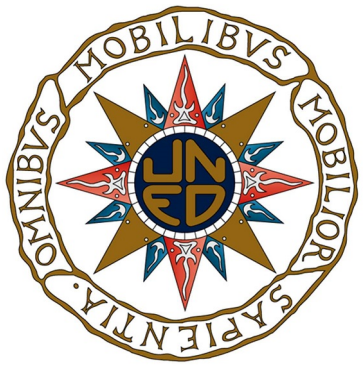
Gestión de datos de entrada y salida parametrizable IO Project

Adrián Maroto Huertas

Dirigido por: Alfonso Urquía Moraleda

Codirigido por: Carla Martín Villalba

Curso 2023/2024, convocatoria junio



Gestión de datos de entrada y salida parametrizable IO Project

Proyecto de fin de Grado en Ingeniería Informática

de modalidad específica

Realizado por: Adrián Maroto Huertas

Dirigido por: Alfonso Urquía Moraleda

Codirigido por: Carla Martín Villalba

Fecha de lectura y defensa: Mayo de 2024

Agradecimientos

A mi mujer Ángeles por todo su apoyo y comprensión durante este viaje.

Resumen

Este documento muestra el proceso de desarrollo y construcción de una nueva herramienta de gestión de datos de entrada y salida parametrizable denominada *IO Project*, la cual se pretende realizar a través del sistema extracción, transformación y carga (ETL) SQL Server Integration Services (SSIS), apoyándose en el sistema de gestión de bases de datos (BBDD) SQL Server, el cual contiene la información core almacenada en una BBDD para su funcionamiento, incluyendo las lógicas de uso por procesos, la trazabilidad y monitoreo de las ejecuciones realizadas, así como el detalle de las fases y tareas de cada proceso dentro de la herramienta.

En la actualidad, la necesidad del intercambio de datos y almacenamiento en BBDD de forma digital en las empresas se ha convertido en una necesidad fundamental de cara a mejorar la eficiencia operativa, garantizar la seguridad de los datos, facilitar su análisis, así como cumplir con las regulaciones normativas. Como se puede apreciar, esto se ha vuelto aún más crítico en la era actual, donde el dato es un activo muy valioso con capacidad para impulsar el crecimiento empresarial.

En términos generales, la herramienta contiene tres fases principales, las cuales permiten la ingesta de datos de diferentes orígenes, transformándolos según unas características específicas demandadas por cada proceso a generar, para finalmente poder depositarlos en un destino disponible, desde un modelo de BBDD, hasta una generación de un informe de salida.

Es por ello que surge este proyecto como una necesidad real dentro de un entorno de trabajo profesional, permitiendo centralizar todas las actividades comentadas, siendo totalmente dinámica, con la flexibilidad necesaria para adaptarse al mayor número posible de casuísticas encontradas en una empresa que trabaje con BBDD, permitiendo un ahorro de costes y tiempo importantes para una empresa o departamento técnico en particular.

La aplicación descrita en este documento está lista para ser usada y plenamente funcional en sus requisitos previstos para su elaboración.

Abstract

This document shows the development and implementation process of a new parameterizable input and output data management tool called *IO Project*, which is intended to be carried out through the ETL SSIS system, relying on the SQL Server management system, which contains the core information stored in a database for its operation, including process usage logic, traceability and monitoring of the executions carried out, as well as the details of the phases and tasks of each process within the tool.

Currently, the need for digital data exchange and storage in database in companies has become a fundamental need in order to improve operational efficiency, guarantee data security, facilitate its analysis, as well as comply with regulatory regulations. As you can see, this has become even more critical in the current era, where data is a very valuable asset with the capacity to drive business growth.

In general terms, the tool contains three main phases, which allow the ingestion of data from different sources, transforming them according to specific characteristics demanded by each process to be generated, to finally be able to deposit them in an available destination, from a database, until a generation of an output report.

That is why this project emerges as a real need within a professional work environment, allowing all the aforementioned activities to be centralized, being totally dynamic, with the necessary flexibility to adapt to the greatest possible number of cases found in a company that works with glsbbdd, allowing significant cost and time savings for a particular company or technical department.

The application described in this document is ready to use and fully functional in its requirements foreseen for its elaboration.

Palabras clave

Modelo de datos, Base de datos relacional, Herramienta gestión de datos, Compartir información, Servicios server.

Keywords

Data model, Relational database, Data management tool, Information sharing, Server services.

Índice

Índice de figuras

Índice de tablas

Capítulo 1

Introducción, objetivos y estructura

1.1. Introducción

En la actualidad, la necesidad del intercambio de datos y almacenamiento en BBDD de forma digital en las empresas se ha convertido en una necesidad fundamental de cara a mejorar la eficiencia operativa, garantizar la seguridad de los datos, facilitar su análisis, así como cumplir con las regulaciones normativas. Como se puede apreciar, esto se ha vuelto aún más crítico en la era actual, donde el dato es un activo muy valioso con capacidad para impulsar el crecimiento empresarial.

Dicho intercambio de información se ha convertido en un estándar entre compañías, sin embargo, a lo largo de mi trayectoria he podido ver diversos problemas derivados de la forma en la que se produce dicho intercambio, o como se gestionan de forma interna estos procesos, pudiendo apreciar una falta de soluciones que llevasen a la compañía o equipos de datos a una mayor eficiencia y optimización de tiempos en muchos casos necesaria.

Centrándonos en los problemas puramente técnicos, habitualmente se tiende al uso de herramientas de tipo ETL para extraer, transformar y cargar la información entre ecosistemas de datos. Aunque tales herramientas suelen ser muy útiles para ejercicios o procesos de tipo *adhoc*, es decir, específicamente elaborados para un problema o fin preciso y, por tanto, no utilizable para otros propósitos, se ven lastradas cuando se pretenden dinamizar procesos de intercambio de información centralizados en una única solución. Adicionalmente, el tipo de herramientas actuales requieren una curva de aprendizaje elevada, ya sea mediante el uso de interfaz, o directamente por código.

Es por ello que surge este proyecto como una necesidad real dentro de un entorno de trabajo profesional. Como herramienta esta solución permite centralizar todas las actividades comentadas, siendo totalmente dinámica, con la flexibilidad necesaria para adaptarse al mayor número posible de casuísticas encontradas en una empresa que trabaje con BBDD, permitiendo un ahorro de costes y tiempo importantes para una empresa o departamento técnico en particular.

1.2. Objetivos

El objetivo principal de este proyecto es el desarrollo de una nueva herramienta de gestión de datos de entrada y salida parametrizable denominada *IO Project*, la cual se pretende realizar a través del sistema ETL SSIS, apoyándose en el sistema de gestión de BBDD SQL Server, el cual contiene la información core almacenada en una BBDD para su funcionamiento, incluyendo las lógicas de uso por procesos, la trazabilidad y monitoreo de las ejecuciones realizadas, así como el detalle de las fases y tareas de cada proceso dentro de la herramienta.

En términos generales, la herramienta contendrá tres fases principales, las cuales permitirán la ingesta de datos de diferentes orígenes, transformándolos según unas características específicas demandadas por cada proceso a generar, para finalmente poder depositarlos en un destino disponible, desde un modelo de BBDD, hasta una generación de un informe de salida.

La solución contendrá cinco paquetes, o dtsx dentro del proyecto de SSIS, los cuales desarrollarán una funcionalidad concreta la cual se detalla a continuación.

- Main_process: Proceso principal encargado de gestionar y orquestar la ejecución dentro de la herramienta IO, es el punto de entrada y único ejecutable desde un automatizador de procesos, como podría ser *Task Scheduler* de *Windows*.
- Ingest_process: Proceso de ingesta, el cual permitirá ingerir datos de origen procedentes de un archivo plano en diferentes formatos, o desde una BBDD, tanto local como remota. Cuando se trata de archivos, existen dos funcionalidades adicionales:
  - Descarga desde un FTP.
  - Descompresión de archivos en formato *ZIP*.
- Transformation_process: Proceso que generará las transformaciones necesarias para adaptar la información al modelo de datos de destino. Incluye el proceso de delivery para identificar de forma univoca cada ejecución.
- Persistence_process: Proceso que realizará las operaciones calculadas previamente por el proceso de transformación las cuales pueden ser de los siguientes tipos.
  - INSERT.
  - UPDATE.
  - DELETE.
- Output_process: Proceso que generará un archivo de salida a partir de los datos generados a través de una consulta embebida o almacenada en un procedimiento almacenado en BBDD. Adicionalmente dispondrá de las siguientes funcionalidades para el tratamiento del archivo una vez generado.
  - Compresión del archivo en formato *ZIP*.
  - Envío del archivo a un FTP remoto.
  - Generación de un correo de información, con posibilidad de adjuntar el archivo generado.
  - Acciones en archivos sobre el directorio local.
    - COPY.
    - MOVE.
    - DELETE.

- RENAME.

Adicional a estas fases principales, el proceso contendrá una capa de trazabilidad y monitorización, a modo de log, con el detalle pormenorizado de las ejecuciones realizadas, pudiendo enviar al usuario alertas por correo, o analizar la información tratada en cada una de las fases, en la BBDD del proyecto.

Se enumeran las tareas a realizar para completar el desarrollo de la aplicación.

1. Establecimiento de los requisitos del proyecto.
2. Estudio teórico del montaje de la arquitectura para el problema planteado.
3. Valoración de uso de las herramientas más adecuadas para desarrollar la solución.
4. Análisis técnico - funcional y generación de requisitos de la solución.
5. Diseño de la BBDD core del proyecto.
6. Desarrollo e implementación de la solución y herramientas de soporte.
7. Elaboración de casos de uso para verificar los requisitos.
8. Evaluación de la solución mediante los resultados obtenidos en los casos de uso.
9. Despliegue de la solución en la arquitectura desarrollada.
10. Automatización de procesos de prueba mediante el gestor de BBDD.
11. Escritura de la memoria del proyecto.

1.3. Estructura

La memoria del proyecto se ha organizado en ocho capítulos y tres anexos.

- Capítulo 1 - *Introducción, objetivos y estructura* 1: El primer capítulo expone un marco actual sobre la problemática del intercambio de datos entre empresas y la forma de desarrollar sus procesos, ofreciendo una posible solución mediante la solución presentada en este proyecto. Se establecen los objetivos principales a realizar y se enumera la estructura de la documentación del proyecto.

- Capítulo 2 - *Revisión de procesos de gestión de datos* 2: En el segundo capítulo muestra las metodologías y herramientas empleadas para la elaboración del proyecto, trabajos y código de soporte desarrollado por terceros, así como trabajos similares en línea con la solución planteada.

- Capítulo 3 - *Análisis* 3: El tercer capítulo se encarga de analizar el comportamiento y las funcionalidades del proyecto dividiendo el contenido en la especificación de requisitos y los diagramas requeridos de cara a poder establecer una especificación inicial que sirva como base para el desarrollo del producto final.

- Capítulo 4 - *Diseño* 4: El cuarto capítulo desarrolla el diseño aplicado a partir de la especificación inicial, dividiéndose por un lado en la arquitectura del hardware y software y por otro en el diseño de la BBDD del proyecto el cual nos permite establecer las funcionalidades de forma dinámica.

- Capítulo 5 - *Implementación* 5: En el quinto capítulo se analiza el resultado de las tareas de implementación desde el punto de vista de la funcionalidad obtenida, así como las metodologías y herramientas utilizadas. Describiendo el desarrollo completo tanto de las herramientas de soporte como del core de la solución en SSIS.

- Capítulo 6 - *Pruebas* 6: El sexto capítulo se centra en las pruebas o casos de uso que nos permiten comprobar el correcto funcionamiento del proyecto en cumplimiento con los requisitos funcionales. Adicionalmente a los casos de uso, se define una BBDD de soporte con datos de ejemplo para dotar de consistencia a las pruebas realizadas.

- Capítulo 7 - *Planificación y costes del proyecto* 7: El séptimo capítulo muestra la planificación del proyecto, la cual se divide en las fases por las que ha estado compuesto, un diagrama de Gantt, así como una estimación de costes asociados. El capítulo concluye con unos comentarios acerca de los beneficios de una correcta planificación.

- Capítulo 8 - *Conclusiones y trabajos futuros* 8: El octavo capítulo finaliza con las conclusiones obtenidas tras la realización del proyecto, analizando los objetivos y tareas completadas, definiendo además posibles mejoras y actualizaciones en caso de querer ampliar la solución.

- Anexo A - *Manual de despliegue* A: El primer anexo describe la instalación de todos los elementos necesarios para que la aplicación se pueda probar y ejecutar en un entorno bajo una determinada arquitectura.

- Anexo B - *Manual de usuario* B: El segundo anexo resume los puntos más relevantes a tener en cuenta como usuario a la hora de generar un nuevo proceso en la herramienta y aporta una serie de pautas de cara a revisar logs o solventar errores.

- Anexo C - *Código fuente* C: El tercer anexo contiene el código fuente del proyecto, el cual incluye scripts tanto para la solución SSIS como para la BBDD.

El soporte de entrega que acompaña a la memoria se ha organizado en los siguientes subdirectorios.

- Directorio *Documentación* contiene la memoria en formato *PDF*.

- Directorio *Código* contiene todos los scripts generados durante el proyecto.

- Directorio *SSIS* contiene la solución desarrollada en la herramienta de *Microsoft*.

- Directorio *IO_Files* contiene la estructura de directorios necesarios para la ejecución de procesos.

Capítulo 2

Revisión de procesos de gestión de datos

2.1. Introducción

Tras hablar acerca del proyecto y los objetivos propuestos para su consecución, pasamos a este segundo capítulo donde se exploran los procesos de gestión de datos. En particular, se describen las herramientas empleadas en el proyecto técnico presentado, las cuales se consideran clave para el correcto funcionamiento de la aplicación, después se detallan las herramientas y trabajos realizados por otros que he necesitado como soporte para completar algunas de las funcionalidades de la aplicación, y finalmente las alternativas que tenemos disponibles en el mercado para el intercambio de datos de una forma similar a la presentada, orientadas tanto a entornos on-premise como en la nube.

2.2. Herramientas empleadas

Mi enfoque durante el desarrollo del proyecto siempre ha sido el de emplear una variedad de herramientas y metodologías asegurando la eficiencia y efectividad del proceso. Para ello, la mayoría de software usado está basado en tecnología *Microsoft* debido a mi expertise adquirido en mi carrera profesional, esto me ha permitido poder comenzar sobre una base muy amplia de conocimientos, pudiendo enfocarme en el core del proyecto, haciéndome ahorrar un tiempo muy valioso evitando tener que aprender acerca de las herramientas. Por otro lado, aunque en un principio no estaba contemplado el uso de C# o PowerShell, según iba avanzando el desarrollo fue necesario su inclusión, en algunos casos para incorporar funcionalidades consideradas como básicas en el diseño de la solución como se puede ver en el siguiente apartado, y en otros para dejar la herramienta mejor rematada de cara al usuario final. En cualquier caso en esta sección nos centramos en lo que se considera core para el desarrollo de la aplicación, por lo que recomiendo ir al capítulo *Diseño* 4 para ver en detalle la estructura general del entorno. Pasamos a ver en detalle el core en el siguiente listado.

- Windows Server 2022 es la última versión del sistema operativo de servidor de *Microsoft* que me ha permitido alojar el proyecto, en mi caso he usado una versión trial gratuita la cual se puede descargar a través de este enlace [1]. Se trata de un sistema operativo muy robusto que favorece mucho la integración con software de la misma casa, considerado muy intuitivo por su interfaz gráfica, siendo además muy similar al sistema operativo de uso doméstico *Windows*. No he tenido ningún problema de rendimiento o errores durante todo el desarrollo, como principal ventaja, me ha permitido configurar e integrar todas las funcionalidades necesarias a nivel de servidor con sus propios roles, a excepción del servidor SMTP el cual se ha eliminado como característica.

- SQL Server 2022 es la última versión del sistema de gestión de bases de datos relacionales de *Microsoft*, una de las herramientas imprescindibles en el proyecto, de uso gratuito en

su versión developer [2]. No solo es clave para almacenar las BBDDs de IO o *VehicleSales* para los casos de uso, sino que posee una integración completa con la herramienta SSIS a través de la BBDD *SSISDB*, sobre la que poder desplegar soluciones de este tipo y ejecutarlas a través del motor. Por otro lado, posee el servicio SQL Server Agent el cual me ha permitido orquestar y automatizar la ejecución de los procesos preparados en los *Casos de uso* 6.3. Considero que es uno de los motores de BBDD más usados a nivel empresarial el cual ofrece muchas ventajas, entre las que destaco la integración con software de la casa, o uno de los mejores *IDE* (entorno integrado de desarrollo) SQL Server Management Studio para administrar cualquier infraestructura de SQL [3], ya sea on-premise o en la nube, proporcionando herramientas para configurar, supervisar y administrar instancias SQL Server y BBDD, en concreto he usado la versión 19 y la interfaz, así como el número de opciones y propiedades que posee son muy buenas.

- Visual Studio Community 2022 *IDE* completo, extensible y gratuito para crear aplicaciones de escritorio, web y servicios en la nube. Dentro del *IDE* se ha instalado la extensión SQL Server Integration Services [4], la cual ha sido otra de las herramientas imprescindibles en el proyecto. SSIS es una plataforma de *Microsoft* que permite la creación de procesos ETL mediante interfaz gráfica, la cual puede extraer y transformar datos de diversos orígenes, para cargarlos después en diferentes destinos. La solución creada se ha dividido en diferentes paquetes con la extensión por defecto dtsx. Adicionalmente, se ha añadido diferente funcionalidad personalizada en lenguaje C#, gracias a que permite desarrollar código embebido con la *tarea Script*, la cual ha permitido incluir envío personalizado de correos en formato *HTML*, o la lectura interna de ficheros *XLSX* una vez instaladas las librerías *Office necesarias* (ver la sección *Instalación de librerías Office* A del *Manual de despliegue* A).

- PowerShell o entorno de línea de comandos y lenguaje de scripting de *Microsoft* para la automatización y administración de sistemas *Windows*. Permitiendo a los usuarios realizar una variedad de tareas administrativas y de configuración de manera eficiente y programática [5]. En particular se ha usado la versión que viene preinstalada por defecto en el sistema, y ha permitido desarrollar las herramientas de soporte adicionales a SSIS, en conjunción con el código de terceros que se puede ver en el siguiente apartado. Para el desarrollo del scripting me he apoyado en Windows PowerShell ISE o aplicación host, pudiendo ejecutar comandos y escribir, probar y depurar los scripts mediante interfaz gráfica. Para más información, tenemos por un lado la documentación oficial, y por otro el detalle con la implementación de las herramientas desarrolladas para el proyecto en PowerShell en la sección *Herramientas de soporte* 5.2 del capítulo 5.

2.2.1. Herramientas de compresión y descompresión de archivos

Durante el desarrollo del proyecto, me he enfrentado a diferentes problemas derivados por un lado de la complejidad de la solución y por otro de limitaciones debido al software escogido. Todos ellos han sido finalmente solventables, salvo en las siguientes dos excepciones.

Uno de los retos que se consideraban clave tanto en la fase de ingesta, como en la de output, era la posibilidad de poder comprimir o descomprimir archivos, independientemente de que estuviesen o no protegidos por contraseña. Es algo que considero se da de forma muy habitual en el intercambio de datos, bien por la reducción en el peso de los archivos, o por la robustez a nivel de seguridad que aporta el cifrado mediante contraseña. Aunque la idea era clara, al

ponerla en la práctica y revisar el cuadro de herramientas de SSIS, me di cuenta de que el software no disponía de ninguna tarea relacionada con esta función, por lo que se trató de buscar una solución que fuese implementable a través de SSIS, la cual permite incluir scripts o ejecutables externos mediante la *tarea Ejecutar proceso*, por lo que a través de PowerShell encontré el módulo gratuito *7Zip4Powershell* que permite la creación y extracción de archivos *7-Zip*. Funciona tanto con x86 como con x64 y utiliza *SevenZipSharp* como contenedor de la API de 7zip. Se puede instalar directamente desde PowerShell. A continuación se comparten los enlaces de la herramienta.

- Github del autor Thomas Freudenberg el cual contiene el código y las instrucciones de uso del módulo [6].
- Galería PowerShell donde se encuentra ubicado el módulo *7Zip4Powershell*.
- Artículo de Jason Fossen donde describe algunos escenarios de uso con *7Zip4Powershell* [7].

Para ver en detalle como se ha implementado el módulo en el proyecto, ver el apartado *Compresión y descompresión de archivos* 5.3 del capítulo 5.

2.2.2. Herramientas para la conexión a servidores FTP/SFTP

De forma similar al problema de la compresión y descompresión de archivos, otra de las funcionalidades que se consideraban básicas dentro del desarrollo de la aplicación era la de conexión a servidores FTP/SFTP, con alta probabilidad el método más usado para el intercambio de archivos entre empresas, para este caso en particular si era conocedor de la *tarea FTP* de SSIS, sin embargo me enfrentaba a dos grandes limitaciones, por un lado el número de opciones disponibles es muy bajo en comparación con clientes de este tipo, por otro, no permite la conexión de tipo SSIS siendo esta una limitación muy importante. Debido a ello, se trató de buscar una solución que fuese implementable a través de SSIS, y de la misma forma que para el caso anterior a través de la *tarea Ejecutar proceso* se creó un script en PowerShell el cual incorpora la librería *WinSCP .NET* [8] de la famosa aplicación de software libre, en este caso la librería está disponible para *Windows* permitiendo la conexión SFTP empleando *SSH*, facilitando la transferencia segura de archivos. Una vez descargada la librería, se carga directamente en el script de PowerShell. A continuación se compoarten los enlaces de la herramienta.

- WinSCP .NET donde poder acceder a la descarga de la librería.
- Documentación del cliente para poder configurar la aplicación independientemente del tipo de versión que usemos.
- Github de *WinSCP* con el código de la herramienta.

Para ver en detalle como se ha implementado el módulo en el proyecto, ver el apartado *Carga y descarga de archivos por FTP/SFTP* 5.2.3 del capítulo 5.

2.2.3. SQLSearch

Por último, aunque esta herramienta no se ha implementado en la solución la he usado como soporte, SQLSearch de la empresa *Red Gate Software* ha sido de gran ayuda especialmente

para la búsqueda de cualquier término en SQL Server, en muchas ocasiones este ejercicio se complica en BBDDs complejas o de gran tamaño. Se trata de un plugin gratuito el cual se integra directamente en el *IDE Management Studio* [9] y te permite encontrar resultados sobre las BBDDs de un servidor en función de la búsqueda realizada. Funciona de manera muy rápida y eficiente, no solo encontrando todos los objetos que contienen el término, sino permitiendo adicionalmente saltar al objeto seleccionado del listado.

2.3. Herramientas existentes para la gestión de datos

Por último, en relación a los trabajos similares al proyecto realizado, es importante comenzar indicando que, aunque en la actualidad hay infinidad de herramientas en el mercado similares a SSIS o diferentes motores de BBDD como SQL Server, no he encontrado una solución capaz de dinamizar y parametrizar tanto el código como las variables, ya sean de configuración o de los procesos para un uso general como en el de nuestro proyecto, donde una vez desarrollado el core solo tienes que preocuparte por abstraer un nuevo proceso usando exclusivamente SQL Server, aportando una serie de ventajas con respecto a las aplicaciones por defecto. Esto no quiere decir que con las herramientas del mercado no se pueda realizar un proyecto como este, si no que en todas ellas tendríamos que realizar un desarrollo extenso como el de esta solución, debido a que de manera inicial ninguna ofrece algo similar.

La solución planteada ha sido pensada para un uso *on-premise*, válido tanto para sistemas de gestión de BBDD OLTP u OLAP, particularmente mediante el uso de software *Microsoft*, donde considero que tengo un fuerte conocimiento y habilidades debido a mi trayectoria profesional. Sin embargo, a la hora de hablar de alternativas me gustaría indicar una serie de opciones diferentes a las planteadas para el proyecto.

En caso de tener que realizar el mismo proyecto orientado a una solución de nuevo *on-premise* pero en este caso de código libre, probablemente lo integraría de la siguiente manera.

- PostgreSQL como motor de BBDD donde almacenar los datos de IO. Considero que en la actualidad es uno de los motores más usados, sino el que más, además de ser de código abierto, tiene una consolidada reputación en el mercado debido a su confiabilidad, solidez de funciones y rendimiento. Adicionalmente se puede encontrar una gran cantidad de información que describe su instalación y uso ya sea a través de la documentación oficial o buscando en la web.

- Pentaho Data Integration, también conocido como *Kettle*, para usarla en la integración y transformación de datos o proceso ETL. Al igual que el caso anterior, se trata de una plataforma de código abierto (en su versión community), y posee lo mismos principios que SSIS, diseñada para extraer datos de múltiples fuentes, transformarlos y cargarlos en una variedad de destinos. Posee una interfaz gráfica intuitiva para la creación de flujos de trabajo, con una serie de componentes preestablecidos para su uso. En definitiva se trata de una alternativa muy usada en las empresas, con la que se trabaja de forma muy parecida a SSIS y de la que se puede encontrar mucha documentación en internet.

- Thunderbird como cliente de correo electrónico desarrollado por la *Fundación Mozilla*. Se trata de un gestor muy potente de código abierto, que permite gestionar múltiples cuentas de diferentes proveedores, filtrar, organizar y cifrar correos, adicionalmente se

puede integrar con otros servicios como *Google Calendar* permitiendo gestionar eventos de forma sencilla. Lo considero muy buena alternativa a *Microsoft Outlook*, con gran soporte de la comunidad y documentación.

Si por el contrario queremos ir a alternativas orientadas a la nube, con tareas ETL para soluciones *datalake* o *lakehouse*, las alternativas de mayor uso en la actualidad serían las detalladas a continuación.

- DBT (Data Build Tool). Es una herramienta de código abierto diseñada para la transformación de datos en almacenes en la nube. Se puede ejecutar tanto localmente como un servicio alojado en la nube. DBT se utiliza para definir, ejecutar y gestionar transformaciones de datos mediante archivos de configuración y consultas SQL, permitiendo la creación de plantillas y macros preconfiguradas. La clave de esta herramienta es que se puede integrar con las plataformas de datos en la nube más habituales como *Google BigQuery, Amazon Redshift, Snowflake o Azure Databricks* y que adicionalmente todo el código se realiza a través de lenguaje SQL, siendo una de sus mayores ventajas. Contiene documentación oficial.

- Azure Data Factory es el sucesor de SSIS integrado como parte de la plataforma de servicios en la nube de *Microsoft Azure*. Diseñado para crear procesos ETL tanto en la nube, como en entornos locales, amplíando de esta forma el abanico de posibilidades a nuestro alcance para el desarrollo de posibles soluciones. Entre sus principales virtudes, nos permite crear flujos de trabajo (pipelines) mediante interfaz, se integra de forma nativa con el resto de servicios de *Microsoft Azure* como *Data Lake Storage o Synapse Analytics*, y posee la opción de escalabilidad automática según las necesidades del usuario para ofrecer siempre el rendimiento necesario en cada momento. Disponible documentación oficial o cursos gratuitos de *Microsoft* para aprender a usarla.

- Snowflake es una aplicación desarrollada por exingenieros de Oracle, proporciona una arquitectura de datos compartidos multiclúster con altos índices de rendimiento, escalabilidad y simultaneidad. Proporciona el acceso a una repositorio común de datos en la nube, incluyendo niveles de almacenamiento, procesamiento y servicios globales integrados lógicamente, aunque separados en el espacio físico. Los ámbitos de aplicación son Data Warehouse, Data Lake, ingeniería de datos, ciencia de datos, intercambio de datos y desarrollo de aplicaciones de datos. Al igual que con *DBT* esta herramienta se puede integrar con las plataformas de datos en la nube más habituales, siendo un análogo a *Synapse, Redshift o BigQuery*. En la actualidad es uno de los almacenes de datos más usados para soluciones en la nube. Ver su documentación oficial.

2.4. Conclusiones

En resumen, este capítulo ofrece una visión completa de las herramientas y metodologías empleadas para nuestra aplicación de intercambio de datos entre diferentes orígenes y destinos, se detalla el uso de módulos, código y APIs realizadas por terceros que he integrado en el proyecto para conseguir disponer de todas las funcionalidades requeridas. Por último, se establece una visión de posibles alternativas tecnológicas y trabajos similares a considerar para proyectos de intercambio de datos. Esta revisión tecnológica me ha permitido mejorar el proyecto, al mismo tiempo que me he mantenido al tanto de las últimas tendencias en el mundo del dato.

Capítulo 3

Análisis

3.1. Introducción

Basándonos en los objetivos y descripción detallada del proyecto, en este capítulo vamos a realizar un análisis detallado mediante la toma de requisitos para poder obtener un prototipo funcional, adicionalmente se mostrarán los diagramas UML empleados para los diferentes casos de uso de los que hablaremos más adelante.

Es importante remarcar, por un lado que un análisis incorrecto puede derivar en un mal diseño, dando como resultado un producto deficiente, y por otro lado la necesaria capacidad de adaptación a los cambios que suceden, especialmente en un contexto ágil. Este concepto de enfoque dinámico y flexible se ha tratado de mantener durante el desarrollo del proyecto, dando lugar a modificaciones en el análisis permitiendo refinar los requisitos expuestos en este capítulo.

3.2. Especificación de requisitos

Durante la especificación de requisitos, se debe recopilar, analizar y documentar las necesidades y expectativas de los usuarios, stakeholders y otras partes interesadas en el proyecto. Se trata de un proceso fundamental para comprender las necesidades o expectativas del cliente. Es relevante indicar que para satisfacer estas necesidades se debe emplear una comunicación fluida y transparente con el cliente de cara a obtener toda la información relevante.

Una vez recopilados, es necesario documentarlos de manera clara y concisa en un documento de especificación como el que tenemos en el presente capítulo. Este servirá como guía para el equipo de desarrollo durante todo el ciclo de vida del proyecto. Se trata de un proceso iterativo y continuo, donde los requisitos evolucionan a lo largo del tiempo tras obtener una mejor comprensión del problema y de las soluciones posibles.

Se distinguen a continuación los requisitos funcionales, no funcionales, y aspectos no contemplados en el desarrollo.

3.2.1. Requisitos funcionales

Un requisito o requerimiento funcional es una característica que un producto, usualmente un software, debe poseer para satisfacer las necesidades de un usuario o cliente de negocio. Para la solución propuesta se podrían indicar los siguientes.

Gestión de Datos de Entrada y Salida

1. La herramienta debe permitir la ingesta de datos desde diferentes orígenes, como archivos de tipo estructurado o semi-estructurado, BBDDs locales y remotas.

2. La herramienta debe ingestar la información en la BBDD de la solución mediante el uso de plantillas de metadatos independientes para cada proceso.

3. La herramienta debe ser capaz de transformar los datos en función a las reglas o características específicas de cada proceso.

4. La herramienta debe realizar operaciones de persistencia como *INSERT, UPDATE y DELETE* en la BBDD de destino.

5. La herramienta debe permitir la escritura de datos en un destino disponible, ya sea un modelo de BBDD o la generación de un informe de salida.

6. La herramienta debe escribir la información en el destino mediante el uso de plantillas de metadatos independientes para cada proceso.

Orquestación de Procesos

1. La herramienta debe contener un proceso principal que gestione y orqueste la ejecución de los procesos.

2. La herramienta debe ser capaz de reiniciar un proceso desde donde terminó su ejecución la anterior vez.

3. La herramienta debe estructurarse en diferentes fases modulables que permitan lanzar etapas de forma individual o procesos completos.

4. La herramienta debe permitir la ejecución de procesos de forma desatendida bajo una programación específica.

5. La herramienta debe impedir lanzar un proceso del mismo tipo sin haber finalizado una entrega anterior.

Funcionalidades Específicas

1. La herramienta debe poder encriptar o desencriptar la información de carácter sensible almacenada en BBDD.

2. La herramienta debe ser capaz de descargar o subir archivos a un servidor FTP.

3. La herramienta debe ser capaz de comprimir o descomprimir archivos en formato *ZIP* con o sin contraseña.

4. La herramienta debe permitir el envío de correos informativos al cliente incluyendo elementos adjuntos en caso necesario.

5. La herramienta debe permitir realizar acciones sobre los archivos en nuestros directorios locales.

Monitorización y Trazabilidad

1. La herramienta debe proporcionar una capa de trazabilidad y monitorización detallada de las ejecuciones realizadas.

2. La herramienta debe enviar alertas por correo electrónico al usuario que contengan la información de la ejecución independientemente del resultado (éxito, error, advertencia).

3. La herramienta debe permitir historificar información y poder disponer de ella en todo momento.

4. La herramienta debe controlar archivos previamente procesados para evitar duplicar información.

5. La herramienta debe ofrecer la posibilidad de continuar su ejecución ante la aparición de alertas.

3.2.2. Requisitos no funcionales

En el caso de un requisitos o requerimiento no funcional, deben describir las limitaciones y restricciones del sistema, pero no afecta directamente su funcionalidad. Se muestran a continuación los requerimientos no funcionales de la solución.

1. La herramienta debe ser eficiente en términos de rendimiento, garantizando tiempos de ejecución aceptables independientemente del volumen de datos.

2. La herramienta debe ofrecer un método de autenticación seguro de acceso al sistema de BBDD.

3. La herramienta debe asegurar la integridad de los datos almacenados en BBDD.

4. La herramienta debe garantizar la seguridad de los datos durante las fases del proceso.

5. La herramienta debe cumplir con las normativas de seguridad y privacidad de datos.

6. La herramienta debe ser capaz de manejar un aumento en el volumen de datos y procesos sin degradación del rendimiento.

7. La herramienta debe ser confiable y tolerable a fallos asegurando la continuidad de los procesos.

3.2.3. Aspectos no contemplados

La herramienta de gestión de datos de entrada y salida *IO Project* no es un proyecto cerrado, durante el desarrollo de la solución han ido apareciendo diferentes funcionalidades que no estaban planteadas en la primer versión, y por tiempo o limitaciones en la arquitectura no han sido posible añadirlas. El siguiente listado de funcionalidades podrían incluirse en versiones posteriores del producto.

1. Ampliación de conexión a servidores SFTP para securizar con este protocolo el intercambio de datos.

2. Generación de interfaz de usuario para el administrador del sistema.

3. Uso de consultoría remota a través de la virtualización de datos con Polybase en SQL Server.

4. Añadir plantillas en la fase de ingesta y output para lenguajes semi estructurados o de etiquetas como JSON o XML.

5. Abstraer el proceso de delivery en un paquete dtsx de la fase de transformación.

6. Desarrollo de sistema de Backup y recuperación de datos completo en caso de fallos o pérdida de datos.

A medida que el producto se vaya usando irán surgiendo nuevas funcionalidades no contempladas en los primeros análisis. Para más información ver el apartado *Trabajos futuros* 8.3 del capítulo 8.

3.3. Diagrama de casos de uso

Los casos de uso son un tipo de diagramas UML, que nos permiten mostrar de forma visual las distintas acciones que un usuario puede realizar en un sistema, además del resto de actores que participan en él. En la *Figura 3.1* se muestra el diagrama principal de casos de uso como resultado del análisis realizado, ofreciéndonos un detalle visual acerca de los requisitos funcionales y sus dependencias.

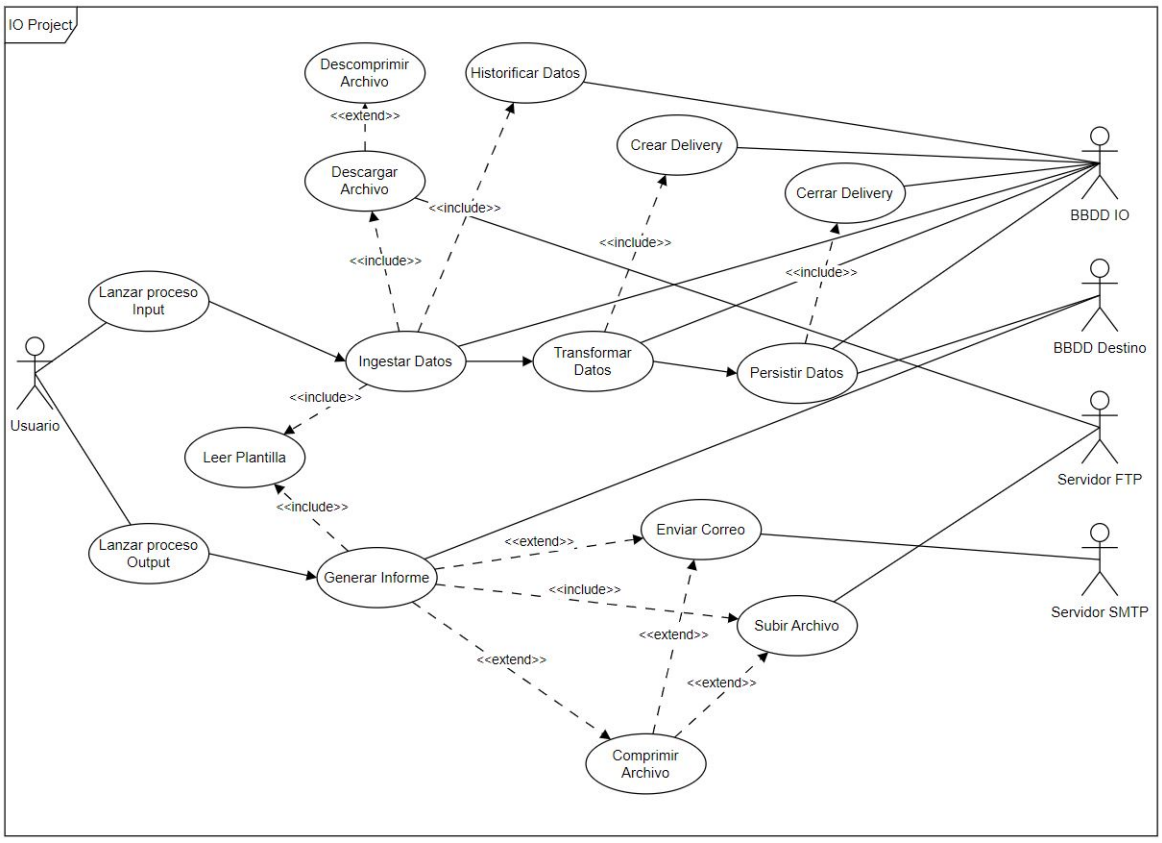


Figura 3.1: Diagrama de casos de uso

Adicional al diagrama se definen los escenarios de casos de uso propuestos detallando los actores involucrados, el escenario principal de éxito, así como escenarios alternativos en caso de haberlos.

Caso de uso 1: Lanzar proceso Input

Actores: Usuario, BBDD IO, BBDD Destino, Servidor FTP.

Descripción: El sistema deberá permitir persistir en un destino los datos ingestados.

Escenario principal

1. El usuario lanza un proceso input en la herramienta.
2. El sistema ejecuta la fase de ingesta.
3. El sistema descarga el archivo del servidor FTP desde la fase de ingesta.
4. El sistema lee la plantilla del proceso desde la fase de ingesta.
5. El sistema almacena los datos del archivo en la BBDD IO desde la fase de ingesta.
6. El sistema historifica los datos del archivo en la BBDD IO desde la fase de ingesta.
7. El sistema ejecuta la fase de transformación.
8. El sistema crea una delivery desde la fase de transfomación.
9. El sistema transforma los datos en función a una reglas en la BBDD IO desde la fase de transformación.
10. El sistema almacena los datos transformados en la BBDD IO desde la fase de transformación.
11. El sistema ejecuta la fase de persistencia.
12. El sistema lee los datos transformados en la BBDD IO desde la fase de persistencia.
13. El sistema almacena los datos transformados en la BBDD Destino desde la fase de persistencia.
14. El sistema cierra la delivery del proceso en ejecución desde la fase de persistencia.
15. El sistema envía un correo al usuario con el resultado del proceso.

Escenarios alternativos

- 3a. El sistema no puede conectarse al servidor FTP y lanza un error.
- 3b. El sistema detecta que no existe archivo para procesar en el servidor FTP y lanza una advertencia.
- 3c. El sistema descomprime el archivo en un directorio local desde la fase de ingesta.
- 8a. El sistema detecta que hay una delivery abierta y lanza un error.
- 13a. El sistema no puede conectarse a la BBDD Destino y lanza un error.

Caso de uso 2: Lanzar proceso Output

Actores: Usuario, BBDD IO, BBDD Destino, Servidor FTP, Sevidor SMTP.

Descripción: El sistema deberá permitir generar un informe de salida.

Escenario principal

1. El usuario lanza un proceso output en la herramienta.
2. El sistema ejecuta la fase de salida.
3. El sistema lee los datos en la BBDD Destino desde la fase de salida.
4. El sistema lee la plantilla del proceso desde la fase de salida.
5. El sistema genera el informe desde la fase de salida.
6. El sistema sube el archivo al servidor FTP destino desde la fase de salida.
7. El sistema envía un correo al usuario con el resultado del proceso.

Escenarios alternativos

- 3a. El sistema no puede conectarse a la BBDD Destino y lanza un error.
- 5a. El sistema comprime el informe en un directorio local desde la fase de salida.
    - 5aa. El sistema envía un correo electrónico adjuntando el fichero comprimido desde la fase de salida.
    - 5ab. El sistema sube el archivo comprimido al servidor FTP destino desde la fase de salida.
- 5b. El sistema envía un correo electrónico informativo desde la fase de salida.
- 6a. El sistema no puede conectarse al servidor FTP destino y lanza un error.
- 3c. El sistema comprime el archivo en un directorio local desde la fase de ingesta.
- 13a. El sistema no puede conectarse a la BBDD Destino y lanza un error.

3.4. Conclusiones

Durante este capítulo se ha realizado un análisis detallado mediante la toma de requisitos obteniendo un prototipo funcional, además de mostrar el principal diagrama de casos de uso indicando el detalle para cada uno de ellos. Gracias a este análisis se puede visualizar con claridad una idea global de las acciones que puede realizar la herramienta de intercambio de datos IO. Para mas información acerca de los diferentes casos de uso generados tras el desarrollo de la solución ir al capítulo 6.

Junto al capítulo de *Diseño* 4, ambos resultan clave de cara a poder definir de forma correcta

la base del proyecto, y facilitar la consecución de objetivos propuestos. Según se ha comentado en la introducción del capítulo durante el desarrollo de la solución se han encontrado nuevas funcionalidades, modificaciones sobre las existentes e incluso descartado algunas alternativas. El haber enfocado de una forma ágil y flexible la herramienta mediante el análisis ha supuesto su éxito.

Capítulo 4

Diseño

4.1. Introducción

Tras el análisis pormenorizado de la solución, el presente capítulo tiene como objetivo estudiar el diseño de la arquitectura que forma el proyecto, teniendo en cuenta las herramientas utilizadas para su desarrollo, se divide por un lado en el diseño hardware o estructura del servidor principal encargado de alojar el proyecto, y por otro en el diseño del software más centrado en el conjunto de procesos que forman la aplicación final. Por último se estudia el diseño y modelo de la BBDD IO la cual junto con la solución SSIS y SQL Server Agent forman la estructura general de la solución.

Aunque en un principio buscar un diseño acorde a la funcionalidad solicitada no parecía resultar complejo, durante el desarrollo se fue complicando debido a la falta de conocimiento de todos los servicios y herramientas que finalmente serían necesarias.

4.2. Diagrama de arquitectura del hardware

Comenzamos con el diseño de la arquitectura del hardware, pensada como un ecosistema completo capaz de funcionar por y para la solución desarrollada, en la *Figura 4.1* podemos ver el esquema general de los componentes necesarios para poder ejecutar el proyecto.

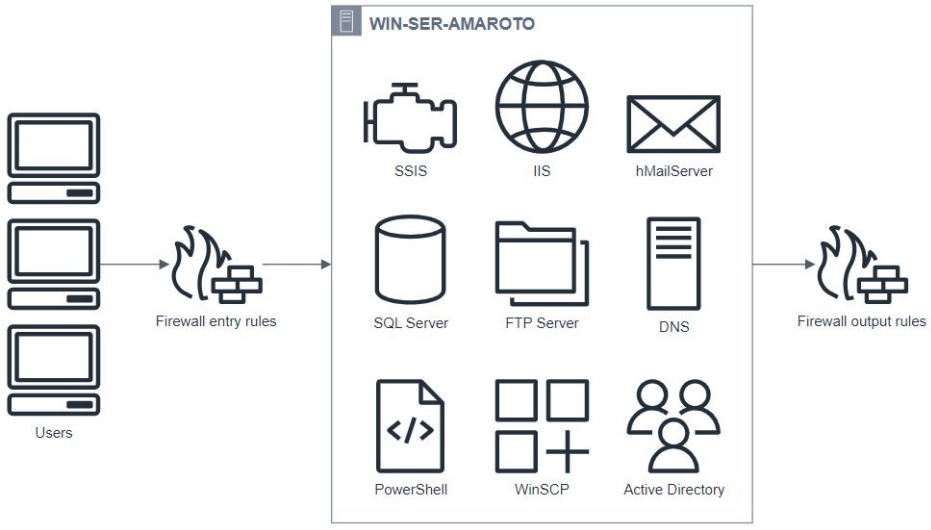


Figura 4.1: Estructura general del servidor

Tal y como se muestra en la *Figura 4.1*, para poder obtener todas las funcionalidades solicitadas es necesario disponer de las siguientes herramientas o servicios.

1. SSIS (SQL Server Integration Services): Herramienta ETL sobre la que se construye la solución que contiene las diferentes fases por las que pasa un proceso durante su ejecución, permitiendo el intercambio de datos entre un origen y un destino.

2. SQL Server: Sistema de gestión de BBDD relacionales de *Microsoft* sobre el que se ha generado la BBDD IO, como *VehicleSales*, se ha configurado SSIS y el servicio job agent para automatizar los procesos.

3. hMailServer: Servidor de correo de código abierto sobre el que se ha configurado las diferentes cuentas de correo necesarias para enviar y recibir las alertas de la solución, o poder enviar los correos de la fase de output.

4. Servicio DNS: Se agrega al servidor el rol *Servidor DNS* y se promueve a controlador de dominio, siendo el nombre del dominio raíz *data.es*, necesario tanto para el *Servidor web IIS* como para el *Servidor FTP*. Toda la configuración posterior del proyecto se realiza mediante DNS en lugar de IP.

5. Active Directory: Se han agregado los servicios de dominio de directorio activo, necesarios para el acceso de los usuarios externos al servidor asociados al mismo dominio, y poder gestionar su autenticación, autorización y recursos de red. Adicionalmente se requiere para crear el usuario y grupo de FTP para su configuración.

6. Servicio IIS (Internet Information Services): Se ha configurado el servidor web de *Microsoft* necesario para permitirnos alojar el servidor FTP sobre el que recepcionar o depositar los diferentes archivos.

7. Servidor FTP: Una vez activado el rol *Servidor web IIS*, se ha configurado un servidor FTP para permitir la transferencia de archivos a través de este protocolo. Adicionalmente se ha creado un certificado *SSL (Secure Sockets Layer)* para permitir sesiones encriptadas entre cliente y servidor.

8. PowerShell: El entorno de línea PowerShell ha sido necesario para realizar todas las herramientas de soporte necesarias para cumplimentar la funcionalidad del proyecto. Mediante estos scripts se ha podido integrar código que no hubiese sido posible realizar a través de SSIS o SQL Server. Es importante destacar que ha sido necesario modificar su política de ejecución a tipo *Bypass* [5]. Adicionalmente se instala el módulo *7Zip4Powershell* [6] para permitir comprimir y descomprimir archivos con o sin contraseña.

9. WinSCP: Cliente FTP/SFTP válido para su uso en *Windows Server*, gracias a este cliente se ha podido automatizar la conexión mediante ambos protocolos a través de PowerShell debido a que SSIS solo permite configurar tareas de tipo FTP.

10. Firewall: Se generan una serie de reglas de entrada y salida para permitir el intercambio de datos necesario con los orígenes y destinos, por un lado se abren los puertos necesarios para el servicio SMTP, por otro se configuran reglas para poder acceder al FTP en modo pasivo.

Considero esta estructura idónea para un proyecto basado en tecnología *Microsoft* debido a la facilidad de integración de su diferente software. Esta estructura completa me ha permitido alojar el proyecto IO mediante el que he podido desarrollar todas las funcionalidades solicitadas.

4.3. Diagrama de arquitectura del software

A continuación se va a describir la estructura general de la arquitectura del software así como la solución modular desarrollada en SSIS.

4.3.1. Estructura general

Una vez revisado el diseño de la arquitectura del hardware, pasamos ahora a hablar del software o aplicación desarrollada. En la *Figura 4.2* se puede apreciar la estructura general de la solución.

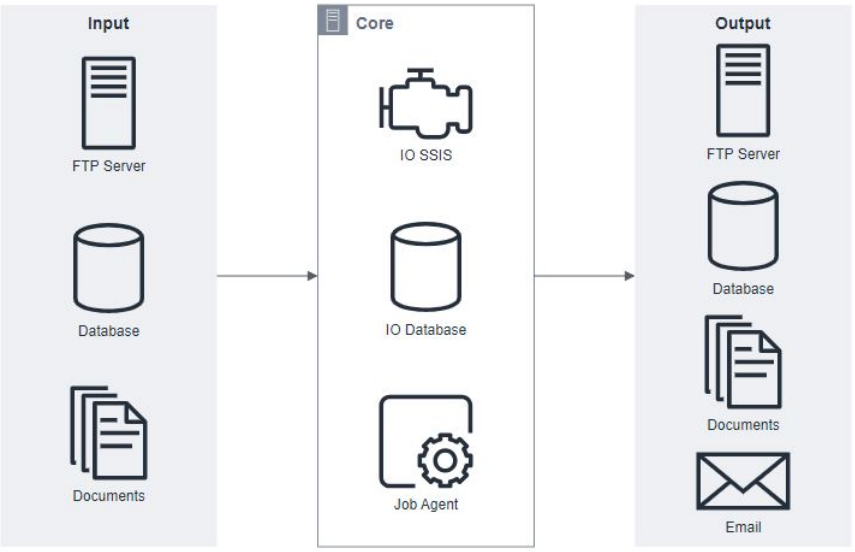


Figura 4.2: Estructura general de la solución

Para poder realizar el intercambio de datos esta estructura se ha compuesto de tres grandes apartados.

1. Input (Entrada): Se trata del origen de la información con los datos que van a ser procesados mediante el core. Dicha información puede provenir de diversas fuentes como una BBDD tanto local como remota, un archivo en diferentes formatos, o a través de un servidor FTP.

2. Core (Núcleo): Es el corazón de la aplicación, donde se lleva a cabo el procesamiento de los datos. Está compuesto por los siguientes elementos:

   a) Solución SSIS IO: Solución principal desarrollada con la herramienta SSIS, mediante la cual formamos el esqueleto compuesta por diferentes fases que serán nutridas a través de la información contenida en la BBDD. Mediante esta solución se realiza la lógica para extraer, transformar y cargar los datos desde la fuente de entrada hacia cualquiera de los posibles destinos, comúnmente conocido como proceso ETL.

   b) BBDD IO: BBDD desarrollada para almacenar todos los datos relacionados con el proyecto y sus procesos. Dicha BBDD consta de diferentes schemas, tablas, procedimientos almacenados y funciones, mediante los cuales podemos encapsular y dinamizar toda la información contenida en la solución SSIS, facilitando el almacenamiento, la manipulación, pero sobre todo la abstracción de los datos en una única

herramienta.

c) Job Agent: Herramienta disponible en SQL Server la cual usamos para poder programar y automatizar mediante jobs los procesos que generemos en la solución, por lo que podremos planificar y ejecutar de manera desatendida una vez hayamos desarrollado y probado cada proceso de datos.

3. Output (Salida): Es el destino de los datos procesados, al igual que en el bloque de entrada, mediante este bloque disponemos de diversas fuentes donde depositar o persistir la información, como escribir la información en una BBDD tanto local como remota, en un archivo en diferentes formatos, depositar dicha información en un servidor FTP, o bien enviar la información a través de correo electrónico, permitiendo de esta forma tener cierta flexibilidad a la hora de entregar los resultados a terceros.

Mediante este diseño u organización basada en tres bloques, se pretende conseguir la extracción, transformación, almacenamiento y distribución de datos de forma eficiente y controlada, pero sobre todo dinámica y modular, el cual era el concepto que se buscaba durante la idea del proyecto.

4.3.2. Solución SSIS IO

Pasamos a hablar de la solución desarrollada en SSIS y de su modularidad dividida en fases. El diseño está compuesto por cinco paquetes dtsx cada uno de ellos diseñado para contener una funcionalidad específica.

En la *Figura 4.3* se muestra el diagrama de flujo del proceso principal compuesto por el siguiente diseño.

- Se inicia la generación de log por cada fase a nivel de proceso maestro y de detalle.
- Se lanza una operación que en función de su valor realiza una determinada acción.
- Se ejecuta el núcleo del proceso principal, el cual lanza las diferentes etapas o subprocesos para el proceso en ejecución.
- Se genera un flujo en función del resultado de la acción para conocer si ha habido algún error durante la ejecución.
- Se finaliza la generación de log por cada fase a nivel de proceso maestro y de detalle.
- Se envía un correo resumen con el detalle de las operaciones realizadas.

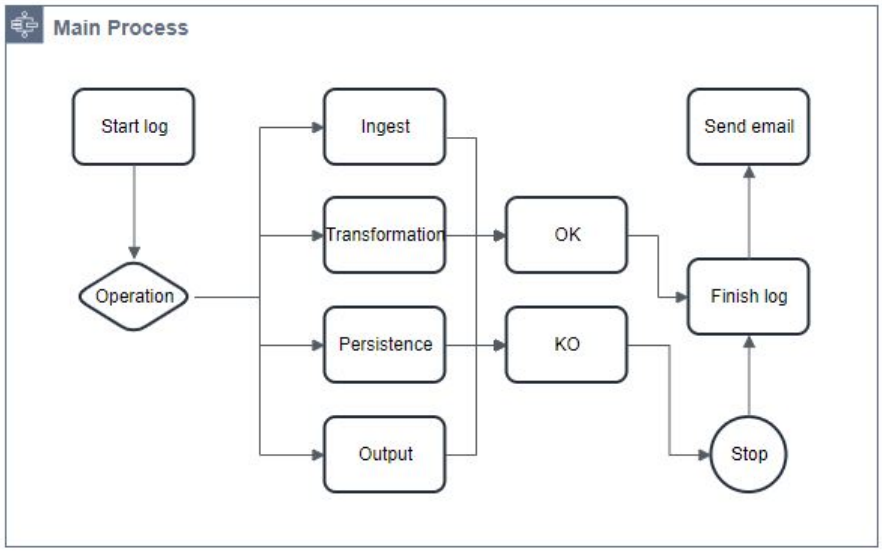


Figura 4.3: Diagrama de flujo del proceso principal

En la *Figura 4.4* se muestra el diagrama de flujo del proceso de ingesta compuesto por el siguiente diseño.

- Se obtienen los parámetros de configuración del proceso que vamos a ejecutar.
- Se inicia la generación de log por cada fase a nivel de proceso maestro y de detalle.
- Se lanza una operación que en función de su valor realiza una determinada acción.
- Se ejecuta el núcleo de la etapa de ingesta el cual puede realizar una ingesta sobre BBDD, leer un archivo de entrada, descargar un fichero del FTP o descomprimirlo.
- En caso de cargar información en BBDD se archiva en un objeto adicional.
- Se genera un flujo en función del resultado de la acción para conocer si ha habido algún error o advertencia durante la ejecución.
- Se finaliza la generación de log por cada fase a nivel de proceso maestro y de detalle.

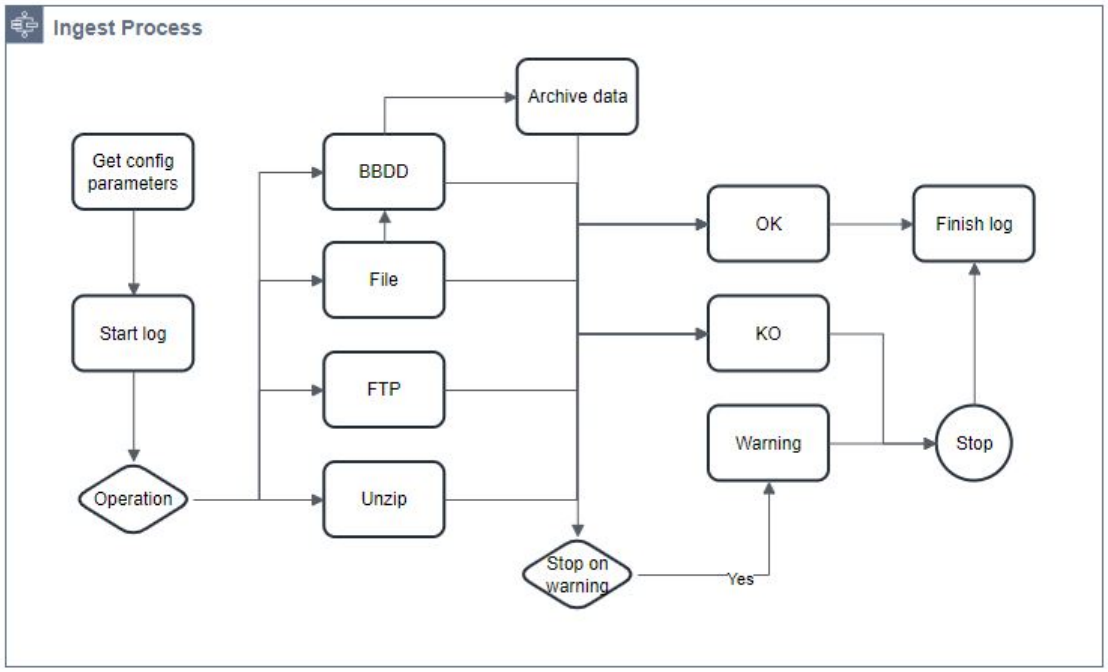


Figura 4.4: Diagrama de flujo del proceso de ingesta

En la *Figura 4.5* se muestra el diagrama de flujo del proceso de transformación compuesto por el siguiente diseño.

- Se obtienen los parámetros de configuración del proceso que vamos a ejecutar.
- Se inicia la generación de log por cada fase a nivel de proceso maestro y de detalle.
- Se lanza una operación que en función de su valor realiza una determinada acción.
- Se crea u obtiene la *delivery* del proceso en ejecución.
- Se ejecuta el núcleo de la etapa de transformación, donde se ejecutan tantos pasos a realizar como tenga el proceso en ejecución mediante un bucle foreach.
- Se genera un flujo en función del resultado de la acción para conocer si ha habido algún error durante la ejecución.
- Se finaliza la generación de log por cada fase a nivel de proceso maestro y de detalle.

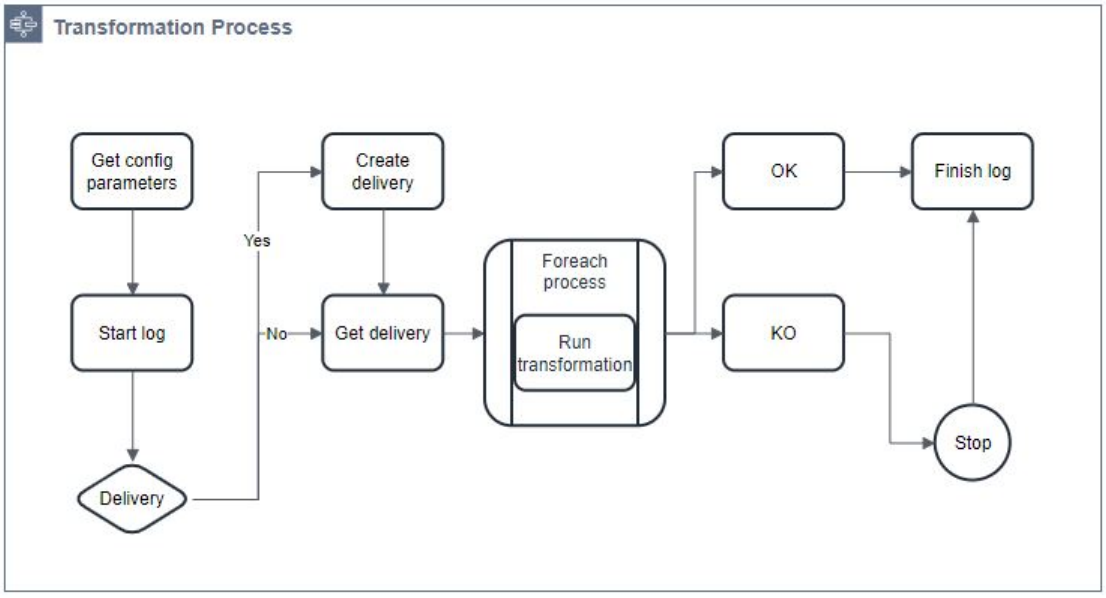


Figura 4.5: Diagrama de flujo del proceso de transformación

En la *Figura 4.6* se muestra el diagrama de flujo del proceso de persistencia compuesto por el siguiente diseño.

- Se obtienen los parámetros de configuración del proceso que vamos a ejecutar.
- Se inicia la generación de log por cada fase a nivel de proceso maestro y de detalle.
- Se ejecuta el núcleo de la etapa de persistencia, donde se ejecutan tantos pasos de escritura como tenga el proceso en ejecución mediante un bucle foreach.
- Se genera un flujo en función del resultado de la acción para conocer si ha habido algún error durante la ejecución.
- Se finaliza la generación de log por cada fase a nivel de proceso maestro y de detalle.

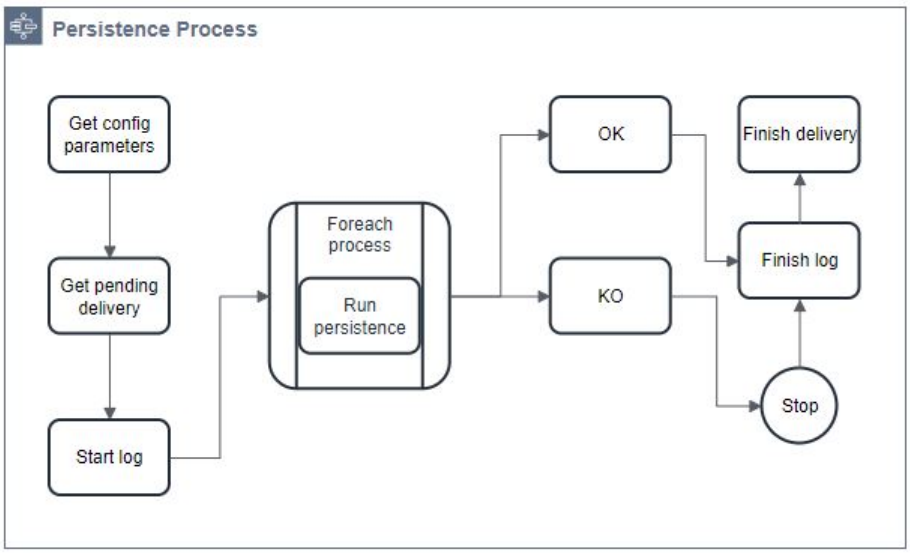


Figura 4.6: Diagrama de flujo del proceso de persistencia

En la *Figura 4.7* se muestra el diagrama de flujo del proceso de salida compuesto por el siguiente diseño.

- Se obtienen los parámetros de configuración del proceso que vamos a ejecutar.
- Se inicia la generación de log por cada fase a nivel de proceso maestro y de detalle.
- Se lanza una operación que en función de su valor realiza una determinada acción.
- Se ejecuta el núcleo de la etapa de salida el cual puede lanzar una acción sobre directorios, enviar un correo a un destinatario, generar un fichero de salida, subir un fichero al FTP o comprimir un fichero.
- Se genera un flujo en función del resultado de la acción para conocer si ha habido algún error o advertencia durante la ejecución.
- Se finaliza la generación de log por cada fase a nivel de proceso maestro y de detalle.

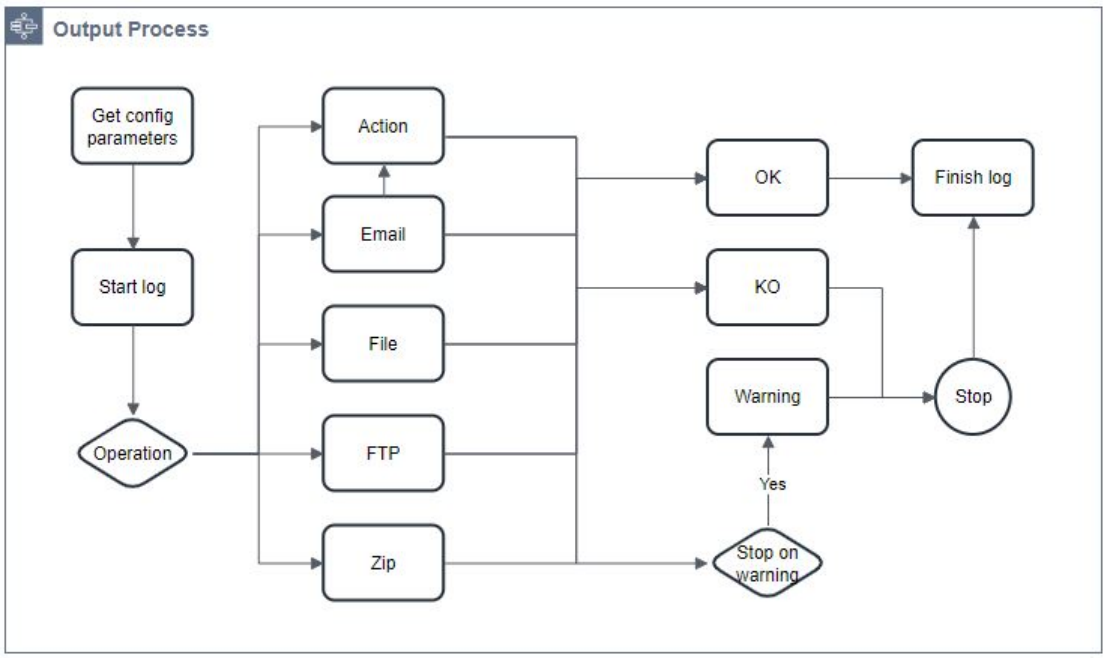


Figura 4.7: Diagrama de flujo del proceso de salida

Como se puede apreciar todos los diagramas siguen la misma tónica. Para una explicación exhaustiva de cada una de las fases ver su implementación en *Procesos SSIS IO* 5.3 del capítulo 5 de la documentación.

4.4. Diseño de BBDD y modelo entidad relación IO

Una de las piezas fundamentales del core de la solución es la BBDD IO, la cual ha sido creada con el propósito de embeber el código del proyecto, independientemente de su tipo, por lo que se encarga de almacenar la información de configuración de cada dtsx, toda la actividad de monitorización, las operaciones necesarias a realizar en las ejecuciones, el código de los procesos a ejecutar y los procedimientos almacenados o funciones, ya sean de procesos o de uso interno de la solución. La BBDD está creada en *Tercera Forma Normal* [10], salvo la tabla de detalle que veremos durante este apartado, la cual se decide mantener desnormalizada por una cuestión de practicidad.

La nomenclatura utilizada ha sido snake_case al igual que en la BBDD para los casos de uso *VehicleSales*. Adicionalmente las tablas maestras también contienen la palabra *type* en el nombre, y de esta forma poder diferenciarlas del resto de objetos.

4.4.1. Definición y creación de BBDD

El primer paso a dar consiste en la creación de la BBDD a la que hemos denominado IO. Se muestra a continuación el *Código 4.1* el cual además de dicha creación incluye un borrado de BBDD en caso de que fuese necesario.

Código 4.1: Creacion BBDD IO

```
1 -- ============================================================
2 -- ==================== CREACION BBDD =========================
3 -- ============================================================
```

```
-- Create IO Database
CREATE DATABASE IO;
GO

-- Use IO Database
USE IO;
GO

/*
USE Master;
ALTER DATABASE IO SET single_user with rollback immediate;
ALTER DATABASE IO SET MULTI_USER;
DROP DATABASE IO;
*/
```

En el *Código 4.2* se crean los siguientes schemas para poder agrupar los objetos por diferentes conceptos.

- auxiliar: Schema para almacenar los datos auxiliares que permitan pasos intermedios.
- config: Schema para almacenar los datos de configuración.
- hist: Schema para almacenar los datos de información histórica.
- input: Schema para almacenar los datos de la fase de ingesta.
- load: Schema para almacenar los datos de la fase de transformación.
- monitor: Schema para almacenar toda la actividad de log.
- output: Schema para almacenar los datos de la fase de output.

Código 4.2: Creacion schemas BBDD IO

```
-- ============================================================
-- ==================== CREACION SCHEMAS ======================
-- ============================================================

-- Create [auxiliar] schema
CREATE SCHEMA [auxiliar]
GO
-- Create [config] schema
CREATE SCHEMA [config]
GO
-- Create [hist] schema
CREATE SCHEMA [hist]
GO
-- Create [input] schema
CREATE SCHEMA [input]
GO
-- Create [load] schema
CREATE SCHEMA [load]
GO
-- Create [monitor] schema
CREATE SCHEMA [monitor]
GO
-- Create [output] schema
CREATE SCHEMA [output]
GO
```

4.4.2. Definición y creación de tablas

En esta sección, seguimos con la definición y creación de tablas de la BBDD. A continuación se detalla cada una de las tablas, sus columnas y *Código 4.3-4.12* de creación, el cual incluye claves primarias, foráneas y valores por defecto.

- config.customer_cluster_type: Tabla para almacenar la información de agrupaciones por clientes.
  - id (PK, int): Identificador único de la tabla.
  - customer_id (int): Identificador único del cliente.
  - name (varchar): Nombre del cliente.
  - cluster_id (int): Identificador único de la agrupación de clientes.
  - cluster_name (varchar): Identificador único de la agrupación de clientes.
  - data_creation_date (DF, datetime): Fecha de creación del registro generada por default.
  - data_modification_date (TR, datetime): Fecha de modificación del registro actualizada por trigger.
  - last_action_user (varchar): Usuario que realizó la última acción en la tabla generado por default.

Código 4.3: Creacion de config.customer_cluster_type

```
-- Create [config].[customer_cluster_type] Table
-- IO.config.customer_cluster_type: tabla que contendrá las agrupaciones por clientes
CREATE TABLE [config].[customer_cluster_type](
   [id] [int] IDENTITY(1,1) NOT NULL,
   [customer_id] [int] NOT NULL,
   [name] [varchar](64) NOT NULL,
   [cluster_id] [int] NULL,
   [cluster_name] [varchar](128) NULL,
   [data_creation_date] [datetime] NULL,
   [data_modification_date] [datetime] NULL,
   [last_action_user] [varchar](1024) NULL,
 CONSTRAINT [PK_customer_book_name_type] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
     ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

- config.customer_persistence_destination: Tabla para almacenar la BBDD de destino por cliente.
  - id (PK, int): Identificador único de la tabla.
  - customer_id (int): Identificador único del cliente.
  - destination_db (varchar): BBDD de destino sobre la que escribir la información.

- data_creation_date (DF, datetime): Fecha de creación del registro generada por default.
- data_modification_date (TR, datetime): Fecha de modificación del registro actualizada por trigger.
- last_action_user (DF, varchar): Usuario que realizó la última acción en la tabla generado por default.

Código 4.4: Creacion de config.customer_persistence_destination

```
-- Create [config].[customer_persistence_destination] Table
-- IO.config.customer_persistence_destination: tabla que contendrá la BBDD destino por cliente
CREATE TABLE [config].[customer_persistence_destination](
   [id] [int] IDENTITY(1,1) NOT NULL,
   [customer_id] [int] NOT NULL,
   [destination_db] [varchar](64) NOT NULL,
   [data_creation_date] [datetime] NULL,
   [data_modification_date] [datetime] NULL,
   [last_action_user] [varchar](1024) NULL,
 CONSTRAINT [PK_customer_persistence_destination_id] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
     ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY],
CONSTRAINT [UQ_customer__persistance_destination_cust_id] UNIQUE NONCLUSTERED
(
   [customer_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
     ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

- config.detail_process_type: Tabla para almacenar la configuración a nivel de detalle de cada uno de los pasos del proceso.
  - id (PK, int): Identificador único del paso concreto a ejecutar durante el proceso.
  - master_process_type_id (FK, int): Clave foránea que referencia la tabla master_process_type.
  - phase (varchar): Fase a la que pertenece el paso concreto a ejecutar (INGESTION, DELIVERY, TRANSFORMATION, PERSISTENCE y OUTPUT).
  - step (int): Indicador numérico del orden en el que se han de ejecutar los pasos de una fase.
  - entity (varchar): Nombre que se le ha dado a la entidad a tratar para ser identificada.
  - name (varchar): Nombre largo del proceso de detalle.
  - description (varchar): Descripción detallada del paso a ejecutar.
  - source_type (varchar): Tipo de origen utilizado en el paso, este campo se utiliza además para diferenciar los modos de ejecución y las herramientas disponibles. Para más información ver el capítulo *Implementación* 5.
  - zip_folder_path (varchar): Ruta de la carpeta que participa en el proceso de compresión/descompresión.

- zip_file_path (varchar): Archivo que participa en el proceso de compresión/descompresión.
- source_file_folder_path (varchar): Ruta de la carpeta donde se encontrarán los archivos de origen durante los procesos de ingesta.
- source_file_pattern (varchar): Patrón que deben cumplir los archivos que se tomarán como origen durante los procesos de ingesta.
- add_eof (int): Indicador para especificar la necesidad del caracter EOF (end of file) al final del archivo.
- skip_final_rows (int): Cantidad de filas finales a omitir en la ingesta de archivos.
- skip_initial_rows (int): Cantidad de filas iniciales a omitir en la ingesta de archivos.
- source_file_template_path (varchar): Ruta de la plantilla de definición del archivo a tratar en la etapa de ingesta.
- source_table_query (varchar): Columna para almacenar la consulta principal a ejecutar durante una fase determinada, Query que tiene que ser ejecutada durante las diferentes fases del proceso, puede contener tanto procedimientos almacenados, como consultas embebidas.
- destination_table (varchar): Tabla destino donde se depositarán los datos cargados durante la fase de ingesta.
- destination_format (varchar): Formato de destino para los ficheros de tipo output (CSV, DAT, TXT, XLS, XLSX).
- generate_empty_file (bit): Flag para indicar si se genera un archivo de tipo output en caso de que no contenga datos.
- destination_file_skip_rows (int): Cantidad de filas a omitir durante el conteo de filas obtenidas en la generación de un archivo de salida.
- destination_path (varchar): Ruta de la carpeta donde se depositará el archivo generado tras la ejecución de un proceso output.
- destination_file_template_path (varchar): Ruta de la plantilla de definición del archivo a tratar en la etapa de salida.
- ftp_server (varchar): Nombre del servidor FTP sobre el que realizar una conexión.
- ftp_port (int): Puerto FTP del servidor al que queremos conectarnos.
- ftp_user (varchar): Usuario de conexión del servidor FTP.
- ftp_pass (varbinary): Contraseña para el usuario de conexión del servidor FTP. Campo encriptado mediante la herramienta *Encriptación y desencriptación de contraseñas* 5.2.1.

- ftp_private_key_path (varchar): Ruta del archivo que contiene la clave o certificado público/privado en caso de conexión SFTP.
- ftp_ssh_host_key_fingerprint (varchar): Identificador de la huella del servidor para conexión de tipo SFTP.
- ftp_private_key_passphrase (varbinary): Contraseña para la clave privada de acceso al servidor SFTP. Campo encriptado mediante la herramienta *Encriptación y desencriptación de contraseñas* 5.2.1.
- ftp_operation (varchar): Operación a realizar tras conectarse a un servidor FTP (UPLOAD y DOWNLOAD).
- ftp_folder (varchar): Directorio del servidor FTP sobre el que se realizará la operación.
- ftp_file_pattern (varchar): Patrón que debe cumplir el nombre del archivo en el servidor FTP.
- ftp_local_folder (varchar): Ruta del directorio local sobre el que realizar la operación.
- ftp_local_file_pattern (varchar): Patrón que debe cumplir el nombre del archivo local durante la operación sobre el servidor FTP.
- email_from (varchar): Cuenta de correo desde la que se envía el email.
- email_to (varchar): Listado de destinatarios del email separados por coma o punto y coma.
- email_subject (varchar): Asunto del email a enviar.
- email_body (varchar): Cuerpo del correo del email a enviar.
- email_has_attachment (int): Flag para indicar si el email tiene ficheros adjuntos.
- email_attachment_path (varchar): Ruta donde se encuentran los archivos adjuntos a añadir al email.
- email_attachment_pattern (varchar): Patrón que debe cumplir los archivos adjuntos al email.
- data_creation_date (DF, datetime): Fecha de creación del registro generada por default.
- data_modification_date (TR, datetime): Fecha de modificación del registro actualizada por trigger.
- last_action_user (DF, varchar): Usuario que realizó la última acción en la tabla generado por default.
- file_action (varchar): Acción a ejecutar sobre archivos en la fase output (COPY, MOVE, DELETE y RENAME).

- file_action_source_pattern (varchar): Patrón que debe cumplir los archivos sobre los que se realiza la acción.
- check_ingestion_filename (int): Indicador para comprobar que el archivo no ha sido cargado con anterioridad.
- source_connection_server (varchar): Nombre o IP del servidor de BBDD de origen remoto.
- source_connection_database (varchar): Nombre de BBDD de origen remoto.
- source_connection_user (varchar): Usuario de conexión de BBDD de origen remoto.
- source_connection_password (varbinary): Contraseña del usuario de BBDD de origen remoto. Campo encriptado mediante la herramienta *Encriptación y desencriptación de contraseñas* 5.2.1.
- destination_connection_server (varchar): Nombre o IP del servidor de BBDD de destino remoto..
- destination_connection_database (varchar): Nombre de BBDD de destino remoto.
- destination_connection_user (varchar): Usuario de conexión de BBDD de destino remoto.
- destination_connection_password (varbinary): Contraseña del usuario de BBDD de destino remoto. Campo encriptado mediante la herramienta *Encriptación y desencriptación de contraseñas* 5.2.1.
- ingestion_encoding_file (varchar): Campo para editar la codificación del archivo en formato ASCII o UTF8.
- zip_rar_pass (varbinary): Contraseña en caso de que el archivo comprimido se encuentre protegido, o se quiera proteger tras la compresión. Campo encriptado mediante la herramienta *Encriptación y desencriptación de contraseñas* 5.2.1.

Código 4.5: Creacion de config.detail_process_type

```
1  -- Create [config].[detail_process_type] Table
2  -- IO.config.detail_process_type: tabla que contendrá la configuración a nivel de detalle de cada uno de los
       pasos del proceso.
3  CREATE TABLE [config].[detail_process_type](
4     [id] [int] NOT NULL,
5     [master_process_type_id] [int] NOT NULL,
6     [phase] [varchar](16) NOT NULL,
7     [step] [int] NOT NULL,
8     [entity] [varchar](512) NULL,
9     [name] [varchar](128) NULL,
10    [description] [varchar](1024) NULL,
11    [source_type] [varchar](5) NULL,
12    [zip_folder_path] [varchar](1024) NULL,
13    [zip_file_path] [varchar](1024) NULL,
14    [source_file_folder_path] [varchar](1024) NULL,
15    [source_file_pattern] [varchar](128) NULL,
16    [add_eof] [int] NULL,
17    [skip_final_rows] [int] NULL,
18    [skip_initial_rows] [int] NULL,
19    [source_file_template_path] [varchar](1024) NULL,
20    [source_table_query] [varchar](max) NULL,
```

```
21    [destination_table] [varchar](1024) NULL,
22    [destination_format] [varchar](1024) NULL,
23    [generate_empty_file] [bit] NULL,
24    [destination_file_skip_rows] [int] NULL,
25    [destination_path] [varchar](1024) NULL,
26    [destination_file_template_path] [varchar](1024) NULL,
27    [ftp_server] [varchar](128) NULL,
28    [ftp_port] [int] NULL,
29    [ftp_user] [varchar](128) NULL,
30    [ftp_pass] [varbinary](256) NULL,
31    [ftp_private_key_path] [varchar](1024) NULL,
32    [ftp_ssh_host_key_fingerprint] [varchar](128) NULL,
33    [ftp_private_key_passphrase] [varbinary](256) NULL,
34    [ftp_operation] [varchar](8) NULL,
35    [ftp_folder] [varchar](1024) NULL,
36    [ftp_file_pattern] [varchar](1024) NULL,
37    [ftp_local_folder] [varchar](1024) NULL,
38    [ftp_local_file_pattern] [varchar](1024) NULL,
39    [email_from] [varchar](256) NULL,
40    [email_to] [varchar](1024) NULL,
41    [email_subject] [varchar](1024) NULL,
42    [email_body] [varchar](max) NULL,
43    [email_has_attachment] [int] NULL,
44    [email_attachment_path] [varchar](1024) NULL,
45    [email_attachment_pattern] [varchar](1024) NULL,
46    [data_creation_date] [datetime] NULL,
47    [data_modification_date] [datetime] NULL,
48    [last_action_user] [varchar](1024) NULL,
49    [file_action] [varchar](6) NULL,
50    [file_action_source_pattern] [varchar](1024) NULL,
51    [file_action_destination_folder] [varchar](1024) NULL,
52    [check_ingestion_filename] [int] NULL,
53    [source_connection_server] [varchar](64) NULL,
54    [source_connection_database] [varchar](64) NULL,
55    [source_connection_user] [varchar](64) NULL,
56    [source_connection_password] [varbinary](256) NULL,
57    [destination_connection_server] [varchar](64) NULL,
58    [destination_connection_database] [varchar](64) NULL,
59    [destination_connection_user] [varchar](64) NULL,
60    [destination_connection_password] [varbinary](256) NULL,
61    [ingestion_encoding_file] [varchar](5) NULL,
62    [zip_rar_pass] [varbinary](256) NULL,
63 PRIMARY KEY CLUSTERED
64 (
65    [id] ASC
66 )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
       ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
67 ) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
68 GO
69
70 ALTER TABLE [config].[detail_process_type] WITH CHECK ADD CONSTRAINT
       [CK_detail_process_type_check_ingestion_filename_logical] CHECK (([phase]='INGESTION' AND
       [source_type]='FILE' AND ([check_ingestion_filename]=(0) OR [check_ingestion_filename]=(1)) OR
       [phase]='INGESTION' AND [source_type]<>'FILE' AND [check_ingestion_filename] IS NULL OR [phase]<>'INGESTION'
       AND [check_ingestion_filename] IS NULL))
71 GO
72 ALTER TABLE [config].[detail_process_type] CHECK CONSTRAINT
       [CK_detail_process_type_check_ingestion_filename_logical]
73 GO
74 ALTER TABLE [config].[detail_process_type] WITH CHECK ADD CONSTRAINT
       [CK_detail_process_type_check_ingestion_filename_values] CHECK (([check_ingestion_filename]=NULL OR
       [check_ingestion_filename]=(0) OR [check_ingestion_filename]=(1)))
75 GO
76 ALTER TABLE [config].[detail_process_type] CHECK CONSTRAINT
       [CK_detail_process_type_check_ingestion_filename_values]
77 GO
78 ALTER TABLE [config].[detail_process_type] WITH CHECK ADD CONSTRAINT
       [CK_detail_process_type_destination_connection] CHECK (([destination_connection_server] IS NOT NULL AND
       [destination_connection_database] IS NOT NULL AND [destination_connection_user] IS NOT NULL AND
       [destination_connection_password] IS NOT NULL OR [destination_connection_server] IS NULL AND
       [destination_connection_database] IS NULL AND [destination_connection_user] IS NULL AND
       [destination_connection_password] IS NULL))
79 GO
```

```
ALTER TABLE [config].[detail_process_type] CHECK CONSTRAINT [CK_detail_process_type_destination_connection]
GO
ALTER TABLE [config].[detail_process_type] WITH CHECK ADD CONSTRAINT [CK_detail_process_type_source_connection]
    CHECK (([source_connection_server] IS NOT NULL AND [source_connection_database] IS NOT NULL AND
    [source_connection_user] IS NOT NULL AND [source_connection_password] IS NOT NULL OR
    [source_connection_server] IS NULL AND [source_connection_database] IS NULL AND [source_connection_user] IS
    NULL AND [source_connection_password] IS NULL))
GO
ALTER TABLE [config].[detail_process_type] CHECK CONSTRAINT [CK_detail_process_type_source_connection]
GO
```

- config.master_process_type: Tabla para almacenar la configuración general del proceso especificando las fases de las que consta.
  - id (PK, int): Identificador único del proceso.
  - customer (varchar): Nombre del cliente.
  - name (varchar): Nombre asignado al proceso.
  - description (varchar): Descripción detallada del proceso.
  - active_flag (bit): Indica si el proceso se encuentra activo.
  - ingestion_flag (bit): Indica si el proceso dispone de fase de ingesta.
  - transformation_flag (bit): Indica si el proceso dispone de fase de transformación.
  - persist_flag (bit): Indica si el proceso dispone de fase de persistencia.
  - output_flag (bit): Indica si el proceso dispone de fase de salida.
  - data_creation_date (DF, datetime): Fecha de creación del registro generada por default.
  - data_modification_date (TR, datetime): Fecha de modificación del registro actualizada por trigger.
  - last_action_user (DF, varchar): Usuario que realizó la última acción en la tabla generado por default.
  - stop_on_warning (bit): Indicador que especifica si se continua con la ejecución de un proceso en caso de warning durante esta.

Código 4.6: Creacion de config.master_process_type

```
-- Create [config].[master_process_type] Table
-- IO.config.master_process_type: tabla que contendrá la configuración a nivel de maestro de proceso completo.
CREATE TABLE [config].[master_process_type](
   [id] [int] NOT NULL,
   [customer] [varchar](128) NULL,
   [name] [varchar](128) NULL,
   [description] [varchar](1024) NULL,
   [active_flag] [bit] NOT NULL,
   [ingestion_flag] [bit] NULL,
   [transformation_flag] [bit] NULL,
   [persist_flag] [bit] NULL,
   [output_flag] [bit] NULL,
   [data_creation_date] [datetime] NULL,
```

```
   [data_modification_date] [datetime] NULL,
   [last_action_user] [varchar](1024) NULL,
   [stop_on_warning] [bit] NULL,
 CONSTRAINT [PK_ingest_master_process_type_id] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
    ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [config].[master_process_type] ADD DEFAULT ((1)) FOR [active_flag]
GO
ALTER TABLE [config].[master_process_type] ADD DEFAULT (getdate()) FOR [data_creation_date]
GO
ALTER TABLE [config].[master_process_type] ADD DEFAULT (original_login()) FOR [last_action_user]
GO
ALTER TABLE [config].[master_process_type] ADD DEFAULT ((1)) FOR [stop_on_warning]
GO
```

- load.delivery: Tabla para almacenar las deliveries por ejecución y cliente.
  - id (PK, int): Identificador único de la tabla.
  - customer_id (int): Identificador único del cliente.
  - master_process_log_id (FK, int): Clave foránea que referencia la tabla master_process_log.
  - registration_date (datetime): Fecha de registro de la fila.

Código 4.7: Creacion de load.delivery

```
-- Create [load].[delivery] Table
-- IO.load.delivery: tabla que contendrá cada una de las deliveries por ejecución y cliente.
CREATE TABLE [load].[delivery](
   [id] [int] IDENTITY(1,1) NOT NULL,
   [customer_id] [int] NOT NULL,
   [master_process_log_id] [int] NOT NULL,
   [registration_date] [datetime] NULL,
 CONSTRAINT [PK_delivery] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
    ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [load].[delivery] WITH CHECK ADD CONSTRAINT [FK_delivery_master_process_log_id] FOREIGN
    KEY([master_process_log_id])
REFERENCES [monitor].[master_process_log] ([id])
GO
ALTER TABLE [load].[delivery] CHECK CONSTRAINT [FK_delivery_master_process_log_id]
GO
```

- load.operation: Tabla para almacenar las operaciones calculadas durante el proceso de transformación.
  - id (PK, int): Identificador único de la tabla.
  - delivery_id (FK, int): Clave foránea que referencia la tabla delivery.
  - load_table_name (varchar): Nombre de la tabla sobre la que se produce la operación.
  - load_reference_id (int): Identificador de referencia del registro de la tabla sobre la

que se produce la operación.

- master_reference_id (int): Identificador de referencia del registro de la tabla destino sobre la que se ha persistido la operación.
- operation_type_id (FK, int): Clave foránea que referencia la tabla operation_type.
- operation_finished (DF, bit): Indicador de marca tras finalizar la operación.

Código 4.8: Creacion de load.operation

```
-- Create [load].[operation] Table
-- IO.load.operation: tabla que contendrá cada una de las operaciones calculadas durante el proceso de transformación.
CREATE TABLE [load].[operation](
   [id] [int] IDENTITY(1,1) NOT NULL,
   [delivery_id] [int] NULL,
   [load_table_name] [varchar](255) NULL,
   [load_reference_id] [int] NULL,
   [master_reference_id] [int] NULL,
   [operation_type_id] [int] NULL,
   [operation_finished] [bit] NULL,
PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [load].[operation] ADD CONSTRAINT [DF_operation_operation_finished] DEFAULT ((0)) FOR [operation_finished]
GO

ALTER TABLE [load].[operation] WITH CHECK ADD CONSTRAINT [FK_operation_operation] FOREIGN KEY([operation_type_id])
REFERENCES [load].[operation_type] ([id])
GO
ALTER TABLE [load].[operation] CHECK CONSTRAINT [FK_operation_operation]
GO
```

- load.operation_type: Tabla para almacenar los tipos de operación permitidos durante el proceso de transformación.
  - id (PK, int): Identificador único de la tabla.
  - operation_type_name (varchar): Nombre de la operación a realizar (INSERT, UPDATE, DELETE).

Código 4.9: Creacion de load.operation_type

```
-- Create [load].[operation_type] Table
-- IO.load.operation_type: tabla que contendrá los tipos de operaciones permitidos durante la fase de transformación.
CREATE TABLE [load].[operation_type](
   [id] [int] NOT NULL,
   [operation_type_name] [varchar](10) NOT NULL,
PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

- monitor.detail_process_log: Tabla para almacenar el log con el detalle de cada paso del proceso.
  - id (PK, int): Identificador único de la tabla.
  - master_process_log_id (int): Clave foránea que referencia la tabla master_process_log.
  - phase (varchar): Almacena las fases del proceso las cuales solo pueden ser INGESTION, TRANSFORMATION, DELIVERY, PERSISTENCE y OUTPUT.
  - entity (varchar): Nombre que se le ha dado a la entidad a tratar para ser identificada.
  - start_time (DF, datetime): Fecha de inicio del registro que contiene el log.
  - end_time (datetime): Fecha fin del registro que contiene el log.
  - quantity_rows (int): Cantidad de filas afectadas para el paso específico del proceso.
  - status (varchar): Almacena el estado del proceso el cual solo puede ser SUCEEDED, FAILED, WARNING.
  - error_description (varchar): Almacena la descripción con el detalle del paso ejecutado.

Código 4.10: Creacion de monitor.detail_process_log

```
-- Create [monitor].[detail_process_log] Table
-- IO.monitor.detail_process_log: tabla de log de la ejecución a nivel de detalle de cada paso del proceso.
CREATE TABLE [monitor].[detail_process_log](
   [id] [int] IDENTITY(1,1) NOT NULL,
   [master_process_log_id] [int] NULL,
   [phase] [varchar](16) NULL,
   [entity] [varchar](128) NULL,
   [start_time] [datetime] NULL,
   [end_time] [datetime] NULL,
   [quantity_rows] [int] NULL,
   [status] [varchar](9) NULL,
   [error_description] [varchar](1024) NULL,
 CONSTRAINT [PK_detail_process_log] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
    ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [monitor].[detail_process_log] ADD DEFAULT (getdate()) FOR [start_time]
GO

ALTER TABLE [monitor].[detail_process_log] WITH CHECK ADD CONSTRAINT
    [FK_detail_process_log_master_process_log_id] FOREIGN KEY([master_process_log_id]) REFERENCES
    [monitor].[master_process_log] ([id])
GO
ALTER TABLE [monitor].[detail_process_log] CHECK CONSTRAINT [FK_detail_process_log_master_process_log_id]
GO
ALTER TABLE [monitor].[detail_process_log] WITH CHECK ADD CONSTRAINT [CK_detail_process_log_phase] CHECK
    (([phase]='OUTPUT' OR [phase]='PERSISTENCE' OR [phase]='TRANSFORMATION' OR [phase]='DELIVERY' OR
    [phase]='INGESTION'))
GO
ALTER TABLE [monitor].[detail_process_log] CHECK CONSTRAINT [CK_detail_process_log_phase]
GO
ALTER TABLE [monitor].[detail_process_log] WITH CHECK ADD CONSTRAINT [CK_detail_process_log_status] CHECK
    (([status]='WARNING' OR [status]='FAILED' OR [status]='SUCCEEDED'))
GO
ALTER TABLE [monitor].[detail_process_log] CHECK CONSTRAINT [CK_detail_process_log_status]
GO
```

- monitor.log: Tabla para almacenar el log de procesos internos de la solución.
  - id (PK, int): Identificador único de la tabla.
  - created_at (datetime): Fecha de creación del registro que contiene el log interno.
  - user (varchar): Usuario de BBDD que almacena el registro.
  - source (varchar): Objeto que realiza la acción y para el que se almacena la operación.
  - pid (varchar): Identificador interno del proceso en BBDD.
  - trace_id (varchar): Identificador interno de seguimiento del proceso en BBDD.
  - message (text): Mensaje de log a registrar que contiene el detalle del paso del objeto.
  - severity (varchar): Severidad de la información almacenada.

Código 4.11: Creacion de monitor.log

```
-- Create [monitor].[log] Table
-- IO.monitor.log: tabla de log detallado de procesos internos de IO.
CREATE TABLE [monitor].[log](
   [id] [int] IDENTITY(1,1) NOT NULL,
   [created_at] [datetime] NULL,
   [user] [varchar](128) NULL,
   [source] [varchar](128) NULL,
   [pid] [smallint] NULL,
   [trace_id] [varchar](64) NULL,
   [message] [text] NULL,
   [severity] [varchar](64) NULL,
 CONSTRAINT [PK_log] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
     ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
```

- monitor.master_process_log: Tabla para almacenar el log general de las fases de un proceso.
  - id (PK, int): Identificador único de la tabla.
  - master_process_type_id (FK, int): Clave foránea que referencia la tabla master_process_type.
  - master_process_start_time (DF, datetime): Fecha de inicio del registro que contiene el log general.
  - master_process_end_time (datetime): Fecha fin del registro que contiene el log general.
  - master_process_status (varchar): Almacena el estado del proceso maestro el cual solo puede ser SUCEEDED, FAILED, WARNING.
  - ingestion_start_time (DF, datetime): Fecha de inicio de la fase de ingesta para el proceso en ejecución.
  - ingestion_end_time (datetime): Fecha fin de la fase de ingesta para el proceso en ejecución.

- ingestion_status (DF, varchar): Almacena el estado de la fase de ingesta el cual solo puede ser SUCEEDED, FAILED, WARNING.
- transformation_start_time (DF, datetime): Fecha de inicio de la fase de transformación para el proceso en ejecución.
- transformation_end_time (datetime): Fecha fin de la fase de transformación para el proceso en ejecución.
- transformation_status (DF, varchar): Almacena el estado de la fase de transformación el cual solo puede ser SUCEEDED, FAILED, WARNING.
- persistence_start_time (DF, datetime): Fecha de inicio de la fase de persistencia para el proceso en ejecución.
- persistence_end_time (datetime): Fecha fin de la fase de persistencia para el proceso en ejecución.
- persistence_status (DF, varchar): Almacena el estado de la fase de persistencia el cual solo puede ser SUCEEDED, FAILED, WARNING.
- output_start_time (DF, datetime): Fecha de inicio de la fase de salida para el proceso en ejecución.
- output_end_time (datetime): Fecha fin de la fase de salida para el proceso en ejecución.
- output_status (DF, varchar): Almacena el estado de la fase de salida el cual solo puede ser SUCEEDED, FAILED, WARNING.
- error_description (varchar): Almacena la información del error en caso de que se produzca.

Código 4.12: Creacion de monitor.master_process_log

```
-- Create [monitor].[master_process_log] Table
-- IO.monitor.master_process_log: tabla de log general de la ejecución a nivel de proceso.
CREATE TABLE [monitor].[master_process_log](
   [id] [int] IDENTITY(1,1) NOT NULL,
   [master_process_type_id] [int] NULL,
   [master_process_start_time] [datetime] NULL,
   [master_process_end_time] [datetime] NULL,
   [master_process_status] [varchar](9) NULL,
   [ingestion_start_time] [datetime] NULL,
   [ingestion_end_time] [datetime] NULL,
   [ingestion_status] [varchar](9) NULL,
   [transformation_start_time] [datetime] NULL,
   [transformation_end_time] [datetime] NULL,
   [transformation_status] [varchar](9) NULL,
   [persistence_start_time] [datetime] NULL,
   [persistence_end_time] [datetime] NULL,
   [persistence_status] [varchar](9) NULL,
   [output_start_time] [datetime] NULL,
   [output_end_time] [datetime] NULL,
   [output_status] [varchar](9) NULL,
   [error_description] [varchar](1024) NULL,
 CONSTRAINT [PK_master_process_log] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
```

```
    ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
```

4.4.3. Generación de datos de prueba

Por último, se puebla la BBDD con los datos necesarios para su correcto funcionamiento. El resto de información se irá generando de dos formas diferentes, por un lado el desarrollo de los procesos irá insertando información en los schemas *auxiliar, config, hist, input, load e output*, mientras que por otro lado las ejecuciones almacenarán información en el schema *monitor*.

Se aportan previamente los scripts, por un lado el *Código 4.13* contiene un borrado ordenado de información en caso de ser necesario realizar la limpieza de los objetos, y por otro el *Código 4.14* una consulta básica de cada tabla.

Código 4.13: Borrado de información en IO

```
/*
TRUNCATE TABLE [IO].config.master_process_type
TRUNCATE TABLE [IO].config.detail_process_type
TRUNCATE TABLE [IO].config.customer_cluster_type
TRUNCATE TABLE [IO].config.customer_persistence_destination
TRUNCATE TABLE [IO].load.operation_type
TRUNCATE TABLE [IO].load.operation
DELETE FROM [IO].load.delivery DBCC CHECKIDENT ('IO.load.delivery',RESEED, 0)
TRUNCATE TABLE [IO].monitor.log
TRUNCATE TABLE [IO].monitor.detail_process_log
DELETE FROM [IO].monitor.master_process_log DBCC CHECKIDENT ('IO.monitor.master_process_log',RESEED, 0)
*/
```

Código 4.14: Consulta de información en IO

```
SELECT * FROM [IO].config.master_process_type
SELECT * FROM [IO].config.detail_process_type
SELECT * FROM [IO].config.customer_cluster_type
SELECT * FROM [IO].config.customer_persistence_destination
SELECT * FROM [IO].load.operation_type
SELECT * FROM [IO].load.operation
SELECT * FROM [IO].load.delivery
SELECT * FROM [IO].monitor.[log]
SELECT * FROM [IO].monitor.detail_process_log
SELECT * FROM [IO].monitor.master_process_log
```

A continuación se muestra el detalle de la información generada. Es importante destacar que estas operaciones se deberían ejecutar mediante las cláusulas *BEGIN TRAN, COMMIT o ROLLBACK* [11] como aparece en el *Código 4.15-4.18*.

Código 4.15: Generación de información para la tabla config.master_process_type

```
-- [IO].config.master_process_type
-- Inserta la configuración a nivel maestro para anular otros procesos
INSERT INTO [IO].[config].[master_process_type]
   ([id],[customer],[name],[description],[active_flag],[ingestion_flag],[transformation_flag],[persist_flag],[output_flag])
VALUES
    (0, NULL, 'Deshabilitar / No usar', 'Master process usado para anular pasos de detail_process_type en caso de
        que sea necesario', 0, 0, 0, 0, 0)
```

Código 4.16: Generación de información para la tabla config.customer_cluster_type

```
-- [IO].config.customer_cluster_type
-- Inserta las agrupaciones por clientes (subdivisiones)
INSERT INTO [IO].config.customer_cluster_type (customer_id, name, cluster_id, cluster_name) VALUES (1, 'BMW', 1,
    'BMW_Group')
```

```
4 INSERT INTO [IO].config.customer_cluster_type (customer_id, name, cluster_id, cluster_name) VALUES (2, 'Mini',
      1, 'BMW_Group')
```

Código 4.17: Generación de información para la tabla config.customer_persistence_destination

```
1 -- [IO].config.customer_persistence_destination
2 -- Insertar los destinos segun el cliente
3 INSERT INTO [IO].config.customer_persistence_destination (customer_id, destination_db) VALUES (1, 'VehicleSales')
4 INSERT INTO [IO].config.customer_persistence_destination (customer_id, destination_db) VALUES (2, 'VehicleSales')
```

Código 4.18: Generación de información para la tabla load.operation_type

```
1 -- [load].operation_type
2 -- Insertar tipos de operaciones
3 INSERT INTO [IO].[load].[operation_type] (id, operation_type_name) VALUES (1, 'Insert')
4 INSERT INTO [IO].[load].[operation_type] (id, operation_type_name) VALUES (2, 'Update')
5 INSERT INTO [IO].[load].[operation_type] (id, operation_type_name) VALUES (3, 'Close')
6 INSERT INTO [IO].[load].[operation_type] (id, operation_type_name) VALUES (4, 'Delete')
7 INSERT INTO [IO].[load].[operation_type] (id, operation_type_name) VALUES (5, 'Read')
```

4.4.4. Modelo entidad-relación

Tras la ejecución de los scripts mostrados durante los apartados anteriores, ya dispondremos de la BBDD *IO* completamente preparada para su uso. Adicionalmente, se ha generado el diagrama entidad-relación en SQL Server, el cual queda como se muestra en la *Figura 4.8*.

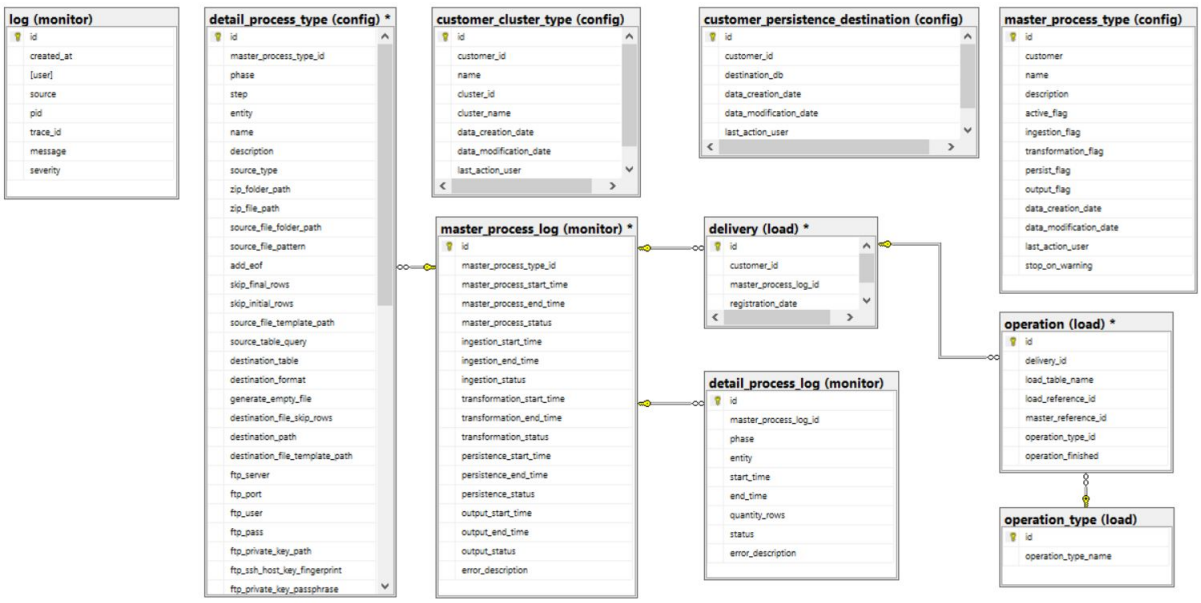


Figura 4.8: Diagrama entidad-relación IO

4.5. Conclusiones

En este capítulo se ha analizado el diseño de la estructura principal de la solución IO, visualizando los principales diagramas de arquitectura, así como la definición de la BBDD principal del core de la solución resultado del proceso de diseño del sistema. Este capítulo nos permite tomarlo como punto de partida tanto para la implementación como para las pruebas a realizar para testear el proyecto.

Para más información sobre el diseño de la solución con la herramienta SSIS ver el capítulo *Implementación* 5 de la documentación.

Capítulo 5

Implementación

5.1. Introducción

En este capítulo se analiza el resultado de las tareas de implementación desde el punto de vista de la funcionalidad obtenida, así como las metodologías y herramientas utilizadas. Podemos describir la implementación como la transformación de la lógica definida durante el diseño en código ejecutable.

El conjunto de procesos implementado es un reflejo de la arquitectura definida en el modelo de diseño. Para cada módulo de dicha arquitectura se ha creado un proceso equivalente, de forma que el resultado final ofrezca una funcionalidad similar al diseño definido durante el capítulo 4.

5.2. Herramientas de soporte

A continuación se va a describir las herramientas de soporte que han sido desarrolladas para dotar a la solución de todas las funcionalidades especificadas durante el análisis.

5.2.1. Encriptación y desencriptación de contraseñas

Como utilidad adicional de seguridad, se ha incorporado la encriptación y desencriptación de contraseñas, para dotar de un plus de robustez a la solución que implementa la herramienta IO. Para ello se han realizado las siguientes modificaciones a nivel de BBDD y del proyecto SSIS que implementa la solución.

5.2.1.1. Modificaciones sobre BBDD

Lo primero a realizar es la creación de una master key para poder encriptar la BBDD de IO mediante el *Código 5.1*.

Código 5.1: 1.2 Encriptado BBDD IO.sql

```
USE IO
IF @@SERVERNAME = 'WIN-SER-AMAROTO'
BEGIN
   CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'jEmY6RjEuaSSsmZf3yAp'
END
--DROP MASTER KEY
```

Posteriormente el *Código 5.2* genera un nuevo certificado que será utilizado durante el proceso de encriptación.

Código 5.2: 1.2 Encriptado BBDD IO.sql

```
CREATE CERTIFICATE io_db_certificate WITH SUBJECT = 'Certificado para el encriptado a nivel de base de datos
    para IO'
```

```
GO
--DROP CERTIFICATE io_db_certificate
```

A continuación el *Código 5.3* crea una clave simétrica basada en el certificado anterior, la encriptación de cada uno de los passwords almacenados en las tablas de configuración de IO se realizará en base a esta clave.

Código 5.3: 1.2 Encriptado BBDD IO.sql

```
CREATE SYMMETRIC KEY password_io_simetric_key WITH
IDENTITY_VALUE = 'password_io_simetric_key',
ALGORITHM = AES_256,
KEY_SOURCE = 'Usa la contraseña de encriptado de master key'
ENCRYPTION BY CERTIFICATE io_db_certificate;
GO

-- DROP SYMMETRIC KEY password_io_simetric_key
```

Es muy importante saber que estos certificados caducan de manera anual y es necesario renovarlos o volver a generarlos, en caso contrario, esto puede afectar a la encriptación/desencriptación que se utiliza dando lugar a errores e incluso haciendo que los valores encriptados no puedan ser recuperados. Como contingencia, es recomendable almacenar las contraseñas por fuera de SQL Server.

Tras realizar estas acciones, la BBDD ya se encuentra en disposición de utilizar la funcionalidad de encriptación/desencriptación para los campos que contengan información sensible, adicionalmente se han creado dos funciones (*Código 5.5-5.6*) y un procedimiento almacenado (*Código 5.4*) que facilita el uso de su aplicación.

Código 5.4: config.sp_open_io_password_encryption_keys.sql

```
USE [IO]
GO
/*
-- ============================================================================================
Autor: Adrian Maroto
Fecha creación: 14/03/2024
Descripción: Procedimiento almacenado para abrir las claves de encriptación de contraseñas en la base de datos
     IO.

Entrada:

Salida:

Ejemplo:
   -- Apertura de las claves de cifrado, siempre se ha de hacer antes de encriptar o desencriptar
   EXEC [IO].config.sp_open_io_password_encryption_keys
   -- Funcion de encriptación, entrada un VARCHAR, salida un VARBINBARY
   SELECT [IO].config.f_password_encryption('P4$$W0rd_unico')
   -- Función de desencriptación, entra un VARBINARY, salida un VARCHAR
   SELECT
        [IO].config.f_password_decryption(0x00BA1DC5F52ACF05CEF7C5F899A696B302000000B16EB65E085C7E2EC39AC7B8C51300496D53D8094598

-- ============================================================================================
*/
CREATE OR ALTER PROCEDURE [config].[sp_open_io_password_encryption_keys]
AS
BEGIN

   -- Activar la opción para no contar las filas afectadas por las instrucciones SELECT
   SET NOCOUNT ON;

   -- Intenta abrir la clave simétrica para desencriptar contraseñas
   BEGIN TRY
      OPEN SYMMETRIC KEY password_io_simetric_key
```

```
      DECRYPTION BY CERTIFICATE io_db_certificate
   END TRY
   BEGIN CATCH
      -- En caso de error, mostrar un mensaje de error
      SELECT 'Some error getting encryption keys: ' + ERROR_MESSAGE()
   END CATCH

END
```

Código 5.5: f_password_decryption.sql

```
/*
-- ================================================================================================
Autor: Adrian Maroto
Fecha creacion: 13/03/2024
Descripcion: Desencripta una cadena encriptada. Necesario abrir primero las claves de cifrado.
Cambios:

Entrada:
   @value_to_decrypt: Cadena a desencriptar.

Salida:
   RETURN: Cadena desencriptada.

Ejemplo:
   -- Apertura de las claves de cifrado, siempre se ha de hacer antes de encriptar o desencriptar
   EXEC [config].[sp_open_io_password_encryption_keys]
   -- Funcion de encriptación, entrada un VARCHAR, salida un VARBINBARY
   SELECT [config].[f_password_encryption]('P4$$WOrd_unico')
   -- Función de desencriptación, entra un VARBINARY, salida un VARCHAR
   SELECT
        [config].[f_password_decryption](0x00BA1DC5F52ACF05CEF7C5F899A696B302000000B16EB65E085C7E2EC39AC7B8C51300496D53D80945989
-- ================================================================================================
*/
CREATE OR ALTER FUNCTION [config].[f_password_decryption] (@value_to_decrypt VARBINARY(256))
RETURNS VARCHAR(MAX)
AS
BEGIN

    DECLARE @result VARCHAR(MAX)

    SET @Result = DECRYPTBYKEY(@value_to_decrypt)

    -- Devuelve el resultado de la función
    RETURN @result

END
```

Código 5.6: f_password_encryption.sql

```
/*
-- ================================================================================================
Autor: Adrian Maroto
Fecha creacion: 13/03/2024
Descripcion: Encripta una cadena desencriptada. Necesario abrir primero las claves de cifrado.
Cambios:

Entrada:
   @value_to_encrypt: Cadena a encriptar.

Salida:
   RETURN: Cadena encriptada.

Ejemplo:
   -- Apertura de las claves de cifrado, siempre se ha de hacer antes de encriptar o desencriptar
   EXEC [config].[sp_open_io_password_encryption_keys]
   -- Funcion de encriptación, entrada un VARCHAR, salida un VARBINBARY
   SELECT [config].[f_password_encryption]('P4$$WOrd_unico')
   -- Función de desencriptación, entra un VARBINARY, salida un VARCHAR
   SELECT
        [config].[f_password_decryption](0x00BA1DC5F52ACF05CEF7C5F899A696B302000000B16EB65E085C7E2EC39AC7B8C51300496D53D80945989
-- ================================================================================================
```

```
*/
CREATE OR ALTER FUNCTION [config].[f_password_encryption] (@value_to_encrypt VARCHAR(MAX))
RETURNS VARBINARY(256)
AS
BEGIN

    DECLARE @Result VARBINARY(256)

    SET @Result = ENCRYPTBYKEY(KEY_GUID('password_io_simetric_key'), @value_to_encrypt)

   -- Devuelve el resultado de la función
    RETURN @Result
END
```

5.2.1.2. Modificaciones sobre IO en SSIS

Para que el proceso de encriptación y desencriptación funcione a través de SSIS, se han incluido las funciones previamente comentadas dentro de los paquetes dtsx de la solución, permitiendo realizar la desencriptación en tiempo de ejecución, no encontrándose disponibles los valores mediante consultoría normal, dotando a la solución de una seguridad ante el dato adicional.

Por lo tanto, se ha añadido en cada una de las tareas que conllevan una lectura de contraseñas, la ejecución previa del procedimiento de apertura de claves *IO.config.sp_open_io_password_encryption_keys*, además de la función de desencriptado *IO.config.f_password_decryption* con la columna *destination_connection_password* que contiene el valor encriptado como parámetro de entrada. En la *Figura 5.1* se muestra una de las expresiones embebidas.

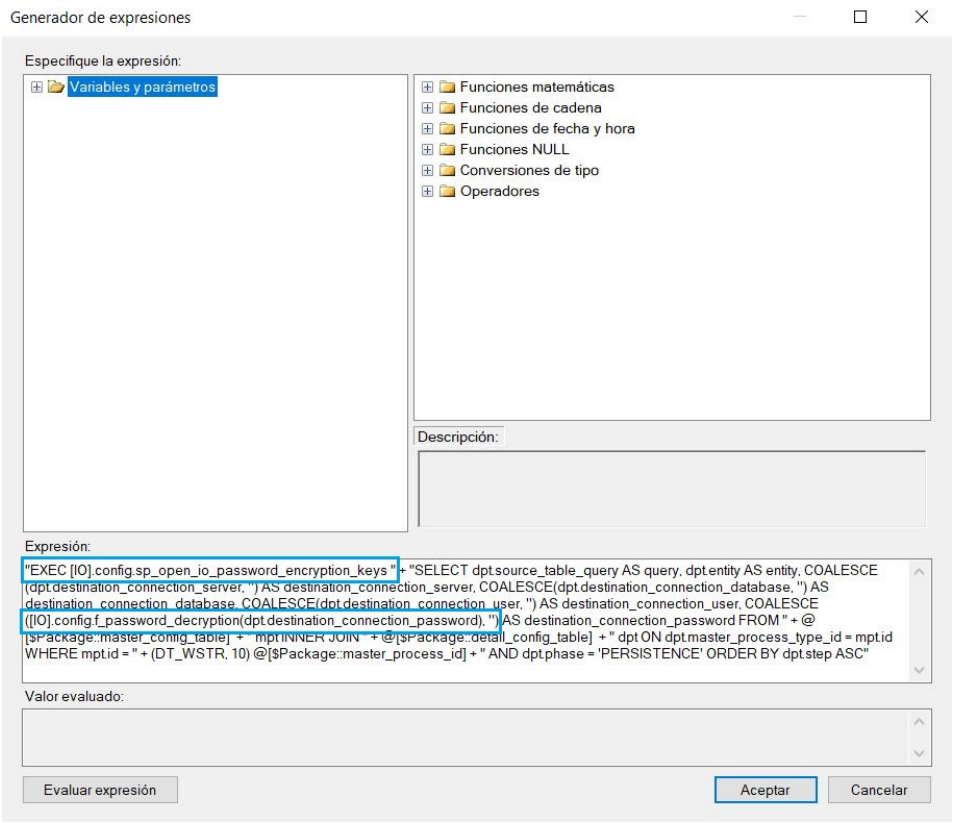


Figura 5.1: Expresión de desencriptado en variable de paquete dtsx

5.2.1.3. Modo de empleo de las funcionalidades

Es primordial que antes de realizar cualquier encriptación o desencriptación, se ejecute previamente el procedimiento almacenado *IO.config.sp_open_io_password_encryption_keys* como en el *Código 5.7* para la apertura de claves, de lo contrario el proceso no funcionará.

Código 5.7: Encriptado BBDD IO.sql

```
-- Apertura de las claves de cifrado, siempre se ha de hacer antes de encriptar o desencriptar
EXEC [IO].config.sp_open_io_password_encryption_keys
-- Funcion de encriptado, entrada VARCHAR, salida VARBINBARY
SELECT [IO].config.f_password_encryption('P4$$W0rd_unico')
-- Función de desencriptado, entrada VARBINARY, salida VARCHAR
SELECT [IO].config.f_password_decryption(0x00BA1DC5F52ACF05CEF7C5F899A696B302000000B16...)
```

5.2.2. Ingesta de archivos en BBDD

Continuando con las herramientas de soporte, se ha desarrollado una herramienta en SQL que permite la ingesta de archivos utilizando como destino una tabla de BBDD. Los formatos de archivo permitidos son los siguientes.

- CSV
- DAT
- TXT
- XLS
- XLSX

Esta herramienta se encuentra dentro del procedimiento almacenado *IO.input.sp_load_file_to_table*, el cual está comentado en detalle, por lo que nos basaremos en sus comentarios para ir explicándolo a continuación.

5.2.2.1. Código del procedimiento

Comenzamos con la cabecera del procedimiento como indica el *Código 5.8*, la cual contiene una breve descripción, sus parámetros de entrada, y un ejemplo de ejecución del procedimiento.

Código 5.8: Cabecera del procedimiento sp_load_file_to_table

```
USE [IO];
GO
/*
-- ============================================================================================
Autor: Adrian Maroto
Fecha creacion: 14/03/2024
Descripcion: Proceso de ingesta de archivos para procesos IO, se realiza la ingesta en función de una serie de
     parámetros y basándose en una
   plantillas .FMT en caso que la ingesta sea con archivos como origen.
Cambios:

Entrada:
   @p_ingest = Identificador del proceso.
   @p_path = Ruta completa al archivo a ingestar.
   @p_template = Ruta completa al archivo .fmt que sirve como plantilla de carga.
   @p_skip_initial_rows = Filas iniciales a descartar de la ingesta (cabeceras).
   @p_skip_final_rows = Filas finales a descartar de la ingesta (pies).
   @p_destination_table = Tabla de destino donde depositar los datos del archivo de ingesta.
```

```
   @p_order = Número de orden en la ingesta del archivo,
      si @p_order = 1 se borra la tabla y se recrea,
      en caso contrario se insertan directamente los datos en la tabla existente.

Salida:

Ejemplo:
   EXEC IO.input.sp_load_file_to_table
      @p_ingest = 1
      ,@p_path = '\\WIN-SER-AMAROTO\IO_Files\Input\BMW Group\Concesionarios\BMW_NewDealers_20240313_1415.csv'
      ,@p_template = '\\WIN-SER-AMAROTO\IO_Files\Template\BMW Group\Concesionarios.fmt'
      ,@p_skip_initial_rows = 1
      ,@p_skip_final_rows = 0
      ,@p_destination_table = '[IO].[input].BMW_NewDealers'
      ,@p_order = 1
-- ============================================================================================
*/
```

Creación de variables de soporte necesarias durante la ejecución, se añade la cláusula *NOCOUNT ON* para omitir log de SQL Server, y se inicializa la variable *p_skip_initial_rows*. Se muestra en el *Código 5.9*.

Código 5.9: Variables del procedimiento sp_load_file_to_table

```
   SET NOCOUNT ON

   -- Varias variables de soporte al proceso
    DECLARE @sql NVARCHAR(MAX)
         ,@log NVARCHAR(MAX)
         ,@table NVARCHAR(MAX)
         ,@firstfield NVARCHAR(1024)
         ,@sql_columns NVARCHAR(MAX)
         ,@sql_columns_func NVARCHAR(MAX)
         ,@campos NVARCHAR(MAX)
         ,@p_extension NVARCHAR(8)

   -- Por defecto le sumamos uno para que elimine filas iniciales
   SET @p_skip_initial_rows += 1;

    -- Tabla auxiliar que sirve para almacenar las columnas del archivo
   DECLARE @columns TABLE(id INT IDENTITY(1,1), [columns] VARCHAR(MAX))
```

El *Código 5.10* inserta una nueva fila en la tabla de log interno *IO.monitor.log* indicando el inicio del proceso de ingesta del archivo.

Código 5.10: Log de inicio del procedimiento sp_load_file_to_table

```
   -- Se inserta una nueva fila en la tabla de log interno indicando el inicio del proceso de ingesta del archivo
    SET @log = 'INICIO carga [input].[sp_load_file_to_table] fichero ' + @p_path + ' orden ' + CAST(@p_order AS
         VARCHAR)
    EXEC monitor.sp_write_log @@PROCID, '', @log,'INFO'
```

Se realizan las siguientes operaciones previas mediante el *Código 5.11*.

- Se obtiene la extensión del archivo para poder saber el tipo de ingesta a realizar.
- En caso que sea la primera ingesta del bloque se borra la tabla de destino.
- Se crea una tabla de carga previa para evitar machacar la tabla de destino final en caso de error.
    - Se borra la tabla de carga previa en caso de existir.

- Se obtienen las columnas que formarán la tabla a partir de la definición existente en la plantilla *FMT* y se insertan los nombres en una variable de tipo tabla creada en los pasos anteriores.

Código 5.11: Código previo a la ingesta del procedimiento sp_load_file_to_table

```
-- Obtenemos la extensión del archivo a partir del parámetro parámetro @p_path
SET @p_extension = REVERSE(LEFT(REVERSE(@p_path),CHARINDEX('.',REVERSE(@p_path))-1))

-- Si el parámetro @p_order es 1 se borra la tabla de destino
IF @p_order = 1
   BEGIN
      --PRINT '-- Entra 1';
      SET @sql = ' DROP TABLE IF EXISTS ' + @p_destination_table

      --PRINT (@sql)
      EXEC (@sql)
   END

/* Se genera una tabla auxiliar añadiendole al nombre de la tabla destino un HASH NEWID */

-- Obtención del nuevo nombre de la tabla auxiliar
SET @table = REPLACE(REPLACE(@p_destination_table, ']', ''), '[', '') + REPLACE(CAST(NEWID() AS
    VARCHAR(1024)), '-', '_')

-- Se borra la tabla auxiliar en caso de existir y se obtienen todas las columnas a partir de la lectura del
    archivo de plantilla
SET @sql = 'SET NOCOUNT ON
        DROP TABLE IF EXISTS '+ @table + '
        SELECT [COLUMNS]
        INTO ' + @table + '
        FROM (
           SELECT REPLACE(SUBSTRING(v_1,1,CHARINDEX(CHAR(9),v_1)), '' '', '''') + ''NVARCHAR(MAX), '' [COLUMNS]
           FROM (
              SELECT SUBSTRING(value,(CHARINDEX(CHAR(9),[value]))+1,len([value])) V_1
              FROM OPENROWSET( BULK ' + CHAR(39) + @p_template + CHAR(39) + ',SINGLE_BLOB) AS x
              CROSS APPLY string_split(CAST(bulkcolumn AS VARCHAR(MAX)),char(10))
           ) TT
           WHERE V_1 != '''' AND SUBSTRING(v_1,1,CHARINDEX(CHAR(9),v_1)) != ''''
        ) T'

--PRINT (@sql)
EXEC (@sql)

-- Una vez tenemos los nombres de todas las columnas se insertan en la tabla creada para ello
SET @sql = 'SELECT Columns FROM ' + @table
INSERT INTO @columns EXEC(@sql)
```

Si el valor del parámetro *@p_order* es igual a 1, se crea la tabla de destino, la cual se ha borrado en el paso anterior. Ver el *Código 5.12*

Código 5.12: CREATE tabla destino del procedimiento sp_load_file_to_table

```
-- Si el p_order = 1, se crea la tabla donde se almacenará el contenido de los archivos a ingestar, el nombre
    de la tabla es el que viene como parámetro, caso contrario se omite este paso
IF @p_order = 1
   BEGIN
      --PRINT '-- Entra 2';
      SET @sql = NULL
      SELECT @sql = COALESCE(@sql + ' ', '') + [COLUMNS] FROM @columns
      -- El nombre del archivo y la fecha de ingesta se añaden en los campos ingested_filename, ingestion_date
          respectivamente
      SET @sql = ' CREATE TABLE ' + @p_destination_table + '(' + SUBSTRING(@sql,1,len(@sql) - 1) + ',
          ingested_filename VARCHAR(MAX), ingestion_date DATETIME) '

      --PRINT (@sql)
      EXEC (@sql)
   END
```

Se crea la tabla auxiliar en el *Código 5.13* con todos los campos de tipo *NVARCHAR(MAX)* para evitar desbordamientos.

Código 5.13: CREATE tabla auxiliar del procedimiento sp_load_file_to_table

```
-- Se elimina y crea la tabla auxiliar para ingestar el archivo, todos los campos se tratan como tipo NVARCHAR(MAX) para evitar desbordamientos
SET @sql = 'DROP TABLE IF EXISTS ' + @table
EXEC(@sql)

SET @sql = NULL
SELECT @sql = COALESCE(@SQL + ' ', '') + [COLUMNS] FROM @columns
SET @table = REPLACE(REPLACE(@p_destination_table, ']', ''), '[', '') + REPLACE(CAST(NEWID() AS VARCHAR(1024)), '-','_')
SET @campos = REPLACE(@sql,'NVARCHAR(MAX)','')
SET @sql = ' CREATE TABLE ' + @table + '( ' + SUBSTRING(@sql,1,len(@sql) - 1) + ') '
EXEC (@sql)
```

A continuación el *Código 5.14* carga los datos del archivo en la tabla temporal.

- Si se trata de un fichero Excel con extensión XLS o XLSX se insertan los datos a partir de un OPENROWSET usando el conector 12.0 de Excel.
- En caso contrario, se ejecuta un BULK INSERT estándar de SQL Server con parámetros *FIRSTROW* y *LASTROW* para, en caso necesario, eliminar filas iniciales o finales respectivamente.

Código 5.14: Carga de datos temporales del procedimiento sp_load_file_to_table

```
/* Insertamos los datos del fichero según la plantilla y la extensión del archivo origen*/

-- En caso de ser un Excel se insertan los datos a partir de un OPENROWSET y usando el conector para Excel 12.0
IF @p_extension IN ('xlsx','xls')
   BEGIN
      --PRINT '-- Entra 3';
      -- Quitamos la última coma
      SET @campos = LEFT(trim(@campos),LEN(trim(@campos))-1)
      SET @sql = 'INSERT INTO ' + @table + '(' + @campos + ')
              SELECT *
              FROM OPENROWSET(''Microsoft.ACE.OLEDB.12.0'', ''Excel 12.0;Database=' + @p_path + ';'', ''SELECT * FROM [Hoja1$]'') '

      --PRINT (@sql)
      EXEC (@sql)
   END
ELSE
   -- En caso que no sea un Excel se ejecuta un bulk insert estandar de SQL Server
   BEGIN
      --PRINT '-- Entra 4';

      DECLARE @par_definition NVARCHAR(128);
      DECLARE @total_rows INT;
      DECLARE @sql_string NVARCHAR(512);
      SET @sql_string = N'DECLARE @file VARBINARY(MAX)
                DECLARE @filetext VARCHAR(MAX)
                SELECT @file = CAST(bulkcolumn AS VARBINARY(MAX)), @filetext = bulkcolumn
                          FROM OPENROWSET (BULK ' + CHAR(39) + @p_path + CHAR(39) + ',SINGLE_BLOB) AS X
                SELECT @total_rows = COUNT(*) FROM string_split(@filetext,CHAR(10)) WHERE value != '''';
      SET @par_definition = N'@total_rows INT OUTPUT';
      EXECUTE sp_executesql @sql_string, @par_definition, @total_rows = @total_rows OUTPUT;

      SET @sql = 'BULK INSERT ' + @table + '
              FROM ' + CHAR(39) + @p_path + CHAR(39) + '
              WITH (
                 BATCHSIZE=200000,
                 ROWS_PER_BATCH=200000,
                 FORMATFILE = ' + CHAR(39) + @p_template + CHAR(39) + ',
                 FIRSTROW = ' + CAST(@p_skip_initial_rows AS VARCHAR(256)) + ',
```

```
                LASTROW = ' + CAST(@total_rows - @p_skip_final_rows AS VARCHAR(256)) + '
            )'

      --PRINT (@sql)
      EXEC (@sql)
   END
```

Posteriormente se almacena en el log interno *IO.monitor.log* un registro indicando que la carga en la tabla temporal se ha insertado de forma correcta mediante el *Código 5.15*

Código 5.15: Log en tabla temporal del procedimiento sp_load_file_to_table

```
-- Se inserta una nueva fila en la tabla de log interno indicando que se han insertado los datos del archivo
    en la tabla auxiliar
 SET @log = ' Datos insertados en tabla ' + @table
 EXEC monitor.sp_write_log @@PROCID, '', @log,'INFO'
```

El *Código 5.16* replica los datos en la tabla final de destino de la ingesta mediante las siguientes operaciones.

- Se obtienen todos los nombres de las columnas excepto los relativos al nombre del archivo y la fecha de carga.
- Se realiza una limpieza de datos utilizando la función *IO - f_from_utf8_to_utf16.sql* C.31 para estandarizar los valores a *UTF8*.
- Se crea una tabla auxiliar para depositar los datos depurados.
- Se insertan los datos directamente en la tabla de destino añadiendo el nombre del archivo y la fecha en la que se efectuó la carga.

Código 5.16: Replica de datos en destino del procedimiento sp_load_file_to_table

```
/* Se insertan los datos en la tabla final de destino*/

-- Se obtienen todos los nombres de las columnas excepto los relativos al nombre del archivo y la fecha de
    carga
SET @sql_columns = null
 SELECT @sql_columns = COALESCE(@sql_columns + ',','') + ' [' + name + '] '
 FROM syscolumns
 WHERE id = OBJECT_ID(@p_destination_table)
   AND [name] NOT IN ('ingested_filename', 'ingestion_date')

-- Se obtienen todos los nombres de las columnas excepto los relativos al nombre del archivo y la fecha de
    carga. Se forma la query que realizará la limpieza de los datos
SET @sql_columns_func = null
 SELECT @sql_columns_func = COALESCE(@sql_columns_func + ',','') +
     '[input].[f_from_utf8_to_utf16](REPLACE(REPLACE(['+ name +'],''"'',' + '''''),''&'',''''')) ' + '['+ name
     +']'
 FROM syscolumns
 WHERE id = OBJECT_ID(@p_destination_table)
   AND [name] NOT IN ('ingested_filename', 'ingestion_date')

-- Se crea una tabla auxiliar para dejar los datos ya depurados
DECLARE @table_aux VARCHAR(MAX)
SET @table_aux = REPLACE(REPLACE(@p_destination_table, ']', ''), '[', '') + REPLACE(CAST(NEWID() AS
    VARCHAR(1024)), '-','_')

SET @sql = 'DROP TABLE IF EXISTS ' + @table_aux + ';
           SELECT ' + @sql_columns_func + 'INTO ' + @table_aux + ' FROM ' + @table
EXEC (@sql)

-- Se insertan los datos directamente en la tabla de destino añadiendo el nombre del archivo y la fecha cuando
    se efectuó la carga
```

```
26    SET @sql = 'INSERT INTO ' + @p_destination_table + '(' + @sql_columns + ', ingested_filename, ingestion_date)
          ' +
27            'SELECT ' + @sql_columns + ',''' + @p_path + ''', GETDATE()' + ' FROM ' + @table_aux
28    EXEC (@sql)
```

Se almacena en el log interno *IO.monitor.log* un registro indicando la carga en la tabla de destino. Ver *Código 5.17*.

Código 5.17: Log de carga de destino del procedimiento sp_load_file_to_table

```
1    -- Se inserta un registro en la tabla de log indicando que se han insertado los datos depurados
2    SET @log = ' Update eliminación de caracteres especiales ' + @p_destination_table + ' (' + cast(@@rowcount as
         varchar) + ')'
3     EXEC monitor.sp_write_log @@PROCID, '', @log,'INFO'
```

Adicionalmente, el *Código 5.18* elimina las tablas temporales utilizadas.

Código 5.18: DROP de tablas auxiliar del procedimiento sp_load_file_to_table

```
1    -- Se eliminan las tablas auxiliares
2     SET @sql = 'DROP TABLE IF EXISTS ' + @table
3     EXEC(@sql)
4    SET @sql = 'DROP TABLE IF EXISTS ' + @table_aux
5     EXEC(@sql)
```

Se almacena en el log interno *IO.monitor.log* un registro indicando la finalización de la ingesta del archivo en la tabla de destino mediante el *Código 5.19*.

Código 5.19: Log de finalización del procedimiento sp_load_file_to_table

```
1    -- Se inserta un registro en la tabla de log indicando que ha finalizado la ingesta del archivo
2     SET @log = 'FIN carga [input].[sp_load_file_to_table] fichero ' + @p_path + ' orden ' + CAST(@p_order AS
          VARCHAR)
3     EXEC monitor.sp_write_log @@PROCID, '', @log, 'INFO'
```

Por último, en caso de que haya algún error durante la ejecución, este provocará el lanzamiento del bloque *CATCH*, el cual ejecuta el *Código 5.20*.

Código 5.20: Bloque CATCH del procedimiento sp_load_file_to_table

```
1  BEGIN CATCH
2
3    /* EN CASO DE ERROR */
4    -- Se obtienen los datos relativos al error
5    SELECT @Log = CONCAT (ErrorNumber, ' :: ' , ErrorState , ' :: ' , ErrorProcedure , ' :: ' , ErrorLine , ' :: '
         , ErrorMessage)
6     FROM (
7       SELECT
8         ERROR_NUMBER() AS ErrorNumber,
9         ERROR_STATE() AS ErrorState,
10        ERROR_PROCEDURE() AS ErrorProcedure,
11        ERROR_LINE() AS ErrorLine,
12        ERROR_MESSAGE() AS ErrorMessage
13     ) aux
14
15   -- Se eliminan las tablas auxiliares
16    SET @sql = 'DROP TABLE IF EXISTS ' + @table
17    EXEC(@sql)
18   SET @sql = 'DROP TABLE IF EXISTS ' + @table_aux
19    EXEC(@sql)
20
21   -- Se inserta un registro en la tabla de log indicando el error que ha ocurrido
22    EXEC monitor.sp_write_log @@PROCID, '', @Log, 'ERROR'
23
24 END CATCH
```

5.2.3. Carga y descarga de archivos por FTP/SFTP

Continuando con las herramientas adicionales a la solución IO, se ha desarrollado una herramienta en PowerShell que permite la carga y/o descarga de archivos a través de protocolo FTP, aunque también está preparada para protocolo SFTP, que por motivos de arquitectura no se ha podido probar.

Esta herramienta se encuentra en el fichero *ftp_sftp.ps1*, el cual está comentado en detalle, por lo que nos basaremos en sus comentarios para ir explicándolo a continuación.

5.2.3.1. Código del fichero

Comenzamos con la cabecera del fichero en el *Código 5.21*, la cual contiene una breve descripción, sus parámetros de entrada, y un ejemplo de ejecución del procedimiento.

Código 5.21: Cabecera del fichero ftp_stftp.ps1

```
<# ============================================================
Autor: Adrian Maroto
Fecha creacion: 09/01/2024
Descripcion: Script que realiza una conexión FTP o SFTP y carga o descarga archivos que tienen que cumplir un patrón.
Cambios:

Entrada:
    $protocol: tipo de conexión al servidor, valores posibles FTP o SFTP
    $server: nombre o IP del servidor al que conectarse
    $port: puerto de conexión
    $user: usuario de las credenciales de conexión al servidor
    $pass: password de las credenciales de conexión al servidor
    $ftp_operation: tipo de operación que se realizará sobre el servidor, valores posibles UPLOAD o DOWNLOAD
    $ftp_folder: directorio remoto sobre el que se realizará la operación
    $ftp_file_pattern: patrón que debe cumplir los archivos en el lado del servidor (sólo válido para operaciones de descarga, DOWNLOAD)
    $local_folder: directorio local sobre el que se realizará la operación
    $local_file_pattern: patrón que debe cumplir los archivos locales (sólo válido para operaciones de carga, UPLOAD)

    $ftp_private_key_path: directorio donde se encuentra la clave privada en caso de conexión segura
    $ftp_ssh_host_key_fingerprint: identificador de la huella del servidor
    $ftp_private_key_passphrase: password de uso de la clave privada de acceso al servidor

Salida:
   - DOWNLOAD/UPLOAD del fichero.

Ejemplo:
    FTP-DOWNLOAD: ftp_sftp.ps1 -protocol "FTP" -server "nombre_o_IP_servidor_ftp" -port "puerto" -user "usuario_conexion" -pass "contraseña_conexion" -ftp_operation "DOWNLOAD" -ftp_folder "directorio_remoto_descarga" -ftp_file_pattern "patrón_archivos_descarga" -local_folder "directorio_local_descarga"
    FTP-UPLOAD: ftp_sftp.ps1 -protocol "FTP" -server "nombre_o_IP_servidor_ftp" -port "puerto" -user "usuario_conexion" -pass "contraseña_conexion" -ftp_operation "UPLOAD" -ftp_folder "directorio_remoto_carga" -local_folder "directorio_local_carga" -local_file_pattern "patrón_archivos_carga"
    SFTP-DOWNLOAD: ftp_sftp.ps1 -protocol "SFTP" -server "nombre_o_IP_servidor_ftp" -port "puerto" -user "usuario_conexion" -pass "contraseña_conexion" -ftp_operation "DOWNLOAD" -ftp_folder "directorio_remoto_descarga" -ftp_file_pattern "patrón_archivos_descarga" -local_folder "directorio_local_descarga"
    SFTP-UPLOAD: ftp_sftp.ps1 -protocol "SFTP" -server "nombre_o_IP_servidor_ftp" -port "puerto" -user "usuario_conexion" -pass "contraseña_conexion" -ftp_operation "UPLOAD" -ftp_folder "directorio_remoto_carga" -local_folder "directorio_local_carga" -local_file_pattern "patrón_archivos_carga"
============================================================ #>
```

Dependiendo del tipo de operación y la seguridad del servidor FTP/SFTP, algunos de estos parámetros pueden ser nulos. El paquete *Ingest_process* de nuestra solución SSIS lo tiene en cuenta según la parametrización del proceso.

Es importante destacar que todas las contraseñas han de llegar en formato plano sin encriptar, de la misma forma el paquete *Ingest_process* de nuestra solución SSIS se encarga de desencriptarlas y enviarlas como parámetro, como hemos visto en *Ingesta de archivos en BBDD* 5.2.2.

El *Código 5.22* realiza la declaración de los parámetros necesarios.

Código 5.22: Declaración de parámetros del fichero ftp_stftp.ps1

```
param (
    [string] $protocol,
    [string] $server,
    [int] $port,
    [string] $user,
    [string] $pass,
    [string] $ftp_operation,
    [string] $ftp_folder,
    [string] $ftp_file_pattern,
    [string] $local_folder,
    [string] $local_file_pattern,

    [string] $ftp_private_key_path,
    [string] $ftp_ssh_host_key_fingerprint,
    [string] $ftp_private_key_passphrase
)
```

Mediante el *Código 5.23* se evita registrar todo el log del proceso, centralizando la información de la ejecución mediante escritura por consola.

Código 5.23: Deshabilita el debug del fichero ftp_stftp.ps1

```
Set-PSDebug -Off
```

Comenzando en el código, se puede observar que hay un bloque *try-catch-finally* que realiza la conexión y la operación de carga o descarga, además de una pequeña gestión de errores la cual veremos posteriormente.

A continuación se detallan los bloques del código principal.

Se reinician las variables de log y desbloquea el archivo .ps1 para poder ser ejecutado en remoto. Además se realiza la carga de la librería *WinSCP* con extensión .NET para poder utilizarla durante el proceso [12]. Se muestra en el *Código 5.24*.

Código 5.24: Carga de la librería WinSCP del fichero ftp_stftp.ps1

```
    # Inicialización de la variable de mensaje de salida
    $error_message = 'OK'

    # desbloqueo del archivo para poder ser ejecutado en remoto
    $main_dir = $PSScriptRoot.ToString()
    Get-ChildItem -Path $main_dir -File -Recurse -Force |  % {
        Unblock-File -Path $_.FullName
    }

    # Carga de la libreria WinSCP .NET
    [System.Reflection.Assembly]::UnsafeLoadFrom($main_dir + "\WinSCP\WinSCPnet.dll") | Out-Null
```

Mediante el *Código 5.25* se estandariza el valor de la variable *protocol*, para evitar errores de transcripción, y se crea un objeto *WinSCP*. Adicionalmente se configuran características especificas del protocolo utilizado e iniciamos sesión en el servidor FTP/SFTP.

- En caso de protocolo FTP:
  - Se establece la seguridad como explicita.
  - Se acepta el certificado del servidor FTP.
- En caso de protocolo SFTP:
  - Se acepta la clave *SSH* predeterminada.
  - Se carga la clave privada en caso de existir.
- En caso de tener el fingerprint de la máquina desde donde se ejecuta el cliente, se carga su valor en ambos casos.

Código 5.25: Configuración del protoclo del fichero ftp_stftp.ps1

```
#Estandarización del protocolo (primera mayúscula, resto minúsculas)
$protocol = $protocol.substring(0,1).ToUpper() + $protocol.substring(1).ToLower()

# Configuración de las opciones de sesión
$sessionOptions = New-Object WinSCP.SessionOptions -Property @{
    Protocol = [WinSCP.Protocol]::$protocol
    HostName = $server
    PortNumber = $port
    UserName = $user
    Password = $pass
}

# En caso de que se trate de una conexión SFTP se acepta la SSH Host Key predeterminada y en caso de existir clave privada, se carga
if ($protocol -eq "Sftp")
{
    $sessionOptions.GiveUpSecurityAndAcceptAnySshHostKey = "true"

    if($ftp_private_key_path -ne ""){
        $sessionOptions.SshPrivateKeyPath = $ftp_private_key_path
    }

    if($ftp_private_key_passphrase -ne ""){
        $sessionOptions.PrivateKeyPassphrase = $ftp_private_key_passphrase
    }
}
else
{
    $sessionOptions.FtpSecure = "Explicit"
    $sessionOptions.GiveUpSecurityAndAcceptAnyTlsHostCertificate = "true"
}

#En caso de tener la fingerprint de la máquina desde donde se ejecuta el cliente, se carga su valor
if($ftp_ssh_host_key_fingerprint -ne ""){
    $sessionOptions.SshHostKeyFingerprint = $ftp_ssh_host_key_fingerprint
}

# Creación de la sesión
$session = New-Object WinSCP.Session

# Conectar
$session.Open($sessionOptions)
```

A continuación el *Código 5.26* realiza la operación de *UPLOAD/DOWNLOAD* según esté establecido por configuración.

Código 5.26: Ejecución de la operación del fichero ftp_stftp.ps1

```
# Según la operación se realizan las diferentes acciones
```

```
    if ($ftp_operation -eq "DOWNLOAD")
    {
        if (!(Test-Path $local_folder))
        {
              New-Item -ItemType Directory -Path $local_folder -Force | Out-Null
        }

        $all_files = $session.EnumerateRemoteFiles($ftp_folder, $ftp_file_pattern,
     [WinSCP.EnumerationOptions]::None)

        $count_files = 0

        foreach ($file in $all_files)
        {
            $count_files = $count_files + 1
            $download_file =  $ftp_folder + $file.Name
            $session.GetFiles($download_file, ($local_folder + '\ ')).Check()
            $session.RemoveFiles($download_file) | Out-Null
        }

        if ($count_files -eq 0){
            throw "No file matching template: " + $ftp_file_pattern + " on " + $protocol.ToUpper() + " server
     path: " + $ftp_folder + ". Unable to DOWNLOAD any file"
        }
    }

    if ($ftp_operation -eq "UPLOAD")
    {
        if (!$session.FileExists($ftp_folder))
        {
            $session.CreateDirectory($ftp_folder)
        }

        $all_files =  Get-ChildItem -Path $local_folder -File | where {$_.name -like $local_file_pattern}

        $count_files = 0

        $transferOptions = New-Object WinSCP.TransferOptions
        $transferOptions.FilePermissions = $Null
        $transferOptions.PreserveTimestamp = $False

        foreach ($file in $all_files)
        {
            $count_files = $count_files + 1
            $upload_file = $local_folder + '\ ' + $file.Name
            $session.PutFiles($upload_file, $ftp_folder, $False, $transferOptions).Check()
        }

        if ($count_files -eq 0){
            throw "No file matching template: " + $local_file_pattern + " on path: " + $local_folder + ". Unable
     to UPLOAD to " + $protocol.ToUpper()
        }
    }
}
```

Por último, en caso de encontrar algún error, el *Código 5.27* ejecuta el bloque *Catch*, el cual nos mostrará *WARNING* si la cantidad de archivos es igual a 0, o *KO* en caso contrario. Por otro lado la cláusula *Finally* realiza la desconexión del servidor y devuelve el mensaje del error.

Código 5.27: Gestión de errores del fichero ftp_stftp.ps1

```
Catch
{
    if ($count_files -eq 0)
    {
        $error_message = 'WARNING: '
    } else
    {
        $error_message = 'KO: '
    }

```

```
    $error_message = $error_message + $_.Exception.Message
}
finally
{
    # Desconexión
    if (Get-Variable -Name session -ErrorAction SilentlyContinue)
    {
        $session.Dispose()
    }

    Write-Host -NoNewline $error_message
}
```

5.2.4. Compresión y descompresión de archivos

Dentro de las herramientas específicas en PowerShell para la solución IO, se ha desarrollado mediante código una utilidad para permitirnos la compresión y descompresión de archivos para los siguiente formatos.

- GZIP
- RAR
- ZIP

Adicionalmente la herramienta nos da la posibilidad de comprimir o descomprimir archivos utilizando protección mediante contraseña. Se encuentra en el fichero *zip_unzip.ps1*, el cual está comentado en detalle, por lo que nos basaremos en sus comentarios para ir explicándolo a continuación.

5.2.4.1. Código del fichero

Comenzamos con la cabecera del fichero como indica el *Código 5.28*, la cual contiene una breve descripción, sus parámetros de entrada, y un ejemplo de ejecución del procedimiento.

Código 5.28: Cabecera del fichero zip_unzip_file.ps1

```
<# ============================================================
Autor: Adrian Maroto
Fecha creacion: 23/01/2024
Descripcion: Script que comprime/descomprime un archivo depositando el resultado en otra carpeta.
Cambios:

Entrada:
    $zip_operation: tipo de conexión al servidor, valores posibles FTP o SFTP
    $zip_folder: carpeta donde se encuentra el archivo comprimido
    $zip_file: nombre del archivo comprimido
    $output_folder: carpeta destino de los archivos descomprimidos
    $archive_folder: ruta_principal_carpeta_archivado
    $zip_rar_pass: contraseña_de_descompresion

Salida:
   - ZIP/UNZIP del fichero.

Ejemplo:
    UNZIP: zip_unzip_files.ps1 -zip_operation "UNZIP" -file "ruta_completa_archivo_zip" -folder
     "carpeta_destino_archivos_descomprimidos" -archive_folder "ruta_principal_carpeta_archivado"
    ZIP: zip_unzip_files.ps1 -zip_operation "ZIP" -folder "ruta_carpeta_archivos_zipear\*" -file
     "ruta_archivo_comprimido_zip"
============================================================ #>
```

En caso que no se quiera proteger o desproteger, el parámetro *zip_rar_pass* deberá estar vacío.

Es importante destacar que todas las contraseñas han de llegar en formato plano sin encriptar, de la misma forma el paquete *Ingest_process* de nuestra solución SSIS se encarga de desencriptarlas y enviarlas como parámetro, como hemos visto en *Ingesta de archivos en BBDD* 5.2.2.

Se realiza la declaración de los parámetros necesarios como se muestra en el *Código 5.29*.

Código 5.29: Declaración de parámetros del fichero zip_unzip_file.ps1

```
param (
   [string] $zip_operation,
   [string] $folder,
   [string] $file,
   [string] $archive_folder,
   [string] $zip_rar_pass
)
```

Mediante el *Código 5.30* se evita registrar todo el log del proceso, centralizando la información de la ejecución mediante escritura por consola.

Código 5.30: Deshabilita el debug del fichero zip_unzip_file.ps1

```
Set-PSDebug -Off
```

Comenzando en el código, se puede observar que hay un bloque *try-catch-finally* que realiza la operación de compresión o descompresión, además de una pequeña gestión de errores al final del código.

A continuación se detallan los bloques del código principal.

El *Código 5.31* crea la carpeta de archivado en caso de no existir y desbloquea el archivo para poder ser ejecutado en remoto.

Código 5.31: Código previo del fichero zip_unzip_file.ps1

```
    # Inicialización de la variable de mensaje de salida
    $error_message = 'OK'

    #Se añade una contrabarra a la ruta de archivado para poder crear el directorio en caso de no existir
    $archive_folder = $archive_folder + "\ "

    if (!(Test-Path $archive_folder))
    {
        New-Item -ItemType Directory -Path $archive_folder -Force | Out-Null
    }

    # Desbloqueo del archivo para poder ser ejecutado en remoto
    $main_dir = $PSScriptRoot.ToString()
    Get-ChildItem -Path $main_dir -File -Recurse -Force |  % {
        Unblock-File -Path $_.FullName
    }
```

Cuando la operación a realizar es de tipo *ZIP* (compresión), se siguen los siguientes pasos como muestra el *Código 5.32*.

1. Se modifica el nombre del archivo sustituyendo el formato de fecha, siempre que la contenga como sigue.

a) YYYY = Año actual en formato de 4 dígitos.

b) MM = Mes actual en formato de 2 dígitos.

c) DD-1 = Día anterior a hoy en formato de 2 dígitos.

d) DD = Día actual en formato de 2 dígitos.

2. Se obtiene la carpeta y el nombre del archivo de destino por separado.
3. En caso de existir el archivo comprimido de destino, se borra para evitar errores.
4. Se comprime el archivo con o sin contraseña en función del parámetro *zip_rar_pass*.
5. Se copia el archivo generado en la ubicación del histórico.

Código 5.32: Código de compresión del fichero zip_unzip_file.ps1

```
if ($zip_operation -eq 'ZIP'){

    $file = $file.Replace("yyyy", (get-date -Format yyyy)).Replace("mm", (get-date -Format
 MM)).Replace("dd-1", (get-date).AddDays(-1).Day.ToString()).Replace("dd", (get-date -Format dd))
    $zip_file_folder = (Split-Path $folder -parent) + "\ "
    $zip_file_template = Split-Path $folder -leaf

    #If(Test-path $file){
    #    Remove-item $file
    #}

    # Compresión del archivo RAR sin contraseña
    if ($zip_rar_pass -eq ''){
        Compress-7Zip -ArchiveFileName $file -Path $zip_file_folder -Filter $zip_file_template -Format Zip
    # Compresión del archivo RAR con contraseña
    } else {
        Compress-7Zip -ArchiveFileName $file -Path $zip_file_folder -Filter $zip_file_template -Format Zip
 -Password $zip_rar_pass
    }

    # Se archiva el fichero ZIP una vez comprimido
    Copy-Item $file -Destination $archive_folder -Force
}
```

Por el contrario, cuando se trata de una operación *UNZIP* (descompresión) se lanza el *Código 5.33*.

1. Se obtienen la carpeta y el nombre del archivo de origen para obtener la ubicación y la plantilla de este.
2. Posteriormente se obtiene el archivo que cumplen la plantilla y se descomprime.
3. Se copia el archivo generado en la ubicación del histórico.

Código 5.33: Código de descompresión del fichero zip_unzip_file.ps1

```
if ($zip_operation -eq 'UNZIP'){

    $zip_file_folder = Split-Path $file -parent
    $zip_file_template = Split-Path $file -leaf

    Get-ChildItem -Path $zip_file_folder -File | where {$_.name -like $zip_file_template} | % {

        $zip_filename = $_.FullName

```

```
            # Descompresión del archivo sin contraseña
            if ($zip_rar_pass -eq ''){
                Expand-7Zip -ArchiveFileName $zip_filename -TargetPath $folder
            # Descompresión del archivo RAR con contraseña
            } else {
                Expand-7Zip -ArchiveFileName $zip_filename -TargetPath $folder -Password $zip_rar_pass
            }
        }

        # Se archiva el fichero ZIP una vez descomprimido
        Move-Item $zip_filename $archive_folder -Force
    }
```

Es importante destacar que esta funcionalidad sirve para uno o varios archivos en la misma ubicación.

Por último, en caso de encontrar algún error, el *Código 5.34* ejecuta el bloque *Catch*, escribiendo *KO* en caso de error, a su vez la cláusula *Finally* devuelve el mensaje del error.

Código 5.34: Gestión de errores del fichero zip_unzip_file.ps1

```
Catch
{
    $error_message = 'KO ' + $_.Exception.Message
}
finally
{
    Write-Host -NoNewline $error_message
}
```

5.2.5. Generación de ficheros

Dentro de las herramientas específicas en PowerShell para la solución IO, se ha desarrollado mediante código una utilidad que genera un fichero en diferentes formatos a partir de una consulta. Los formatos disponibles son los siguientes.

- CSV
- DAT
- TXT
- XLS
- XLSX

La herramienta se encuentra en el fichero *output_generator.ps1*, el cual está comentado en detalle, por lo que nos basaremos en sus comentarios para ir explicándolo a continuación.

5.2.5.1. Código del fichero

El *Código 5.35* comienza con la cabecera del fichero, la cual contiene una breve descripción, sus parámetros de entrada, y un ejemplo de ejecución del procedimiento.

Código 5.35: Cabecera del fichero output_generator.ps1

```
<# ============================================================
Autor: Adrian Maroto
Fecha creacion: 16/01/2024
```

```
Descripcion: Script que genera un fichero en diferentes formatos a partir de una query.
Cambios:

Entrada:
    $database: base de datos sobre la que se ejecutará la consulta que genera el output
    $query: query que se ejecutará para generar el output
    $out_path: ruta donde se depositará el output a partir de la query
    $fmt_path: ruta donde se encuentra el archivo de configuración del output
    $sql_server: servidor donde se ejecutará la query para generar el output
    $output_format: formato del archivo de salida
    $header_rows: número de filas que tiene la cabecera
    $generate_empty: indicador de si se debe generar o no el fichero en caso que solo haya cabecera en el
     contenido
    $archive_folder: carpeta en la que se archivará el archivo generado copiandose el original

Salida:
   - Fichero con el formato indicado.

Ejemplo:
    output_generator.ps1 -query "SELECT * FROM db.schema.tabla" -out_path "ruta_destino_incluyendo_archivo"
     -fmt_path "ruta_archivo_configuración" -sql_server "nombre_servidor_SQL_origen" -database "bbdd_origen"
     -output_format "formato archivo output" -header_rows 3 -generate_empty 1)
============================================================ #>
```

Se realiza la declaración de los parámetros necesarios como muestra el *Código 5.36*.

Código 5.36: Declaración de parámetros del fichero output_generator.ps1

```
param (
   [string] $database,
   [string] $query,
   [string] $out_path,
   [string] $fmt_path,
   [string] $sql_server,
   [string] $output_format,
   [int] $header_rows,
   [int] $generate_empty,
   [string] $archive_folder
)
```

Mediante el *Código 5.37* se evita registrar todo el log del proceso, centralizando la información de la ejecución mediante escritura por consola.

Código 5.37: Deshabilita el debug del fichero output_generator.ps1

```
Set-PSDebug -Off
```

Comenzando en el código, se puede observar que hay un bloque *try-catch-finally* que realiza la operación de generación del fichero, además de una gestión básica de errores al final del código.

A continuación se detallan los bloques del código principal.

Creación de la carpeta de destino en caso de no existir y desbloquea el archivo para poder ser ejecutado en remoto mediante el *Código 5.38*

Código 5.38: Código previo del fichero output_generator.ps1

```
    # Inicialización de la variable de mensaje de salida
    $error_message = 'OK'

    # Se crea el directorio de destino en caso de no existir
    If(!(test-path -PathType container $archive_folder))
    {
        New-Item -ItemType Directory -Path $archive_folder | Out-Null
    }
```

```
9
10  # Se reemplazan los carácteres especiales @@@ por "" que son necesarias dentro de la query, además se omiten
     los saltos de línea
11  $query = $query.Replace("@@@", '`"`"').replace("\n", ' ').replace("`n"," ")
12
13  # desbloqueo del archivo para poder ser ejecutado en remoto
14  $main_dir = $PSScriptRoot.ToString()
15  Get-ChildItem -Path $main_dir -File -Recurse -Force |  % {
16      Unblock-File -Path $_.FullName
17  }
```

El *Código 5.39* modifica el nombre del archivo y la extensión con las siguientes reglas.

1. Se modifica el nombre del archivo sustituyendo el formato de fecha, siempre que la contenga como sigue.
   a) YYYY = Año actual en formato de 4 dígitos.
   b) MM = Mes actual en formato de 2 dígitos.
   c) DD = Día actual en formato de 2 dígitos.
   d) hh = Hora actual en formato de 2 dígitos.
   e) min = Minutos actuales en formato de 2 dígitos.
   f) min = Segundos actuales en formato de 2 dígitos.
2. Se incluye la extensión 'TXT' o 'CSV' al nombre del archivo.

Código 5.39: Modificación del nombre de salida del fichero output_generator.ps1

```
1   # Modificación del nombre del archivo de salida para que contenga la fecha de hoy
2   $out_path = $out_path.Replace('yyyy', (Get-Date -Format yyyy).ToString()).Replace('mm', (Get-Date -Format
     MM).ToString()).Replace('dd', (Get-Date -Format dd).ToString()).Replace('hh', (Get-Date -Format
     hh).ToString()).Replace('min', (Get-Date -Format mm).ToString()).Replace('ss', (Get-Date -Format
     ss).ToString())
3
4   # Modificación del nombre del archivo para que contenga la extensión
5   # En caso de ser formato Excel se creará primero un CSV para posteriormente modificarlo
6   if (($output_format -eq 'TXT') -or ($output_format -eq 'CSV'))
7   {
8       $bcp_out_file = $out_path + '.' + $output_format.ToLower()
9   }
10  else
11  {
12      $bcp_out_file = $out_path + '.csv'
13  }
```

Posteriormente eliminamos el archivo del destino en caso de existir, generamos la nueva versión con el comando BCP [13], realizamos una copia en la ubicación del archivado y revisamos que el archivo no esté vacío mediante el *Código 5.40*.

Código 5.40: Creación del archivo con el comando BCP del fichero output_generator.ps1

```
1   # Creación del archivo CSV a partir de la query de entrada y el archivo de configuración FMT
2   $bcp_command = "BCP `"$query`" queryout `"$bcp_out_file`" -f `"$fmt_path`" -S `"$sql_server`" -d
     `"$database`" -T"
3
4   Invoke-Expression $bcp_command | Out-Null
5
6   #Se copia el archivo en la ubicación de archivado
7   Copy-Item $bcp_out_file -Destination $archive_folder -Force
```

```
8
9       # Se calcula si la cantidad de filas en el archivo resultado es únicamente el de las cabeceras
10      $empty_file = (Get-Content -Path $bcp_out_file | Measure-Object -Line | Select -ExpandProperty "Lines") -le
         $header_rows
```

El *Código 5.41* se encarga de generar ficheros de tipo XLS o XLSX, para ello previamente generamos un fichero CSV el cual transformamos a la nueva extensión. Para generar el archivo Excel usamos la librería *excel.application*, el código va recorriendo el fichero CSV y escribiendo los datos en el nuevo archivo. Por último se guarda el archivo con la extensión elegida y, en caso de existir, se cierran los procesos abiertos de Excel para evitar errores de escritura.

Código 5.41: Generación del archivo en formato XLS o XLSX del fichero output_generator.ps1

```
1       # Si el archivo solo contiene las cabeceras, se borra en caso contrario se continua
2       if ($empty_file -and $generate_empty -eq 0)
3       {
4           # Se elimina el archivo CSV del BCP
5           Remove-Item $bcp_out_file
6       }
7       else
8       {
9           # Si se espera un Excel como formato del output, se transforma el CSV en Excel
10          if (($output_format -eq 'XLS') -or ($output_format -eq 'XLSX'))
11          {
12              # Ruta de salida del Excel
13              $excel_out_path = $out_path + '.' + $output_format.ToLower()
14
15              # Creación del Excel añadiendo un nuevo libro y hoja
16              $excel = New-Object -ComObject excel.application
17              $workbook = $excel.Workbooks.Add(1)
18              $worksheet = $workbook.worksheets.Item(1)
19
20              # Se especifica el formato como solo texto
21              $worksheet.Cells.NumberFormat = "@"
22
23              # Se recorre cada una de las filas y columnas leyendo y asignado el valor al Excel
24              $i = 1
25              Import-Csv $bcp_out_file -Delimiter ";" | ForEach-Object {
26                $j = 1
27                foreach ($prop in $_.PSObject.Properties) {
28                  if ($i -eq 1) {
29                    $worksheet.Cells.Item($i, $j++).Value = $prop.Name
30                    $worksheet.Cells.Item($i+1, $j-1).Value = $prop.Value
31                  } else {
32                    $worksheet.Cells.Item($i+1, $j++).Value = $prop.Value
33                  }
34                }
35                $i++
36              }
37
38              # Se elimina el archivo Excel en caso de existir
39              if (Test-Path $excel_out_path) {
40                  Remove-Item $excel_out_path
41              }
42
43              # Se guarda el archivo Excel diferenciando entre versiones de Excel
44              if ($output_format -eq 'XLSX')
45              {
46                  $Workbook.SaveAs($excel_out_path,51)
47              }
48              else
49              {
50                  $Workbook.SaveAs($excel_out_path,56)
51              }
52
53              # Se cierra el archivo Excel
54              $excel.Quit()
55
56              # Se eliminan todos los procesos de tipo Excel para evitar errores
57              # Check for process by name
```

```
            Try {
                $processes = Get-Process -Name "Excel" -ErrorAction Stop

                foreach($process in $processes)
                {
                    Stop-Process -InputObject $process
                }

                $clean = 1
            }
            Catch {
                $clean = 0
            }

            #Se copia el archivo en la ubicación de archivado
            Copy-Item $excel_out_path -Destination $archive_folder -Force

            # Se elimina el archivo CSV del BCP
            Remove-Item $bcp_out_file
        }
    }
}
```

Por último, en caso de encontrar algún error, el *Código 5.42* ejecuta el bloque *Catch*, escribiendo *KO* en caso de error, a su vez la cláusula *Finally* devuelve el mensaje del error.

Código 5.42: Gestión de errores del fichero output_generator.ps1

```
Catch
{
    $error_message = 'KO ' + $_.Exception.Message
}
finally
{
    Write-Host -NoNewline $error_message
}
```

5.3. Proceso main

Main_process es el proceso maestro de la solución IO, necesario para realizar cualquiera de los pasos que contiene el proyecto, por lo tanto hará las veces de director de cada uno de los pasos que hayamos programado para nuestro proceso. Se muestra en las *Figuras 5.2-5.3*.

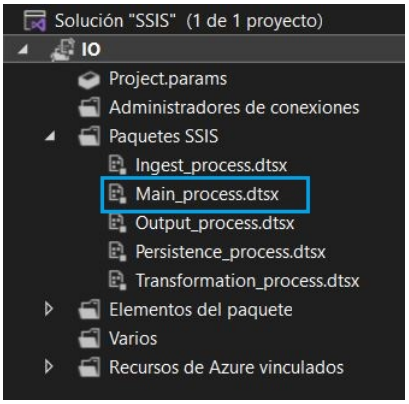


Figura 5.2: Paquete Main_process de la solución SSIS

Para la ejecución de *Main_process* se han generado una serie de parámetros los cuales nos permiten configurar la información core del mismo. Estos se muestran en la *Figura 5.4*.

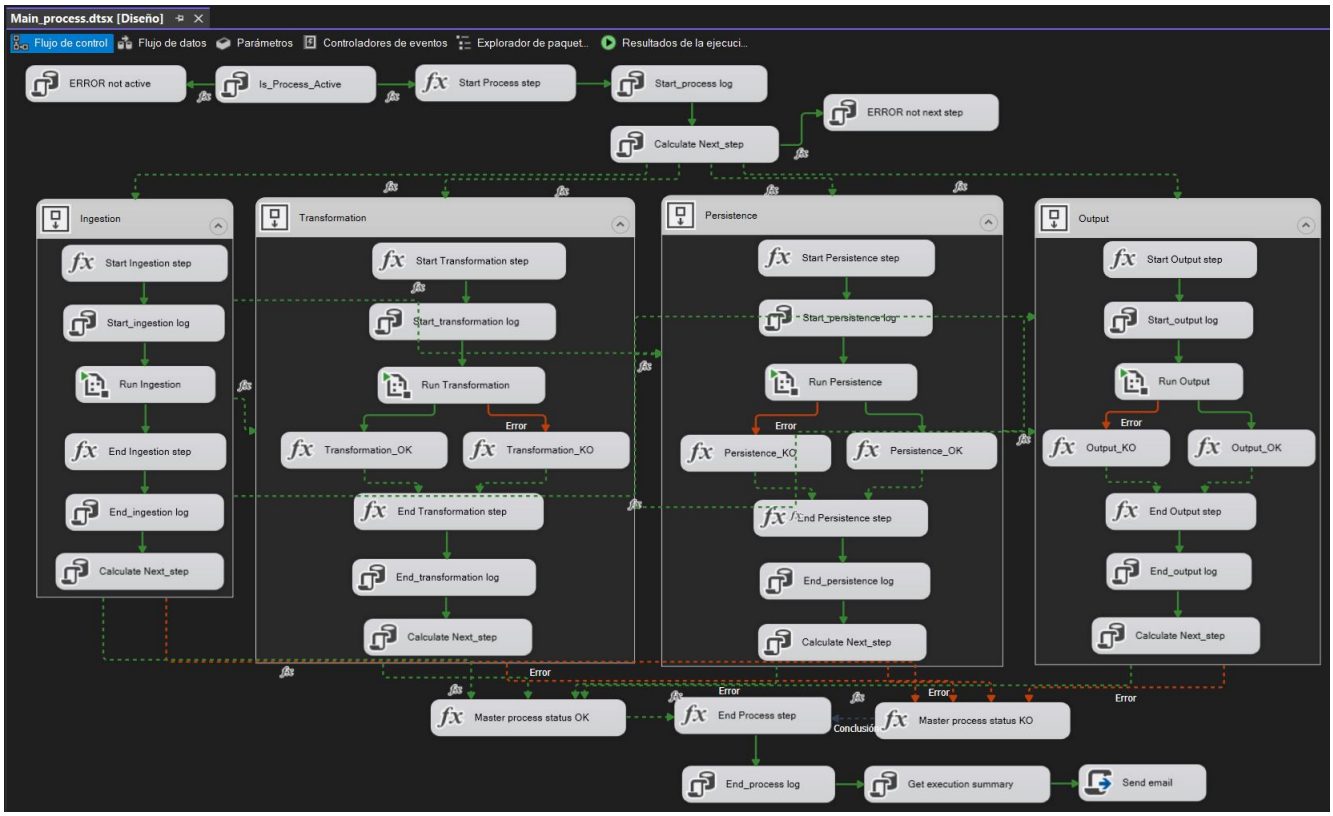
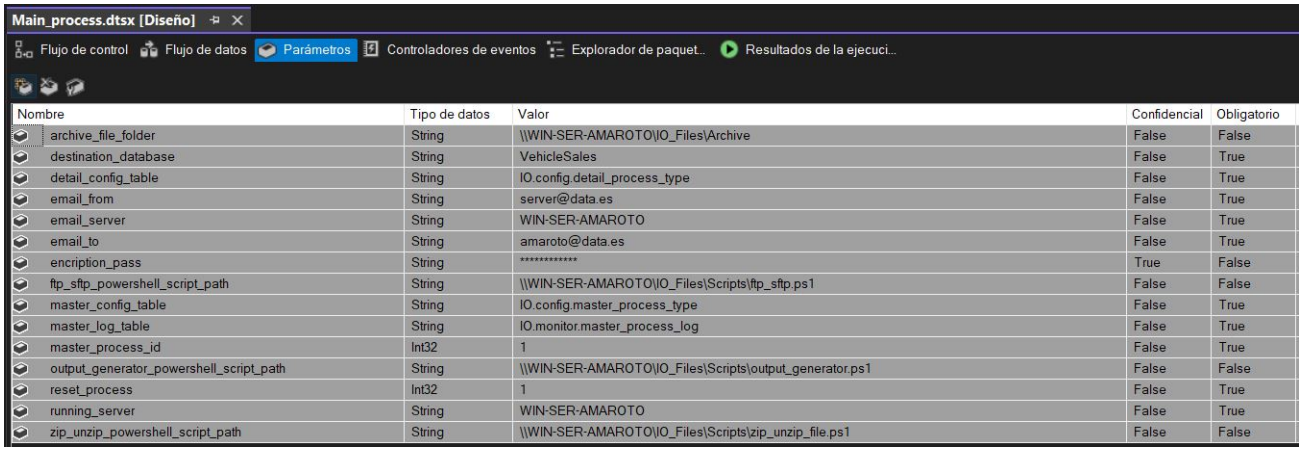


Figura 5.3: Flujo de control del paquete Main_process



Nombre	Tipo de datos	Valor	Confidencial	Obligatorio
archive_file_folder	String	\\WIN-SER-AMAROTO\IO_Files\Archive	False	False
destination_database	String	VehicleSales	False	True
detail_config_table	String	IO.config.detail_process_type	False	True
email_from	String	server@data.es	False	True
email_server	String	WIN-SER-AMAROTO	False	True
email_to	String	amaroto@data.es	False	True
encription_pass	String	**********	True	False
ftp_sftp_powershell_script_path	String	\\WIN-SER-AMAROTO\IO_Files\Scripts\ftp_sftp.ps1	False	False
master_config_table	String	IO.config.master_process_type	False	True
master_log_table	String	IO.monitor.master_process_log	False	True
master_process_id	Int32	1	False	True
output_generator_powershell_script_path	String	\\WIN-SER-AMAROTO\IO_Files\Scripts\output_generator.ps1	False	False
reset_process	Int32	1	False	True
running_server	String	WIN-SER-AMAROTO	False	True
zip_unzip_powershell_script_path	String	\\WIN-SER-AMAROTO\IO_Files\Scripts\zip_unzip_file.ps1	False	False

Figura 5.4: Parámetros del paquete Main_process

A continuación se aporta una breve descripción de cada uno de ellos:

- archive_file_folder: Ruta a la carpeta principal de archivado de ficheros, tanto de entrada como de salida.
- destination_database: BBDD destinto donde escribirá el proceso a ejecutar.
- detail_config_table: Tabla de configuración a nivel de detalle. Ver *Diseño de BBDD y modelo entidad relación IO* 4.4 del capítulo 4 para más información.
    - Por defecto *IO.config.detail_process_type*.

- email_from: Dirección de correo desde la que se envían los correos en caso necesario.
  - Por defecto *server@data.es*.
- email_server: Servidor SMTP de correo.
  - Por defecto *WIN_SER_AMAROTO*.
- email_to: Listado de direcciones, separadas por coma "," o punto y coma ";", a las que se enviará el correo resumen con el resultado de la ejecución del proceso.
- encription_pass: Contraseña de encriptación de archivos de entrada que se inserta una vez tratados.
- ftp_sftp_powershell_script_path: Ruta donde se encuentra el script de gestión de conexiones FTP/SFTP. Ver *Carga y descarga de archivos por FTP/SFTP* 5.2.3 en este capítulo para más información.
- master_config_table: Tabla de configuración a nivel de maestro. Ver *Diseño de BBDD y modelo entidad relación IO* 4.4 del capítulo 4 para más información.
  - Por defecto *IO.config.master_process_type*.
- master_log_table: Tabla de log maestra para persistir la traza del proceso.
  - Por defecto *IO.monitor.master_process_log*.
- master_process_id: Identificador único del proceso a ejecutar.
- output_generator_powershell_script_path: Ruta donde se encuentra el script de generación de procesos de salida. Ver *Proceso de generación de outputs* 5.7 en este capítulo para más información.
- reset_process: Flag para identificar si el proceso comienza desde el principio o se ejecuta a partir del último paso correcto.
  - 0: En caso de error continuará donde falló la última ejecución.
  - 1: Comienza siempre desde el primer paso del proceso.
- running_server: Servidor donde se encuentra el motor de IO.
- zip_unzip_powershell_script_path: Ruta donde se encuentra el script de compresión y descompresión de archivos. Ver *Compresión y descompresión de archivos* en este capítulo para más información.

5.3.0.1. Detalle de los bloques del proceso

Al ejecutar el proceso se realiza una comprobación para verificar que el proceso se encuentra habilitado (*Código 5.5*). Para esta comprobación se ejecutan dos componentes de ejecución de tareas SQL, alimentados con las variables *sql_get_process_active* y *sql_not_active_process*.

En caso de no aprobar la condición, se inserta un registro de error en *IO.monitor.master_process_log* y se provoca el fallo en el proceso.

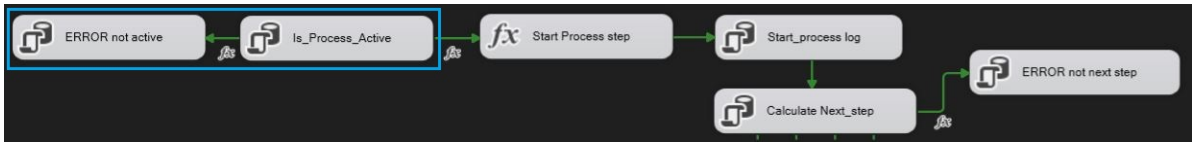


Figura 5.5: Comprobaciones de proceso activo

En caso contrario, si todo funciona correctamente se inserta un registro de ejecución general en *IO.monitor.master_process_log* como muestra la *Figura 5.6*.

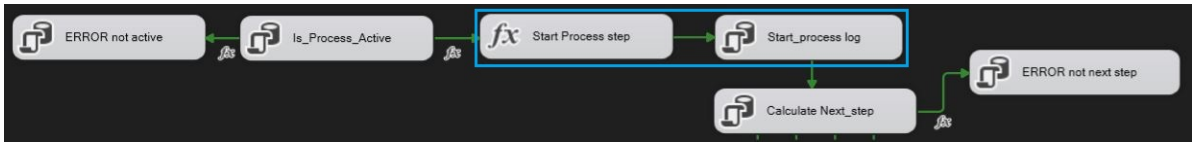


Figura 5.6: Log de ejecución general del proceso

A continuación la *Figura 5.7* muestra el cálculo del siguiente step a ejecutar, para ello se ejecuta el procedimiento almacenado *IO.config.sp_calculate_next_step* el cual determina el siguiente paso a ejecutar dependiendo de la configuración maestra del proceso, además del valor del parámetro *reset_process*.

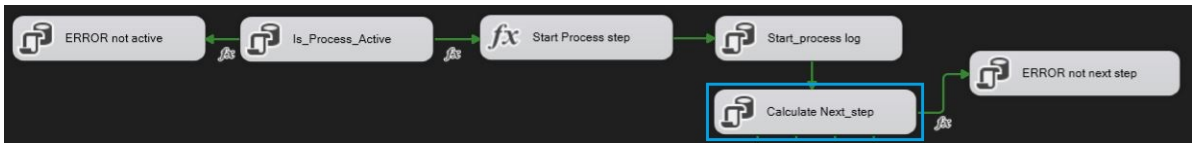


Figura 5.7: Cálculo del siguiente step del proceso

Este procedimiento almacenado necesita los siguientes parámetros.

- p_master_process_id: Identificador único del proceso a ejecutar. El valor se obtiene del parámetro *"master_process_id"*.
- p_reset: Flag para identificar si el proceso comienza desde el principio o se ejecuta a partir del último paso correcto. El valor se obtiene del parámetro *reset_process*.
- p_step = Nombre del step actual en el que se encuentra el proceso. El valor se obtiene de la variable *v_process_step*. El valor por defecto es *BEGINNING*.
- p_out_next_step = ? OUTPUT: Parámetro de salida que proporciona el siguiente step a ejecutar del proceso. Los valores disponibles son:
  - BEGINNING: Primer step de cada ejecución, se identifica con el inicio del proceso.
  - INGESTION: Ejecuta el contenedor de secuencia *Ingestion*, el cual a su vez llama al paquete *Ingest_process* mediante la tarea *Run Ingestion* (*Figura 5.8*). Para más información ver el apartado *Proceso de ingesta* 5.4 de este capítulo.

Figura 5.8: Contenedor de secuencia de ingesta

- TRANSFORMATION: Ejecuta el contenedor de secuencia *Transformation*, el cual a su vez llama al paquete *Transformation_process* mediante la tarea *Run Transformation* (*Figura 5.9*). Para más información ver el apartado *Proceso de transformación* 5.5 de este capítulo.

Figura 5.9: Contenedor de secuencia de transformación

- PERSISTENCE: Ejecuta el contenedor de secuencia *Persistence*, el cual a su vez llama al paquete *Persistence_process* mediante la tarea *Run Persistence* (*Figura 5.10*). Para más información ver el apartado *Proceso de persistencia* 5.6 de este capítulo.

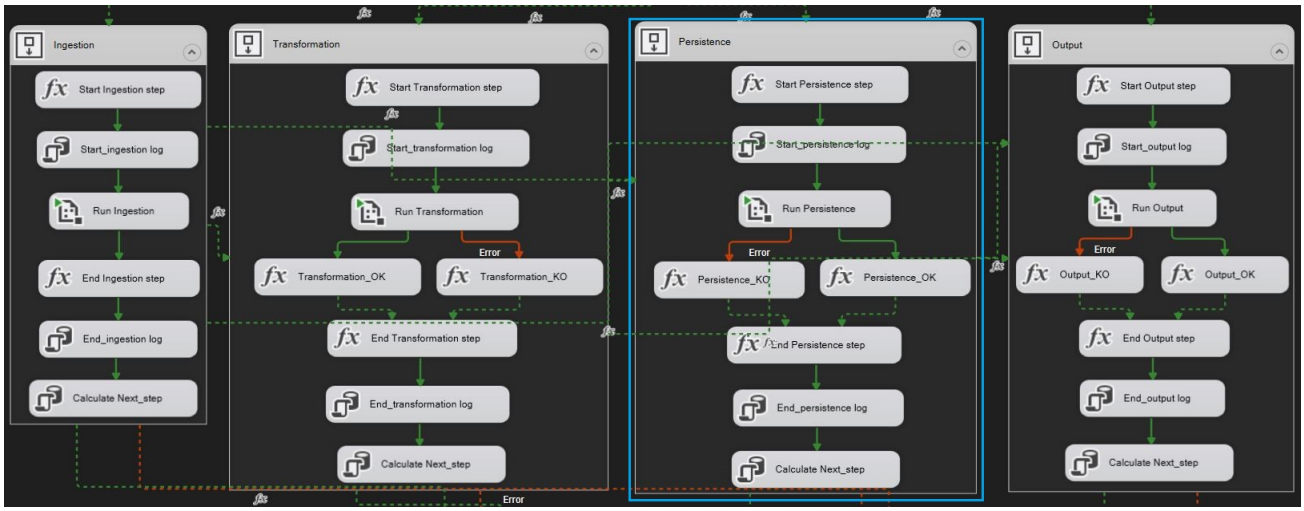


Figura 5.10: Contenedor de secuencia de persistencia

- OUTPUT: Ejecuta el contenedor de secuencia *Output*, el cual a su vez llama al paquete *Output_process* mediante la tarea *Run Output* (*Figura 5.11*). Para más información ver el apartado *Proceso de generación de outputs* 5.7 de este capítulo.

Figura 5.11: Contenedor de secuencia de salida

- ERROR: En caso de no haber siguiente step para el proceso lanza la tarea *ERROR not next step* generando un error como muestra la *Figura 5.12*.

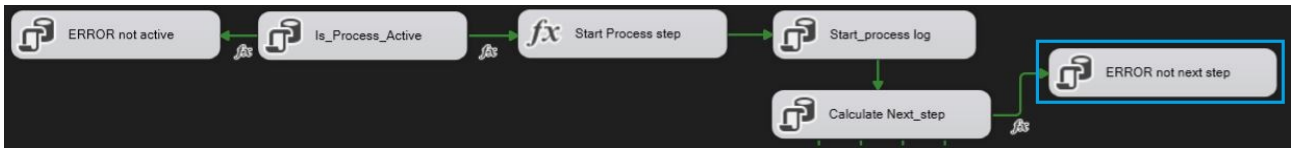


Figura 5.12: Error por no existencia de step

- END: El proceso concluye de forma correcta, habiendo ejecutado todos los steps del proceso como muestra la *Figura 5.13*.

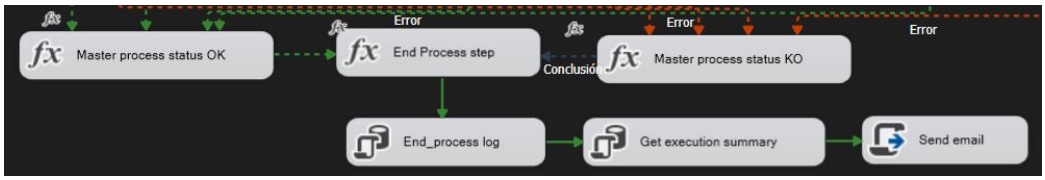


Figura 5.13: Bloque final tras ejecución del proceso

Una vez finalizado el conjunto de steps, el paquete genera un correo con el resumen de la ejecución del proceso (*Figura 5.13*). Para ello se utilizan dos componentes de SSIS:

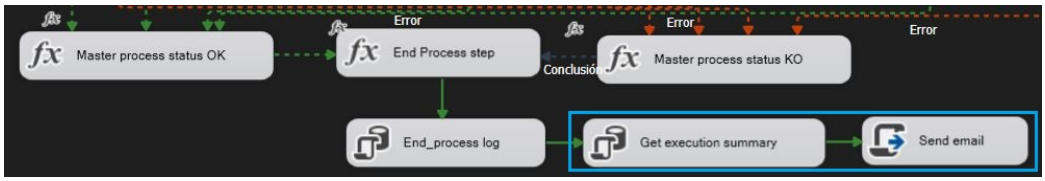


Figura 5.14: Tarea SQL get execution summary

Se detallan a continuación ambos componentes. *Get execution summary*: Componente de tarea SQL que ejecuta el procedimiento almacenado *IO.monitor.sp_get_execution_summary* el cual devuelve el log de la ejecución (*Figura 5.15*).

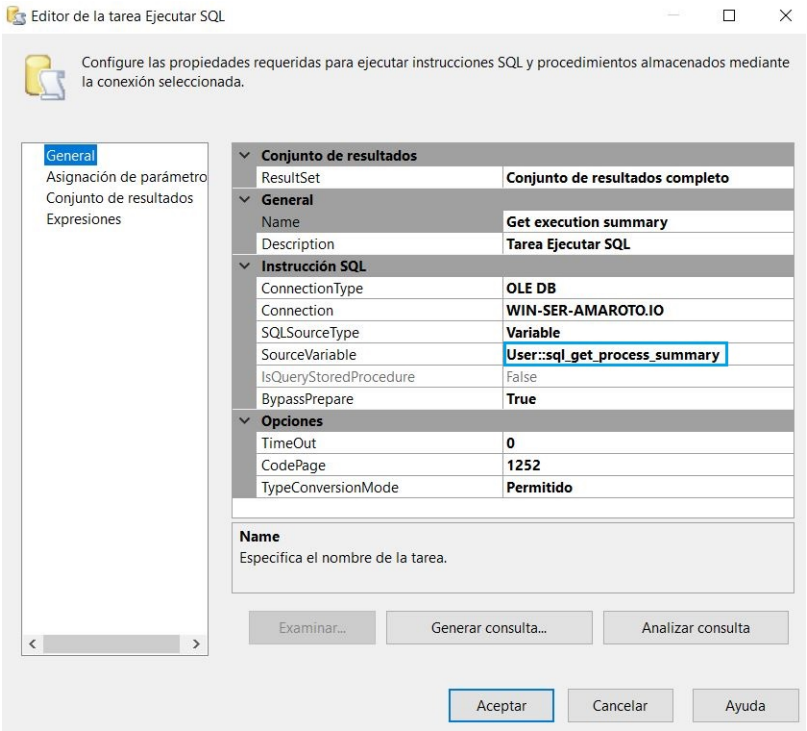


Figura 5.15: Detalle de tarea get execution summary

El *Código 5.43* muestra el procedimiento almacenado.

Código 5.43: monitor.sp_get_execution_summary.sql

```
USE [IO]
GO
/*
-- ============================================================================================
Autor: Adrian Maroto
```

```
Fecha creación: 14/03/2024
Descripción: Procedimiento almacenado para obtener un resumen de la ejecución de procesos.

Entrada:
    @master_process_type_id: ID del proceso maestro.
    @master_process_log_id: ID del log de registro del proceso maestro.

Salida:

Ejemplo:
   EXEC [monitor].[sp_get_execution_summary]
      @master_process_type_id = 5,
      @master_process_log_id = 1

   SELECT * FROM [IO].config.master_process_type
   SELECT * FROM [IO].monitor.master_process_log
-- ============================================================================================
*/
CREATE OR ALTER PROCEDURE [monitor].[sp_get_execution_summary]
   @master_process_type_id INT,
   @master_process_log_id INT
AS
BEGIN

   DROP TABLE IF EXISTS #process_execution

   -- Crea una tabla temporal para almacenar el resumen de la ejecución del proceso
   SELECT dpl.id AS id,
      mpt.customer AS cartera,
      mpt.[name] AS process_name,
      mpt.[description] AS process_description,
      mpl.master_process_status AS process_status,
      mpl.master_process_start_time AS process_start_time,
      mpl.master_process_end_time AS process_end_time,
      mpl.error_description AS process_message,
      mpl.ingestion_status AS ingestion_status,
      mpl.ingestion_start_time AS ingestion_start_time,
      mpl.ingestion_end_time AS ingestion_end_time,
      mpl.ingestion_end_time - mpl.ingestion_start_time AS ingestion_duration,
      mpl.transformation_status AS transformation_status,
      mpl.transformation_start_time AS transformation_start_time,
      mpl.transformation_end_time AS transformation_end_time,
      mpl.transformation_end_time - mpl.transformation_start_time AS transformation_duration,
      mpl.persistence_status AS persistence_status,
      mpl.persistence_start_time AS persistence_start_time,
      mpl.persistence_end_time AS persistence_end_time,
      mpl.persistence_end_time - mpl.persistence_start_time AS persistence_duration,
      mpl.output_status AS output_status,
      mpl.output_start_time AS output_start_time,
      mpl.output_end_time AS output_end_time,
      mpl.output_end_time - mpl.output_start_time AS output_duration,
      dpl.phase AS phase,
      dpl.entity AS entity,
      dpl.[status] AS phase_entity_status,
      dpl.quantity_rows AS phase_entity_affected_rows,
      dpl.start_time AS phase_entity_start_time,
      dpl.end_time AS phase_entity_end_time,
      CONVERT(NVARCHAR, dpl.end_time - dpl.start_time, 114) AS phase_entity_duration,
      dpl.error_description AS phase_entity_message
   INTO #process_execution
   FROM [IO].config.master_process_type mpt
      INNER JOIN [IO].monitor.master_process_log mpl ON mpt.id = mpl.master_process_type_id
      INNER JOIN [IO].monitor.detail_process_log dpl ON mpl.id = dpl.master_process_log_id
   WHERE mpt.id = @master_process_type_id
      AND mpl.id = @master_process_log_id

   -- Devuelve el resumen de la ejecución del proceso
   SELECT
      cartera,
      process_name,
      process_description,
      process_status,
      process_start_time,
```

```
79      process_end_time,
80      CONVERT(NVARCHAR, (COALESCE(ingestion_duration,
81         '1900-01-01 00:00:00.000')
82         + COALESCE(transformation_duration,
83         '1900-01-01 00:00:00.000')
84         + COALESCE(persistence_duration,
85         '1900-01-01 00:00:00.000')
86         + COALESCE(output_duration,
87         '1900-01-01 00:00:00.000') ), 114)
88      AS process_duration,
89      process_message,
90      ingestion_status,
91      ingestion_start_time,
92      ingestion_end_time,
93      CONVERT(NVARCHAR, ingestion_duration, 114) AS ingestion_duration,
94      CASE
95         WHEN (SELECT COUNT(*) FROM #process_execution WHERE phase_entity_status = 'WARNING') > 0
96            AND transformation_status IS NOT NULL AND transformation_status != 'N/A'
97         THEN 'WARNING'
98         ELSE transformation_status
99      END AS transformation_status,
100     transformation_start_time,
101     transformation_end_time,
102     CONVERT(NVARCHAR, transformation_duration, 114) AS transformation_duration,
103     persistence_status,
104     persistence_start_time,
105     persistence_end_time,
106     CONVERT(NVARCHAR, persistence_duration, 114) AS persistence_duration,
107     output_status,
108     output_start_time,
109     output_end_time,
110     CONVERT(NVARCHAR, output_duration, 114) AS output_duration,
111     phase,
112     entity,
113     phase_entity_status,
114     phase_entity_affected_rows,
115     phase_entity_start_time,
116     phase_entity_end_time,
117     phase_entity_duration,
118     phase_entity_message
119  FROM #process_execution
120  ORDER BY id ASC
121
122 END
```

Se recuperan todos los registros del estado de la ejecución del proceso, y se almacenan en la variable de tipo objeto *obj_execution_summary*.

Send email: Componente de tipo script que ejecuta el envío de un correo en formato HTML con el resumen de la ejecución del proceso (*Figura 5.16*). Se muestra a continuación el *Código 5.44*.

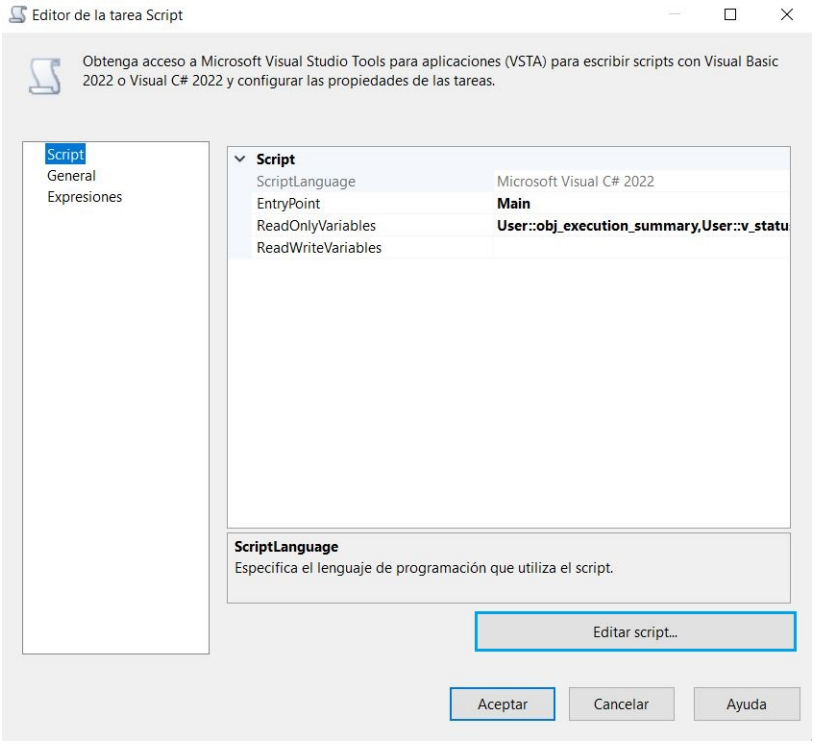


Figura 5.16: Tarea get process summary

Código 5.44: Main - Send email.cs

```
#region Namespaces
using System;
using System.Data;
using System.Data.OleDb;
using System.Net.Mail;
using Microsoft.SqlServer.Dts.Runtime;
using System.Windows.Forms;
#endregion

namespace ST_712178489edc4ca69c38495b49170f8d
{
   [Microsoft.SqlServer.Dts.Tasks.ScriptTask.SSISScriptTaskEntryPointAttribute]
   public partial class ScriptMain : Microsoft.SqlServer.Dts.Tasks.ScriptTask.VSTARTScriptObjectModelBase
   {
      public void Main()
      {
            try
            {
                String from = (String)Dts.Variables["$Package::email_from"].Value;
                String to = (String)Dts.Variables["$Package::email_to"].Value;
                String smtp_host = (String)Dts.Variables["$Package::email_server"].Value;
                String computer_name = Environment.MachineName.ToString();
                String status = (String)Dts.Variables["User::v_status"].Value;

                DataTable table = new DataTable();
                OleDbDataAdapter adapter = new OleDbDataAdapter();
                adapter.Fill(table, Dts.Variables["User::obj_execution_summary"].Value);

                String subject = "";
                String summary_header_text = "";
                String summary_text = "";
                String detail_text = "<p> <u> Detailed execution </u> </p>" +
               "<table cellpadding=\"5\" cellspacing=\"0\"> " +
               "<tr bgcolor=\"DeepSkyBlue\"> " +
               "<th style=\"border: 1px solid black; border-collapse: collapse\"> PHASE </th> " +
               "<th style=\"border: 1px solid black; border-collapse: collapse\"> ENTITY </th> " +
               "<th style=\"border: 1px solid black; border-collapse: collapse\"> STATUS </th> " +
               "<th style=\"border: 1px solid black; border-collapse: collapse\"> AFFECTED ROWS </th> " +
               "<th style=\"border: 1px solid black; border-collapse: collapse\"> START TIME </th> " +
```

```
                "<th style=\"border: 1px solid black; border-collapse: collapse\"> END TIME </th> " +
                "<th style=\"border: 1px solid black; border-collapse: collapse\"> DURATION </th> " +
                "<th style=\"border: 1px solid black; border-collapse: collapse\" > MESSAGE </th> " +
                "</tr> ";

                 foreach (DataRow row in table.Rows)
                 {
                     Object[] row_content = row.ItemArray;

                     String customer = row_content[0].ToString();
                     String process_name = row_content[1].ToString();
                     String process_description = row_content[2].ToString();
                     String process_status = row_content[3].ToString();
                     String process_start_time = row_content[4].ToString();
                     String process_end_time = row_content[5].ToString();
                     String process_duration = row_content[6].ToString();
                     String process_message = row_content[7].ToString();
                     String ingestion_status = row_content[8].ToString();
                     String ingestion_start_time = row_content[9].ToString();
                     String ingestion_end_time = row_content[10].ToString();
                     String ingestion_duration = row_content[11].ToString();
                     String transformation_status = row_content[12].ToString();
                     String transformation_start_time = row_content[13].ToString();
                     String transformation_end_time = row_content[14].ToString();
                     String transformation_duration = row_content[15].ToString();
                     String persistence_status = row_content[16].ToString();
                     String persistence_start_time = row_content[17].ToString();
                     String persistence_end_time = row_content[18].ToString();
                     String persistence_duration = row_content[19].ToString();
                     String output_status = row_content[20].ToString();
                     String output_start_time = row_content[21].ToString();
                     String output_end_time = row_content[22].ToString();
                     String output_duration = row_content[23].ToString();
                     String phase = row_content[24].ToString();
                     String entity = row_content[25].ToString();
                     String phase_entity_status = row_content[26].ToString();
                     String phase_entity_affected_rows = row_content[27].ToString();
                     String phase_entity_start_time = row_content[28].ToString();
                     String phase_entity_end_time = row_content[29].ToString();
                     String phase_entity_duration = row_content[30].ToString();
                     String phase_entity_message = row_content[31].ToString();

                     subject = "[" + customer.ToUpper() + "] [" + status + "]: " + process_name + " execution
    summary on " + computer_name.ToUpper();

                     summary_header_text = "<p> <b> Running Server: [" + computer_name.ToUpper() + "] </b> <
    /p>" +
                 "<p> <b> Customer: [" + customer.ToUpper() + "] </b> </p>" +
                 "<p> <b> Execution summary for process: " + process_name + " </b> </p>" +
                 "<p> Process description: " + process_description + "</p>" +
                 "<p> <u> Main summary </u> </p>" +
                 "<table cellpadding=\"5\" cellspacing=\"0\"> " +
                 "<tr bgcolor=\"DeepSkyBlue\"> " +
                 "<th style=\"border: 1px solid black; border-collapse: collapse\"> STATUS </th> " +
                 "<th style=\"border: 1px solid black; border-collapse: collapse\"> START TIME </th> " +
                 "<th style=\"border: 1px solid black; border-collapse: collapse\"> END TIME </th> " +
                 "<th style=\"border: 1px solid black; border-collapse: collapse\"> TOTAL DURATION </th> " +
                 "<th style=\"border: 1px solid black; border-collapse: collapse\"> MESSAGE </th> " +
                 "</tr> " +
                 "<tr> " +
                 (process_status == "SUCCEEDED" ? "<td style=\"border: 1px solid black; text-align: center;
    border-collapse: collapse; color:  green;\"> " : process_status == "WARNING" ? "<td style=\"border: 1px
    solid black; text-align: center; border-collapse: collapse; color:  orange;\"> " : "<td style=\"border: 1px
    solid black; text-align: center; border-collapse: collapse; color: red;\">") + "<b>" + process_status + " <
    /b> </td> " +
                 "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " +
    process_start_time + " </td> " +
                 "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " +
    process_end_time + " </td> " +
                 "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " +
    process_duration + " </td> " +
                 "<td style=\"border: 1px solid black; border-collapse: collapse\"> " + process_message + " <
    /td> " +
```

```
103                 "</tr> " +
104                 "</table>";
105 
106                   summary_text = "<p> <u> Phase summary </u> </p>" +
107                 "<table cellpadding=\"5\" cellspacing=\"0\"> " +
108                 "<tr bgcolor=\"DeepSkyBlue\"> " +
109                 "<th style=\"border: 1px solid black; border-collapse: collapse\"> PHASE </th> " +
110                 "<th style=\"border: 1px solid black; border-collapse: collapse\"> STATUS </th> " +
111                 "<th style=\"border: 1px solid black; border-collapse: collapse\"> START TIME </th> " +
112                 "<th style=\"border: 1px solid black; border-collapse: collapse\"> END TIME </th> " +
113                 "<th style=\"border: 1px solid black; border-collapse: collapse\"> TOTAL DURATION </th> " +
114                 "</tr> " +
115                 (ingestion_status == "N/A" ? "" : "<tr> " +
116                 "<td style=\"border: 1px solid black; border-collapse: collapse\"> INGESTION </td> " +
117                 (ingestion_status == "SUCCEEDED" ? "<td style=\"border: 1px solid black; text-align: center;
    border-collapse: collapse; color: green;\"> " : ingestion_status == "WARNING" ? "<td style=\"border: 1px
    solid black; text-align: center; border-collapse: collapse; color: orange;\"> " : "<td style=\"border: 1px
    solid black; text-align: center; border-collapse: collapse; color: red;\"> ") + "<b>" + ingestion_status + "
    </b> </td> " +
118                 "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " +
    ingestion_start_time + " </td> " +
119                 "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " +
    ingestion_end_time + " </td> " +
120                 "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " +
    ingestion_duration + " </td> " +
121                 "</tr> ") +
122                 (transformation_status == "N/A" ? "" : "<tr> " +
123                 "<td style=\"border: 1px solid black; border-collapse: collapse\"> TRANSFORMATION </td> " +
124                 (transformation_status == "SUCCEEDED" ? "<td style=\"border: 1px solid black; text-align:
    center; border-collapse: collapse; color: green;\"> " : transformation_status == "WARNING" ? "<td
    style=\"border: 1px solid black; text-align: center; border-collapse: collapse; color: orange;\"> " : "<td
    style=\"border: 1px solid black; text-align: center; border-collapse: collapse; color: red;\"> ") + "<b>" +
    transformation_status + " </b> </td> " +
125                 "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " +
    transformation_start_time + " </td> " +
126                 "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " +
    transformation_end_time + " </td> " +
127                 "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " +
    transformation_duration + " </td> " +
128                 "</tr> ") +
129                 (persistence_status == "N/A" ? "" : "<tr> " +
130                 "<td style=\"border: 1px solid black; border-collapse: collapse\"> PERSISTENCE </td> " +
131                 (persistence_status == "SUCCEEDED" ? "<td style=\"border: 1px solid black; text-align:
    center; border-collapse: collapse; color: green;\"> " : persistence_status == "WARNING" ? "<td
    style=\"border: 1px solid black; text-align: center; border-collapse: collapse; color: orange;\"> " : "<td
    style=\"border: 1px solid black; text-align: center; border-collapse: collapse; color: red;\"> ") + "<b>" +
    persistence_status + " </b> </td> " +
132                 "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " +
    persistence_start_time + " </td> " +
133                 "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " +
    persistence_end_time + " </td> " +
134                 "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " +
    persistence_duration + " </td> " +
135                 "</tr> ") +
136                 (output_status == "N/A" ? "" : "<tr> " +
137                 "<td style=\"border: 1px solid black; border-collapse: collapse\"> OUTPUT </td> " +
138                 (output_status == "SUCCEEDED" ? "<td style=\"border: 1px solid black; text-align: center;
    border-collapse: collapse; color: green;\"> " : output_status == "WARNING" ? "<td style=\"border: 1px
    solid black; text-align: center; border-collapse: collapse; color: orange;\"> " : "<td style=\"border: 1px
    solid black; text-align: center; border-collapse: collapse; color: red;\"> ") + "<b>" + output_status + " <
    /b> </td> " +
139                 "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " +
    output_start_time + " </td> " +
140                 "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " +
    output_end_time + " </td> " +
141                 "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " +
    output_duration + " </td> " +
142                 "</tr> ") +
143                 "</table>";
144 
145                   detail_text = detail_text + "<tr> " +
146                 "<td style=\"border: 1px solid black; border-collapse: collapse\"> " + phase + " </td> " +
147                 "<td style=\"border: 1px solid black; border-collapse: collapse\"> " + entity + " </td> " +
```

```
148                 (phase_entity_status == "SUCCEEDED" ? "<td style=\"border: 1px solid black; text-align:
    center; border-collapse: collapse; color: green;\"> " : phase_entity_status == "WARNING" ? "<td
    style=\"border: 1px solid black; text-align: center; border-collapse: collapse; color: orange;\"> " : "<td
    style=\"border: 1px solid black; text-align: center; border-collapse: collapse; color: red;\"> ") + "<b>" +
    phase_entity_status + " <b> </td> " +
149                 "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " +
    phase_entity_affected_rows + " </td> " +
150                 "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " +
    phase_entity_start_time + " </td> " +
151                 "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " +
    phase_entity_end_time + " </td> " +
152                 "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " +
    phase_entity_duration + " </td> " +
153                 "<td style=\"border: 1px solid black; border-collapse: collapse\"> " + phase_entity_message +
    " </td> " +
154                 "</tr> ";
155             }
156
157             detail_text = detail_text + "</table>";
158             String body = summary_header_text + summary_text + detail_text;
159
160             MailMessage mail = new MailMessage(from, to.Replace(";", ","), subject, body);
161             mail.IsBodyHtml = true;
162             SmtpClient client = new SmtpClient(smtp_host);
163             client.Credentials = new System.Net.NetworkCredential(from, "Usuario2011");
164             client.Send(mail);
165
166             Dts.TaskResult = (int)ScriptResults.Success;
167         }
168         catch (Exception e)
169         {
170             Dts.Events.FireError(0, "Report Mail Script", e.Message + "\r" + e.StackTrace, String.Empty, 0);
171             Dts.TaskResult = (int)ScriptResults.Failure;
172         }
173
174     }
175       enum ScriptResults
176       {
177           Success = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Success,
178           Failure = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Failure
179       };
180       #endregion
181   }
182 }
```

5.4. Proceso de ingesta

Ingest_process es el proceso de la solución IO encargado de recoger la información procedente de diferentes orígenes, ya sean locales o remotos, almacenándola en tablas de la BBDD IO en el schema *input*. Se muestra en las *Figuras 5.17-5.18*.

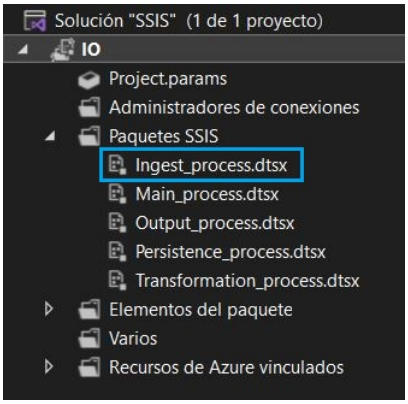


Figura 5.17: Paquete Ingest_process de la solución SSIS

El paquete *Ingest_process* comprueba el id del proceso en ejecución en *IO.config.master_process_type*, si la columna *ingestion_flag* está activa se ejecutará el contenedor de secuencia de ingesta.

Figura 5.18: Contenedor de secuencia de ingesta

Este contenedor inserta un registro de inicio en la tabla de log maestro *IO.monitor.master_process_log* para el proceso de ingesta, ejecuta el paquete *Ingest_process*, inserta un registro de final de proceso en la misma tabla y calcula el siguiente step a ejecutar.

5.4.0.1. Detalle del proceso

En la *Figura 5.19* se muestra el flujo de control del paquete *Ingest_process*.

Durante la primera parte del proceso, se recupera la información necesaria de las tablas de configuración para la ejecución del proceso mediante la consulta almacenada en la variable *sql_get_main_config*. Ver *Figuras 5.20-5.21*.

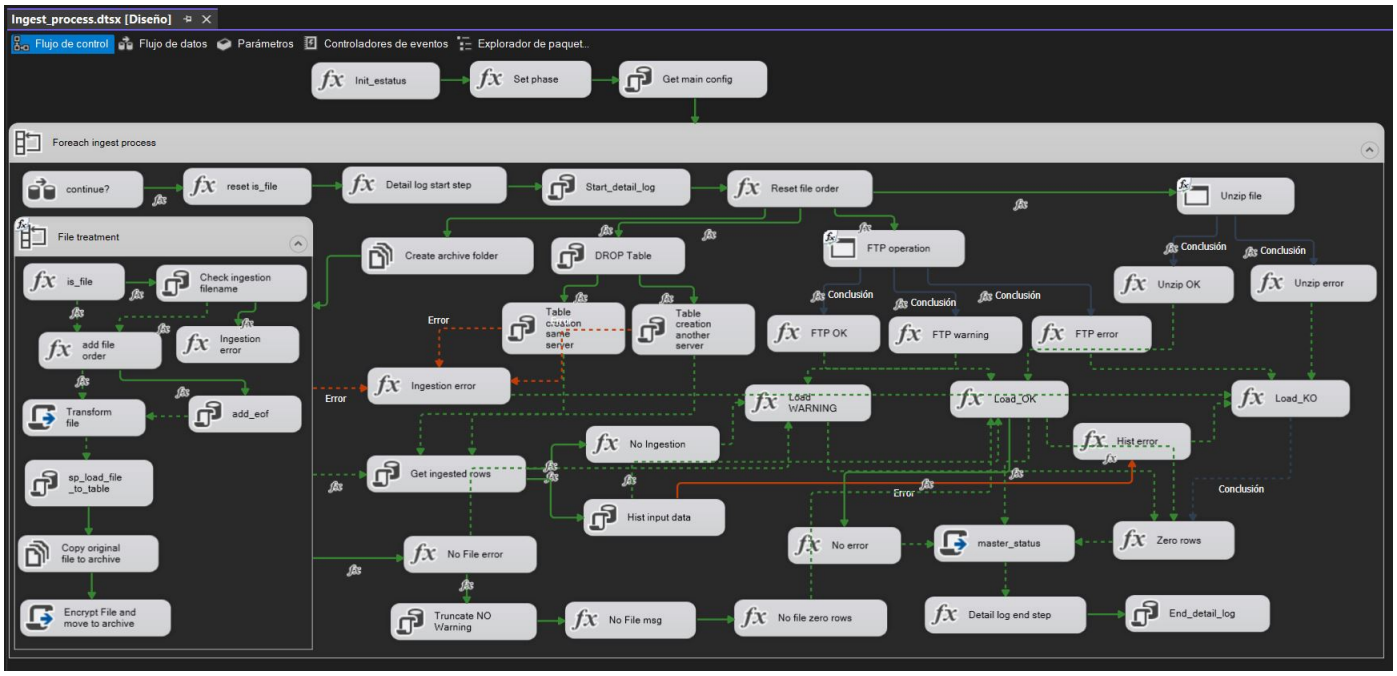


Figura 5.19: Flujo de control del paquete Ingest_process

Figura 5.20: Bloque inicial del proceso de ingesta

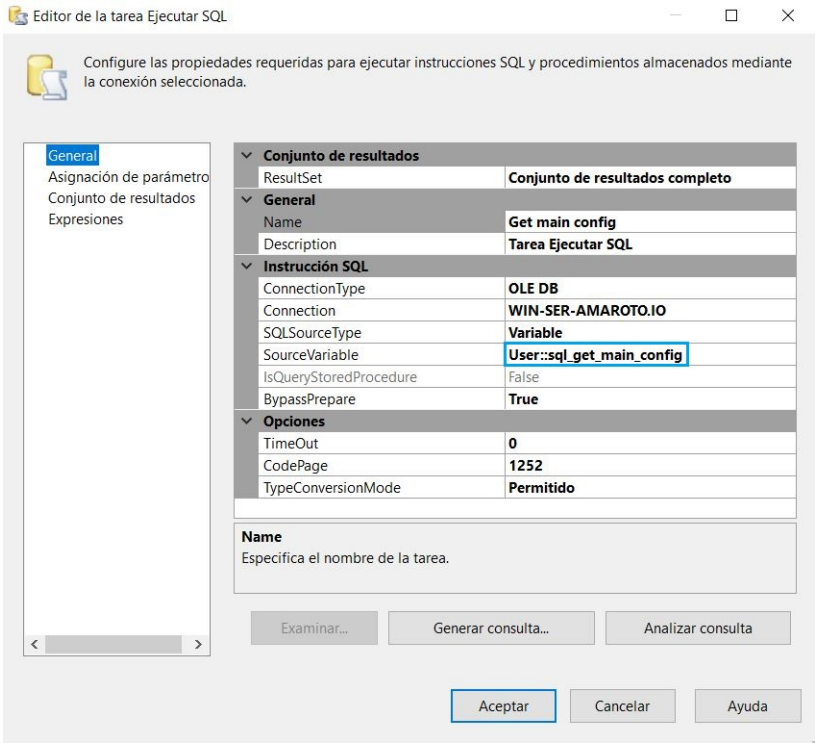


Figura 5.21: Recupera la información de la variable sql_get_main_config

Los resultados obtenidos en la consulta se almacenan en la variable de tipo Objeto *obj_processes* (*Figura 5.22*). Posteriormente se lanza el bucle *Foreach ingest process* el cual va iterando por

cada registro obtenido de la consulta, a la vez que recupera y almacena los datos en diferentes variables, a continuación la *Tabla 5.1* detalla dicho mapeo.

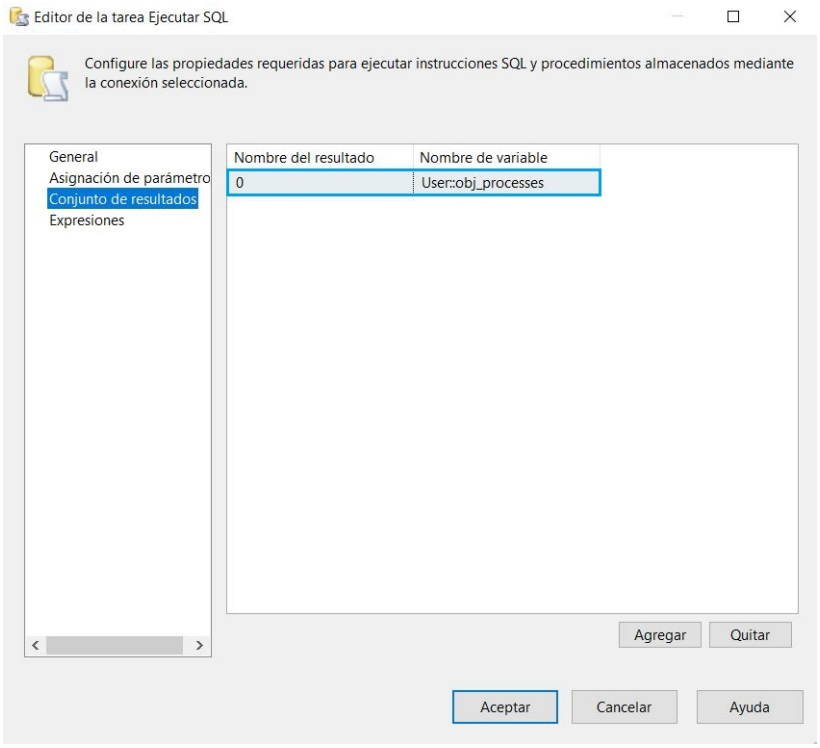


Figura 5.22: Almacena el resultado en la variable obj_processes

Tabla 5.1: Detalle de la asignación de variables en Foreach ingest process

Id	Tabla BBDD	Columna BBDD	Variable SSIS
0	IO.config.detail_process_type	source_type	v_source_type
1	IO.config.detail_process_type	source_file_folder_path	v_source_file_folder_path
2	IO.config.detail_process_type	source_file_pattern	v_source_file_pattern
3	IO.config.detail_process_type	skip_initial_rows	v_skip_initial_rows
4	IO.config.detail_process_type	source_table_query	v_source_table_query
5	IO.config.detail_process_type	destination_table	v_destination_table
6	IO.config.detail_process_type	source_file_template_path	v_source_file_template_path
7	IO.config.detail_process_type	name	v_process_name
8	IO.config.detail_process_type	zip_folder_path	v_zip_folder
9	IO.config.detail_process_type	zip_file_path	v_zip_file
10	IO.config.detail_process_type	ftp_server	v_ftp_server
11	IO.config.detail_process_type	ftp_port	v_ftp_port
12	IO.config.detail_process_type	ftp_user	v_ftp_user
13	IO.config.detail_process_type	ftp_pass	v_ftp_pass
14	IO.config.detail_process_type	ftp_operation	v_ftp_operation
15	IO.config.detail_process_type	ftp_folder	v_ftp_folder
16	IO.config.detail_process_type	ftp_file_pattern	v_ftp_file_pattern
17	IO.config.detail_process_type	ftp_local_folder	v_ftp_local_folder
18	IO.config.detail_process_type	ftp_local_file_pattern	v_ftp_local_file_pattern
19	IO.config.detail_process_type	skip_final_rows	v_skip_final_rows
20	IO.config.detail_process_type	add_eof	v_add_eof
21	IO.config.detail_process_type	ftp_private_key_path	v_ftp_private_key_path
22	IO.config.detail_process_type	ftp_ssh_host_key_fingerprint	v_ftp_ssh_host_key_fingerprint
23	IO.config.detail_process_type	ftp_private_key_passphrase	v_ftp_private_key_passphrase
24	IO.config.detail_process_type	source_connection_server	v_source_connection_server
25	IO.config.detail_process_type	source_connection_database	v_source_connection_database
26	IO.config.detail_process_type	source_connection_user	v_source_connection_user
27	IO.config.detail_process_type	source_connection_password	v_source_connection_password
28	IO.config.detail_process_type	customer	v_customer
29	IO.config.detail_process_type	check_ingestion_filename	v_check_ingestion_filename
30	IO.config.detail_process_type	stop_on_warning	v_stop_on_warning
31	IO.config.detail_process_type	ingestion_encoding_file	v_ingestion_encoding_file
32	IO.config.detail_process_type	zip_rar_pass	v_zip_rar_pass

Tras la asignación de variables se realiza el bloque previo para cada ingesta o iteración, ejecutando varios componentes de comprobación y parametrización como muestra la *Figura 5.23*.

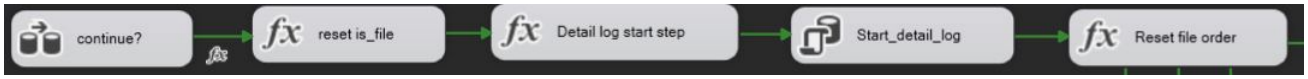


Figura 5.23: Reinicio de variables e inicio del log del proceso de ingesta

- reset_is_file: Reinicia el valor de la variable *v_is_file* la cual verifica la existencia de archivos cuando se realizan ingestas de este tipo.
- Detail log start step y Start_detail_log: Se inserta un registro de inicio de proceso en la tabla de log *IO.monitor.detail_process_log*.
- Reset file order: Reinicia el valor de la variable *v_file_order* la cual controla la cantidad de archivos cargados cuando se realizan ingestas de este tipo.

A continuación, el proceso se divide en diferentes caminos, pudiéndose ejecutar cada uno de ellos en función de la configuración del valor de la variable *v_source_type*, la cual puede contener los siguientes valores.

- v_source_type == 'BBDD': Se realiza la ingesta a través de la ejecución de una consulta desde un servidor de BBDD. Se muestra el bloque en la *Figura 5.24*.

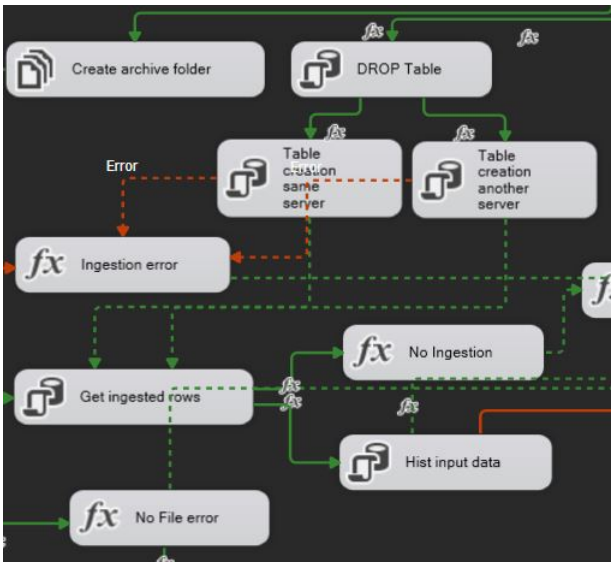


Figura 5.24: Bloque BBDD del proceso de ingesta

- Se realiza el borrado de la tabla de destino para evitar posibles errores durante el proceso. Para ello se ejecuta la consulta almacenada en la variable *v_sql_drop_table* como se aprecia en la *Figura 5.25*.

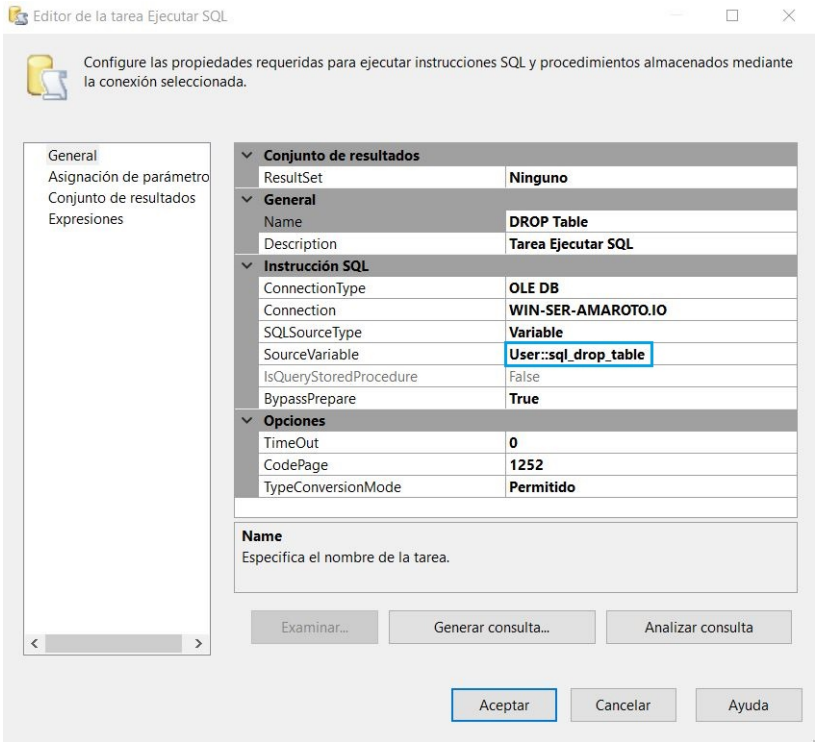


Figura 5.25: Tarea SQL DROP Table del proceso de ingesta

- En función del valor de la variable *v_source_connection_server*, se obtiene la tabla de nuestro SQL Server local si la variable no contiene valor (*Figura 5.26*), o por el contrario de otro servidor si contiene información (*Figura 5.27*). La obtención de la tabla se puede realizar mediante una consulta embebida, o a través de un procedimiento almacenado.

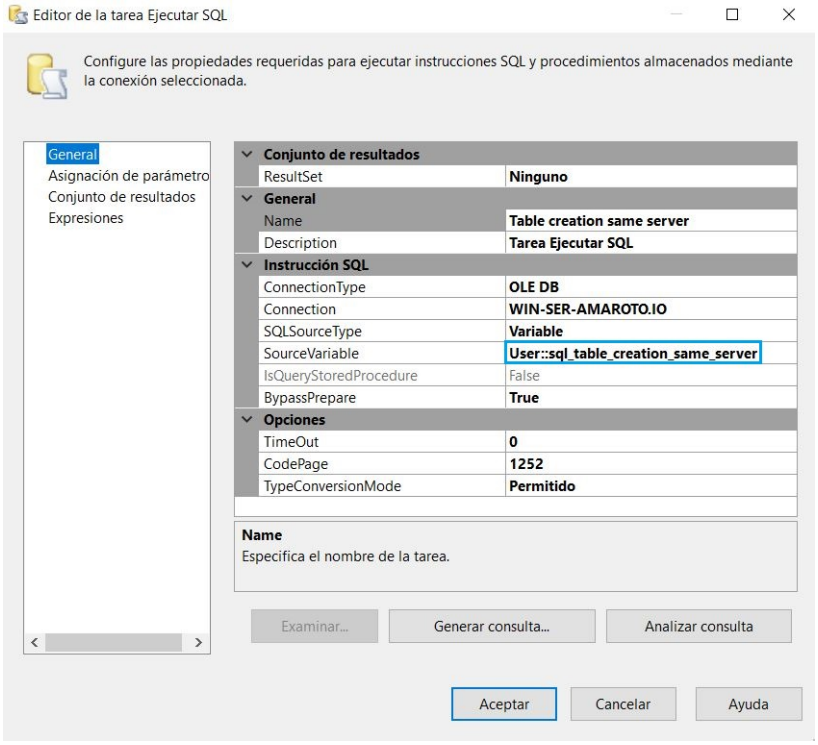


Figura 5.26: Creación de tabla con origen local del proceso de ingesta

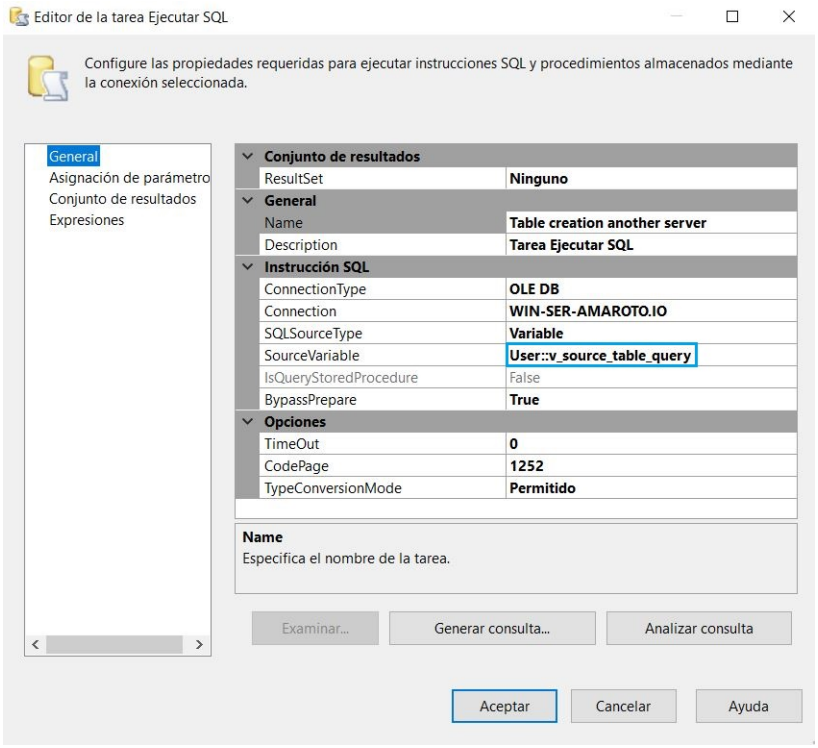


Figura 5.27: Creación de tabla con origen remoto del proceso de ingesta

- Si nuestro origen es un servidor remoto, necesitamos asignar valores a los parámetros que se muestran en la *Figura 5.28*.

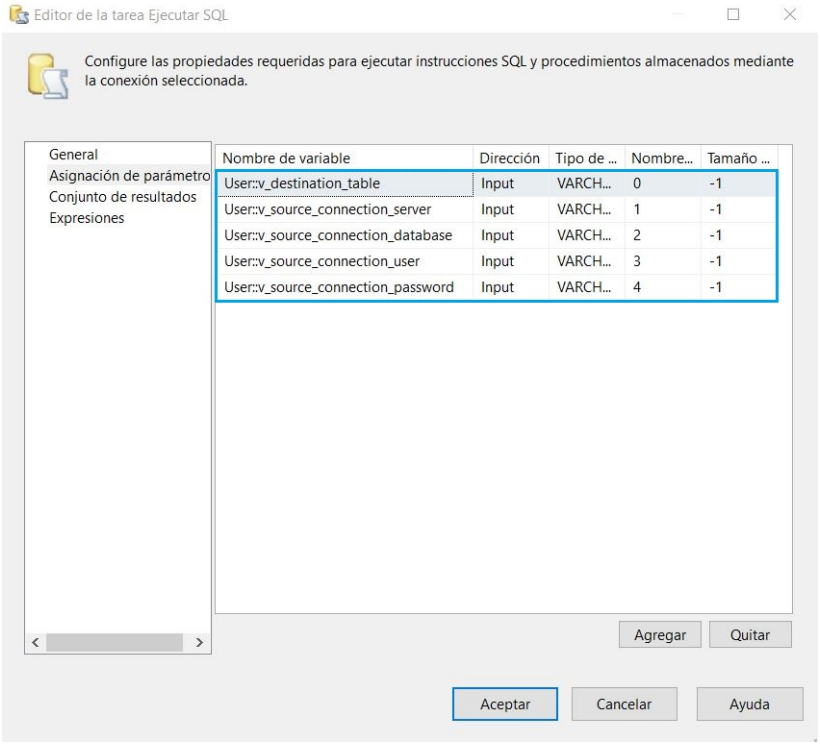


Figura 5.28: Asignación de parámetros para acceder a la tabla remota del proceso de ingesta

- Dependiendo del resultado del script se ejecuta uno de los siguientes componentes de estado.

  - Get ingested rows: Se ha creado la tabla en la BBDD IO, por lo que esta tarea comprueba si adicionalmente se han insertado registros ejecutando la variable *sql_get_ingested_rows*. En función del resultado almacenado en *v_p_quantity_rows* el flujo deriva en el componente *Load_OK* o *Load_WARNING* si tiene o no registros respectivamente.

  - Ingestion error: Se ha producido un error durante la creación de la tabla en la BBDD IO. Deriva en el componente *Load_KO* forzando el error del paquete.

- v_source_type == 'FILE': Se realiza la ingesta a través de la lectura de un archivo. siendo los formatos disponibles los siguientes.

  - CSV
  - DAT
  - TXT
  - XLS
  - XLSX

- Creamos una carpeta para archivar el fichero tras cargarlo en la BBDD IO, donde la ruta de creación tendrá el formato *'\\[Servidor]\IO\Archive\[Customer] \[Proceso]'*. Ver tarea en la *Figura 5.29*.

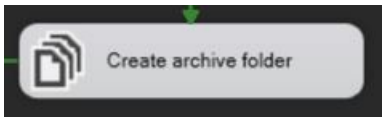


Figura 5.29: Tarea Sistema de archivos para crear el directorio archive del proceso

- A continuación la *Figura 5.30* ejecuta el bucle *Foreach File treatment* para cada uno de los archivos de la ruta de origen que cumplan el formato de nomenclatura que hayamos indicado.

- Se actualiza el valor de la variable *v_is_file* para contrastar que exista al menos un archivo en la ruta y contrastar si se ha podido leerlo.

- En caso de que la variable *v_check_ingestion_filename* sea igual a uno, La tarea *Check ingestion filename* (*Figura 5.31*) comprueba si se ha cargado previamente un archivo con el mismo nombre. Para ello se ejecuta el script SQL de la variable *sql_check_ingestion_filename*.

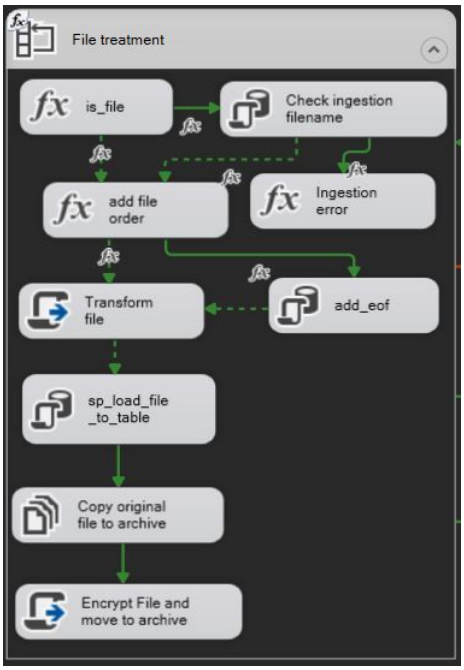


Figura 5.30: Bloque FILE del proceso de ingesta

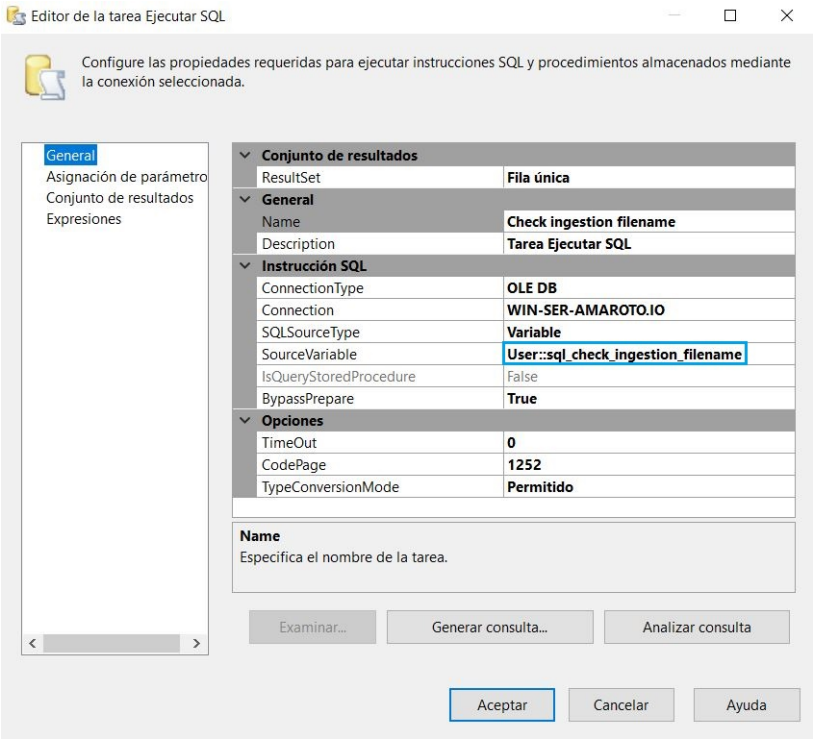


Figura 5.31: Tarea SQL Check ingestion filename

- En caso de que se esté intentando cargar un fichero duplicado, se fuerza el error. Para más información ver el apartado *Ejecución de proceso y revisión de logs* B del *Manual de usuario* B.

- A continuación se modifica el valor de la variable *v_file_order* incrementando su valor en uno, de esta forma el proceso puede crear la tabla de destino y añadir su contenido.

- Se lanza el script C# *Transform file* para realizar las modificaciones mostradas en el *Código 5.45*.

Código 5.45: Ingest - Transform file.cs

```
#region Namespaces
using System.IO;
using System.Runtime.InteropServices;
using System.Text;
using Microsoft.Office.Interop.Excel;
#endregion

namespace ST_58afe4e3bd604440b3cf7066b3bc6ce0
{
   [Microsoft.SqlServer.Dts.Tasks.ScriptTask.SSISScriptTaskEntryPointAttribute]
   public partial class ScriptMain : Microsoft.SqlServer.Dts.Tasks.ScriptTask.VSTARTScriptObjectModelBase
   {
      public void Main()
      {
            var source = Dts.Variables["v_path_filename"].Value.ToString();
            var file_extension = Dts.Variables["v_file_extension"].Value.ToString().ToUpper();
            var ingestion_encoding_file =
    Dts.Variables["v_ingestion_encoding_file"].Value.ToString().ToUpper();

            if (file_extension == "XLS" || file_extension == "XLSX")
            {
                Microsoft.Office.Interop.Excel.Application excel = new
    Microsoft.Office.Interop.Excel.Application();

                Microsoft.Office.Interop.Excel.Workbooks wkbks = excel.Workbooks;
                Microsoft.Office.Interop.Excel.Workbook wkbk = wkbks.Open(source);

                excel.Visible = false;
                excel.DisplayAlerts = false;

                Worksheet worksheet = wkbk.Worksheets.get_Item(1);
                if (worksheet.Name != "Hoja1")
                {
                    worksheet.Name = "Hoja1";
                }

                worksheet.Cells.NumberFormat = "@";

                if (wkbk != null) {
                    try
                    {
                        wkbk.Save();
                        wkbk.Close(true, source);
                    }
                    catch
                    {
                    }

                    Marshal.FinalReleaseComObject(wkbk);
                    wkbk = null;
                }
            }
            else
            {
                if (ingestion_encoding_file != "")
                {
                    Encoding destination_encode = null;

                    if (ingestion_encoding_file == "ASCII")
                    {
                        destination_encode = Encoding.ASCII;
                    } else if (ingestion_encoding_file == "UTF8")
                    {
                        destination_encode = Encoding.UTF8;
                    }

                    StreamReader sourceStream = new StreamReader(source);
                    string fileContent = sourceStream.ReadToEnd();
```

```
67                      sourceStream.Close();
68
69                      StreamWriter destinationStream = new StreamWriter(source, false, destination_encode);
70                      destinationStream.Write(fileContent);
71                      destinationStream.Close();
72                  }
73              }
74
75
76              Dts.TaskResult = (int)ScriptResults.Success;
77          }
78          enum ScriptResults
79          {
80              Success = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Success,
81              Failure = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Failure
82          };
83      }
84  }
```

- Una vez se han realizado las transformaciones necesarias para el tratamiento del archivo, se lanza la tarea *sp_load_file_to_table* (*Figuras 5.32-5.33*) la cual ejecuta el procedimiento almacenado *IO.input.sp_load_file_to_table*, el cual hemos detallado en *Ingesta de archivos en BBDD* 5.2.2 de este capítulo.

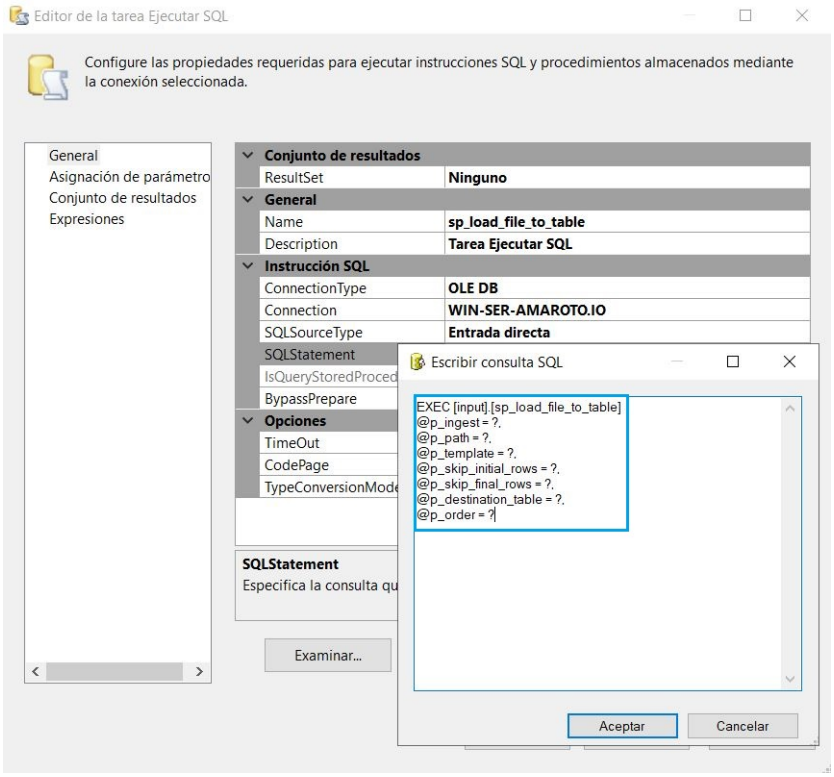


Figura 5.32: Tarea SQL sp_load_file_to_table

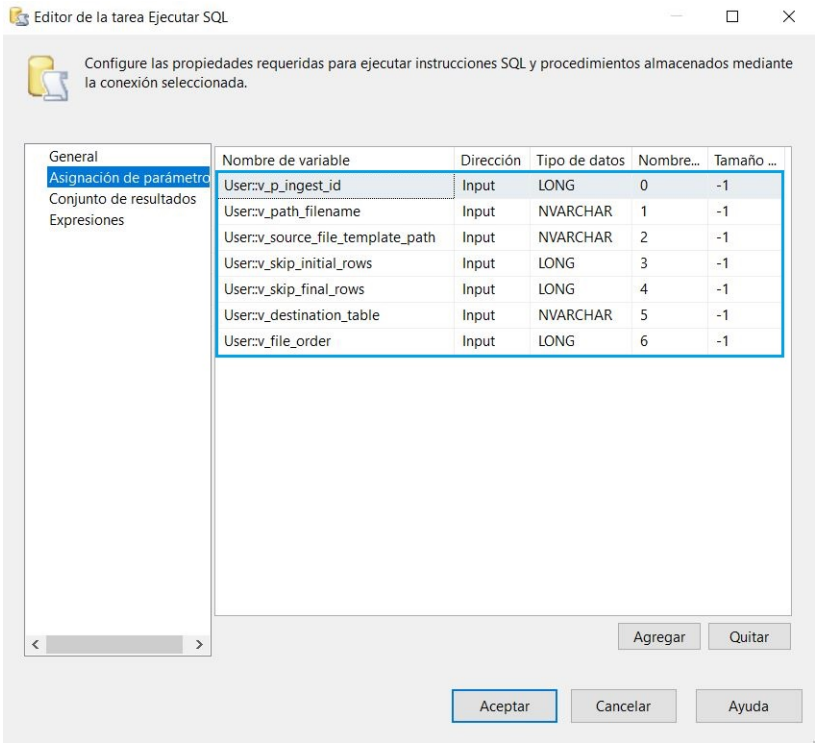


Figura 5.33: Asignación de parámetros en IO.input.sp_load_file_to_table

- Tras la ejecución del procedimiento almacenado se procede a archivar el archivo tratado, copiándolo a la ruta creada previamente como muestra la *Figura 5.34*.

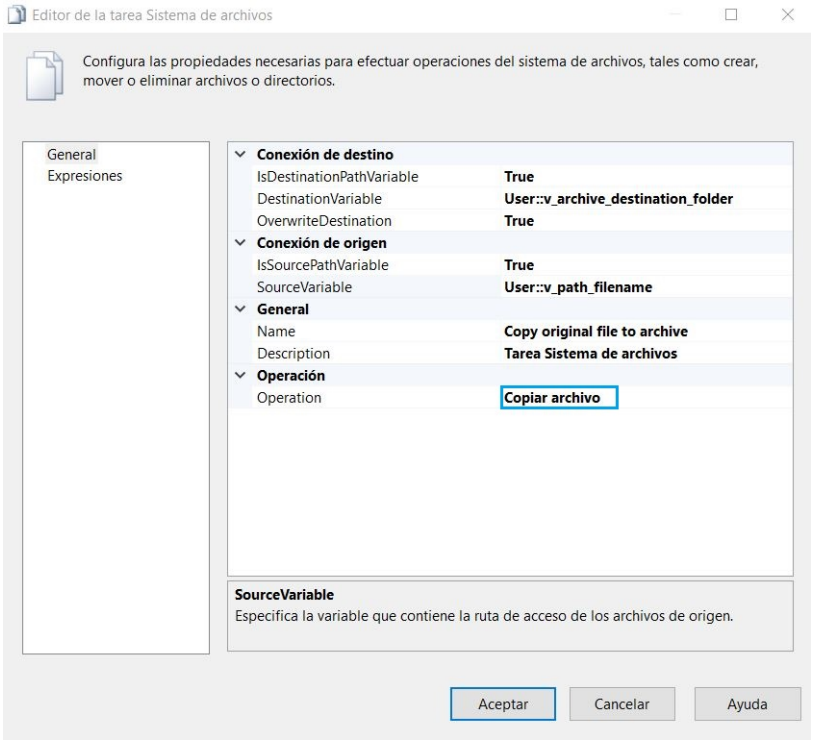


Figura 5.34: Tarea Sistema de archivos para archivar el fichero tratado

- Dependiendo del resultado del tratamiento del archivo se ejecuta uno de los siguientes componentes de estado.

- Get ingested rows: Se ha creado la tabla en la BBDD IO, por lo que esta tarea comprueba si adicionalmente se han insertado registros ejecutando la variable *sql_get_ingested_rows*. En función del resultado almacenado en *v_p_quantity_rows* el flujo deriva en el componente *Load_OK* o *Load_WARNING* si tiene o no registros respectivamente.

- Ingestion error: Se ha producido un error durante la creación de la tabla en la BBDD IO. Deriva en el componente *Load_KO* forzando el error del paquete.

- No File error: No hay ningún archivo en la carpeta de lectura. Deriva en el componente *Load_WARNING*, revisando previamente el contenido de la variable *v_stop_on_warning*, la cual trunca la tabla de destino y evita usar datos de cargas anteriores, se puede ver el flujo en la *Figura 5.35*.

Figura 5.35: Rama No File error del proceso de ingesta

- v_source_type == 'FTP' —— v_source_type == 'SFTP': Descarga de un archivo desde un servidor FTP/SFTP como muestra el bloque en la *Figura 5.36*.

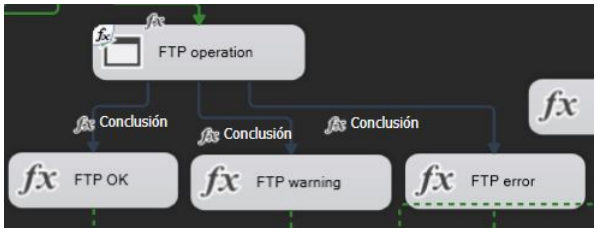


Figura 5.36: Bloque FTP del proceso de ingesta

- La tarea *FTP operation* lanza el script en PowerShell *ftp_sftp*, el cual hemos detallado en *Carga y descarga de archivos por FTP/SFTP* 5.2.3 de este capítulo. Dependiendo del resultado del script se ejecuta uno de los siguientes componentes de estado.

  - FTP OK: Descarga del archivo o archivos efectuada correctamente. Deriva en el componente *Load_OK*.

  - FTP warning: Descarga del archivo o archivos efectuada parcialmente. Deriva en el componente *Load WARNING*.

  - FTP error: Se ha producido un error durante el proceso de descarga. Deriva en el componente *Load_KO* forzando el error del paquete.

- v_source_type == 'UNZIP': Descompresión de un archivo ZIP/RAR con o sin protección por contraseña (*Figura 5.37*).

- La tarea *Unzip file* lanza el script en PowerShell *zip_unzip_file*, el cual hemos detallado en

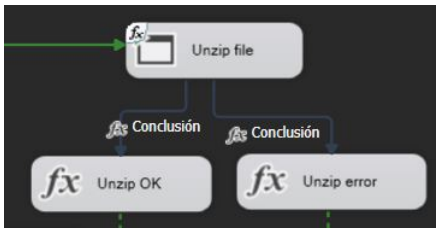


Figura 5.37: Bloque UNZIP del proceso de ingesta

Compresión y descompresión de archivos 5.3 de este capítulo. Dependiendo del resultado del script se ejecuta uno de los siguientes componentes de estado.

- Unzip OK: Descompresión del archivo comprimido efectuada correctamente. Deriva en el componente *Load_OK*.
- Unzip error: Se ha producido un error durante la descompresión del archivo. Deriva en el componente *Load_KO* forzando el error del paquete.

Una vez finaliza cualquiera de estos caminos, se inserta el log de la iteración en la tabla *IO.monitor.detail_process_log* con el detalle del proceso como muestra la *Figura 5.38*, con la excepción de que se produzca un error, fallando en este caso el proceso de manera controlada.

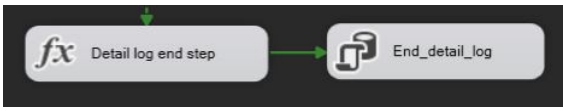


Figura 5.38: Inserta el registro final en el log del proceso de ingesta

Al finalizar todas las iteraciones del bucle *Foreach ingest process*, termina la ejecución del proceso de ingesta, recuperando *Main_process* el control de la ejecución.

5.5. Proceso de transformación

Transformation_process es el proceso de la solución IO encargado de adaptar la información ingestada al modelo de datos de destino (*Figura 5.39*). Para ello, el proceso ejecuta una serie de procedimientos almacenados que generan una nueva delivery, además de un conjunto de transformaciones, almacenando los datos en tablas bajo el schema *load* en la BBDD IO.

El paquete *Main_process* comprueba el id del proceso en ejecución en *IO.config.master_process_type*, si la columna *transformation_flag* está activa se ejecutará el contenedor de secuencia de transformación (*Figura 5.40*).

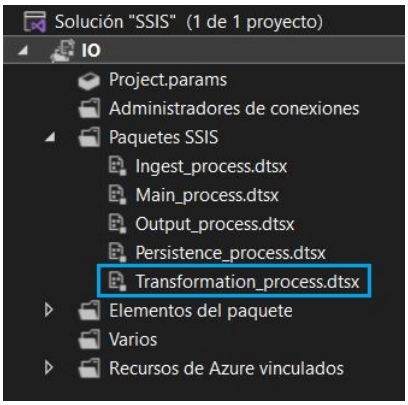


Figura 5.39: Paquete Transformation_process de la solución SSIS

Figura 5.40: Contenedor de secuencia de transformación

Este contenedor inserta un registro de inicio en la tabla de log maestro *IO.monitor.master_process_log* para el proceso de transformación, ejecuta el paquete *Transformation_process*, inserta un registro de final de proceso en la misma tabla y calcula el siguiente step a ejecutar.

5.5.0.1. Detalle del proceso

En la *Figura 5.41* se muestra el flujo de control del paquete *Transformation_process*.

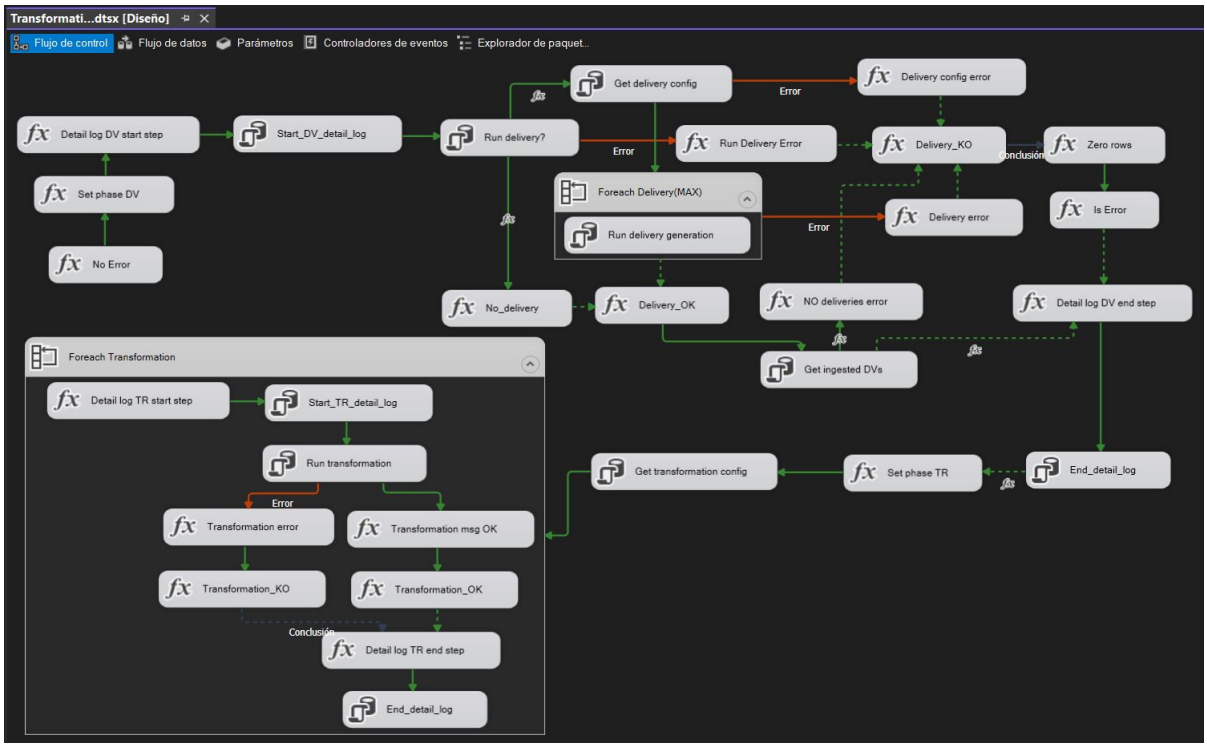


Figura 5.41: Flujo de control del paquete Transformation_process

Este proceso está dividido en dos fases, por un lado una primera fase opcional para crear una nueva delivery, y por otro lado la ejecución de todas las transformaciones necesarias que hayan sido configuradas para el proceso a ejecutar. Para más información ver *Procedimientos y tablas de carga* B del *Manual de usuario* B.

Durante la primera parte del proceso, se verifica la necesidad de comprobar la existencia o creación de una nueva delivery mediante la ejecución de la query almacenada en la variable *sql_exists_delivery* (*Figuras 5.42-5.43*).

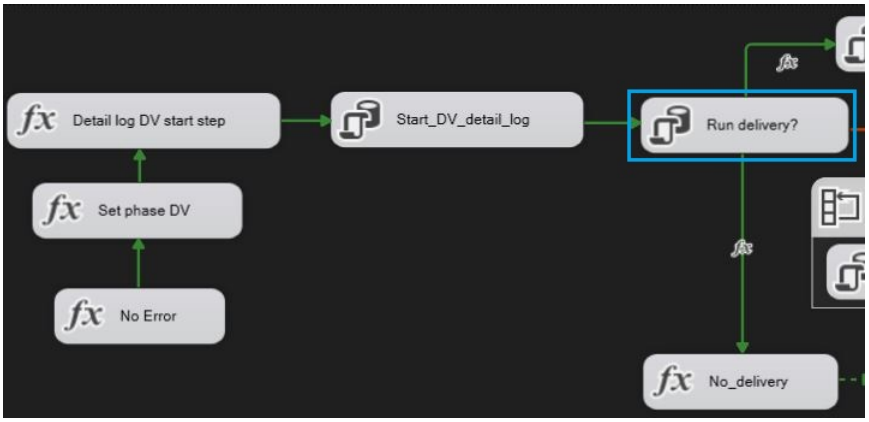


Figura 5.42: Comprueba la existencia de delivery

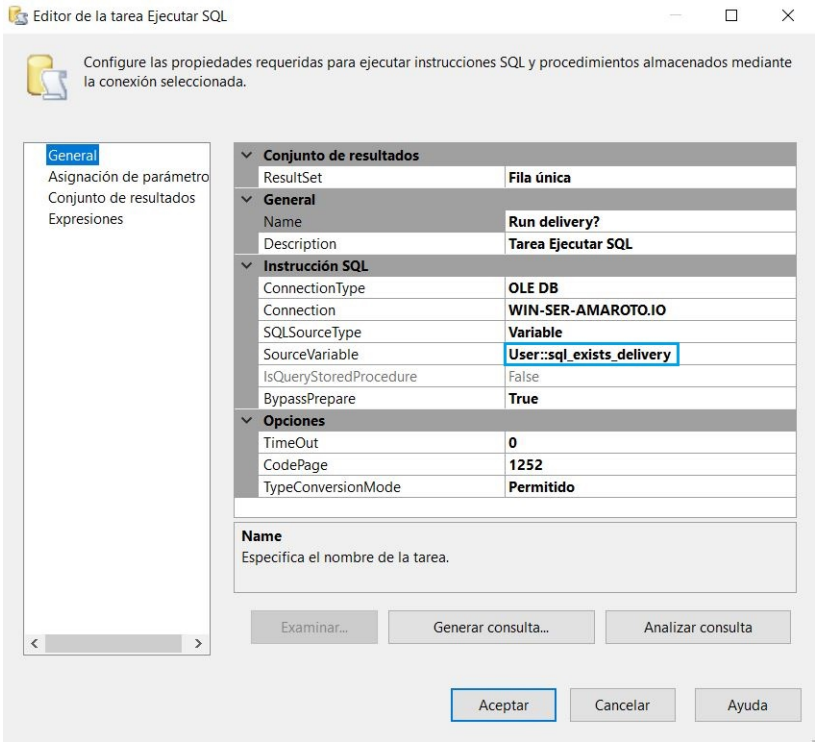


Figura 5.43: Recupera la información de la variable sql_exists_delivery

Los resultados obtenidos se almacenan en la variable de tipo entero *v_is_delivery*, la cual contiene la cantidad de registros de configuración de la fase *DELIVERY* para el identificador del proceso en ejecución. Dependiendo de su valor el proceso se comporta como sigue.

- *v_is_delivery* genera un error de ejecución.

    - Se asignan valores neutros a las variables de control relacionadas con la generación de delivery, y se inserta un registro de error en *IO.monitor.detail_process_log* con el detalle como muestra la *Figura 5.44*.

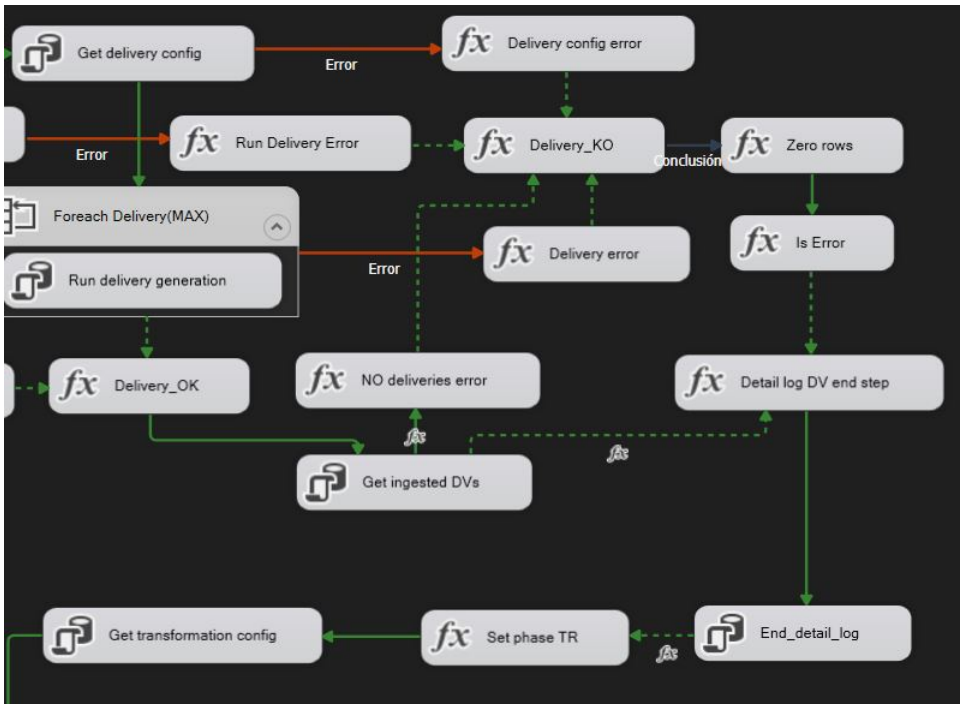


Figura 5.44: Asignación de valores e inserción de error en el log

- *v_is_delivery* = 1.

  - Continuamos con la ejecución del proceso de creación de delivery recuperando su información a partir de la tabla de configuración *IO.config.detail_process_type* (*Figura 5.45*), la consulta a ejecutar se encuentra en la variable SSIS *sql_get_delivery_config* (*Figura 5.46*).

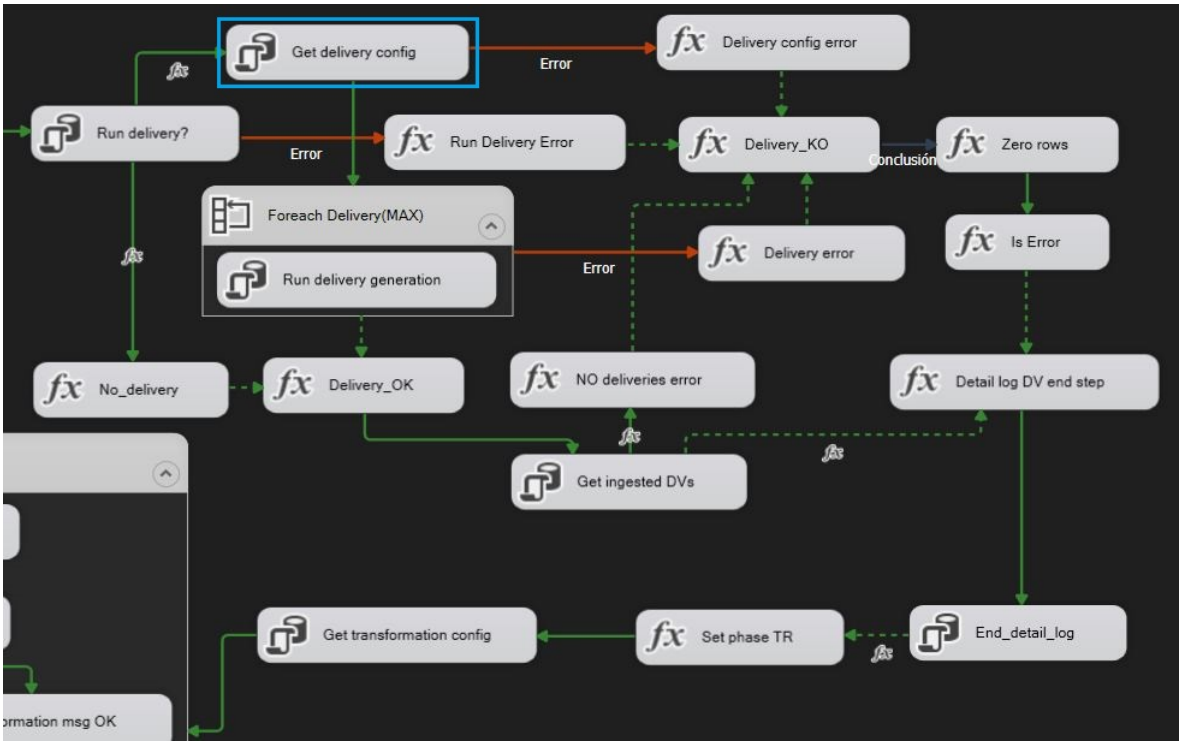


Figura 5.45: Tarea SQL Get_delivery_config

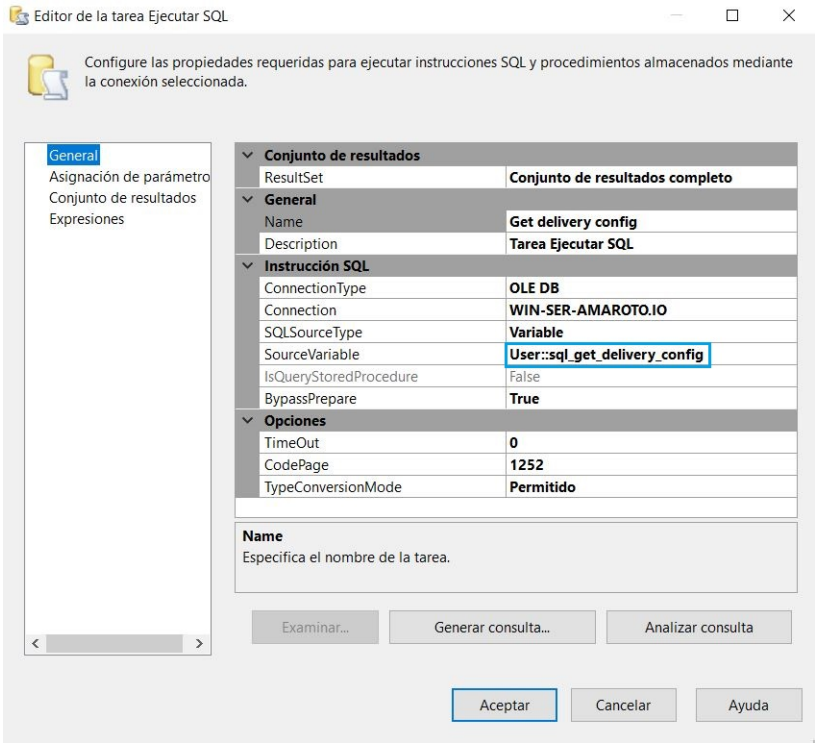


Figura 5.46: Recupera la información de la variable sql_get_delivery_config

- Almacenamos a su vez los resultados en la variable de tipo objeto *obj_deliveries* como muestra la *Figura 5.47*.

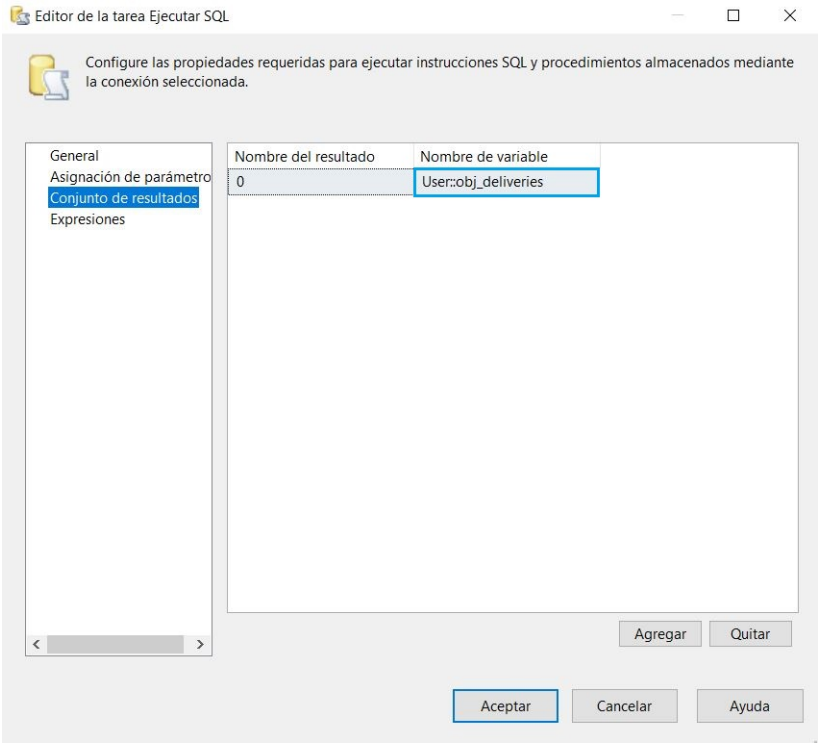


Figura 5.47: Almacena el resultado en la variable obj_deliveries

- *obj_deliveries* contiene la consulta a ejecutar para configurar una nueva delivery, el

contenido del script se encontrará en un procedimiento almacenado en la BBDD IO, el cual comprobará la existencia de una delivery abierta (columna *registration_date* con valor *NULL*) en la tabla *IO.load.delivery*.

- En caso de haber una delivery abierta para el proceso en ejecución se devolverá *ERROR*, debido a que no se puede ejecutar un proceso con una delivery sin finalizar. Ver *Ejecución de proceso y revisión de logs* B del *Manual de usuario* B para más información.

- En caso contrario, se creará un nuevo registro en la tabla *IO.load.delivery* (*Figura 5.48*) para el id del proceso en ejecución y se continuará con el resto de steps.

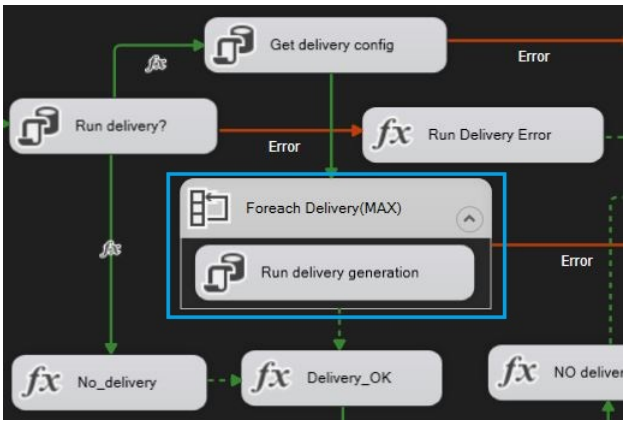


Figura 5.48: Tarea SQL Run delivery generation

- Una vez ejecutado el script de creación de delivery, se comprueba que se ha creado correctamente (*Figura 5.49*), se inserta un registro de actualización en el log, y continuamos con el proceso de transformación.

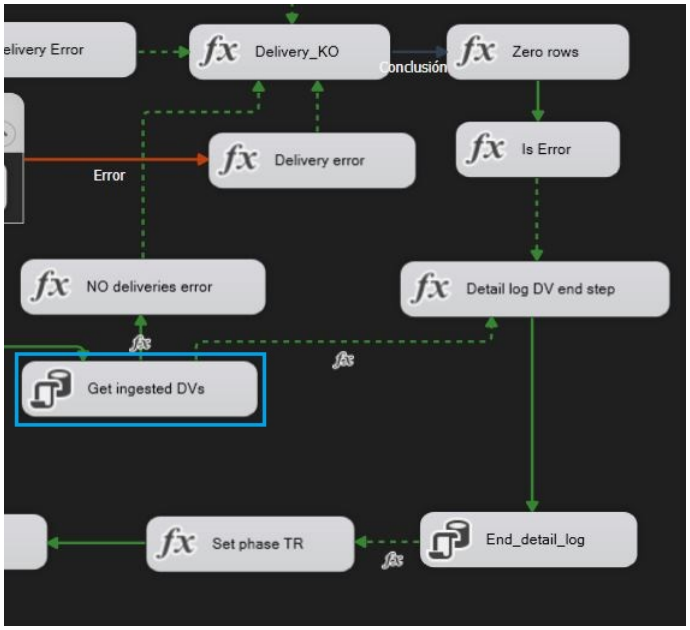


Figura 5.49: Tarea SQL Get ingested DVs

- En caso contrario, se genera un error forzando el fallo en el proceso, y se inserta un registro con esta información en el log.

- v_is_delivery = 0.

  - No hay configuración de generación de delivery, por lo que el proceso pasa directamente al apartado de transformación.

  - Se recupera la configuración de cada una de las transformaciones asociadas al identificador de proceso en ejecución, mediante la ejecución de la consulta de la variable SSIS *sql_get_transformation_configs* (*Figuras 5.50-5.51*).

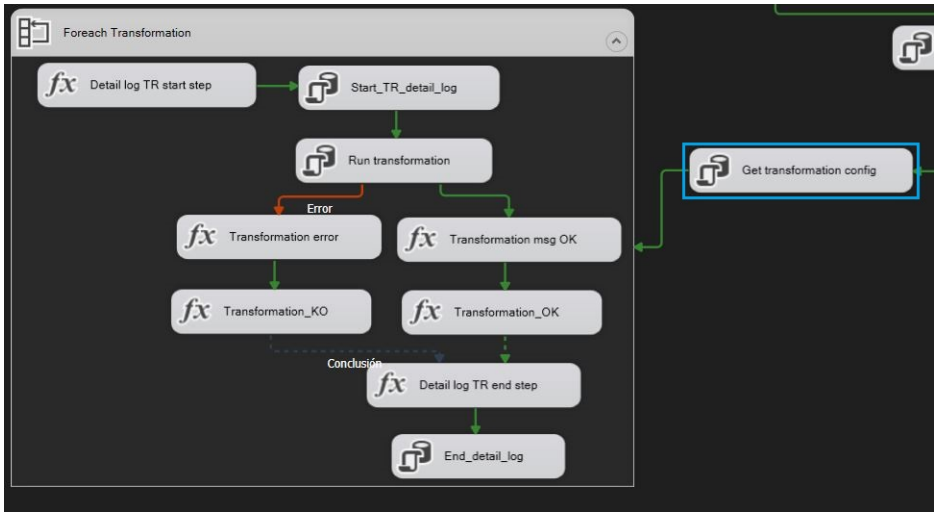


Figura 5.50: Tarea SQL Get transformations config y bloque Foreach Transformation

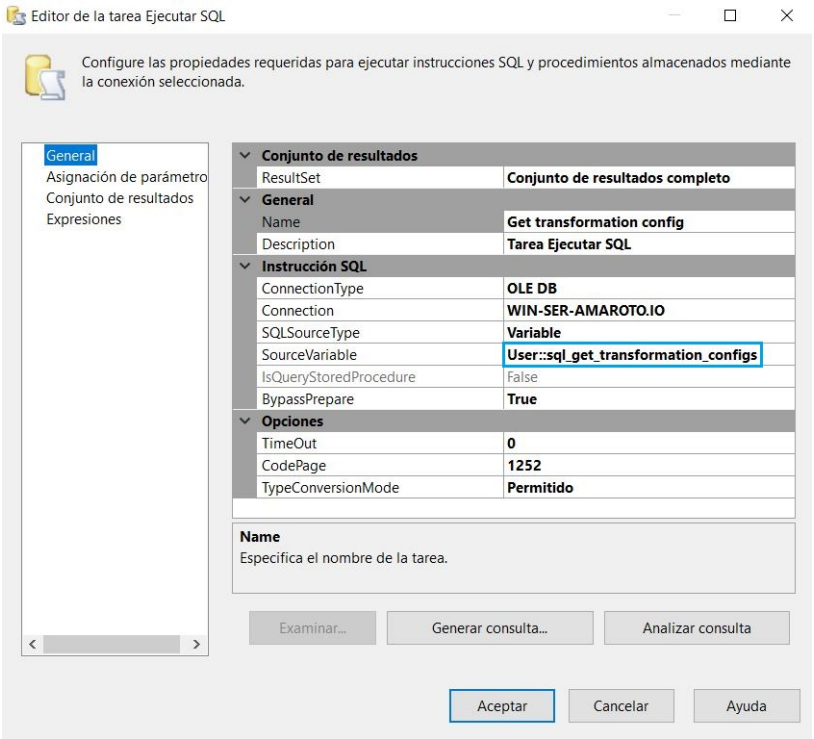


Figura 5.51: Recupera la información de la variable sql_get_transformation_configs

- Como muestra la *Figura 5.52* los resultados son almacenados en la variable de tipo objeto *obj_transformations*.

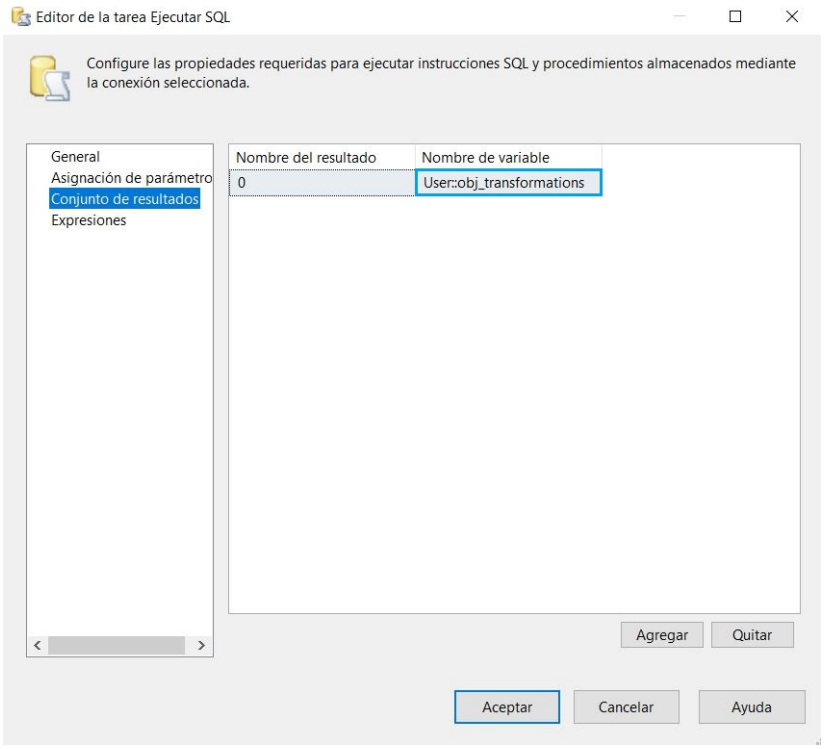


Figura 5.52: Almacena el resultado en la variable obj_transformations

- A continuación, se recuperan en el bucle foreach las variables que se muestran en la *Tabla 5.2* y se realizan las siguientes tareas.

Tabla 5.2: Detalle de la asignación de variables en Foreach transformation

Id	**Tabla BBDD**	**Columna BBDD**	**Variable SSIS**
0	IO.config.detail_process_type	source_table_query	v_sql_transformation_query
1	IO.config.detail_process_type	entity	v_p_entity

- Inicialización del log del proceso de transformación (*Figura 5.53*).

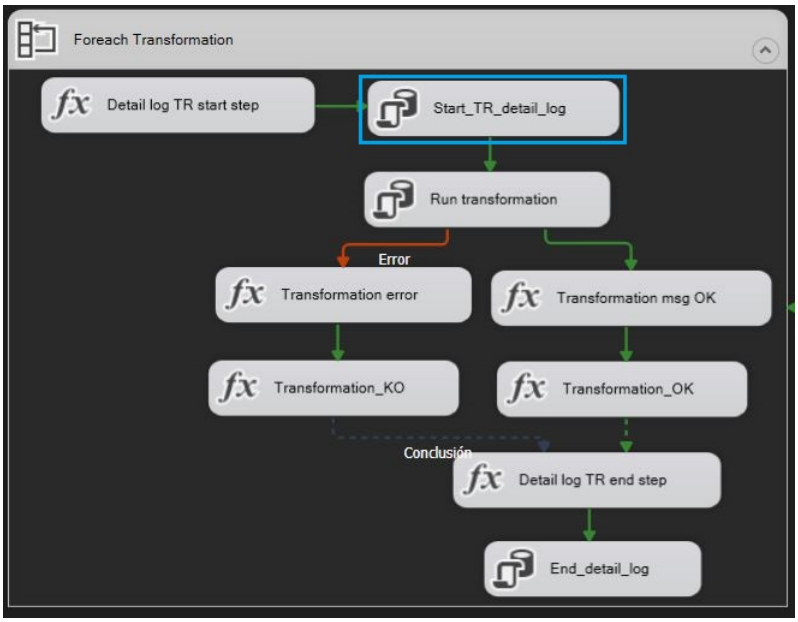


Figura 5.53: Tarea SQL Start TR detail log

- Ejecución del proceso de transformación (*Figura 5.54*).

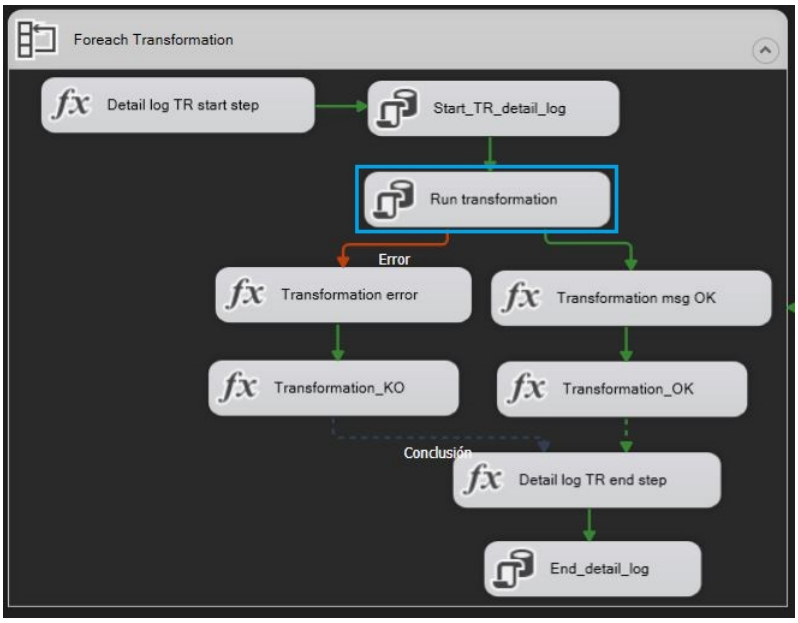


Figura 5.54: Tarea SQL Run transformation

- Cierre del log del proceso de transformación ejecutado (*Figura 5.55*).

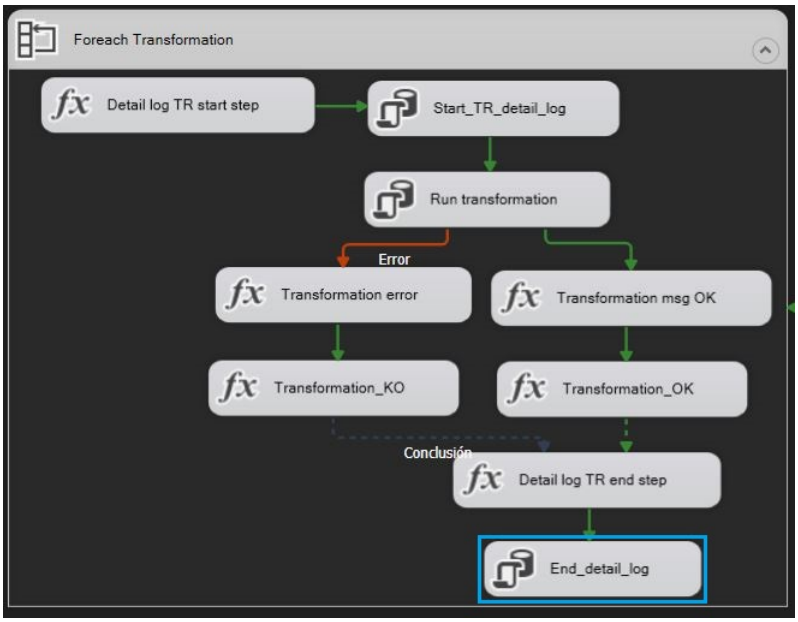


Figura 5.55: Escribe el final de la tarea en el log

- Al finalizar todas las iteraciones del bucle *Foreach Transformation*, finaliza la ejecución del proceso de transformación, recuperando *Main_process* el control de la ejecución.

5.6. Proceso de persistencia

Persistence_process es el proceso de la solución IO (*Figura 5.56*) encargado de realizar las operaciones calculadas durante el proceso de transformación en una BBDD local o remota, las cuales pueden ser las siguientes.

- INSERT
- UPDATE
- DELETE

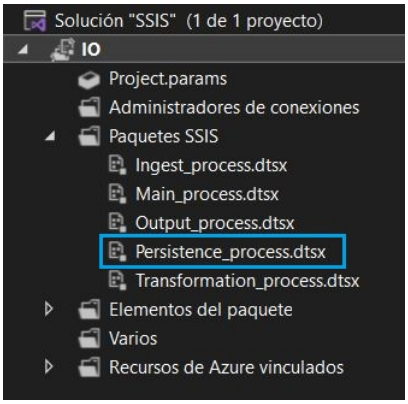


Figura 5.56: Paquete Persistence_process de la solución SSIS

Las operaciones están contenidas en la tabla *IO.load.operation*, cada registro contiene la columna delivery con el bloque de ejecución concreto, y la columna *load_table_name* con la tabla del

schema *load* donde habremos almacenado los registros que queremos persistir durante la etapa de transformación.

El paquete *Main_process* comprueba el id del proceso en ejecución en *IO.config.master_process_type*, si la columna *persistence_flag* está activa se ejecutará el contenedor de secuencia de persistencia como muestra la *Figura 5.57*.

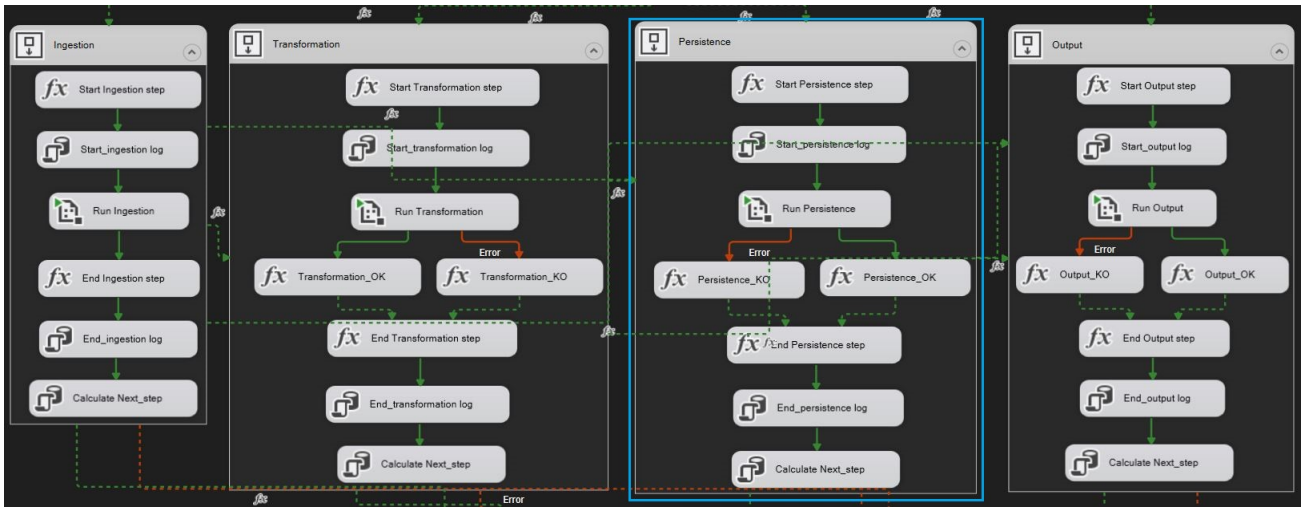


Figura 5.57: Contenedor de secuencia de persistencia

Este contenedor inserta un registro de inicio en la tabla de log *IO.monitor.master_process_log* para el proceso de persistencia, ejecuta el paquete *Persistence_process*, inserta un registro de final de proceso en la misma tabla y calcula el siguiente step a ejecutar.

5.6.0.1. Detalle del proceso

En la *Figura 5.58* se muestra el flujo de control del paquete *Persistence_process*.

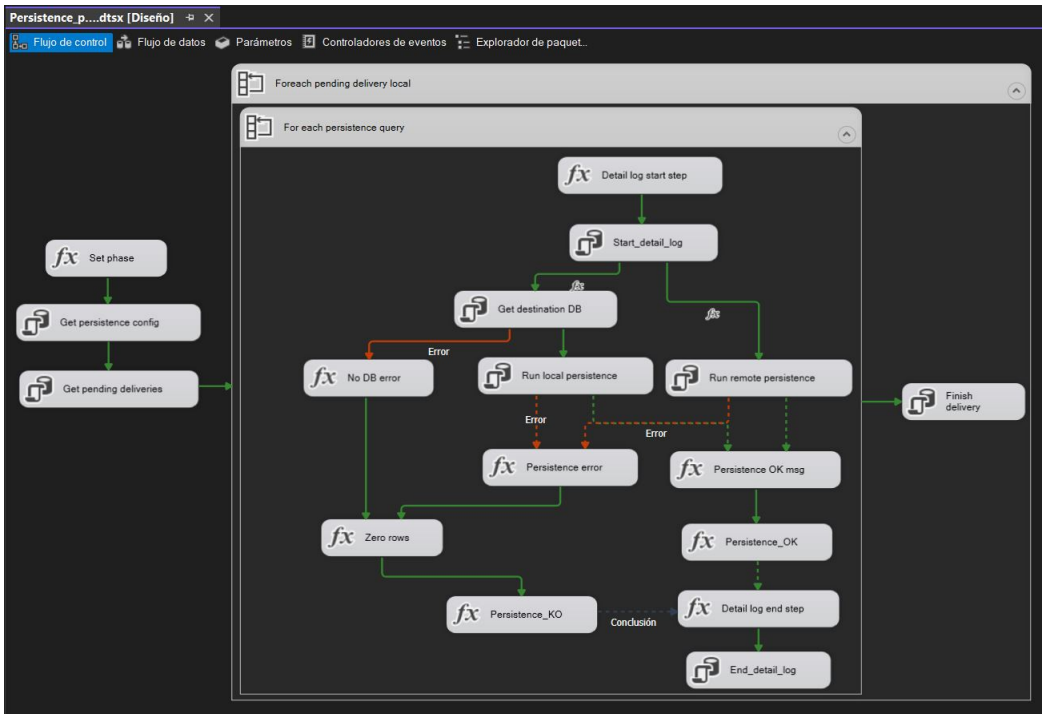


Figura 5.58: Flujo de control del paquete Persistence_process

Al comienzo del proceso obtenemos la información necesaria para ejecutar la persistencia, mostrando el detalle en la *Figura 5.59*.

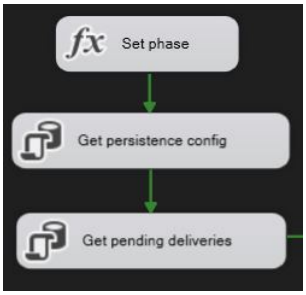


Figura 5.59: Bloque de obtención de información del proceso de persistencia

- Por un lado se ejecuta la tarea SQL *Get persistence config* la cual obtiene la parametrización de la fase de persistencia para el proceso en ejecución, la consulta está almacenada en la variable *sql_get_persistence_config* (*Figura 5.60*).

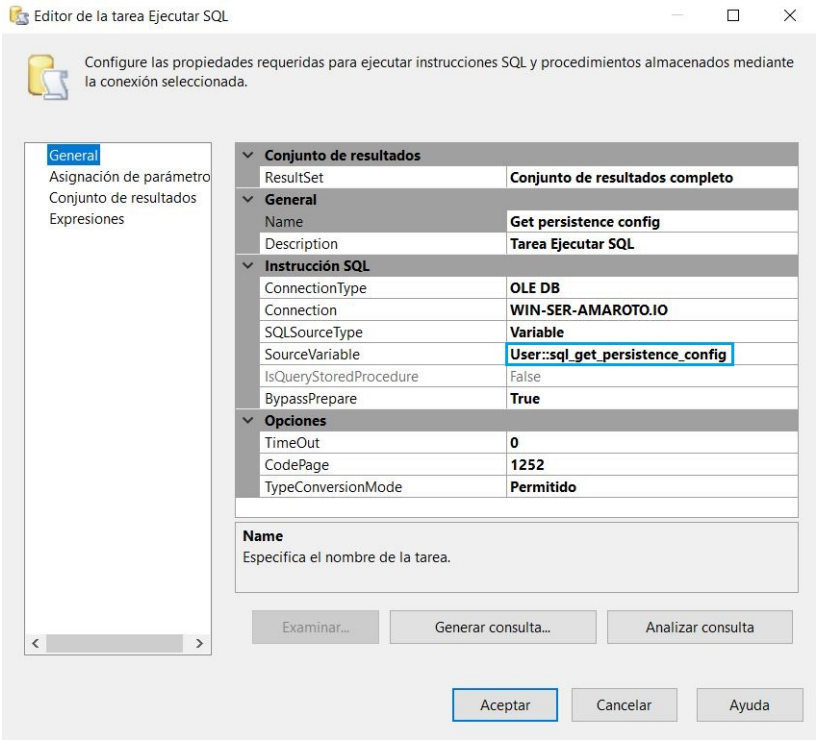


Figura 5.60: Recupera la información de la variable sql_get_persistence_config

- Los resultados obtenidos se almacenan en la variable de tipo objeto *obj_persistence_operations*.
- Por otro lado se obtienen las deliverys abiertas (*Figura 5.61*) para el proceso en ejecución mediante la tarea SQL *Get pending deliveries* que a su vez ejecuta una consulta almacenada en la variable *sql_get_pending_deliveries*.
- Los resultados obtenidos se almacenan en la variable de tipo objeto *obj_pending_deliveries* como muestra la *Figura 5.62*.

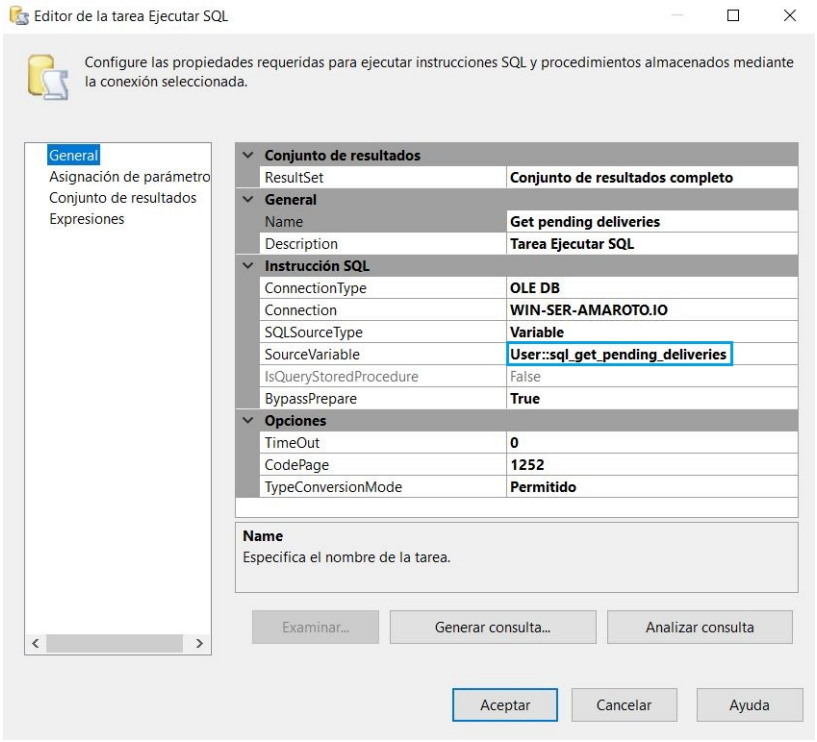


Figura 5.61: Recupera la información de la variable sql_get_pending_deliveries

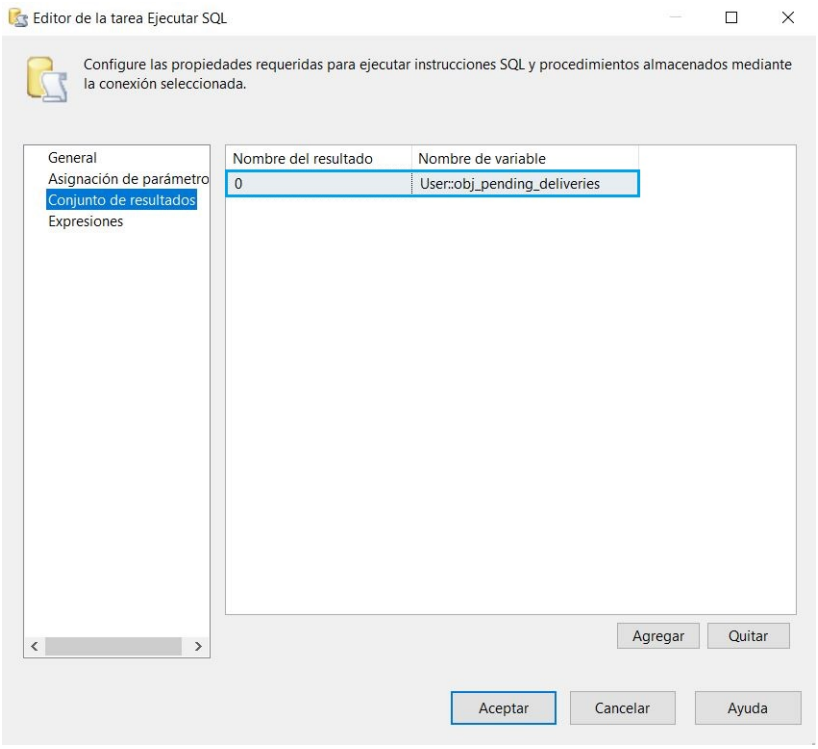


Figura 5.62: Almacena el resultado en la variable obj_pending_deliveries

Una vez recuperados estos valores se realizan los siguientes steps para cada delivery.

- Se recuperan los valores de la variable *obj_pending_deliveries* en el primer bucle foreach (*Figura 5.63*).

- v_delivery_id: Identificador de delivery.
- v_customer_id: Identificador del cliente relativo a la ejecución del proceso.

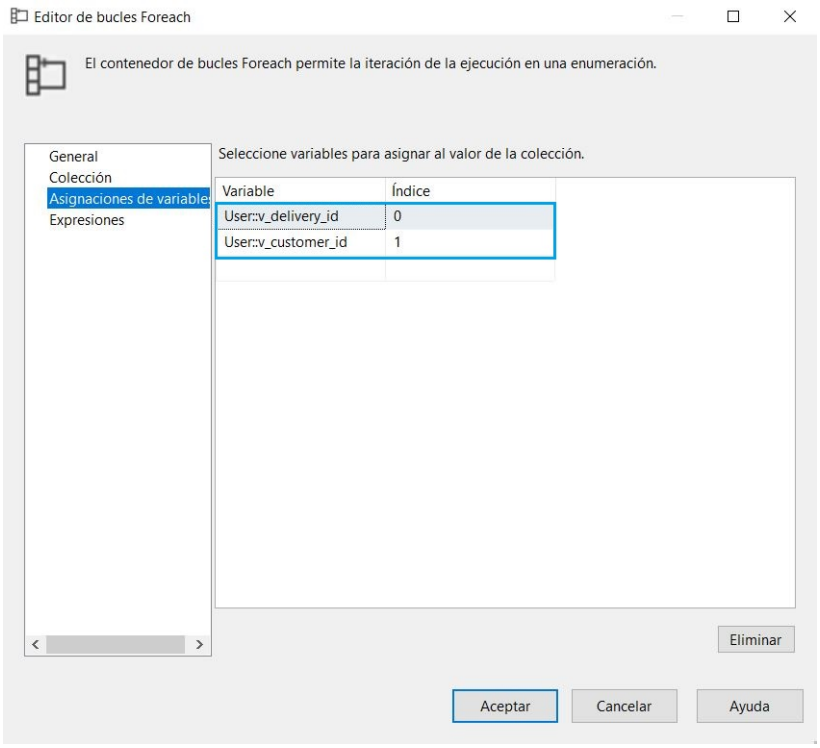


Figura 5.63: Asignación de variables en Foreach pending delivery local

- Se recuperan los valores de la variable *obj_persistence_operations* en el segundo bucle foreach (*Figura 5.64*).
  - v_sql_persistence_query: Consulta que contiene los datos a persistir.
  - v_p_entity: Entidad a ejecutar durante la iteración en curso.
  - v_destination_connection_server: Servidor de destino.
  - v_destination_connection_database: BBDD de destino.
  - v_destination_connection_user: Usuario de la BBDD de destino.
  - v_destination_connection_password: Contraseña del usuario de la BBDD de destino.

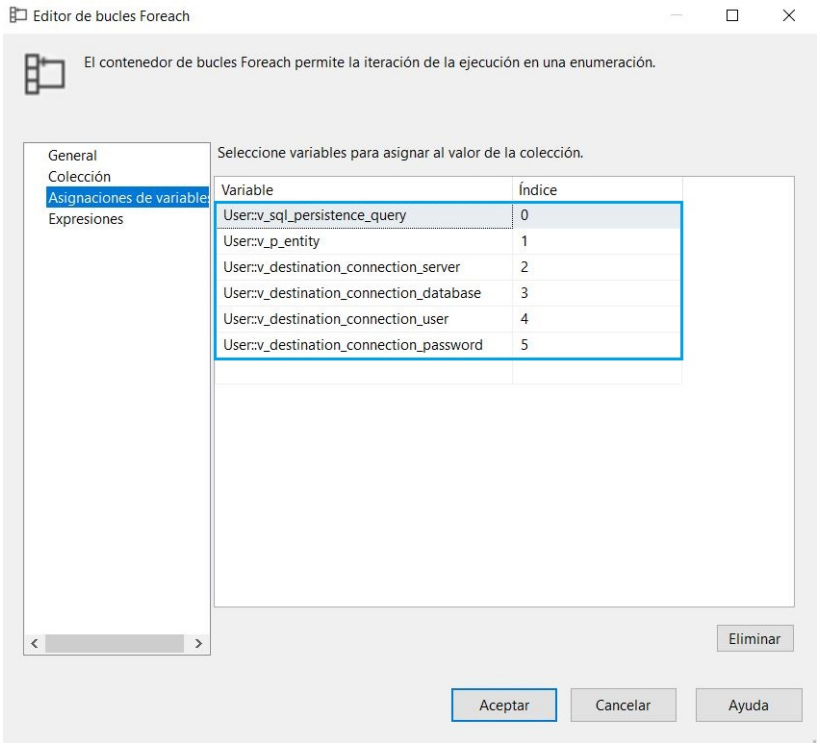


Figura 5.64: Asignación de variables en Foreach persistence query

- La *Tabla 5.3* muestra la relación entre la BBDD y el paquete SSIS.

Tabla 5.3: Detalle de la asignación de variables en Foreach persistence query

Id	Tabla BBDD	Columna BBDD	Variable SSIS
0	IO.config.detail_process_type	source_table_query	v_sql_persistence_query
1	IO.config.detail_process_type	entity	v_p_entity
2	IO.config.detail_process_type	destination_connection_server	v_destination_connection_server
3	IO.config.detail_process_type	destination_connection_database	v_destination_connection_database
4	IO.config.detail_process_type	destination_connection_user	v_destination_connection_user
5	IO.config.detail_process_type	destination_connection_password	v_destination_connection_password

- Se inserta un registro de inicio en el log de detalle para el step de persistencia en ejecución como muestra la *Figura 5.65*.

- Posteriormente, en función de si el valor de la variable *v_destination_connection_server* está vacío o no, se produce una bifurcación para realizar la operación de persistencia en nuestra BBDD local, o bien en una BBDD remota (*Figura 5.66*).

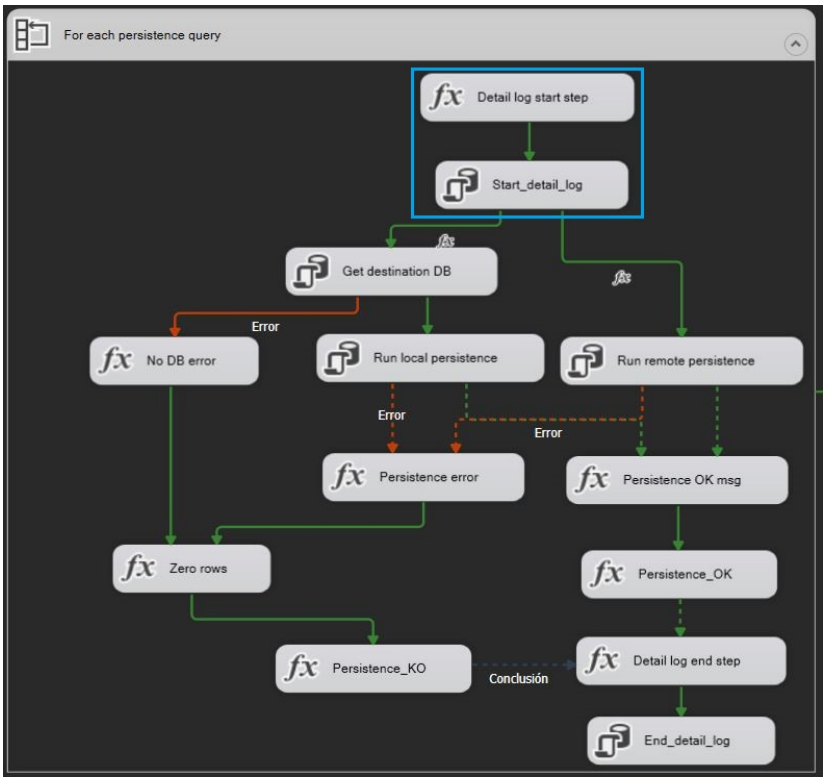


Figura 5.65: Bloque de registro de log inicial en el proceso de persistencia

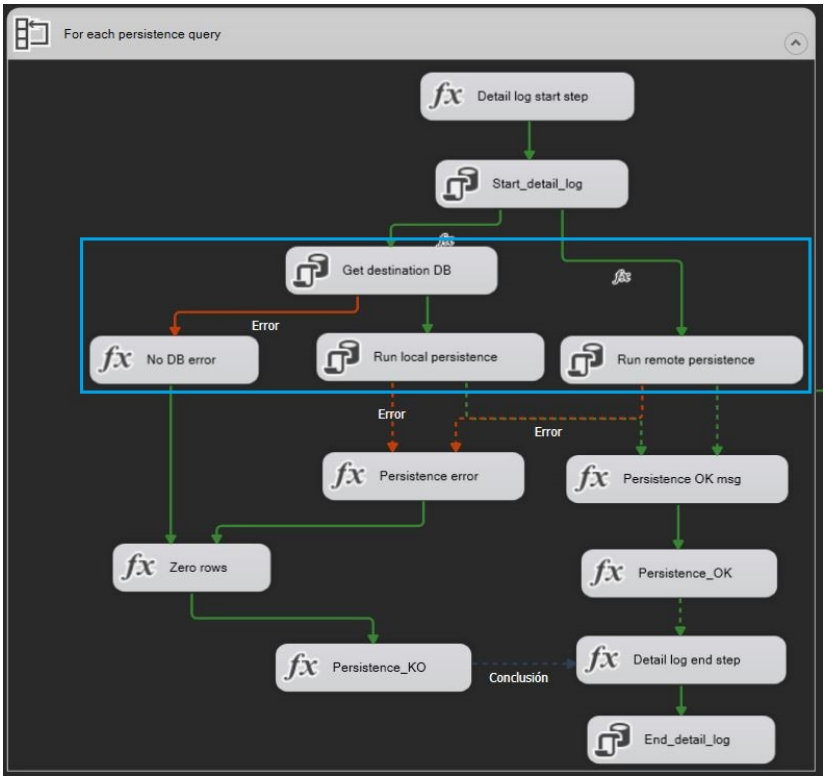


Figura 5.66: Bloque de escritura local o remota del proceso de persistencia

- Si se ejecuta la tarea SQL *Run local persistence* esta lanza la variable contenida en *v_sql_persistence_query*, la cual contiene una consulta o procedimiento almacenado encapsulado. Contiene los parámetros mostrados en la *Figura 5.67*.

- v_delivery_id: [Input] Identificador de la delivery de la ejecución.
- v_persistence_affected_rows: [Output] Cantidad de filas resultado de la operación ejecutada.

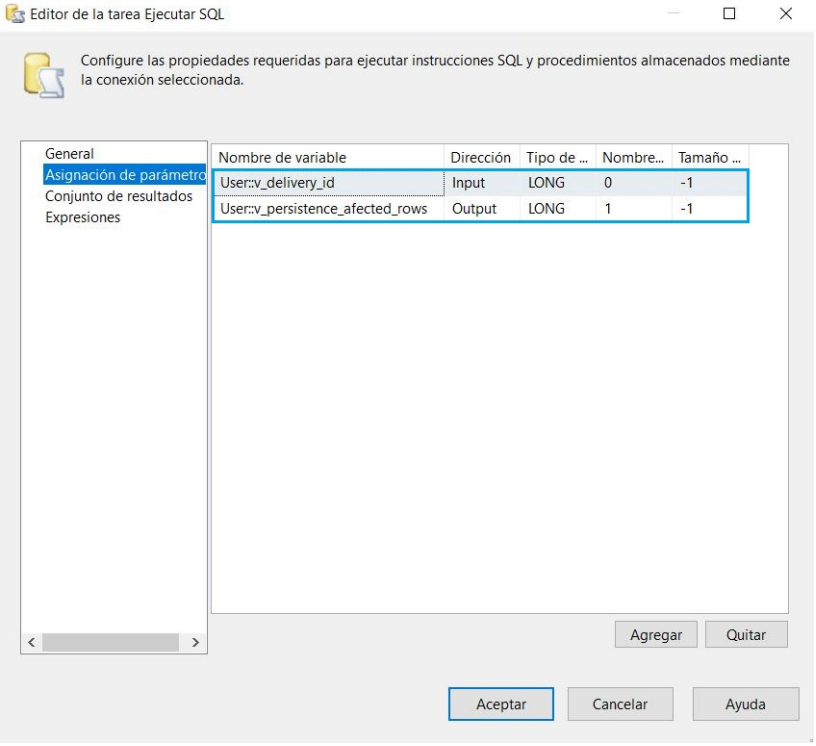


Figura 5.67: Asignación de parámetros de la tarea Run local persistence

- Si por el contrario se ejecuta la tarea SQL *Run remote persistence* esta varía con respecto a la tarea anterior en la asignación de parámetros, los cuales se muestran en la *Figura 5.68*.
  - v_delivery_id: [Input] Identificador de la delivery de la ejecución.
  - v_destination_connection_server: [Input] Servidor de destino.
  - v_destination_connection_database: [Input] BBDD de destino.
  - v_destination_connection_user: [Input] Usuario de la BBDD de destino.
  - v_destination_connection_password: [Input] Contraseña del usuario de la BBDD de destino.
  - v_persistence_affected_rows: [Output] Cantidad de filas resultado de la operación ejecutada.
- Tras realizar la persistencia se obtiene la cantidad de filas de las operaciones realizadas, o en caso de error un mensaje con el detalle que lo ha ocasionado. En ambos casos se inserta un registro de log en la tabla *IO.monitor.detail_process_log* con la traza obtenida (*Figura 5.69*).

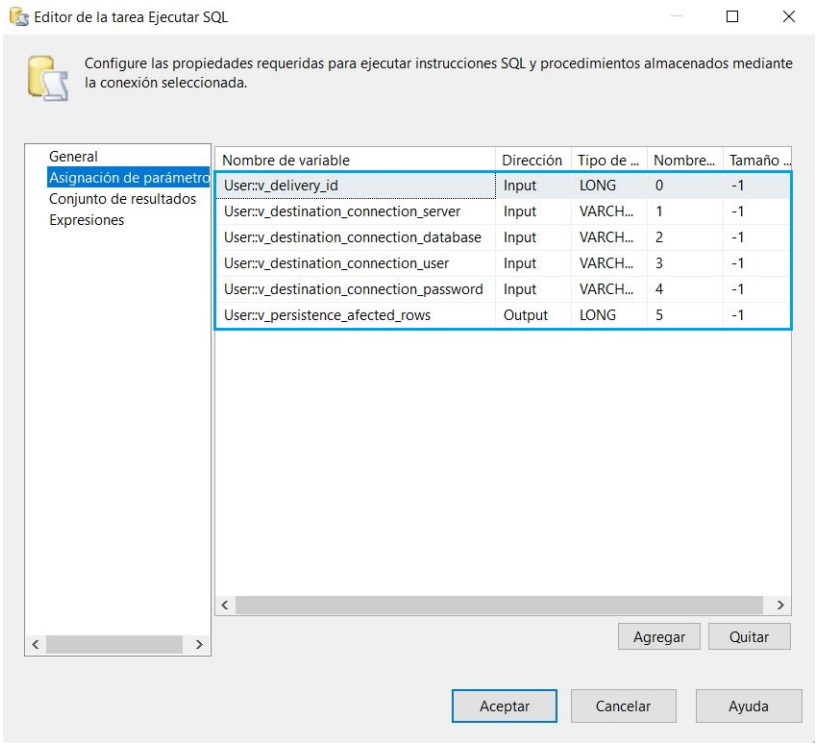


Figura 5.68: Asignación de parámetros de la tarea Run remote persistence

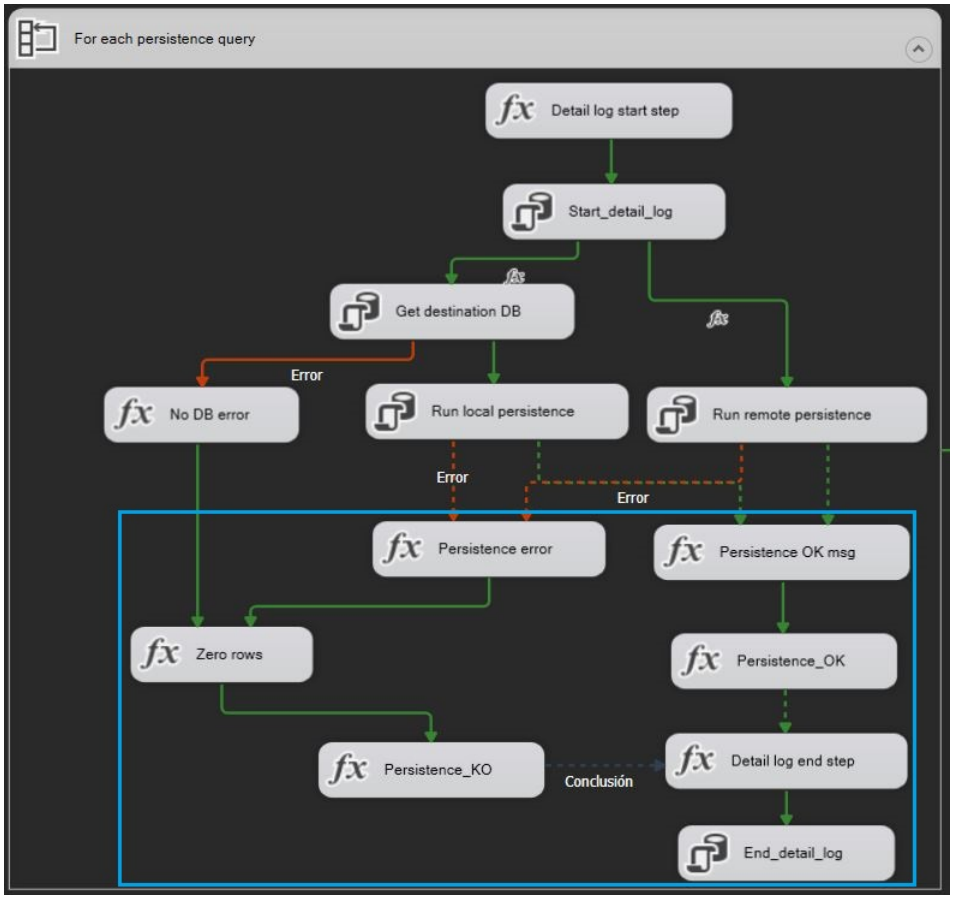


Figura 5.69: Bloque de registro de log final en el proceso de persistencia

- Una vez realizadas todas las iteraciones se finaliza la delivery como muestra la *Figura 5.70*.

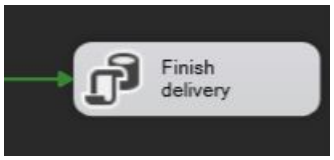


Figura 5.70: Tarea SQL Finish delivery

- Al finalizar todas las iteraciones del bucle *Foreach pending delivery local*, termina la ejecución del proceso de persistencia, recuperando *Main_process* el control de la ejecución.

5.7. Proceso de generación de outputs

Output_process es el proceso de la solución IO (*Figura 5.71*) encargado de generar informes a partir de consultas de BBDD, además de las siguientes operaciones sobre ficheros Outputs una vez generados.

- Compresión de archivos.
- Subida de ficheros a servidor FTP/SFTP.
- Envío de correo electrónico.
- Operaciones sobre archivos.
  - MOVE
  - COPY
  - DELETE
  - RENAME

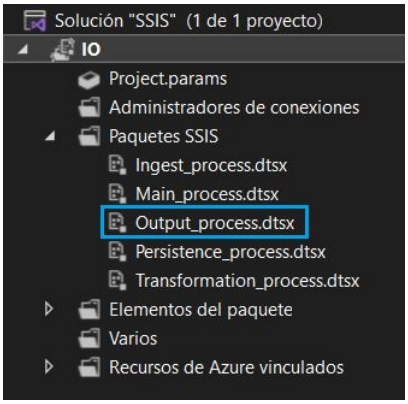


Figura 5.71: Paquete Output_process de la solución SSIS

El paquete *Main_process* comprueba el id del proceso en ejecución en *IO.config.master_process_type*, si la columna *output_flag* está activa se ejecutará el contenedor de secuencia Output como muestra la *Figura 5.72*.

Figura 5.72: Contenedor de secuencia Output

Este contenedor inserta un registro de inicio en la tabla de log *IO.monitor.master_process_log* para el proceso Output, ejecuta el paquete *Output_process*, inserta un registro de final de proceso en la misma tabla y calcula el siguiente step a ejecutar.

5.7.0.1. Detalle del proceso

En la *Figura 5.73* se muestra el flujo de control del paquete *Output_process*.

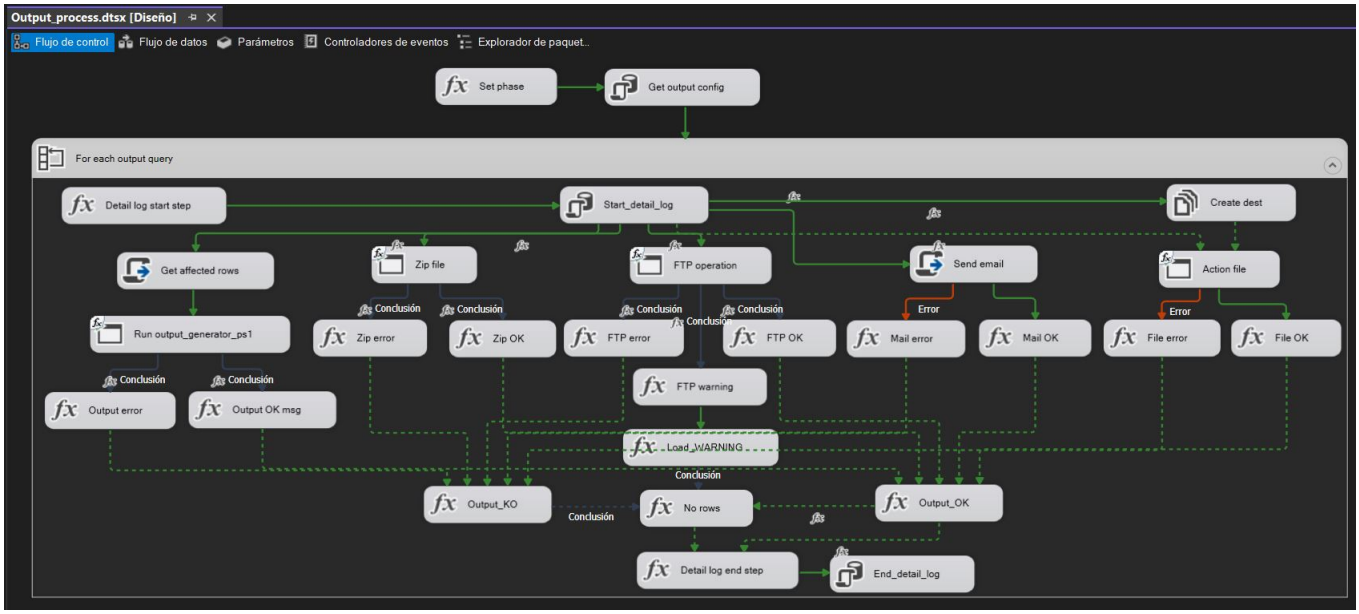


Figura 5.73: Flujo de control del paquete Output_process

La primera parte del proceso ejecuta la tarea SQL *Get output config* la cual obtiene la parametrización de la fase Output para el proceso en ejecución (*Figura 5.74*), la consulta está almacenada en la variable *sql_get_output_config* (*Figura 5.75*).

Se recuperan los valores de la variable *obj_output_operations* en el bucle *Foreach output query*. La relación entre la BBDD y el paquete SSIS se muestra en la *Tabla 5.4*.

Tabla 5.4: Detalle de la asignación de variables en Foreach output query

Id	Tabla BBDD	Columna BBDD	Variable SSIS
0	IO.config.detail_process_type	source_table_query	v_sql_output_query
1	IO.config.detail_process_type	entity	v_p_entity
2	IO.config.detail_process_type	destination_path	v_destination_path
3	IO.config.detail_process_type	destination_file_template_path	v_destination_file_template_path
4	IO.config.detail_process_type	destination_format	v_destination_format
5	IO.config.detail_process_type	destination_file_skip_rows	v_destination_file_skip_rows
6	IO.config.detail_process_type	generate_empty_file	v_generate_empty_file
7	IO.config.detail_process_type	source_type	v_source_type
8	IO.config.detail_process_type	zip_folder_path	v_zip_folder
9	IO.config.detail_process_type	zip_file_path	v_zip_file
10	IO.config.detail_process_type	ftp_server	v_ftp_server
11	IO.config.detail_process_type	ftp_port	v_ftp_port
12	IO.config.detail_process_type	ftp_user	v_ftp_user
13	IO.config.detail_process_type	ftp_pass	v_ftp_pass
14	IO.config.detail_process_type	ftp_operation	v_ftp_operation
15	IO.config.detail_process_type	ftp_folder	v_ftp_folder
16	IO.config.detail_process_type	ftp_local_folder	v_ftp_local_folder
17	IO.config.detail_process_type	ftp_local_file_pattern	v_ftp_local_file_pattern
18	IO.config.detail_process_type	email_from	v_email_from
19	IO.config.detail_process_type	email_to	v_email_to
20	IO.config.detail_process_type	email_subject	v_email_subject
21	IO.config.detail_process_type	email_body	v_email_body
22	IO.config.detail_process_type	email_has_attachment	v_email_has_attachment
23	IO.config.detail_process_type	email_attachment_path	v_email_attachment_path
24	IO.config.detail_process_type	email_attachment_pattern	v_email_attachment_pattern
25	IO.config.detail_process_type	ftp_private_key_path	v_ftp_private_key_path
26	IO.config.detail_process_type	ftp_ssh_host_key_fingerprint	v_ftp_ssh_host_key_fingerprint
27	IO.config.detail_process_type	ftp_private_key_passphrase	v_ftp_private_key_passphrase
28	IO.config.detail_process_type	file_action	v_file_action
29	IO.config.detail_process_type	file_action_source_pattern	v_file_action_source_pattern
30	IO.config.detail_process_type	file_action_destination_folder	v_file_action_destination_folder
31	IO.config.detail_process_type	customer	v_customer
32	IO.config.detail_process_type	process_name	v_process_name
33	IO.config.detail_process_type	zip_rar_pass	v_zip_rar_pass

Figura 5.74: Bloque de obtención de información del proceso Output

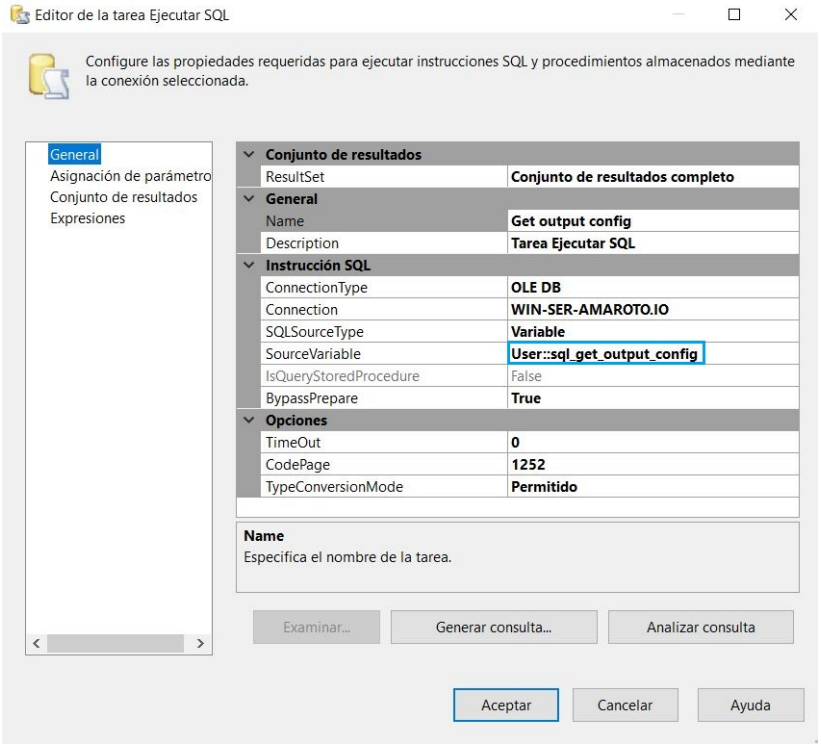


Figura 5.75: Recupera la información de la variable sql_get_output_config

El bucle realizará una iteración por cada registro en *IO.config.detail_process_type* para el proceso Output seleccionado, realizando los pasos que se detallan a continuación.

- Se inserta un registro de inicio en *IO.monitor.detail_process_log* con el detalle como muestra la *Figura 5.76*.

Figura 5.76: Inserta el registro de inicio en el log del proceso Output

- A continuación, el proceso se divide en diferentes caminos, pudiéndose ejecutar cada uno de ellos en función de la configuración del valor de la variable *v_source_type*, la cual puede contener los siguientes valores.
  - v_source_type == 'BBDD'.
  - El apartado de BBDD contiene dos tareas principales (*Figura 5.77*). La primera de ellas *Get affected rows* ejecuta un script en C# , el cual obtiene el número de filas afectadas mediante el *Código 5.46* que se muestra a continuación.

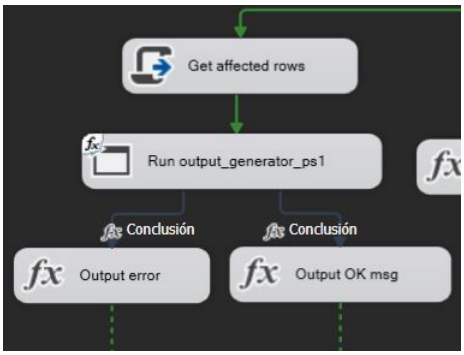


Figura 5.77: Camino tipo BBDD del proceso Output

Código 5.46: Output - Get affected rows.cs

```
#region Namespaces
using System;
using Microsoft.SqlServer.Dts.Runtime;
using System.Data.SqlClient;
using System.Data.OleDb;
using System.Windows.Forms;
using System.Data;
#endregion

namespace ST_9895c08fc965486faa99eac2b5d0e276
{
   [Microsoft.SqlServer.Dts.Tasks.ScriptTask.SSISScriptTaskEntryPointAttribute]
   public partial class ScriptMain :
     Microsoft.SqlServer.Dts.Tasks.ScriptTask.VSTARTScriptObjectModelBase
   {
      public void Main()
      {
            try
            {
                int affected_rows = 0;
                String query = (String)Dts.Variables["User::v_sql_output_query"].Value;
                int skip_rows = (int)Dts.Variables["User::v_destination_file_skip_rows"].Value;
                ConnectionManager cm = Dts.Connections["WIN-SER-AMAROTO.IO"];

                using (OleDbConnection connection = new OleDbConnection(cm.ConnectionString))
                {
                    OleDbCommand command = new OleDbCommand(query, connection);
                    connection.Open();

                    OleDbDataReader reader = command.ExecuteReader();

                    while (reader.Read())
                    {
                        affected_rows++;
                    }

                    Dts.Variables["User::v_affected_rows"].Value = affected_rows - skip_rows;
                }

                Dts.TaskResult = (int)ScriptResults.Success;
            }
            catch (Exception e)
            {
                Dts.Events.FireError(0, "Get Output affected rows Script", e.Message + "\r" +
    e.StackTrace, String.Empty, 0);
                Dts.TaskResult = (int)ScriptResults.Failure;
            }
      }
        #region ScriptResults declaration
        enum ScriptResults
        {
            Success = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Success,
            Failure = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Failure
        };
        #endregion
```

```
54
55      }
56  }
```

- Por otro lado, *Run output_generator_ps1* lanza el script en PowerShell *output_generator*, el cual hemos detallado en *Generación de ficheros* 5.2.5 de este capítulo.

- v_source_type = 'ZIP'.

- El apartado *ZIP* mostrado en la *Figura 5.78* realiza la compresión de contenido Output en una carpeta en formato *ZIP*, *RAR* o *GZIP* a través de la herramienta *Compresión y descompresión de archivos* 5.3 que hemos visto en este capítulo.

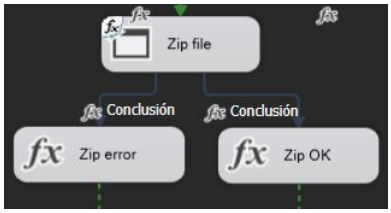


Figura 5.78: Camino tipo ZIP del proceso Output

- v_source_type = 'FTP' o v_source_type = 'SFTP'.

- El apartado FTP/SFTP mostrado en la *Figura 5.79* realiza una conexión con un servidor FTP o SFTP y deposita los archivos indicados en este a través de la herramienta *Carga y descarga de archivos por FTP/SFTP* 5.2.3 que hemos visto en este capítulo.

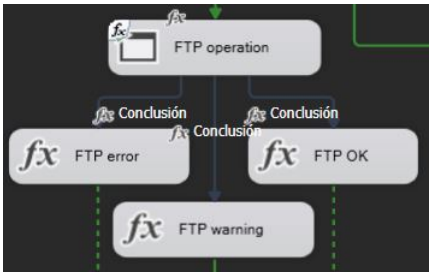


Figura 5.79: Camino tipo FTP/SFTP del proceso Output

- v_source_type = 'EMAIL'.

- El apartado *EMAIL* mostrado en la *Figura 5.80* envía un correo el cual puede contener adjuntos si se necesita, a diferentes destinatarios. Para ello la tarea *Send email* ejecuta un script en C# ejecutando el *Código 5.47*.

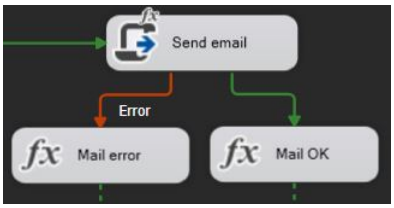


Figura 5.80: Camino tipo EMAIL del proceso Output

Código 5.47: Output - Send email.cs

```
#region Namespaces
using System;
using System.Net.Mail;
using System.IO;
#endregion

namespace ST_712178489edc4ca69c38495b49170f8d
{
   [Microsoft.SqlServer.Dts.Tasks.ScriptTask.SSISScriptTaskEntryPointAttribute]
   public partial class ScriptMain :
     Microsoft.SqlServer.Dts.Tasks.ScriptTask.VSTARTScriptObjectModelBase
   {
      public void Main()
      {
            try
            {
                String smtp_host = (String)Dts.Variables["$Package::email_server"].Value;

                String from = (String)Dts.Variables["User::v_email_from"].Value;
                String to = (String)Dts.Variables["User::v_email_to"].Value;
                String subject = (String)Dts.Variables["User::v_email_subject"].Value;
                String body = (String)Dts.Variables["User::v_email_body"].Value;
                int has_attachment = (int)Dts.Variables["User::v_email_has_attachment"].Value;
                String attachment_path = (String)Dts.Variables["User::v_email_attachment_path"].Value;
                String attachment_pattern = (String)Dts.Variables["User::v_email_attachment_pattern"].Value;

                SmtpClient client = new SmtpClient(smtp_host);
                client.Credentials = new System.Net.NetworkCredential(from, "Usuario2011");

                MailMessage mail = new MailMessage(from, to.Replace(";", ","), subject, body);
                mail.IsBodyHtml = true;

                if (has_attachment == 1)
                {
                    if (Directory.Exists(attachment_path))
                    {
                        string[] fileEntries = Directory.GetFiles(attachment_path, attachment_pattern);
                        Dts.Variables["User::v_affected_rows"].Value = fileEntries.Length;

                        body = body + "<p> " + fileEntries.Length.ToString() + " files have been attached into mail: </p>";

                        body = body + "<table cellpadding=\"5\" cellspacing=\"0\"> " +
                    "<tr bgcolor=\"DeepSkyBlue\"> " +
                    "<th style=\"border: 1px solid black; border-collapse: collapse\"> File name </th> " +
                    "<th style=\"border: 1px solid black; border-collapse: collapse\"> Size </th> " +
                    "</tr> ";

                        Attachment file_attachment;
                        float size = 0;

                        foreach (string fileName in fileEntries)
                        {
                            FileInfo fi = new FileInfo(fileName);
                            size = size + fi.Length;

```

```
                        body = body + "<tr> " +
                    "<td style=\"border: 1px solid black; text-align: left; border-collapse: collapse\"> " + fi.Name + " </td> " +
                    "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " + fi.Length.ToString() + " bytes </td> " +
                    "</tr> ";

                        file_attachment = new Attachment(fileName);
                        mail.Attachments.Add(file_attachment);
                    }

                    size = size / 1000000;

                    if (size >= 20)
                    {
                        mail.Attachments.Clear();
                        body = body + "</table> <p style = \"color: red;\"> <b> " + fileEntries.Length.ToString() + " files should have been attached into mail but the total size (" + size + " MB) exceeds the allowed size (20 MB), NO FILE HAS BEEN ATTACHED </b> </p>";
                    }
                    else
                    {
                        body = body + " </table > <p style = \"color: green;\"> <b> " + fileEntries.Length.ToString() + " files have been attached into mail with " + size + " MB of total size</p>";
                    }

                    mail.Body = body;
                }
                else
                {
                    throw new InvalidOperationException("Folder for attached files does not exist");
                }
            }
            else
            {
                Dts.Variables["User::v_affected_rows"].Value = 1;
            }

            client.Send(mail);

            client.Dispose();
            mail.Dispose();

            Dts.TaskResult = (int)ScriptResults.Success;
        }
        catch (Exception e)
        {
            Dts.Events.FireError(0, "Report Mail Script", e.Message + "\r" + e.StackTrace, String.Empty, 0);
            Dts.TaskResult = (int)ScriptResults.Failure;
        }

    }
    #region ScriptResults declaration
    enum ScriptResults
    {
        Success = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Success,
        Failure = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Failure
    };
    #endregion
}
}
```

- Dentro del script se utilizan las siguientes variables obtenidas de los parámetros del paquete.
    - Package::email_server: Servidor SMTP desde el que se enviarán los correos.

- User::v_email_from: Cuenta de correo desde la que se envía el correo.
- User::v_email_to: Listado de destinatarios del correo separados por coma ',' o por punto y coma ';'.
- User::v_email_subject: Asunto del correo.
- User::v_email_body: Texto que conforma el contenido del correo.
- User::v_email_has_attachment: Flag para indicar si el correo contiene elementos adjuntos.
  - 1: contiene adjuntos
  - 0: NO contiene adjuntos
- User::v_email_attachment_path: Ruta de los elementos adjuntos.
- User::v_email_attachment_pattern: Plantilla que han de cumplir los elementos adjuntos del correo.

- v_source_type = 'FILE' & v_file_action_destination_folder != '' & v_file_action != 'RENAME'.
- El apartado *FILE* mostrado en la *Figura 5.81* es capaz de realizar las siguientes acciones sobre archivos Output.
  - COPY: Realiza la copia de un archivo en una ruta diferente. (En caso de no existir la carpeta de destino se crea en tiempo de ejecución).
  - MOVE: Mueve el archivo a una ruta diferente. (En caso de no existir la carpeta de destino se crea en tiempo de ejecución).
  - DELETE: Elimina el archivo indicado.
  - RENAME: Modifica el nombre del archivo por el indicado.

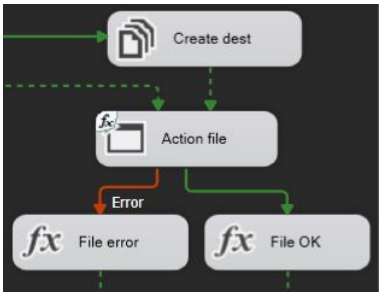


Figura 5.81: Camino tipo FILE del proceso Output

- Una vez finaliza cualquiera de estos caminos, se inserta el log de la iteración en la tabla *IO.monitor.detail_process_log* con el detalle del proceso, con la excepción de que se produzca un error, fallando en este caso el proceso de manera controlada como muestra la *Figura 5.82*.

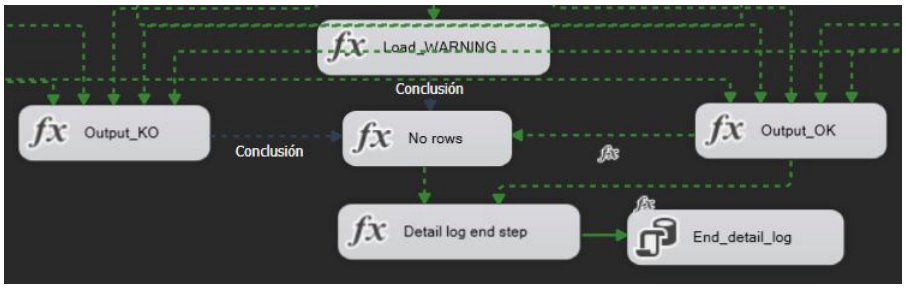


Figura 5.82: Inserta el registro final en el log del proceso Output

- Al finalizar todas las iteraciones del bucle *Foreach output query*, termina la ejecución del proceso Output, recuperando *Main_process* el control de la ejecución.

5.8. Conclusiones

En este capítulo se ha descrito el desarrollo completo para la solución IO, comenzando por las herramientas de soporte que se encuentran embebidas en el resto del proyecto, y continuando con los procesos diseñados para el core del proyecto mediante el uso del software SSIS.

Para más información sobre como desplegar la herramienta en SQL Server o como generar un proceso en IO, ver los anexos *Manual de despliegue* A o *Manual de usuario* B.

Capítulo 6

Pruebas

6.1. Introducción

En los capítulos anteriores, nos hemos centrado en el diseño e implementación de la solución teniendo en cuenta las distintas tecnologías aplicadas. En este capítulo nos centraremos en las pruebas o casos de uso que nos permitan comprobar el correcto funcionamiento del proyecto en cumplimiento con los requisitos funcionales.

6.2. Diseño de BBDD y modelo entidad relación VehicleSales

Previo a la generación de los casos de uso, se ha creado una BBDD a modo de ejemplo en *Tercera Forma Normal* [10], el sector de esta es el automovilístico, el cual tiene una definición sencilla y nos permite realizar pruebas de una forma exhaustiva.

El objetivo de esta BBDD consiste en tener información básica de las ventas de vehículos para una determinada marca. Para ello, además de tener información de los coches, como la marca, el tipo de modelo o combustible, podemos encontrar la información relacionada con el proceso de ventas, vendedores, clientes, propuestas y sus tipos, ofertas, y finalmente ventas realizadas a éstos. Aunque se trata de un modelo de datos sencillo, que al mismo tiempo nos permite probar de diferentes formas las funcionalidades del proyecto.

La nomenclatura utilizada ha sido snake_case al igual que en la BBDD IO. Adicionalmente las tablas maestras también contienen la palabra *type* en el nombre, y de esta forma podemos diferenciarlas del resto de objetos.

6.2.1. Definición y creación de BBDD

El primer paso a dar consiste en la creación de la BBDD a la que hemos denominado *VehicleSales*. Se muestra el *Código 6.1* a continuación el cual adicionalmente incluye un borrado de esta en caso de que fuese necesario.

Código 6.1: Creacion BBDD VehicleSales

```
1  -- Create VehicleSales Database
2  CREATE DATABASE VehicleSales;
3  GO
4
5  -- Use VehicleSales Database
6  USE VehicleSales;
7  GO
8
9  /*
10 USE Master;
```

```
ALTER DATABASE VehicleSales SET single_user with rollback immediate;
ALTER DATABASE VehicleSales SET MULTI_USER;
DROP DATABASE VehicleSales;
*/
```

6.2.2. Definición y creación de tablas

En esta sección, seguimos con la definición y creación de tablas de nuestra BBDD *VehicleSales*. A modo de resumen se van a crear las siguientes diez tablas.

- brand_type.
- fuel_type.
- model_type.
- proposal_type.
- concessionare.
- customer.
- seller.
- proposal.
- offer.
- sale.

A continuación su detalla cada una de ellas, incluyendo sus columnas y el *Código 6.2-6.11* de creación, el cual incluye claves primarias, foráneas y valores por defecto. Adicionalmente se mostrará el diagrama entidad-relación del modelo completo en la *Figura 6.1*.

- brand_type: Tabla para almacenar información sobre las marcas de vehículos.
  - id (PK, int): Identificador único de la marca.
  - name (nvarchar): Nombre de la marca de vehículos.
  - active (bit): Indica si el registro está o no activo.
  - creation_date (DF, datetime): Fecha de creación del registro generada por default.
  - modification_date (TR, datetime): Fecha de modificación del registro actualizada por trigger.
  - io_id (int): Clave de la operación realizada en la BBDD IO.

Código 6.2: Creacion de tabla brand_type

```
-- Create brand_type Table
CREATE TABLE [dbo].[brand_type] (
    id INT IDENTITY(1, 1),
    [name] NVARCHAR(64),
```

```
   active BIT,
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_brand_type] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
     ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[brand_type] ADD DEFAULT (1) FOR [active]
GO

ALTER TABLE [dbo].[brand_type] ADD CONSTRAINT [df_brand_type_creation_date] DEFAULT (getdate()) FOR
     [creation_date]
GO
```

- fuel_type: Tabla para almacenar información sobre el tipo de combustible del modelo.
  - id (PK, int): Identificador único del combustible.
  - name (nvarchar): Tipo de combustible del vehículo (gasolina, diésel, eléctrico, híbrido).
  - active (bit): Indica si el registro está o no activo.
  - creation_date (DF, datetime): Fecha de creación del registro generada por default.
  - modification_date (TR, datetime): Fecha de modificación del registro actualizada por trigger.
  - io_id (int): Clave de la operación realizada en la BBDD IO.

Código 6.3: Creacion de tabla fuel_type

```
-- Create fuel_type Table
CREATE TABLE [dbo].[fuel_type] (
    id INT IDENTITY(1, 1),
    [name] NVARCHAR(64),
   active BIT,
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_fuel_type] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
     ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[fuel_type] ADD DEFAULT (1) FOR [active]
GO

ALTER TABLE [dbo].[fuel_type] ADD CONSTRAINT [df_fuel_type_creation_date] DEFAULT (getdate()) FOR [creation_date]
GO
```

- model_type: Tabla para almacenar información sobre los modelos de vehículos.
  - id (PK, int): Identificador único del modelo.

- model_name (nvarchar): Nombre del modelo de vehículo.
- fuel_type_id (FK, int): Clave foránea que referencia la tabla fuel_type.
- brand_id (FK, int): Clave foránea que referencia la tabla concessionare.
- active (bit): Indica si el registro está o no activo.
- creation_date (DF, datetime): Fecha de creación del registro generada por default.
- modification_date (TR, datetime): Fecha de modificación del registro actualizada por trigger.
- io_id (int): Clave de la operación realizada en la BBDD IO.

Código 6.4: Creacion de tabla model_type

```
-- Create model_type Table
CREATE TABLE [dbo].[model_type] (
    id INT IDENTITY(1, 1),
    [name] NVARCHAR(128),
    brand_type_id INT,
   fuel_type_id INT,
   active BIT,
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_model_type] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
     ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[model_type] ADD DEFAULT (1) FOR [active]
GO

ALTER TABLE [dbo].[model_type] ADD CONSTRAINT [df_model_type_creation_date] DEFAULT (getdate()) FOR
     [creation_date]
GO

ALTER TABLE [dbo].[model_type] WITH CHECK ADD CONSTRAINT [fk_model_type_brand_type_id] FOREIGN
     KEY([brand_type_id])
REFERENCES [dbo].[brand_type] ([id])
GO
ALTER TABLE [dbo].[model_type] CHECK CONSTRAINT [fk_model_type_brand_type_id]
GO

ALTER TABLE [dbo].[model_type] WITH CHECK ADD CONSTRAINT [fk_model_type_fuel_type_id] FOREIGN KEY([fuel_type_id])
REFERENCES [dbo].[fuel_type] ([id])
GO
ALTER TABLE [dbo].[model_type] CHECK CONSTRAINT [fk_model_type_brand_type_id]
GO
```

- proposal_type: Tabla para almacenar información sobre el tipo de propuesta realizada.
  - id (PK, int): Identificador único del tipo de propuesta.
  - name (nvarchar): Tipo de propuesta realizada (nuevo, seminuevo, renting, leasing).
  - active (bit): Indica si el registro está o no activo.
  - creation_date (DF, datetime): Fecha de creación del registro generada por default.

- modification_date (TR, datetime): Fecha de modificación del registro actualizada por trigger.
- io_id (int): Clave de la operación realizada en la BBDD IO.

Código 6.5: Creacion de tabla proposal_type

```
-- Create proposal_type Table
CREATE TABLE [dbo].[proposal_type] (
    id INT IDENTITY(1, 1),
    [name] NVARCHAR(128),
   active BIT,
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_sale_type] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
     ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[proposal_type] ADD DEFAULT (1) FOR [active]
GO

ALTER TABLE [dbo].[proposal_type] ADD CONSTRAINT [df_proposal_type_creation_date] DEFAULT (getdate()) FOR
     [creation_date]
GO
```

- concessionare: Tabla para almacenar información sobre los concesionarios de vehículos.
  - id (PK, int): Identificador único del concesionario.
  - name (nvarchar): Nombre del concesionario.
  - address (nvarchar): Dirección del concesionario.
  - city (nvarchar): Ciudad donde se encuentra el concesionario.
  - country (nvarchar): País donde se encuentra el concesionario.
  - postal_code (nvarchar): Código postal donde se encuentra el concesionario.
  - active (bit): Indica si el registro está o no activo.
  - creation_date (DF, datetime): Fecha de creación del registro generada por default.
  - modification_date (TR, datetime): Fecha de modificación del registro actualizada por trigger.
  - io_id (int): Clave de la operación realizada en la BBDD IO.

Código 6.6: Creacion de tabla concessionare

```
-- Create concessionaire Table
CREATE TABLE [dbo].[concessionaire] (
    id INT IDENTITY(1, 1),
    [name] NVARCHAR(128),
    [address] NVARCHAR (512),
   city NVARCHAR(64),
    country NVARCHAR(32),
   postal_code NVARCHAR(18),
```

```
   info NVARCHAR(512),
   active BIT,
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_concessionaire] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
     ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[concessionaire] ADD DEFAULT (1) FOR [active]
GO

ALTER TABLE [dbo].[concessionaire] ADD CONSTRAINT [df_concessionaire_creation_date] DEFAULT (getdate()) FOR
     [creation_date]
GO
```

- customer: Tabla para almacenar información sobre los clientes.
  - id (PK, int): Identificador único del cliente.
  - name (nvarchar): Nombre del cliente.
  - phone (nvarchar): Teléfono del cliente.
  - email (nvarchar): Dirección de correo electrónico del cliente.
  - creation_date (DF, datetime): Fecha de creación del registro generada por default.
  - modification_date (TR, datetime): Fecha de modificación del registro actualizada por trigger.
  - io_id (int): Clave de la operación realizada en la BBDD IO.

Código 6.7: Creacion de tabla customer

```
-- Create customer Table
CREATE TABLE [dbo].[customer] (
    id INT IDENTITY(1, 1),
    [name] NVARCHAR(128),
   phone NVARCHAR(18),
    email NVARCHAR(256),
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_customer] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
     ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[customer] ADD CONSTRAINT [df_customer_creation_date] DEFAULT (getdate()) FOR [creation_date]
GO
```

- seller: Tabla para almacenar información sobre los vendedores de los concesionarios.
  - id (PK, int): Identificador único del vendedor.

- name (nvarchar): Nombre del vendedor.
- phone (nvarchar): Teléfono del vendedor.
- email (nvarchar): Dirección de correo electrónico del vendedor.
- concessionaire_id (FK, int): Clave foránea que referencia la tabla concessionaire.
- active (bit): Indica si el registro está o no activo.
- creation_date (DF, datetime): Fecha de creación del registro generada por default.
- modification_date (TR, datetime): Fecha de modificación del registro actualizada por trigger.
- io_id (int): Clave de la operación realizada en la BBDD IO.

Código 6.8: Creacion de tabla seller

```
-- Create seller Table
CREATE TABLE [dbo].[seller] (
    id INT IDENTITY(1, 1),
    [name] NVARCHAR(128),
   phone NVARCHAR(18),
    email NVARCHAR(256),
    concessionaire_id INT,
   active BIT,
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_seller] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
     ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[seller] ADD DEFAULT (1) FOR [active]
GO

ALTER TABLE [dbo].[seller] ADD CONSTRAINT [df_seller_creation_date] DEFAULT (getdate()) FOR [creation_date]
GO

ALTER TABLE [dbo].[seller] WITH CHECK ADD CONSTRAINT [fk_seller_concessionaire_id] FOREIGN
     KEY([concessionaire_id])
REFERENCES [dbo].[concessionaire] ([id])
GO
ALTER TABLE [dbo].[seller] CHECK CONSTRAINT [fk_seller_concessionaire_id]
GO
```

- proposal: Tabla para almacenar información sobre las propuestas realizadas a los clientes.
  - id (PK, int): Identificador único de la propuesta.
  - customer_id (FK, int): Clave foránea que referencia la tabla customer.
  - seller_id (FK, int): Clave foránea que referencia la tabla seller.
  - model_type_id (FK, int): Clave foránea que referencia la tabla model_type.
  - proposal_date (date): Fecha en que se realizó la propuesta.

- proposal_amount (decimal): Precio de la propuesta.
- proposal_type_id (FK, int): Clave foránea que referencia la tabla proposal_type.
- active (bit): Indica si el registro está o no activo.
- creation_date (DF, datetime): Fecha de creación del registro generada por default.
- modification_date (TR, datetime): Fecha de modificación del registro actualizada por trigger.
- io_id (int): Clave de la operación realizada en la BBDD IO.

Código 6.9: Creacion de tabla proposal

```
-- Create proposal Table
CREATE TABLE [dbo].[proposal] (
    id INT IDENTITY(1, 1),
    customer_id INT,
    seller_id INT,
   model_type_id INT,
    proposal_date DATE,
   proposal_amount DECIMAL(10, 2),
   proposal_type_id INT,
   active BIT,
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_proposal] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
     ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[proposal] ADD DEFAULT (1) FOR [active]
GO

ALTER TABLE [dbo].[proposal] ADD CONSTRAINT [df_prposal_creation_date] DEFAULT (getdate()) FOR [creation_date]
GO

ALTER TABLE [dbo].[proposal] WITH CHECK ADD CONSTRAINT [fk_proposal_customer_id] FOREIGN KEY([customer_id])
REFERENCES [dbo].[customer] ([id])
GO
ALTER TABLE [dbo].[proposal] CHECK CONSTRAINT [fk_proposal_customer_id]
GO

ALTER TABLE [dbo].[proposal] WITH CHECK ADD CONSTRAINT [fk_proposal_seller_id] FOREIGN KEY([seller_id])
REFERENCES [dbo].[seller] ([id])
GO
ALTER TABLE [dbo].[proposal] CHECK CONSTRAINT [fk_proposal_seller_id]
GO

ALTER TABLE [dbo].[proposal] WITH CHECK ADD CONSTRAINT [fk_proposal_model_type_id] FOREIGN KEY([model_type_id])
REFERENCES [dbo].[model_type] ([id])
GO
ALTER TABLE [dbo].[proposal] CHECK CONSTRAINT [fk_proposal_model_type_id]
GO

ALTER TABLE [dbo].[proposal] WITH CHECK ADD CONSTRAINT [fk_proposal_proposal_type_id] FOREIGN
    KEY([proposal_type_id])
REFERENCES [dbo].[proposal_type] ([id])
GO
ALTER TABLE [dbo].[proposal] CHECK CONSTRAINT [fk_proposal_proposal_type_id]
GO
```

- offer: Tabla para almacenar información sobre las ofertas realizadas a los clientes basadas en las propuestas.
  - id (PK, int): Identificador único de la oferta.
  - proposal_id (FK, int): Clave foránea que referencia la tabla proposal.
  - offer_date (date): Fecha en que se realizó la oferta.
  - offer_amount (decimal): Precio de la oferta.
  - active (bit): Indica si el registro está o no activo.
  - creation_date (DF, datetime): Fecha de creación del registro generada por default.
  - modification_date (TR, datetime): Fecha de modificación del registro actualizada por trigger.
  - io_id (int): Clave de la operación realizada en la BBDD IO.

Código 6.10: Creacion de tabla offer

```
-- Create offer Table
CREATE TABLE [dbo].[offer] (
    id INT IDENTITY(1, 1),
    proposal_id INT,
    offer_date DATE,
    offer_amount DECIMAL(10, 2),
   active BIT,
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_offer] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
     ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[offer] ADD DEFAULT (1) FOR [active]
GO

ALTER TABLE [dbo].[offer] ADD CONSTRAINT [df_offer_creation_date] DEFAULT (getdate()) FOR [creation_date]
GO

ALTER TABLE [dbo].[offer] WITH CHECK ADD CONSTRAINT [fk_offer_proposal_id] FOREIGN KEY([proposal_id])
REFERENCES [dbo].[proposal] ([id])
GO
ALTER TABLE [dbo].[offer] CHECK CONSTRAINT [fk_offer_proposal_id]
GO
```

- sale: Tabla para almacenar información sobre las ventas realizadas a los clientes basadas en las ofertas.
  - id (PK, int): Identificador único de la venta.
  - offer_id (FK, int): Clave foránea que referencia la tabla offer.
  - sale_date (date): Fecha en que se realizó la venta.

- creation_date (DF, datetime): Fecha de creación del registro generada por default.
- modification_date (TR, datetime): Fecha de modificación del registro actualizada por trigger.
- io_id (int): Clave de la operación realizada en la BBDD IO.

Código 6.11: Creacion de tabla offer

```
-- Create sale Table
CREATE TABLE [dbo].[sale] (
    id INT IDENTITY(1, 1),
    offer_id INT,
    sale_date DATE,
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_sale] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
     ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[sale] ADD CONSTRAINT [df_sale_creation_date] DEFAULT (getdate()) FOR [creation_date]
GO

ALTER TABLE [dbo].[sale] WITH CHECK ADD CONSTRAINT [fk_sale_offer_id] FOREIGN KEY([offer_id])
REFERENCES [dbo].[offer] ([id])
GO
ALTER TABLE [dbo].[sale] CHECK CONSTRAINT [fk_sale_offer_id]
GO
```

6.2.3. Creación de triggers

Continuamos con la creación de triggers para la BBDD. La definición es la misma para todas las tablas, se crea un desencadenante que al actualizar cualquier columna, actualiza el valor de la columna *modification_date* por la fecha y hora actuales. Se detalla a continuación el *Código 6.12-6.21* de creación.

Código 6.12: Creacion trigger brand_type

```
-- Create brand_type Trigger
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TRIGGER [dbo].[tr_data_update_brand_type] ON [dbo].[brand_type]
   AFTER UPDATE
   AS
   BEGIN
      -- Continua el trigger SOLO si hay resultados
      IF (@@ROWCOUNT = 0) RETURN;

      -- Ejecuta el trigger una sola vez (evita la recursividad)
      IF (TRIGGER_NESTLEVEL() > 1) RETURN;

      UPDATE dbo.brand_type SET
         modification_date = GETDATE()
      WHERE id IN (SELECT id FROM inserted)
   END
GO

ALTER TABLE [dbo].[brand_type] ENABLE TRIGGER [tr_data_update_brand_type]
```

```
25 GO
```

Código 6.13: Creacion trigger fuel_type

```
1  -- Create fuel_type Trigger
2  SET ANSI_NULLS ON
3  GO
4
5  SET QUOTED_IDENTIFIER ON
6  GO
7
8  CREATE TRIGGER [dbo].[tr_data_update_fuel_type] ON [dbo].[fuel_type]
9     AFTER UPDATE
10    AS
11    BEGIN
12       -- Continua el trigger SOLO si hay resultados
13       IF (@@ROWCOUNT = 0) RETURN;
14
15       -- Ejecuta el trigger una sola vez (evita la recursividad)
16       IF (TRIGGER_NESTLEVEL() > 1) RETURN;
17
18       UPDATE dbo.fuel_type SET
19          modification_date = GETDATE()
20       WHERE id IN (SELECT id FROM inserted)
21    END
22 GO
23
24 ALTER TABLE [dbo].[fuel_type] ENABLE TRIGGER [tr_data_update_fuel_type]
25 GO
```

Código 6.14: Creacion trigger model_type

```
1  -- Create model_type Trigger
2  SET ANSI_NULLS ON
3  GO
4
5  SET QUOTED_IDENTIFIER ON
6  GO
7
8  CREATE TRIGGER [dbo].[tr_data_update_model_type] ON [dbo].[model_type]
9     AFTER UPDATE
10    AS
11    BEGIN
12       -- Continua el trigger SOLO si hay resultados
13       IF (@@ROWCOUNT = 0) RETURN;
14
15       -- Ejecuta el trigger una sola vez (evita la recursividad)
16       IF (TRIGGER_NESTLEVEL() > 1) RETURN;
17
18       UPDATE dbo.model_type SET
19          modification_date = GETDATE()
20       WHERE id IN (SELECT id FROM inserted)
21    END
22 GO
```

Código 6.15: Creacion trigger proposal_type

```
1  -- Create proposal_type Trigger
2  SET ANSI_NULLS ON
3  GO
4
5  SET QUOTED_IDENTIFIER ON
6  GO
7
8  CREATE TRIGGER [dbo].[tr_data_update_proposal_type] ON [dbo].[proposal_type]
9     AFTER UPDATE
10    AS
11    BEGIN
12       -- Continua el trigger SOLO si hay resultados
13       IF (@@ROWCOUNT = 0) RETURN;
14
```

```
    -- Ejecuta el trigger una sola vez (evita la recursividad)
    IF (TRIGGER_NESTLEVEL() > 1) RETURN;

    UPDATE dbo.proposal_type SET
       modification_date = GETDATE()
    WHERE id IN (SELECT id FROM inserted)
  END
GO
```

Código 6.16: Creacion trigger concessionare

```
-- Create concessionare Trigger
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TRIGGER [dbo].[tr_data_update_concessionaire] ON [dbo].[concessionaire]
  AFTER UPDATE
  AS
  BEGIN
    -- Continua el trigger SOLO si hay resultados
    IF (@@ROWCOUNT = 0) RETURN;

    -- Ejecuta el trigger una sola vez (evita la recursividad)
    IF (TRIGGER_NESTLEVEL() > 1) RETURN;

    UPDATE dbo.concessionaire SET
       modification_date = GETDATE()
    WHERE id IN (SELECT id FROM inserted)
  END
GO
```

Código 6.17: Creacion trigger customer

```
-- Create customer Trigger
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TRIGGER [dbo].[tr_data_update_customer] ON [dbo].[customer]
  AFTER UPDATE
  AS
  BEGIN
    -- Continua el trigger SOLO si hay resultados
    IF (@@ROWCOUNT = 0) RETURN;

    -- Ejecuta el trigger una sola vez (evita la recursividad)
    IF (TRIGGER_NESTLEVEL() > 1) RETURN;

    UPDATE dbo.customer SET
       modification_date = GETDATE()
    WHERE id IN (SELECT id FROM inserted)
  END
GO
```

Código 6.18: Creacion trigger seller

```
-- Create seller Trigger
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TRIGGER [dbo].[tr_data_update_seller] ON [dbo].[seller]
  AFTER UPDATE
  AS
```

```
   BEGIN
      -- Continua el trigger SOLO si hay resultados
      IF (@@ROWCOUNT = 0) RETURN;

      -- Ejecuta el trigger una sola vez (evita la recursividad)
      IF (TRIGGER_NESTLEVEL() > 1) RETURN;

      UPDATE dbo.seller SET
         modification_date = GETDATE()
      WHERE id IN (SELECT id FROM inserted)
   END
GO
```

Código 6.19: Creacion trigger proposal

```
-- Create proposal Trigger
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TRIGGER [dbo].[tr_data_update_proposal] ON [dbo].[proposal]
   AFTER UPDATE
   AS
   BEGIN
      -- Continua el trigger SOLO si hay resultados
      IF (@@ROWCOUNT = 0) RETURN;

      -- Ejecuta el trigger una sola vez (evita la recursividad)
      IF (TRIGGER_NESTLEVEL() > 1) RETURN;

      UPDATE dbo.proposal SET
         modification_date = GETDATE()
      WHERE id IN (SELECT id FROM inserted)
   END
GO
```

Código 6.20: Creacion trigger offer

```
-- Create offer Trigger
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TRIGGER [dbo].[tr_data_update_offer] ON [dbo].[offer]
   AFTER UPDATE
   AS
   BEGIN
      -- Continua el trigger SOLO si hay resultados
      IF (@@ROWCOUNT = 0) RETURN;

      -- Ejecuta el trigger una sola vez (evita la recursividad)
      IF (TRIGGER_NESTLEVEL() > 1) RETURN;

      UPDATE dbo.offer SET
         modification_date = GETDATE()
      WHERE id IN (SELECT id FROM inserted)
   END
GO
```

Código 6.21: Creacion trigger sale

```
-- Create sale Trigger
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
```

```
7
8  CREATE TRIGGER [dbo].[tr_data_update_sale] ON [dbo].[sale]
9     AFTER UPDATE
10    AS
11    BEGIN
12       -- Continua el trigger SOLO si hay resultados
13       IF (@@ROWCOUNT = 0) RETURN;
14
15       -- Ejecuta el trigger una sola vez (evita la recursividad)
16       IF (TRIGGER_NESTLEVEL() > 1) RETURN;
17
18       UPDATE dbo.sale SET
19          modification_date = GETDATE()
20       WHERE id IN (SELECT id FROM inserted)
21    END
22 GO
```

6.2.4. Generación de datos de prueba

Por último, se puebla la BBDD con datos de prueba. Es importante destacar que toda la información de carácter personal es totalmente ficticia, la cual se ha generado con el único propósito de aportarnos flexibilidad a la hora de desarrollar los tests. A modo de resumen se va a insertar la siguiente información por cada tabla.

- 2 marcas de coche.
- 4 tipos de combustible.
- 22 modelos.
- 4 tipos de propuesta.
- 5 concesionarios.
- 20 clientes.
- 20 vendedores.
- 50 propuestas.
- 40 ofertas.
- 30 ventas.

Se aportan previamente dos scripts, por un lado el *Código 6.22* de borrado ordenado de información en caso de ser necesario realizar la limpieza de los objetos, y por otro el *Código 6.23* con una consulta básica de cada tabla.

Código 6.22: Borrado de información en VehicleSales

```
1  /*
2  DELETE FROM [dbo].[sale] DBCC CHECKIDENT ('[dbo].[sale]',RESEED, 0)
3  DELETE FROM [dbo].[offer] DBCC CHECKIDENT ('[dbo].[offer]',RESEED, 0)
4  DELETE FROM [dbo].[proposal] DBCC CHECKIDENT ('[dbo].[proposal]',RESEED, 0)
5  DELETE FROM [dbo].[seller] DBCC CHECKIDENT ('[dbo].[seller]',RESEED, 0)
6  DELETE FROM [dbo].[customer] DBCC CHECKIDENT ('[dbo].[customer]',RESEED, 0)
7  DELETE FROM [dbo].[concessionaire] DBCC CHECKIDENT ('[dbo].[concessionaire]',RESEED, 0)
8  DELETE FROM [dbo].[proposal_type] DBCC CHECKIDENT ('[dbo].[proposal_type]',RESEED, 0)
9  DELETE FROM [dbo].[model_type] DBCC CHECKIDENT ('[dbo].[model_type]',RESEED, 0)
10 DELETE FROM [dbo].[fuel_type] DBCC CHECKIDENT ('[dbo].[fuel_type]',RESEED, 0)
```

```
DELETE FROM [dbo].[brand_type] DBCC CHECKIDENT ('[dbo].[brand_type]',RESEED, 0)
*/
```

Código 6.23: Consulta de información en VehicleSales

```
SELECT * FROM [dbo].[sale]
SELECT * FROM [dbo].[offer]
SELECT * FROM [dbo].[proposal]
SELECT * FROM [dbo].[seller]
SELECT * FROM [dbo].[customer]
SELECT * FROM [dbo].[concessionaire]
SELECT * FROM [dbo].[proposal_type]
SELECT * FROM [dbo].[model_type]
SELECT * FROM [dbo].[fuel_type]
SELECT * FROM [dbo].[brand_type]
```

A continuación se muestra el detalle de la información generada. Es importante destacar que estas operaciones se deberían ejecutar mediante las cláusulas *BEGIN TRAN, COMMIT o ROLLBACK* [11] como aparece en el el *Código 6.24-6.33* de creación.

Código 6.24: Generación de información para la tabla brand_type

```
-- [dbo].[brand_type]
-- Inserta BMW/Mini
INSERT INTO [dbo].[brand_type] ([name], active, creation_date)
VALUES ('BMW', 1, GETDATE()),
       ('Mini', 1, GETDATE());
```

Código 6.25: Generación de información para la tabla fuel_type

```
-- [dbo].[fuel_type]
-- Insert tipos de combustible
INSERT INTO [dbo].[fuel_type] ([name], active, creation_date)
VALUES ('Gasolina', 1, GETDATE()),
       ('Diésel', 1, GETDATE()),
       ('Eléctrico', 1, GETDATE()),
       ('Híbrido', 1, GETDATE());
```

Código 6.26: Generación de información para la tabla model_type

```
-- [dbo].[model_type]
-- Inserta modelos de BMW/Mini
INSERT INTO [dbo].[model_type] ([name], [brand_type_id], [fuel_type_id], active, creation_date)
VALUES ('Serie 1', 1, 1, 1, GETDATE()),
       ('Serie 2', 1, 4, 1, GETDATE()),
       ('Serie 3', 1, 1, 1, GETDATE()),
       ('Serie 4', 1, 2, 1, GETDATE()),
       ('Serie 5', 1, 2, 1, GETDATE()),
       ('Serie 6', 1, 1, 1, GETDATE()),
       ('Serie 7', 1, 1, 1, GETDATE()),
       ('Serie 8', 1, 1, 1, GETDATE()),
      ('I3', 1, 3, 1, GETDATE()),
      ('I8', 1, 3, 1, GETDATE()),
       ('X1', 1, 2, 1, GETDATE()),
       ('X2', 1, 2, 1, GETDATE()),
       ('X3', 1, 1, 1, GETDATE()),
       ('X4', 1, 4, 1, GETDATE()),
       ('X5', 1, 4, 1, GETDATE()),
       ('X6', 1, 1, 1, GETDATE()),
       ('X7', 1, 1, 1, GETDATE()),
      ('Cooper', 2, 1, 1, GETDATE()),
       ('Countryman', 2, 2, 1, GETDATE()),
       ('Clubman', 2, 3, 1, GETDATE()),
       ('Convertible', 2, 1, 1, GETDATE()),
       ('Paceman', 2, 4, 1, GETDATE());
```

Código 6.27: Generación de información para la tabla proposal_type

```sql
-- [dbo].[proposal_type]
-- Insertar tipos de propuestas
INSERT INTO [dbo].[proposal_type] ([name], active, creation_date)
VALUES ('Nuevo', 1, GETDATE()),
       ('Seminuevo', 1, GETDATE()),
       ('Renting', 1, GETDATE()),
       ('Leasing', 1, GETDATE());
```

Código 6.28: Generación de información para la tabla concessionare

```sql
-- [dbo].[concessionaire]
-- Inserta concesionarios de BMW
INSERT INTO [dbo].[concessionaire] ([name], [address], [city], [country], [postal_code], active, creation_date)
VALUES ('BYmyCAR Madrid', 'Avenida de Burgos, 133', 'Madrid', 'España', '28050', 1, GETDATE()),
       ('Avilcar', 'Jorge de Santayana, 74', 'Ávila', 'España', '05004', 1, GETDATE()),
      ('Vehinter', NULL, NULL, 'España', NULL, 1, GETDATE()),
       ('Movitransa', 'Avenida Tío Pepe, 55', 'Jerez de la Frontera', 'España', '11407', 1, GETDATE()),
       ('Barcelona Premium', 'c/ Entença, 324-326', 'Barcelona', 'España', NULL, 1, GETDATE());
```

Código 6.29: Generación de información para la tabla customer

```sql
-- [dbo].[customer]
-- Inserta clientes ficticios
INSERT INTO [dbo].[customer] ([name], phone, email, creation_date) VALUES
    ('Laura Rodríguez', '111222333', 'laura@example.com', GETDATE()),
    ('Carlos Gómez', '444555666', 'carlos@example.com', GETDATE()),
    ('Sara López', '777888999', 'sara@example.com', GETDATE()),
    ('Javier Fernández', '123456789', 'javier@example.com', GETDATE()),
    ('Lucía Martínez', '987654321', 'lucia@example.com', GETDATE()),
    ('Alejandro Pérez', '456789123', 'alejandro@example.com', GETDATE()),
    ('Elena García', '789123456', 'elena@example.com', GETDATE()),
    ('Pablo Ruiz', '111333555', 'pablo@example.com', GETDATE()),
    ('Carmen Sánchez', '444666888', 'carmen@example.com', GETDATE()),
    ('Daniel Díaz', '777999111', 'daniel@example.com', GETDATE()),
    ('Marina Vázquez', '222444666', 'marina@example.com', GETDATE()),
    ('Mario Torres', '555777999', 'mario@example.com', GETDATE()),
    ('Clara Romero', '888111333', 'clara@example.com', GETDATE()),
    ('Antonio Navarro', '333555777', 'antonio@example.com', GETDATE()),
    ('Isabel Jiménez', '666888111', 'isabel@example.com', GETDATE()),
    ('Sergio Gutiérrez', '999111333', 'sergio@example.com', GETDATE()),
    ('Paula Castro', '222555888', 'paula@example.com', GETDATE()),
    ('Alberto Molina', '555888222', 'alberto@example.com', GETDATE()),
    ('Cristina Ortega', '888222555', 'cristina@example.com', GETDATE()),
    ('David Delgado', '111444777', 'david@example.com', GETDATE());
```

Código 6.30: Generación de información para la tabla seller

```sql
-- [dbo].[seller]
-- Insertar vendedores ficticios
INSERT INTO [dbo].[seller] ([name], phone, email, concessionaire_id, active, creation_date) VALUES
    ('Ana Martín', '111222333', 'ana@example.com', 1, 1, GETDATE()),
    ('Pedro Sánchez', '444555666', 'pedro@example.com', 1, 1, GETDATE()),
    ('María García', '777888999', 'maria@example.com', 1, 1, GETDATE()),
    ('Juan López', '123456789', 'juan@example.com', 1, 1, GETDATE()),
    ('Lucía Pérez', '987654321', 'lucia@example.com', 1, 1, GETDATE()),
    ('Carlos Gómez', '456789123', 'carlos@example.com', 2, 1, GETDATE()),
    ('Elena Rodríguez', '789123456', 'elena@example.com', 2, 1, GETDATE()),
    ('Marta Martínez', '111333555', 'marta@example.com', 3, 1, GETDATE()),
    ('David Díaz', '444666888', 'david@example.com', 3, 1, GETDATE()),
    ('Laura Ruiz', '777999111', 'laura@example.com', 4, 1, GETDATE()),
    ('Javier Torres', '222444666', 'javier@example.com', 4, 1, GETDATE()),
    ('Sara Jiménez', '555777999', 'sara@example.com', 4, 1, GETDATE()),
    ('Pablo Romero', '888111333', 'pablo@example.com', 4, 1, GETDATE()),
    ('Cristina Navarro', '333555777', 'cristina@example.com', 5, 1, GETDATE()),
    ('Alberto Ortega', '666888111', 'alberto@example.com', 5, 1, GETDATE()),
    ('Marta Gutiérrez', '999111333', 'marta@example.com', 5, 1, GETDATE()),
    ('Diego Castro', '222555888', 'diego@example.com', 5, 1, GETDATE()),
    ('Sofía Molina', '555888222', 'sofia@example.com', 5, 1, GETDATE()),
    ('José Ortega', '888222555', 'jose@example.com', 5, 1, GETDATE()),
    ('Ana Delgado', '111444777', 'ana@example.com', 5, 1, GETDATE());
```

Código 6.31: Generación de información para la tabla proposal

```
-- [dbo].[proposal]
-- Insertar propuestas ficticias con foreign keys
INSERT INTO [dbo].[proposal] (customer_id, seller_id, model_type_id, proposal_date, proposal_amount,
    proposal_type_id, active, creation_date)
VALUES
   (1, 1, 1, '2023-01-01', 30000.00, 1, 1, GETDATE()),
   (2, 1, 2, '2023-01-02', 25000.00, 1, 1, GETDATE()),
   (3, 1, 3, '2023-01-03', 20000.00, 2, 1, GETDATE()),
   (4, 3, 4, '2023-01-04', 38000.00, 3, 1, GETDATE()),
   (5, 3, 5, '2023-01-05', 22000.00, 4, 1, GETDATE()),
   (6, 4, 6, '2023-01-06', 46000.00, 1, 1, GETDATE()),
   (7, 4, 7, '2023-01-07', 59000.00, 1, 1, GETDATE()),
   (8, 4, 8, '2023-01-08', 22000.00, 2, 1, GETDATE()),
   (9, 4, 9, '2023-01-09', 17000.00, 2, 1, GETDATE()),
   (10, 4, 10, '2023-01-10', 33000.00, 2, 1, GETDATE()),
   (11, 4, 11, '2023-01-11', 34000.00, 4, 1, GETDATE()),
   (12, 5, 1, '2023-01-12', 25000.00, 4, 1, GETDATE()),
   (13, 6, 14, '2023-01-13', 23000.00, 3, 1, GETDATE()),
   (14, 6, 19, '2023-01-14', 24000.00, 2, 1, GETDATE()),
   (15, 6, 21, '2023-01-15', 26000.00, 1, 1, GETDATE()),
   (16, 7, 20, '2023-01-16', 27000.00, 3, 1, GETDATE()),
   (17, 7, 20, '2023-01-17', 28000.00, 3, 1, GETDATE()),
   (18, 8, 11, '2023-01-18', 29000.00, 3, 1, GETDATE()),
   (19, 8, 10, '2023-01-19', 30000.00, 1, 1, GETDATE()),
   (20, 9, 12, '2023-01-20', 31000.00, 2, 1, GETDATE()),
   (1, 9, 1, '2023-01-21', 32000.00, 2, 1, GETDATE()),
   (2, 9, 2, '2023-01-22', 33000.00, 2, 1, GETDATE()),
   (3, 10, 3, '2023-01-23', 34000.00, 4, 1, GETDATE()),
   (4, 10, 4, '2023-01-24', 35000.00, 4, 1, GETDATE()),
   (1, 10, 3, '2023-01-25', 36000.00, 4, 1, GETDATE()),
   (2, 10, 3, '2023-01-26', 37000.00, 1, 1, GETDATE()),
   (3, 10, 2, '2023-01-27', 38000.00, 1, 1, GETDATE()),
   (4, 12, 2, '2023-01-28', 39000.00, 1, 1, GETDATE()),
   (8, 13, 1, '2023-01-29', 40000.00, 1, 1, GETDATE()),
   (8, 14, 1, '2023-01-30', 41000.00, 1, 1, GETDATE()),
   (8, 14, 11, '2023-01-31', 42000.00, 2, 1, GETDATE()),
   (10, 15, 10, '2023-02-01', 43000.00, 3, 1, GETDATE()),
   (11, 15, 10, '2023-02-02', 44000.00, 2, 1, GETDATE()),
   (12, 15, 9, '2023-02-03', 45000.00, 2, 1, GETDATE()),
   (13, 15, 8, '2023-02-04', 46000.00, 4, 1, GETDATE()),
   (14, 15, 12, '2023-02-05', 47000.00, 4, 1, GETDATE()),
   (14, 15, 13, '2023-02-06', 48000.00, 1, 1, GETDATE()),
   (14, 15, 15, '2023-02-07', 49000.00, 1, 1, GETDATE()),
   (14, 16, 5, '2023-02-08', 50000.00, 3, 1, GETDATE()),
   (14, 18, 6, '2023-02-09', 51000.00, 1, 1, GETDATE()),
   (15, 18, 7, '2023-02-10', 52000.00, 1, 1, GETDATE()),
   (16, 18, 20, '2023-02-11', 53000.00, 2, 1, GETDATE()),
   (1, 19, 21, '2023-02-12', 54000.00, 3, 1, GETDATE()),
   (18, 19, 22, '2023-02-13', 55000.00, 4, 1, GETDATE()),
   (19, 20, 9, '2023-02-14', 56000.00, 2, 1, GETDATE()),
   (20, 20, 6, '2023-02-15', 57000.00, 2, 1, GETDATE()),
   (11, 20, 5, '2023-02-16', 58000.00, 2, 1, GETDATE()),
   (3, 20, 4, '2023-02-17', 59000.00, 1, 1, GETDATE()),
   (3, 20, 1, '2023-02-18', 60000.00, 1, 1, GETDATE()),
   (7, 20, 1, '2023-02-19', 61000.00, 3, 1, GETDATE());
```

Código 6.32: Generación de información para la tabla offer

```
-- [dbo].[offer]
-- Insertar ofertas ficticias
INSERT INTO [dbo].[offer] (proposal_id, offer_date, offer_amount, active, creation_date) VALUES
    (1, '2023-03-15', 24000.00, 1, GETDATE()),
    (2, '2023-03-16', 27000.00, 1, GETDATE()),
    (3, '2023-03-17', 29000.00, 1, GETDATE()),
    (4, '2023-03-18', 26000.00, 1, GETDATE()),
    (5, '2023-03-19', 31000.00, 1, GETDATE()),
    (6, '2023-03-20', 28000.00, 1, GETDATE()),
    (7, '2023-03-21', 25000.00, 1, GETDATE()),
    (8, '2023-03-22', 32000.00, 1, GETDATE()),
    (9, '2023-03-23', 27000.00, 1, GETDATE()),
    (10, '2023-03-24', 30000.00, 1, GETDATE()),
```

```
    (11, '2023-03-25', 29000.00, 1, GETDATE()),
    (12, '2023-03-26', 26000.00, 1, GETDATE()),
    (13, '2023-03-27', 28000.00, 1, GETDATE()),
    (14, '2023-03-28', 31000.00, 1, GETDATE()),
    (15, '2023-03-29', 24000.00, 1, GETDATE()),
    (16, '2023-03-30', 33000.00, 1, GETDATE()),
    (17, '2023-03-31', 27000.00, 1, GETDATE()),
    (18, '2023-04-01', 30000.00, 1, GETDATE()),
    (19, '2023-04-02', 28000.00, 1, GETDATE()),
    (20, '2023-04-03', 29000.00, 1, GETDATE()),
    (21, '2023-04-04', 26000.00, 1, GETDATE()),
    (22, '2023-04-05', 30000.00, 1, GETDATE()),
    (23, '2023-04-06', 27000.00, 1, GETDATE()),
    (24, '2023-04-07', 29000.00, 1, GETDATE()),
    (25, '2023-04-08', 31000.00, 1, GETDATE()),
    (26, '2023-04-09', 28000.00, 1, GETDATE()),
    (27, '2023-04-10', 26000.00, 1, GETDATE()),
    (28, '2023-04-11', 33000.00, 1, GETDATE()),
    (29, '2023-04-12', 27000.00, 1, GETDATE()),
    (30, '2023-04-13', 30000.00, 1, GETDATE()),
    (31, '2023-04-14', 29000.00, 1, GETDATE()),
    (32, '2023-04-15', 26000.00, 1, GETDATE()),
    (33, '2023-04-16', 28000.00, 1, GETDATE()),
    (34, '2023-04-17', 31000.00, 1, GETDATE()),
    (35, '2023-04-18', 24000.00, 1, GETDATE()),
    (36, '2023-04-19', 32000.00, 1, GETDATE()),
    (37, '2023-04-20', 27000.00, 1, GETDATE()),
    (38, '2023-04-21', 30000.00, 1, GETDATE()),
    (39, '2023-04-22', 28000.00, 1, GETDATE()),
    (40, '2023-04-23', 29000.00, 1, GETDATE());
```

Código 6.33: Generación de información para la tabla sale

```
-- [dbo].[sale]
-- Insertar ventas ficticias
INSERT INTO dbo.sale (offer_id, sale_date, creation_date) VALUES
   (1, '2023-05-01', GETDATE()),
   (2, '2023-05-02', GETDATE()),
   (3, '2023-05-03', GETDATE()),
   (4, '2023-05-04', GETDATE()),
   (5, '2023-05-05', GETDATE()),
   (6, '2023-05-06', GETDATE()),
   (7, '2023-05-07', GETDATE()),
   (8, '2023-05-08', GETDATE()),
   (9, '2023-05-09', GETDATE()),
   (10, '2023-05-10', GETDATE()),
   (11, '2023-05-11', GETDATE()),
   (12, '2023-05-12', GETDATE()),
   (13, '2023-05-13', GETDATE()),
   (14, '2023-06-14', GETDATE()),
   (15, '2023-06-15', GETDATE()),
   (16, '2023-06-16', GETDATE()),
   (17, '2023-06-17', GETDATE()),
   (18, '2023-06-18', GETDATE()),
   (19, '2023-06-19', GETDATE()),
   (20, '2023-06-20', GETDATE()),
   (21, '2023-06-21', GETDATE()),
   (22, '2023-06-22', GETDATE()),
   (23, '2023-06-23', GETDATE()),
   (24, '2023-06-24', GETDATE()),
   (25, '2023-06-25', GETDATE()),
   (26, '2023-06-26', GETDATE()),
   (27, '2023-06-27', GETDATE()),
   (28, '2023-06-28', GETDATE()),
   (29, '2023-06-29', GETDATE()),
   (30, '2023-06-30', GETDATE());
```

6.2.5. Modelo entidad-relación

Tras la ejecución de los scripts mostrados durante los apartados anteriores, ya dispondremos de la BBDD *VehicleSales* completamente preparada para su uso. Adicionalmente, se ha generado el diagrama entidad-relación en SQL Server, el cual queda como se muestra en la *Figura 6.1.*

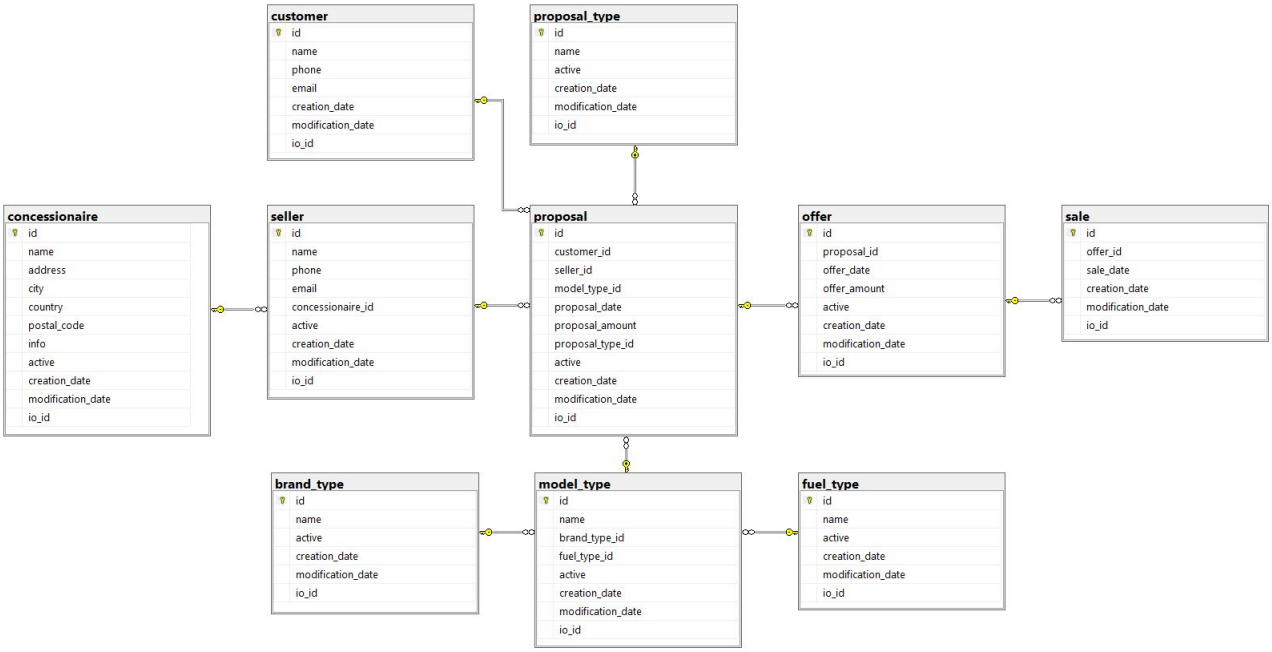


Figura 6.1: Diagrama entidad-relación VehicleSales

6.3. Caso de uso 1: Input - NewDealers csv

El primer caso de uso del proyecto *Input - NewDealers csv* presenta un proceso input simple con un fichero en formato *CSV*, el cual será procesado por nuestra solución IO para persistir la información en la BBDD *VehicleSales*.

Mediante este caso de uso se verifican los requisitos de *Gestión de Datos de Entrada y Salida (1-6)*, *Orquestación de Procesos (1 y 3)*, *Funcionalidades Específicas (1, 2 y 5)* y *Monitorización y Trazabilidad (1)*.

6.3.1. Definición del proceso

El cliente realiza una petición por la cual pretende mantener actualizada la información de sus concesionarios, para ello nos define los siguientes detalles funcionales.

- Se depositará un fichero en nuestro servidor FTP en el directorio de entrada.
- El fichero contendrá información de nuevos concesionarios, así como modificaciones sobre registros existentes.
- Para el caso de las modificaciones necesita que los valores de nuestra BBDD sean reemplazados.

- La nomenclatura del fichero será *NombreCliente_NewDealers_Fecha_Hora*.
- El formato del fichero será *CSV* donde las columnas vienen entrecomilladas con ’’ y separadas con ’,’.
- La información de columnas serán el nombre, dirección, ciudad, país, código postal, además de dos columnas auxiliares las cuales podrán contener texto con detalle adicional en caso necesario.

6.3.2. Configuración del proceso

Una vez se recibe la definición del proceso comenzamos con su configuración. Para ello, el primer punto a revisar es el formato del fichero el cual podemos ver en la *Figura 6.2*.

BMW_NewDealers_20240313_1415.csv

```
"nombre","direccion","ciudad","pais","codigopostal","auxiliar1","auxiliar2"
"Bertolín","Avda. General Avilés, 68","Valencia","España","46015","34-963898974","bertolin@bertolin.es"
"VehÍnter","Ctra. Madrid-Toledo","Getafe","España","28905","",""
"Barcelonä Premium","Carrer d'Entença, 332","Barcelona","España","08029","34-933319800","marketing@barcelonapremium.net.bmw.es"
"Celtamotor","Ctra.de Camposancos, 115","Vigo","España","36213","34-986213645",""
```

Figura 6.2: Formato de fichero BMW_NewDealers csv

Posteriormente generamos la plantilla *DAT* para que pueda ser procesado mediante la herramienta *Ingesta de archivos en BBDD* 5.2.2 detallada en el capítulo 5. Como muestra el *Código 6.34*, esta contiene las mismas columnas que el fichero además de los separadores de columna indicados.

Código 6.34: Input NewDealers.fmt

```
12.0
7
1   SQLCHAR   0   0   ",\""   1 Nombre          SQL_Latin1_General_CP1_CI_AS
2   SQLCHAR   0   0   ",\""   2 Direccion       SQL_Latin1_General_CP1_CI_AS
3   SQLCHAR   0   0   ",\""   3 Ciudad          SQL_Latin1_General_CP1_CI_AS
4   SQLCHAR   0   0   ",\""   4 Pais         SQL_Latin1_General_CP1_CI_AS
5   SQLCHAR   0   0   ",\""   5 CodigoPostal    SQL_Latin1_General_CP1_CI_AS
6   SQLCHAR   0   0   ",\""   6 Auxiliar1       SQL_Latin1_General_CP1_CI_AS
7   SQLCHAR   0   0   "\n"    7 Auxiliar2       SQL_Latin1_General_CP1_CI_AS
```

A partir de este punto pasamos al apartado de la BBDD IO donde el primer paso consiste en insertar el registro con la configuración maestra en la tabla *IO.config.master_process_type*, el cual contiene el detalle de la *Figura 6.3*.

Results Messages

	id	customer	name	description	active_flag	ingestion_flag	transformation_flag	persist_flag	output_flag	stop_on_warning
1	1	BMW	NewDealers_Input	Incorpora los nuevos concesionarios y actualiza ...	1	1	1	1	0	1

Figura 6.3: Registro del proceso en la tabla master_process_type

Tras la inserción del registro de configuración maestra, pasamos a dar de alta los registros con las fases e información detallada en la tabla *IO.config.detail_process_type* la cual tendrá la información que se muestra en la *Figura 6.4*.

- Fase de ingesta con dos steps, el primero para descargar el fichero del FTP a nuestro directorio local y el segundo para cargar la información en la tabla *IO.input.BMW_NewDealers*.
- Fase de delivery para insertar un registro de seguimiento en la tabla *IO.load.delivery*.

- Fase de transformación con dos steps, el primero inserta la información nueva de los concesionarios y el segundo la información a actualizar, ambos en la tabla intermedia *IO.load.concessionaire*.

- Fase de persistencia con dos steps, el primero inserta y el segundo actualiza la información de concesionarios en la tabla de destino final *VehicleSales.dbo.concessionaire*.

	id	master_process_type_id	phase	step	entity	name	description	source_type
1	100	1	INGESTION	1	insert_BMW_NewDealers	Ingesta NewDealers	Recoge del FTP y mueve al Input el fichero BMW_NewDealers	FTP
2	110	1	INGESTION	2	load_file_input_table	concessionaire	Archivos de concessionaire	FILE
3	120	1	DELIVERY	1	new_delivery	Alta delivery concessionaire	Alta de una nueva delivery de concessionaire por cada marca	BBDD
4	130	1	TRANSFORMATION	1	insert_concessionaire	Insert load concessionaire	Incorporación de los datos calculando las operaciones	BBDD
5	140	1	TRANSFORMATION	2	update_concessionaire	Update load concessionaire	Incorporación de los datos calculando las operaciones	BBDD
6	150	1	PERSISTENCE	1	insert_concessionaire	Insert VehicleSales concessionaire	Incorporación de los datos en destino insert concessionaire	BBDD
7	160	1	PERSISTENCE	2	update_concessionaire	Update VehicleSales concessionaire	Incorporación de los datos en destino update concessionaire	BBDD

Figura 6.4: Registros del proceso en la tabla detail_process_type

Es importante destacar que la tabla intermedia *IO.load.concessionaire* deberá ser generada por nosotros y estar preparada previa a la ejecución del proceso. Para ver en detalle la inserción de los registros de configuración así como la generación de la tabla intermedia, ver el script *1 Input - NewDealers csv.sql* C.35 del *Código fuente* C.

Adicionalmente, cada step de la fase delivery, transformación y persistencia contiene un procedimiento almacenado con la lógica a seguir, cuya ejecución se produce desde la columna *source_table_query* de la tabla *IO.config.detail_process_type*. Con respecto al código de los procedimientos almacenados, ver el apartado *Procedimientos BMW_NewDealers* C del *Código fuente* C.

6.3.3. Ejecución y resultados del proceso

Tras la configuración del proceso, pasaremos por último a su ejecución y contrastaremos los resultados persistidos acordes con el fichero aportado.

La situación original de la tabla *VehicleSales.dbo.concessionaire* es la que se muestra en la *Figura 6.5*.

	id	name	address	city	country	postal_code	info	active	creation_date	modification_date	io_id
1	1	BYmyCAR Madrid	Avenida de Burgos, 133	Madrid	España	28050	NULL	1	2024-03-15 20:00:22.407	NULL	NULL
2	2	Avilcar	Jorge de Santayana, 74	Ávila	España	05004	NULL	1	2024-03-15 20:00:22.407	NULL	NULL
3	3	Vehinter	NULL	NULL	España	NULL	NULL	1	2024-03-15 20:00:22.407	2024-04-16 16:05:19.950	NULL
4	4	Movitransa	Avenida Tío Pepe, 55	Jerez de la Frontera	España	11407	NULL	1	2024-03-15 20:00:22.407	NULL	NULL
5	5	Barcelona Premium	c/ Entença, 324-326	Barcelona	España	NULL	NULL	1	2024-03-15 20:00:22.407	2024-04-16 16:05:19.960	NULL

Figura 6.5: Situación original en VehicleSales.dbo.concessionaire

Antes de comenzar con la ejecución del proceso, contrastamos que el fichero se encuentra en el directorio indicado del FTP (*Figura 6.6*).

Tras la revisión ejecutamos el proceso y contrastamos como la información va pasando por las diferentes tablas preparadas para almacenar la información (*Figuras 6.7-6.8*).

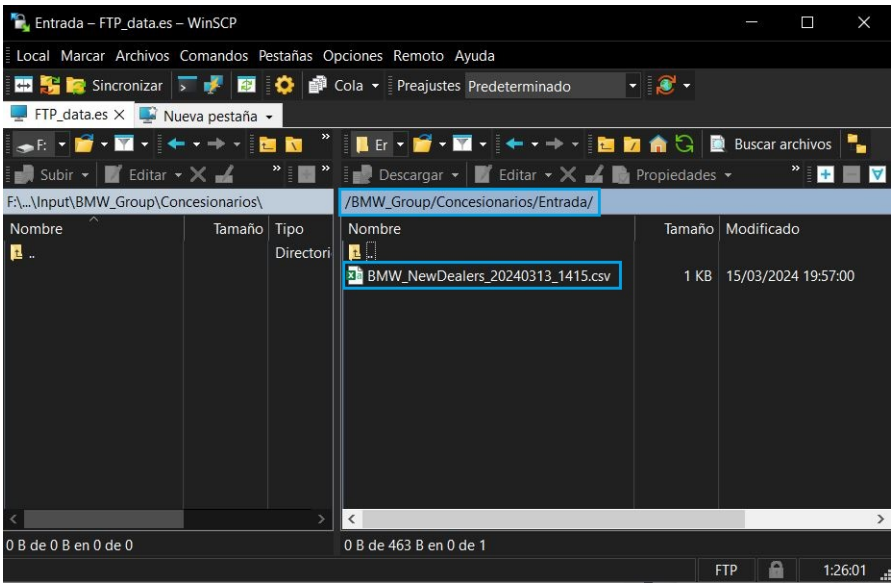


Figura 6.6: Fichero disponible en el directorio de entrada del FTP

Nombre	Direccion	Ciudad	Pais	CodigoPostal	Auxiliar1	Auxiliar2
Bertolín	Avda. General Avilés, 68	Valencia	España	46015	34-963898974	bertolin@bertolin.es
VehÍnter	Ctra. Madrid-Toledo	Getafe	España	28905		
Barcelonä Premium	Carrer d'Entença, 332	Barcelona	España	08029	34-933319800	marketing@barcelonapremium.net.bmw.es
Celtamotor	Ctra.de Camposancos, 115	Vigo	España	36213	34-986213645	

Figura 6.7: Registros cargados en IO.input.BMW_NewDealers

	id	delivery_id	mdb_id	name	address	city	country	postal_code	info	active	creation_date	modification_date
1	1	1	NULL	Bertolin	Avda. General Avilés, 68	Valencia	España	46015	34-963898974 - bertolin@bertolin.es	1	NULL	NULL
2	2	1	NULL	Celtamotor	Ctra.de Camposancos, 115	Vigo	España	36213	34-986213645 -	1	NULL	NULL
3	3	1	3	VehÍnter	Ctra. Madrid-Toledo	Getafe	España	28905	-	1	NULL	NULL
4	4	1	5	Barcelonä Premium	Carrer d'Entença, 332	Barcelona	España	08029	34-933319800 - marketing@barcelonapremium.net.bm...	1	NULL	NULL

Figura 6.8: Registros cargados en IO.load.concessionaire

Posteriormente recibimos el correo con los resultados del proceso ejecutado de forma satisfactoria comom muestra la *Figura 6.9*.

[BMW] [SUCCEEDED]: NewDealers_Input execution summary on WIN-SER-AMAROTO

server@data.es
Para amaroto@data.es

Responder | Responder a todos | Reenviar

vi. 26/04/2

Running Server: [WIN-SER-AMAROTO]

Portfolio: [BMW]

Execution summary for process: NewDealers_Input

Process description: Incorpora los nuevos concesionarios y actualiza los existentes

Main summary

STATUS	START TIME	END TIME	TOTAL DURATION	MESSAGE
SUCCEEDED	26/04/2024 14:52:00	26/04/2024 14:52:07	00:00:07:243	

Phase summary

PHASE	STATUS	START TIME	END TIME	TOTAL DURATION
INGESTION	SUCCEEDED	26/04/2024 14:52:00	26/04/2024 14:52:05	00:00:05:657
TRANSFORMATION	SUCCEEDED	26/04/2024 14:52:05	26/04/2024 14:52:06	00:00:00:907
PERSISTENCE	SUCCEEDED	26/04/2024 14:52:06	26/04/2024 14:52:07	00:00:00:680

Detailed execution

PHASE	ENTITY	STATUS	AFFECTED ROWS	START TIME	END TIME	DURATION	MESSAGE
INGESTION	FTP_Download	SUCCEEDED	0	26/04/2024 14:52:01	26/04/2024 14:52:05	00:00:03:990	Files downloaded correctly
INGESTION	[IO].[input].[BMW_NewDealers]	SUCCEEDED	4	26/04/2024 14:52:05	26/04/2024 14:52:05	00:00:00:720	
DELIVERY	NEW DELIVERIES	SUCCEEDED	1	26/04/2024 14:52:06	26/04/2024 14:52:06	00:00:00:163	
TRANSFORMATION	insert_concessionaire	SUCCEEDED	2	26/04/2024 14:52:06	26/04/2024 14:52:06	00:00:00:130	Transformation process has been done correctly
TRANSFORMATION	update_concessionaire	SUCCEEDED	2	26/04/2024 14:52:06	26/04/2024 14:52:06	00:00:00:083	Transformation process has been done correctly
PERSISTENCE	insert_concessionaire	SUCCEEDED	2	26/04/2024 14:52:07	26/04/2024 14:52:07	00:00:00:110	Persistence for delivery [ID]: 1 has been done
PERSISTENCE	update_concessionaire	SUCCEEDED	2	26/04/2024 14:52:07	26/04/2024 14:52:07	00:00:00:117	Persistence for delivery [ID]: 1 has been done

Figura 6.9: Correo recibido del proceso BMW_NewDealers

Adicionalmente, si consultamos la tabla *VehicleSales.dbo.concessionaire* como muestra la *Figura 6.10* podemos apreciar como esta contiene la nueva información procedente del fichero procesado.

	id	name	address	city	country	postal_code	info	active	creation_date	modification_date	io_id
1	1	BYmyCAR Madrid	Avenida de Burgos, 133	Madrid	España	28050	NULL	1	2024-03-15 20:00:22.407	NULL	NULL
2	2	Avilcar	Jorge de Santayana, 74	Ávila	España	05004	NULL	1	2024-03-15 20:00:22.407	NULL	NULL
3	3	Vehinter	Ctra. Madrid-Toledo	Getafe	España	28905	-	1	2024-03-15 20:00:22.407	2024-04-26 14:52:07.510	3
4	4	Movitransa	Avenida Tío Pepe, 55	Jerez de la Frontera	España	11407	NULL	1	2024-03-15 20:00:22.407	NULL	NULL
5	5	Barcelona Premium	Carrer d'Entença, 332	Barcelona	España	08029	34-933319800 - marketing@barcelonapremium.net.bm...	1	2024-03-15 20:00:22.407	2024-04-26 14:52:07.510	4
6	6	Bertolin	Avda. General Avilés, 68	Valencia	España	46015	34-963898974 - bertolin@bertolin.es	1	2024-04-26 14:52:07.363	NULL	1
7	7	Celtamotor	Ctra.de Camposancos, 115	Vigo	España	36213	34-986213645 -	1	2024-04-26 14:52:07.363	NULL	2

Figura 6.10: Situación final en VehicleSales.dbo.concessionaire

Por último, el fichero procesado se archiva en nuestra ruta local y se insertan los datos en la tabla histórica *IO.hist.BMW_NewDealers* (*Figuras 6.11-6.12*).

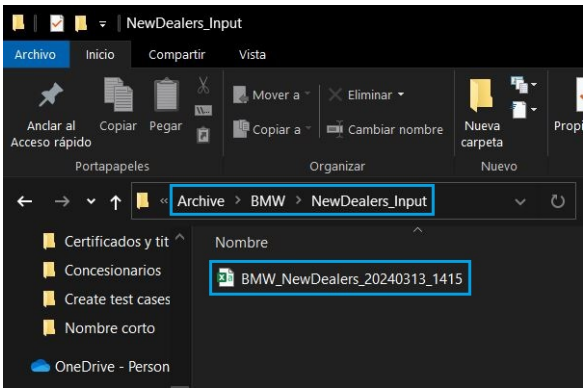


Figura 6.11: Fichero disponible en el directorio archive del proceso

master_process_log_id	Nombre	Direccion	Ciudad	Pais	CodigoPostal	Auxiliar1	Auxiliar2
1	Bertolín	Avda. General Avilés, 68	Valencia	España	46015	34-963898974	bertolin@bertolin.es
1	VehÍnter	Ctra. Madrid-Toledo	Getafe	España	28905		
1	Barcelonä Premium	Carrer d'Entença, 332	Barcelona	España	08029	34-933319800	marketing@barcelonapremium.net.bmw.es
1	Celtamotor	Ctra.de Camposancos, 115	Vigo	España	36213	34-986213645	

Figura 6.12: Información cargada en la tabla IO.hist.BMW_NewDealers

6.4. Caso de uso 2: Output - TopSellers xlsx

El segundo caso de uso del proyecto *Output - TopSellers* presenta un proceso output el cual genera un cuadro de mando completo para el cliente en un formato de salida *XLSX*, donde la información de origen se extrae de la BBDD *VehicleSales* a través de nuestra solución IO.

Mediante este caso de uso se verifican los requisitos de *Gestión de Datos de Entrada y Salida (1-6)*, *Orquestación de Procesos (1 y 3)*, *Funcionalidades Específicas (1-5)* y *Monitorización y Trazabilidad (1)*.

6.4.1. Definición del proceso

El cliente realiza una petición por la cual pretende conocer a los mejores vendedores de la compañía independientemente de la marca, para ello nos define los siguientes detalles funcionales.

- Depositaremos un fichero en nuestro servidor FTP en el directorio de salida específico del proceso para que el cliente pueda recogerlo.
- Adicionalmente solicita el envío del mismo informe a través de correo a los destinatarios indicados.
- El fichero deberá comprimirse mediante contraseña la cual se facilitará al cliente.
- El fichero deberá contener únicamente el listado con los diez mejores vendedores de la compañía, dando elección al equipo a extraer la información que considere de utilidad.
- Desde el equipo se decide extraer la información base del vendedor y el concesionario al que pertenece, así como cifras totales y promedios de las ventas, ofertas y propuestas.
- La nomenclatura del fichero será *NombreGrupo_TopSellers_Fecha_Hora*.

- El formato del fichero será *XLSX*.

6.4.2. Configuración del proceso

En el caso de los procesos de tipo output comenzamos por el apartado de generación de la plantilla *DAT*, para que pueda generar el informe mediante la herramienta *Generación de ficheros* 5.2.5 detallada en el capítulo 5. El *Código 6.35* muestra la colección de columnas que contendrá el fichero además de sus delimitadores.

Código 6.35: Output TopSellers.fmt

```
12.0
13
1   SQLCHAR   0   0   ";"   1 top                     SQL_Latin1_General_CP1_CI_AS
2   SQLCHAR   0   0   ";"   2 concesionario           SQL_Latin1_General_CP1_CI_AS
3   SQLCHAR   0   0   ";"   3 vendedor              SQL_Latin1_General_CP1_CI_AS
4   SQLCHAR   0   0   ";"   4 ventas_totales          SQL_Latin1_General_CP1_CI_AS
5   SQLCHAR   0   0   ";"   5 importe_ventas_totales    SQL_Latin1_General_CP1_CI_AS
6   SQLCHAR   0   0   ";"   6 fecha_ultima_venta        SQL_Latin1_General_CP1_CI_AS
7   SQLCHAR   0   0   ";"   7 ventas_ultimo_mes       SQL_Latin1_General_CP1_CI_AS
8   SQLCHAR   0   0   ";"   8 promedio_ventas_mensuales  SQL_Latin1_General_CP1_CI_AS
9   SQLCHAR   0   0   ";"   9 promedio_importe_ventas   SQL_Latin1_General_CP1_CI_AS
10  SQLCHAR   0   0   ";"   10  ofertas_totales           SQL_Latin1_General_CP1_CI_AS
11  SQLCHAR   0   0   ";"   11  fecha_ultima_oferta       SQL_Latin1_General_CP1_CI_AS
12  SQLCHAR   0   0   ";"   12  propuestas_totales        SQL_Latin1_General_CP1_CI_AS
13  SQLCHAR   0   0   "\n"  13  fecha_ultima_propuesta    SQL_Latin1_General_CP1_CI_AS
```

Pasamos al apartado de BBDD IO donde el primer paso consiste en insertar el registro con la configuración maestra en la tabla *IO.config.master_process_type*, el cual contiene el detalle mostrado en la *Figura 6.13*.

id	customer	name	description	active_flag	ingestion_flag	transformation_flag	persist_flag	output_flag	stop_on_warning
2	BMW_Group	TopSellers_Output	Genera un informe con el TOP 10 de vendedores d...	1	0	0	0	1	1

Figura 6.13: Registro del proceso en la tabla master_process_type

Tras la inserción del registro de configuración maestra, pasamos a dar de alta los registros con las fases e información detallada en la tabla *IO.config.detail_process_type* la cual tendrá una sola fase de tipo output con la información mostrada en la *Figura 6.14*.

- Step de generación del fichero de salida a través del procedimiento almacenado *IO.output.sp_BMWGroup_TopSellers_report*.
- Step de compresión del fichero generado en formato *ZIP* con contraseña el cual se deposita en nuestro directorio local.
- Subida del fichero en formato *ZIP* al directorio de salida específico de este proceso en nuestro servidor FTP.
- Envío de correo al cliente con el fichero adjunto generado.
- Step final para eliminar el fichero generado en nuestra ruta local.

id	master_process_type_id	phase	step	entity	name	description	source_type
200	2	OUTPUT	1	TopSellers_Generation	TopSellers Exec output procedure	Generación del archivo de salida a través de procedimiento almacenado	BBDD
210	2	OUTPUT	2	TopSellers_ZIP	TopSellers compress report	Compresión del archivo con la información .ZIP con contraseña	ZIP
220	2	OUTPUT	3	TopSellers_FTP_Upload	TopSellers upload report into FTP	Subida del report TopSellers al FTP	FTP
230	2	OUTPUT	4	TopSellers_Email	TopSellers business email	Envío de email informando a Negocio, incluyendo el report como adjunto	EMAIL
240	2	OUTPUT	5	TopSellers_Delete	TopSellers delete source file	Eliminar el report de la ruta original para evitar subirlo tras cada ejecución	FILE

Figura 6.14: Registros del proceso en la tabla detail_process_type

Es importante destacar que el proceso generará la tabla intermedia *IO.output.BMWGroup_TopSellers* tras cada ejecución que contendrá la información antes de persistirla en el fichero final. Para ver en detalle la inserción de los registros de configuración, ver el script *2 Output - TopSellers xlsx.sql* C.36 del *Código fuente* C.

Adicionalmente, el primer step de la fase output contiene un procedimiento almacenado con la lógica de generación del informe, cuya ejecución se produce desde la columna *source_table_query* de la tabla *IO.config.detail_process_type*. Con respecto al código del procedimiento almacenado, ver el apartado *Procedimiento BMWGroup_TopSellers* C del *Código fuente* C.

6.4.3. Ejecución y resultados del proceso

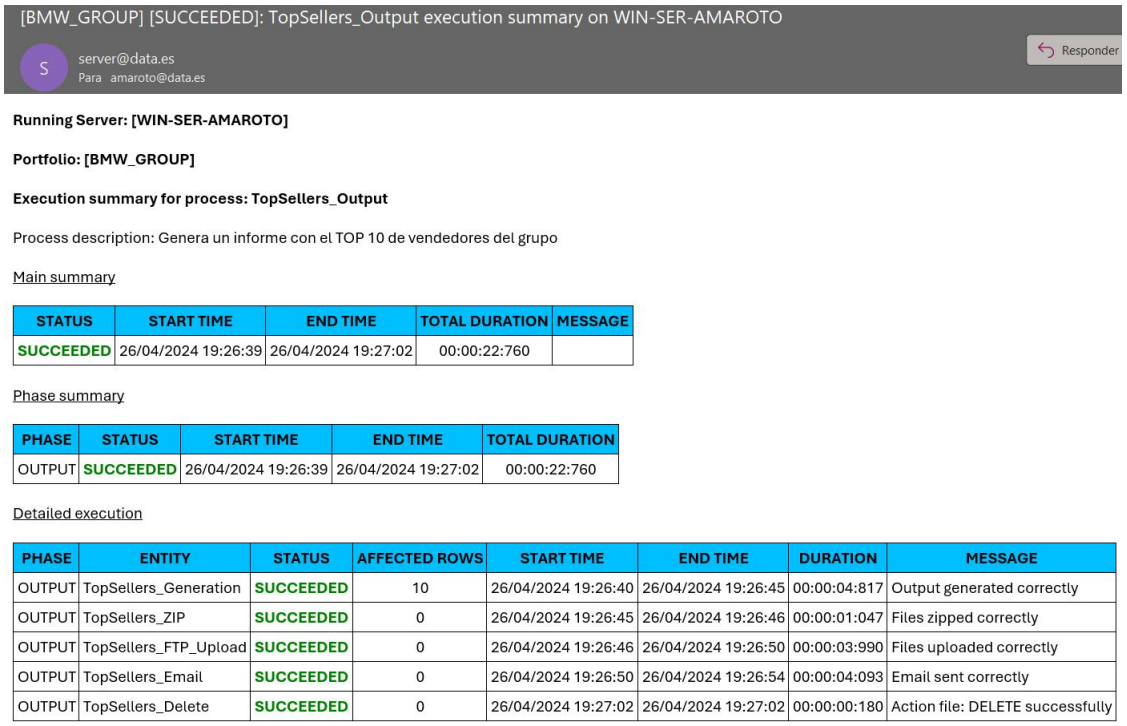
Tras la configuración del proceso, pasaremos por último a su ejecución y contrastaremos los resultados generados acordes con los requisitos solicitados.

El primer paso consiste en ejecutar el proceso y contrastar la creación de la tabla intermedia con la información que contendrá el informe como muestra la *Figura 6.15*.

top	concesionario	vendedor	ventas_totales	importe_ventas_totales	fecha_ultima_venta	ventas_ultimo_mes	promedio_ventas_mensuales	promedio_importe_ventas
1	BYmyCAR Madrid	Juan López	6	171000.00	2023-05-11	0	6	28500.00
2	Movitransa	Laura Ruiz	5	141000.00	2023-06-27	0	5	28200.00
3	BYmyCAR Madrid	Ana Martín	3	80000.00	2023-05-03	0	3	26666.67
4	Avilcar	Carlos Gómez	3	83000.00	2023-06-15	0	1	27666.67
5	Vehinter	David Díaz	3	85000.00	2023-06-22	0	3	28333.33
6	BYmyCAR Madrid	María García	2	57000.00	2023-05-05	0	2	28500.00
7	Avilcar	Elena Rodríguez	2	60000.00	2023-06-17	0	2	30000.00
8	Vehinter	Marta Martínez	2	58000.00	2023-06-19	0	2	29000.00
9	BYmyCAR Madrid	Lucía Pérez	1	26000.00	2023-05-12	0	1	26000.00
10	Movitransa	Sara Jiménez	1	33000.00	2023-06-28	0	1	33000.00

Figura 6.15: Registros cargados en IO.output.BMWGroup_TopSellers

Posteriormente recibiremos dos correos, por un lado el correo con los resultados del proceso ejecutado (*Figura 6.16*), y por otro el correo que recibirá el cliente con el informe en formato *ZIP* (*Figura 6.17*).



[BMW_GROUP] [SUCCEEDED]: TopSellers_Output execution summary on WIN-SER-AMAROTO

server@data.es
Para amaroto@data.es

Responder

Running Server: [WIN-SER-AMAROTO]

Portfolio: [BMW_GROUP]

Execution summary for process: TopSellers_Output

Process description: Genera un informe con el TOP 10 de vendedores del grupo

Main summary

STATUS	START TIME	END TIME	TOTAL DURATION	MESSAGE
SUCCEEDED	26/04/2024 19:26:39	26/04/2024 19:27:02	00:00:22:760	

Phase summary

PHASE	STATUS	START TIME	END TIME	TOTAL DURATION
OUTPUT	SUCCEEDED	26/04/2024 19:26:39	26/04/2024 19:27:02	00:00:22:760

Detailed execution

PHASE	ENTITY	STATUS	AFFECTED ROWS	START TIME	END TIME	DURATION	MESSAGE
OUTPUT	TopSellers_Generation	SUCCEEDED	10	26/04/2024 19:26:40	26/04/2024 19:26:45	00:00:04:817	Output generated correctly
OUTPUT	TopSellers_ZIP	SUCCEEDED	0	26/04/2024 19:26:45	26/04/2024 19:26:46	00:00:01:047	Files zipped correctly
OUTPUT	TopSellers_FTP_Upload	SUCCEEDED	0	26/04/2024 19:26:46	26/04/2024 19:26:50	00:00:03:990	Files uploaded correctly
OUTPUT	TopSellers_Email	SUCCEEDED	0	26/04/2024 19:26:50	26/04/2024 19:26:54	00:00:04:093	Email sent correctly
OUTPUT	TopSellers_Delete	SUCCEEDED	0	26/04/2024 19:27:02	26/04/2024 19:27:02	00:00:00:180	Action file: DELETE successfully

Figura 6.16: Correo recibido del proceso BMWGroup_TopSellers

BMW Group - TopSeller report

server@data.es
Para amaroto@data.es

BMWGroup_TopSellers_20240426.zip
8 KB

Buenos dias, adjuntamos el fichero mensual de TopSellers.

Un cordial saludo.

Departamento de Data

1 files have been attached into mail:

File name	Size
BMWGroup_TopSellers_20240426.zip	7823 bytes

1 files have been attached into mail with 0,007823 MB of total size

Figura 6.17: Correo enviado al cliente con el informe adjunto

Adicionalmente, contrastamos que el fichero se encuentra en el directorio indicado del FTP (*Figura 6.18*).

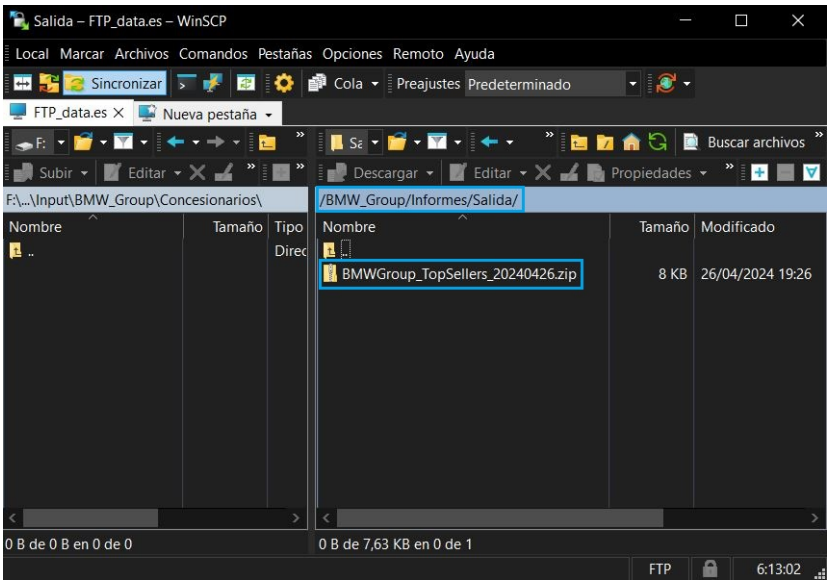


Figura 6.18: Fichero disponible en el directorio de salida del FTP

Al intentar descomprimir el fichero nos solicitará insertar la contraseña como muestra la *Figura 6.19*.

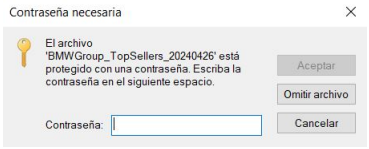


Figura 6.19: Contraseña necesaria para descomprimir el fichero

Contrastamos que los resultados generados en el informe en formato *XLSX* son correctos como muestra la *Figura 6.20*.

A	B	C	D	E	F	G	H
Top	Concesionario	Vendedor	Ventas_totales	Importe_ventas_totales	Fecha_ultima_venta	Ventas_ultimo_mes	Promedio_ventas_mensuales
1	BYmyCAR Madrid	Juan Lopez	6	171000.00	2023-05-11	0	6
2	Movitransa	Laura Ruiz	5	141000.00	2023-06-27	0	5
3	BYmyCAR Madrid	Ana Martin	3	80000.00	2023-05-03	0	3
4	Avilcar	Carlos Gomez	3	83000.00	2023-06-15	0	1
5	Vehinter	David Diaz	3	85000.00	2023-06-22	0	3
6	BYmyCAR Madrid	Maria Garcia	2	57000.00	2023-05-05	0	2
7	Avilcar	Elena Rodriguez	2	60000.00	2023-06-17	0	2
8	Vehinter	Marta Martinez	2	58000.00	2023-06-19	0	2
9	BYmyCAR Madrid	Lucia Perez	1	26000.00	2023-05-12	0	1
10	Movitransa	Sara Jimenez	1	33000.00	2023-06-28	0	1

Figura 6.20: Resultados generados del informe BMWGroup_TopSellers

Por último, los ficheros procesados se archivan en nuestra ruta local como muestra la *Figura 6.21*.

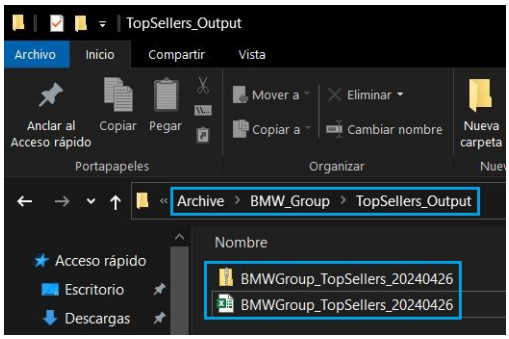


Figura 6.21: Ficheros disponibles en el directorio archive del proceso

6.5. Caso de uso 3: Output - MiniSales txt

El tercer caso de uso del proyecto *Output - MiniSales* presenta un proceso output el cual genera un informe simple para el cliente en un formato de salida *TXT*, donde la información de origen se extrae de la BBDD *VehicleSales* a través de nuestra solución IO.

Mediante este caso de uso se verifican los requisitos de *Gestión de Datos de Entrada y Salida (1-6)*, *Orquestación de Procesos (1 y 3)*, *Funcionalidades Específicas (1, 2, 3 y 5)* y *Monitorización y Trazabilidad (1)*.

6.5.1. Definición del proceso

El cliente realiza una petición por la cual pretende conocer las ventas producidas de vehículos de la marca *Mini*, para ello nos define los siguientes detalles funcionales.

- Depositaremos un fichero en nuestro servidor FTP en el directorio de salida específico del proceso para que el cliente pueda recogerlo.
- El fichero deberá comprimirse de forma normal sin contraseña.
- El fichero deberá contener todas las ventas producidas hasta el momento para la marca, dando elección al equipo a extraer la información que considere de utilidad.
- Desde el equipo se decide extraer la información base de la marca, el modelo, tipo de propuesta, concesionario, vendedor, así como los importes y fechas de las ventas, ofertas y propuestas.
- La nomenclatura del fichero será *NombreCliente\MiniSales_Fecha_Hora*.
- El formato del fichero será *CSV*.

6.5.2. Configuración del proceso

En el caso de los procesos de tipo output comenzamos por el apartado de generación de la plantilla *DAT*, para que pueda generar el informe mediante la herramienta *Generación de ficheros* 5.2.5 detallada en el capítulo 5. El *Código 6.36* muestra la colección de columnas que contendrá el fichero además de sus delimitadores.

Código 6.36: Output MiniSales.fmt

```
12.0
12
1   SQLCHAR   0   0   ";"   1   marca           SQL_Latin1_General_CP1_CI_AS
2   SQLCHAR   0   0   ";"   2   modelo            SQL_Latin1_General_CP1_CI_AS
3   SQLCHAR   0   0   ";"   3   tipo_propuesta    SQL_Latin1_General_CP1_CI_AS
4   SQLCHAR   0   0   ";"   4   tipo_modelo       SQL_Latin1_General_CP1_CI_AS
5   SQLCHAR   0   0   ";"   5   importe_venta     SQL_Latin1_General_CP1_CI_AS
6   SQLCHAR   0   0   ";"   6   fecha_venta       SQL_Latin1_General_CP1_CI_AS
7   SQLCHAR   0   0   ";"   7   fecha_oferta      SQL_Latin1_General_CP1_CI_AS
8   SQLCHAR   0   0   ";"   8   cliente           SQL_Latin1_General_CP1_CI_AS
9   SQLCHAR   0   0   ";"   9   importe_propuesta  SQL_Latin1_General_CP1_CI_AS
10  SQLCHAR   0   0   ";"   10  fecha_propuesta      SQL_Latin1_General_CP1_CI_AS
11  SQLCHAR   0   0   ";"   11  concesionario     SQL_Latin1_General_CP1_CI_AS
12  SQLCHAR   0   0   "\n"  12  vendedor        SQL_Latin1_General_CP1_CI_AS
```

Pasamos al apartado de BBDD IO donde el primer paso consiste en insertar el registro con la configuración maestra en la tabla *IO.config.master_process_type*, el cual contiene el detalle de la *Figura 6.22*.

id	customer	name	description	active_flag	ingestion_flag	transformation_flag	persist_flag	output_flag	stop_on_warning
3	Mini	MiniSales_Output	Genera un informe con los ventas de vehículos de ...	1	0	0	0	1	1

Figura 6.22: Registro del proceso en la tabla master_process_type

Tras la inserción del registro de configuración maestra, pasamos a dar de alta los registros con las fases e información detallada en la tabla *IO.config.detail_process_type* la cual tendrá una sola fase de tipo output con la información de la *Figura 6.23*.

- Step de generación del fichero de salida a través de consulta embebida.
- Step para modificar la extensión del archivo de *TXT* a *CSV*
- Step de compresión del fichero generado en formato *ZIP* sin contraseña el cual se deposita en nuestro directorio local.
- Subida del fichero en formato *ZIP* al directorio de salida específico de este proceso en nuestro servidor FTP.
- Step final para eliminar el fichero generado en nuestra ruta local.

id	master_process_type_id	phase	step	entity	name	description	source_type
300	3	OUTPUT	1	MiniSales_Generation	MiniSales Exec output procedure	Generación del archivo de salida a través de con...	BBDD
310	3	OUTPUT	2	MiniSales_Rename	MiniSales check report name	Modifica la extensión del archivo de TXT a CSV	FILE
320	3	OUTPUT	3	MiniSales_ZIP	MiniSales compress report	Compresión del archivo con la información .ZIP	ZIP
330	3	OUTPUT	4	MiniSales_FTP_Upload	MiniSales upload report into FTP	Subida del report MiniSales al FTP	FTP
340	3	OUTPUT	5	MiniSales_Delete	MiniSales delete source file	Eliminar el report de la ruta original para evitar su...	FILE

Figura 6.23: Registros del proceso en la tabla detail_process_type

Es importante destacar la diferencia con respecto al anterior caso de uso, en este caso no se genera una tabla intermedia ni un procedimiento almacenado con la lógica de generación del informe, sino que se genera directamente por consulta en la columna *source_table_query* de la tabla *IO.config.detail_process_type*. Para revisar en detalle la consulta embebida, ver el script *3 Output - MiniSales txt.sql* C.37 del *Código fuente* C.

6.5.3. Ejecución y resultados del proceso

Tras la configuración del proceso, pasaremos por último a su ejecución y contrastaremos los resultados generados acordes con los requisitos solicitados.

El primer paso consiste en ejecutar el proceso y generar el fichero de salida en formato *TXT* como muestra la *Figura 6.24*.

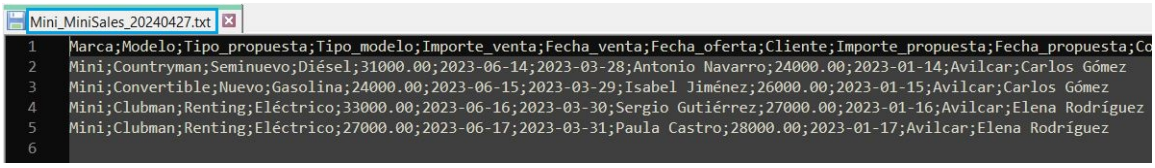
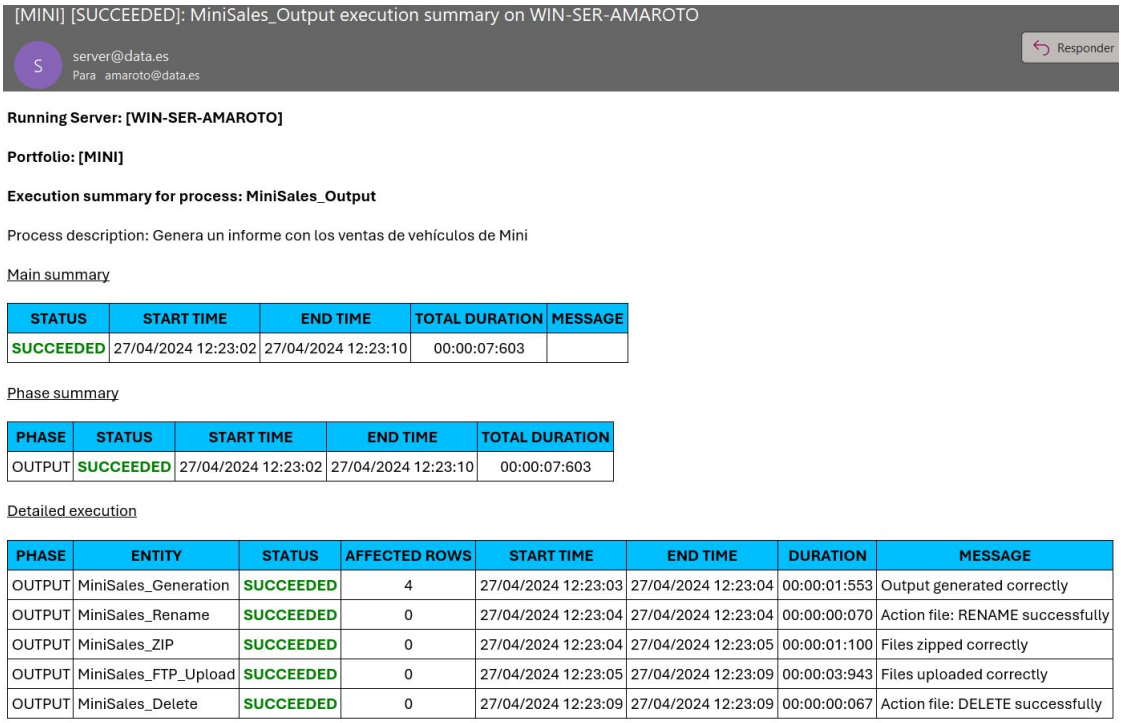


Figura 6.24: Resultados generados en formato TXT

Posteriormente recibimos el correo como muestra la *Figura 6.25* con los resultados del proceso ejecutado de forma satisfactoria.


[MINI] [SUCCEEDED]: MiniSales_Output execution summary on WIN-SER-AMAROTO

server@data.es
Para amaroto@data.es

Responder

Running Server: [WIN-SER-AMAROTO]

Portfolio: [MINI]

Execution summary for process: MiniSales_Output

Process description: Genera un informe con los ventas de vehículos de Mini

Main summary

STATUS	START TIME	END TIME	TOTAL DURATION	MESSAGE
SUCCEEDED	27/04/2024 12:23:02	27/04/2024 12:23:10	00:00:07:603	

Phase summary

PHASE	STATUS	START TIME	END TIME	TOTAL DURATION
OUTPUT	SUCCEEDED	27/04/2024 12:23:02	27/04/2024 12:23:10	00:00:07:603

Detailed execution

PHASE	ENTITY	STATUS	AFFECTED ROWS	START TIME	END TIME	DURATION	MESSAGE
OUTPUT	MiniSales_Generation	SUCCEEDED	4	27/04/2024 12:23:03	27/04/2024 12:23:04	00:00:01:553	Output generated correctly
OUTPUT	MiniSales_Rename	SUCCEEDED	0	27/04/2024 12:23:04	27/04/2024 12:23:04	00:00:00:070	Action file: RENAME successfully
OUTPUT	MiniSales_ZIP	SUCCEEDED	0	27/04/2024 12:23:04	27/04/2024 12:23:05	00:00:01:100	Files zipped correctly
OUTPUT	MiniSales_FTP_Upload	SUCCEEDED	0	27/04/2024 12:23:05	27/04/2024 12:23:09	00:00:03:943	Files uploaded correctly
OUTPUT	MiniSales_Delete	SUCCEEDED	0	27/04/2024 12:23:09	27/04/2024 12:23:09	00:00:00:067	Action file: DELETE successfully

Figura 6.25: Correo recibido del proceso Mini_MiniSales

Adicionalmente, contrastamos que el fichero se encuentra en el directorio indicado del FTP como muestra la *Figura 6.26*.

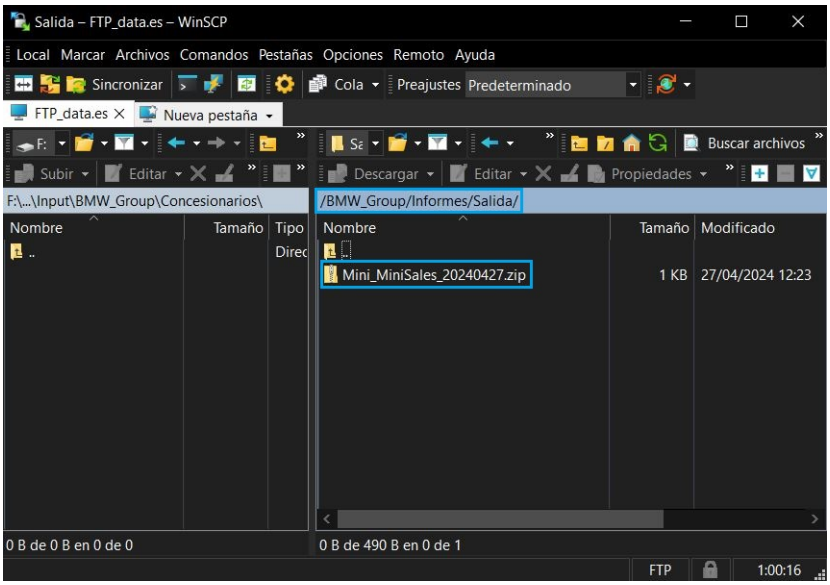


Figura 6.26: Fichero disponible en el directorio de salida del FTP

Al descomprimir el fichero en este caso no se solicita contraseña. La *Figura 6.27* contrasta que los resultados generados en el informe en formato *CSV* son correctos.

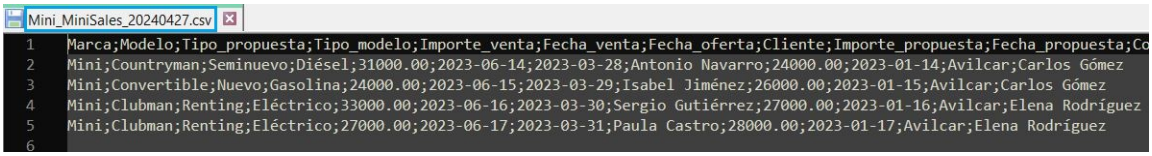


Figura 6.27: Resultados generados del informe Mini_MiniSales

Por último, la *Figura 6.28* muestra los ficheros procesados archivados en nuestra ruta local.

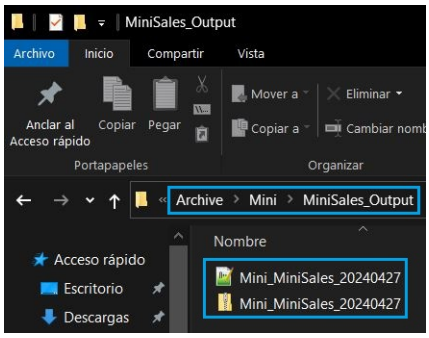


Figura 6.28: Ficheros disponibles en el directorio archive del proceso

6.6. Caso de uso 4: Input - RemoteConnections BBDD

El cuarto caso de uso del proyecto *Input - RemoteConnections BBDD* presenta un proceso input simple con un traspaso de información entre las BBDDs *VehicleSales* y *VehicleAlt*, el cual será procesado por nuestra solución IO.

Mediante este caso de uso se verifican los requisitos de *Gestión de Datos de Entrada y Salida (1-6)*, *Orquestación de Procesos (1 y 3)*, *Funcionalidades Específicas (1 y 5)* y *Monitorización y Trazabilidad (1)*.

6.6.1. Definición del proceso

El equipo necesita realizar una copia de la tabla *VehicleSales.dbo.sale* a la BBDD alternativa *VehicleAlt*, y poder mantener de esta forma un *backup* con la información original, por lo que en este caso al tratarse de un proceso de mantenimiento técnico no hay detalles funcionales.

6.6.2. Configuración del proceso

Una vez se ha definido el proceso de mantenimiento comenzamos con su configuración. En este caso de uso, al no haber un fichero de ingesta no se genera plantilla *DAT*, por lo que pasamos directamente al apartado de la BBDD IO donde el primer paso consiste en insertar el registro con la configuración maestra en la tabla *IO.config.master_process_type* con el detalle mostrado en la *Figura 6.29*.

id	customer	name	description	active_flag	ingestion_flag	transformation_flag	persist_flag	output_flag	stop_on_warning
4	BMW_Group	RemoteConnections_Input	Test para comprobar la conexión remota de la IO ...	1	1	1	1	0	1

Figura 6.29: Registro del proceso en la tabla master_process_type

Tras la inserción del registro de configuración maestra, pasamos a dar de alta los registros con las fases e información detallada en la tabla *IO.config.detail_process_type* la cual tendrá la información de la *Figura 6.30*.

- Fase de ingesta con un step que genera la tabla intermedia *IO.auxiliar.sale* en la BBDD IO.
- Fase de delivery para insertar un registro de seguimiento en la tabla *IO.load.delivery*.
- Fase de persistencia para incorporar los datos de la tabla auxiliar a la tabla de destino final *VehicleAlt.dbo.sale*.

id	master_process_type_id	phase	step	entity	name	description	source_type
400	4	INGESTION	1	load_remote_VehicleSales.dbo.sale	RemoteConnections load sales	Generación de tabla auxiliar con la información de v...	BBDD
410	4	DELIVERY	1	new_delivery	Alta delivery RemoteConnections	Alta de una nueva delivery de RemoteConnections_...	BBDD
420	4	PERSISTENCE	1	insert_remote_VehicleAlt.dbo.sale	Insert VehicleAlt sale	Incorporación de los datos en BBDD destino Vehicl...	BBDD

Figura 6.30: Registros del proceso en la tabla detail_process_type

Es importante destacar que la tabla intermedia *IO.auxiliar.sale* se generará de forma automática a través de la ejecución del procedimiento almacenado de la fase de ingesta, borrándola en caso de existir previamente. Para ver en detalle la inserción de los registros de configuración, ver el script *4 Input - RemoteConnections BBDD.sql* C.38 del *Código fuente* C.

Adicionalmente, cada fase contiene un procedimiento almacenado con la lógica a seguir, cuya ejecución se produce desde la columna *source_table_query* de la tabla *IO.config.detail_process_type*. Con respecto al detalle de este código, ver el apartado *Procedimientos RemoteConnections* C del *Código fuente* C.

6.6.3. Ejecución y resultados del proceso

Tras la configuración del proceso, pasaremos por último a su ejecución y contrastaremos los resultados persistidos acordes con la información de la tabla original.

La tabla de destino *VehicleAlt.dbo.sale* se encuentra originalmente vacía como muestra la *Figura 6.31*.

id	offer_id	sale_date	creation_date	modification_date	io_id

Figura 6.31: Situación original en VehicleAlt.dbo.sale

Tras la revisión ejecutamos el proceso y contrastamos la creación de la tabla intermedia con la información que contendrá la tabla destino como muestra la *Figura 6.32*.

id	delivery_id	offer_id	sale_date	io_id
1	1	1	2023-05-01	NULL
2	1	2	2023-05-02	NULL
3	1	3	2023-05-03	NULL
4	1	4	2023-05-04	NULL
5	1	5	2023-05-05	NULL
6	1	6	2023-05-06	NULL
7	1	7	2023-05-07	NULL
8	1	8	2023-05-08	NULL
9	1	9	2023-05-09	NULL
10	1	10	2023-05-10	NULL

Figura 6.32: Registros cargados en IO.auxiliar.sale

Posteriormente recibimos el correo como muestra la *Figura 6.33* con los resultados del proceso ejecutado de forma satisfactoria.

[BMW_GROUP] [SUCCEEDED]: RemoteConnections_Input execution summary on WIN-SER-AMAROTO

server@data.es
Para amaroto@data.es

Responder | Responder a todos | Re

sá. 2

Running Server: [WIN-SER-AMAROTO]

Portfolio: [BMW_GROUP]

Execution summary for process: RemoteConnections_Input

Process description: Test para comprobar la conexión remota de la IO tanto para el origen como para el destino

Main summary

STATUS	START TIME	END TIME	TOTAL DURATION	MESSAGE
SUCCEEDED	27/04/2024 13:37:40	27/04/2024 13:37:43	00:00:02:683	

Phase summary

PHASE	STATUS	START TIME	END TIME	TOTAL DURATION
INGESTION	SUCCEEDED	27/04/2024 13:37:40	27/04/2024 13:37:41	00:00:01:507
TRANSFORMATION	SUCCEEDED	27/04/2024 13:37:41	27/04/2024 13:37:42	00:00:00:617
PERSISTENCE	SUCCEEDED	27/04/2024 13:37:42	27/04/2024 13:37:43	00:00:00:560

Detailed execution

PHASE	ENTITY	STATUS	AFFECTED ROWS	START TIME	END TIME	DURATION	MESSAGE
INGESTION	[IO].auxiliar.sale	SUCCEEDED	30	27/04/2024 13:37:41	27/04/2024 13:37:41	00:00:00:270	
DELIVERY	NEW DELIVERIES	SUCCEEDED	1	27/04/2024 13:37:42	27/04/2024 13:37:42	00:00:00:183	
PERSISTENCE	insert_remote_VehicleAlt.dbo.sale	SUCCEEDED	30	27/04/2024 13:37:43	27/04/2024 13:37:43	00:00:00:127	Persistence for delivery [ID]: 1 has been done

Figura 6.33: Correo recibido del proceso RemoteConnections BBDD

id	offer_id	sale_date	creation_date	modification_date	io_id
1	1	2023-05-01	2024-04-27 13:37:43.093	NULL	1
2	2	2023-05-02	2024-04-27 13:37:43.097	NULL	2
3	3	2023-05-03	2024-04-27 13:37:43.097	NULL	3
4	4	2023-05-04	2024-04-27 13:37:43.097	NULL	4
5	5	2023-05-05	2024-04-27 13:37:43.097	NULL	5
6	6	2023-05-06	2024-04-27 13:37:43.097	NULL	6
7	7	2023-05-07	2024-04-27 13:37:43.097	NULL	7
8	8	2023-05-08	2024-04-27 13:37:43.100	NULL	8
9	9	2023-05-09	2024-04-27 13:37:43.100	NULL	9
10	10	2023-05-10	2024-04-27 13:37:43.100	NULL	10

Figura 6.34: Situación final en VehicleSales.dbo.sale

Adicionalmente, si consultamos la tabla *VehicleAlt.dbo.sale* podemos apreciar como esta contiene la nueva información procedente de la BBDD *VehicleSales* (*Figura 6.34*).

En este caso de uso al no haber ficheros en el proceso la información no se archiva en nuestro directorio local.

6.7. Caso de uso 5: Input - StatusUpdate xlsx

El quinto caso de uso del proyecto *Input - StatusUpdate xlsx* presenta un proceso input complejo con un fichero en formato *XLSX*, el cual será procesado por nuestra solución IO para persistir la información en la BBDD *VehicleSales*.

Mediante este caso de uso se verifican los requisitos de *Gestión de Datos de Entrada y Salida (1-6)*, *Orquestación de Procesos (1 y 3)*, *Funcionalidades Específicas (1, 2, 3 y 5)* y *Monitorización y Trazabilidad (1)*.

6.7.1. Definición del proceso

El cliente realiza una petición por la cual pretende mantener actualizada la información relacionada con las propuestas, ofertas y ventas de sus concesionarios, para ello nos define los siguientes detalles funcionales.

- Se depositará un fichero en nuestro servidor FTP en el directorio de entrada del proceso.
- El fichero estará comprimido en formato *ZIP* con contraseña que nos aportará el cliente.
- El fichero contendrá el status actualizado de propuestas, ofertas y ventas realizadas a clientes.
- Puede haber información nueva o existente de clientes, vendedores y modelos, por lo que en este caso tendremos que crear los nuevos registros en la BBDD.
- La nomenclatura del fichero será *NombreGrupo_StatusUpdate_Fecha_Hora*.
- El formato del fichero será *XLSX*.

- La información de columnas serán el modelo, datos del cliente, datos del vendedor y la información de propuestas, ofertas y ventas para cada caso.

6.7.2. Configuración del proceso

Una vez se recibe la definición del proceso comenzamos con su configuración. Para ello, el primer punto a revisar es el formato del fichero el cual podemos ver en la *Figura 6.35*.

Modelo	Cliente	Tel Cliente	Email Cliente	Fecha Propuesta	Cantidad Propuesta	Tipo Propuesta	Fecha Oferta	Cantidad Oferta	Fecha Venta	Concesionario	Vendedor	Tel Vendedor	Email Vendedor
Serie 1	Mario Torres	555777999	mario@example.com	2024-03-01	43000	Nuevo	2024-03-10	38000	2024-03-22	BYmyCAR Madrid	Ana Martín	111222333	ana@example.com
Serie 2	Mario Torres	555777999	mario@example.com	2024-03-01	48000	Seminuevo	2024-03-10	45000		BYmyCAR Madrid	Ana Martín	111222333	ana@example.com
S3	Mario Torres	555777999	mario@example.com	2024-03-01	55000	Nuevo				BYmyCAR Madrid	Ana Martín	111222333	ana@example.com
I4	Elena Navarro	666333222	elena@example.com	2024-03-02	70000	Nuevo	2024-03-08	65000	2024-03-18	Vehinter	David Díaz	444666888	david@example.com
X4 2024	Elena Navarro	666333222	elena@example.com	2024-03-02	90000	Leasing	2024-03-08	80000		Vehinter	David Díaz	444666888	david@example.com
M2	Silvia Redondo	654321234	silvia@example.com	2024-03-03	80000	Nuevo				Celtamotor	Roberto Suarez	712345678	roberto@example.com
M3	Silvia Redondo	654321234	silvia@example.com	2024-03-03	120000	Renting	2024-03-12	110000	2024-04-21	Celtamotor	Roberto Suarez	712345678	roberto@example.com

Figura 6.35: Formato de fichero BMW_StatusUpdate xlsx

Posteriormente generamos la plantilla *DAT* para que pueda ser procesado mediante la herramienta *Ingesta de archivos en BBDD* 5.2.2 detallada en el capítulo 5. Como muestra el *Código 6.37*, esta contiene las mismas columnas que el fichero además de los separadores de columna habituales para ficheros en formato *XLSX*.

Código 6.37: Input StatusUpdate.fmt

```
12.0
14
1    SQLCHAR   0   0    ";"   1 modelo               SQL_Latin1_General_CP1_CI_AS
2    SQLCHAR   0   0    ";"   2 cliente              SQL_Latin1_General_CP1_CI_AS
3    SQLCHAR   0   0    ";"   3 tel_cliente          SQL_Latin1_General_CP1_CI_AS
4    SQLCHAR   0   0    ";"   4 email_cliente        SQL_Latin1_General_CP1_CI_AS
5    SQLCHAR   0   0    ";"   5 fecha_propuesta         SQL_Latin1_General_CP1_CI_AS
6    SQLCHAR   0   0    ";"   6 cantidad_propuesta      SQL_Latin1_General_CP1_CI_AS
7    SQLCHAR   0   0    ";"   7 tipo_propuesta       SQL_Latin1_General_CP1_CI_AS
8    SQLCHAR   0   0    ";"   8 fecha_oferta         SQL_Latin1_General_CP1_CI_AS
9    SQLCHAR   0   0    ";"   9 cantidad_oferta         SQL_Latin1_General_CP1_CI_AS
10   SQLCHAR   0   0    ";"   10  fecha_venta          SQL_Latin1_General_CP1_CI_AS
11   SQLCHAR   0   0    ";"   11  concesionario        SQL_Latin1_General_CP1_CI_AS
12   SQLCHAR   0   0    ";"   12  vendedor            SQL_Latin1_General_CP1_CI_AS
13   SQLCHAR   0   0    ";"  13 tel_vendedor         SQL_Latin1_General_CP1_CI_AS
14   SQLCHAR   0   0    "\n"  14  email_vendedor       SQL_Latin1_General_CP1_CI_AS
```

A partir de este punto pasamos al apartado de la BBDD IO donde el primer paso consiste en insertar el registro con la configuración maestra en la tabla *IO.config.master_process_type*, el cual contiene el detalle mostrado en la *Figura 6.36*.

id	customer	name	description	active_flag	ingestion_flag	transformation_flag	persist_flag	output_flag	stop_on_warning
5	BMW_Group	StatusUpdate_Input	Incorpora la información relacionada con el proc...	1	1	1	1	0	1

Figura 6.36: Registro del proceso en la tabla master_process_type

Tras la inserción del registro de configuración maestra, pasamos a dar de alta los registros con las fases e información detallada en la tabla *IO.config.detail_process_type* la cual tendrá la información de la *Figura 6.37*.

- Fase de ingesta con tres steps, el primero para descargar el fichero del FTP a nuestro directorio local, el segundo para descomprimir el fichero generado en formato *ZIP* con contraseña, y el tercero para cargar la información en la tabla *IO.input.BMWGroup_StatusUpdate*.

- Fase de delivery para insertar un registro de seguimiento en la tabla *IO.load.delivery*.

- Fase de transformación con seis steps, los cuales insertan la información en las tablas intermedias del schema *load* de forma ordenada.
- Fase de persistencia con seis steps para insertar la nueva información en las tablas de destino en la BBDD *VehicleSales*.

id	master_process_type_id	phase	step	entity	name	description	source_type
500	5	INGESTION	1	download_StatusUpdate	Descarga archivo StatusUpdate	Recoge del FTP y mueve al Input el fichero Status...	FTP
505	5	INGESTION	2	unzip_StatusUpdate	Descomprime zip StatusUpdate	Decompresión del archivo .ZIP con contraseña que...	UNZIP
510	5	INGESTION	3	load_file_input_table	BMWGroup_StatusUpdate	Archivos de propuestas/ofertas/ventas	FILE
515	5	DELIVERY	1	new_delivery	Alta delivery StatusUpdate	Alta de una nueva delivery de StatusUpdate por ca...	BBDD
520	5	TRANSFORMATION	1	insert_model_type	Insert load model_type	Incorporación de los datos de input a load.model_t...	BBDD
525	5	TRANSFORMATION	2	insert_seller	Insert load seller	Incorporación de los datos de input a load.seller	BBDD
530	5	TRANSFORMATION	3	insert_customer	Insert load customer	Incorporación de los datos de input a load.customer	BBDD
535	5	TRANSFORMATION	4	insert_proposal	Insert load proposal	Incorporación de los datos de input a load.proposal	BBDD
540	5	TRANSFORMATION	5	insert_offer	Insert load offer	Incorporación de los datos de input a load.offer	BBDD
545	5	TRANSFORMATION	6	insert_sale	Insert load sale	Incorporación de los datos de input a load.sale	BBDD
550	5	PERSISTENCE	1	insert_model_type	Insert VehicleSales model_type	Incorporación de los datos de load a VehicleSales....	BBDD
555	5	PERSISTENCE	2	insert_seller	Insert VehicleSales seller	Incorporación de los datos de load a VehicleSales....	BBDD
560	5	PERSISTENCE	3	insert_customer	Insert VehicleSales customer	Incorporación de los datos de load a VehicleSales....	BBDD
565	5	PERSISTENCE	4	insert_proposal	Insert VehicleSales proposal	Incorporación de los datos de load a VehicleSales....	BBDD
570	5	PERSISTENCE	5	insert_offer	Insert VehicleSales offer	Incorporación de los datos de load a VehicleSales....	BBDD
575	5	PERSISTENCE	6	insert_sale	Insert VehicleSales sale	Incorporación de los datos de load a VehicleSales....	BBDD

Figura 6.37: Registros del proceso en la tabla detail_process_type

Es importante destacar que deberemos generar las siguientes tablas intermedias para que estén preparadas previa a la ejecución del proceso.

- *IO.load.model_type*.
- *IO.load.seller*.
- *IO.load.customer*.
- *IO.load.proposal*.
- *IO.load.offer*.
- *IO.load.sale*.

Es necesario crear además la tabla auxiliar *BMWGroup_model_type_mapping* para mapear los modelos recibidos con los de nuestra BBDD *VehichleSales*, y mantener de esta forma la consistencia y normalización de los registros existentes en nuestra tabla destino VehicleSales.dbo.model_type. Para ver en detalle esta información, ver el script *5 Input - StatusUpdate xlsx.sql* C.39 del *Código fuente* C.

Adicionalmente, cada step de la fase delivery, transformación y persistencia contiene diferentes procedimientos almacenados con la lógica a seguir, cuya ejecución se produce desde la columna *source_table_query* de la tabla *IO.config.detail_process_type*. Con respecto al código de los procedimientos almacenados, ver el apartado *Procedimientos BMWGroup_StatusUpdate* C del *Código fuente* C.

6.7.3. Ejecución y resultados del proceso

Tras la configuración del proceso, pasaremos por último a su ejecución y contrastaremos los resultados persistidos acordes con el fichero aportado.

Antes de comenzar con la ejecución del proceso, contrastamos que el fichero se encuentra en el directorio indicado del FTP como muestra la *Figura 6.38*.

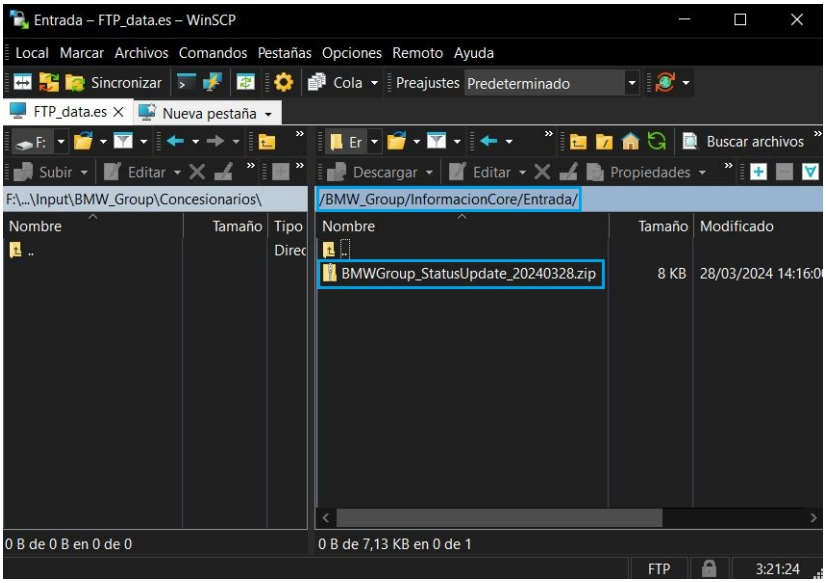


Figura 6.38: Fichero disponible en el directorio de entrada del FTP

Tras la revisión ejecutamos el proceso y contrastamos como la información va pasando por las diferentes tablas preparadas para almacenar la información (*Figura 6.39-6.45*).

modelo	cliente	tel_cliente	email_cliente	fecha_propuesta	cantidad_propuesta	tipo_propuesta	fecha_oferta
Serie 1	Mario Torres	555777999	mario@example.com	2024-03-01	43000	Nuevo	2024-03-10
Serie 2	Mario Torres	555777999	mario@example.com	2024-03-01	48000	Seminuevo	2024-03-10
S3	Mario Torres	555777999	mario@example.com	2024-03-01	55000	Nuevo	NULL
I4	Elena Navarro	666333222	elena@example.com	2024-03-02	70000	Nuevo	2024-03-08
X4 2024	Elena Navarro	666333222	elena@example.com	2024-03-02	90000	Leasing	2024-03-08
M2	Silvia Redondo	654321234	silvia@example.com	2024-03-03	80000	Nuevo	NULL
M3	Silvia Redondo	654321234	silvia@example.com	2024-03-03	120000	Renting	2024-03-12

Figura 6.39: Registros cargados en IO.input.BMWGroup_StatusUpdate

id	delivery_id	mdb_id	name	brand_type_id	fuel_type_id	active
1	1	NULL	I4	1	NULL	1
2	1	NULL	M2	1	NULL	1
3	1	NULL	M3	1	NULL	1

Figura 6.40: Registros cargados en IO.load.model_type

id	delivery_id	mdb_id	name	phone	email	concessionaire_id	active
1	1	NULL	Roberto Suarez	712345678	roberto@example.com	7	1

Figura 6.41: Registros cargados en IO.load.seller

id	delivery_id	mdb_id	name	phone	email
1	1	NULL	Elena Navarro	666333222	elena@example.com
2	1	NULL	Silvia Redondo	654321234	silvia@example.com

Figura 6.42: Registros cargados en IO.load.customer

id	delivery_id	mdb_id	customer_id	cliente	seller_id	vendedor	model_type_id	modelo	proposal_date	proposal_amount
1	1	NULL	NULL	Mario Torres	NULL	Ana Martín	NULL	Serie 1	2024-03-01	43000.00
2	1	NULL	NULL	Mario Torres	NULL	Ana Martín	NULL	Serie 2	2024-03-01	48000.00
3	1	NULL	NULL	Mario Torres	NULL	Ana Martín	NULL	Serie 3	2024-03-01	55000.00
4	1	NULL	NULL	Elena Navarro	NULL	David Díaz	NULL	I4	2024-03-02	70000.00
5	1	NULL	NULL	Elena Navarro	NULL	David Díaz	NULL	X4	2024-03-02	90000.00
6	1	NULL	NULL	Silvia Redondo	NULL	Roberto Suarez	NULL	M2	2024-03-03	80000.00
7	1	NULL	NULL	Silvia Redondo	NULL	Roberto Suarez	NULL	M3	2024-03-03	120000.00

Figura 6.43: Registros cargados en IO.load.proposal

id	delivery_id	mdb_id	proposal_id	proposal_io_id	offer_date	offer_amount	active
1	1	NULL	NULL	1	2024-03-10	38000.00	1
2	1	NULL	NULL	2	2024-03-10	45000.00	1
3	1	NULL	NULL	4	2024-03-08	65000.00	1
4	1	NULL	NULL	5	2024-03-08	80000.00	1
5	1	NULL	NULL	7	2024-03-12	110000.00	1

Figura 6.44: Registros cargados en IO.load.offer

id	delivery_id	mdb_id	offer_id	offer_io_id	sale_date
1	1	NULL	NULL	1	2024-03-22
2	1	NULL	NULL	3	2024-03-18
3	1	NULL	NULL	5	2024-04-21

Figura 6.45: Registros cargados en IO.load.sale

Si nos fijamos en la tabla auxiliar *BMWGroup_model_type_mapping* (*Figura 6.46*) esta debe contener tanto los modelos nuevos, como el mapeo de nombres incorrectos al valor por defecto de nuestros modelos.

id	model_type_file	model_type_table	brand_type	active
1	Serie 1	Serie 1	BMW	1
2	Serie 2	Serie 2	BMW	1
3	Serie 3	Serie 3	BMW	1
4	Serie 4	Serie 4	BMW	1
5	Serie 5	Serie 5	BMW	1
6	Serie 6	Serie 6	BMW	1
7	Serie 7	Serie 7	BMW	1
8	Serie 8	Serie 8	BMW	1
9	I3	I3	BMW	1
10	I8	I8	BMW	1
11	X1	X1	BMW	1
12	X2	X2	BMW	1
13	X3	X3	BMW	1
14	X4	X4	BMW	1
15	X5	X5	BMW	1
16	X6	X6	BMW	1
17	X7	X7	BMW	1
18	Cooper	Cooper	Mini	1
19	Countryman	Countryman	Mini	1
20	Clubman	Clubman	Mini	1
21	Convertible	Convertible	Mini	1
22	Paceman	Paceman	Mini	1
23	S3	Serie 3	BMW	1
24	I4	I4	BMW	1
25	X4 2024	X4	BMW	1
26	M2	M2	BMW	1
27	M3	M3	BMW	1
28	Coper	Cooper	Mini	1
29	Cbman	Clubman	Mini	1
30	Cabrio	Cabrio	Mini	1
31	5 Puertas	5 Puertas	Mini	1
32	John Coper Works	John Coper Works	Mini	1

Figura 6.46: Registros en la tabla BMWGroup_model_type_mapping

Posteriormente recibimos el correo como muestra la *Figura 6.47* con los resultados del proceso ejecutado de forma satisfactoria.

[BMW_GROUP] [SUCCEEDED]: StatusUpdate_Input execution summary on WIN-SER-AMAROTO

S server@data.es
Para amaroto@data.es

Running Server: [WIN-SER-AMAROTO]

Portfolio: [BMW_GROUP]

Execution summary for process: StatusUpdate_Input

Process description: Incorpora la información relacionada con el proceso de compra de un vehículo

Main summary

STATUS	START TIME	END TIME	TOTAL DURATION	MESSAGE
SUCCEEDED	27/04/2024 14:53:01	27/04/2024 14:53:23	00:00:22:313	

Phase summary

PHASE	STATUS	START TIME	END TIME	TOTAL DURATION
INGESTION	SUCCEEDED	27/04/2024 14:53:01	27/04/2024 14:53:20	00:00:19:727
TRANSFORMATION	SUCCEEDED	27/04/2024 14:53:20	27/04/2024 14:53:22	00:00:01:390
PERSISTENCE	SUCCEEDED	27/04/2024 14:53:22	27/04/2024 14:53:23	00:00:01:197

Detailed execution

PHASE	ENTITY	STATUS	AFFECTED ROWS	START TIME	END TIME
INGESTION	FTP_Download	SUCCEEDED	0	27/04/2024 14:53:01	27/04/2024 14:53:08
INGESTION	UNZIP_File	SUCCEEDED	0	27/04/2024 14:53:08	27/04/2024 14:53:17

Figura 6.47: Correo recibido del proceso BMWGroup_StatusUpdate

Adicionalmente, si consultamos las tablas podemos apreciar como contienen la nueva información procedente del fichero procesado (*Figuras 6.48-6.53*).

id	name	brand_type_id	fuel_type_id	active	creation_date	modification_date	io_id
23	I4	1	NULL	1	2024-04-27 14:53:22.563	NULL	1
24	M2	1	NULL	1	2024-04-27 14:53:22.563	NULL	2
25	M3	1	NULL	1	2024-04-27 14:53:22.563	NULL	3

Figura 6.48: Registros cargados en IO.load.model_type

id	name	phone	email	creation_date	modification_date	io_id
21	Elena Navarro	666333222	elena@example.com	2024-04-27 14:53:22.810	NULL	1
22	Silvia Redondo	654321234	silvia@example.com	2024-04-27 14:53:22.810	NULL	2

Figura 6.49: Registros cargados en IO.load.seller

id	name	phone	email	concessionaire_id	active	creation_date	modification_date	io_id
21	Roberto Suarez	712345678	roberto@example.com	7	1	2024-04-27 14:53:22.690	NULL	1

Figura 6.50: Registros cargados en IO.load.customer

id	customer_id	seller_id	model_type_id	proposal_date	proposal_amount	proposal_type_id	active	creation_date	modification_date	io_id
51	12	1	1	2024-03-01	43000.00	1	1	2024-04-27 14:53:22.940	NULL	1
52	12	1	2	2024-03-01	48000.00	2	1	2024-04-27 14:53:22.940	NULL	2
53	12	1	3	2024-03-01	55000.00	1	1	2024-04-27 14:53:22.940	NULL	3
54	12	9	23	2024-03-02	70000.00	1	1	2024-04-27 14:53:22.940	NULL	4
55	12	9	14	2024-03-02	90000.00	4	1	2024-04-27 14:53:22.940	NULL	5
56	12	21	24	2024-03-03	80000.00	1	1	2024-04-27 14:53:22.940	NULL	6
57	12	21	25	2024-03-03	120000.00	3	1	2024-04-27 14:53:22.940	NULL	7

Figura 6.51: Registros cargados en IO.load.proposal

id	proposal_id	offer_date	offer_amount	active	creation_date	modification_date	io_id
41	51	2024-03-10	38000.00	1	2024-04-27 14:53:23.160	NULL	1
42	52	2024-03-10	45000.00	1	2024-04-27 14:53:23.160	NULL	2
43	54	2024-03-08	65000.00	1	2024-04-27 14:53:23.160	NULL	3
44	55	2024-03-08	80000.00	1	2024-04-27 14:53:23.160	NULL	4
45	57	2024-03-12	110000.00	1	2024-04-27 14:53:23.160	NULL	5

Figura 6.52: Registros cargados en IO.load.offer

id	offer_id	sale_date	creation_date	modification_date	io_id
31	41	2024-03-22	2024-04-27 14:53:23.323	NULL	1
32	43	2024-03-18	2024-04-27 14:53:23.323	NULL	2
33	45	2024-04-21	2024-04-27 14:53:23.323	NULL	3

Figura 6.53: Registros cargados en IO.load.sale

Por último, el fichero procesado se archiva en nuestra ruta local y se insertan los datos en la tabla histórica *IO.hist.BMWGroup_StatusUpdate* (*Figuras 6.54-6.55*).

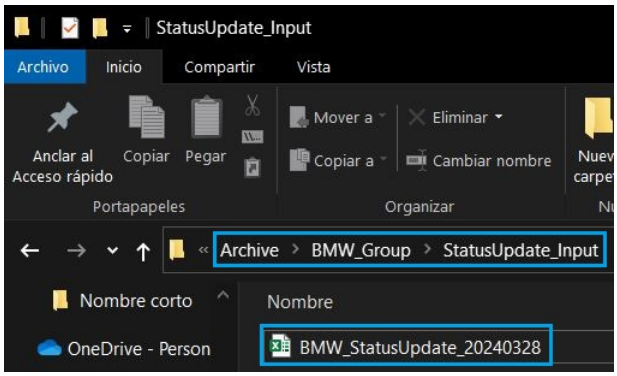


Figura 6.54: Fichero disponible en el directorio archive del proceso

master_process_log_id	modelo	cliente	tel_cliente	email_cliente	fecha_propuesta	cantidad_propuesta
1	Serie 1	Mario Torres	555777999	mario@example.com	2024-03-01	43000
1	Serie 2	Mario Torres	555777999	mario@example.com	2024-03-01	48000
1	S3	Mario Torres	555777999	mario@example.com	2024-03-01	55000
1	I4	Elena Navarro	666333222	elena@example.com	2024-03-02	70000
1	X4 2024	Elena Navarro	666333222	elena@example.com	2024-03-02	90000
1	M2	Silvia Redondo	654321234	silvia@example.com	2024-03-03	80000
1	M3	Silvia Redondo	654321234	silvia@example.com	2024-03-03	120000

Figura 6.55: Información cargada en la tabla IO.hist.BMWGroup_StatusUpdate

6.8. Conclusiones

En este capítulo se han analizado los casos de uso desarrollados para probar la funcionalidad de la solución IO, asegurando el cumplimiento de los requisitos funcionales vistos durante el *Análisis* 3.

Para más información sobre la generación de un proceso como los revisados durante este capitulo, ver el anexo *Manual de usuario* B.

Capítulo 7

Planificación y costes del proyecto

7.1. Introducción

En este capítulo vamos a mostrar la planificación para el desarrollo del proyecto, la cual se divide en las fases por las que ha estado compuesto, un diagrama de Gantt correspondiente a la planificación de tareas del proyecto, así como una estimación de costes asociados. El capítulo concluye con los comentarios acerca de la realización de la planificación, y el soporte que ha supuesto el haber realizado el proyecto con una buena planificación.

7.2. Planificación

Para el desarrollo del proyecto se ha considerado el uso del marco de trabajo ágil Scrum, el cual como se verá en esta sección ha permitido dividir los requisitos en pequeñas tareas asumibles, organizadas en *sprints* de dos semanas de duración. El grueso de tareas a realizar en cada sprint se ha planificado previo al comienzo de cada *sprint*. Adicionalmente el seguimiento de cada tarea se ha realizado mediante los estados pendiente, en curso o finalizada.

El haber elegido este marco de trabajo ha permitido tener una alta flexibilidad durante el desarrollo, permitiendo realizar correcciones, aportando/modificando requerimientos o documentación funcional mientras se continuaba con el desarrollo de las fases, a la vez que ha permitido prevenir, en la medida de lo posible, retrasos durante el desarrollo del proyecto.

7.2.1. Fases del proyecto

El proyecto ha estado compuesto de cuatro grandes fases, donde cada una de ellas ha tenido las siguientes tareas clave con los objetivos a alcanzar.

- Plan de trabajo y análisis: En este primera fase se ha llevado a cabo un estudio para conocer la información necesaria de cara a poder realizar el proyecto, y tras haber contrastado esta, se han planificado y estimado las diferentes tareas. La fase ha llevado aproximadamente un mes y medio de desarrollo.
    - T1.1: Búsqueda de información.
    - T1.2: Análisis inicial del proyecto.
    - T1.3: Planificación de tareas bajo nivel de detalle.
    - T1.4: Estimación de horas y fechas de completitud de las tareas.
- Diseño e implementación: La segunda fase ha contenido el grueso del desarrollo del pro-

yecto, la cual se ha centrado en dos tareas principales (T2.2 y T2.3) debido a que han sido las piezas centrales del proyecto. Sobre estas tareas se han ido desarrollando el conjunto de herramientas de soporte. La fase ha llevado aproximadamente cuatro meses y medio de desarrollo.

  - T2.1: Definición del modelo conceptual.
  - T2.2: Desarrollo de procesos ETL.
  - T2.3: Diseño de base de datos.
  - T2.4: Herramienta de encriptado y desencriptado de contraseñas.
  - T2.5: Procedimiento de ingesta de archivos en BBDD.
  - T2.6: Herramienta de carga y descarga de archivos por FTP/SFTP.
  - T2.7: Herramienta de compresión/descompresión de archivos.
  - T2.8: Script C# de envío de correos de información.

- Pruebas: La tercera fase ha compuesto el apartado de pruebas, para las cuales se han generado los diferentes casos de uso basados en la definición inicial del proyecto, se han revisado todas las funcionalidades implementadas, y se han automatizado los procesos mediante la creación de jobs. La fase ha llevado aproximadamente un mes de desarrollo.

  - T3.1: Generación de casos de uso.
  - T3.2: Test sobre los casos de uso.
  - T3.3: Revisión de funcionalidades básicas.
  - T3.4: Revisión de funcionalidades añadidas.
  - T3.5: Automatización de casos de uso a través de task scheduler.

- Documentación y entrega: Finalizas el resto de fases, por último se ha organizado la memoria junto al tutor y cotutor del proyecto, se ha desarrollado la documentación y se ha planificado la presentación para su defensa. La fase ha llevado aproximadamente dos meses de desarrollo.

  - T4.1: Estructura y organización de la memoria.
  - T4.2: Documentación de memoria del proyecto.
  - T4.3: Montaje de presentación para la defensa.
  - Contingencia: Periodo alternativo en caso de ser necesario.

Es importante destacar que ciertas fases y tareas se han podido paralelizar, permitiendo modificar etapas previas tras ir analizando o resolviendo problemas posteriores.

7.2.2. Diagrama de Gantt

Como complemento a las fases del proyecto se presenta el diagrama de Gantt de la *Figura 7.1*, donde se aprecia que el proyecto comenzó en octubre de 2023 y ha finalizado en mayo del 2024, manteniendo un periodo de contingencia, en caso de ser necesario tener que presentar finalmente en septiembre.

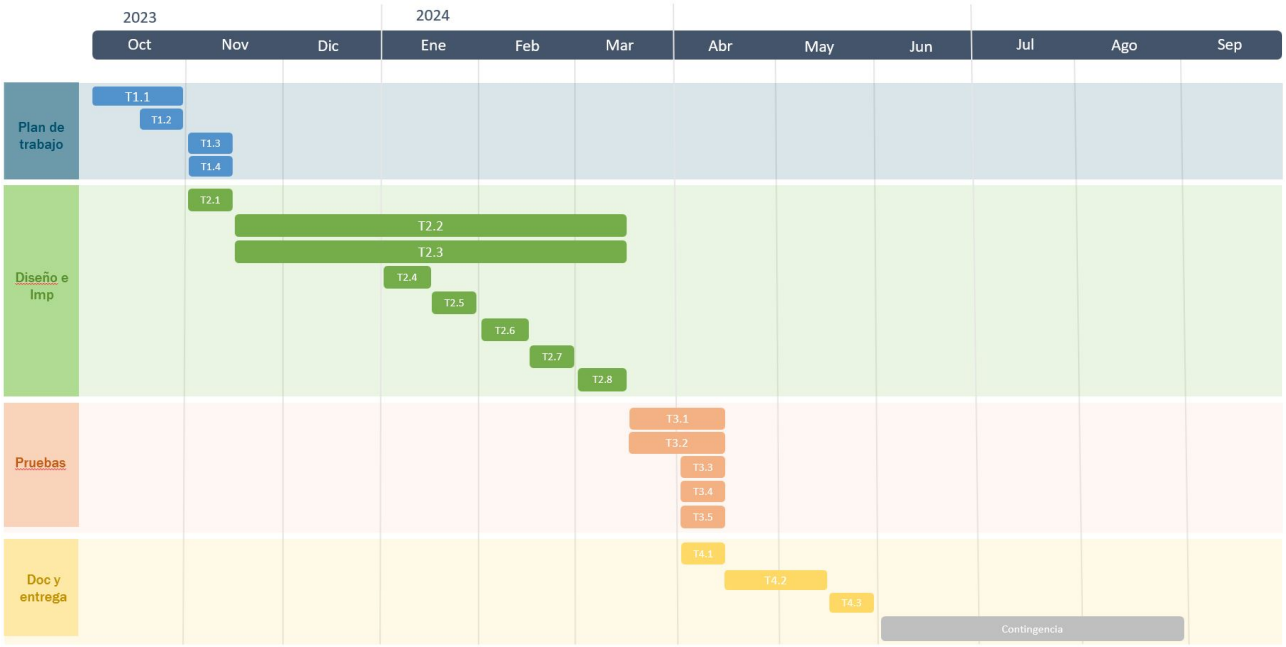


Figura 7.1: Diagrama de Gantt del proyecto

7.3. Costes del proyecto

Los costes que se han considerado para el proyecto han sido los siguientes.

- Las horas de desarrollo, divididas entre un equipo Scrum normal, formado por un *Scrum Master*, un *Product Owner* y un desarrollador.
- El coste de licencias necesarias, para las cuales se indica un promedio tras buscar su precio.
- Un servidor que permita desarrollar y alojar la solución.

La *Tabla 7.1* muestra un desglose de precios tomando como promedio un coste de 60 euros la hora para el equipo técnico.

Tabla 7.1: Tabla de costes del proyecto

Concepto	**Unidades**	**Precio**	**Total**
Horas Scrum Master	160	60 €	9600 €
Horas Product Owner	160	60 €	9600 €
Horas Desarrollador	960	60 €	57600 €
Licencias Windows Server	1	500 €	500 €
Licencias SQL Server	1	1000 €	1000 €
Licencias Microsoft Office	1	20 €	20 €
Servidor	1	2000 €	2000 €
			80.320 €

Para el manejo de unidades se ha tenido en cuenta un desarrollo de ocho meses, donde cada semana contiene cuarenta horas laborables. A su vez, tanto el *Scrum Master*, como el *Product Owner* han incurrido una hora diaria, mientras que el desarrollador ha imputado seis horas. Es importante destacar que el coste de licenciamiento tendría lugar únicamente en una empresa.

7.4. Conclusiones

En este capítulo se ha revisado la planificación del proyecto, dividida en fases, tanto detalladas como mostradas a través de un diagrama de Gantt. Adicionalmente se ha realizado una estimación de costes si se realizase su desarrollo en un entorno profesional.

Me gustaría remarcar que el haber dedicado un tiempo a la idea, el plan de trabajo a realizar, dividir y organizar el proyecto e informarme antes de comenzar el proyecto, ha supuesto una ayuda fundamental de cara al desarrollo de la solución, permitiendo enfocarme en el desarrollo al tener claras las fases, así como el software y la configuración que he necesitado durante el proyecto. El haber organizado el proyecto de esta manera se ha traducido en desvíos menores, y en general en un desarrollo centrado en los objetivos a realizar.

Capítulo 8

Conclusiones y trabajos futuros

8.1. Introducción

El capítulo final expone las conclusiones una vez ha finalizado el desarrollo del proyecto. Por un lado se detallan las conclusiones finales repasando la problemática y motivación del proyecto, continuando con los objetivos indicados al comienzo de esta memoria. Por otro lado la sección de trabajos futuros justifica posibles modificaciones que podrían realizarse en la primera versión de la solución como continuación del trabajo, detallando adicionalmente posibles problemáticas que puedan surgir.

8.2. Conclusiones

Como vimos en el capítulo *Introducción, objetivos y estructura* 1 tenía como principal motivación el desarrollo de una herramienta de gestión de datos de entrada y salida parametrizable realizada mediante el sistema ETL SSIS, apoyándose en el sistema de gestión de BBDD SQL Server, de cara a crear una solución robusta y escalable capaz de manejar eficientemente grandes volúmenes de datos entre diferentes origenes y destinos de una forma dinámica. Motivada debido a la necesidad actual de digitalización en las empresas las cuales se enfrentan a problemas como la eficiencia y gestión de sus procesos técnicos, tanto en coste como en tiempo.

Adicionalmente se destaca la capacidad para facilitar la integración de datos provenientes de diversas fuentes y su transformación según las necesidades del negocio, pudiendo apreciar una mayor agilidad en los procesos a implementar, así como una mejora en la calidad y la consistencia de los datos con respecto a otro tipo de soluciones, lo que se traduce en una toma de decisiones más informada y precisa por parte de la empresa.

Los objetivos marcados para el desarrollo de la solución se han cubierto en su totalidad, mediante el uso de la herramienta SSIS y el sistema de BBDD SQL Server, por un lado tal y como hemos visto en el apartado de diseño se ha creado una BBDD como almacén de datos para los procesos, y por otro se han diseñado las siguientes funcionalidades básicas y herramientas de soporte.

- Se ha incorporado la herramienta *Encriptación y desencriptación de contraseñas* 5.2.1 en diferentes procedimientos almacenados de SQL para dotar de un plus de robustez y seguridad a la solución.

- Se ha desarrollado la herramienta *Ingesta de archivos en BBDD* 5.2.2 en SQL que permite la ingesta de archivos utilizando como destino una tabla de BBDD, siendo los formatos de archivo permitidos CSV, DAT, TXT, XLS y XLSX.

- Se ha desarrollado la herramienta *Carga y descarga de archivos por FTP/SFTP* 5.2.3 en

PowerShell que permite la carga y/o descarga de archivos a través de protocolo FTP, aunque también está preparada para protocolo SFTP, que por motivos de arquitectura no se ha podido probar como se indica en el apartado *Trabajos futuros*.

- Se ha desarrollado la herramienta *Compresión y descompresión de archivos* 5.3 en PowerShell para permitirnos la compresión y descompresión de archivos GZIP, RAR y ZIP dándonos la posibilidad de hacerlo utilizando protección mediante contraseña.

- Se incorpora la herramienta *Generación de ficheros* 5.2.5 en PowerShell para generar un fichero en diferentes formatos a partir de una consulta, siendo los formatos de archivo permitidos CSV, DAT, TXT, XLS y XLSX.

- Se genera la fase *Proceso main* 5.3 o proceso maestro de la solución IO, necesario para realizar cualquiera de las fases de la solución, haciendo de director de cada uno de los pasos que hayamos programado para nuestro proceso.

- Se genera la fase *Proceso de ingesta* 5.4 o *Ingest_process* de la solución IO encargado de recoger la información procedente de diferentes orígenes, ya sean locales o remotos, almacenándola en tablas de la BBDD IO en el schema *input*.

- Se genera la fase *Proceso de transformación* 5.5 o *Transformation_process* de la solución IO encargado de adaptar la información ingestada al modelo de datos de destino, ejecutando una serie de procedimientos almacenados que generan una nueva delivery, además de un conjunto de transformaciones, almacenando los datos en tablas bajo el schema *load* en la BBDD IO.

- Se genera la fase *Proceso de persistencia* 5.6 o *Persistence_process* de la solución IO encargado de realizar las operaciones calculadas durante el proceso de transformación en una BBDD local o remota, las cuales pueden ser las siguientes INSERT, UPDATE y DELETE.

- Se genera la fase *Proceso de generación de outputs* 5.7 o *Output_process* de la solución IO encargado de generar informes a partir de consultas de BBDD, además de operaciones sobre ficheros Outputs como compresión, subida a servidor FTP/SFTP o envío de correo.

Todo ello acompañado de una capa adicional de trazabilidad y monitorización, almacenando el detalle pormenorizado de las ejecuciones realizadas, así como la integración de alertas por correo al usuario.

A continuación para poder probar las funcionalidades implementadas, se ha creado una BBDD con ejemplos cercanos a un entorno real junto a los siguientes casos de uso.

- *Caso de uso 1: Input - NewDealers csv* 6.3 presenta un proceso input simple con un fichero en formato *CSV*, el cual se procesa mediante la solución IO persistiendo la información en la BBDD *VehicleSales*.

- *Caso de uso 2: Output - TopSellers xlsx* 6.4 presenta un proceso output el cual genera un cuadro de mando completo para el cliente en un formato de salida *XLSX*, donde la información de origen se extrae de la BBDD *VehicleSales* a través de nuestra solución IO.

- *Caso de uso 3: Output - MiniSales txt* 6.5 el cual genera un informe simple para el cliente en un formato de salida *TXT*, donde la información de origen se extrae de la BBDD *VehicleSales* a través de nuestra solución IO.

- *Caso de uso 4: Input - RemoteConnections BBDD* 6.6 presenta un proceso input simple con un traspaso de información entre las BBDDs *VehicleSales* y *VehicleAlt*, el cual será procesado por nuestra solución IO.

- *Caso de uso 5: Input - StatusUpdate xlsx* 6.7 presenta un proceso input complejo con un fichero en formato *XLSX*, el cual será procesado por nuestra solución IO para persistir la información en la BBDD *VehicleSales*.

Por tanto, la solución de gestión de datos de entrada y salida parametrizable IO está preparada para desplegarse en los entornos apropiados y ponerse en funcionamiento tras la configuración inicial, dando así por finalizado el desarrollo de la primera versión del producto.

8.3. Trabajos futuros

Aunque la solución implementada se considera estable y robusta, siendo capaz de dar respuesta a un gran número de procesos de forma totalmente dinámica y automática, se ha podido ver durante su desarrollo diferentes funcionalidades que en una primera versión no estaban planteadas y que, por tiempo o limitaciones en la arquitectura planteada no ha sido posible añadirlas.

El siguiente listado muestra las mejoras comentadas las cuales podrían desarrollarse en sucesivas versiones de la solución.

- Añadir plantillas en la fase de ingesta y de salida para lenguajes semi estructurados o de etiquetas como JSON o XML: Durante los casos de uso se han planteado ejemplos muy comunes para el intercambio de datos como *CSV* o *XLSX*, sin embargo, en muchas ocasiones la información necesita ser recibida o enviada en formatos de etiquetas, pensados para una integración directa en soluciones que no conlleven una interacción manual. Esta funcionalidad debería añadirse en las herramientas de soporte *Ingesta de archivos en BBDD* 5.2.2 y en *Generación de ficheros* 5.2.5, se tendrían que revisar adicionalmente los procesos dtsx para la consistencia general. La problemática que puede acarrear la inclusión de esta funcionalidad sería por un lado el almacenamiento de la información, y por otro la extracción del dato de las etiquetas. Ver los siguientes artículos para una primera aproximación [14] y [15].

- Uso de consultoría remota a través de Polybase: El desarrollo planteado ha tenido en cuenta conexiones con motores de BBDD SQL Server, aunque este suele ser uno de los más usados a nivel empresarial existen todo tipo de motores y orígenes diferentes, es aquí donde entraría la funcionalidad *Polybase*, la cual permite que la instancia de SQL Server consulte datos directamente de orígenes de diferentes tipos, tanto *on-premise* como *cloud* sin necesidad de instalar de forma independiente software de conexión de cliente, integrándose esta información directamente en nuestro SQL Server para la cual sería necesario activar el servicio *SQL Server Polybase* en nuestro servidor. La problemática presentada en esta funcionalidad sería el rendimiento contra los orígenes remotos debido a la capa intermedia que incorpora esta solución, ya que los mismos tiempos de respuesta

no serán los mismos que los que podamos obtener consultando directamente al origen. Para mayor información ver los artículos [16] y [17].

- Ampliación de conexión a servidores SFTP: Aunque durante el desarrollo se ha tenido en cuenta la integración de servidores de tipo SFTP de cara a securizar las conexiones con este protocolo para el intercambio de datos, no se ha podido probar por limitaciones en la arquitectura planteada, así como la no disponibilidad de un servidor de dichas características. La implementación de esta característica se ha realizado en la tabla *IO.config.detail_process_type* detallada en *Diseño de BBDD y modelo entidad relación IO* 4.4 y en la herramienta PowerShell *Carga y descarga de archivos por FTP/SFTP* 5.2.3. En general esta funcionalidad solo requeriría probarse, por lo que se puede decir que el desarrollo está prácticamente realizado, una de las posibles problemáticas que se pueden encontrar sería la sincronización con el cliente mediante el uso de claves seguras punto a punto *SSH*, aunque la gran mayoría de compañías están preparadas para este tipo de conexión, por lo que debería ser algo facilmente solventable. Ver el siguiente artículo para conocer la integración con *WinSCP* [18].

- Abstraer el proceso de delivery de la fase de transformación: Tras el desarrollo e implementación de la solución se ha tenido en mente la modularización del proyecto, para según la necesidad del proceso a tratar se pudiesen lanzar las fases que fuesen estrictamente necesarias, en un primer análisis la fase de *delivery* se consideró para ser ejecutada junto a la fase de *transformación*, sin embargo durante la preparación de los diferentes casos de uso se hizo latente la necesidad de separar ambas etapas, para aquellos casos donde el proceso es tan simple que no hace falta transformar los datos, como pudimos ver en *Caso de uso 4: Input - RemoteConnections BBDD* 6.6. Para poder aplicar esta mejora se debería aislar el proceso de *delivery* en un paquete dtsx individual y realizar las modificaciones pertinentes tanto en el paquete *main*, como en el de *transformación* en nuestra solución SSIS. Al tener como ejemplo el resto de fases e incluso disponer de la lógica en el paquete de *transformación* la modificación no debería ser problemática, por lo que se necesitaría tiempo para aplicar el cambio y realizar una batería de pruebas con procesos simples para contrastar que la arquitectura general de la solución no ha variado.

Como se puede apreciar, todo desarrollo de software tiene margen de mejora sabiendo cuando se comienza un proyecto pero en raras ocasiones cuando se finaliza.

Bibliografía

[1] Microsoft. (30 de agosto de 2023) Introducción a windows server. [Online]. Available: https://learn.microsoft.com/es-es/windows-server/get-started/get-started-with-windows-server

[2] Microsoft. (9 de enero de 2024) Guía de instalación de sql server. [Online]. Available: https://learn.microsoft.com/es-es/sql/database-engine/install-windows/install-sql-server

[3] Microsoft. (3 de septiembre de 2023) Tutorial: Introducción al motor de base de datos. [Online]. Available: https://learn.microsoft.com/es-es/sql/relational-databases/tutorial-getting-started-with-the-database-engine

[4] Microsoft. (29 de octubre de 2023) Guía de solución de problemas de la extensión de proyectos ssis para vs2019. [Online]. Available: https://learn.microsoft.com/es-es/sql/ssdt/ssis-vs2019-troubleshooting-guide

[5] Microsoft. (04 de marzo de 2024) about_execution_policies. [Online]. Available: https://learn.microsoft.com/es-es/powershell/module/microsoft.powershell.core/about/about_execution_policies

[6] T. Freudenberg. (7 de enero de 2024) Powershell module for creating and extracting 7-zip archives supporting powershell's writeprogress api. [Online]. Available: https://github.com/thoemmi/7Zip4Powershell

[7] J. Fossen. (6 de junio de 2016) Powershell 7-zip module versus compress-archive with encryption. [Online]. Available: https://www.sans.org/blog/powershell-7-zip-module-versus-compress-archive-with-encryption

[8] Martín. (30 de octubre de 2023) Using winscp .net assembly from powershell. [Online]. Available: https://winscp.net/eng/docs/library_powershell

[9] Redgate. (8 de abril de 2014) Using sql search. [Online]. Available: https://documentation.red-gate.com/ss1/using-sql-search

[10] Wikipedia. (17 de mayo de 2024) Normalización de bases de datos. [Online]. Available: https://es.wikipedia.org/wiki/Normalizacin_de_bases_de_datos

[11] Microsoft. (31 de mayo de 2023) Transacciones sql. [Online]. Available: https://learn.microsoft.com/es-es/sql/t-sql/language-elements/transactions-transact-sql

[12] Martín. (16 de febrero de 2023) Winscp documentation. [Online]. Available: https://winscp.net/eng/docs

[13] Microsoft. (15 de noviembre de 2023) Utilidad de copia masiva bcp. [Online]. Available: https://learn.microsoft.com/es-es/sql/tools/bcp-utility

[14] Microsoft. (21 de mayo de 2024) Datos json en sql server. [Online]. Available: https://learn.microsoft.com/es-es/sql/relational-databases/json/json-data-sql-server

[15] Microsoft. (4 de abril de 2023) Xml, archivos de formato (sql server). [Online]. Available: https://learn.microsoft.com/es-es/sql/relational-databases/import-export/xml-format-files-sql-server

[16] Microsoft. (16 de enero de 2024) Características y limitaciones de polybase. [Online]. Available: https://learn.microsoft.com/es-es/sql/relational-databases/polybase/polybase-versioned-feature-summary

[17] SQLShack. (28 de febrero de 2017) Sql server 2016 – tutorial de polybase. [Online]. Available: https://www.sqlshack.com/es/sql-server-2016-tutorial-de-polybase

[18] Martín. (12 de agosto de 2022) Sftp task for ssis/ssdt. [Online]. Available: https://winscp.net/eng/docs/guide_ssis

[19] B. Petrovic. (21 de mayo de 2019) Cómo poder configurar la base de datos del correo en sql server. [Online]. Available: https://www.sqlshack.com/es/como-poder-configurar-la-base-de-datos-del-correo-en-sql-server

[20] C. INFORMATICAS. (13 de febrero de 2021) Configuración de hmailserver en windows server 2016. [Online]. Available: https://www.youtube.com/watch?v=BJoP7c9DLS0&ab_channel=COSICASINFORMATICAS

[21] M. Knafve. (12 de marzo de 2023) hmailserver documentation. [Online]. Available: https://www.hmailserver.com/documentation

[22] Microsoft. (7 de agosto de 2023) Descarga de sql server data tools (ssdt) para visual studio. [Online]. Available: https://learn.microsoft.com/es-es/sql/ssdt/download-sql-server-data-tools-ssdt

[23] Microsoft. (9 de abril de 2024) Descarga de sql server management studio (ssms). [Online]. Available: https://learn.microsoft.com/es-es/sql/ssms/download-sql-server-management-studio-ssms

[24] Microsoft. (23 de mayo de 2023) Implementación de proyectos y paquetes de integration services (ssis). [Online]. Available: https://learn.microsoft.com/es-es/sql/integration-services/packages/deploy-integration-services-ssis-projects-and-packages

[25] R. Velasco. (1 de marzo de 2024) Descargar e instalar o reinstalar microsoft 365 u office 2021 en un equipo pc o mac. [Online]. Available: https://www.softzone.es/windows/como-se-hace/descargar-instalar-office

[26] Solvetic. (28 de enero de 2021) Crear dominio e instalar y configurar active directory en windows server 2022. [Online]. Available: https://www.youtube.com/watch?v=_FOXZ8MujKg&ab_channel=solvetic.com

[27] Solvetic. (11 de agosto de 2020) Instalar y configurar ftp en windows server 2019. [Online]. Available: https://www.youtube.com/watch?v=9xfHoCZ5VtY&ab_channel=solvetic.com

Glosario de términos

BBDD (Base de Datos) Colección organizada de datos que se almacenan y acceden electrónicamente desde un sistema informático.. 1–4, 8, 10, 13–15, 21–25, 29, 31, 35, 41, 43, 44, 47, 48, 51, 69, 81, 85, 88, 93, 94, 100, 104, 109, 110, 112, 114, 115, 117, 125–129, 131–134, 138, 143, 144, 148, 149, 153, 154, 156, 157, 159–161, 168, 173–175, 187, 196–198, 213, 224, 232, 233, 242, 249, 261, III

BCP (Bulk Copy Program) Utilidad de línea de comandos que se utiliza para importar y exportar grandes volúmenes de datos entre SQL Server y archivos de datos.. 66, 196

BULK INSERT Comando en SQL Server que permite la inserción eficiente de grandes volúmenes de datos en una tabla sin necesidad de insertar cada fila de forma individual.. 54

C# Lenguaje de programación orientado a objetos, moderno y de propósito general desarrollado por Microsoft.. 7, 8, 90, 117, 119, 168, 213, 215, 217

collation Conjunto de reglas que determina cómo se comparan y ordenan los caracteres al realizar operaciones en una base de datos.. 197

core Núcleo o componente central de un sistema o aplicación.. 249, XVIII

delivery Concepto de bloque de ejecución que permite la paralelización de procesos en IO.. 2, 94, 96–101, 104, 107–109, 112, 114, 144, 157, 160, 174, 200, 201, 209, 210

dtsx Extensión de archivos de paquete de Microsoft SQL Server Integration Services (SSIS).. 2, 8, 16, 24, 29, 50, 175, 176, 213

ETL (Extract, Transform, Load) Diferentes etapas de un proceso de integración para mover datos desde un origen a un destino.. 1, 8, 10, 11, 22, 23, 173, III, V

fingerprint (Huella digital del servidor) Representación única de la identidad de un servidor, que se utiliza para verificar su autenticidad y establecer conexiones seguras.. 59

FTP (File Transfer Protocol) Protocolo estándar utilizado para la transferencia de archivos entre un cliente y un servidor en una determinada red.. 2, 9, 14, 22–25, 28, 33, 34, 57–59, 70, 93, 114, 119, 143–145, 148, 149, 151, 153–155, 159, 160, 162, 168, 174, 195

HTML (HyperText Markup Language) Lenguaje estándar utilizado para la creación y diseño de páginas web.. 76

input Entrada de datos o información introducida en la solución para ser procesada o manipulada.. 112, 143, 156, 159, 174, 175, 195, 200, 203, 205, 266, 273, 275, 282, 288, 291

IO Input/Output (Entrada/Salida) Se refiere a la gestión o intercambio de datos entre sistemas informáticos.. 2, 8, 10, 18, 21, 22, 29, 44, 47, 48, 57, 61, 64, 68, 70, 81, 88, 93, 94, 100, 104, 114, 123, 125–134, 143, 144, 148, 149, 153, 154, 156, 157, 159, 160, 165, 174, 175, 187, 188, 190, 196, 197, 213, 224, 229, 232, 249, 261, 266, 282, 303

job Tareas automatizadas en SQL Server que se ejecutan según un horario o evento específico dentro de la base de datos.. 22, 24, 168, 191–193, 196, 229

OLAP (Online analytical process) Sistema de administración de base de datos que se utiliza para los sistemas analíticos, optimizado para consultas de datos complejos y adaptan para sistemas de grandes volúmenes de datos.. 10

OLEDB (Object Linking and Embedding, Database) API de Microsoft que permite el acceso y manipulación de datos desde una variadad de fuentes.. 186

OLTP (Online transactional process) Sistema de gestión de bases de datos que se utiliza para sistemas transaccionales, diseñado para registrar y administrar datos en tiempo real.. 10

OPENROWSET Función en SQL Server que permite a los usuarios realizar operaciones de lectura y escritura en archivos externos o fuentes de datos remotas directamente desde una consulta SQL.. 54

output Salida de datos o resultados producidos en la solución tras ser procesados.. 8, 16, 22, 30, 33, 34, 112, 114, 115, 117, 119, 122, 123, 148–150, 153, 154, 174, 195, 204, 269, 271, 287

PowerShell Entorno de shell y lenguaje de scripting desarrollado por Microsoft.. 7–9, 22, 57, 61, 64, 93, 119, 174, 176, 219

schema Esquema o estructura lógica y organizativa de base de datos.. 23, 30, 43, 81, 94, 105, 161, 174, 203, 210

script Archivo de texto plano que contiene una serie de comandos o instrucciones que pueden ser ejecutados por un intérprete de comandos o un entorno de ejecución específico.. 4, 8, 9, 22, 43, 44, 70, 76, 87, 88, 90, 93, 94, 100, 117, 119, 138, 143, 145, 150, 154, 157, 161, 168, 193, 203, 211, 266

scrum Marco de trabajo ágil que enfatiza la colaboración, la adaptabilidad y la entrega incremental de valor comúnmente utilizado en el desarrollo de software y proyectos de gestión.. 167, 169

SFTP (Secure File Transfer Protocol) Protocolo de transferencia segura de archivos entre un cliente y un servior. Utiliza una conexión segura SSH (Secure Shell) para cifrar todos los datos transferidos.. 9, 15, 22, 34, 57–59, 70, 93, 114, 119, 168, 174, 176

SMTP (Simple Mail Transfer Protocol) Protocolo de red utilizado para el envío de correos electrónicos a través de Internet.. 22, 121

snake_case Convención de escritura de código en la que las palabras se separan con guiones bajos y todas las letras son minúsculas.. 29, 125, 213

SQL (Structured Query Language) Lenguaje de programación utilizado para administrar y manipular bases de datos relacionales.. 8, 11, 51, 70, 74, 88, 106, 107, 111, 112, 115, 173, 186, 213, 224, 229, 232, 233, 242, 249, 261, 266, 269, 271, 275, 282, 287, 288, 291

SQL Server Sistema de gestión de base de datos relacional, desarrollado por Microsoft, el cual utiliza Transact-SQL como lenguaje de desarrollo.. 1, 8, 10, 15, 22, 24, 44, 48, 52, 54, 86, 123, 143, 173, 175, 185, 187, 190, 196, III, V

SQL Server Agent Servicio de Microsoft Windows que ejecuta tareas administrativas programadas, denominadas trabajos, en SQL Server.. 8, 21, 191

SSIS SQL Server Integration Services (SSIS) Plataforma de Microsoft utilizada para construir soluciones de integración de datos o ETL.. 1, 2, 4, 8–11, 21–24, 45, 47, 50, 57, 58, 62, 74, 98, 101, 110, 115, 123, 173, 176, 187, 206, 213, 219, III, V

SSISDB Catálogo de base de datos en SQL Server necesario implementar proyectos de Integration Services (SSIS).. 187, 188, 190

trigger Tipo especial de procedimiento almacenado que se activa automáticamente en respuesta a ciertos eventos ocurridos en una tabla o vista de la bbdd.. 31, 32, 34, 37, 126–134

UML (Unified Modeling Language) Lenguaje gráfico de modelado de sistemas de software, incluye aspectos conceptuales tales como procesos, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación o esquemas de bases de datos.. 13, 16

Anexo A

Manual de despliegue

Introducción

En este anexo se va describir la instalación de todos los elementos necesarios para que la aplicación se pueda probar y ejecutar en un entorno bajo el sistema operativo Windows Server 2022. Es importante destacar que el entorno está previamente configurado, lo cual implica haber instalado y/o configurado las siguientes herramientas:

- SQL Server[2]
- SQL Server Management Studio[23]
- Visual Studio Community Edition[22]
- Integration Services[22]
- hMailServer[20][21]
- Microsoft Outlook[25]
- WinSCP .NET assembly[12]
- 7zip4Powershell[6]
- Servicio de dominio de Active directory[26]
- Servidor web (IIS)[27]
- Servidor FTP[27]
- Configuración de políticas PowerShell[5]
- Configuración Database Mail en SQL Server[19]

Instalación de librerías Office

Para el correcto funcionamiento de la herramienta, así como la ingesta y generación de archivos en formato Excel, es necesario que las siguientes librerías se encuentren instaladas y registradas en SQL Server:

- Microsoft Access Database Engine 2010 Redistributable

- Microsoft Access Database Engine 2016 Redistributable

Debemos verificar que se trata de la versión de 64 bits. Una vez instaladas ambas librerías, estas aparecen dentro de objetos del servidor como proveedores OLEDB disponibles, tendremos que habilitar las opciones de *Dynamic parameter* y *Allow inprocess* (*Figura A.1*), de lo contrario podrían producirse errores en la lectura y generación de archivos Excel.

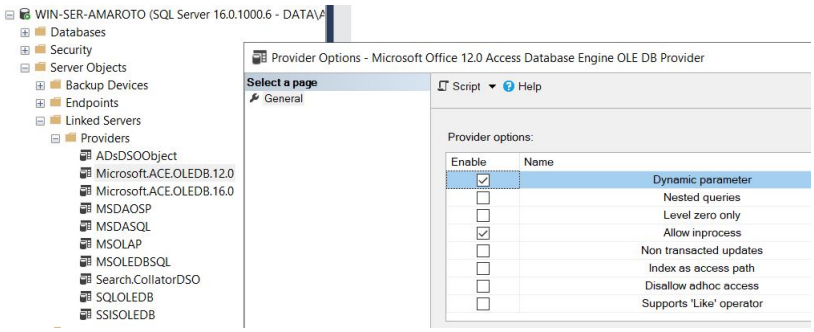


Figura A.1: Habilitar opciones en los proveedores instalados

Adicionalmente, la *Figura A.2* comprueba el resultado mediante ejecución de la consulta del *Código A.1*.

Código A.1: Proveedores del servidor

```
1    EXEC sp_MSset_oledb_prop
```

provider_name	allow_in_process	disallow_adhoc_access	dynamic_parameters	index_as_access_path	level_zero_only	nested_queries	non_transacted_updates	sql_server_like
SQLOLEDB	0	0	0	0	0	0	0	0
Microsoft.ACE.OLEDB.12.0	1	0	1	0	0	0	0	0
Microsoft.ACE.OLEDB.16.0	1	0	1	0	0	0	0	0
ADsDSOObject	1	0	0	0	0	0	0	0
MSOLEDBSQL	1	0	0	0	0	0	0	0
Search.CollatorDSO	0	0	0	0	0	0	0	0
SSISOLEDB	0	0	0	0	0	0	0	0
MSDASQL	1	0	0	0	0	0	0	0
MSOLAP	1	0	0	0	0	0	0	0
MSDAOSP	0	0	0	0	0	0	0	0

Figura A.2: Resultado devuelto por la consulta

Por otro lado, para que la herramienta funcione correctamente es necesario que los servicios del motor de base de datos, Integration Services y Agente SQL estén en ejecución de manera indefinida (Automático) como muestra la *Figura A.3*.

Nombre	Descripción	Estado	Tipo de inicio	Iniciar sesión como
SQL Server Integration Services 16.0	Proporciona administración para el almacenamiento y la ejecución de paq...		Automático	.\Administrador
SQL Server (MSSQLSERVER)	Proporciona almacenamiento, procesamiento y acceso controlado de datos...	En ejecu...	Automático	.\Administrador
SQL Full-text Filter Daemon Launcher (MSSQLSERVER)	Servicio que iniciará el proceso del demonio de filtro de texto completo, el ...	En ejecu...	Automático	.\Administrador
SQL Server CEIP service (MSSQLSERVER)	CEIP service for Sql server		Deshabilitado	.\Administrador
SQL Server Integration Services CEIP service 16.0	CEIP service for Sql server Integration Services		Deshabilitado	.\Administrador
SQL Server Launchpad (MSSQLSERVER)	Servicio que permite iniciar el proceso de Extensiones de análisis avanzado...		Deshabilitado	.\Administrador
Agente SQL Server (MSSQLSERVER)	Ejecuta tareas, realiza el seguimiento de SQL Server, activa alertas y permite...		Automático	DATA\Administrador

Figura A.3: Servicios SQL en ejecución automático

Creación de catálogo SSISDB

Para poder realizar el despliegue de la solución, es necesario crear el catálogo SSISDB en BBDD. Una vez en la ventana de creación, habilitamos la ejecución automática de Integration Services al iniciar SQL Server y proporcionamos una contraseña de encriptación.

Es muy importante almacenar la contraseña de la *Figura A.4* en un lugar seguro, nos la solicitará en caso de migrar el catálogo a otro servidor.

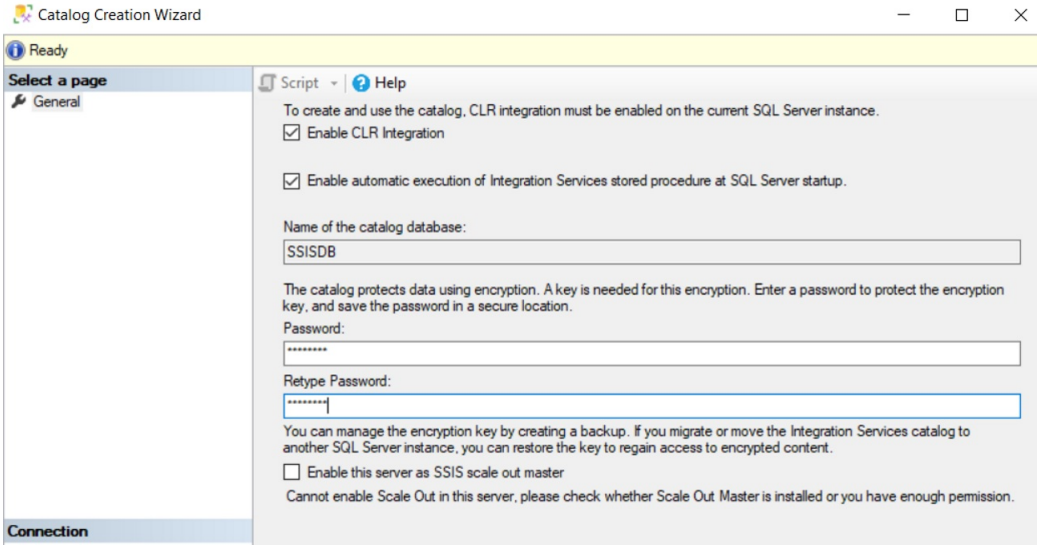


Figura A.4: Ventana de creación de catálogo SSISDB

Tras este paso, estaremos preparados para poder realizar despliegues de proyectos SSIS en SQL Server.

Implementación de la solución

Una vez creado el catálogo SSISDB, el siguiente paso a realizar es el despliegue de nuestra solución en el servidor.

Para ello, realizamos el proceso desde nuestra solución IO en SSIS. Una vez abierta, se pueden realizar despliegues de paquetes individuales, o bien subir un proyecto completo, el cual va a ser nuestro caso. Para ello, seleccionamos IO, botón derecho e implementar.

En la *Figura A.5* se solicita el destino de implementación, en nuestro caso será SSIS en SQL Server.

En la siguiente ventana tendremos que indicar el servidor al que nos queremos conectar, que en nuestro caso es *WIN-SER-AMAROTO*, así como la ruta de acceso donde queremos almacenar nuestro proyecto, en la *Figura A.6* se puede apreciar un nuevo directorio *Data* y la solución IO. Por lo tanto en caso de que ya exista el proyecto, se reemplazará la versión antigua por la nueva.

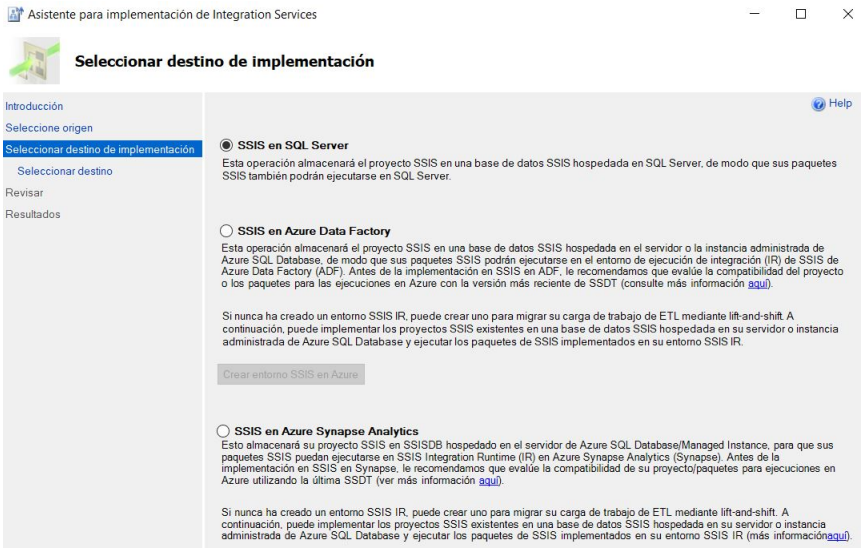


Figura A.5: Selección destino de implementación

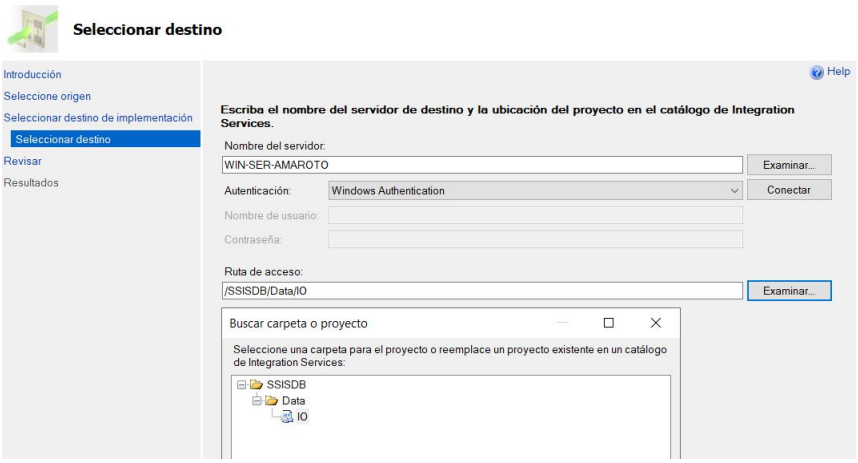


Figura A.6: Selección ubicación del proyecto en SSISDB

Por último, en la *Figura A.7* podemos ver el resultado del despliegue.

Figura A.7: Ventana de resultados tras la subida

Tras ello, nuestra solución se habrá desplegado correctamente en el servidor, como revisión final si expandimos el catálogo SSISDB (*Figura A.8*), nos aparecerá tanto el nuevo directorio, como el proyecto IO, si accedemos a sus propiedades, podemos ver la última fecha de despliegue realizada.

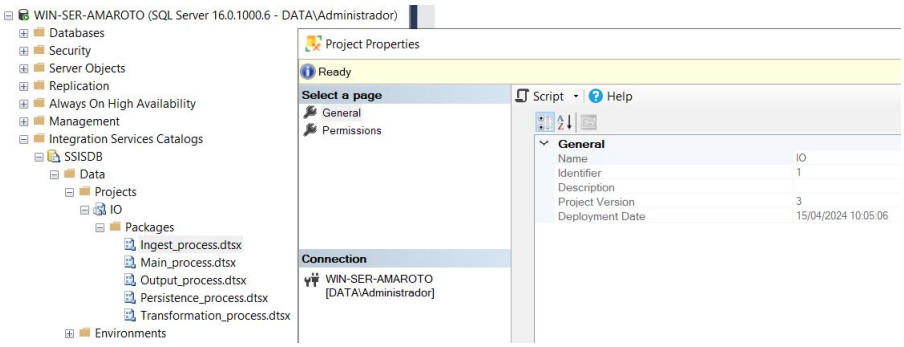


Figura A.8: Propiedades del proyecto en SQL Server

Podemos acudir al artículo de la guía oficial para obtener más información [24].

Configuración de parámetros

Una vez realizado el despliegue del proyecto, será necesario crear variables de entorno, las cuales coinciden con los parámetros de entrada del *Proceso main* 5.3, y se encuentren mapeadas dentro de la configuración del paquete. Por lo tanto, creamos el entorno y generamos las variables como se muestra las *Figuras A.9-A.10*.

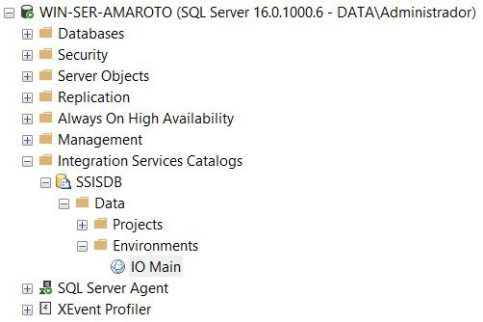


Figura A.9: Entorno IO Main

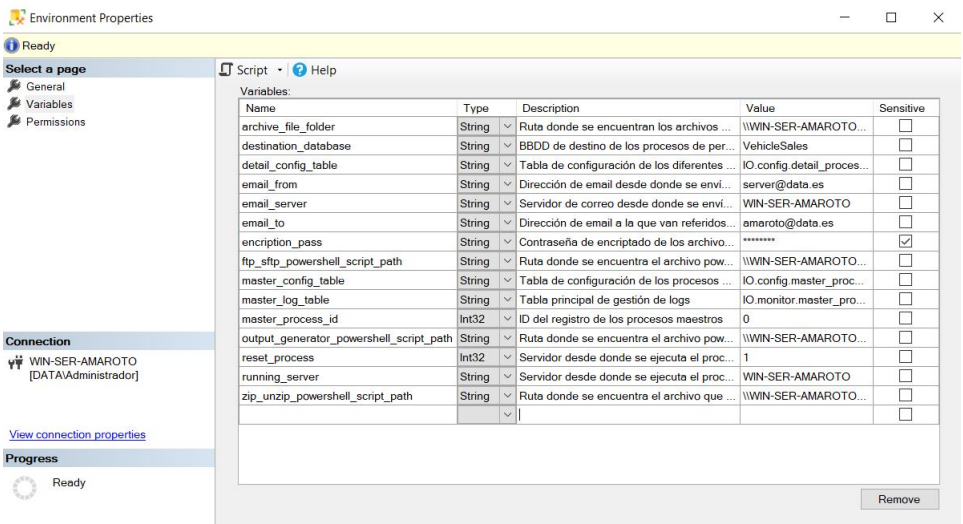


Figura A.10: Variables del entorno IO Main

Como se puede apreciar, se indican todos sus campos, siendo clave el nombre, tipo y valor, podemos marcar *Sensitive* para información sensible como contraseñas, en nuestro caso *encription_pass*. De esta forma el valor de la variable no será visible.

El siguiente paso consiste en configurar el proyecto dentro del catálogo, para ello accedemos a *Configure* de IO, y añadimos en el apartado de referencias el entorno recién creado como muestra la *Figura A.11*.

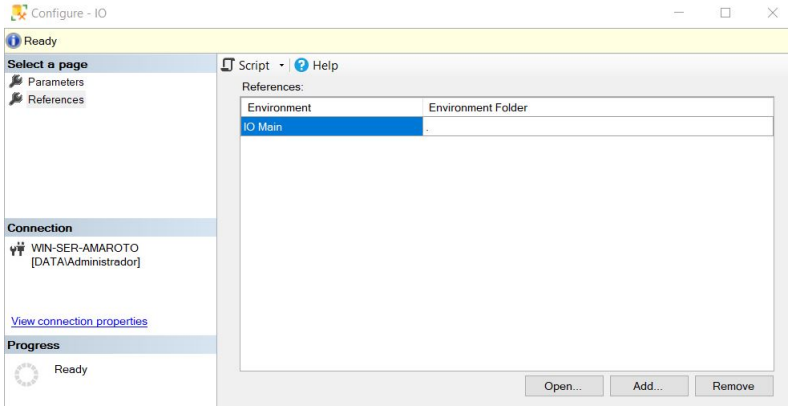


Figura A.11: Asignación de entorno de referencias en el proyecto

Una vez añadido, si los nombres de las variables de entorno coinciden con los de los parámetros, se habrán asignado de forma automática como se aprecia en la *Figura A.12*.

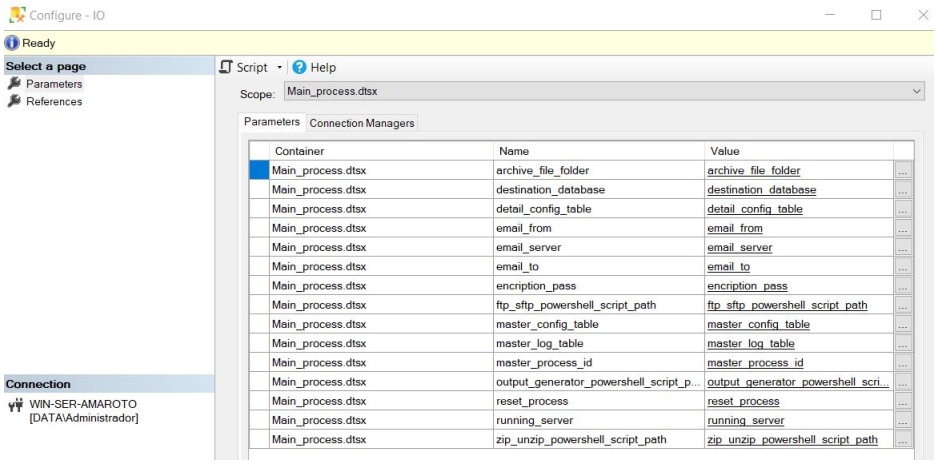


Figura A.12: Asignación de variables de entorno en parámetros

Es importante destacar que el único valor de nuestras referencias que será editable es *master_process_id*, el cual depende del proceso que se desea ejecutar en cada ocasión.

Si se han seguido todos los pasos hasta el momento, ya tendremos completamente configurada nuestra solución IO en SQL Server. Podremos ejecutar nuestros procesos directamente desde el catálogo SSISDB, sin embargo, se busca que la ejecución de nuestros desarrollos se realice de forma autónoma, para ello realizaremos un último paso.

Automatización mediante Jobs

Para finalizar el manual de despliegue, vamos a crear un job que nos permita ejecutar un proceso del proyecto y, en caso necesario, automatizarlo estableciendo una periodicidad determinada. Generamos un job para el *Caso de uso 1: Input - NewDealers csv* 6.3.

Creamos el job desde SQL Server Agent y le damos un nombre como se muestra en las *Figuras A.13-A.14*.

Figura A.13: Carpetas del apartado SQL Server Agent

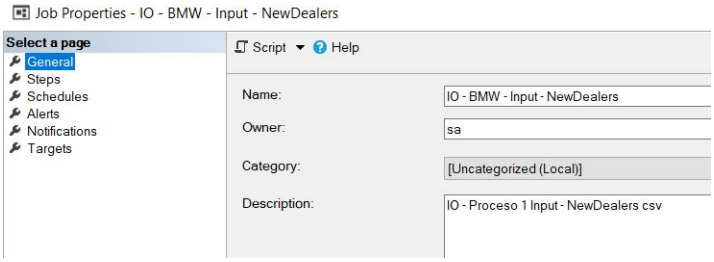


Figura A.14: Página general del job

En la página *Steps* invocaremos al proceso principal de la solución, indicando *SQL Server Integration Services Package* en el tipo y conectándonos al servidor como se aprecia en la *Figura A.15*.

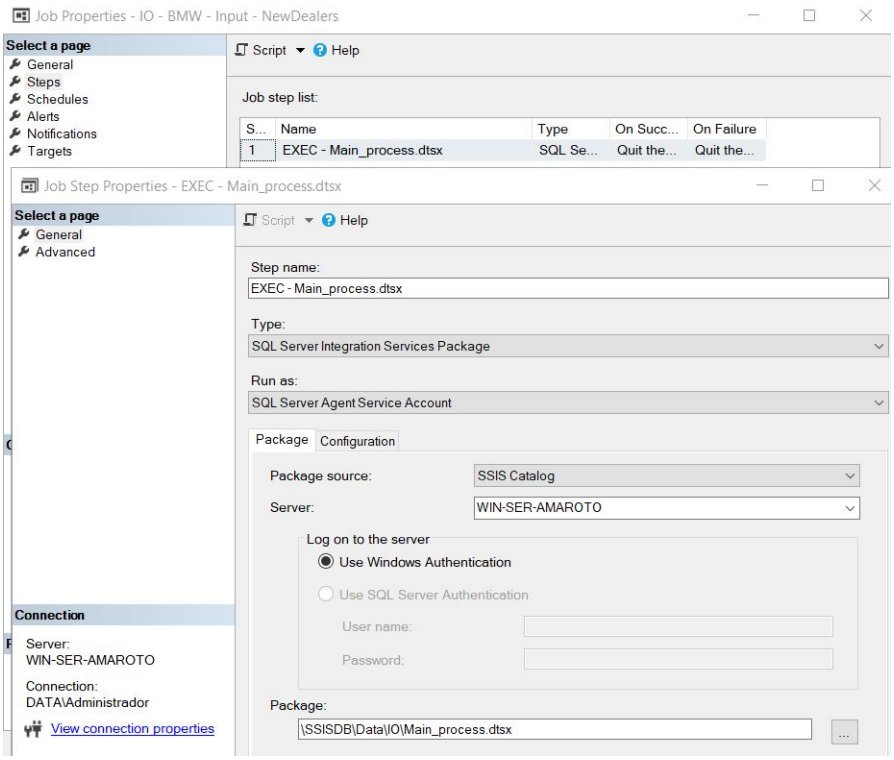


Figura A.15: Asignación de tipo y servidor en la página steps del job

Con respecto a la pestaña *Configuration*, es importante destacar tanto el marcaje del uso de variables de entorno, como de asignar el valor del identificador del proceso maestro tal y como se muestra en la *Figura A.16*.

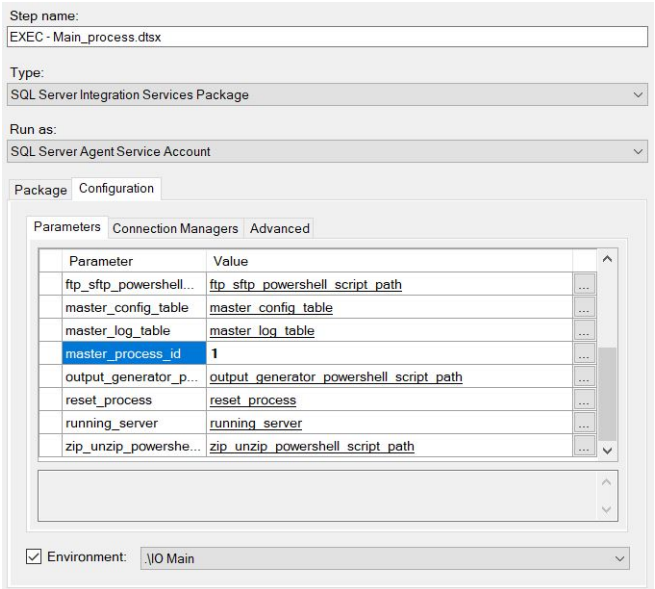


Figura A.16: Configuración del paquete en la página steps del job

Una vez generado, ejecutamos nuestro nuevo job, si todo ha ido correctamente aparecerá el mensaje *Success* como muestra la *Figura A.17*. Adicionalmente podremos ver el histórico de ejecuciones desde el monitor de actividades como se muestra a continuación.

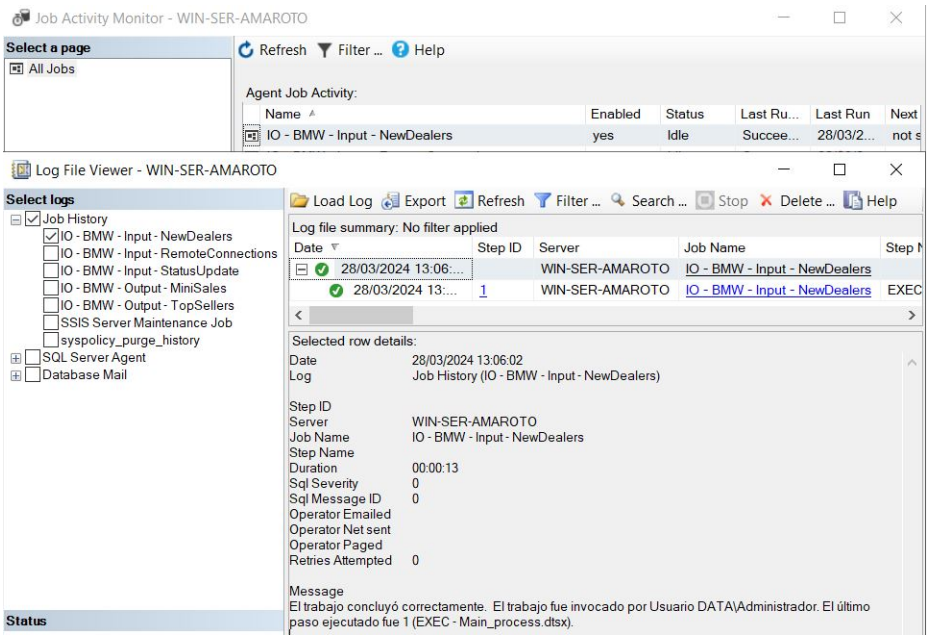


Figura A.17: Histórico de ejecuciones del job creado

Para ver el script de creación de jobs del proyecto ir a C.11 del *Anexo C*.

Anexo B

Manual de usuario

Introducción

En este anexo vamos a resumir los puntos más relevantes a tener en cuenta como usuario a la hora de generar un nuevo proceso en la herramienta. Para ver información acerca de como crear diferentes procesos ir al capítulo *Pruebas* 6, donde se explica en detalle los pasos a realizar.

Revisión de requisitos

Como usuarios de la solución, el primer punto que deberemos realizar será revisar la solicitud del nuevo proceso, de cara a poder obtener la definición funcional a través de los requisitos indicados por el negocio.

Mediante este primer paso podremos establecer si se trata de un proceso input, un proceso output, o quizá ambos tipos. Adicionalmente, tendremos que revisar el número de elementos que vamos a necesitar procesar, o lo que es lo mismo, el número de ingestas, transformaciones, persistencias y/o informes de salida.

Como ejemplo, imaginemos que tenemos el siguiente caso:

- Solicitud de un nuevo proceso para tratar información en nuestro sistema.
- Esta información está contenida en varios ficheros con diferente formato.
- A su vez la información trata diferentes conceptos.
- Los ficheros a procesar serán depositados en el FTP del cliente.
- Estos contendrán una foto completa de los datos recibidos de forma histórica.
- La periodicidad de recepción de ficheros será semanal.

En consecuencia, o como respuesta al ejemplo sería conveniente pensar lo siguiente:

- Se trata de un proceso input.
- Vamos a necesitar varios procesos de ingesta para tratar cada fichero.
- Diferentes conceptos supondrá crear varias transformaciones/persistencias por cada uno.
- Necesitaremos las credenciales de acceso al servidor y una fase de ingesta para descargar los ficheros.

- Indicador claro de que habrá que insertar o actualizar la información recibida mediante reglas.
- Crearemos un job que ejecute el proceso semanalmente.

Aunque una solicitud debería contener mayor detalle, el ejemplo nos da una idea general de como enfocar el análisis. Es importante remarcar que será necesario conocer las diferentes apreciaciones del proceso, como por ejemplo la omisión de registros en un fichero de entrada, o una regla específica que nos indique cuando actualizar un registro existente en nuestra BBDD.

Tener estas premisas claras nos ayudará a ver el proceso de forma completa, dar forma a las fases posteriores con un objetivo claro, a la vez que reducimos la revisión del código durante el desarrollo.

Generación de plantilla

El siguiente paso a realizar es el de definición de la plantilla de carga para poder ingestar la información en la BBDD IO.

El formato de archivo es *fmt*, el cual contiene los metadatos que describen la estructura de datos en el archivo de origen y cómo deben ser insertados en la tabla de destino. Este formato se ingesta mediante la herramienta BCP (para más información ir al apartado *Ingesta de archivos en BBDD* 5.2.2 de la documentación).

Tomamos como referencia el ejemplo del *Código B.1.*

Código B.1: Ejemplo de plantilla en formato fmt

```
12.0
3
1   SQLCHAR   0   0   ";"   1  Columna_1       SQL_Latin1_General_CP1_CI_AS
2   SQLCHAR   0   0   ";"   2  Columna_2       SQL_Latin1_General_CP1_CI_AS
3   SQLCHAR   0   0   "\n"  3  Columna_3       SQL_Latin1_General_CP1_CI_AS
```

El cual se define de la siguiente manera:

1. La primera línea indica la versión de compatibilidad con archivos del formato.
   a) La versión 12.0 generada por la utilidad BCP es la más estable.
   b) Esta versión de formato es válida para versiones posteriores como la nuestra SQL Server 2022 (16.x).
2. La segunda línea especifica el número total de columnas en el archivo.
3. La tercera línea y sucesivas contienen la definición de columnas. siendo el formato:
   a) Posición de la columna en el fichero.
   b) Tipo de datos de caracteres.
   c) Longitud del campo (0 significa longitud variable).

d) El carácter utilizado para separar los valores en el archivo.

e) Número de columna.

f) Nombre de columna.

g) Collation utilizado para ordenamiento y comparación de caracteres.

Se puede apreciar como la última definición de columnas, con el mismo formato que las anteriores, contiene el separador $\backslash n$ para indicar el final de la fila. Es muy importante que la plantilla contenga un salto de línea al final del fichero, tras la definición de la última columna, de lo contrario se pueden producir errores durante el procesamiento del fichero.

Una vez ejecutemos nuestro proceso, si todo a ido bien se generará una tabla con el contenido de la plantilla en la BBDD IO.

Creación de proceso maestro y fases de detalle

Continuamos generando tanto el registro maestro del proceso en la tabla *IO.config.master_process_type*, como los registros de detalle o fases en la tabla *IO.config.detail_process_type*.

A modo de soporte, se indica la plantilla de *Código B.2* con el ejemplo de *INSERT* para cada una de las tablas.

Código B.2: Plantilla insert config.master_process_type

```
USE [IO]
GO

INSERT INTO [config].[master_process_type]
    ([id]
    ,[customer]
    ,[name]
    ,[description]
    ,[active_flag]
    ,[ingestion_flag]
    ,[transformation_flag]
    ,[persist_flag]
    ,[output_flag]
    ,[data_creation_date]
    ,[data_modification_date]
    ,[last_action_user]
    ,[stop_on_warning])
VALUES
    (<id, int,>
    ,<customer, varchar(128),>
    ,<name, varchar(128),>
    ,<description, varchar(1024),>
    ,<active_flag, bit,>
    ,<ingestion_flag, bit,>
    ,<transformation_flag, bit,>
    ,<persist_flag, bit,>
    ,<output_flag, bit,>
    ,<data_creation_date, datetime,>
    ,<data_modification_date, datetime,>
    ,<last_action_user, varchar(1024),>
    ,<stop_on_warning, bit,>)
GO
```

Es importante dar un nombre y una descripción detallada del proceso que vamos a dar de alta,

esto aportará claridad y nos facilitará la búsqueda en un futuro. En el caso de que el proceso sea para un cliente nuevo, tendremos que generar un registro en la tabla *IO.config.customer_cluster_type* mediante el siguiente *Código B.3*.

Código B.3: Plantilla insert config.customer_cluster_type

```
USE [IO]
GO

INSERT INTO [config].[customer_cluster_type]
          ([customer_id]
          ,[name]
          ,[cluster_id]
          ,[cluster_name]
          ,[data_creation_date]
          ,[data_modification_date]
          ,[last_action_user])
     VALUES
          (<customer_id, int,>
          ,<name, varchar(64),>
          ,<cluster_id, int,>
          ,<cluster_name, varchar(128),>
          ,<data_creation_date, datetime,>
          ,<data_modification_date, datetime,>
          ,<last_action_user, varchar(1024),>)
GO
```

A su vez, se definirá adicionalmente en qué BBDD se escribirá por defecto para ese cliente. Para ello, adicionalmente insertamos otro registro en la tabla *IO.config.config.customer_persistence_destination* como indica el *Código B.4*.

Código B.4: Plantilla insert config.customer_persistence_destination

```
USE [IO]
GO

INSERT INTO [config].[customer_persistence_destination]
          ([customer_id]
          ,[destination_db]
          ,[data_creation_date]
          ,[data_modification_date]
          ,[last_action_user])
     VALUES
          (<customer_id, int,>
          ,<destination_db, varchar(64),>
          ,<data_creation_date, datetime,>
          ,<data_modification_date, datetime,>
          ,<last_action_user, varchar(1024),>)
GO
```

El nombre e identificador para el cliente deberá guardar relación entre tablas (columnas *customer_id* y *name* respectivamente).

Se aconseja dejar un margen entre los valores de la columna *id* para registros de un mismo proceso en *IO.config.detail_process_type*, de tal forma que si durante el desarrollo fuese necesario incluir una fase adicional entre medias de otras dos, los registros sigan manteniendo el orden correcto. El *Código B.5* muestra un ejemplo de inserción.

Código B.5: Plantilla insert config.detail_process_type

```
USE [IO]
GO

INSERT INTO [config].[detail_process_type]
    ([id]
```

```
    ,[master_process_type_id]
    ,[phase]
    ,[step]
    ,[entity]
    ,[name]
    ,[description]
    ,[source_type]
    ,[zip_folder_path]
    ,[zip_file_path]
    ,[source_file_folder_path]
    ,[source_file_pattern]
    ,[add_eof]
    ,[skip_final_rows]
    ,[skip_initial_rows]
    ,[source_file_template_path]
    ,[source_table_query]
    ,[destination_table]
    ,[destination_format]
    ,[generate_empty_file]
    ,[destination_file_skip_rows]
    ,[destination_path]
    ,[destination_file_template_path]
    ,[ftp_server]
    ,[ftp_port]
    ,[ftp_user]
    ,[ftp_pass]
    ,[ftp_private_key_path]
    ,[ftp_ssh_host_key_fingerprint]
    ,[ftp_private_key_passphrase]
    ,[ftp_operation]
    ,[ftp_folder]
    ,[ftp_file_pattern]
    ,[ftp_local_folder]
    ,[ftp_local_file_pattern]
    ,[email_from]
    ,[email_to]
    ,[email_subject]
    ,[email_body]
    ,[email_has_attachment]
    ,[email_attachment_path]
    ,[email_attachment_pattern]
    ,[data_creation_date]
    ,[data_modification_date]
    ,[last_action_user]
    ,[file_action]
    ,[file_action_source_pattern]
    ,[file_action_destination_folder]
    ,[check_ingestion_filename]
    ,[source_connection_server]
    ,[source_connection_database]
    ,[source_connection_user]
    ,[source_connection_password]
    ,[destination_connection_server]
    ,[destination_connection_database]
    ,[destination_connection_user]
    ,[destination_connection_password]
    ,[ingestion_encoding_file]
    ,[zip_rar_pass])
VALUES
    (<id, int,>
    ,<master_process_type_id, int,>
    ,<phase, varchar(16),>
    ,<step, int,>
    ,<entity, varchar(512),>
    ,<name, varchar(128),>
    ,<description, varchar(1024),>
    ,<source_type, varchar(5),>
    ,<zip_folder_path, varchar(1024),>
    ,<zip_file_path, varchar(1024),>
    ,<source_file_folder_path, varchar(1024),>
    ,<source_file_pattern, varchar(128),>
    ,<add_eof, int,>
    ,<skip_final_rows, int,>
```

```
    ,<skip_initial_rows, int,>
    ,<source_file_template_path, varchar(1024),>
    ,<source_table_query, varchar(max),>
    ,<destination_table, varchar(1024),>
    ,<destination_format, varchar(1024),>
    ,<generate_empty_file, bit,>
    ,<destination_file_skip_rows, int,>
    ,<destination_path, varchar(1024),>
    ,<destination_file_template_path, varchar(1024),>
    ,<ftp_server, varchar(128),>
    ,<ftp_port, int,>
    ,<ftp_user, varchar(128),>
    ,<ftp_pass, varbinary(256),>
    ,<ftp_private_key_path, varchar(1024),>
    ,<ftp_ssh_host_key_fingerprint, varchar(128),>
    ,<ftp_private_key_passphrase, varbinary(256),>
    ,<ftp_operation, varchar(8),>
    ,<ftp_folder, varchar(1024),>
    ,<ftp_file_pattern, varchar(1024),>
    ,<ftp_local_folder, varchar(1024),>
    ,<ftp_local_file_pattern, varchar(1024),>
    ,<email_from, varchar(256),>
    ,<email_to, varchar(1024),>
    ,<email_subject, varchar(1024),>
    ,<email_body, varchar(max),>
    ,<email_has_attachment, int,>
    ,<email_attachment_path, varchar(1024),>
    ,<email_attachment_pattern, varchar(1024),>
    ,<data_creation_date, datetime,>
    ,<data_modification_date, datetime,>
    ,<last_action_user, varchar(1024),>
    ,<file_action, varchar(6),>
    ,<file_action_source_pattern, varchar(1024),>
    ,<file_action_destination_folder, varchar(1024),>
    ,<check_ingestion_filename, int,>
    ,<source_connection_server, varchar(64),>
    ,<source_connection_database, varchar(64),>
    ,<source_connection_user, varchar(64),>
    ,<source_connection_password, varbinary(256),>
    ,<destination_connection_server, varchar(64),>
    ,<destination_connection_database, varchar(64),>
    ,<destination_connection_user, varchar(64),>
    ,<destination_connection_password, varbinary(256),>
    ,<ingestion_encoding_file, varchar(5),>
    ,<zip_rar_pass, varbinary(256),>)
GO
```

Para conocer el detalle específico del contenido de las columnas, así como de la información a insertar en función de la casuística, ver el apartado *Diseño de BBDD y modelo entidad relación IO* 4.4 del capítulo 4.

Procedimientos y tablas de carga

Una vez hayamos insertado los registros en las tablas de configuración, solo falta crear los procedimientos almacenados para las diferentes fases del proceso, a menos que hayamos embebido la consulta en la columna *source_table_query* de la tabla *IO.config.detail_process_type* (en cuyo caso la solución leerá directamente el código a ejecutar de la propia columna).

En el caso de un proceso de tipo input, habitualmente necesitaremos la generación de una nueva delivery, uno o varios procesos de transformación, y otros tantos para el caso de la persistencia.

Se adjunta un ejemplo de plantilla (*Códigos B.6-B.7*) para cada uno de los casos, a modo de soporte en el desarrollo de un nuevo proceso para el usuario.

Código B.6: Ejemplo de procedimiento de nueva delivery

```
USE [IO];
GO
/*
-- ============================================================================================
Autor:
Fecha creacion:
Descripcion:
Cambios:

Entrada:
   @master_id: Id configuración maestra del proceso.
   @master_log_id: Id log de la ejecución en curso.

Salida:

Ejemplo:
   EXEC [IO].[input].[sp_customer_new_delivery] @master_id = 1, @master_log_id = 1

   SELECT * FROM [IO].[load].delivery ORDER BY id DESC
-- ============================================================================================
*/
CREATE OR ALTER PROCEDURE [input].[sp_customer_new_delivery]
   @master_id INT,
   @master_log_id INT
AS
BEGIN
   -- Variables de logging en caso de error
   DECLARE @msg VARCHAR(1024)
   DECLARE @error_values VARCHAR(1024)

   -- Eliminación de la tabla temporal en caso de que exista
   DROP TABLE IF EXISTS #incorporation_temp

   -- Se calcula el/los cluster_id basándose en los names que contiene la tabla del archivo ingestado
   SELECT DISTINCT COALESCE(cct.cluster_id, 0) AS customer_id
   INTO #incorporation_temp
   FROM [IO].config.customer_cluster_type cct
   WHERE cluster_name = 'customer'

   -- Se verifica la existencia de una delivery abierta
   IF EXISTS (
      SELECT * FROM #incorporation_temp tt
      INNER JOIN [IO].[load].delivery dv ON dv.customer_id = tt.customer_id
      INNER JOIN [IO].monitor.master_process_log mpl ON dv.master_process_log_id = mpl.id
      INNER JOIN [IO].config.master_process_type mpt ON mpl.master_process_type_id = mpt.id
      WHERE mpt.id = @master_id AND dv.registration_date IS NULL
   )
      -- En caso afirmativo, se genera el error adecuado sobre el master process id asociado
      BEGIN
         SET @msg = 'There is some pending delivery associated to main process ' + CONVERT(VARCHAR(10),
              @master_id)
         EXEC [IO].[monitor].[sp_write_log] @ProcId = @@PROCID, @TraceId = '', @Message = @msg , @Severity =
              'ERROR'
      END
   ELSE
      -- En caso negativo, se generan las deliveries necesarias según los clientes/clusters
      BEGIN
         INSERT INTO [IO].[load].delivery (customer_id, master_process_log_id)
         SELECT customer_id, @master_log_id FROM #incorporation_temp
      END
END
```

En el caso de la plantilla de delivery, será suficiente modificar el valor de la columna *cluster_name*, por el del cliente para el que vayamos a realizar el proceso.

Código B.7: Ejemplo de procedimiento de transformación

```
USE [IO];
GO
/*
-- ============================================================================================
Autor:
Fecha creacion:
Descripcion:
Cambios:

Entrada:
   @master_process_id: Id configuración maestra del proceso.

Salida:

Ejemplo:
   EXEC [IO].[input].[sp_customer_transformation_insert_tabla1] @master_process_id = 1
-- ============================================================================================
*/
CREATE OR ALTER PROCEDURE [input].[sp_customer_transformation_insert_tabla1]
   @master_process_id INT
AS
BEGIN
   -- Se insertan en la tabla del schema [load] los registros provenientes del archivo
   -- que no existan ya en la tabla
   INSERT INTO [IO].[load].[tabla1]
      ([delivery_id], [mdb_id], [nombre1], [nombre2], [nombre3], [nombre4])
   SELECT
      delivery_id
      ,mdb_id
      ,[columna1]
      ,[columna2]
      ,[columna3]
      ,[columna4]
   FROM (
      -- Se obtienen todos los datos de los concesionarios nuevos
      SELECT
         d.id AS delivery_id,
         NULL AS mdb_id,
         f.columna1,
         f.columna2,
         f.columna3,
         f.columna4,
      FROM [IO].[input].[ingesta_fichero] f
         -- Revisamos mediante este cruce los registros que puedan existir en la tabla destino
         LEFT JOIN [bbdd].dbo.tabla1 t1 ON [auxiliar].[f_clean_rare_characters](f.[columna1]) =
              [auxiliar].[f_clean_rare_characters](t1.[nombre1])
         INNER JOIN [IO].config.customer_cluster_type cct ON 1 = cct.customer_id
         INNER JOIN [IO].[load].delivery d ON d.customer_id = cct.cluster_id
         INNER JOIN [IO].monitor.master_process_log mpl ON d.master_process_log_id = mpl.id
      WHERE
         -- Omitimos los registros que existan en la tabla destino
         t1.id IS NULL
         -- Pertenece al proceso maestro adecuado
         AND mpl.master_process_type_id = @master_process_id
         -- Tiene delivery abierta
         AND d.registration_date IS NULL
   ) sub

   -- Se insertan todas las operaciones INSERT de los registros calculados anteriormente
   INSERT INTO [IO].[load].operation
      ([delivery_id], [load_table_name], [load_reference_id], [master_reference_id], [operation_type_id])
   SELECT DISTINCT
      d.id AS delivery_id,
      'tabla1' AS load_table_name,
      c.id AS load_reference_id,
      c.mdb_id AS master_reference_id,
      (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Insert') AS operation_type_id
   FROM [IO].[load].[concessionaire] c
      INNER JOIN [IO].[load].delivery d ON c.delivery_id= d.id
      INNER JOIN [IO].monitor.master_process_log mpl ON d.master_process_log_id = mpl.id
   WHERE
      -- Son concesionarios nuevos
```

```
72      c.mdb_id IS NULL
73      -- Pertenece al proceso maestro adecuado
74      AND mpl.master_process_type_id = @master_process_id
75      -- Tiene delivery abierta
76      AND d.registration_date IS NULL
77
78   -- Se devuelve el número de operaciones almacenadas
79   SELECT @@ROWCOUNT AS operations_quantity
80
81 END
```

Durante la fase de transformación nos interesa realizar el mayor número de operaciones posibles, tales como filtrar, agrupar, ordenar la información, consiguiendo de esta manera encapsular las reglas en este apartado, simplificando al mismo tiempo la etapa de persistencia.

Adicionalmente, como se puede apreciar en la plantilla del procedimiento (*Código B.7*), se usa una tabla de carga (schema *load*) que almacena la información en un formato muy similar al del destino, facilitando de esta forma la inserción en la tabla final en el procedimiento de persistencia, para ello podemos usar directamente el script de creación de la tabla final, con la salvedad de que tendremos que incluir las columnas *delivery_id* y *mdb_id* para poder almacenar la traza de la ejecución específica del proceso.

Se indica en el *Código B.8* un posible ejemplo de tabla.

Código B.8: Ejemplo de tabla schema load

```
1  CREATE TABLE [IO].[load].[tabla1](
2     id INT IDENTITY(1,1) NOT NULL,
3     delivery_id INT NOT NULL,
4     mdb_id INT NULL,
5     [nombre1] NVARCHAR(128)
6     ,[nombre2] INT
7     ,[nombre3] DECIMAL(10,2)
8     ,[nombre4] DATE
9  PRIMARY KEY CLUSTERED
10 (
11    [id] ASC
12 )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON,
       ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
13 ) ON [PRIMARY]
14 GO
15
16 ALTER TABLE [IO].[load].[tabla1] WITH CHECK ADD CONSTRAINT [fk_tabla1_delivery_id] FOREIGN KEY([delivery_id])
17 REFERENCES [load].[delivery] ([id])
18 GO
19
20 ALTER TABLE [IO].[load].[tabla1] CHECK CONSTRAINT [fk_tabla1_delivery_id]
21 GO
```

Finalizamos el proceso de tipo input mostrando la plantilla de un proceso de persistencia en el *Código B.9*.

Código B.9: Ejemplo de procedimiento de persistencia

```
USE [IO];
GO
/*
-- ============================================================================================
Autor:
Fecha creacion:
Descripcion:
Cambios:

Entrada:
   @delivery_id: Id de la delivery abierta en curso.

Salida:
   @affected_rows: Número de filas afectadas por el proceso.

Ejemplo:
   EXEC [IO].[input].[sp_customer_persistence_insert_tabla1] @delivery_id = 1, @affected_rows = 0
-- ============================================================================================
*/
CREATE OR ALTER PROCEDURE [input].[sp_customer_persistence_insert_tabla1]
   @delivery_id INT,
   @affected_rows INT OUTPUT
AS
BEGIN

   -- Se realizan todas las operaciones de INSERT calculadas en el proceso de transformación
   INSERT INTO [bbdd].dbo.tabla1
      ([nombre1], [nombre2], [nombre3], [nombre4], [io_id])
   SELECT DISTINCT
      t.nombre1,
      t.nombre2,
      t.nombre3,
      t.nombre4
      t.id AS io_id
   FROM [IO].[load].[tabla1] t
      INNER JOIN [IO].[load].operation o ON o.delivery_id = t.delivery_id
         AND o.load_reference_id = t.id
   WHERE
      -- Tabla correcta
      o.load_table_name = 'tabla1'
      -- Operación de tipo Insert
      AND o.operation_type_id = (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Insert')
      -- Operación sin persistir
      AND o.operation_finished = 0
      -- Para la delivery en curso
      AND o.delivery_id = @delivery_id

   -- Se finalizan (operation_finished = 1) las operaciones realizadas en el paso anterior
   UPDATE o SET operation_finished = 1
   FROM [IO].[load].tabla1 t
      INNER JOIN [IO].[load].operation o ON o.delivery_id = t.delivery_id
         AND o.load_reference_id = t.id
   WHERE
      -- Tabla correcta
      o.load_table_name = 'tabla1'
      -- Operación de tipo Insert
      AND o.operation_type_id = (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Insert')
      -- Operación sin persistir
      AND o.operation_finished = 0
      -- Para la delivery en curso
      AND o.delivery_id = @delivery_id

   -- Se devuelve el número de operaciones actualizadas
   SELECT @affected_rows = @@ROWCOUNT

END
```

Por último, para un proceso de tipo output podemos crear un procedimiento que genere el informe deseado, o insertar la consulta directamente en la columna *source_table_query* de la

tabla *IO.config.detail_process_type* como hemos indicado previamente para el tipo input.

Si optamos por crear un procedimiento almacenado, se puede seguir la plantilla de *Código B.10* la cual contiene una serie de buenas prácticas para este formato.

Código B.10: Ejemplo de procedimiento para generar un informe

```
USE [IO];
GO
/*
-- ============================================================================================
Autor:
Fecha creacion:
Descripcion:
Cambios:

Entrada:

Salida:
   - Conjunto de filas con los resultados.

Ejemplo:
   EXEC [IO].[output].[sp_customer_report]
-- ============================================================================================
*/
CREATE OR ALTER PROCEDURE [output].[sp_customer_report]
AS
BEGIN

   -- Creacion tabla temporal
   DECLARE @report_table TABLE (
      [columna1] NVARCHAR(128)
      ,[columna2] INT
      ,[columna3] DECIMAL(10,2)
      ,[columna4] DATE
   )

   -- Generación del report e inserción en tabla temporal
   INSERT INTO @report_table
   SELECT
      t1.[nombre1] AS [columna1]
      ,t1.[nombre2] AS [columna2]
      ,t1.[nombre3] AS [columna3]
      ,t1.[nombre4] AS [columna4]
   FROM [bbdd].dbo.tabla1 t1
      -- Descomentar si es necesario realizar un cruce
      --INNER JOIN [bbdd].dbo.tabla2 t2 ON t1.id = t2.tabla1_id
   -- Descomentar si es necesario filtrar la consulta
   --WHERE
   -- Descomentar si es necesario agrupar el resultado
   --GROUP BY

   -- Borramos tabla ultima ejecución
   DROP TABLE IF EXISTS [output].customer_report

   -- Formato final y creación de tabla final
   SELECT
      -- Descomentar si el informe contiene una clausula TOP
      -- TOP 10 CONVERT(INT, ROW_NUMBER() OVER(ORDER BY [columna1] DESC)) AS [top],
      *
      ,GETDATE() AS creation_date
   INTO [output].customer_report
   FROM @report_table

   -- Salida desde tabla final con cabeceras
   SELECT
      'columna1','columna2','columna3','columna4'
   UNION ALL
   SELECT
      ,CONVERT(NVARCHAR(16), [columna1])
      ,CONVERT(NVARCHAR(16), [columna2])
      ,CONVERT(NVARCHAR(16), [columna3])
```

```
66      ,CONVERT(NVARCHAR(16), [columna4])
67    FROM [output].customer_report
68
69  END
```

Ejecución de proceso y revisión de logs

Tras finalizar la generación completa del código necesario para el nuevo proceso, tendremos que ejecutarlo a través del paquete *Main_process.dtsx* de nuestra solución SSIS, para ello, insertamos el valor del identificador de nuestro proceso maestro en *master_process_id* del apartado *Parámetros*, tal y como se aprecia en la *Figura B.1*.

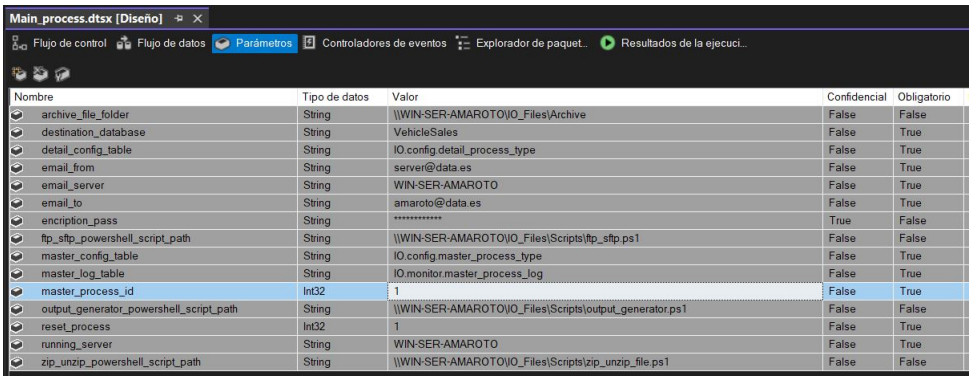


Figura B.1: Asignación de valor al parámetro master_process_id

Tras ello, iniciamos el proceso. Este irá pasando por las fases que le hayamos indicado en los registros de la tabla *IO.config.master_process_type*.

En las *Figuras B.2-B.5*, se muestra un ejemplo de ejecución a través de SSIS para el *Caso de uso 1: Input - NewDealers csv* 6.3 del capítulo 6.

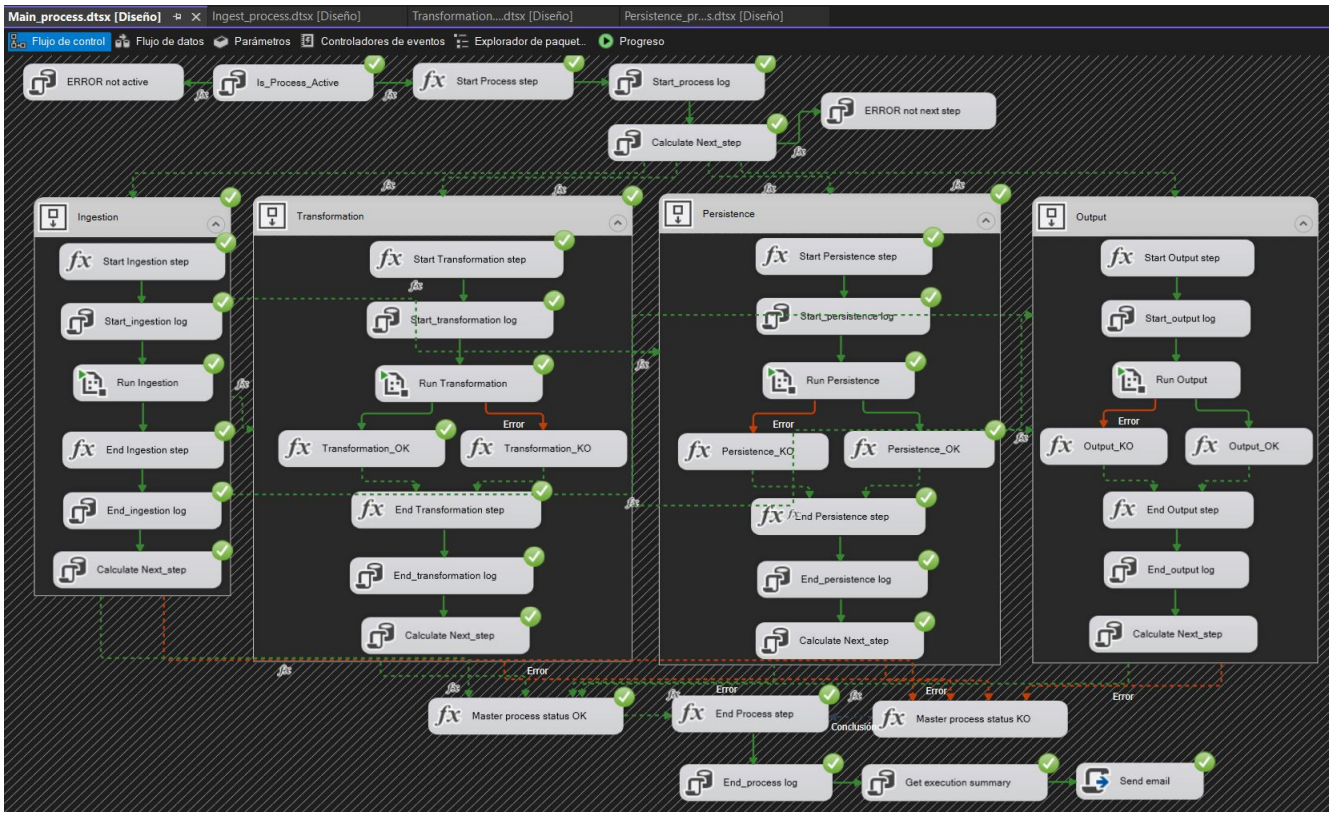


Figura B.2: Ejemplo de ejecución del paquete Main_process.dtsx

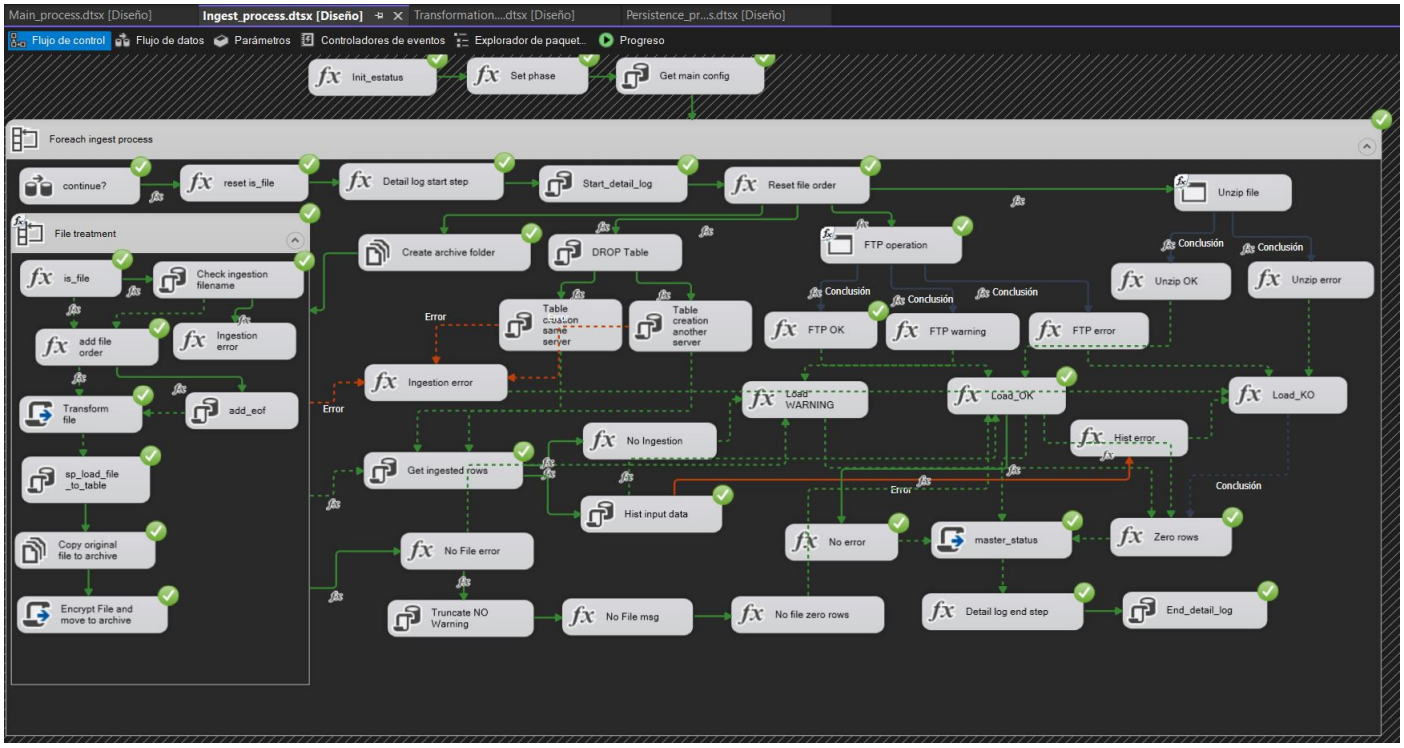


Figura B.3: Ejemplo de ejecución del paquete Ingest_process.dtsx

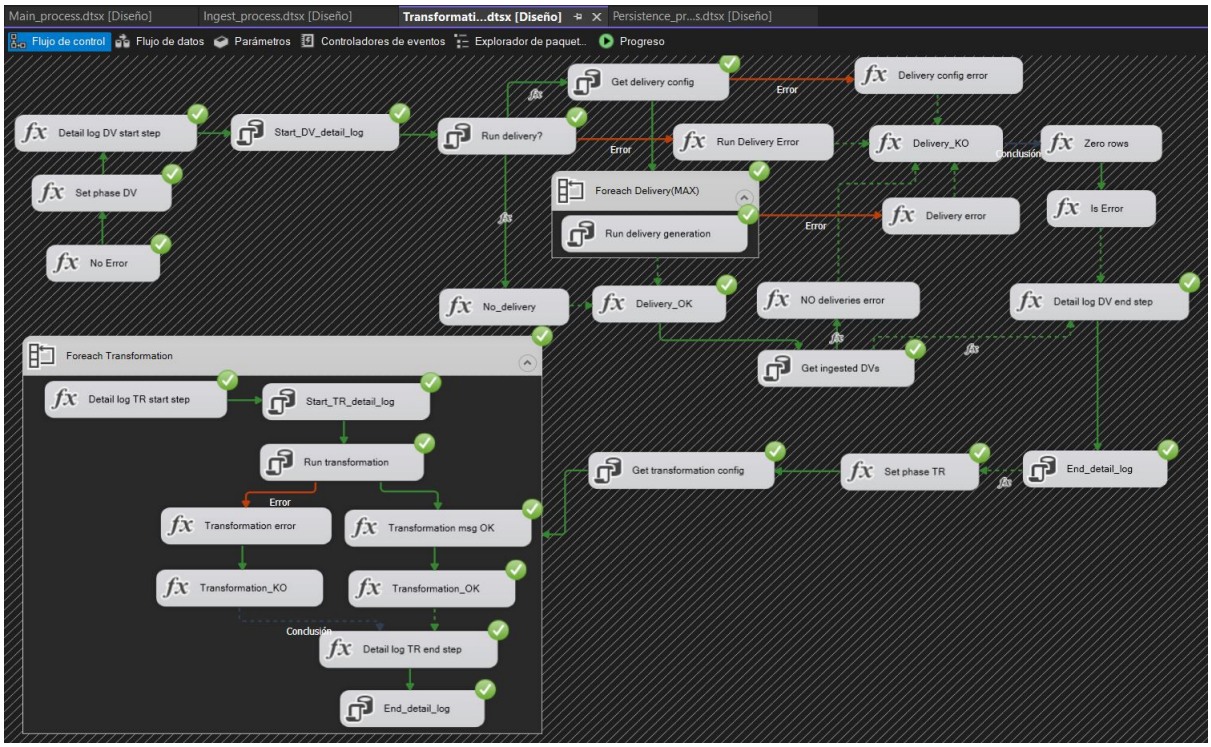


Figura B.4: Ejemplo de ejecución del paquete Transformation_process.dtsx

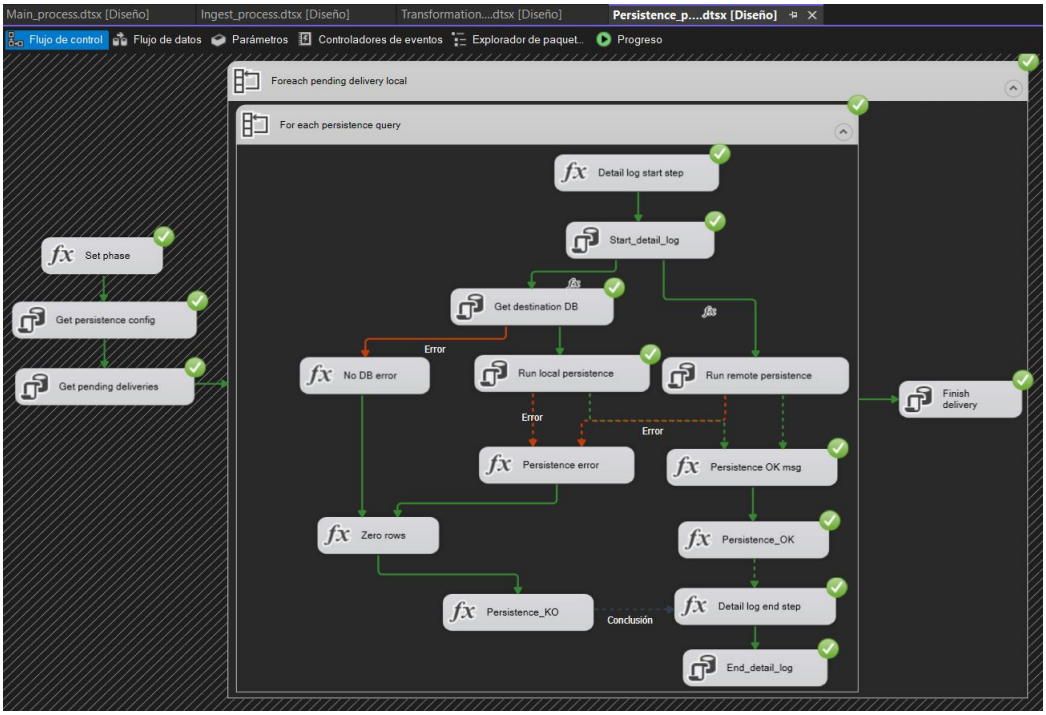
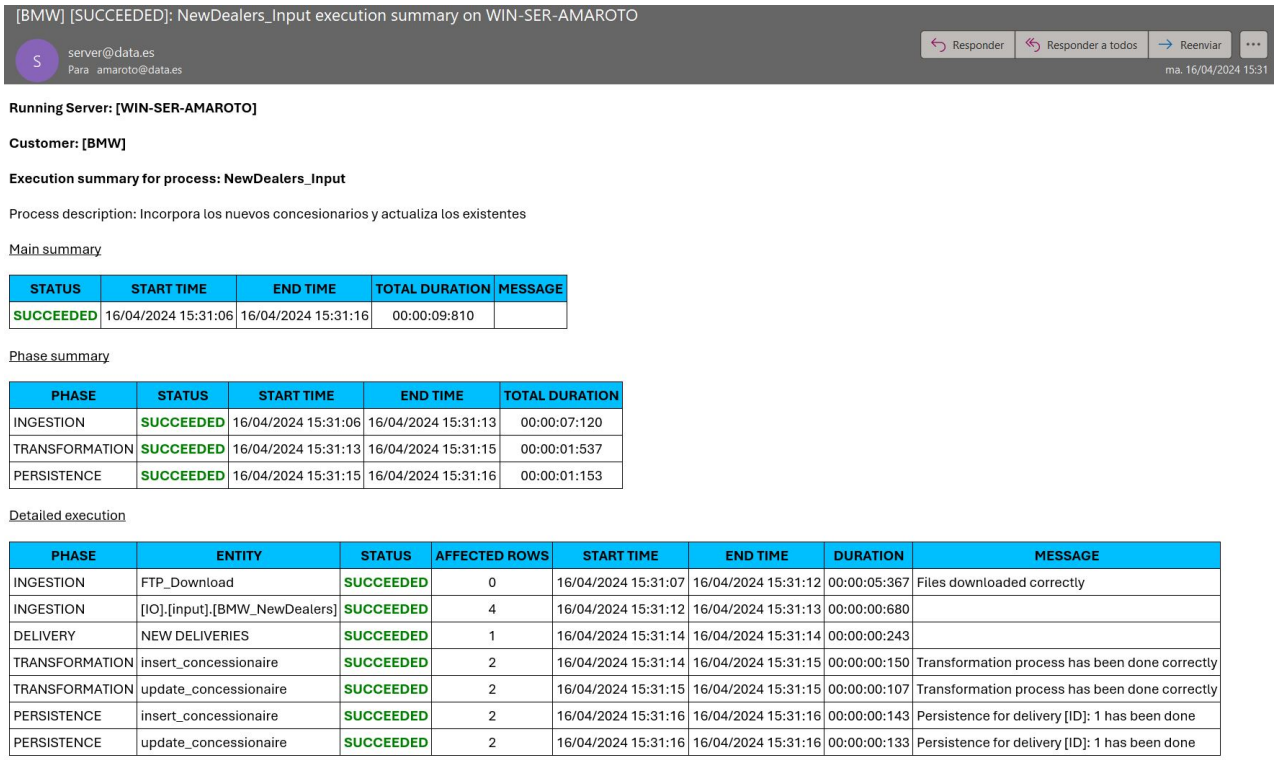


Figura B.5: Ejemplo de ejecución del paquete Persistence_process.dtsx

Si el proceso se ha ejecutado de forma correcta, se recibirá un correo como el que se muestra en la *Figura B.6*.



[BMW] [SUCCEEDED]: NewDealers_Input execution summary on WIN-SER-AMAROTO

server@data.es
Para amaroto@data.es

Responder | Responder a todos | Reenviar | ...
ma. 16/04/2024 15:31

Running Server: [WIN-SER-AMAROTO]

Customer: [BMW]

Execution summary for process: NewDealers_Input

Process description: Incorpora los nuevos concesionarios y actualiza los existentes

Main summary

STATUS	START TIME	END TIME	TOTAL DURATION	MESSAGE
SUCCEEDED	16/04/2024 15:31:06	16/04/2024 15:31:16	00:00:09:810	

Phase summary

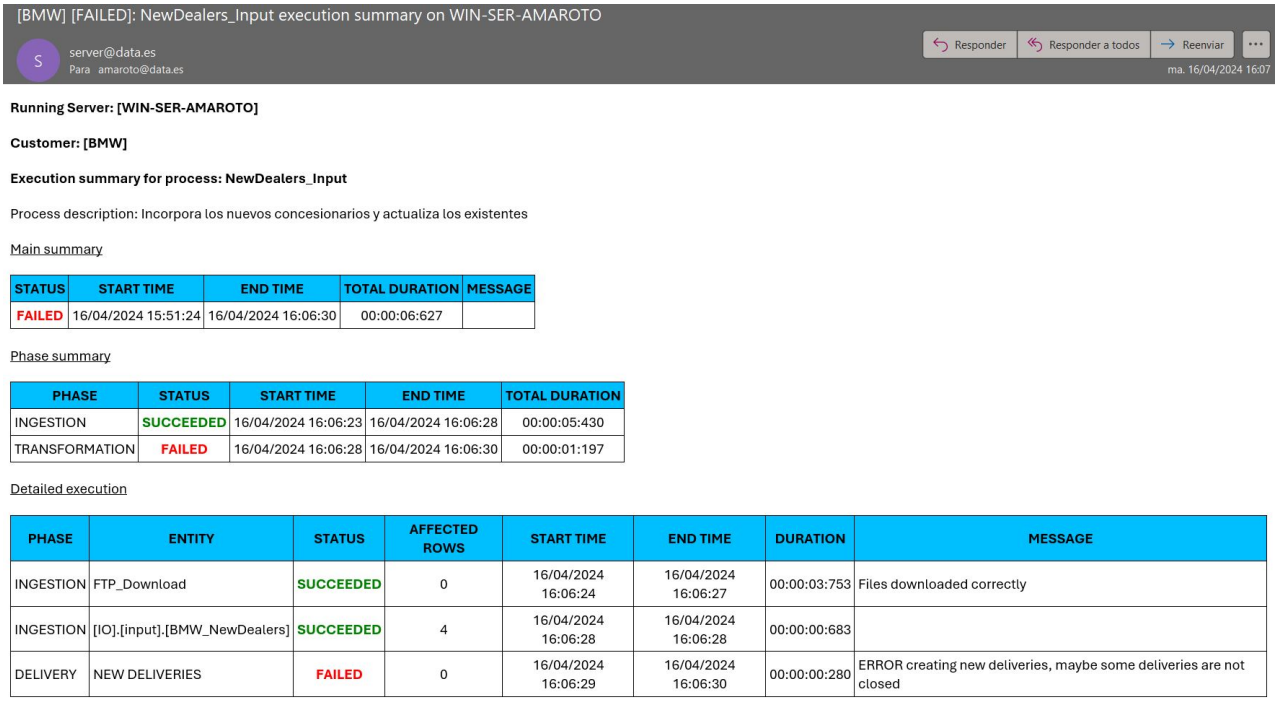
PHASE	STATUS	START TIME	END TIME	TOTAL DURATION
INGESTION	SUCCEEDED	16/04/2024 15:31:06	16/04/2024 15:31:13	00:00:07:120
TRANSFORMATION	SUCCEEDED	16/04/2024 15:31:13	16/04/2024 15:31:15	00:00:01:537
PERSISTENCE	SUCCEEDED	16/04/2024 15:31:15	16/04/2024 15:31:16	00:00:01:153

Detailed execution

PHASE	ENTITY	STATUS	AFFECTED ROWS	START TIME	END TIME	DURATION	MESSAGE
INGESTION	FTP_Download	SUCCEEDED	0	16/04/2024 15:31:07	16/04/2024 15:31:12	00:00:05:367	Files downloaded correctly
INGESTION	[IO].[input].[BMW_NewDealers]	SUCCEEDED	4	16/04/2024 15:31:12	16/04/2024 15:31:13	00:00:00:680	
DELIVERY	NEW DELIVERIES	SUCCEEDED	1	16/04/2024 15:31:14	16/04/2024 15:31:14	00:00:00:243	
TRANSFORMATION	insert_concessionaire	SUCCEEDED	2	16/04/2024 15:31:14	16/04/2024 15:31:15	00:00:00:150	Transformation process has been done correctly
TRANSFORMATION	update_concessionaire	SUCCEEDED	2	16/04/2024 15:31:15	16/04/2024 15:31:15	00:00:00:107	Transformation process has been done correctly
PERSISTENCE	insert_concessionaire	SUCCEEDED	2	16/04/2024 15:31:16	16/04/2024 15:31:16	00:00:00:143	Persistence for delivery [ID]: 1 has been done
PERSISTENCE	update_concessionaire	SUCCEEDED	2	16/04/2024 15:31:16	16/04/2024 15:31:16	00:00:00:133	Persistence for delivery [ID]: 1 has been done

Figura B.6: Ejemplo de correo recibido tras la ejecución de un proceso

Es especialmente relevante prestar atención a un posible error de delivery abierta para el proceso que estemos intentando ejecutar, en este caso el proceso fallará y nos enviará un correo como el de la *Figura B.7*.



[BMW] [FAILED]: NewDealers_Input execution summary on WIN-SER-AMAROTO

server@data.es
Para amaroto@data.es

Responder | Responder a todos | Reenviar | ...
ma. 16/04/2024 16:07

Running Server: [WIN-SER-AMAROTO]

Customer: [BMW]

Execution summary for process: NewDealers_Input

Process description: Incorpora los nuevos concesionarios y actualiza los existentes

Main summary

STATUS	START TIME	END TIME	TOTAL DURATION	MESSAGE
FAILED	16/04/2024 15:51:24	16/04/2024 16:06:30	00:00:06:627	

Phase summary

PHASE	STATUS	START TIME	END TIME	TOTAL DURATION
INGESTION	SUCCEEDED	16/04/2024 16:06:23	16/04/2024 16:06:28	00:00:05:430
TRANSFORMATION	FAILED	16/04/2024 16:06:28	16/04/2024 16:06:30	00:00:01:197

Detailed execution

PHASE	ENTITY	STATUS	AFFECTED ROWS	START TIME	END TIME	DURATION	MESSAGE
INGESTION	FTP_Download	SUCCEEDED	0	16/04/2024 16:06:24	16/04/2024 16:06:27	00:00:03:753	Files downloaded correctly
INGESTION	[IO].[input].[BMW_NewDealers]	SUCCEEDED	4	16/04/2024 16:06:28	16/04/2024 16:06:28	00:00:00:683	
DELIVERY	NEW DELIVERIES	FAILED	0	16/04/2024 16:06:29	16/04/2024 16:06:30	00:00:00:280	ERROR creating new deliveries, maybe some deliveries are not closed

Figura B.7: Asignación de valor al parámetro master_process_id

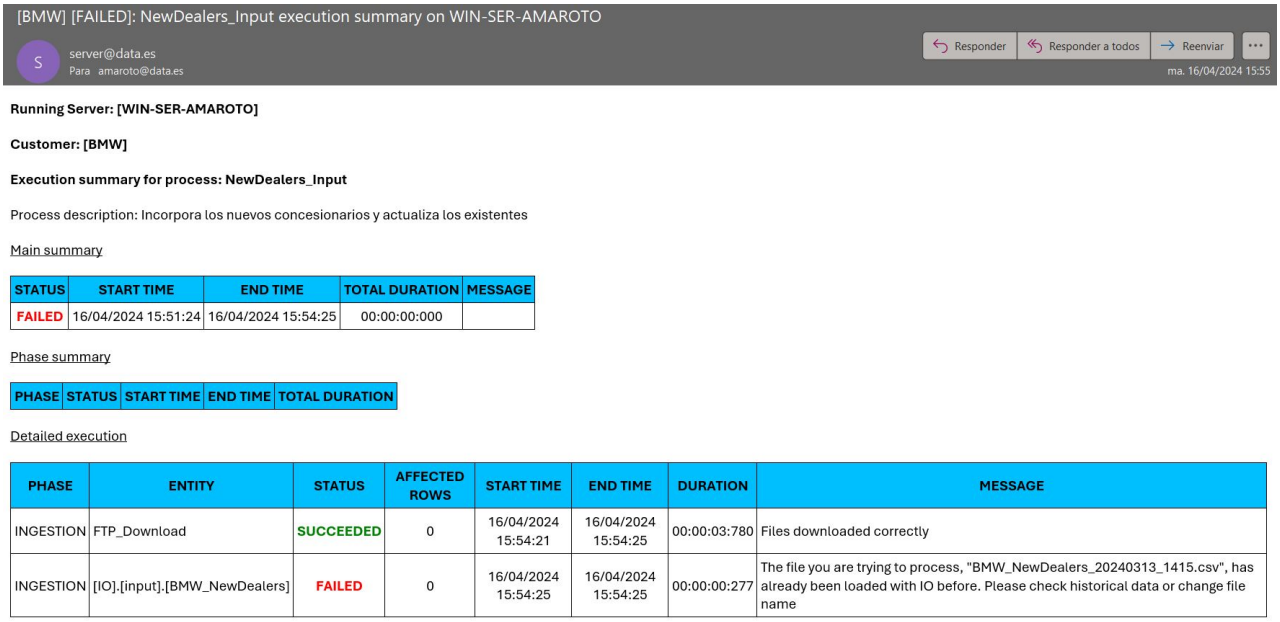
El tener una delivery abierta nos indica que alguna de las ejecuciones anteriores no ha finalizado correctamente, necesitando revisar dicha ejecución, si esta operación no se realiza con cuidado

podríamos insertar registros duplicados en el destino. Una vez analizada la delivery, podemos cerrarla con el *Código B.11* y volver a cargar de nuevo nuestro fichero.

Código B.11: Proveedores del servidor

```
1       UPDATE IO.load.delivery SET registration_date = GETDATE() WHERE registration_date IS NULL AND
            master_process_log_id = 1
```

Por otro lado, si el fichero que estamos intentando procesar para nuestro proceso se ha cargado anteriormente, al tratar de ejecutarlo de nuevo nos devolverá un error por correo tal y como se muestra en la siguiente figura.



[BMW] [FAILED]: NewDealers_Input execution summary on WIN-SER-AMAROTO

server@data.es
Para amaroto@data.es

Responder | Responder a todos | Reenviar | ...

ma. 16/04/2024 15:55

Running Server: [WIN-SER-AMAROTO]

Customer: [BMW]

Execution summary for process: NewDealers_Input

Process description: Incorpora los nuevos concesionarios y actualiza los existentes

Main summary

STATUS	START TIME	END TIME	TOTAL DURATION	MESSAGE
FAILED	16/04/2024 15:51:24	16/04/2024 15:54:25	00:00:00:000	

Phase summary

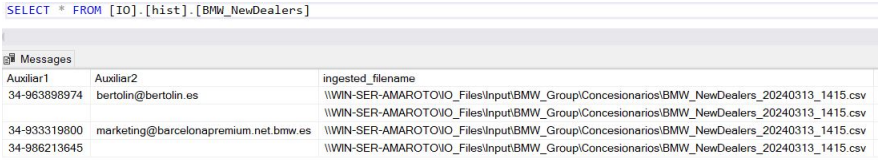
PHASE	STATUS	START TIME	END TIME	TOTAL DURATION

Detailed execution

PHASE	ENTITY	STATUS	AFFECTED ROWS	START TIME	END TIME	DURATION	MESSAGE
INGESTION	FTP_Download	SUCCEEDED	0	16/04/2024 15:54:21	16/04/2024 15:54:25	00:00:03:780	Files downloaded correctly
INGESTION	[IO].[input].[BMW_NewDealers]	FAILED	0	16/04/2024 15:54:25	16/04/2024 15:54:25	00:00:00:277	The file you are trying to process, "BMW_NewDealers_20240313_1415.csv", has already been loaded with IO before. Please check historical data or change file name

Figura B.8: Error de fichero cargado previamente

Si necesitamos volver a procesar el mismo fichero por cualquier circunstancia, tendremos que modificar su nombre, o bien eliminar la información insertada en su tabla histórica, para el ejemplo que estamos tratando la tabla es *IO.hist.BMW_NewDealers*, en la *Figura B.9* se puede apreciar a través de la columna *ingested_filename* que la información del fichero ya se ha cargado.



```
SELECT * FROM [IO].[hist].[BMW_NewDealers]
```

Messages

Auxiliar1	Auxiliar2	ingested_filename
34-963898974	bertolin@bertolin.es	\\WIN-SER-AMAROTO\IO_Files\Input\BMW_Group\Concesionarios\BMW_NewDealers_20240313_1415.csv
		\\WIN-SER-AMAROTO\IO_Files\Input\BMW_Group\Concesionarios\BMW_NewDealers_20240313_1415.csv
34-933319800	marketing@barcelonapremium.net.bmw.es	\\WIN-SER-AMAROTO\IO_Files\Input\BMW_Group\Concesionarios\BMW_NewDealers_20240313_1415.csv
34-986213645		\\WIN-SER-AMAROTO\IO_Files\Input\BMW_Group\Concesionarios\BMW_NewDealers_20240313_1415.csv

Figura B.9: Registros del fichero en la tabla histórica

Además de los correos electrónicos, disponemos de las tablas de los schemas *monitor* y *load* para obtener la traza generada durante la ejecución, en el *Código B.12* se adjuntan las consultas principales para realizar el seguimiento.

Código B.12: Proveedores del servidor

```
1       SELECT TOP 100 * FROM IO.monitor.log ORDER BY id DESC
2       SELECT TOP 100 * FROM IO.monitor.master_process_log ORDER BY id DESC
3       SELECT TOP 100 * FROM IO.monitor.detail_process_log ORDER BY id DESC
4       SELECT TOP 100 * FROM IO.load.delivery ORDER BY id DESC
5       SELECT TOP 100 * FROM IO.load.operation
```

Para automatizar la ejecución de este proceso, ir al apartado *Automatizar la ejecución mediante jobs* del *Anexo A*.

Se recomienda la creación de scripts de ejecución y seguimiento de los procesos tal y como los que se muestran en la *Estructura de scripts de creación SQL de los casos de uso del proyecto IO* C.6 del anexo C.

Anexo C

Código fuente

Introducción

En este anexo se muestra el código fuente del proyecto. El código tanto en el apartado SQL como en los dtsx de la solución SSIS está escrito en inglés para tratar de seguir una normalización en la nomenclatura de las variables, funciones, procedimientos almacenados u objetos generados. Los nombres son auto-explicativos y en la mayoría de los casos se usa la notación snake_case.

Comenzamos por el código embebido dentro de la solución SSIS IO, para luego pasar al apartado de BBDD.

Solución IO

La *Figura C.1* muestra el conjunto de scripts C# de la solución SSIS IO.

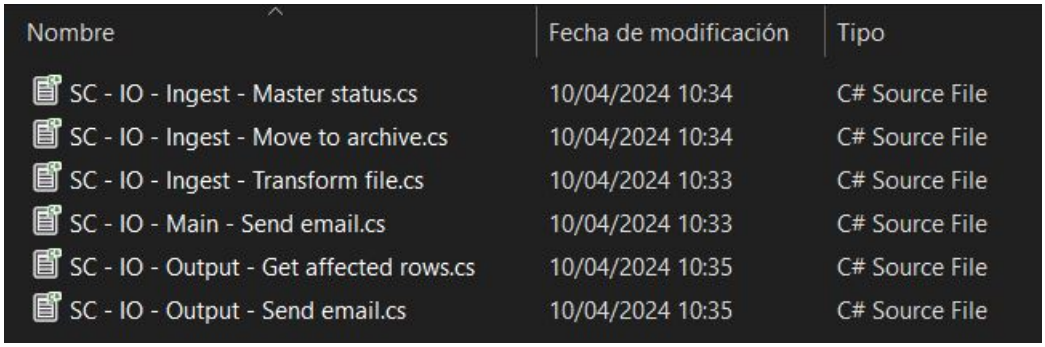


Figura C.1: Estructura de scripts C# embebidos en la solución IO

A continuación se muestra el *Código C.1* en C# del fichero *Main_process*.

Código C.1: Main - Send email.cs

```
#region Namespaces
using System;
using System.Data;
using System.Data.OleDb;
using System.Net.Mail;
using Microsoft.SqlServer.Dts.Runtime;
using System.Windows.Forms;
#endregion

namespace ST_712178489edc4ca69c38495b49170f8d
{
   [Microsoft.SqlServer.Dts.Tasks.ScriptTask.SSISScriptTaskEntryPointAttribute]
   public partial class ScriptMain : Microsoft.SqlServer.Dts.Tasks.ScriptTask.VSTARTScriptObjectModelBase
   {
      public void Main()
      {
            try
            {
                String from = (String)Dts.Variables["$Package::email_from"].Value;
                String to = (String)Dts.Variables["$Package::email_to"].Value;
                String smtp_host = (String)Dts.Variables["$Package::email_server"].Value;
                String computer_name = Environment.MachineName.ToString();
```

```
            String status = (String)Dts.Variables["User::v_status"].Value;

            DataTable table = new DataTable();
            OleDbDataAdapter adapter = new OleDbDataAdapter();
            adapter.Fill(table, Dts.Variables["User::obj_execution_summary"].Value);

            String subject = "";
            String summary_header_text = "";
            String summary_text = "";
            String detail_text = "<p> <u> Detailed execution </u> </p>" +
           "<table cellpadding=\"5\" cellspacing=\"0\"> " +
           "<tr bgcolor=\"DeepSkyBlue\"> " +
           "<th style=\"border: 1px solid black; border-collapse: collapse\"> PHASE </th> " +
           "<th style=\"border: 1px solid black; border-collapse: collapse\"> ENTITY </th> " +
           "<th style=\"border: 1px solid black; border-collapse: collapse\"> STATUS </th> " +
           "<th style=\"border: 1px solid black; border-collapse: collapse\"> AFFECTED ROWS </th> " +
           "<th style=\"border: 1px solid black; border-collapse: collapse\"> START TIME </th> " +
           "<th style=\"border: 1px solid black; border-collapse: collapse\"> END TIME </th> " +
           "<th style=\"border: 1px solid black; border-collapse: collapse\"> DURATION </th> " +
           "<th style=\"border: 1px solid black; border-collapse: collapse\" > MESSAGE </th> " +
           "</tr> ";

            foreach (DataRow row in table.Rows)
            {
                Object[] row_content = row.ItemArray;

                String customer = row_content[0].ToString();
                String process_name = row_content[1].ToString();
                String process_description = row_content[2].ToString();
                String process_status = row_content[3].ToString();
                String process_start_time = row_content[4].ToString();
                String process_end_time = row_content[5].ToString();
                String process_duration = row_content[6].ToString();
                String process_message = row_content[7].ToString();
                String ingestion_status = row_content[8].ToString();
                String ingestion_start_time = row_content[9].ToString();
                String ingestion_end_time = row_content[10].ToString();
                String ingestion_duration = row_content[11].ToString();
                String transformation_status = row_content[12].ToString();
                String transformation_start_time = row_content[13].ToString();
                String transformation_end_time = row_content[14].ToString();
                String transformation_duration = row_content[15].ToString();
                String persistence_status = row_content[16].ToString();
                String persistence_start_time = row_content[17].ToString();
                String persistence_end_time = row_content[18].ToString();
                String persistence_duration = row_content[19].ToString();
                String output_status = row_content[20].ToString();
                String output_start_time = row_content[21].ToString();
                String output_end_time = row_content[22].ToString();
                String output_duration = row_content[23].ToString();
                String phase = row_content[24].ToString();
                String entity = row_content[25].ToString();
                String phase_entity_status = row_content[26].ToString();
                String phase_entity_affected_rows = row_content[27].ToString();
                String phase_entity_start_time = row_content[28].ToString();
                String phase_entity_end_time = row_content[29].ToString();
                String phase_entity_duration = row_content[30].ToString();
                String phase_entity_message = row_content[31].ToString();

                subject = "[" + customer.ToUpper() + "] [" + status + "]: " + process_name + " execution summary on " + computer_name.ToUpper();

                summary_header_text = "<p> <b> Running Server: [" + computer_name.ToUpper() + "] </b> </p>" +
               "<p> <b> Customer: [" + customer.ToUpper() + "] </b> </p>" +
               "<p> <b> Execution summary for process: " + process_name + " </b> </p>" +
               "<p> Process description: " + process_description + "</p>" +
               "<p> <u> Main summary </u> </p>" +
               "<table cellpadding=\"5\" cellspacing=\"0\"> " +
               "<tr bgcolor=\"DeepSkyBlue\"> " +
               "<th style=\"border: 1px solid black; border-collapse: collapse\"> STATUS </th> " +
               "<th style=\"border: 1px solid black; border-collapse: collapse\"> START TIME </th> " +
               "<th style=\"border: 1px solid black; border-collapse: collapse\"> END TIME </th> " +
               "<th style=\"border: 1px solid black; border-collapse: collapse\"> TOTAL DURATION </th> " +
               "<th style=\"border: 1px solid black; border-collapse: collapse\"> MESSAGE </th> " +
               "</tr> " +
               "<tr> " +
               (process_status == "SUCCEEDED" ? "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse; color:  green;\"> " : process_status == "WARNING" ? "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse; color:  orange;\"> " : "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse; color: red;\">") + "<b>" + process_status + " </b> </td> " +
               "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " + process_start_time + " </td> " +
               "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " + process_end_time + " </td> " +
               "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " + process_duration + " </td> " +
               "<td style=\"border: 1px solid black; border-collapse: collapse\"> " + process_message + " </td> " +
               "</tr> " +
               "</table>";

                summary_text = "<p> <u> Phase summary </u> </p>" +
               "<table cellpadding=\"5\" cellspacing=\"0\"> " +
               "<tr bgcolor=\"DeepSkyBlue\"> " +
               "<th style=\"border: 1px solid black; border-collapse: collapse\"> PHASE </th> " +
               "<th style=\"border: 1px solid black; border-collapse: collapse\"> STATUS </th> " +
               "<th style=\"border: 1px solid black; border-collapse: collapse\"> START TIME </th> " +
               "<th style=\"border: 1px solid black; border-collapse: collapse\"> END TIME </th> " +
               "<th style=\"border: 1px solid black; border-collapse: collapse\"> TOTAL DURATION </th> " +
               "</tr> " +
               (ingestion_status == "N/A" ? "" : "<tr> " +
               "<td style=\"border: 1px solid black; border-collapse: collapse\"> INGESTION </td> " +
               (ingestion_status == "SUCCEEDED" ? "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse; color: green;\"> " : ingestion_status == "WARNING" ? "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse; color:
```

```
      orange;\"> " : "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse; color: red;\"> ") + "<b>" +
      ingestion_status + " </b> </td> " +
                "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " + ingestion_start_time + " </td> " +
                "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " + ingestion_end_time + " </td> " +
                "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " + ingestion_duration + " </td> " +
                "</tr> ") +
                (transformation_status == "N/A" ? "" : "<tr> " +
                "<td style=\"border: 1px solid black; border-collapse: collapse\"> TRANSFORMATION </td> " +
                (transformation_status == "SUCCEEDED" ? "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse; color:
       green;\"> " : transformation_status == "WARNING" ? "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse; color
      : orange;\"> " : "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse; color: red;\"> ") + "<b>" +
      transformation_status + " </b> </td> " +
                "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " + transformation_start_time + " </td> " +
                "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " + transformation_end_time + " </td> " +
                "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " + transformation_duration + " </td> " +
                "</tr> ") +
                (persistence_status == "N/A" ? "" : "<tr> " +
                "<td style=\"border: 1px solid black; border-collapse: collapse\"> PERSISTENCE </td> " +
                (persistence_status == "SUCCEEDED" ? "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse; color:
      green;\"> " : persistence_status == "WARNING" ? "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse; color:
      orange;\"> " : "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse; color: red;\"> ") + "<b>" +
      persistence_status + " </b> </td> " +
                "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " + persistence_start_time + " </td> " +
                "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " + persistence_end_time + " </td> " +
                "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " + persistence_duration + " </td> " +
                "</tr> ") +
                (output_status == "N/A" ? "" : "<tr> " +
                "<td style=\"border: 1px solid black; border-collapse: collapse\"> OUTPUT </td> " +
                (output_status == "SUCCEEDED" ? "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse; color: green;\"
      > " : output_status == "WARNING" ? "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse; color: orange;\"> " :
      "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse; color: red;\"> ") + "<b>" + output_status + " </b> </td
      > " +
                "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " + output_start_time + " </td> " +
                "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " + output_end_time + " </td> " +
                "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " + output_duration + " </td> " +
                "</tr> ") +
                "</table>";

                  detail_text = detail_text + "<tr> " +
                "<td style=\"border: 1px solid black; border-collapse: collapse\"> " + phase + " </td> " +
                "<td style=\"border: 1px solid black; border-collapse: collapse\"> " + entity + " </td> " +
                (phase_entity_status == "SUCCEEDED" ? "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse; color:
      green;\"> " : phase_entity_status == "WARNING" ? "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse; color:
      orange;\"> " : "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse; color: red;\"> ") + "<b>" +
      phase_entity_status + " <b> </td> " +
                "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " + phase_entity_affected_rows + " </td> "
      +
                "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " + phase_entity_start_time + " </td> " +
                "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " + phase_entity_end_time + " </td> " +
                "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " + phase_entity_duration + " </td> " +
                "<td style=\"border: 1px solid black; border-collapse: collapse\"> " + phase_entity_message + " </td> " +
                "</tr> ";
            }

            detail_text = detail_text + "</table>";
            String body = summary_header_text + summary_text + detail_text;

            MailMessage mail = new MailMessage(from, to.Replace(";", ","), subject, body);
            mail.IsBodyHtml = true;
            SmtpClient client = new SmtpClient(smtp_host);
            client.Credentials = new System.Net.NetworkCredential(from, "Usuario2011");
            client.Send(mail);

            Dts.TaskResult = (int)ScriptResults.Success;
        }
        catch (Exception e)
        {
            Dts.Events.FireError(0, "Report Mail Script", e.Message + "\r" + e.StackTrace, String.Empty, 0);
            Dts.TaskResult = (int)ScriptResults.Failure;
        }

    }
      enum ScriptResults
      {
          Success = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Success,
          Failure = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Failure
      };
      #endregion
  }
}
```

A continuación se muestra el *Código C.2-C.4* en C# del fichero *Ingest_process*.

Código C.2: Ingest - Master status.cs

```
#region Namespaces
using System;
using System.Data;
using Microsoft.SqlServer.Dts.Runtime;
using System.Windows.Forms;
#endregion

namespace ST_352ff06fa2584b4180d9ec4eb421d970
{
   [Microsoft.SqlServer.Dts.Tasks.ScriptTask.SSISScriptTaskEntryPointAttribute]
   public partial class ScriptMain : Microsoft.SqlServer.Dts.Tasks.ScriptTask.VSTARTScriptObjectModelBase
```

```
    {
        public void Main()
        {
            Dts.Variables["User::v_status"].Value = Dts.Variables["User::v_p_status"].Value;

            Dts.TaskResult = (int)ScriptResults.Success;
        }

        #region ScriptResults declaration
        enum ScriptResults
        {
            Success = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Success,
            Failure = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Failure
        };
        #endregion

    }
}
```

Código C.3: Ingest - Move to archive.cs

```
#region Namespaces
using System;
using System.Data;
using Microsoft.SqlServer.Dts.Runtime;
using System.Windows.Forms;
using System.Security.Cryptography;
using System.Text;
using System.IO;
#endregion

namespace ST_afb8bdae899a4501830deecc39f059cd
{
    [Microsoft.SqlServer.Dts.Tasks.ScriptTask.SSISScriptTaskEntryPointAttribute]
    public partial class ScriptMain : Microsoft.SqlServer.Dts.Tasks.ScriptTask.VSTARTScriptObjectModelBase
    {
        public void Main()
        {
            var password = Dts.Variables["encription_pass"].GetSensitiveValue().ToString();

            var source = Dts.Variables["v_path_filename"].Value.ToString();
            var destination_folder = Dts.Variables["v_archive_destination_folder"].Value.ToString();
            var destination_file = Path.Combine(destination_folder, String.Concat(Path.GetFileNameWithoutExtension(source), "_encripted", Path.
    GetExtension(source)));

            File.Delete(source);

            System.IO.File.Move(destination_file, destination_folder);

            Dts.TaskResult = (int)ScriptResults.Success;
        }

        #region ScriptResults declaration
        enum ScriptResults
        {
            Success = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Success,
            Failure = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Failure
        };
        #endregion

    }
}
```

Código C.4: Ingest - Transform file.cs

```
#region Namespaces
using System.IO;
using System.Runtime.InteropServices;
using System.Text;
using Microsoft.Office.Interop.Excel;
#endregion

namespace ST_58afe4e3bd604440b3cf7066b3bc6ce0
{
    [Microsoft.SqlServer.Dts.Tasks.ScriptTask.SSISScriptTaskEntryPointAttribute]
    public partial class ScriptMain : Microsoft.SqlServer.Dts.Tasks.ScriptTask.VSTARTScriptObjectModelBase
    {
        public void Main()
        {
            var source = Dts.Variables["v_path_filename"].Value.ToString();
            var file_extension = Dts.Variables["v_file_extension"].Value.ToString().ToUpper();
            var ingestion_encoding_file = Dts.Variables["v_ingestion_encoding_file"].Value.ToString().ToUpper();

            if (file_extension == "XLS" || file_extension == "XLSX")
            {
                Microsoft.Office.Interop.Excel.Application excel = new Microsoft.Office.Interop.Excel.Application();

                Microsoft.Office.Interop.Excel.Workbooks wkbks = excel.Workbooks;
                Microsoft.Office.Interop.Excel.Workbook wkbk = wkbks.Open(source);

                excel.Visible = false;
                excel.DisplayAlerts = false;

                Worksheet worksheet = wkbk.Worksheets.get_Item(1);
                if (worksheet.Name != "Hoja1")
```

```
                {
                    worksheet.Name = "Hoja1";
                }

                worksheet.Cells.NumberFormat = "@";

                if (wkbk != null) {
                    try
                    {
                        wkbk.Save();
                        wkbk.Close(true, source);
                    }
                    catch
                    {
                    }

                    Marshal.FinalReleaseComObject(wkbk);
                    wkbk = null;
                }
            }
            else
            {
                if (ingestion_encoding_file != "")
                {
                    Encoding destination_encode = null;

                    if (ingestion_encoding_file == "ASCII")
                    {
                        destination_encode = Encoding.ASCII;
                    } else if (ingestion_encoding_file == "UTF8")
                    {
                        destination_encode = Encoding.UTF8;
                    }

                    StreamReader sourceStream = new StreamReader(source);
                    string fileContent = sourceStream.ReadToEnd();
                    sourceStream.Close();

                    StreamWriter destinationStream = new StreamWriter(source, false, destination_encode);
                    destinationStream.Write(fileContent);
                    destinationStream.Close();
                }
            }


            Dts.TaskResult = (int)ScriptResults.Success;
        }
          enum ScriptResults
          {
              Success = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Success,
              Failure = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Failure
          };
    }
}
```

A continuación se muestra el *Código C.5-C.6* en C# del fichero *Output_process*.

Código C.5: Output - Get affected rows.cs

```
#region Namespaces
using System;
using Microsoft.SqlServer.Dts.Runtime;
using System.Data.SqlClient;
using System.Data.OleDb;
using System.Windows.Forms;
using System.Data;
#endregion

namespace ST_9895c08fc965486faa99eac2b5d0e276
{
   [Microsoft.SqlServer.Dts.Tasks.ScriptTask.SSISScriptTaskEntryPointAttribute]
   public partial class ScriptMain : Microsoft.SqlServer.Dts.Tasks.ScriptTask.VSTARTScriptObjectModelBase
   {
      public void Main()
      {
            try
            {
                int affected_rows = 0;
                String query = (String)Dts.Variables["User::v_sql_output_query"].Value;
                int skip_rows = (int)Dts.Variables["User::v_destination_file_skip_rows"].Value;
                ConnectionManager cm = Dts.Connections["WIN-SER-AMAROTO.IO"];

                using (OleDbConnection connection = new OleDbConnection(cm.ConnectionString))
                {
                    OleDbCommand command = new OleDbCommand(query, connection);
                    connection.Open();

                    OleDbDataReader reader = command.ExecuteReader();

                    while (reader.Read())
                    {
                        affected_rows++;
                    }

                    Dts.Variables["User::v_affected_rows"].Value = affected_rows - skip_rows;
                }
```

```
38
39              Dts.TaskResult = (int)ScriptResults.Success;
40          }
41          catch (Exception e)
42          {
43              Dts.Events.FireError(0, "Get Output affected rows Script", e.Message + "\r" + e.StackTrace, String.Empty, 0);
44              Dts.TaskResult = (int)ScriptResults.Failure;
45          }
46      }
47        #region ScriptResults declaration
48        enum ScriptResults
49        {
50            Success = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Success,
51            Failure = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Failure
52        };
53        #endregion
54
55    }
56 }
```

Código C.6: Output - Send email.cs

```
1  #region Namespaces
2  using System;
3  using System.Net.Mail;
4  using System.IO;
5  #endregion
6
7  namespace ST_712178489edc4ca69c38495b49170f8d
8  {
9     [Microsoft.SqlServer.Dts.Tasks.ScriptTask.SSISScriptTaskEntryPointAttribute]
10    public partial class ScriptMain : Microsoft.SqlServer.Dts.Tasks.ScriptTask.VSTARTScriptObjectModelBase
11    {
12       public void Main()
13       {
14            try
15            {
16                String smtp_host = (String)Dts.Variables["$Package::email_server"].Value;
17
18                String from = (String)Dts.Variables["User::v_email_from"].Value;
19                String to = (String)Dts.Variables["User::v_email_to"].Value;
20                String subject = (String)Dts.Variables["User::v_email_subject"].Value;
21                String body = (String)Dts.Variables["User::v_email_body"].Value;
22                int has_attachment = (int)Dts.Variables["User::v_email_has_attachment"].Value;
23                String attachment_path = (String)Dts.Variables["User::v_email_attachment_path"].Value;
24                String attachment_pattern = (String)Dts.Variables["User::v_email_attachment_pattern"].Value;
25
26                SmtpClient client = new SmtpClient(smtp_host);
27                client.Credentials = new System.Net.NetworkCredential(from, "Usuario2011");
28
29                MailMessage mail = new MailMessage(from, to.Replace(";", ","), subject, body);
30                mail.IsBodyHtml = true;
31
32                if (has_attachment == 1)
33                {
34                    if (Directory.Exists(attachment_path))
35                    {
36                        string[] fileEntries = Directory.GetFiles(attachment_path, attachment_pattern);
37                        Dts.Variables["User::v_affected_rows"].Value = fileEntries.Length;
38
39                        body = body + "<p> " + fileEntries.Length.ToString() + " files have been attached into mail: </p>";
40
41                        body = body + "<table cellpadding=\"5\" cellspacing=\"0\"> " +
42                    "<tr bgcolor=\"DeepSkyBlue\"> " +
43                    "<th style=\"border: 1px solid black; border-collapse: collapse\"> File name </th> " +
44                    "<th style=\"border: 1px solid black; border-collapse: collapse\"> Size </th> " +
45                    "</tr> ";
46
47                        Attachment file_attachment;
48                        float size = 0;
49
50                        foreach (string fileName in fileEntries)
51                        {
52                            FileInfo fi = new FileInfo(fileName);
53                            size = size + fi.Length;
54
55                            body = body + "<tr> " +
56                        "<td style=\"border: 1px solid black; text-align: left; border-collapse: collapse\"> " + fi.Name + " </td> " +
57                        "<td style=\"border: 1px solid black; text-align: center; border-collapse: collapse\"> " + fi.Length.ToString() + " bytes </td
      > " +
58                        "</tr> ";
59
60                            file_attachment = new Attachment(fileName);
61                            mail.Attachments.Add(file_attachment);
62                        }
63
64                        size = size / 1000000;
65
66                        if (size >= 20)
67                        {
68                            mail.Attachments.Clear();
69                            body = body + "</table> <p style = \"color: red;\"> <b> " + fileEntries.Length.ToString() + " files should have been
      attached into mail but the total size (" + size + " MB) exceeds the allowed size (20 MB), NO FILE HAS BEEN ATTACHED </b> </p>";
70                        }
71                        else
72                        {
73                            body = body + " </table > <p style = \"color: green;\"> <b> " + fileEntries.Length.ToString() + " files have been
      attached into mail with " + size + " MB of total size</p>";
```

```
                        }

                        mail.Body = body;
                    }
                    else
                    {
                        throw new InvalidOperationException("Folder for attached files does not exist");
                    }
                }
                else
                {
                    Dts.Variables["User::v_affected_rows"].Value = 1;
                }

                client.Send(mail);

                client.Dispose();
                mail.Dispose();

                Dts.TaskResult = (int)ScriptResults.Success;
            }
            catch (Exception e)
            {
                Dts.Events.FireError(0, "Report Mail Script", e.Message + "\r" + e.StackTrace, String.Empty, 0);
                Dts.TaskResult = (int)ScriptResults.Failure;
            }

        }
        #region ScriptResults declaration
        enum ScriptResults
        {
            Success = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Success,
            Failure = Microsoft.SqlServer.Dts.Runtime.DTSExecResult.Failure
        };
        #endregion
    }
}
```

La *Figura C.2* muestra el conjunto de scripts (*Código C.7-C.9*) PowerShell de la solución SSIS.

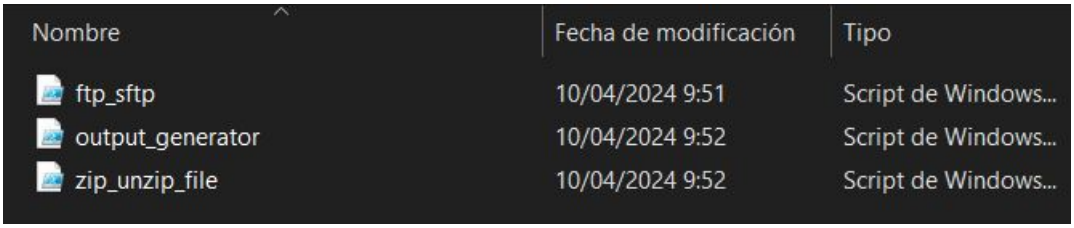


Figura C.2: Estructura de scripts PowerShell embebidos en la solución IO

Código C.7: ftp_stftp.ps1

```
<# ============================================================
Autor: Adrian Maroto
Fecha creacion: 09/01/2024
Descripcion: Script que realiza una conexión FTP o SFTP y carga o descarga archivos que tienen que cumplir un patrón.
Cambios:

Entrada:
    $protocol: tipo de conexión al servidor, valores posibles FTP o SFTP
    $server: nombre o IP del servidor al que conectarse
    $port: puerto de conexión
    $user: usuario de las credenciales de conexión al servidor
    $pass: password de las credenciales de conexión al servidor
    $ftp_operation: tipo de operación que se realizará sobre el servidor, valores posibles UPLOAD o DOWNLOAD
    $ftp_folder: directorio remoto sobre el que se realizará la operación
    $ftp_file_pattern: patrón que debe cumplir los archivos en el lado del servidor (sólo válido para operaciones de descarga, DOWNLOAD)
    $local_folder: directorio local sobre el que se realizará la operación
    $local_file_pattern: patrón que debe cumplir los archivos locales (sólo válido para operaciones de carga, UPLOAD)

    $ftp_private_key_path: directorio donde se encuentra la clave privada en caso de conexión segura
    $ftp_ssh_host_key_fingerprint: identificador de la huella del servidor
    $ftp_private_key_passphrase: password de uso de la clave privada de acceso al servidor

Salida:
   - DOWNLOAD/UPLOAD del fichero.

Ejemplo:
    FTP-DOWNLOAD: ftp_sftp.ps1 -protocol "FTP" -server "nombre_o_IP_servidor_ftp" -port "puerto" -user "usuario_conexion" -pass "contraseña_conexion" -
      ftp_operation "DOWNLOAD" -ftp_folder "directorio_remoto_descarga" -ftp_file_pattern "patrón_archivos_descarga" -local_folder "
      directorio_local_descarga"
    FTP-UPLOAD: ftp_sftp.ps1 -protocol "FTP" -server "nombre_o_IP_servidor_ftp" -port "puerto" -user "usuario_conexion" -pass "contraseña_conexion" -
      ftp_operation "UPLOAD" -ftp_folder "directorio_remoto_carga" -local_folder "directorio_local_carga" -local_file_pattern "patrón_archivos_carga"
    SFTP-DOWNLOAD: ftp_sftp.ps1 -protocol "SFTP" -server "nombre_o_IP_servidor_ftp" -port "puerto" -user "usuario_conexion" -pass "contraseña_conexion"
       -ftp_operation "DOWNLOAD" -ftp_folder "directorio_remoto_descarga" -ftp_file_pattern "patrón_archivos_descarga" -local_folder "
      directorio_local_descarga"
    SFTP-UPLOAD: ftp_sftp.ps1 -protocol "SFTP" -server "nombre_o_IP_servidor_ftp" -port "puerto" -user "usuario_conexion" -pass "contraseña_conexion" -
      ftp_operation "UPLOAD" -ftp_folder "directorio_remoto_carga" -local_folder "directorio_local_carga" -local_file_pattern "patrón_archivos_carga"
```

```
31  ============================================================ #>
32
33  param (
34      [string] $protocol,
35      [string] $server,
36      [int] $port,
37      [string] $user,
38      [string] $pass,
39      [string] $ftp_operation,
40      [string] $ftp_folder,
41      [string] $ftp_file_pattern,
42      [string] $local_folder,
43      [string] $local_file_pattern,
44
45      [string] $ftp_private_key_path,
46      [string] $ftp_ssh_host_key_fingerprint,
47      [string] $ftp_private_key_passphrase
48  )
49
50  Set-PSDebug -Off
51
52  Try
53  {
54      # Inicialización de la variable de mensaje de salida
55      $error_message = 'OK'
56
57      # desbloqueo del archivo para poder ser ejecutado en remoto
58      $main_dir = $PSScriptRoot.ToString()
59      Get-ChildItem -Path $main_dir -File -Recurse -Force |  % {
60          Unblock-File -Path $_.FullName
61      }
62
63      # Carga de la libreria WinSCP .NET
64      [System.Reflection.Assembly]::UnsafeLoadFrom($main_dir + "\WinSCP\WinSCPnet.dll") | Out-Null
65
66      #Estandarización del protocolo (primera mayúscula, resto minúsculas)
67      $protocol = $protocol.substring(0,1).ToUpper() + $protocol.substring(1).ToLower()
68
69      # Configuración de las opciones de sesión
70      $sessionOptions = New-Object WinSCP.SessionOptions -Property @{
71          Protocol = [WinSCP.Protocol]::$protocol
72          HostName = $server
73          PortNumber = $port
74          UserName = $user
75          Password = $pass
76      }
77
78      # En caso de que se trate de una conexión SFTP se acepta la SSH Host Key predeterminada y en caso de existir clave privada, se carga
79      if ($protocol -eq "Sftp")
80      {
81          $sessionOptions.GiveUpSecurityAndAcceptAnySshHostKey = "true"
82
83          if($ftp_private_key_path -ne ""){
84              $sessionOptions.SshPrivateKeyPath = $ftp_private_key_path
85          }
86
87          if($ftp_private_key_passphrase -ne ""){
88              $sessionOptions.PrivateKeyPassphrase = $ftp_private_key_passphrase
89          }
90      }
91      else
92      {
93          $sessionOptions.FtpSecure = "Explicit"
94          $sessionOptions.GiveUpSecurityAndAcceptAnyTlsHostCertificate = "true"
95      }
96
97      #En caso de tener la fingerprint de la máquina desde donde se ejecuta el cliente, se carga su valor
98      if($ftp_ssh_host_key_fingerprint -ne ""){
99          $sessionOptions.SshHostKeyFingerprint = $ftp_ssh_host_key_fingerprint
100     }
101
102     # Creación de la sesión
103     $session = New-Object WinSCP.Session
104
105     # Conectar
106     $session.Open($sessionOptions)
107
108     # Según la operación se realizan las diferentes acciones
109     if ($ftp_operation -eq "DOWNLOAD")
110     {
111         if (!(Test-Path $local_folder))
112         {
113               New-Item -ItemType Directory -Path $local_folder -Force | Out-Null
114         }
115
116         $all_files = $session.EnumerateRemoteFiles($ftp_folder, $ftp_file_pattern, [WinSCP.EnumerationOptions]::None)
117
118         $count_files = 0
119
120         foreach ($file in $all_files)
121         {
122             $count_files = $count_files + 1
123             $download_file =  $ftp_folder + $file.Name
124             $session.GetFiles($download_file, ($local_folder + '\ ')).Check()
125             $session.RemoveFiles($download_file) | Out-Null
126         }
127
128         if ($count_files -eq 0){
129             throw "No file matching template: " + $ftp_file_pattern + " on " + $protocol.ToUpper() + " server path: " + $ftp_folder + ". Unable to
```

```
      DOWNLOAD any file"
        }
    }

    if ($ftp_operation -eq "UPLOAD")
    {
        if (!$session.FileExists($ftp_folder))
        {
            $session.CreateDirectory($ftp_folder)
        }

        $all_files =  Get-ChildItem -Path $local_folder -File | where {$_.name -like $local_file_pattern}

        $count_files = 0

        $transferOptions = New-Object WinSCP.TransferOptions
        $transferOptions.FilePermissions = $Null
        $transferOptions.PreserveTimestamp = $False

        foreach ($file in $all_files)
        {
            $count_files = $count_files + 1
            $upload_file = $local_folder + '\ ' + $file.Name
            $session.PutFiles($upload_file, $ftp_folder, $False, $transferOptions).Check()
        }

        if ($count_files -eq 0){
            throw "No file matching template: " + $local_file_pattern + " on path: " + $local_folder + ". Unable to UPLOAD to " + $protocol.ToUpper()
        }
    }
}
Catch
{
    if ($count_files -eq 0)
    {
        $error_message = 'WARNING: '
    } else
    {
        $error_message = 'KO: '
    }

    $error_message = $error_message + $_.Exception.Message
}
finally
{
    # Desconexión
    if (Get-Variable -Name session -ErrorAction SilentlyContinue)
    {
        $session.Dispose()
    }

    Write-Host -NoNewline $error_message
}
```

Código C.8: output_generator.ps1

```
<# ============================================================
Autor: Adrian Maroto
Fecha creacion: 16/01/2024
Descripcion: Script que genera un fichero en diferentes formatos a partir de una query.
Cambios:

Entrada:
    $database: base de datos sobre la que se ejecutará la consulta que genera el output
    $query: query que se ejecutará para generar el output
    $out_path: ruta donde se depositará el output a partir de la query
    $fmt_path: ruta donde se encuentra el archivo de configuración del output
    $sql_server: servidor donde se ejecutará la query para generar el output
    $output_format: formato del archivo de salida
    $header_rows: número de filas que tiene la cabecera
    $generate_empty: indicador de si se debe generar o no el fichero en caso que solo haya cabecera en el contenido
    $archive_folder: carpeta en la que se archivará el archivo generado copiandose el original

Salida:
   - Fichero con el formato indicado.

Ejemplo:
    output_generator.ps1 -query "SELECT * FROM db.schema.tabla" -out_path "ruta_destino_incluyendo_archivo" -fmt_path "ruta_archivo_configuración" -
      sql_server "nombre_servidor_SQL_origen" -database "bbdd_origen" -output_format "formato archivo output" -header_rows 3 -generate_empty 1)
============================================================ #>

param (
   [string] $database,
   [string] $query,
   [string] $out_path,
   [string] $fmt_path,
   [string] $sql_server,
   [string] $output_format,
   [int] $header_rows,
   [int] $generate_empty,
   [string] $archive_folder
)


Set-PSDebug -Off

Try
{
```

```
    # Inicialización de la variable de mensaje de salida
    $error_message = 'OK'

    # Se crea el directorio de destino en caso de no existir
    If(!(test-path -PathType container $archive_folder))
    {
        New-Item -ItemType Directory -Path $archive_folder | Out-Null
    }

    # Se reemplazan los carácteres especiales @@@ por "" que son necesarias dentro de la query, además se omiten los saltos de línea
    $query = $query.Replace("@@@", '`"`"').replace("`n"," ")

    # desbloqueo del archivo para poder ser ejecutado en remoto
    $main_dir = $PSScriptRoot.ToString()
    Get-ChildItem -Path $main_dir -File -Recurse -Force |  % {
        Unblock-File -Path $_.FullName
    }

    # Modificación del nombre del archivo de salida para que contenga la fecha de hoy
    $out_path = $out_path.Replace('yyyy', (Get-Date -Format yyyy).ToString()).Replace('mm', (Get-Date -Format MM).ToString()).Replace('dd', (Get-Date -
      Format dd).ToString()).Replace('hh', (Get-Date -Format hh).ToString()).Replace('min', (Get-Date -Format mm).ToString()).Replace('ss', (Get-Date -
      Format ss).ToString())

    # Modificación del nombre del archivo para que contenga la extensión
    # En caso de ser formato Excel se creará primero un CSV para posteriormente modificarlo
    if (($output_format -eq 'TXT') -or ($output_format -eq 'CSV'))
    {
        $bcp_out_file = $out_path + '.' + $output_format.ToLower()
    }
    else
    {
        $bcp_out_file = $out_path + '.csv'
    }

    # Se elimina el archivo CSV en caso de existir
    if (Test-Path $bcp_out_file) {
        Remove-Item $bcp_out_file
    }

    # Creación del archivo CSV a partir de la query de entrada y el archivo de configuración FMT
    $bcp_command = "BCP `"$query`" queryout `"$bcp_out_file`" -f `"$fmt_path`" -S `"$sql_server`" -d `"$database`" -T"

    Invoke-Expression $bcp_command | Out-Null

    #Se copia el archivo en la ubicación de archivado
    Copy-Item $bcp_out_file -Destination $archive_folder -Force

    # Se calcula si la cantidad de filas en el archivo resultado es únicamente el de las cabeceras
    $empty_file = (Get-Content -Path $bcp_out_file | Measure-Object -Line | Select -ExpandProperty "Lines") -le $header_rows

    # Si el archivo solo contiene las cabeceras, se borra en caso contrario se continua
    if ($empty_file -and $generate_empty -eq 0)
    {
        # Se elimina el archivo CSV del BCP
        Remove-Item $bcp_out_file
    }
    else
    {
        # Si se espera un Excel como formato del output, se transforma el CSV en Excel
        if (($output_format -eq 'XLS') -or ($output_format -eq 'XLSX'))
        {
            # Ruta de salida del Excel
            $excel_out_path = $out_path + '.' + $output_format.ToLower()

            # Creación del Excel añadiendo un nuevo libro y hoja
            $excel = New-Object -ComObject excel.application
            $workbook = $excel.Workbooks.Add(1)
            $worksheet = $workbook.worksheets.Item(1)

            # Se especifica el formato como solo texto
            $worksheet.Cells.NumberFormat = "@"

            # Se recorre cada una de las filas y columnas leyendo y asignado el valor al Excel
            $i = 1
            Import-Csv $bcp_out_file -Delimiter ";" | ForEach-Object {
              $j = 1
              foreach ($prop in $_.PSObject.Properties) {
                if ($i -eq 1) {
                  $worksheet.Cells.Item($i, $j++).Value = $prop.Name
                  $worksheet.Cells.Item($i+1, $j-1).Value = $prop.Value
                } else {
                  $worksheet.Cells.Item($i+1, $j++).Value = $prop.Value
                }
              }
              $i++
            }

            # Se elimina el archivo Excel en caso de existir
            if (Test-Path $excel_out_path) {
                Remove-Item $excel_out_path
            }

            # Se guarda el archivo Excel diferenciando entre versiones de Excel
            if ($output_format -eq 'XLSX')
            {
                $Workbook.SaveAs($excel_out_path,51)
            }
            else
            {
```

```
                $Workbook.SaveAs($excel_out_path,56)
            }

            # Se cierra el archivo Excel
            $excel.Quit()

            # Se eliminan todos los procesos de tipo Excel para evitar errores
            # Check for process by name
            Try {
                $processes = Get-Process -Name "Excel" -ErrorAction Stop

                foreach($process in $processes)
                {
                    Stop-Process -InputObject $process
                }

                $clean = 1
            }
            Catch {
                $clean = 0
            }

            #Se copia el archivo en la ubicación de archivado
            Copy-Item $excel_out_path -Destination $archive_folder -Force

            # Se elimina el archivo CSV del BCP
            Remove-Item $bcp_out_file
        }
    }
}
Catch
{
    $error_message = 'KO ' + $_.Exception.Message
}
finally
{
    Write-Host -NoNewline $error_message
}
```

Código C.9: zip_unzip_file.ps1

```
<# ============================================================
Autor: Adrian Maroto
Fecha creacion: 23/01/2024
Descripcion: Script que comprime/descomprime un archivo depositando el resultado en otra carpeta.
Cambios:

Entrada:
    $zip_operation: tipo de conexión al servidor, valores posibles FTP o SFTP
    $zip_folder: carpeta donde se encuentra el archivo comprimido
    $zip_file: nombre del archivo comprimido
    $output_folder: carpeta destino de los archivos descomprimidos
    $archive_folder: ruta_principal_carpeta_archivado
    $zip_rar_pass: contraseña_de_descompresion

Salida:
   - ZIP/UNZIP del fichero.

Ejemplo:
    UNZIP: zip_unzip_files.ps1 -zip_operation "UNZIP" -file "ruta_completa_archivo_zip" -folder "carpeta_destino_archivos_descomprimidos" -
      archive_folder "ruta_principal_carpeta_archivado"
    ZIP: zip_unzip_files.ps1 -zip_operation "ZIP" -folder "ruta_carpeta_archivos_zipear\*" -file "ruta_archivo_comprimido_zip"
============================================================ #>

param (
   [string] $zip_operation,
   [string] $folder,
   [string] $file,
   [string] $archive_folder,
   [string] $zip_rar_pass
)

Set-PSDebug -Off

Try
{
    # Inicialización de la variable de mensaje de salida
    $error_message = 'OK'

    #Se añade una contrabarra a la ruta de archivado para poder crear el directorio en caso de no existir
    $archive_folder = $archive_folder + "\ "

    if (!(Test-Path $archive_folder))
    {
        New-Item -ItemType Directory -Path $archive_folder -Force | Out-Null
    }

    # Desbloqueo del archivo para poder ser ejecutado en remoto
    $main_dir = $PSScriptRoot.ToString()
    Get-ChildItem -Path $main_dir -File -Recurse -Force |  % {
        Unblock-File -Path $_.FullName
    }

    if ($zip_operation -eq 'ZIP'){

        $file = $file.Replace("yyyy", (get-date -Format yyyy)).Replace("mm", (get-date -Format MM)).Replace("dd-1", (get-date).AddDays(-1).Day.ToString
      ()).Replace("dd", (get-date -Format dd))
        $zip_file_folder = (Split-Path $folder -parent) + "\ "
```

```
        $zip_file_template = Split-Path $folder -leaf

        #If(Test-path $file){
        #    Remove-item $file
        #}

        # Compresión del archivo RAR sin contraseña
        if ($zip_rar_pass -eq ''){
            Compress-7Zip -ArchiveFileName $file -Path $zip_file_folder -Filter $zip_file_template -Format Zip
        # Compresión del archivo RAR con contraseña
        } else {
            Compress-7Zip -ArchiveFileName $file -Path $zip_file_folder -Filter $zip_file_template -Format Zip -Password $zip_rar_pass
        }

        # Se archiva el fichero ZIP una vez comprimido
        Copy-Item $file -Destination $archive_folder -Force
    }

    if ($zip_operation -eq 'UNZIP'){

        $zip_file_folder = Split-Path $file -parent
        $zip_file_template = Split-Path $file -leaf

        Get-ChildItem -Path $zip_file_folder -File | where {$_.name -like $zip_file_template} | % {

            $zip_filename = $_.FullName

            # Descompresión del archivo sin contraseña
            if ($zip_rar_pass -eq ''){
                Expand-7Zip -ArchiveFileName $zip_filename -TargetPath $folder
            # Descompresión del archivo RAR con contraseña
            } else {
                Expand-7Zip -ArchiveFileName $zip_filename -TargetPath $folder -Password $zip_rar_pass
            }
        }

        # Se archiva el fichero ZIP una vez descomprimido
        Move-Item $zip_filename $archive_folder -Force
    }

}
Catch
{
    $error_message = 'KO ' + $_.Exception.Message
}
finally
{
    Write-Host -NoNewline $error_message
}
```

Solución BBDD

La *Figura C.3* muestra el conjunto de scripts de creación SQL para las BBDD del proyecto IO.

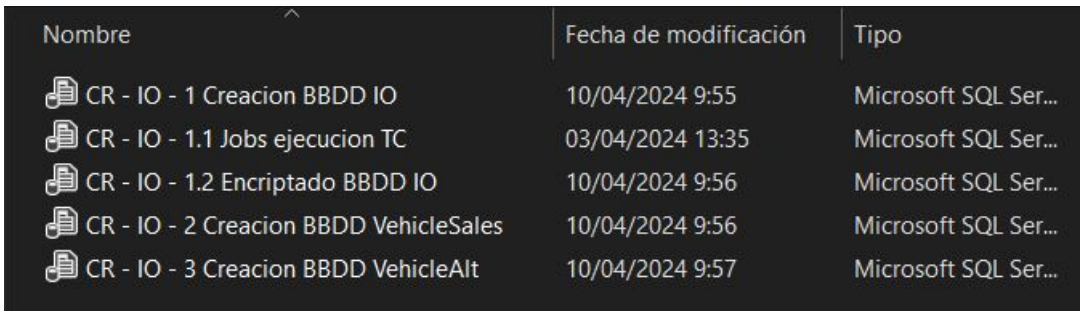


Figura C.3: Estructura de scripts de creación SQL para las BBDD del proyecto IO

A continuación se muestra el *Código C.10* SQL para la creación de la BBDD IO.

Código C.10: 1 Creacion BBDD IO.sql

```
/*
Es muy importante que la configuración de los componentes OLEDB sea como se indica en la documentación,
de lo contrario, se producen errores en la lectura y generación de archivos Excel:

Una vez instaladas ambas librerías, estas aparecen en los proveedores de SQL Server.
Se puede comprobar con la ejecución de la siguiente query:
*/
EXEC sp_MSset_oledb_prop

-- ============================================================
```

```
-- ========== CREACION BBDD ==========
-- ============================================================
-- Create IO Database
CREATE DATABASE IO;
GO

-- Use IO Database
USE IO;
GO

/*
USE Master;
ALTER DATABASE IO SET single_user with rollback immediate;
ALTER DATABASE IO SET MULTI_USER;
DROP DATABASE IO;
*/

-- ============================================================
-- ========== CREACION SCHEMAS ==========
-- ============================================================

-- Create [auxiliar] schema
CREATE SCHEMA [auxiliar]
GO
-- Create [config] schema
CREATE SCHEMA [config]
GO
-- Create [hist] schema
CREATE SCHEMA [hist]
GO
-- Create [input] schema
CREATE SCHEMA [input]
GO
-- Create [load] schema
CREATE SCHEMA [load]
GO
-- Create [monitor] schema
CREATE SCHEMA [monitor]
GO
-- Create [output] schema
CREATE SCHEMA [output]
GO

-- ============================================================
-- ========== CREACION TABLAS ==========
-- ============================================================

-- Create [config].[customer_cluster_type] Table
-- IO.config.customer_cluster_type: tabla que contendrá las agrupaciones por clientes
CREATE TABLE [config].[customer_cluster_type](
   [id] [int] IDENTITY(1,1) NOT NULL,
   [customer_id] [int] NOT NULL,
   [name] [varchar](64) NOT NULL,
   [cluster_id] [int] NULL,
   [cluster_name] [varchar](128) NULL,
   [data_creation_date] [datetime] NULL,
   [data_modification_date] [datetime] NULL,
   [last_action_user] [varchar](1024) NULL,
 CONSTRAINT [PK_customer_book_name_type] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [config].[customer_cluster_type] ADD DEFAULT (getdate()) FOR [data_creation_date]
GO
ALTER TABLE [config].[customer_cluster_type] ADD DEFAULT (original_login()) FOR [last_action_user]
GO


-- Create [config].[customer_persistence_destination] Table
-- IO.config.customer_persistence_destination: tabla que contendrá la BBDD destino por cliente
CREATE TABLE [config].[customer_persistence_destination](
   [id] [int] IDENTITY(1,1) NOT NULL,
   [customer_id] [int] NOT NULL,
   [destination_db] [varchar](64) NOT NULL,
   [data_creation_date] [datetime] NULL,
   [data_modification_date] [datetime] NULL,
   [last_action_user] [varchar](1024) NULL,
 CONSTRAINT [PK_customer_persistence_destination_id] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF) ON [PRIMARY],
CONSTRAINT [UQ_customer__persistance_destination_cust_id] UNIQUE NONCLUSTERED
(
   [customer_id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [config].[customer_persistence_destination] ADD  DEFAULT (getdate()) FOR [data_creation_date]
GO
ALTER TABLE [config].[customer_persistence_destination] ADD  DEFAULT (original_login()) FOR [last_action_user]
GO

```

```
107
108 -- Create [config].[detail_process_type] Table
109 -- IO.config.detail_process_type: tabla que contendrá la configuración a nivel de detalle de cada uno de los pasos del proceso.
110 CREATE TABLE [config].[detail_process_type](
111    [id] [int] NOT NULL,
112    [master_process_type_id] [int] NOT NULL,
113    [phase] [varchar](16) NOT NULL,
114    [step] [int] NOT NULL,
115    [entity] [varchar](512) NULL,
116    [name] [varchar](128) NULL,
117    [description] [varchar](1024) NULL,
118    [source_type] [varchar](5) NULL,
119    [zip_folder_path] [varchar](1024) NULL,
120    [zip_file_path] [varchar](1024) NULL,
121    [source_file_folder_path] [varchar](1024) NULL,
122    [source_file_pattern] [varchar](128) NULL,
123    [add_eof] [int] NULL,
124    [skip_final_rows] [int] NULL,
125    [skip_initial_rows] [int] NULL,
126    [source_file_template_path] [varchar](1024) NULL,
127    [source_table_query] [varchar](max) NULL,
128    [destination_table] [varchar](1024) NULL,
129    [destination_format] [varchar](1024) NULL,
130    [generate_empty_file] [bit] NULL,
131    [destination_file_skip_rows] [int] NULL,
132    [destination_path] [varchar](1024) NULL,
133    [destination_file_template_path] [varchar](1024) NULL,
134    [ftp_server] [varchar](128) NULL,
135    [ftp_port] [int] NULL,
136    [ftp_user] [varchar](128) NULL,
137    [ftp_pass] [varbinary](256) NULL,
138    [ftp_private_key_path] [varchar](1024) NULL,
139    [ftp_ssh_host_key_fingerprint] [varchar](128) NULL,
140    [ftp_private_key_passphrase] [varbinary](256) NULL,
141    [ftp_operation] [varchar](8) NULL,
142    [ftp_folder] [varchar](1024) NULL,
143    [ftp_file_pattern] [varchar](1024) NULL,
144    [ftp_local_folder] [varchar](1024) NULL,
145    [ftp_local_file_pattern] [varchar](1024) NULL,
146    [email_from] [varchar](256) NULL,
147    [email_to] [varchar](1024) NULL,
148    [email_subject] [varchar](1024) NULL,
149    [email_body] [varchar](max) NULL,
150    [email_has_attachment] [int] NULL,
151    [email_attachment_path] [varchar](1024) NULL,
152    [email_attachment_pattern] [varchar](1024) NULL,
153    [data_creation_date] [datetime] NULL,
154    [data_modification_date] [datetime] NULL,
155    [last_action_user] [varchar](1024) NULL,
156    [file_action] [varchar](6) NULL,
157    [file_action_source_pattern] [varchar](1024) NULL,
158    [file_action_destination_folder] [varchar](1024) NULL,
159    [check_ingestion_filename] [int] NULL,
160    [source_connection_server] [varchar](64) NULL,
161    [source_connection_database] [varchar](64) NULL,
162    [source_connection_user] [varchar](64) NULL,
163    [source_connection_password] [varbinary](256) NULL,
164    [destination_connection_server] [varchar](64) NULL,
165    [destination_connection_database] [varchar](64) NULL,
166    [destination_connection_user] [varchar](64) NULL,
167    [destination_connection_password] [varbinary](256) NULL,
168    [ingestion_encoding_file] [varchar](5) NULL,
169    [zip_rar_pass] [varbinary](256) NULL,
170 PRIMARY KEY CLUSTERED
171 (
172    [id] ASC
173 )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
       OFF) ON [PRIMARY]
174 ) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
175 GO
176
177 ALTER TABLE [config].[detail_process_type]  WITH CHECK ADD  CONSTRAINT [CK_detail_process_type_check_ingestion_filename_logical] CHECK
       (([phase]='INGESTION' AND [source_type]='FILE' AND ([check_ingestion_filename]=(0) OR [check_ingestion_filename]=(1)) OR [phase]='INGESTION' AND
       [source_type]<>'FILE' AND [check_ingestion_filename] IS NULL OR [phase]<>'INGESTION' AND [check_ingestion_filename] IS NULL))
178 GO
179 ALTER TABLE [config].[detail_process_type] CHECK CONSTRAINT [CK_detail_process_type_check_ingestion_filename_logical]
180 GO
181 ALTER TABLE [config].[detail_process_type]  WITH CHECK ADD  CONSTRAINT [CK_detail_process_type_check_ingestion_filename_values] CHECK
       (([check_ingestion_filename]=NULL OR [check_ingestion_filename]=(0) OR [check_ingestion_filename]=(1)))
182 GO
183 ALTER TABLE [config].[detail_process_type] CHECK CONSTRAINT [CK_detail_process_type_check_ingestion_filename_values]
184 GO
185 ALTER TABLE [config].[detail_process_type]  WITH CHECK ADD  CONSTRAINT [CK_detail_process_type_destination_connection] CHECK
       (([destination_connection_server] IS NOT NULL AND [destination_connection_database] IS NOT NULL AND [destination_connection_user] IS NOT NULL
       AND [destination_connection_password] IS NOT NULL OR [destination_connection_server] IS NULL AND [destination_connection_database] IS NULL AND
       [destination_connection_user] IS NULL AND [destination_connection_password] IS NULL))
186 GO
187 ALTER TABLE [config].[detail_process_type] CHECK CONSTRAINT [CK_detail_process_type_destination_connection]
188 GO
189 ALTER TABLE [config].[detail_process_type]  WITH CHECK ADD  CONSTRAINT [CK_detail_process_type_source_connection] CHECK (([source_connection_server]
       IS NOT NULL AND [source_connection_database] IS NOT NULL AND [source_connection_user] IS NOT NULL AND [source_connection_password] IS NOT NULL
       OR [source_connection_server] IS NULL AND [source_connection_database] IS NULL AND [source_connection_user] IS NULL AND
       [source_connection_password] IS NULL))
190 GO
191 ALTER TABLE [config].[detail_process_type] CHECK CONSTRAINT [CK_detail_process_type_source_connection]
192 GO
193
194
195 -- Create [config].[master_process_type] Table
```

```
-- IO.config.master_process_type: tabla que contendrá la configuración a nivel de maestro de proceso completo.
CREATE TABLE [config].[master_process_type](
   [id] [int] NOT NULL,
   [customer] [varchar](128) NULL,
   [name] [varchar](128) NULL,
   [description] [varchar](1024) NULL,
   [active_flag] [bit] NOT NULL,
   [ingestion_flag] [bit] NULL,
   [transformation_flag] [bit] NULL,
   [persist_flag] [bit] NULL,
   [output_flag] [bit] NULL,
   [data_creation_date] [datetime] NULL,
   [data_modification_date] [datetime] NULL,
   [last_action_user] [varchar](1024) NULL,
   [stop_on_warning] [bit] NULL,
 CONSTRAINT [PK_ingest_master_process_type_id] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [config].[master_process_type] ADD  DEFAULT ((1)) FOR [active_flag]
GO
ALTER TABLE [config].[master_process_type] ADD  DEFAULT (getdate()) FOR [data_creation_date]
GO
ALTER TABLE [config].[master_process_type] ADD  DEFAULT (original_login()) FOR [last_action_user]
GO
ALTER TABLE [config].[master_process_type] ADD  DEFAULT ((1)) FOR [stop_on_warning]
GO


-- Create [load].[delivery] Table
-- IO.load.delivery: tabla que contendrá cada una de las deliveries por ejecución y cliente.
CREATE TABLE [load].[delivery](
   [id] [int] IDENTITY(1,1) NOT NULL,
   [customer_id] [int] NOT NULL,
   [master_process_log_id] [int] NOT NULL,
   [registration_date] [datetime] NULL,
 CONSTRAINT [PK_delivery] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [load].[delivery]  WITH CHECK ADD  CONSTRAINT [FK_delivery_master_process_log_id] FOREIGN KEY([master_process_log_id])
REFERENCES [monitor].[master_process_log] ([id])
GO
ALTER TABLE [load].[delivery] CHECK CONSTRAINT [FK_delivery_master_process_log_id]
GO


-- Create [load].[operation] Table
-- IO.load.operation: tabla que contendrá cada una de las operaciones calculadas durante el proceso de transformación.
CREATE TABLE [load].[operation](
   [id] [int] IDENTITY(1,1) NOT NULL,
   [delivery_id] [int] NULL,
   [load_table_name] [varchar](255) NULL,
   [load_reference_id] [int] NULL,
   [master_reference_id] [int] NULL,
   [operation_type_id] [int] NULL,
   [operation_finished] [bit] NULL,
PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [load].[operation] ADD  CONSTRAINT [DF_operation_operation_finished]  DEFAULT ((0)) FOR [operation_finished]
GO

ALTER TABLE [load].[operation]  WITH CHECK ADD  CONSTRAINT [FK_operation_operation] FOREIGN KEY([operation_type_id])
REFERENCES [load].[operation_type] ([id])
GO
ALTER TABLE [load].[operation] CHECK CONSTRAINT [FK_operation_operation]
GO


-- Create [load].[operation_type] Table
-- IO.load.operation_type: tabla que contendrá los tipos de operaciones permitidos durante la fase de transformación.
CREATE TABLE [load].[operation_type](
   [id] [int] NOT NULL,
   [operation_type_name] [varchar](10) NOT NULL,
PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF) ON [PRIMARY]
) ON [PRIMARY]
GO


-- Create [monitor].[detail_process_log] Table
-- IO.monitor.detail_process_log: tabla de log de la ejecución a nivel de detalle de cada paso del proceso.
```

```
CREATE TABLE [monitor].[detail_process_log](
   [id] [int] IDENTITY(1,1) NOT NULL,
   [master_process_log_id] [int] NULL,
   [phase] [varchar](16) NULL,
   [entity] [varchar](128) NULL,
   [start_time] [datetime] NULL,
   [end_time] [datetime] NULL,
   [quantity_rows] [int] NULL,
   [status] [varchar](9) NULL,
   [error_description] [varchar](1024) NULL,
 CONSTRAINT [PK_detail_process_log] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [monitor].[detail_process_log] ADD  DEFAULT (getdate()) FOR [start_time]
GO

ALTER TABLE [monitor].[detail_process_log]  WITH CHECK ADD  CONSTRAINT [FK_detail_process_log_master_process_log_id] FOREIGN
      KEY([master_process_log_id]) REFERENCES [monitor].[master_process_log] ([id])
GO
ALTER TABLE [monitor].[detail_process_log] CHECK CONSTRAINT [FK_detail_process_log_master_process_log_id]
GO
ALTER TABLE [monitor].[detail_process_log]  WITH CHECK ADD  CONSTRAINT [CK_detail_process_log_phase] CHECK  (([phase]='OUTPUT' OR
      [phase]='PERSISTENCE' OR [phase]='TRANSFORMATION' OR [phase]='DELIVERY' OR [phase]='INGESTION'))
GO
ALTER TABLE [monitor].[detail_process_log] CHECK CONSTRAINT [CK_detail_process_log_phase]
GO
ALTER TABLE [monitor].[detail_process_log]  WITH CHECK ADD  CONSTRAINT [CK_detail_process_log_status] CHECK  (([status]='WARNING' OR [status]='FAILED'
      OR [status]='SUCCEEDED'))
GO
ALTER TABLE [monitor].[detail_process_log] CHECK CONSTRAINT [CK_detail_process_log_status]
GO

-- Create [monitor].[log] Table
-- IO.monitor.log: tabla de log detallado de procesos internos de IO.
CREATE TABLE [monitor].[log](
   [id] [int] IDENTITY(1,1) NOT NULL,
   [created_at] [datetime] NULL,
   [user] [varchar](128) NULL,
   [source] [varchar](128) NULL,
   [pid] [smallint] NULL,
   [trace_id] [varchar](64) NULL,
   [message] [text] NULL,
   [severity] [varchar](64) NULL,
 CONSTRAINT [PK_log] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO


-- Create [monitor].[master_process_log] Table
-- IO.monitor.master_process_log: tabla de log general de la ejecución a nivel de proceso.
CREATE TABLE [monitor].[master_process_log](
   [id] [int] IDENTITY(1,1) NOT NULL,
   [master_process_type_id] [int] NULL,
   [master_process_start_time] [datetime] NULL,
   [master_process_end_time] [datetime] NULL,
   [master_process_status] [varchar](9) NULL,
   [ingestion_start_time] [datetime] NULL,
   [ingestion_end_time] [datetime] NULL,
   [ingestion_status] [varchar](9) NULL,
   [transformation_start_time] [datetime] NULL,
   [transformation_end_time] [datetime] NULL,
   [transformation_status] [varchar](9) NULL,
   [persistence_start_time] [datetime] NULL,
   [persistence_end_time] [datetime] NULL,
   [persistence_status] [varchar](9) NULL,
   [output_start_time] [datetime] NULL,
   [output_end_time] [datetime] NULL,
   [output_status] [varchar](9) NULL,
   [error_description] [varchar](1024) NULL,
 CONSTRAINT [PK_master_process_log] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [monitor].[master_process_log] ADD  DEFAULT (getdate()) FOR [master_process_start_time]
GO
ALTER TABLE [monitor].[master_process_log] ADD  DEFAULT ('N/A') FOR [ingestion_status]
GO
ALTER TABLE [monitor].[master_process_log] ADD  DEFAULT ('N/A') FOR [transformation_status]
GO
ALTER TABLE [monitor].[master_process_log] ADD  DEFAULT ('N/A') FOR [persistence_status]
GO
ALTER TABLE [monitor].[master_process_log] ADD  DEFAULT ('N/A') FOR [output_status]
GO

ALTER TABLE [monitor].[master_process_log]  WITH CHECK ADD CONSTRAINT [CK_master_process_log_ingest_status] CHECK (([ingestion_status]='WARNING' OR
      [ingestion_status]='FAILED' OR [ingestion_status]='SUCCEEDED' OR [ingestion_status]='N/A'))
```

```
GO
ALTER TABLE [monitor].[master_process_log] CHECK CONSTRAINT [CK_master_process_log_ingest_status]
GO
ALTER TABLE [monitor].[master_process_log]  WITH CHECK ADD CONSTRAINT [CK_master_process_log_master_process_status] CHECK
      (([master_process_status]='WARNING' OR [master_process_status]='FAILED' OR [master_process_status]='SUCCEEDED'))
GO
ALTER TABLE [monitor].[master_process_log] CHECK CONSTRAINT [CK_master_process_log_master_process_status]
GO
ALTER TABLE [monitor].[master_process_log]  WITH CHECK ADD CONSTRAINT [CK_master_process_log_output_status] CHECK (([output_status]='WARNING' OR
      [output_status]='FAILED' OR [output_status]='SUCCEEDED' OR [output_status]='N/A'))
GO
ALTER TABLE [monitor].[master_process_log] CHECK CONSTRAINT [CK_master_process_log_output_status]
GO
ALTER TABLE [monitor].[master_process_log]  WITH CHECK ADD CONSTRAINT [CK_master_process_log_persist_status] CHECK (([persistence_status]='WARNING' OR
      [persistence_status]='FAILED' OR [persistence_status]='SUCCEEDED' OR [persistence_status]='N/A'))
GO
ALTER TABLE [monitor].[master_process_log] CHECK CONSTRAINT [CK_master_process_log_persist_status]
GO
ALTER TABLE [monitor].[master_process_log]  WITH CHECK ADD CONSTRAINT [CK_master_process_log_transform_status] CHECK
      (([transformation_status]='WARNING' OR [transformation_status]='FAILED' OR [transformation_status]='SUCCEEDED' OR [transformation_status]='N/A'))
GO
ALTER TABLE [monitor].[master_process_log] CHECK CONSTRAINT [CK_master_process_log_transform_status]
GO


-- ============================================================
-- ================== CREACION DATOS ==========================
-- ============================================================

BEGIN TRAN

/*
TRUNCATE TABLE [IO].config.master_process_type
TRUNCATE TABLE [IO].config.detail_process_type
TRUNCATE TABLE [IO].config.customer_cluster_type
TRUNCATE TABLE [IO].config.customer_persistence_destination
TRUNCATE TABLE [IO].load.operation_type
TRUNCATE TABLE [IO].load.operation
DELETE FROM [IO].load.delivery DBCC CHECKIDENT ('IO.load.delivery',RESEED, 0)
TRUNCATE TABLE [IO].monitor.log
TRUNCATE TABLE [IO].monitor.detail_process_log
DELETE FROM [IO].monitor.master_process_log DBCC CHECKIDENT ('IO.monitor.master_process_log',RESEED, 0)
*/

-- [IO].config.master_process_type
-- Inserta la configuración a nivel maestro para anular otros procesos
INSERT INTO [IO].[config].[master_process_type]
   ([id],[customer],[name],[description],[active_flag],[ingestion_flag],[transformation_flag],[persist_flag],[output_flag])
VALUES
    (0, NULL, 'Deshabilitar / No usar', 'Master process usado para anular pasos de detail_process_type en caso de que sea necesario', 0, 0, 0, 0, 0)

-- [IO].config.customer_cluster_type
-- Inserta las agrupaciones por clientes (subdivisiones)
INSERT INTO [IO].config.customer_cluster_type (customer_id, name, cluster_id, cluster_name) VALUES (1, 'BMW', 1, 'BMW_Group')
INSERT INTO [IO].config.customer_cluster_type (customer_id, name, cluster_id, cluster_name) VALUES (2, 'Mini', 1, 'BMW_Group')

-- [IO].config.customer_persistence_destination
-- Insertar los destinos segun el cliente
INSERT INTO [IO].config.customer_persistence_destination (customer_id, destination_db) VALUES (1, 'VehicleSales')
INSERT INTO [IO].config.customer_persistence_destination (customer_id, destination_db) VALUES (2, 'VehicleSales')

-- [load].operation_type
-- Insertar tipos de operaciones
INSERT INTO [IO].[load].[operation_type] (id, operation_type_name) VALUES (1, 'Insert')
INSERT INTO [IO].[load].[operation_type] (id, operation_type_name) VALUES (2, 'Update')
INSERT INTO [IO].[load].[operation_type] (id, operation_type_name) VALUES (3, 'Close')
INSERT INTO [IO].[load].[operation_type] (id, operation_type_name) VALUES (4, 'Delete')
INSERT INTO [IO].[load].[operation_type] (id, operation_type_name) VALUES (5, 'Read')

/*
SELECT * FROM [IO].config.master_process_type
SELECT * FROM [IO].config.detail_process_type
SELECT * FROM [IO].config.customer_cluster_type
SELECT * FROM [IO].config.customer_persistence_destination
SELECT * FROM [IO].load.operation_type
SELECT * FROM [IO].load.operation
SELECT * FROM [IO].load.delivery
SELECT * FROM [IO].monitor.[log]
SELECT * FROM [IO].monitor.detail_process_log
SELECT * FROM [IO].monitor.master_process_log
*/

-- COMMIT
-- ROLLBACK
```

A continuación se muestra el *Código C.11* SQL para la creación de jobs de los procesos IO.

Código C.11: 1.1 Jobs ejecucion TC.sql

```
USE [msdb]
GO

/****** Object:  Job [IO - BMW - Input - NewDealers]    Script Date: 03/04/2024 13:32:02 ******/
BEGIN TRANSACTION
DECLARE @ReturnCode INT
```

```
SELECT @ReturnCode = 0
/****** Object:  JobCategory [[Uncategorized (Local)]]    Script Date: 03/04/2024 13:32:02 ******/
IF NOT EXISTS (SELECT name FROM msdb.dbo.syscategories WHERE name=N'[Uncategorized (Local)]' AND category_class=1)
BEGIN
EXEC @ReturnCode = msdb.dbo.sp_add_category @class=N'JOB', @type=N'LOCAL', @name=N'[Uncategorized (Local)]'
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback

END

DECLARE @jobId BINARY(16)
EXEC @ReturnCode =  msdb.dbo.sp_add_job @job_name=N'IO - BMW - Input - NewDealers',
      @enabled=1,
      @notify_level_eventlog=0,
      @notify_level_email=0,
      @notify_level_netsend=0,
      @notify_level_page=0,
      @delete_level=0,
      @description=N'IO - Proceso 1 Input - NewDealers csv',
      @category_name=N'[Uncategorized (Local)]',
      @owner_login_name=N'sa', @job_id = @jobId OUTPUT
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
/****** Object:  Step [EXEC - Main_process.dtsx]    Script Date: 03/04/2024 13:32:03 ******/
EXEC @ReturnCode = msdb.dbo.sp_add_jobstep @job_id=@jobId, @step_name=N'EXEC - Main_process.dtsx',
      @step_id=1,
      @cmdexec_success_code=0,
      @on_success_action=1,
      @on_success_step_id=0,
      @on_fail_action=2,
      @on_fail_step_id=0,
      @retry_attempts=0,
      @retry_interval=0,
      @os_run_priority=0, @subsystem=N'SSIS',
      @command=N'/ISSERVER "\"\SSISDB\Data\IO\Main_process.dtsx\"" /SERVER "\"WIN-SER-AMAROTO\"" /ENVREFERENCE 1 /Par "\"master_process_id(Int32)\"";1
       /Par "\"$ServerOption::LOGGING_LEVEL(Int16)\"";2 /Par "\"$ServerOption::SYNCHRONIZED(Boolean)\"";True /CALLERINFO SQLAGENT /REPORTING E',
      @database_name=N'master',
      @flags=0
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
EXEC @ReturnCode = msdb.dbo.sp_update_job @job_id = @jobId, @start_step_id = 1
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
EXEC @ReturnCode = msdb.dbo.sp_add_jobserver @job_id = @jobId, @server_name = N'(local)'
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
COMMIT TRANSACTION
GOTO EndSave
QuitWithRollback:
    IF (@@TRANCOUNT > 0) ROLLBACK TRANSACTION
EndSave:
GO


/****** Object:  Job [IO - BMW - Output - TopSellers]    Script Date: 03/04/2024 13:33:42 ******/
BEGIN TRANSACTION
DECLARE @ReturnCode INT
SELECT @ReturnCode = 0
/****** Object:  JobCategory [[Uncategorized (Local)]]    Script Date: 03/04/2024 13:33:42 ******/
IF NOT EXISTS (SELECT name FROM msdb.dbo.syscategories WHERE name=N'[Uncategorized (Local)]' AND category_class=1)
BEGIN
EXEC @ReturnCode = msdb.dbo.sp_add_category @class=N'JOB', @type=N'LOCAL', @name=N'[Uncategorized (Local)]'
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback

END

DECLARE @jobId BINARY(16)
EXEC @ReturnCode =  msdb.dbo.sp_add_job @job_name=N'IO - BMW - Output - TopSellers',
      @enabled=1,
      @notify_level_eventlog=0,
      @notify_level_email=0,
      @notify_level_netsend=0,
      @notify_level_page=0,
      @delete_level=0,
      @description=N'IO - Proceso 2 Output - TopSellers xlsx',
      @category_name=N'[Uncategorized (Local)]',
      @owner_login_name=N'DATA\Administrador', @job_id = @jobId OUTPUT
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
/****** Object:  Step [EXEC Main_process.dtsx]    Script Date: 03/04/2024 13:33:42 ******/
EXEC @ReturnCode = msdb.dbo.sp_add_jobstep @job_id=@jobId, @step_name=N'EXEC Main_process.dtsx',
      @step_id=1,
      @cmdexec_success_code=0,
      @on_success_action=1,
      @on_success_step_id=0,
      @on_fail_action=2,
      @on_fail_step_id=0,
      @retry_attempts=0,
      @retry_interval=0,
      @os_run_priority=0, @subsystem=N'SSIS',
      @command=N'/ISSERVER "\"\SSISDB\Data\IO\Main_process.dtsx\"" /SERVER "\"WIN-SER-AMAROTO\"" /ENVREFERENCE 1 /Par "\"master_process_id(Int32)\"";2
       /Par "\"$ServerOption::LOGGING_LEVEL(Int16)\"";2 /Par "\"$ServerOption::SYNCHRONIZED(Boolean)\"";True /CALLERINFO SQLAGENT /REPORTING E',
      @database_name=N'master',
      @flags=0
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
EXEC @ReturnCode = msdb.dbo.sp_update_job @job_id = @jobId, @start_step_id = 1
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
EXEC @ReturnCode = msdb.dbo.sp_add_jobserver @job_id = @jobId, @server_name = N'(local)'
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
COMMIT TRANSACTION
GOTO EndSave
QuitWithRollback:
    IF (@@TRANCOUNT > 0) ROLLBACK TRANSACTION
EndSave:
GO
```

```
104
105
106 /****** Object:  Job [IO - BMW - Output - MiniSales]    Script Date: 03/04/2024 13:33:16 ******/
107 BEGIN TRANSACTION
108 DECLARE @ReturnCode INT
109 SELECT @ReturnCode = 0
110 /****** Object:  JobCategory [[Uncategorized (Local)]]    Script Date: 03/04/2024 13:33:16 ******/
111 IF NOT EXISTS (SELECT name FROM msdb.dbo.syscategories WHERE name=N'[Uncategorized (Local)]' AND category_class=1)
112 BEGIN
113 EXEC @ReturnCode = msdb.dbo.sp_add_category @class=N'JOB', @type=N'LOCAL', @name=N'[Uncategorized (Local)]'
114 IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
115
116 END
117
118 DECLARE @jobId BINARY(16)
119 EXEC @ReturnCode =  msdb.dbo.sp_add_job @job_name=N'IO - BMW - Output - MiniSales',
120       @enabled=1,
121       @notify_level_eventlog=0,
122       @notify_level_email=0,
123       @notify_level_netsend=0,
124       @notify_level_page=0,
125       @delete_level=0,
126       @description=N'IO - Proceso 3 Output - MiniSales txt',
127       @category_name=N'[Uncategorized (Local)]',
128       @owner_login_name=N'sa', @job_id = @jobId OUTPUT
129 IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
130 /****** Object:  Step [EXEC - Main_process.dtsx]    Script Date: 03/04/2024 13:33:16 ******/
131 EXEC @ReturnCode = msdb.dbo.sp_add_jobstep @job_id=@jobId, @step_name=N'EXEC - Main_process.dtsx',
132       @step_id=1,
133       @cmdexec_success_code=0,
134       @on_success_action=1,
135       @on_success_step_id=0,
136       @on_fail_action=2,
137       @on_fail_step_id=0,
138       @retry_attempts=0,
139       @retry_interval=0,
140       @os_run_priority=0, @subsystem=N'SSIS',
141       @command=N'/ISSERVER "\"\SSISDB\Data\IO\Main_process.dtsx\"" /SERVER "\"WIN-SER-AMAROTO\"" /ENVREFERENCE 1 /Par "\"master_process_id(Int32)\"";3
       /Par "\"$ServerOption::LOGGING_LEVEL(Int16)\"";2 /Par "\"$ServerOption::SYNCHRONIZED(Boolean)\"";True /CALLERINFO SQLAGENT /REPORTING E',
142       @database_name=N'master',
143       @flags=0
144 IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
145 EXEC @ReturnCode = msdb.dbo.sp_update_job @job_id = @jobId, @start_step_id = 1
146 IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
147 EXEC @ReturnCode = msdb.dbo.sp_add_jobserver @job_id = @jobId, @server_name = N'(local)'
148 IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
149 COMMIT TRANSACTION
150 GOTO EndSave
151 QuitWithRollback:
152     IF (@@TRANCOUNT > 0) ROLLBACK TRANSACTION
153 EndSave:
154 GO
155
156
157 /****** Object:  Job [IO - BMW - Input - RemoteConnections]    Script Date: 03/04/2024 13:32:38 ******/
158 BEGIN TRANSACTION
159 DECLARE @ReturnCode INT
160 SELECT @ReturnCode = 0
161 /****** Object:  JobCategory [[Uncategorized (Local)]]    Script Date: 03/04/2024 13:32:38 ******/
162 IF NOT EXISTS (SELECT name FROM msdb.dbo.syscategories WHERE name=N'[Uncategorized (Local)]' AND category_class=1)
163 BEGIN
164 EXEC @ReturnCode = msdb.dbo.sp_add_category @class=N'JOB', @type=N'LOCAL', @name=N'[Uncategorized (Local)]'
165 IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
166
167 END
168
169 DECLARE @jobId BINARY(16)
170 EXEC @ReturnCode =  msdb.dbo.sp_add_job @job_name=N'IO - BMW - Input - RemoteConnections',
171       @enabled=1,
172       @notify_level_eventlog=0,
173       @notify_level_email=0,
174       @notify_level_netsend=0,
175       @notify_level_page=0,
176       @delete_level=0,
177       @description=N'IO - Proceso 4 Input - RemoteConnections BBDD',
178       @category_name=N'[Uncategorized (Local)]',
179       @owner_login_name=N'DATA\Administrador', @job_id = @jobId OUTPUT
180 IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
181 /****** Object:  Step [EXEC Main_process.dtsx]    Script Date: 03/04/2024 13:32:38 ******/
182 EXEC @ReturnCode = msdb.dbo.sp_add_jobstep @job_id=@jobId, @step_name=N'EXEC Main_process.dtsx',
183       @step_id=1,
184       @cmdexec_success_code=0,
185       @on_success_action=1,
186       @on_success_step_id=0,
187       @on_fail_action=2,
188       @on_fail_step_id=0,
189       @retry_attempts=0,
190       @retry_interval=0,
191       @os_run_priority=0, @subsystem=N'SSIS',
192       @command=N'/ISSERVER "\"\SSISDB\Data\IO\Main_process.dtsx\"" /SERVER "\"WIN-SER-AMAROTO\"" /ENVREFERENCE 1 /Par "\"master_process_id(Int32)\"";4
       /Par "\"$ServerOption::LOGGING_LEVEL(Int16)\"";2 /Par "\"$ServerOption::SYNCHRONIZED(Boolean)\"";True /CALLERINFO SQLAGENT /REPORTING E',
193       @database_name=N'master',
194       @flags=0
195 IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
196 EXEC @ReturnCode = msdb.dbo.sp_update_job @job_id = @jobId, @start_step_id = 1
197 IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
198 EXEC @ReturnCode = msdb.dbo.sp_add_jobserver @job_id = @jobId, @server_name = N'(local)'
199 IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
200 COMMIT TRANSACTION
```

```
GOTO EndSave
QuitWithRollback:
    IF (@@TRANCOUNT > 0) ROLLBACK TRANSACTION
EndSave:
GO


/****** Object:  Job [IO - BMW - Input - StatusUpdate]    Script Date: 03/04/2024 13:32:47 ******/
BEGIN TRANSACTION
DECLARE @ReturnCode INT
SELECT @ReturnCode = 0
/****** Object:  JobCategory [[Uncategorized (Local)]]    Script Date: 03/04/2024 13:32:47 ******/
IF NOT EXISTS (SELECT name FROM msdb.dbo.syscategories WHERE name=N'[Uncategorized (Local)]' AND category_class=1)
BEGIN
EXEC @ReturnCode = msdb.dbo.sp_add_category @class=N'JOB', @type=N'LOCAL', @name=N'[Uncategorized (Local)]'
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback

END

DECLARE @jobId BINARY(16)
EXEC @ReturnCode =  msdb.dbo.sp_add_job @job_name=N'IO - BMW - Input - StatusUpdate',
      @enabled=1,
      @notify_level_eventlog=0,
      @notify_level_email=0,
      @notify_level_netsend=0,
      @notify_level_page=0,
      @delete_level=0,
      @description=N'IO - Proceso 5 Input - StatusUpdate xlsx',
      @category_name=N'[Uncategorized (Local)]',
      @owner_login_name=N'sa', @job_id = @jobId OUTPUT
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
/****** Object:  Step [EXEC - Main_process.dtsx]    Script Date: 03/04/2024 13:32:47 ******/
EXEC @ReturnCode = msdb.dbo.sp_add_jobstep @job_id=@jobId, @step_name=N'EXEC - Main_process.dtsx',
      @step_id=1,
      @cmdexec_success_code=0,
      @on_success_action=1,
      @on_success_step_id=0,
      @on_fail_action=2,
      @on_fail_step_id=0,
      @retry_attempts=0,
      @retry_interval=0,
      @os_run_priority=0, @subsystem=N'SSIS',
      @command=N'/ISSERVER "\"\SSISDB\Data\IO\Main_process.dtsx\"" /SERVER "\"WIN-SER-AMAROTO\"" /ENVREFERENCE 1 /Par "\"master_process_id(Int32)\"";5
       /Par "\"$ServerOption::LOGGING_LEVEL(Int16)\"";2 /Par "\"$ServerOption::SYNCHRONIZED(Boolean)\"";True /CALLERINFO SQLAGENT /REPORTING E',
      @database_name=N'master',
      @flags=0
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
EXEC @ReturnCode = msdb.dbo.sp_update_job @job_id = @jobId, @start_step_id = 1
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
EXEC @ReturnCode = msdb.dbo.sp_add_jobserver @job_id = @jobId, @server_name = N'(local)'
IF (@@ERROR <> 0 OR @ReturnCode <> 0) GOTO QuitWithRollback
COMMIT TRANSACTION
GOTO EndSave
QuitWithRollback:
    IF (@@TRANCOUNT > 0) ROLLBACK TRANSACTION
EndSave:
GO
```

A continuación se muestra el *Código C.12* SQL para la creación de encriptado de la BBDD IO.

Código C.12: 1.2 Encriptado BBDD IO.sql

```
-- ================================ AUXILIARES ================================
-- ============================================================================
SELECT * FROM sys.symmetric_keys --WHERE name = '##MS_ServiceMasterKey##';
SELECT * FROM sys.certificates --WHERE name = 'Temis_certificate';

-- 1. Lo primero que se ha realizado es la creación de una master key para la encriptación de la base de datos de IO:
USE IO
IF @@SERVERNAME = 'WIN-SER-AMAROTO'
BEGIN
   CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'jEmY6RjEuaSSsmZf3yAp'
END
--DROP MASTER KEY

-- 2. Posteriormente se ha generado un nuevo certificado que será utilizado durante el proceso de encriptación.
CREATE CERTIFICATE io_db_certificate WITH SUBJECT = 'Certificado para el encriptado a nivel de base de datos para IO'
GO
--DROP CERTIFICATE io_db_certificate

--3. A continuación se ha creado una clave simétrica basada en el certificado anterior, es en base a esta clave en la que realizará la
-- encriptación de cada uno de los passwords almacenados en las tablas de configuración de IO
-- Certificado creado con las siguientes instrucciones:
CREATE SYMMETRIC KEY password_io_simetric_key WITH
IDENTITY_VALUE = 'password_io_simetric_key',
ALGORITHM = AES_256,
KEY_SOURCE = 'Usa la contraseña de encriptado de master key'
ENCRYPTION BY CERTIFICATE io_db_certificate;
GO

-- DROP SYMMETRIC KEY password_io_simetric_key

-- Apertura de las claves de cifrado, siempre se ha de hacer antes de encriptar o desencriptar
EXEC [IO].config.sp_open_io_password_encryption_keys
-- Funcion de encriptado, entrada VARCHAR, salida VARBINBARY
```

```
SELECT [IO].config.f_password_encryption('P4$$W0rd_unico')
-- Función de desencriptado, entrada VARBINARY, salida VARCHAR
SELECT [IO].config.f_password_decryption(0x00BA1DC5F52ACF05CEF7C5F899A696B302000000B16...)
```

A continuación se muestra el *Código C.13* SQL para la creación de la BBDD *VehicleSales*.

Código C.13: 2 Creacion BBDD VehicleSales.sql

```
/*
-- ============================================================
-- ==================== DESCRIPCION ===========================
-- ============================================================
1. brand_type:
   . Tabla para almacenar información sobre las marcas de vehículos.
      - id (PK, int): Identificador único de la marca.
      - name (NVARCHAR): Nombre de la marca de vehículos.
      - active (bit): Indica si el registro está o no activo.
      - creation_date (DF, datetime): Fecha de creación del registro generada por default.
      - modification_date (TR, datetime): Fecha de modificación del registro actualizada por trigger.
      - io_id (int): Clave de la operación realizada en la BBDD IO.
2. fuel_type:
   . Tabla para almacenar información sobre el tipo de combustible del modelo.
      - id (PK, int): Identificador único del combustible.
      - name (NVARCHAR): Tipo de combustible del vehículo (gasolina, diésel, eléctrico, híbrido).
      - active (bit): Indica si el registro está o no activo.
      - creation_date (DF, datetime): Fecha de creación del registro generada por default.
      - modification_date (TR, datetime): Fecha de modificación del registro actualizada por trigger.
      - io_id (int): Clave de la operación realizada en la BBDD IO.
3. model_type:
   . Tabla para almacenar información sobre los modelos de vehículos.
      - id (PK, int): Identificador único del modelo.
      - model_name (NVARCHAR): Nombre del modelo de vehículo.
      - fuel_type_id (FK, int): Clave foránea que referencia la tabla fuel_type.
      - brand_id (FK, int): Clave foránea que referencia la tabla concessionare.
      - active (bit): Indica si el registro está o no activo.
      - creation_date (DF, datetime): Fecha de creación del registro generada por default.
      - modification_date (TR, datetime): Fecha de modificación del registro actualizada por trigger.
      - io_id (int): Clave de la operación realizada en la BBDD IO.
4. proposal_type:
   . Tabla para almacenar información sobre el tipo de propuesta realizada.
      - id (PK, int): Identificador único del tipo de propuesta.
      - name (NVARCHAR): Tipo de propuesta realizada (nuevo, seminuevo, renting, leasing).
      - active (bit): Indica si el registro está o no activo.
      - creation_date (DF, datetime): Fecha de creación del registro generada por default.
      - modification_date (TR, datetime): Fecha de modificación del registro actualizada por trigger.
      - io_id (int): Clave de la operación realizada en la BBDD IO.
5. concessionare:
   . Tabla para almacenar información sobre los concesionarios de vehículos.
      - id (PK, int): Identificador único del concesionario.
      - name (NVARCHAR): Nombre del concesionario.
      - address (NVARCHAR): Dirección del concesionario.
      - city (NVARCHAR): Ciudad donde se encuentra el concesionario.
      - country (NVARCHAR): País donde se encuentra el concesionario.
      - postal_code (NVARCHAR): Código postal donde se encuentra el concesionario.
      - active (bit): Indica si el registro está o no activo.
      - creation_date (DF, datetime): Fecha de creación del registro generada por default.
      - modification_date (TR, datetime): Fecha de modificación del registro actualizada por trigger.
      - io_id (int): Clave de la operación realizada en la BBDD IO.
6. customer:
   . Tabla para almacenar información sobre los clientes.
      - id (PK, int): Identificador único del cliente.
      - name (NVARCHAR): Nombre del cliente.
      - phone (NVARCHAR): Teléfono del cliente.
      - email (NVARCHAR): Dirección de correo electrónico del cliente.
      - creation_date (DF, datetime): Fecha de creación del registro generada por default.
      - modification_date (TR, datetime): Fecha de modificación del registro actualizada por trigger.
      - io_id (int): Clave de la operación realizada en la BBDD IO.
7. seller:
   . Tabla para almacenar información sobre los vendedores de los concesionarios.
      - id (PK, int): Identificador único del vendedor.
      - name (NVARCHAR): Nombre del vendedor.
      - phone (NVARCHAR): Teléfono del vendedor.
      - email (NVARCHAR): Dirección de correo electrónico del vendedor.
      - concessionaire_id (FK, int): Clave foránea que referencia la tabla concessionaire.
      - active (bit): Indica si el registro está o no activo.
      - creation_date (DF, datetime): Fecha de creación del registro generada por default.
      - modification_date (TR, datetime): Fecha de modificación del registro actualizada por trigger.
      - io_id (int): Clave de la operación realizada en la BBDD IO.
8. proposal:
   . Tabla para almacenar información sobre las propuestas realizadas a los clientes.
      - id (PK, int): Identificador único de la propuesta.
      - customer_id (FK, int): Clave foránea que referencia la tabla customer.
      - seller_id (FK, int): Clave foránea que referencia la tabla seller.
      - model_type_id (FK, int): Clave foránea que referencia la tabla model_type.
      - proposal_date (date): Fecha en que se realizó la propuesta.
      - proposal_amount (decimal): Precio de la propuesta.
      - proposal_type_id (FK, int): Clave foránea que referencia la tabla proposal_type.
      - active (bit): Indica si el registro está o no activo.
      - creation_date (DF, datetime): Fecha de creación del registro generada por default.
      - modification_date (TR, datetime): Fecha de modificación del registro actualizada por trigger.
      - io_id (int): Clave de la operación realizada en la BBDD IO.
9. offer:
   . Tabla para almacenar información sobre las ofertas realizadas a los clientes basadas en las propuestas.
      - id (PK, int): Identificador único de la oferta.
      - proposal_id (FK, int): Clave foránea que referencia la tabla proposal.
```

```
      - offer_date (date): Fecha en que se realizó la oferta.
      - offer_amount (decimal): Precio de la oferta.
      - active (bit): Indica si el registro está o no activo.
      - creation_date (DF, datetime): Fecha de creación del registro generada por default.
      - modification_date (TR, datetime): Fecha de modificación del registro actualizada por trigger.
      - io_id (int): Clave de la operación realizada en la BBDD IO.
10. sale:
   . Tabla para almacenar información sobre las ventas realizadas a los clientes basadas en las ofertas.
      - id (PK, int): Identificador único de la venta.
      - offer_id (FK, int): Clave foránea que referencia la tabla offer.
      - sale_date (date): Fecha en que se realizó la venta.
      - creation_date (DF, datetime): Fecha de creación del registro generada por default.
      - modification_date (TR, datetime): Fecha de modificación del registro actualizada por trigger.
      - io_id (int): Clave de la operación realizada en la BBDD IO.
*/

-- ============================================================
-- ==================== CREACION BBDD =========================
-- ============================================================
-- Create VehicleSales Database
CREATE DATABASE VehicleSales;
GO

-- Use VehicleSales Database
USE VehicleSales;
GO

/*
USE Master;
ALTER DATABASE VehicleSales SET single_user with rollback immediate;
ALTER DATABASE VehicleSales SET MULTI_USER;
DROP DATABASE VehicleSales;
*/

-- ============================================================
-- ================== CREACION TABLAS =========================
-- ============================================================
-- Create brand_type Table
CREATE TABLE [dbo].[brand_type] (
    id INT IDENTITY(1, 1),
    [name] NVARCHAR(64),
   active BIT,
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_brand_type] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[brand_type] ADD DEFAULT (1) FOR [active]
GO

ALTER TABLE [dbo].[brand_type] ADD CONSTRAINT [df_brand_type_creation_date]  DEFAULT (getdate()) FOR [creation_date]
GO

-- Create fuel_type Table
CREATE TABLE [dbo].[fuel_type] (
    id INT IDENTITY(1, 1),
    [name] NVARCHAR(64),
   active BIT,
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_fuel_type] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[fuel_type] ADD DEFAULT (1) FOR [active]
GO

ALTER TABLE [dbo].[fuel_type] ADD CONSTRAINT [df_fuel_type_creation_date]  DEFAULT (getdate()) FOR [creation_date]
GO

-- Create model_type Table
CREATE TABLE [dbo].[model_type] (
    id INT IDENTITY(1, 1),
    [name] NVARCHAR(128),
    brand_type_id INT,
   fuel_type_id INT,
   active BIT,
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_model_type] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

```

```
ALTER TABLE [dbo].[model_type] ADD DEFAULT (1) FOR [active]
GO

ALTER TABLE [dbo].[model_type] ADD  CONSTRAINT [df_model_type_creation_date]  DEFAULT (getdate()) FOR [creation_date]
GO

ALTER TABLE [dbo].[model_type] WITH CHECK ADD  CONSTRAINT [fk_model_type_brand_type_id] FOREIGN KEY([brand_type_id])
REFERENCES [dbo].[brand_type] ([id])
GO
ALTER TABLE [dbo].[model_type] CHECK CONSTRAINT [fk_model_type_brand_type_id]
GO

ALTER TABLE [dbo].[model_type] WITH CHECK ADD  CONSTRAINT [fk_model_type_fuel_type_id] FOREIGN KEY([fuel_type_id])
REFERENCES [dbo].[fuel_type] ([id])
GO
ALTER TABLE [dbo].[model_type] CHECK CONSTRAINT [fk_model_type_brand_type_id]
GO

-- Create proposal_type Table
CREATE TABLE [dbo].[proposal_type] (
    id INT IDENTITY(1, 1),
    [name] NVARCHAR(128),
   active BIT,
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_sale_type] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[proposal_type] ADD DEFAULT (1) FOR [active]
GO

ALTER TABLE [dbo].[proposal_type] ADD  CONSTRAINT [df_proposal_type_creation_date]  DEFAULT (getdate()) FOR [creation_date]
GO

-- Create concessionaire Table
CREATE TABLE [dbo].[concessionaire] (
    id INT IDENTITY(1, 1),
    [name] NVARCHAR(128),
    [address] NVARCHAR (512),
   city NVARCHAR(64),
    country NVARCHAR(32),
   postal_code NVARCHAR(18),
   info NVARCHAR(512),
   active BIT,
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_concessionaire] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[concessionaire] ADD DEFAULT (1) FOR [active]
GO

ALTER TABLE [dbo].[concessionaire] ADD  CONSTRAINT [df_concessionaire_creation_date]  DEFAULT (getdate()) FOR [creation_date]
GO

-- Create customer Table
CREATE TABLE [dbo].[customer] (
    id INT IDENTITY(1, 1),
    [name] NVARCHAR(128),
   phone NVARCHAR(18),
    email NVARCHAR(256),
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_customer] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[customer] ADD  CONSTRAINT [df_customer_creation_date]  DEFAULT (getdate()) FOR [creation_date]
GO

-- Create seller Table
CREATE TABLE [dbo].[seller] (
    id INT IDENTITY(1, 1),
    [name] NVARCHAR(128),
   phone NVARCHAR(18),
    email NVARCHAR(256),
    concessionaire_id INT,
   active BIT,
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_seller] PRIMARY KEY CLUSTERED
```

```
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[seller] ADD DEFAULT (1) FOR [active]
GO

ALTER TABLE [dbo].[seller] ADD  CONSTRAINT [df_seller_creation_date]  DEFAULT (getdate()) FOR [creation_date]
GO

ALTER TABLE [dbo].[seller] WITH CHECK ADD  CONSTRAINT [fk_seller_concessionaire_id] FOREIGN KEY([concessionaire_id])
REFERENCES [dbo].[concessionaire] ([id])
GO
ALTER TABLE [dbo].[seller] CHECK CONSTRAINT [fk_seller_concessionaire_id]
GO

-- Create proposal Table
CREATE TABLE [dbo].[proposal] (
    id INT IDENTITY(1, 1),
    customer_id INT,
    seller_id INT,
   model_type_id INT,
    proposal_date DATE,
   proposal_amount DECIMAL(10, 2),
   proposal_type_id INT,
   active BIT,
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_proposal] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[proposal] ADD DEFAULT (1) FOR [active]
GO

ALTER TABLE [dbo].[proposal] ADD  CONSTRAINT [df_prposal_creation_date]  DEFAULT (getdate()) FOR [creation_date]
GO

ALTER TABLE [dbo].[proposal] WITH CHECK ADD  CONSTRAINT [fk_proposal_customer_id] FOREIGN KEY([customer_id])
REFERENCES [dbo].[customer] ([id])
GO
ALTER TABLE [dbo].[proposal] CHECK CONSTRAINT [fk_proposal_customer_id]
GO

ALTER TABLE [dbo].[proposal] WITH CHECK ADD  CONSTRAINT [fk_proposal_seller_id] FOREIGN KEY([seller_id])
REFERENCES [dbo].[seller] ([id])
GO
ALTER TABLE [dbo].[proposal] CHECK CONSTRAINT [fk_proposal_seller_id]
GO

ALTER TABLE [dbo].[proposal] WITH CHECK ADD  CONSTRAINT [fk_proposal_model_type_id] FOREIGN KEY([model_type_id])
REFERENCES [dbo].[model_type] ([id])
GO
ALTER TABLE [dbo].[proposal] CHECK CONSTRAINT [fk_proposal_model_type_id]
GO

ALTER TABLE [dbo].[proposal] WITH CHECK ADD  CONSTRAINT [fk_proposal_proposal_type_id] FOREIGN KEY([proposal_type_id])
REFERENCES [dbo].[proposal_type] ([id])
GO
ALTER TABLE [dbo].[proposal] CHECK CONSTRAINT [fk_proposal_proposal_type_id]
GO

-- Create offer Table
CREATE TABLE [dbo].[offer] (
    id INT IDENTITY(1, 1),
    proposal_id INT,
    offer_date DATE,
    offer_amount DECIMAL(10, 2),
   active BIT,
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_offer] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[offer] ADD DEFAULT (1) FOR [active]
GO

ALTER TABLE [dbo].[offer] ADD CONSTRAINT [df_offer_creation_date] DEFAULT (getdate()) FOR [creation_date]
GO

ALTER TABLE [dbo].[offer] WITH CHECK ADD CONSTRAINT [fk_offer_proposal_id] FOREIGN KEY([proposal_id])
REFERENCES [dbo].[proposal] ([id])
GO
ALTER TABLE [dbo].[offer] CHECK CONSTRAINT [fk_offer_proposal_id]
GO
```

```
376
377 -- Create sale Table
378 CREATE TABLE [dbo].[sale] (
379     id INT IDENTITY(1, 1),
380     offer_id INT,
381     sale_date DATE,
382    creation_date DATETIME,
383    modification_date DATETIME,
384    io_id INT,
385 CONSTRAINT [pk_sale] PRIMARY KEY CLUSTERED
386 (
387    [id] ASC
388 )WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
        OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
389 ) ON [PRIMARY]
390 GO
391
392 ALTER TABLE [dbo].[sale] ADD CONSTRAINT [df_sale_creation_date] DEFAULT (getdate()) FOR [creation_date]
393 GO
394
395 ALTER TABLE [dbo].[sale] WITH CHECK ADD CONSTRAINT [fk_sale_offer_id] FOREIGN KEY([offer_id])
396 REFERENCES [dbo].[offer] ([id])
397 GO
398 ALTER TABLE [dbo].[sale] CHECK CONSTRAINT [fk_sale_offer_id]
399 GO
400
401 -- ============================================================
402 -- ================== CREACION TRIGGERS =======================
403 -- ============================================================
404
405 -- Create brand_type Trigger
406 SET ANSI_NULLS ON
407 GO
408
409 SET QUOTED_IDENTIFIER ON
410 GO
411
412 CREATE TRIGGER [dbo].[tr_data_update_brand_type] ON [dbo].[brand_type]
413    AFTER UPDATE
414    AS
415    BEGIN
416       -- Continua el trigger SOLO si hay resultados
417       IF (@@ROWCOUNT = 0) RETURN;
418
419       -- Ejecuta el trigger una sola vez (evita la recursividad)
420       IF (TRIGGER_NESTLEVEL() > 1) RETURN;
421
422       UPDATE dbo.brand_type SET
423          modification_date = GETDATE()
424       WHERE id IN (SELECT id FROM inserted)
425    END
426 GO
427
428 ALTER TABLE [dbo].[brand_type] ENABLE TRIGGER [tr_data_update_brand_type]
429 GO
430
431 -- Create fuel_type Trigger
432 SET ANSI_NULLS ON
433 GO
434
435 SET QUOTED_IDENTIFIER ON
436 GO
437
438 CREATE TRIGGER [dbo].[tr_data_update_fuel_type] ON [dbo].[fuel_type]
439    AFTER UPDATE
440    AS
441    BEGIN
442       -- Continua el trigger SOLO si hay resultados
443       IF (@@ROWCOUNT = 0) RETURN;
444
445       -- Ejecuta el trigger una sola vez (evita la recursividad)
446       IF (TRIGGER_NESTLEVEL() > 1) RETURN;
447
448       UPDATE dbo.fuel_type SET
449          modification_date = GETDATE()
450       WHERE id IN (SELECT id FROM inserted)
451    END
452 GO
453
454 ALTER TABLE [dbo].[fuel_type] ENABLE TRIGGER [tr_data_update_fuel_type]
455 GO
456
457 -- Create model_type Trigger
458 SET ANSI_NULLS ON
459 GO
460
461 SET QUOTED_IDENTIFIER ON
462 GO
463
464 CREATE TRIGGER [dbo].[tr_data_update_model_type] ON [dbo].[model_type]
465    AFTER UPDATE
466    AS
467    BEGIN
468       -- Continua el trigger SOLO si hay resultados
469       IF (@@ROWCOUNT = 0) RETURN;
470
471       -- Ejecuta el trigger una sola vez (evita la recursividad)
472       IF (TRIGGER_NESTLEVEL() > 1) RETURN;
473
```

```
474       UPDATE dbo.model_type SET
475          modification_date = GETDATE()
476       WHERE id IN (SELECT id FROM inserted)
477    END
478 GO
479
480 ALTER TABLE [dbo].[model_type] ENABLE TRIGGER [tr_data_update_model_type]
481 GO
482
483 -- Create proposal_type Trigger
484 SET ANSI_NULLS ON
485 GO
486
487 SET QUOTED_IDENTIFIER ON
488 GO
489
490 CREATE TRIGGER [dbo].[tr_data_update_proposal_type] ON [dbo].[proposal_type]
491    AFTER UPDATE
492    AS
493    BEGIN
494       -- Continua el trigger SOLO si hay resultados
495       IF (@@ROWCOUNT = 0) RETURN;
496
497       -- Ejecuta el trigger una sola vez (evita la recursividad)
498       IF (TRIGGER_NESTLEVEL() > 1) RETURN;
499
500       UPDATE dbo.proposal_type SET
501          modification_date = GETDATE()
502       WHERE id IN (SELECT id FROM inserted)
503    END
504 GO
505
506 ALTER TABLE [dbo].[proposal_type] ENABLE TRIGGER [tr_data_update_proposal_type]
507 GO
508
509 -- Create concessionare Trigger
510 SET ANSI_NULLS ON
511 GO
512
513 SET QUOTED_IDENTIFIER ON
514 GO
515
516 CREATE TRIGGER [dbo].[tr_data_update_concessionaire] ON [dbo].[concessionaire]
517    AFTER UPDATE
518    AS
519    BEGIN
520       -- Continua el trigger SOLO si hay resultados
521       IF (@@ROWCOUNT = 0) RETURN;
522
523       -- Ejecuta el trigger una sola vez (evita la recursividad)
524       IF (TRIGGER_NESTLEVEL() > 1) RETURN;
525
526       UPDATE dbo.concessionaire SET
527          modification_date = GETDATE()
528       WHERE id IN (SELECT id FROM inserted)
529    END
530 GO
531
532 ALTER TABLE [dbo].[concessionaire] ENABLE TRIGGER [tr_data_update_concessionaire]
533 GO
534
535 -- Create customer Trigger
536 SET ANSI_NULLS ON
537 GO
538
539 SET QUOTED_IDENTIFIER ON
540 GO
541
542 CREATE TRIGGER [dbo].[tr_data_update_customer] ON [dbo].[customer]
543    AFTER UPDATE
544    AS
545    BEGIN
546       -- Continua el trigger SOLO si hay resultados
547       IF (@@ROWCOUNT = 0) RETURN;
548
549       -- Ejecuta el trigger una sola vez (evita la recursividad)
550       IF (TRIGGER_NESTLEVEL() > 1) RETURN;
551
552       UPDATE dbo.customer SET
553          modification_date = GETDATE()
554       WHERE id IN (SELECT id FROM inserted)
555    END
556 GO
557
558 ALTER TABLE [dbo].[customer] ENABLE TRIGGER [tr_data_update_customer]
559 GO
560
561 -- Create seller Trigger
562 SET ANSI_NULLS ON
563 GO
564
565 SET QUOTED_IDENTIFIER ON
566 GO
567
568 CREATE TRIGGER [dbo].[tr_data_update_seller] ON [dbo].[seller]
569    AFTER UPDATE
570    AS
571    BEGIN
572       -- Continua el trigger SOLO si hay resultados
```

```
      IF (@@ROWCOUNT = 0) RETURN;

      -- Ejecuta el trigger una sola vez (evita la recursividad)
      IF (TRIGGER_NESTLEVEL() > 1) RETURN;

      UPDATE dbo.seller SET
         modification_date = GETDATE()
      WHERE id IN (SELECT id FROM inserted)
   END
GO

ALTER TABLE [dbo].[seller] ENABLE TRIGGER [tr_data_update_seller]
GO

-- Create proposal Trigger
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TRIGGER [dbo].[tr_data_update_proposal] ON [dbo].[proposal]
   AFTER UPDATE
   AS
   BEGIN
      -- Continua el trigger SOLO si hay resultados
      IF (@@ROWCOUNT = 0) RETURN;

      -- Ejecuta el trigger una sola vez (evita la recursividad)
      IF (TRIGGER_NESTLEVEL() > 1) RETURN;

      UPDATE dbo.proposal SET
         modification_date = GETDATE()
      WHERE id IN (SELECT id FROM inserted)
   END
GO

ALTER TABLE [dbo].[proposal] ENABLE TRIGGER [tr_data_update_proposal]
GO

-- Create offer Trigger
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TRIGGER [dbo].[tr_data_update_offer] ON [dbo].[offer]
   AFTER UPDATE
   AS
   BEGIN
      -- Continua el trigger SOLO si hay resultados
      IF (@@ROWCOUNT = 0) RETURN;

      -- Ejecuta el trigger una sola vez (evita la recursividad)
      IF (TRIGGER_NESTLEVEL() > 1) RETURN;

      UPDATE dbo.offer SET
         modification_date = GETDATE()
      WHERE id IN (SELECT id FROM inserted)
   END
GO

ALTER TABLE [dbo].[offer] ENABLE TRIGGER [tr_data_update_offer]
GO

-- Create sale Trigger
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO

CREATE TRIGGER [dbo].[tr_data_update_sale] ON [dbo].[sale]
   AFTER UPDATE
   AS
   BEGIN
      -- Continua el trigger SOLO si hay resultados
      IF (@@ROWCOUNT = 0) RETURN;

      -- Ejecuta el trigger una sola vez (evita la recursividad)
      IF (TRIGGER_NESTLEVEL() > 1) RETURN;

      UPDATE dbo.sale SET
         modification_date = GETDATE()
      WHERE id IN (SELECT id FROM inserted)
   END
GO

ALTER TABLE [dbo].[sale] ENABLE TRIGGER [tr_data_update_sale]
GO

-- ============================================================
-- ================== CREACION DATOS ==========================
-- ============================================================

BEGIN TRAN

/*
```

```
DELETE FROM [dbo].[sale] DBCC CHECKIDENT ('[dbo].[sale]',RESEED, 0)
DELETE FROM [dbo].[offer] DBCC CHECKIDENT ('[dbo].[offer]',RESEED, 0)
DELETE FROM [dbo].[proposal] DBCC CHECKIDENT ('[dbo].[proposal]',RESEED, 0)
DELETE FROM [dbo].[seller] DBCC CHECKIDENT ('[dbo].[seller]',RESEED, 0)
DELETE FROM [dbo].[customer] DBCC CHECKIDENT ('[dbo].[customer]',RESEED, 0)
DELETE FROM [dbo].[concessionaire] DBCC CHECKIDENT ('[dbo].[concessionaire]',RESEED, 0)
DELETE FROM [dbo].[proposal_type] DBCC CHECKIDENT ('[dbo].[proposal_type]',RESEED, 0)
DELETE FROM [dbo].[model_type] DBCC CHECKIDENT ('[dbo].[model_type]',RESEED, 0)
DELETE FROM [dbo].[fuel_type] DBCC CHECKIDENT ('[dbo].[fuel_type]',RESEED, 0)
DELETE FROM [dbo].[brand_type] DBCC CHECKIDENT ('[dbo].[brand_type]',RESEED, 0)
*/

-- [dbo].[brand_type]
-- Inserta BMW/Mini
INSERT INTO [dbo].[brand_type] ([name], active, creation_date)
VALUES ('BMW', 1, GETDATE()),
       ('Mini', 1, GETDATE());

-- [dbo].[fuel_type]
-- Insert tipos de combustible
INSERT INTO [dbo].[fuel_type] ([name], active, creation_date)
VALUES ('Gasolina', 1, GETDATE()),
       ('Diésel', 1, GETDATE()),
       ('Eléctrico', 1, GETDATE()),
       ('Híbrido', 1, GETDATE());

-- [dbo].[model_type]
-- Inserta modelos de BMW/Mini
INSERT INTO [dbo].[model_type] ([name], [brand_type_id], [fuel_type_id], active, creation_date)
VALUES ('Serie 1', 1, 1, 1, GETDATE()),
       ('Serie 2', 1, 4, 1, GETDATE()),
       ('Serie 3', 1, 1, 1, GETDATE()),
       ('Serie 4', 1, 2, 1, GETDATE()),
       ('Serie 5', 1, 2, 1, GETDATE()),
       ('Serie 6', 1, 1, 1, GETDATE()),
       ('Serie 7', 1, 1, 1, GETDATE()),
       ('Serie 8', 1, 1, 1, GETDATE()),
      ('I3', 1, 3, 1, GETDATE()),
      ('I8', 1, 3, 1, GETDATE()),
       ('X1', 1, 2, 1, GETDATE()),
       ('X2', 1, 2, 1, GETDATE()),
       ('X3', 1, 1, 1, GETDATE()),
       ('X4', 1, 4, 1, GETDATE()),
       ('X5', 1, 4, 1, GETDATE()),
       ('X6', 1, 1, 1, GETDATE()),
       ('X7', 1, 1, 1, GETDATE()),
      ('Cooper', 2, 1, 1, GETDATE()),
       ('Countryman', 2, 2, 1, GETDATE()),
       ('Clubman', 2, 3, 1, GETDATE()),
       ('Convertible', 2, 1, 1, GETDATE()),
       ('Paceman', 2, 4, 1, GETDATE());

-- [dbo].[proposal_type]
-- Insertar tipos de propuestas
INSERT INTO [dbo].[proposal_type] ([name], active, creation_date)
VALUES ('Nuevo', 1, GETDATE()),
       ('Seminuevo', 1, GETDATE()),
       ('Renting', 1, GETDATE()),
       ('Leasing', 1, GETDATE());

-- [dbo].[concessionaire]
-- Inserta concesionarios de BMW
INSERT INTO [dbo].[concessionaire] ([name], [address], [city], [country], [postal_code], active, creation_date)
VALUES ('BYmyCAR Madrid', 'Avenida de Burgos, 133', 'Madrid', 'España', '28050', 1, GETDATE()),
       ('Avilcar', 'Jorge de Santayana, 74', 'Ávila', 'España', '05004', 1, GETDATE()),
      ('Vehinter', NULL, NULL, 'España', NULL, 1, GETDATE()),
       ('Movitransa', 'Avenida Tío Pepe, 55', 'Jerez de la Frontera', 'España', '11407', 1, GETDATE()),
       ('Barcelona Premium', 'c/ Entença, 324-326', 'Barcelona', 'España', NULL, 1, GETDATE());

-- [dbo].[customer]
-- Inserta clientes ficticios
INSERT INTO [dbo].[customer] ([name], phone, email, creation_date) VALUES
    ('Laura Rodríguez', '111222333', 'laura@example.com', GETDATE()),
    ('Carlos Gómez', '444555666', 'carlos@example.com', GETDATE()),
    ('Sara López', '777888999', 'sara@example.com', GETDATE()),
    ('Javier Fernández', '123456789', 'javier@example.com', GETDATE()),
    ('Lucía Martínez', '987654321', 'lucia@example.com', GETDATE()),
    ('Alejandro Pérez', '456789123', 'alejandro@example.com', GETDATE()),
    ('Elena García', '789123456', 'elena@example.com', GETDATE()),
    ('Pablo Ruiz', '111333555', 'pablo@example.com', GETDATE()),
    ('Carmen Sánchez', '444666888', 'carmen@example.com', GETDATE()),
    ('Daniel Díaz', '777999111', 'daniel@example.com', GETDATE()),
    ('Marina Vázquez', '222444666', 'marina@example.com', GETDATE()),
    ('Mario Torres', '555777999', 'mario@example.com', GETDATE()),
    ('Clara Romero', '888111333', 'clara@example.com', GETDATE()),
    ('Antonio Navarro', '333555777', 'antonio@example.com', GETDATE()),
    ('Isabel Jiménez', '666888111', 'isabel@example.com', GETDATE()),
    ('Sergio Gutiérrez', '999111333', 'sergio@example.com', GETDATE()),
    ('Paula Castro', '222555888', 'paula@example.com', GETDATE()),
    ('Alberto Molina', '555888222', 'alberto@example.com', GETDATE()),
    ('Cristina Ortega', '888222555', 'cristina@example.com', GETDATE()),
    ('David Delgado', '111444777', 'david@example.com', GETDATE());

-- [dbo].[seller]
-- Insertar vendedores ficticios
INSERT INTO [dbo].[seller] ([name], phone, email, concessionaire_id, active, creation_date) VALUES
    ('Ana Martín', '111222333', 'ana@example.com', 1, 1, GETDATE()),
    ('Pedro Sánchez', '444555666', 'pedro@example.com', 1, 1, GETDATE()),
    ('María García', '777888999', 'maria@example.com', 1, 1, GETDATE()),
```

```
    ('Juan López', '123456789', 'juan@example.com', 1, 1, GETDATE()),
    ('Lucía Pérez', '987654321', 'lucia@example.com', 1, 1, GETDATE()),
    ('Carlos Gómez', '456789123', 'carlos@example.com', 2, 1, GETDATE()),
    ('Elena Rodríguez', '789123456', 'elena@example.com', 2, 1, GETDATE()),
    ('Marta Martínez', '111333555', 'marta@example.com', 3, 1, GETDATE()),
    ('David Díaz', '444666888', 'david@example.com', 3, 1, GETDATE()),
    ('Laura Ruiz', '777999111', 'laura@example.com', 4, 1, GETDATE()),
    ('Javier Torres', '222444666', 'javier@example.com', 4, 1, GETDATE()),
    ('Sara Jiménez', '555777999', 'sara@example.com', 4, 1, GETDATE()),
    ('Pablo Romero', '888111333', 'pablo@example.com', 4, 1, GETDATE()),
    ('Cristina Navarro', '333555777', 'cristina@example.com', 5, 1, GETDATE()),
    ('Alberto Ortega', '666888111', 'alberto@example.com', 5, 1, GETDATE()),
    ('Marta Gutiérrez', '999111333', 'marta@example.com', 5, 1, GETDATE()),
    ('Diego Castro', '222555888', 'diego@example.com', 5, 1, GETDATE()),
    ('Sofía Molina', '555888222', 'sofia@example.com', 5, 1, GETDATE()),
    ('José Ortega', '888222555', 'jose@example.com', 5, 1, GETDATE()),
    ('Ana Delgado', '111444777', 'ana@example.com', 5, 1, GETDATE());

-- [dbo].[proposal]
-- Insertar propuestas ficticias con foreign keys
INSERT INTO [dbo].[proposal] (customer_id, seller_id, model_type_id, proposal_date, proposal_amount, proposal_type_id, active, creation_date)
VALUES
   (1, 1, 1, '2023-01-01', 30000.00, 1, 1, GETDATE()),
   (2, 1, 2, '2023-01-02', 25000.00, 1, 1, GETDATE()),
   (3, 1, 3, '2023-01-03', 20000.00, 2, 1, GETDATE()),
   (4, 3, 4, '2023-01-04', 38000.00, 3, 1, GETDATE()),
   (5, 3, 5, '2023-01-05', 22000.00, 4, 1, GETDATE()),
   (6, 4, 6, '2023-01-06', 46000.00, 1, 1, GETDATE()),
   (7, 4, 7, '2023-01-07', 59000.00, 1, 1, GETDATE()),
   (8, 4, 8, '2023-01-08', 22000.00, 2, 1, GETDATE()),
   (9, 4, 9, '2023-01-09', 17000.00, 2, 1, GETDATE()),
   (10, 4, 10, '2023-01-10', 33000.00, 2, 1, GETDATE()),
   (11, 4, 11, '2023-01-11', 34000.00, 4, 1, GETDATE()),
   (12, 5, 1, '2023-01-12', 25000.00, 4, 1, GETDATE()),
   (13, 6, 14, '2023-01-13', 23000.00, 3, 1, GETDATE()),
   (14, 6, 19, '2023-01-14', 24000.00, 2, 1, GETDATE()),
   (15, 6, 21, '2023-01-15', 26000.00, 1, 1, GETDATE()),
   (16, 7, 20, '2023-01-16', 27000.00, 3, 1, GETDATE()),
   (17, 7, 20, '2023-01-17', 28000.00, 3, 1, GETDATE()),
   (18, 8, 11, '2023-01-18', 29000.00, 3, 1, GETDATE()),
   (19, 8, 10, '2023-01-19', 30000.00, 1, 1, GETDATE()),
   (20, 9, 12, '2023-01-20', 31000.00, 2, 1, GETDATE()),
   (1, 9, 1, '2023-01-21', 32000.00, 2, 1, GETDATE()),
   (2, 9, 2, '2023-01-22', 33000.00, 2, 1, GETDATE()),
   (3, 10, 3, '2023-01-23', 34000.00, 4, 1, GETDATE()),
   (4, 10, 4, '2023-01-24', 35000.00, 4, 1, GETDATE()),
   (1, 10, 3, '2023-01-25', 36000.00, 4, 1, GETDATE()),
   (2, 10, 3, '2023-01-26', 37000.00, 1, 1, GETDATE()),
   (3, 10, 2, '2023-01-27', 38000.00, 1, 1, GETDATE()),
   (4, 12, 2, '2023-01-28', 39000.00, 1, 1, GETDATE()),
   (8, 13, 1, '2023-01-29', 40000.00, 1, 1, GETDATE()),
   (8, 14, 1, '2023-01-30', 41000.00, 1, 1, GETDATE()),
   (8, 14, 11, '2023-01-31', 42000.00, 2, 1, GETDATE()),
   (10, 15, 10, '2023-02-01', 43000.00, 3, 1, GETDATE()),
   (11, 15, 10, '2023-02-02', 44000.00, 2, 1, GETDATE()),
   (12, 15, 9, '2023-02-03', 45000.00, 2, 1, GETDATE()),
   (13, 15, 8, '2023-02-04', 46000.00, 4, 1, GETDATE()),
   (14, 15, 12, '2023-02-05', 47000.00, 4, 1, GETDATE()),
   (14, 15, 13, '2023-02-06', 48000.00, 1, 1, GETDATE()),
   (14, 15, 15, '2023-02-07', 49000.00, 1, 1, GETDATE()),
   (14, 16, 5, '2023-02-08', 50000.00, 3, 1, GETDATE()),
   (14, 18, 6, '2023-02-09', 51000.00, 1, 1, GETDATE()),
   (15, 18, 7, '2023-02-10', 52000.00, 1, 1, GETDATE()),
   (16, 18, 20, '2023-02-11', 53000.00, 2, 1, GETDATE()),
   (1, 19, 21, '2023-02-12', 54000.00, 3, 1, GETDATE()),
   (18, 19, 22, '2023-02-13', 55000.00, 4, 1, GETDATE()),
   (19, 20, 9, '2023-02-14', 56000.00, 2, 1, GETDATE()),
   (20, 20, 6, '2023-02-15', 57000.00, 2, 1, GETDATE()),
   (11, 20, 5, '2023-02-16', 58000.00, 2, 1, GETDATE()),
   (3, 20, 4, '2023-02-17', 59000.00, 1, 1, GETDATE()),
   (3, 20, 1, '2023-02-18', 60000.00, 1, 1, GETDATE()),
   (7, 20, 1, '2023-02-19', 61000.00, 3, 1, GETDATE());

-- [dbo].[offer]
-- Insertar ofertas ficticias
INSERT INTO [dbo].[offer] (proposal_id, offer_date, offer_amount, active, creation_date) VALUES
    (1, '2023-03-15', 24000.00, 1, GETDATE()),
    (2, '2023-03-16', 27000.00, 1, GETDATE()),
    (3, '2023-03-17', 29000.00, 1, GETDATE()),
    (4, '2023-03-18', 26000.00, 1, GETDATE()),
    (5, '2023-03-19', 31000.00, 1, GETDATE()),
    (6, '2023-03-20', 28000.00, 1, GETDATE()),
    (7, '2023-03-21', 25000.00, 1, GETDATE()),
    (8, '2023-03-22', 32000.00, 1, GETDATE()),
    (9, '2023-03-23', 27000.00, 1, GETDATE()),
    (10, '2023-03-24', 30000.00, 1, GETDATE()),
    (11, '2023-03-25', 29000.00, 1, GETDATE()),
    (12, '2023-03-26', 26000.00, 1, GETDATE()),
    (13, '2023-03-27', 28000.00, 1, GETDATE()),
    (14, '2023-03-28', 31000.00, 1, GETDATE()),
    (15, '2023-03-29', 24000.00, 1, GETDATE()),
    (16, '2023-03-30', 33000.00, 1, GETDATE()),
    (17, '2023-03-31', 27000.00, 1, GETDATE()),
    (18, '2023-04-01', 30000.00, 1, GETDATE()),
    (19, '2023-04-02', 28000.00, 1, GETDATE()),
    (20, '2023-04-03', 29000.00, 1, GETDATE()),
    (21, '2023-04-04', 26000.00, 1, GETDATE()),
    (22, '2023-04-05', 30000.00, 1, GETDATE()),
    (23, '2023-04-06', 27000.00, 1, GETDATE()),
```

```
    (24, '2023-04-07', 29000.00, 1, GETDATE()),
    (25, '2023-04-08', 31000.00, 1, GETDATE()),
    (26, '2023-04-09', 28000.00, 1, GETDATE()),
    (27, '2023-04-10', 26000.00, 1, GETDATE()),
    (28, '2023-04-11', 33000.00, 1, GETDATE()),
    (29, '2023-04-12', 27000.00, 1, GETDATE()),
    (30, '2023-04-13', 30000.00, 1, GETDATE()),
    (31, '2023-04-14', 29000.00, 1, GETDATE()),
    (32, '2023-04-15', 26000.00, 1, GETDATE()),
    (33, '2023-04-16', 28000.00, 1, GETDATE()),
    (34, '2023-04-17', 31000.00, 1, GETDATE()),
    (35, '2023-04-18', 24000.00, 1, GETDATE()),
    (36, '2023-04-19', 32000.00, 1, GETDATE()),
    (37, '2023-04-20', 27000.00, 1, GETDATE()),
    (38, '2023-04-21', 30000.00, 1, GETDATE()),
    (39, '2023-04-22', 28000.00, 1, GETDATE()),
    (40, '2023-04-23', 29000.00, 1, GETDATE());

-- [dbo].[sale]
-- Insertar ventas ficticias
INSERT INTO dbo.sale (offer_id, sale_date, creation_date) VALUES
   (1, '2023-05-01', GETDATE()),
   (2, '2023-05-02', GETDATE()),
   (3, '2023-05-03', GETDATE()),
   (4, '2023-05-04', GETDATE()),
   (5, '2023-05-05', GETDATE()),
   (6, '2023-05-06', GETDATE()),
   (7, '2023-05-07', GETDATE()),
   (8, '2023-05-08', GETDATE()),
   (9, '2023-05-09', GETDATE()),
   (10, '2023-05-10', GETDATE()),
   (11, '2023-05-11', GETDATE()),
   (12, '2023-05-12', GETDATE()),
   (13, '2023-05-13', GETDATE()),
   (14, '2023-06-14', GETDATE()),
   (15, '2023-06-15', GETDATE()),
   (16, '2023-06-16', GETDATE()),
   (17, '2023-06-17', GETDATE()),
   (18, '2023-06-18', GETDATE()),
   (19, '2023-06-19', GETDATE()),
   (20, '2023-06-20', GETDATE()),
   (21, '2023-06-21', GETDATE()),
   (22, '2023-06-22', GETDATE()),
   (23, '2023-06-23', GETDATE()),
   (24, '2023-06-24', GETDATE()),
   (25, '2023-06-25', GETDATE()),
   (26, '2023-06-26', GETDATE()),
   (27, '2023-06-27', GETDATE()),
   (28, '2023-06-28', GETDATE()),
   (29, '2023-06-29', GETDATE()),
   (30, '2023-06-30', GETDATE());

/*
SELECT * FROM [dbo].[sale]
SELECT * FROM [dbo].[offer]
SELECT * FROM [dbo].[proposal]
SELECT * FROM [dbo].[seller]
SELECT * FROM [dbo].[customer]
SELECT * FROM [dbo].[concessionaire]
SELECT * FROM [dbo].[proposal_type]
SELECT * FROM [dbo].[model_type]
SELECT * FROM [dbo].[fuel_type]
SELECT * FROM [dbo].[brand_type]
*/

-- COMMIT
-- ROLLBACK
```

A continuación se muestra el *Código C.14* SQL para la creación de la BBDD *VehicleSalesAlt*.

Código C.14: 3 Creacion BBDD VehicleAlt.sql

```
/*
-- ============================================================
-- ==================== DESCRIPCION ===========================
-- ============================================================
1. brand_type:
   . Tabla para almacenar información sobre las marcas de vehículos.
      - id (PK, int): Identificador único de la marca.
      - name (NVARCHAR): Nombre de la marca de vehículos.
      - active (bit): Indica si el registro está o no activo.
      - creation_date (DF, datetime): Fecha de creación del registro generada por default.
      - modification_date (TR, datetime): Fecha de modificación del registro actualiada por trigger.
      - io_id (int): Clave de la operación realizada en la BBDD IO.
2. fuel_type:
   . Tabla para almacenar información sobre el tipo de combustible del modelo.
      - id (PK, int): Identificador único del combustible.
      - name (NVARCHAR): Tipo de combustible del vehículo (gasolina, diésel, eléctrico, híbrido).
      - active (bit): Indica si el registro está o no activo.
      - creation_date (DF, datetime): Fecha de creación del registro generada por default.
      - modification_date (TR, datetime): Fecha de modificación del registro actualiada por trigger.
      - io_id (int): Clave de la operación realizada en la BBDD IO.
3. model_type:
   . Tabla para almacenar información sobre los modelos de vehículos.
      - id (PK, int): Identificador único del modelo.
```

```
      - model_name (NVARCHAR): Nombre del modelo de vehículo.
      - fuel_type_id (FK, int): Clave foránea que referencia la tabla fuel_type.
      - brand_id (FK, int): Clave foránea que referencia la tabla concessionare.
      - active (bit): Indica si el registro está o no activo.
      - creation_date (DF, datetime): Fecha de creación del registro generada por default.
      - modification_date (TR, datetime): Fecha de modificación del registro actualiada por trigger.
      - io_id (int): Clave de la operación realizada en la BBDD IO.
4. proposal_type:
   . Tabla para almacenar información sobre el tipo de propuesta realizada.
      - id (PK, int): Identificador único del tipo de propuesta.
      - name (NVARCHAR): Tipo de propuesta realizada (nuevo, seminuevo, renting, leasing).
      - active (bit): Indica si el registro está o no activo.
      - creation_date (DF, datetime): Fecha de creación del registro generada por default.
      - modification_date (TR, datetime): Fecha de modificación del registro actualiada por trigger.
      - io_id (int): Clave de la operación realizada en la BBDD IO.
5. concessionare:
   . Tabla para almacenar información sobre los concesionarios de vehículos.
      - id (PK, int): Identificador único del concesionario.
      - name (NVARCHAR): Nombre del concesionario.
      - address (NVARCHAR): Dirección del concesionario.
      - city (NVARCHAR): Ciudad donde se encuentra el concesionario.
      - country (NVARCHAR): País donde se encuentra el concesionario.
      - postal_code (NVARCHAR): Código postal donde se encuentra el concesionario.
      - active (bit): Indica si el registro está o no activo.
      - creation_date (DF, datetime): Fecha de creación del registro generada por default.
      - modification_date (TR, datetime): Fecha de modificación del registro actualiada por trigger.
      - io_id (int): Clave de la operación realizada en la BBDD IO.
6. customer:
   . Tabla para almacenar información sobre los clientes.
      - id (PK, int): Identificador único del cliente.
      - name (NVARCHAR): Nombre del cliente.
      - phone (NVARCHAR): Teléfono del cliente.
      - email (NVARCHAR): Dirección de correo electrónico del cliente.
      - creation_date (DF, datetime): Fecha de creación del registro generada por default.
      - modification_date (TR, datetime): Fecha de modificación del registro actualiada por trigger.
      - io_id (int): Clave de la operación realizada en la BBDD IO.
7. seller:
   . Tabla para almacenar información sobre los vendedores de los concesionarios.
      - id (PK, int): Identificador único del vendedor.
      - name (NVARCHAR): Nombre del vendedor.
      - phone (NVARCHAR): Teléfono del vendedor.
      - email (NVARCHAR): Dirección de correo electrónico del vendedor.
      - concessionaire_id (FK, int): Clave foránea que referencia la tabla concessionaire.
      - active (bit): Indica si el registro está o no activo.
      - creation_date (DF, datetime): Fecha de creación del registro generada por default.
      - modification_date (TR, datetime): Fecha de modificación del registro actualiada por trigger.
      - io_id (int): Clave de la operación realizada en la BBDD IO.
8. proposal:
   . Tabla para almacenar información sobre las propuestas realizadas a los clientes.
      - id (PK, int): Identificador único de la propuesta.
      - customer_id (FK, int): Clave foránea que referencia la tabla customer.
      - seller_id (FK, int): Clave foránea que referencia la tabla seller.
      - model_type_id (FK, int): Clave foránea que referencia la tabla model_type.
      - proposal_date (date): Fecha en que se realizó la propuesta.
      - proposal_amount (decimal): Precio de la propuesta.
      - proposal_type_id (FK, int): Clave foránea que referencia la tabla proposal_type.
      - active (bit): Indica si el registro está o no activo.
      - creation_date (DF, datetime): Fecha de creación del registro generada por default.
      - modification_date (TR, datetime): Fecha de modificación del registro actualiada por trigger.
      - io_id (int): Clave de la operación realizada en la BBDD IO.
9. offer:
   . Tabla para almacenar información sobre las ofertas realizadas a los clientes basadas en las propuestas.
      - id (PK, int): Identificador único de la oferta.
      - proposal_id (FK, int): Clave foránea que referencia la tabla Proposal.
      - offer_date (date): Fecha en que se realizó la oferta.
      - offer_amount (decimal): Precio de la oferta.
      - active (bit): Indica si el registro está o no activo.
      - creation_date (DF, datetime): Fecha de creación del registro generada por default.
      - modification_date (TR, datetime): Fecha de modificación del registro actualiada por trigger.
      - io_id (int): Clave de la operación realizada en la BBDD IO.
10. sale:
   . Tabla para almacenar información sobre las ventas realizadas a los clientes basadas en las ofertas.
      - id (PK, int): Identificador único de la venta.
      - offer_id (FK, int): Clave foránea que referencia la tabla Offer.
      - sale_date (date): Fecha en que se realizó la venta.
      - creation_date (DF, datetime): Fecha de creación del registro generada por default.
      - modification_date (TR, datetime): Fecha de modificación del registro actualiada por trigger.
      - io_id (int): Clave de la operación realizada en la BBDD IO.
*/

-- ============================================================
-- ==================== CREACION BBDD =========================
-- ============================================================
-- Create VehicleAlt Database
CREATE DATABASE VehicleAlt;
GO

-- Use VehicleAlt Database
USE VehicleAlt;
GO

/*
USE Master;
ALTER DATABASE VehicleAlt SET single_user with rollback immediate;
ALTER DATABASE VehicleAlt SET MULTI_USER;
DROP DATABASE VehicleAlt;
*/

-- ============================================================
```

```
-- ================== CREACION TABLAS ======================
-- ==========================================================
-- Create brand_type Table
CREATE TABLE [dbo].[brand_type] (
    id INT IDENTITY(1, 1),
    [name] NVARCHAR(64),
   active BIT,
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_brand_type] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

-- Create fuel_type Table
CREATE TABLE [dbo].[fuel_type] (
    id INT IDENTITY(1, 1),
    [name] NVARCHAR(64),
   active BIT,
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_fuel_type] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

-- Create model_type Table
CREATE TABLE [dbo].[model_type] (
    id INT IDENTITY(1, 1),
    [name] NVARCHAR(128),
    brand_type_id INT,
   fuel_type_id INT,
   active BIT,
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_model_type] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

-- Create proposal_type Table
CREATE TABLE [dbo].[proposal_type] (
    id INT IDENTITY(1, 1),
    [name] NVARCHAR(128),
   active BIT,
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_sale_type] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

-- Create concessionaire Table
CREATE TABLE [dbo].[concessionaire] (
    id INT IDENTITY(1, 1),
    [name] NVARCHAR(128),
    [address] NVARCHAR (512),
   city NVARCHAR(64),
    country NVARCHAR(32),
   postal_code NVARCHAR(18),
   info NVARCHAR(512),
   active BIT,
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_concessionaire] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

-- Create customer Table
CREATE TABLE [dbo].[customer] (
    id INT IDENTITY(1, 1),
    [name] NVARCHAR(128),
   phone NVARCHAR(18),
    email NVARCHAR(256),
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_customer] PRIMARY KEY CLUSTERED
```

```
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

-- Create seller Table
CREATE TABLE [dbo].[seller] (
    id INT IDENTITY(1, 1),
    [name] NVARCHAR(128),
   phone NVARCHAR(18),
    email NVARCHAR(256),
    concessionaire_id INT,
   active BIT,
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_seller] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

-- Create proposal Table
CREATE TABLE [dbo].[proposal] (
    id INT IDENTITY(1, 1),
    customer_id INT,
    seller_id INT,
   model_type_id INT,
    proposal_date DATE,
   proposal_amount DECIMAL(10, 2),
   proposal_type_id INT,
   active BIT,
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_proposal] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

-- Create offer Table
CREATE TABLE [dbo].[offer] (
    id INT IDENTITY(1, 1),
    proposal_id INT,
    offer_date DATE,
    offer_amount DECIMAL(10, 2),
   active BIT,
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_offer] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

-- Create sale Table
CREATE TABLE [dbo].[sale] (
    id INT IDENTITY(1, 1),
    offer_id INT,
    sale_date DATE,
   creation_date DATETIME,
   modification_date DATETIME,
   io_id INT,
CONSTRAINT [pk_sale] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[sale] ADD CONSTRAINT [df_sale_creation_date] DEFAULT (getdate()) FOR [creation_date]
GO

-- ============================================================
-- ================== CREACION DATOS ==========================
-- ============================================================

BEGIN TRAN

/*
DELETE FROM [dbo].[sale] DBCC CHECKIDENT ('[dbo].[sale]',RESEED, 0)
DELETE FROM [dbo].[offer] DBCC CHECKIDENT ('[dbo].[offer]',RESEED, 0)
DELETE FROM [dbo].[proposal] DBCC CHECKIDENT ('[dbo].[proposal]',RESEED, 0)
DELETE FROM [dbo].[seller] DBCC CHECKIDENT ('[dbo].[seller]',RESEED, 0)
DELETE FROM [dbo].[customer] DBCC CHECKIDENT ('[dbo].[customer]',RESEED, 0)
DELETE FROM [dbo].[concessionaire] DBCC CHECKIDENT ('[dbo].[concessionaire]',RESEED, 0)
DELETE FROM [dbo].[proposal_type] DBCC CHECKIDENT ('[dbo].[proposal_type]',RESEED, 0)
DELETE FROM [dbo].[model_type] DBCC CHECKIDENT ('[dbo].[model_type]',RESEED, 0)
```

```
DELETE FROM [dbo].[fuel_type] DBCC CHECKIDENT ('[dbo].[fuel_type]',RESEED, 0)
DELETE FROM [dbo].[brand_type] DBCC CHECKIDENT ('[dbo].[brand_type]',RESEED, 0)
*/

-- [dbo].[brand_type]
-- Inserta BMW/Mini
INSERT INTO [dbo].[brand_type] ([name], active, creation_date)
VALUES ('BMW', 1, GETDATE()),
       ('Mini', 1, GETDATE());

-- [dbo].[fuel_type]
-- Insert Gasolina
INSERT INTO [dbo].[fuel_type] ([name], active, creation_date)
VALUES ('Gasolina', 1, GETDATE()),
       ('Diésel', 1, GETDATE()),
       ('Eléctrico', 1, GETDATE()),
       ('Híbrido', 1, GETDATE());

-- [dbo].[model_type]
-- Inserta modelos de BMW/Mini
INSERT INTO [dbo].[model_type] ([name], [brand_type_id], [fuel_type_id], active, creation_date)
VALUES ('Serie 1', 1, 1, 1, GETDATE()),
       ('Serie 2', 1, 4, 1, GETDATE()),
       ('Serie 3', 1, 1, 1, GETDATE()),
       ('Serie 4', 1, 2, 1, GETDATE()),
       ('Serie 5', 1, 2, 1, GETDATE()),
       ('Serie 6', 1, 1, 1, GETDATE()),
       ('Serie 7', 1, 1, 1, GETDATE()),
       ('Serie 8', 1, 1, 1, GETDATE()),
      ('I3', 1, 3, 1, GETDATE()),
      ('I8', 1, 3, 1, GETDATE()),
       ('X1', 1, 2, 1, GETDATE()),
       ('X2', 1, 2, 1, GETDATE()),
       ('X3', 1, 1, 1, GETDATE()),
       ('X4', 1, 4, 1, GETDATE()),
       ('X5', 1, 4, 1, GETDATE()),
       ('X6', 1, 1, 1, GETDATE()),
       ('X7', 1, 1, 1, GETDATE()),
      ('Cooper', 2, 1, 1, GETDATE()),
       ('Countryman', 2, 2, 1, GETDATE()),
       ('Clubman', 2, 3, 1, GETDATE()),
       ('Convertible', 2, 1, 1, GETDATE()),
       ('Paceman', 2, 4, 1, GETDATE());

-- [dbo].[proposal_type]
-- Insertar tipos de propuestas
INSERT INTO [dbo].[proposal_type] ([name], active, creation_date)
VALUES ('Nuevo', 1, GETDATE()),
       ('Seminuevo', 1, GETDATE()),
       ('Renting', 1, GETDATE()),
       ('Leasing', 1, GETDATE());

-- [dbo].[concessionaire]
-- Inserta concesionarios de BMW
INSERT INTO [dbo].[concessionaire] ([name], [address], [city], [country], [postal_code], active, creation_date)
VALUES ('BYmyCAR Madrid', 'Avenida de Burgos, 133', 'Madrid', 'España', '28050', 1, GETDATE()),
       ('Avilcar', 'Jorge de Santayana, 74', 'Ávila', 'España', '05004', 1, GETDATE()),
      ('Vehinter', NULL, NULL, 'España', NULL, 1, GETDATE()),
       ('Movitransa', 'Avenida Tío Pepe, 55', 'Jerez de la Frontera', 'España', '11407', 1, GETDATE()),
       ('Barcelona Premium', 'c/ Entença, 324-326', 'Barcelona', 'España', NULL, 1, GETDATE());

-- [dbo].[customer]
-- Inserta clientes ficticios
INSERT INTO [dbo].[customer] ([name], phone, email, creation_date) VALUES
    ('Laura Rodríguez', '111222333', 'laura@example.com', GETDATE()),
    ('Carlos Gómez', '444555666', 'carlos@example.com', GETDATE()),
    ('Sara López', '777888999', 'sara@example.com', GETDATE()),
    ('Javier Fernández', '123456789', 'javier@example.com', GETDATE()),
    ('Lucía Martínez', '987654321', 'lucia@example.com', GETDATE()),
    ('Alejandro Pérez', '456789123', 'alejandro@example.com', GETDATE()),
    ('Elena García', '789123456', 'elena@example.com', GETDATE()),
    ('Pablo Ruiz', '111333555', 'pablo@example.com', GETDATE()),
    ('Carmen Sánchez', '444666888', 'carmen@example.com', GETDATE()),
    ('Daniel Díaz', '777999111', 'daniel@example.com', GETDATE()),
    ('Marina Vázquez', '222444666', 'marina@example.com', GETDATE()),
    ('Mario Torres', '555777999', 'mario@example.com', GETDATE()),
    ('Clara Romero', '888111333', 'clara@example.com', GETDATE()),
    ('Antonio Navarro', '333555777', 'antonio@example.com', GETDATE()),
    ('Isabel Jiménez', '666888111', 'isabel@example.com', GETDATE()),
    ('Sergio Gutiérrez', '999111333', 'sergio@example.com', GETDATE()),
    ('Paula Castro', '222555888', 'paula@example.com', GETDATE()),
    ('Alberto Molina', '555888222', 'alberto@example.com', GETDATE()),
    ('Cristina Ortega', '888222555', 'cristina@example.com', GETDATE()),
    ('David Delgado', '111444777', 'david@example.com', GETDATE());

-- [dbo].[seller]
-- Insertar vendedores ficticios
INSERT INTO [dbo].[seller] ([name], phone, email, concessionaire_id, active, creation_date) VALUES
    ('Ana Martín', '111222333', 'ana@example.com', 1, 1, GETDATE()),
    ('Pedro Sánchez', '444555666', 'pedro@example.com', 1, 1, GETDATE()),
    ('María García', '777888999', 'maria@example.com', 1, 1, GETDATE()),
    ('Juan López', '123456789', 'juan@example.com', 1, 1, GETDATE()),
    ('Lucía Pérez', '987654321', 'lucia@example.com', 1, 1, GETDATE()),
    ('Carlos Gómez', '456789123', 'carlos@example.com', 2, 1, GETDATE()),
    ('Elena Rodríguez', '789123456', 'elena@example.com', 2, 1, GETDATE()),
    ('Marta Martínez', '111333555', 'marta@example.com', 3, 1, GETDATE()),
    ('David Díaz', '444666888', 'david@example.com', 3, 1, GETDATE()),
    ('Laura Ruiz', '777999111', 'laura@example.com', 4, 1, GETDATE()),
    ('Javier Torres', '222444666', 'javier@example.com', 4, 1, GETDATE()),
```

```
    ('Sara Jiménez', '555777999', 'sara@example.com', 4, 1, GETDATE()),
    ('Pablo Romero', '888111333', 'pablo@example.com', 4, 1, GETDATE()),
    ('Cristina Navarro', '333555777', 'cristina@example.com', 5, 1, GETDATE()),
    ('Alberto Ortega', '666888111', 'alberto@example.com', 5, 1, GETDATE()),
    ('Marta Gutiérrez', '999111333', 'marta@example.com', 5, 1, GETDATE()),
    ('Diego Castro', '222555888', 'diego@example.com', 5, 1, GETDATE()),
    ('Sofía Molina', '555888222', 'sofia@example.com', 5, 1, GETDATE()),
    ('José Ortega', '888222555', 'jose@example.com', 5, 1, GETDATE()),
    ('Ana Delgado', '111444777', 'ana@example.com', 5, 1, GETDATE());

-- [dbo].[proposal]
-- Insertar propuestas ficticias con foreign keys
INSERT INTO [dbo].[proposal] (customer_id, seller_id, model_type_id, proposal_date, proposal_amount, proposal_type_id, active, creation_date)
VALUES
   (1, 1, 1, '2023-01-01', 30000.00, 1, 1, GETDATE()),
   (2, 1, 2, '2023-01-02', 25000.00, 1, 1, GETDATE()),
   (3, 1, 3, '2023-01-03', 20000.00, 2, 1, GETDATE()),
   (4, 3, 4, '2023-01-04', 38000.00, 3, 1, GETDATE()),
   (5, 3, 5, '2023-01-05', 22000.00, 4, 1, GETDATE()),
   (6, 4, 6, '2023-01-06', 46000.00, 1, 1, GETDATE()),
   (7, 4, 7, '2023-01-07', 59000.00, 1, 1, GETDATE()),
   (8, 4, 8, '2023-01-08', 22000.00, 2, 1, GETDATE()),
   (9, 4, 9, '2023-01-09', 17000.00, 2, 1, GETDATE()),
   (10, 4, 10, '2023-01-10', 33000.00, 2, 1, GETDATE()),
   (11, 4, 11, '2023-01-11', 34000.00, 4, 1, GETDATE()),
   (12, 5, 1, '2023-01-12', 25000.00, 4, 1, GETDATE()),
   (13, 6, 14, '2023-01-13', 23000.00, 3, 1, GETDATE()),
   (14, 6, 19, '2023-01-14', 24000.00, 2, 1, GETDATE()),
   (15, 6, 21, '2023-01-15', 26000.00, 1, 1, GETDATE()),
   (16, 7, 20, '2023-01-16', 27000.00, 3, 1, GETDATE()),
   (17, 7, 20, '2023-01-17', 28000.00, 3, 1, GETDATE()),
   (18, 8, 11, '2023-01-18', 29000.00, 3, 1, GETDATE()),
   (19, 8, 10, '2023-01-19', 30000.00, 1, 1, GETDATE()),
   (20, 9, 12, '2023-01-20', 31000.00, 2, 1, GETDATE()),
   (1, 9, 1, '2023-01-21', 32000.00, 2, 1, GETDATE()),
   (2, 9, 2, '2023-01-22', 33000.00, 2, 1, GETDATE()),
   (3, 10, 3, '2023-01-23', 34000.00, 4, 1, GETDATE()),
   (4, 10, 4, '2023-01-24', 35000.00, 4, 1, GETDATE()),
   (1, 10, 3, '2023-01-25', 36000.00, 4, 1, GETDATE()),
   (2, 10, 3, '2023-01-26', 37000.00, 1, 1, GETDATE()),
   (3, 10, 2, '2023-01-27', 38000.00, 1, 1, GETDATE()),
   (4, 12, 2, '2023-01-28', 39000.00, 1, 1, GETDATE()),
   (8, 13, 1, '2023-01-29', 40000.00, 1, 1, GETDATE()),
   (8, 14, 1, '2023-01-30', 41000.00, 1, 1, GETDATE()),
   (8, 14, 11, '2023-01-31', 42000.00, 2, 1, GETDATE()),
   (10, 15, 10, '2023-02-01', 43000.00, 3, 1, GETDATE()),
   (11, 15, 10, '2023-02-02', 44000.00, 2, 1, GETDATE()),
   (12, 15, 9, '2023-02-03', 45000.00, 2, 1, GETDATE()),
   (13, 15, 8, '2023-02-04', 46000.00, 4, 1, GETDATE()),
   (14, 15, 12, '2023-02-05', 47000.00, 4, 1, GETDATE()),
   (14, 15, 13, '2023-02-06', 48000.00, 1, 1, GETDATE()),
   (14, 15, 15, '2023-02-07', 49000.00, 1, 1, GETDATE()),
   (14, 16, 5, '2023-02-08', 50000.00, 3, 1, GETDATE()),
   (14, 18, 6, '2023-02-09', 51000.00, 1, 1, GETDATE()),
   (15, 18, 7, '2023-02-10', 52000.00, 1, 1, GETDATE()),
   (16, 18, 20, '2023-02-11', 53000.00, 2, 1, GETDATE()),
   (1, 19, 21, '2023-02-12', 54000.00, 3, 1, GETDATE()),
   (18, 19, 22, '2023-02-13', 55000.00, 4, 1, GETDATE()),
   (19, 20, 9, '2023-02-14', 56000.00, 2, 1, GETDATE()),
   (20, 20, 6, '2023-02-15', 57000.00, 2, 1, GETDATE()),
   (11, 20, 5, '2023-02-16', 58000.00, 2, 1, GETDATE()),
   (3, 20, 4, '2023-02-17', 59000.00, 1, 1, GETDATE()),
   (3, 20, 1, '2023-02-18', 60000.00, 1, 1, GETDATE()),
   (7, 20, 1, '2023-02-19', 61000.00, 3, 1, GETDATE());

-- [dbo].[offer]
-- Insertar ofertas ficticias
INSERT INTO [dbo].[offer] (proposal_id, offer_date, offer_amount, active, creation_date) VALUES
    (1, '2023-03-15', 24000.00, 1, GETDATE()),
    (2, '2023-03-16', 27000.00, 1, GETDATE()),
    (3, '2023-03-17', 29000.00, 1, GETDATE()),
    (4, '2023-03-18', 26000.00, 1, GETDATE()),
    (5, '2023-03-19', 31000.00, 1, GETDATE()),
    (6, '2023-03-20', 28000.00, 1, GETDATE()),
    (7, '2023-03-21', 25000.00, 1, GETDATE()),
    (8, '2023-03-22', 32000.00, 1, GETDATE()),
    (9, '2023-03-23', 27000.00, 1, GETDATE()),
    (10, '2023-03-24', 30000.00, 1, GETDATE()),
    (11, '2023-03-25', 29000.00, 1, GETDATE()),
    (12, '2023-03-26', 26000.00, 1, GETDATE()),
    (13, '2023-03-27', 28000.00, 1, GETDATE()),
    (14, '2023-03-28', 31000.00, 1, GETDATE()),
    (15, '2023-03-29', 24000.00, 1, GETDATE()),
    (16, '2023-03-30', 33000.00, 1, GETDATE()),
    (17, '2023-03-31', 27000.00, 1, GETDATE()),
    (18, '2023-04-01', 30000.00, 1, GETDATE()),
    (19, '2023-04-02', 28000.00, 1, GETDATE()),
    (20, '2023-04-03', 29000.00, 1, GETDATE()),
    (21, '2023-04-04', 26000.00, 1, GETDATE()),
    (22, '2023-04-05', 30000.00, 1, GETDATE()),
    (23, '2023-04-06', 27000.00, 1, GETDATE()),
    (24, '2023-04-07', 29000.00, 1, GETDATE()),
    (25, '2023-04-08', 31000.00, 1, GETDATE()),
    (26, '2023-04-09', 28000.00, 1, GETDATE()),
    (27, '2023-04-10', 26000.00, 1, GETDATE()),
    (28, '2023-04-11', 33000.00, 1, GETDATE()),
    (29, '2023-04-12', 27000.00, 1, GETDATE()),
    (30, '2023-04-13', 30000.00, 1, GETDATE()),
    (31, '2023-04-14', 29000.00, 1, GETDATE()),
```

```
    (32, '2023-04-15', 26000.00, 1, GETDATE()),
    (33, '2023-04-16', 28000.00, 1, GETDATE()),
    (34, '2023-04-17', 31000.00, 1, GETDATE()),
    (35, '2023-04-18', 24000.00, 1, GETDATE()),
    (36, '2023-04-19', 32000.00, 1, GETDATE()),
    (37, '2023-04-20', 27000.00, 1, GETDATE()),
    (38, '2023-04-21', 30000.00, 1, GETDATE()),
    (39, '2023-04-22', 28000.00, 1, GETDATE()),
    (40, '2023-04-23', 29000.00, 1, GETDATE());

-- [dbo].[sale]
-- Insertar ventas ficticias
/*
INSERT INTO dbo.sale (offer_id, sale_date, creation_date) VALUES
   (1, '2023-05-01', GETDATE()),
   (2, '2023-05-02', GETDATE()),
   (3, '2023-05-03', GETDATE()),
   (4, '2023-05-04', GETDATE()),
   (5, '2023-05-05', GETDATE()),
   (6, '2023-05-06', GETDATE()),
   (7, '2023-05-07', GETDATE()),
   (8, '2023-05-08', GETDATE()),
   (9, '2023-05-09', GETDATE()),
   (10, '2023-05-10', GETDATE()),
   (11, '2023-05-11', GETDATE()),
   (12, '2023-05-12', GETDATE()),
   (13, '2023-05-13', GETDATE()),
   (14, '2023-06-14', GETDATE()),
   (15, '2023-06-15', GETDATE()),
   (16, '2023-06-16', GETDATE()),
   (17, '2023-06-17', GETDATE()),
   (18, '2023-06-18', GETDATE()),
   (19, '2023-06-19', GETDATE()),
   (20, '2023-06-20', GETDATE()),
   (21, '2023-06-21', GETDATE()),
   (22, '2023-06-22', GETDATE()),
   (23, '2023-06-23', GETDATE()),
   (24, '2023-06-24', GETDATE()),
   (25, '2023-06-25', GETDATE()),
   (26, '2023-06-26', GETDATE()),
   (27, '2023-06-27', GETDATE()),
   (28, '2023-06-28', GETDATE()),
   (29, '2023-06-29', GETDATE()),
   (30, '2023-06-30', GETDATE());
*/

/*
SELECT * FROM [dbo].[sale]
SELECT * FROM [dbo].[offer]
SELECT * FROM [dbo].[proposal]
SELECT * FROM [dbo].[seller]
SELECT * FROM [dbo].[customer]
SELECT * FROM [dbo].[concessionaire]
SELECT * FROM [dbo].[proposal_type]
SELECT * FROM [dbo].[model_type]
SELECT * FROM [dbo].[fuel_type]
SELECT * FROM [dbo].[brand_type]
*/

-- COMMIT
-- ROLLBACK
```

La *Figura C.4* muestra el conjunto de procedimientos core SQL de la BBDD IO.

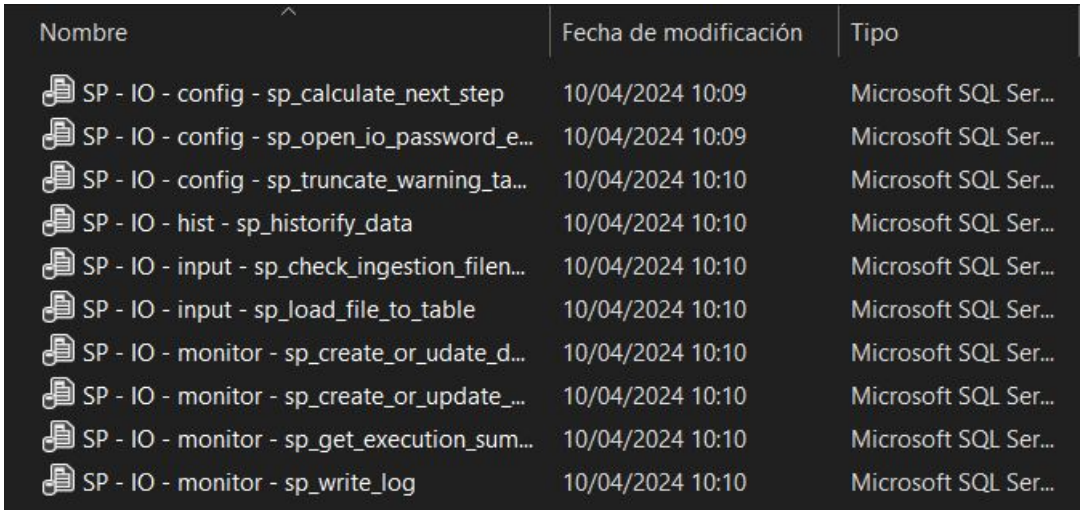


Figura C.4: Estructura de procedimientos core SQL de la BBDD IO

A continuación el *Código C.15-C.24* muestra los procedimientos core SQL de la BBDD IO.

Código C.15: config.sp_calculate_next_step.sql

```sql
USE [IO]
GO
/*
-- ============================================================================================
Autor: Adrian Maroto
Fecha creacion: 14/03/2024
Descripcion: Proceso para calcular los pasos de un proceso maestro y poder ejecutarlo según una serie de fases ordenadas.
Cambios:

Entrada:
   @p_master_process_id = Identificador del proceso maestro a ejecutar.
   @p_reset = Variable para resetear el proceso en caso de marcar a 1. 0 continua donde lo dejo la ejecución anterior.
   @p_step = Paso del proceso de ejecución.

Salida:
   @p_out_next_step = Salida con el siguiente paso del proceso.

Ejemplo:
   EXEC [config].[sp_calculate_next_step]
      @p_master_process_id = 5,
      @p_reset = 1,
      @p_step = 'INGESTION',
      @p_out_next_step = NULL
-- ============================================================================================
*/
CREATE OR ALTER PROCEDURE [config].[sp_calculate_next_step]
   @p_master_process_id INT,
   @p_reset INT,
   @p_step NVARCHAR(50),
   @p_out_next_step NVARCHAR(50) OUTPUT
AS
BEGIN

   -- Declaracion de variables internas
   DECLARE @ingestion_flag INT; --Indica si se ha realizado la ingestión
   DECLARE @transformation_flag INT; --Indica si se ha realizado la transformación
   DECLARE @persist_flag INT; --Indica si se ha realizado la persistencia
   DECLARE @output_flag INT; --Indica si se ha realizado la salida
   DECLARE @last_ingestion_status INT; --Estado de la última ingestión
   DECLARE @last_transformation_status INT; --Estado de la última transformación
   DECLARE @last_persist_status INT; --Estado de la última persistencia
   DECLARE @last_output_status INT; --Estado de la última salida
   DECLARE @last_status NVARCHAR(10); --último estado registrado

   -- Verifica si hay registros de procesos anteriores
   IF EXISTS ( SELECT *
            FROM [IO].monitor.master_process_log
            WHERE master_process_type_id = @p_master_process_id
   )
      -- Obtiene el status y los flags de procesos más recientes
      SELECT   TOP 1
            @ingestion_flag = mp.ingestion_flag,
            @transformation_flag = mp.transformation_flag,
            @persist_flag = mp.persist_flag,
            @output_flag = mp.output_flag,
            @last_ingestion_status = CASE WHEN ml.ingestion_status = 'SUCCEEDED' THEN 1 ELSE 0 END,
            @last_transformation_status = CASE WHEN ml.transformation_status = 'SUCCEEDED' THEN 1 ELSE 0 END,
            @last_persist_status = CASE WHEN ml.persistence_status = 'SUCCEEDED' THEN 1 ELSE 0 END,
```

```
            @last_output_status = CASE WHEN  ml.output_status = 'SUCCEEDED' THEN 1 ELSE 0 END
        FROM [IO].config.master_process_type mp
            INNER JOIN [IO].monitor.master_process_log ml ON mp.id = ml.master_process_type_id
        WHERE mp.id = @p_master_process_id
        ORDER BY ml.id DESC
    ELSE
        -- Si no hay registros anteriores, establece los flags a 0
        SELECT   @ingestion_flag = mp.ingestion_flag,
                @transformation_flag = mp.transformation_flag,
                @persist_flag = mp.persist_flag,
                @output_flag = mp.output_flag,
                @last_ingestion_status = 0,
                @last_transformation_status = 0,
                @last_persist_status = 0,
                @last_output_status = 0
        FROM [IO].config.master_process_type mp
        WHERE mp.id = @p_master_process_id

    -- Obtiene el último status registrado del proceso detallado
    SELECT TOP 1 @last_status = [status]
    FROM [IO].monitor.detail_process_log dpl
        INNER JOIN [IO].monitor.master_process_log mpl ON dpl.master_process_log_id = mpl.id
    WHERE mpl.master_process_type_id = @p_master_process_id
    ORDER BY dpl.id DESC

    -- Determina la siguiente acción basada en el status anterior y el paso actual
    IF @last_status = 'WARNING' AND @p_step != 'BEGINNING'
        BEGIN
            SET @p_out_next_step = 'END'
        END
    ELSE
        BEGIN
            -- Verifica si se ha reseteado el proceso y establece el siguiente paso en función de los flags y el paso actual
            IF @p_reset = 0
                BEGIN
                    IF (@ingestion_flag = @last_ingestion_status
                        AND @transformation_flag = @last_transformation_status
                        AND @persist_flag = @last_persist_status
                        AND @output_flag = @last_output_status)
                        BEGIN
                            -- Establece el siguiente paso según las banderas y el paso actual
                            IF @ingestion_flag = 1 AND @p_step = 'BEGINNING' SET @p_out_next_step = 'INGESTION'
                            ELSE IF @transformation_flag = 1 AND @p_step = 'BEGINNING' SET @p_out_next_step = 'TRANSFORMATION'
                            ELSE IF @persist_flag = 1 AND @p_step = 'BEGINNING' SET @p_out_next_step = 'PERSISTENCE'
                            ELSE IF @output_flag = 1 AND @p_step = 'BEGINNING' SET @p_out_next_step = 'OUTPUT'
                            ELSE IF @p_step = 'BEGINNING'  SET @p_out_next_step = 'ERROR'
                            ELSE SET @p_out_next_step = 'END'
                        END
                    ELSE
                        BEGIN
                            -- Establece el siguiente paso según los flags anteriores y actuales
                            IF @ingestion_flag = 1 AND @last_ingestion_status = 0
                                SET @p_out_next_step = 'INGESTION'
                            ELSE IF @transformation_flag = 1 AND @last_transformation_status = 0
                                SET @p_out_next_step = 'TRANSFORMATION'
                            ELSE IF @persist_flag = 1 AND @last_persist_status = 0
                                SET @p_out_next_step = 'PERSISTENCE'
                            ELSE IF @output_flag = 1 AND @last_output_status = 0
                                SET @p_out_next_step = 'OUTPUT'
                            ELSE SET @p_out_next_step = 'END'
                        END
                END
            ELSE
                -- Si se ha reseteado, establece el paso en mayúsculas y determina el siguiente paso según el paso actual y los flags
                BEGIN
                    SET @p_step = UPPER(REPLACE(REPLACE(@p_step, 'Start_', ''), 'End_', ''))
                    -- Si el paso es el inicio, establece el siguiente paso según los flags
                    IF @p_step = 'BEGINNING'
                        BEGIN
                            IF @ingestion_flag = 1 SET @p_out_next_step = 'INGESTION'
                            ELSE IF @transformation_flag = 1 SET @p_out_next_step = 'TRANSFORMATION'
                            ELSE IF @persist_flag = 1 SET @p_out_next_step = 'PERSISTENCE'
                            ELSE IF @output_flag = 1 SET @p_out_next_step = 'OUTPUT'
                            ELSE SET @p_out_next_step = 'ERROR'
                        END
                    -- Determina el siguiente paso según el paso actual y los flags
                    ELSE IF @p_step = 'INGESTION'
                        BEGIN
                            IF @transformation_flag = 1 SET @p_out_next_step = 'TRANSFORMATION'
                            ELSE IF @persist_flag = 1 SET @p_out_next_step = 'PERSISTENCE'
                            ELSE IF @output_flag = 1 SET @p_out_next_step = 'OUTPUT'
                            ELSE SET @p_out_next_step = 'END'
                        END
                    ELSE IF @p_step = 'TRANSFORMATION'
                        BEGIN
                            IF @persist_flag = 1 SET @p_out_next_step = 'PERSISTENCE'
                            ELSE IF @output_flag = 1 SET @p_out_next_step = 'OUTPUT'
                            ELSE SET @p_out_next_step = 'END'
                        END
                    ELSE IF @p_step = 'PERSISTENCE'
                        BEGIN
                            IF @output_flag = 1 SET @p_out_next_step = 'OUTPUT'
                            ELSE SET @p_out_next_step = 'END'
                        END
                    ELSE IF @p_step = 'OUTPUT'
                        BEGIN
                            SET @p_out_next_step = 'END'
                        END
                END
```

```
158       END
159
160   END
```

Código C.16: config.sp_open_io_password_encryption_keys.sql

```
1   USE [IO]
2   GO
3   /*
4   -- ============================================================================================
5   Autor: Adrian Maroto
6   Fecha creación: 14/03/2024
7   Descripción: Procedimiento almacenado para abrir las claves de encriptación de contraseñas en la base de datos IO.
8
9   Entrada:
10
11  Salida:
12
13  Ejemplo:
14     -- Apertura de las claves de cifrado, siempre se ha de hacer antes de encriptar o desencriptar
15     EXEC [IO].config.sp_open_io_password_encryption_keys
16     -- Funcion de encriptación, entrada un VARCHAR, salida un VARBINBARY
17     SELECT [IO].config.f_password_encryption('P4$$WOrd_unico')
18     -- Función de desencriptación, entra un VARBINARY, salida un VARCHAR
19     SELECT
          [IO].config.f_password_decryption(0x00BA1DC5F52ACF05CEF7C5F899A696B302000000B16EB65E085C7E2EC39AC7B8C51300496D53D80945989550CE140474318D2116D3F398E8AE95AB406F24A27D
20
21  -- ============================================================================================
22  */
23  CREATE OR ALTER PROCEDURE [config].[sp_open_io_password_encryption_keys]
24  AS
25  BEGIN
26
27     -- Activar la opción para no contar las filas afectadas por las instrucciones SELECT
28     SET NOCOUNT ON;
29
30     -- Intenta abrir la clave simétrica para desencriptar contraseñas
31     BEGIN TRY
32        OPEN SYMMETRIC KEY password_io_simetric_key
33        DECRYPTION BY CERTIFICATE io_db_certificate
34     END TRY
35     BEGIN CATCH
36        -- En caso de error, mostrar un mensaje de error
37        SELECT 'Some error getting encryption keys: ' + ERROR_MESSAGE()
38     END CATCH
39
40  END
```

Código C.17: config.sp_truncate_warning_table.sql

```
1   USE [IO]
2   GO
3   /*
4   -- ============================================================================================
5   Autor: Adrian Maroto
6   Fecha creación: 14/03/2024
7   Descripción: Procedimiento almacenado para truncar una tabla de advertencias.
8
9   Entrada:
10     @table_complete: Nombre completo de la tabla a truncar (incluyendo BBDD, schema y nombre de tabla).
11
12  Salida:
13
14  Ejemplo:
15     SELECT * INTO load.model_type_TEST FROM [IO].[load].model_type
16     SELECT * FROM [IO].[load].model_type_TEST
17     -- DROP TABLE [IO].[load].model_type_TEST
18
19      EXEC [config].[sp_truncate_warning_table] @table_complete = '[IO].[load].model_type_TEST'
20  -- ============================================================================================
21  */
22  CREATE OR ALTER PROCEDURE [config].[sp_truncate_warning_table]
23     @table_complete NVARCHAR(1024)
24  AS
25  BEGIN
26
27     -- Declaración de variables locales
28      DECLARE @db NVARCHAR(256) -- Nombre de la base de datos
29      DECLARE @schema NVARCHAR(256) -- Nombre del esquema
30      DECLARE @table NVARCHAR(512) -- Nombre de la tabla
31      DECLARE @sql NVARCHAR(MAX) -- Sentencia SQL dinámica para truncar la tabla
32
33     -- Extraer el nombre de la base de datos, el esquema y la tabla del nombre completo de la tabla proporcionada
34      SELECT @db = REPLACE(REPLACE(REPLACE(LEFT(@table_complete, CHARINDEX('.',@table_complete)-1), '[', ''), ']', ''), '.', '')  -- Extraer el nombre
          de la base de datos
35      SELECT @schema = REPLACE(REPLACE(REPLACE(SUBSTRING(@table_complete, CHARINDEX('.', @table_complete, 1), CHARINDEX('.', @table_complete,
          (CHARINDEX('.', @table_complete, 1))+1) - CHARINDEX('.', @table_complete, 1)), '[', ''), ']', ''), '.', '')  -- Extraer el nombre del esquema
36      SELECT @table = REPLACE(REPLACE(REPLACE(REVERSE(LEFT(REVERSE(@table_complete),CHARINDEX('.',REVERSE(@table_complete))-1)), '[', ''), ']', ''),
          '.', '')  -- Extraer el nombre de la tabla
37
38      -- Construye la sentencia SQL para truncar la tabla
39      SET @sql = 'TRUNCATE TABLE ' + @table_complete
40
41      -- Verificar si la tabla existe en la BBDD especificada y ejecuta la sentencia SQL dinámica
42      IF EXISTS (
```

```
        SELECT *
        FROM [IO].INFORMATION_SCHEMA.TABLES
        WHERE TABLE_CATALOG = @db
            AND TABLE_SCHEMA = @schema
            AND TABLE_NAME = @table
    )
    BEGIN
        EXEC(@sql) -- Ejecuta la sentencia SQL dinámica para truncar la tabla
    END

END
```

Código C.18: hist.sp_historify_data.sql

```
USE [IO]
GO
/*
-- ============================================================================================
Autor: Adrian Maroto
Fecha creación: 14/03/2024
Descripción: Procedimiento almacenado para historificar datos en una tabla de destino.

Entrada:
    @p_destination_table: Nombre completo de la tabla de destino (incluyendo BBDD, schema y nombre de tabla).
    @p_hist_table: Nombre completo de la tabla histórica (incluyendo BBDD, schema y nombre de tabla).
    @p_master_process_log_id: Identificador del proceso maestro a ejecutar.

Salida:

Ejemplo:
    EXEC [hist].[sp_historify_data] @p_destination_table = '[IO].[input].BMW_NewDealers',
        @p_hist_table = '[IO].hist.test',
        @p_master_process_log_id = 5

   -- DROP TABLE [IO].hist.test
-- ============================================================================================
*/
CREATE OR ALTER PROCEDURE [hist].[sp_historify_data]
   @p_destination_table VARCHAR(256),
   @p_hist_table VARCHAR(256),
   @p_master_process_log_id INT
AS
BEGIN

   -- Declaración de variables locales
   DECLARE @destination_database NVARCHAR(128)   -- Nombre de la base de datos de la tabla destino
   DECLARE @destination_schema NVARCHAR(128)   -- Nombre del esquema de la tabla destino
   DECLARE @destination_table NVARCHAR(128)   -- Nombre de la tabla destino
   DECLARE @hist_database NVARCHAR(128)   -- Nombre de la base de datos de la tabla de historial
   DECLARE @hist_schema NVARCHAR(128)   -- Nombre del esquema de la tabla de historial
   DECLARE @hist_table NVARCHAR(128)   -- Nombre de la tabla de historial

   DECLARE @sql NVARCHAR(MAX) -- Sentencia SQL dinámica para historificar los datos
   DECLARE @columns NVARCHAR(MAX) = '' -- Columnas de la tabla destino
   DECLARE @exists INT -- Variable para verificar si la tabla historica ya existe

   -- Extraer nombres de BBDD, schema y tabla tanto para la tabla destino como para la historica
   SELECT @destination_database = REPLACE(REPLACE(PARSENAME(@p_destination_table, 3), '[', ''), ']', '')   -- BBDD de la tabla destino
   SELECT @destination_schema = REPLACE(REPLACE(PARSENAME(@p_destination_table, 2), '[', ''), ']', '')   -- Schema de la tabla destino
   SELECT @destination_table = REPLACE(REPLACE(PARSENAME(@p_destination_table, 1), '[', ''), ']', '')   -- Tabla destino
   SELECT @hist_database = REPLACE(REPLACE(PARSENAME(@p_hist_table, 3), '[', ''), ']', '')   -- BBDD de la tabla historica
   SELECT @hist_schema = REPLACE(REPLACE(PARSENAME(@p_hist_table, 2), '[', ''), ']', '')   -- Schema de la tabla historica
   SELECT @hist_table = REPLACE(REPLACE(PARSENAME(@p_hist_table, 1), '[', ''), ']', '')   -- Tabla historica

   -- Verificar si la tabla hist ya existe en BBDD
   SELECT @exists = COUNT(*)
   FROM [IO].INFORMATION_SCHEMA.TABLES
   WHERE REPLACE(REPLACE(TABLE_CATALOG, '[', ''), ']', '') = @hist_database
      AND REPLACE(REPLACE(TABLE_SCHEMA, '[', ''), ']', '') = @hist_schema
      AND REPLACE(REPLACE(TABLE_NAME, '[', ''), ']', '') = @hist_table

   -- Obtener las columnas de la tabla destino
   SET @columns = NULL

   SELECT  @columns = COALESCE(@columns + ', ','') + '[' +  COLUMN_NAME + ']'
   FROM [IO].INFORMATION_SCHEMA.COLUMNS
   WHERE TABLE_CATALOG = @destination_database
      AND TABLE_SCHEMA = @destination_schema
      AND TABLE_NAME = @destination_table

   -- Construir la sentencia SQL para historificar los datos
   IF @exists = 0
      SET @sql = 'SELECT ' + CONVERT(VARCHAR, @p_master_process_log_id) + ' AS master_process_log_id, ' + @columns + ' INTO ' + @p_hist_table + ' FROM
      ' + @p_destination_table
   ELSE
      SET @sql = 'INSERT INTO ' + @p_hist_table + '(master_process_log_id, ' + @columns + ') SELECT ' + CONVERT(VARCHAR, @p_master_process_log_id) + '
      AS master_process_log_id, ' + @columns + ' FROM ' + @p_destination_table

   -- Ejecutar la sentencia SQL dinámica para historificar los datos
   EXEC(@sql)

END
```

Código C.19: input.sp_check_ingestion_filename.sql

```sql
USE [IO]
GO
/*
-- ============================================================================================
Autor: Adrian Maroto
Fecha creación: 14/03/2024
Descripción: Procedimiento almacenado para verificar si un nombre de archivo de ingestión existe en una tabla histórica.

Entrada:
    @p_ingest_filename: Nombre del archivo de ingesta a verificar.
    @p_hist_table: Nombre completo de la tabla histórica (incluyendo BBDD, schema y nombre de tabla).

Salida:
    Devuelve 1 si existe el nombre del archivo de ingesta en la tabla histórica, o 0 si no existe.

Ejemplo:
    EXEC [input].[sp_check_ingestion_filename]
      @p_ingest_filename = 'BMW_NewDealers_20240313_1415.csv',
      @p_hist_table = '[IO].[hist].[BMW_NewDealers]'
-- ============================================================================================
*/
CREATE OR ALTER PROCEDURE [input].[sp_check_ingestion_filename]
   @p_ingest_filename NVARCHAR(1024),
   @p_hist_table NVARCHAR(256)
AS
BEGIN

   -- Declaración de variables internas para la BBDD, schema y tabla historica de ingesta
   DECLARE @hist_database NVARCHAR(128) = REPLACE(REPLACE(PARSENAME(@p_hist_table, 3), '[', ''), ']', '')
   DECLARE @hist_schema NVARCHAR(128) = REPLACE(REPLACE(PARSENAME(@p_hist_table, 2), '[', ''), ']', '')
   DECLARE @hist_table NVARCHAR(128) = REPLACE(REPLACE(PARSENAME(@p_hist_table, 1), '[', ''), ']', '')

   -- Declaración de variables internas adicionales
   DECLARE @sql NVARCHAR(MAX) -- Variable para almacenar la consulta dinámica
   DECLARE @exists INT -- Variable para almacenar el resultado de la verificación de existencia de la tabla de historial de ingestión

   -- Verifica si existe la tabla histórica
   SELECT @exists = COUNT(*)
   FROM [IO].INFORMATION_SCHEMA.TABLES
   WHERE REPLACE(REPLACE(TABLE_CATALOG, '[', ''), ']', '') = @hist_database
      AND REPLACE(REPLACE(TABLE_SCHEMA, '[', ''), ']', '') = @hist_schema
      AND REPLACE(REPLACE(TABLE_NAME, '[', ''), ']', '') = @hist_table

   -- Construir la consulta dinámica en función de la existencia de la tabla histórica
   IF @exists = 0
      SET @sql = 'SELECT 0' -- Si la tabla no existe, devuelve 0
   ELSE
      SET @sql = 'SELECT IIF (COUNT(*) > 0, 1, 0)
               FROM ' + @p_hist_table +
               ' WHERE
               LTRIM(
                  RTRIM(
                     REVERSE(
                        SUBSTRING(
                           REVERSE(ingested_filename),
                           0,
                           CHARINDEX(''\'', REVERSE(ingested_filename),0)
                        )
                     )
                  )
               ) = ''' + @p_ingest_filename + '''' -- Si la tabla existe, construir la consulta para verificar la existencia del nombre de archivo

   -- Ejecutar la consulta dinámica
   EXEC (@sql)

END
```

Código C.20: input.sp_load_file_to_table.sql

```sql
USE [IO];
GO
/*
-- ============================================================================================
Autor: Adrian Maroto
Fecha creacion: 14/03/2024
Descripcion: Proceso de ingesta de archivos para procesos IO, se realiza la ingesta en función de una serie de parámetros y basándose en una
   plantillas .FMT en caso que la ingesta sea con archivos como origen.
Cambios:

Entrada:
   @p_ingest = Identificador del proceso.
   @p_path = Ruta completa al archivo a ingestar.
   @p_template = Ruta completa al archivo .fmt que sirve como plantilla de carga.
   @p_skip_initial_rows = Filas iniciales a descartar de la ingesta (cabeceras).
   @p_skip_final_rows = Filas finales a descartar de la ingesta (pies).
   @p_destination_table = Tabla de destino donde depositar los datos del archivo de ingesta.
   @p_order = Número de orden en la ingesta del archivo,
      si @p_order = 1 se borra la tabla y se recrea,
      en caso contrario se insertan directamente los datos en la tabla existente.

Salida:

Ejemplo:
   EXEC IO.input.sp_load_file_to_table
      @p_ingest = 1
      ,@p_path = '\\WIN-SER-AMAROTO\IO_Files\Input\BMW Group\Concesionarios\BMW_NewDealers_20240313_1415.csv'
      ,@p_template = '\\WIN-SER-AMAROTO\IO_Files\Template\BMW Group\Concesionarios.fmt'
```

```
      ,@p_skip_initial_rows = 1
      ,@p_skip_final_rows = 0
      ,@p_destination_table = '[IO].[input].BMW_NewDealers'
      ,@p_order  = 1
-- ============================================================================================
*/
CREATE OR ALTER PROCEDURE [input].[sp_load_file_to_table]
   @p_ingest INT,
   @p_path NVARCHAR(1024),
   @p_template NVARCHAR(1024),
   @p_destination_table NVARCHAR(1024),
   @p_order INT,
   @p_skip_initial_rows INT = 0,
   @p_skip_final_rows INT = 0
AS
BEGIN TRY

   SET NOCOUNT ON

   -- Varias variables de soporte al proceso
    DECLARE  @sql NVARCHAR(MAX)
         ,@log NVARCHAR(MAX)
         ,@table NVARCHAR(MAX)
         ,@firstfield NVARCHAR(1024)
         ,@sql_columns NVARCHAR(MAX)
         ,@sql_columns_func NVARCHAR(MAX)
         ,@campos NVARCHAR(MAX)
         ,@p_extension NVARCHAR(8)

   -- Por defecto le sumamos uno para que elimine filas iniciales
   SET @p_skip_initial_rows += 1;

    -- Tabla auxiliar que sirve para almacenar las columnas del archivo
   DECLARE @columns TABLE(id INT IDENTITY(1,1), [columns] VARCHAR(MAX))

   -- Se inserta una nueva fila en la tabla de log interno indicando el inicio del proceso de ingesta del archivo
    SET @log = 'INICIO carga [input].[sp_load_file_to_table] fichero '  + @p_path + ' orden ' + CAST(@p_order AS VARCHAR)
    EXEC monitor.sp_write_log @@PROCID, '', @log,'INFO'

   -- Obtenemos la extensión del archivo a partir del parámetro parámetro @p_path
   SET @p_extension = REVERSE(LEFT(REVERSE(@p_path),CHARINDEX('.',REVERSE(@p_path))-1))

   -- Si el parámetro @p_order es 1 se borra la tabla de destino
   IF @p_order = 1
      BEGIN
         --PRINT '-- Entra 1';
         SET @sql = ' DROP TABLE IF EXISTS ' + @p_destination_table

         --PRINT (@sql)
         EXEC (@sql)
      END

   /* Se genera una tabla auxiliar añadiendole al nombre de la tabla destino un HASH NEWID */

   -- Obtención del nuevo nombre de la tabla auxiliar
   SET @table = REPLACE(REPLACE(@p_destination_table, ']', ''), '[', '') + REPLACE(CAST(NEWID() AS VARCHAR(1024)), '-', '_')

   -- Se borra la tabla auxiliar en caso de existir y se obtienen todas las columnas a partir de la lectura del archivo de plantilla
   SET @sql =  'SET NOCOUNT ON
            DROP TABLE IF EXISTS '+ @table + '
            SELECT [COLUMNS]
            INTO ' + @table + '
            FROM (
               SELECT REPLACE(SUBSTRING(v_1,1,CHARINDEX(CHAR(9),v_1)), '' '', '''') + ''NVARCHAR(MAX), '' [COLUMNS]
               FROM (
                  SELECT SUBSTRING(value,(CHARINDEX(CHAR(9),[value]))+1,len([value])) V_1
                  FROM OPENROWSET( BULK ' + CHAR(39) + @p_template + CHAR(39) + ',SINGLE_BLOB) AS x
                  CROSS APPLY string_split(CAST(bulkcolumn AS VARCHAR(MAX)),char(10))
               ) TT
               WHERE V_1 != '''' AND SUBSTRING(v_1,1,CHARINDEX(CHAR(9),v_1)) != ''''
            ) T'

   --PRINT (@sql)
   EXEC (@sql)

   -- Una vez tenemos los nombres de todas las columnas se insertan en la tabla creada para ello
   SET @sql = 'SELECT Columns FROM ' + @table
   INSERT INTO @columns EXEC(@sql)

   -- Si el p_order = 1, se crea la tabla donde se almacenará el contenido de los archivos a ingestar, el nombre de la tabla es el que viene como
      parámetro, caso contrario se omite este paso
   IF @p_order = 1
      BEGIN
         --PRINT '-- Entra 2';
         SET @sql = NULL
         SELECT @sql = COALESCE(@sql + ' ', '') + [COLUMNS] FROM @columns
         -- El nombre del archivo y la fecha de ingesta se añaden en los campos ingested_filename, ingestion_date respectivamente
         SET @sql = ' CREATE TABLE ' + @p_destination_table + '(' + SUBSTRING(@sql,1,len(@sql) - 1) + ', ingested_filename VARCHAR(MAX),
      ingestion_date DATETIME) '

         --PRINT (@sql)
         EXEC (@sql)
      END

   -- Se elimina y crea la tabla auxiliar para ingestar el archivo, todos los campos se tratan como tipo NVARCHAR(MAX) para evitar desbordamientos
   SET @sql = 'DROP TABLE IF EXISTS ' + @table
   EXEC(@sql)

   SET @sql = NULL
```

```
126     SELECT @sql = COALESCE(@SQL + ' ', '') + [COLUMNS] FROM @columns
127     SET @table = REPLACE(REPLACE(@p_destination_table, ']', ''), '[', '') + REPLACE(CAST(NEWID() AS VARCHAR(1024)), '-','_')
128     SET @campos = REPLACE(@sql,'NVARCHAR(MAX)','')
129     SET @sql = ' CREATE TABLE ' + @table + '( ' + SUBSTRING(@sql,1,len(@sql) - 1) + ') '
130     EXEC (@sql)
131
132     /* Insertamos los datos del fichero según la plantilla y la extensión del archivo origen*/
133
134     -- En caso de ser un Excel se insertan los datos a partir de un OPENROWSET y usando el conector para Excel 12.0
135     IF @p_extension IN ('xlsx','xls')
136        BEGIN
137           --PRINT '-- Entra 3';
138           -- Quitamos la última coma
139           SET @campos = LEFT(trim(@campos),LEN(trim(@campos))-1)
140           SET @sql = 'INSERT INTO ' + @table + '(' + @campos + ')
141                   SELECT *
142                   FROM OPENROWSET(''Microsoft.ACE.OLEDB.12.0'', ''Excel 12.0;Database=' + @p_path + ';'', ''SELECT * FROM [Hoja1$]'') '
143
144           --PRINT (@sql)
145           EXEC (@sql)
146        END
147     ELSE
148        -- En caso que no sea un Excel se ejecuta un bulk insert estandar de SQL Server
149        BEGIN
150           --PRINT '-- Entra 4';
151
152           DECLARE @par_definition NVARCHAR(128);
153           DECLARE @total_rows INT;
154           DECLARE @sql_string NVARCHAR(512);
155           SET @sql_string = N'DECLARE @file VARBINARY(MAX)
156                       DECLARE @filetext VARCHAR(MAX)
157                       SELECT @file = CAST(bulkcolumn AS VARBINARY(MAX)), @filetext = bulkcolumn
158                                   FROM OPENROWSET (BULK ' + CHAR(39) + @p_path + CHAR(39) + ',SINGLE_BLOB) AS X
159                       SELECT @total_rows = COUNT(*) FROM string_split(@filetext,CHAR(10)) WHERE value != '''';
160           SET @par_definition = N'@total_rows INT OUTPUT';
161           EXECUTE sp_executesql @sql_string, @par_definition, @total_rows = @total_rows OUTPUT;
162
163           SET @sql = 'BULK INSERT ' + @table + '
164                    FROM ' + CHAR(39) + @p_path + CHAR(39) + '
165                    WITH (
166                       BATCHSIZE=200000,
167                       ROWS_PER_BATCH=200000,
168                       FORMATFILE = ' + CHAR(39) + @p_template + CHAR(39) + ',
169                       FIRSTROW = ' + CAST(@p_skip_initial_rows AS VARCHAR(256)) + ',
170                       LASTROW = ' + CAST(@total_rows - @p_skip_final_rows AS VARCHAR(256)) + '
171                    )'
172
173           --PRINT (@sql)
174           EXEC (@sql)
175        END
176
177     -- Se inserta una nueva fila en la tabla de log interno indicando que se han insertado los datos del archivo en la tabla auxiliar
178      SET @log = ' Datos insertados en tabla ' +  @table
179      EXEC monitor.sp_write_log @@PROCID, '', @log,'INFO'
180
181     /* Se insertan los datos en la tabla final de destino*/
182
183     -- Se obtienen todos los nombres de las columnas excepto los relativos al nombre del archivo y la fecha de carga
184     SET @sql_columns = null
185      SELECT @sql_columns = COALESCE(@sql_columns + ',','') + ' [' +  name + '] '
186      FROM syscolumns
187      WHERE id = OBJECT_ID(@p_destination_table)
188        AND [name] NOT IN ('ingested_filename', 'ingestion_date')
189
190     -- Se obtienen todos los nombres de las columnas excepto los relativos al nombre del archivo y la fecha de carga. Se forma la query que realizará
           la limpieza de los datos
191     SET @sql_columns_func = null
192      SELECT @sql_columns_func = COALESCE(@sql_columns_func + ',','') + '[input].[f_from_utf8_to_utf16](REPLACE(REPLACE(['+ name +'],''"'',' +
           ''''''),''&'',''''))) ' +  '['+ name +']'
193      FROM syscolumns
194      WHERE id = OBJECT_ID(@p_destination_table)
195        AND [name] NOT IN ('ingested_filename', 'ingestion_date')
196
197     -- Se crea una tabla auxiliar para dejar los datos ya depurados
198     DECLARE @table_aux VARCHAR(MAX)
199     SET @table_aux = REPLACE(REPLACE(@p_destination_table, ']', ''), '[', '') + REPLACE(CAST(NEWID() AS VARCHAR(1024)), '-','_')
200
201     SET @sql = 'DROP TABLE IF EXISTS ' + @table_aux + ';
202                 SELECT ' + @sql_columns_func + 'INTO ' + @table_aux + ' FROM ' + @table
203     EXEC (@sql)
204
205     -- Se insertan los datos directamente en la tabla de destino añadiendo el nombre del archivo y la fecha cuando se efectuó la carga
206     SET @sql = 'INSERT INTO ' + @p_destination_table + '(' + @sql_columns + ', ingested_filename, ingestion_date) ' +
207             'SELECT ' + @sql_columns + ',''' + @p_path + ''', GETDATE()' + ' FROM ' + @table_aux
208     EXEC (@sql)
209
210     -- Se inserta un registro en la tabla de log indicando que se han insertado los datos depurados
211     SET @log = ' Update eliminación de caracteres especiales ' + @p_destination_table + ' (' + cast(@@rowcount as varchar) + ')'
212      EXEC monitor.sp_write_log @@PROCID, '', @log,'INFO'
213
214     -- Se eliminan las tablas auxiliares
215      SET @sql = 'DROP TABLE IF EXISTS ' + @table
216      EXEC(@sql)
217     SET @sql = 'DROP TABLE IF EXISTS ' + @table_aux
218      EXEC(@sql)
219
220     -- Se inserta un registro en la tabla de log indicando que ha finalizado la ingesta del archivo
221      SET @log = 'FIN carga [input].[sp_load_file_to_table]  fichero '  + @p_path + ' orden ' + CAST(@p_order AS VARCHAR)
222      EXEC monitor.sp_write_log @@PROCID, '', @log, 'INFO'
```

```
223
224 END TRY
225
226 BEGIN CATCH
227
228    /* EN CASO DE ERROR */
229    -- Se obtienen los datos relativos al error
230    SELECT @Log = CONCAT (ErrorNumber, ' :: ' , ErrorState , ' :: ' , ErrorProcedure , ' :: ' , ErrorLine , ' :: ' , ErrorMessage)
231     FROM (
232       SELECT
233         ERROR_NUMBER() AS ErrorNumber,
234         ERROR_STATE() AS ErrorState,
235         ERROR_PROCEDURE() AS ErrorProcedure,
236         ERROR_LINE() AS ErrorLine,
237         ERROR_MESSAGE() AS ErrorMessage
238      ) aux
239
240    -- Se eliminan las tablas auxiliares
241     SET @sql = 'DROP TABLE IF EXISTS ' + @table
242     EXEC(@sql)
243    SET @sql = 'DROP TABLE IF EXISTS ' + @table_aux
244     EXEC(@sql)
245
246    -- Se inserta un registro en la tabla de log indicando el error que ha ocurrido
247     EXEC monitor.sp_write_log @@PROCID, '', @Log, 'ERROR'
248
249 END CATCH
```

Código C.21: monitor.sp_create_or_update_detail_process_log.sql

```
1  USE [IO]
2  GO
3  /*
4  -- ============================================================================================
5  Autor: Adrian Maroto
6  Fecha creación: 14/03/2024
7  Descripción: Procedimiento almacenado para crear o actualizar registros en la tabla IO.monitor.detail_process_log.
8
9  Entrada:
10     @p_master_log_process_id: ID del registro del log del proceso maestro (monitor.master_process_log).
11     @p_step: Paso del proceso ('Start' o 'End').
12     @p_in_detail_process_log_id: ID del registro (monitor.detail_process_log).
13     @p_phase: Fase del proceso.
14     @p_entity: Entidad relacionada con el proceso.
15     @p_status: Estado del proceso.
16     @p_quantity_rows: Cantidad de filas procesadas.
17     @p_error_description: Descripción del error (si corresponde).
18
19 Salida:
20     @p_out_detail_process_log_id: ID del registro de proceso detallado de salida (opcional, parámetro de salida).
21
22 Ejemplo:
23    EXEC [monitor].[sp_create_or_update_detail_process_log]
24       @p_master_log_process_id = 1,
25       @p_step = 'Start',
26       @p_phase = 'INGESTION',
27       @p_entity = 'insert_seller',
28       @p_out_detail_process_log_id = NULL
29
30    EXEC [monitor].[sp_create_or_update_detail_process_log]
31       @p_master_log_process_id = 1,
32       @p_step = 'End',
33       @p_in_detail_process_log_id = 18,
34       @p_status = 'SUCCEEDED',
35       @p_quantity_rows = 2,
36       @p_out_detail_process_log_id = NULL
37
38    SELECT * FROM [monitor].[detail_process_log]
39 -- ============================================================================================
40 */
41 CREATE OR ALTER PROCEDURE [monitor].[sp_create_or_update_detail_process_log]
42    @p_master_log_process_id INT = NULL,
43    @p_step NVARCHAR(5),
44    @p_in_detail_process_log_id INT = NULL,
45    @p_phase NVARCHAR(16) = NULL,
46    @p_entity NVARCHAR(128) = NULL,
47    @p_status NVARCHAR(9) = NULL,
48    @p_quantity_rows INT = NULL,
49    @p_error_description NVARCHAR(1024) = NULL,
50    @p_out_detail_process_log_id INT = NULL OUTPUT
51 AS
52 BEGIN
53
54    DECLARE @quantity_rows INT -- Variable para almacenar la cantidad de filas procesadas
55    DECLARE @query NVARCHAR(MAX) -- Variable para almacenar la consulta dinámica
56
57    -- Verifica el paso del proceso
58    IF @p_step = 'Start'
59       BEGIN
60          -- Insertar un nuevo registro y obtiene su ID
61          INSERT INTO IO.monitor.detail_process_log ([master_process_log_id], [phase], [entity]) VALUES (@p_master_log_process_id, @p_phase, @p_entity)
62          SELECT @p_out_detail_process_log_id = CAST(IDENT_CURRENT('[monitor].[detail_process_log]') AS INT)
63       END
64    ELSE IF @p_step = 'End'
65       BEGIN
66          -- Actualiza el registro existente con los datos proporcionados
67          UPDATE IO.monitor.detail_process_log
```

```
            SET [end_time] = GETDATE(), [status] = @p_status, [quantity_rows] = @p_quantity_rows, [error_description] = @p_error_description
         WHERE id = @p_in_detail_process_log_id
      END
   ELSE
      BEGIN
         -- Generar un error si el paso del proceso no está disponible
         RAISERROR('Process Step not available',0,1) WITH NOWAIT
      END

END
```

Código C.22: monitor.sp_create_or_update_master_process_log.sql

```
USE [IO]
GO
/*
-- ============================================================================================
Autor: Adrian Maroto
Fecha creación: 14/03/2024
Descripción: Procedimiento almacenado para crear o actualizar registros en [IO].monitor.master_process_log.

Entrada:
   @p_master_process_id: ID del proceso maestro.
   @p_in_process_log_id: ID del registro ([IO].monitor.master_process_log).
   @p_step: Paso del proceso ('Beginning' o 'End').
   @p_status: Estado de la fase.
   @p_master_process_status: Estado del proceso maestro.

Salida:
    @p_out_process_log_id: ID del registro de proceso de salida.

Ejemplo:
    EXEC [monitor].[sp_create_or_update_master_process_log]
      @p_master_process_id = 2,
      @p_step = 'Beginning',
      @p_master_process_status = 'RUNNING',
      @p_out_process_log_id = @out_id OUTPUT

   SELECT * FROM [monitor].[master_process_log]
-- ============================================================================================
*/
CREATE OR ALTER PROCEDURE [monitor].[sp_create_or_update_master_process_log]
   @p_master_process_id INT = NULL,
   @p_in_process_log_id INT = NULL,
   @p_step NVARCHAR(128),
   @p_status NVARCHAR(9) = NULL,
   @p_master_process_status NVARCHAR(9) = NULL,
   @p_out_process_log_id INT = NULL OUTPUT
AS
BEGIN

   -- Verifica el paso del proceso y realiza las acciones correspondientes
   IF @p_step = 'Beginning'
      BEGIN

         DECLARE @ingestion_flag INT
         DECLARE @transformation_flag INT
         DECLARE @persist_flag INT
         DECLARE @output_flag INT
         DECLARE @last_ingestion_status INT
         DECLARE @last_transformation_status INT
         DECLARE @last_persist_status INT
         DECLARE @last_output_status INT
         DECLARE @last_master_process_status NVARCHAR(32)

         -- Obtiene el último estado del proceso maestro
         SET @last_master_process_status = (
            SELECT TOP 1 master_process_status
            FROM IO.monitor.master_process_log
            WHERE master_process_type_id = @p_master_process_id
            ORDER BY id DESC
         )

         -- Verifica si el último estado del proceso maestro es 'FAILED'
         IF @last_master_process_status = 'FAILED'
            BEGIN
               -- Obtiene los últimos estados de los subprocesos del proceso maestro
               SELECT   TOP 1
                     @ingestion_flag = mp.ingestion_flag,
                     @transformation_flag = mp.transformation_flag,
                     @persist_flag = mp.persist_flag,
                     @output_flag = mp.output_flag,
                     @last_ingestion_status = CASE WHEN ml.ingestion_status = 'SUCCEEDED' THEN 1 ELSE 0 END,
                     @last_transformation_status = CASE WHEN ml.transformation_status = 'SUCCEEDED' THEN 1 ELSE 0 END,
                     @last_persist_status = CASE WHEN ml.persistence_status = 'SUCCEEDED' THEN 1 ELSE 0 END,
                     @last_output_status =CASE WHEN  ml.output_status = 'SUCCEEDED' THEN 1 ELSE 0 END
               FROM IO.config.master_process_type mp
               INNER JOIN IO.monitor.master_process_log ml
               ON mp.id = ml.master_process_type_id
               WHERE mp.id = @p_master_process_id
               ORDER BY ml.id DESC
            END
         ELSE
            BEGIN
               -- Inserta un nuevo registro en master_process_log
               INSERT INTO IO.monitor.master_process_log ([master_process_type_id]) VALUES (@p_master_process_id)
               SELECT @p_out_process_log_id = CAST(IDENT_CURRENT('[monitor].[master_process_log]') AS INT)
```

```
85                RETURN
86             END
87
88          -- Verifica si los estados de los subprocesos coinciden con los anteriores
89          IF (@ingestion_flag = @last_ingestion_status
90             AND @transformation_flag = @last_transformation_status
91             AND @persist_flag = @last_persist_status
92             AND @output_flag = @last_output_status)
93             BEGIN
94                -- Inserta un nuevo registro en master_process_log si coinciden los estados
95                INSERT INTO IO.monitor.master_process_log ([master_process_type_id]) VALUES (@p_master_process_id)
96             END
97          ELSE
98             BEGIN
99                -- Inserta un nuevo registro en master_process_log copiando el último registro
100               INSERT INTO IO.monitor.master_process_log
101               SELECT TOP 1 [master_process_type_id], [master_process_start_time], [master_process_end_time], [master_process_status],
      [ingestion_start_time], [ingestion_end_time], [ingestion_status], [transformation_start_time], [transformation_end_time],
      [transformation_status], [persistence_start_time], [persistence_end_time], [persistence_status], [output_start_time], [output_end_time],
      [output_status], [error_description]
102               FROM IO.monitor.master_process_log WHERE master_process_type_id = @p_master_process_id ORDER BY id DESC
103            END
104
105         -- Obtiene el ID del nuevo registro de master_process_log
106         SELECT @p_out_process_log_id = CAST(IDENT_CURRENT('[monitor].[master_process_log]') AS INT)
107      END
108   -- Actualiza las fases del proceso maestro según el paso proporcionado
109   ELSE IF @p_step = 'Start_ingestion'
110      BEGIN
111         -- Inicia el tiempo de la ingesta
112         UPDATE IO.monitor.master_process_log
113         SET [ingestion_start_time] = GETDATE()
114         WHERE id = @p_in_process_log_id
115      END
116      ELSE IF @p_step = 'End_ingestion'
117      BEGIN
118         -- Finaliza el tiempo de la ingesta y establece el estado
119         UPDATE IO.monitor.master_process_log
120         SET [ingestion_end_time] = GETDATE(), [ingestion_status] = @p_status
121         WHERE id = @p_in_process_log_id
122      END
123      ELSE IF @p_step = 'Start_transformation'
124      BEGIN
125         -- Inicia el tiempo de la transformación
126         UPDATE IO.monitor.master_process_log
127         SET [transformation_start_time] = GETDATE()
128         WHERE id = @p_in_process_log_id
129      END
130      ELSE IF @p_step = 'End_transformation'
131      BEGIN
132         -- Finaliza el tiempo de la transformación y establece el estado
133         UPDATE IO.monitor.master_process_log
134         SET [transformation_end_time] = GETDATE(), [transformation_status] = @p_status
135         WHERE id = @p_in_process_log_id
136      END
137      ELSE IF @p_step = 'Start_persistence'
138      BEGIN
139         -- Inicia el tiempo de la persistencia
140         UPDATE IO.monitor.master_process_log
141         SET [persistence_start_time] = GETDATE()
142         WHERE id = @p_in_process_log_id
143      END
144      ELSE IF @p_step = 'End_persistence'
145      BEGIN
146         -- Finaliza el tiempo de la persistencia y establece el estado
147         UPDATE IO.monitor.master_process_log
148         SET [persistence_end_time] = GETDATE(), [persistence_status] = @p_status
149         WHERE id = @p_in_process_log_id
150      END
151      ELSE IF @p_step = 'Start_output'
152      BEGIN
153         -- Inicia el tiempo de la salida
154         UPDATE IO.monitor.master_process_log
155         SET [output_start_time] = GETDATE()
156         WHERE id = @p_in_process_log_id
157      END
158      ELSE IF @p_step = 'End_output'
159      BEGIN
160         -- Finaliza el tiempo de la salida y establece el estado
161         UPDATE IO.monitor.master_process_log
162         SET [output_end_time] = GETDATE(), [output_status] = @p_status
163         WHERE id = @p_in_process_log_id
164      END
165      ELSE IF @p_step = 'End'
166      BEGIN
167         -- Finaliza el tiempo del proceso maestro y establece el estado del proceso maestro
168         UPDATE IO.monitor.master_process_log
169         SET [master_process_end_time] = GETDATE(), [master_process_status] = @p_master_process_status
170         WHERE id = @p_in_process_log_id
171      END
172      ELSE
173      BEGIN
174         -- Genera un mensaje de error si el paso del proceso no está disponible
175         RAISERROR('Process Step not available',0,1) WITH NOWAIT
176      END
177
178   END
```

Código C.23: monitor.sp_get_execution_summary.sql

```sql
USE [IO]
GO
/*
-- ============================================================================================
Autor: Adrian Maroto
Fecha creación: 14/03/2024
Descripción: Procedimiento almacenado para obtener un resumen de la ejecución de procesos.

Entrada:
    @master_process_type_id: ID del proceso maestro.
    @master_process_log_id: ID del log de registro del proceso maestro.

Salida:

Ejemplo:
   EXEC [monitor].[sp_get_execution_summary]
      @master_process_type_id = 5,
      @master_process_log_id = 1

   SELECT * FROM [IO].config.master_process_type
   SELECT * FROM [IO].monitor.master_process_log
-- ============================================================================================
*/
CREATE OR ALTER PROCEDURE [monitor].[sp_get_execution_summary]
   @master_process_type_id INT,
   @master_process_log_id INT
AS
BEGIN

   DROP TABLE IF EXISTS #process_execution

   -- Crea una tabla temporal para almacenar el resumen de la ejecución del proceso
   SELECT dpl.id AS id,
      mpt.customer AS cartera,
      mpt.[name] AS process_name,
      mpt.[description] AS process_description,
      mpl.master_process_status AS process_status,
      mpl.master_process_start_time AS process_start_time,
      mpl.master_process_end_time AS process_end_time,
      mpl.error_description AS process_message,
      mpl.ingestion_status AS ingestion_status,
      mpl.ingestion_start_time AS ingestion_start_time,
      mpl.ingestion_end_time AS ingestion_end_time,
      mpl.ingestion_end_time - mpl.ingestion_start_time AS ingestion_duration,
      mpl.transformation_status AS transformation_status,
      mpl.transformation_start_time AS transformation_start_time,
      mpl.transformation_end_time AS transformation_end_time,
      mpl.transformation_end_time - mpl.transformation_start_time AS transformation_duration,
      mpl.persistence_status AS persistence_status,
      mpl.persistence_start_time AS persistence_start_time,
      mpl.persistence_end_time AS persistence_end_time,
      mpl.persistence_end_time - mpl.persistence_start_time AS persistence_duration,
      mpl.output_status AS output_status,
      mpl.output_start_time AS output_start_time,
      mpl.output_end_time AS output_end_time,
      mpl.output_end_time - mpl.output_start_time AS output_duration,
      dpl.phase AS phase,
      dpl.entity AS entity,
      dpl.[status] AS phase_entity_status,
      dpl.quantity_rows AS phase_entity_affected_rows,
      dpl.start_time AS phase_entity_start_time,
      dpl.end_time AS phase_entity_end_time,
      CONVERT(NVARCHAR, dpl.end_time - dpl.start_time, 114) AS phase_entity_duration,
      dpl.error_description AS phase_entity_message
   INTO #process_execution
   FROM [IO].config.master_process_type mpt
      INNER JOIN [IO].monitor.master_process_log mpl ON mpt.id = mpl.master_process_type_id
      INNER JOIN [IO].monitor.detail_process_log dpl ON mpl.id = dpl.master_process_log_id
   WHERE mpt.id = @master_process_type_id
      AND mpl.id = @master_process_log_id

   -- Devuelve el resumen de la ejecución del proceso
   SELECT
      cartera,
      process_name,
      process_description,
      process_status,
      process_start_time,
      process_end_time,
      CONVERT(NVARCHAR, (COALESCE(ingestion_duration,
         '1900-01-01 00:00:00.000')
         + COALESCE(transformation_duration,
         '1900-01-01 00:00:00.000')
         + COALESCE(persistence_duration,
         '1900-01-01 00:00:00.000')
         + COALESCE(output_duration,
         '1900-01-01 00:00:00.000') ), 114)
      AS process_duration,
      process_message,
      ingestion_status,
      ingestion_start_time,
      ingestion_end_time,
      CONVERT(NVARCHAR, ingestion_duration, 114) AS ingestion_duration,
      CASE
         WHEN (SELECT COUNT(*) FROM #process_execution WHERE phase_entity_status = 'WARNING') > 0
            AND transformation_status IS NOT NULL AND transformation_status != 'N/A'
         THEN 'WARNING'
```

```
        ELSE transformation_status
    END AS transformation_status,
    transformation_start_time,
    transformation_end_time,
    CONVERT(NVARCHAR, transformation_duration, 114) AS transformation_duration,
    persistence_status,
    persistence_start_time,
    persistence_end_time,
    CONVERT(NVARCHAR, persistence_duration, 114) AS persistence_duration,
    output_status,
    output_start_time,
    output_end_time,
    CONVERT(NVARCHAR, output_duration, 114) AS output_duration,
    phase,
    entity,
    phase_entity_status,
    phase_entity_affected_rows,
    phase_entity_start_time,
    phase_entity_end_time,
    phase_entity_duration,
    phase_entity_message
 FROM #process_execution
 ORDER BY id ASC

END
```

Código C.24: monitor.sp_write_log.sql

```
USE [IO]
GO
/*
-- ============================================================================================
Autor: Adrian Maroto
Fecha creación: 14/03/2024
Descripción: Procedimiento para guardar traza de log en [IO].monitor.log.

Entrada:
    @ProcId: ID del proceso.
    @TraceId: ID de seguimiento.
    @Message: Mensaje a registrar.
    @Severity: Severidad del mensaje ('ERROR' o cualquier otro valor).

Salida:

Ejemplo:
   EXEC [monitor].[sp_write_log]
      @ProcId = 71,
      @TraceId = 5,
      @Message = 'Se ha producido un error en el proceso.',
      @Severity = 'INFO'

   SELECT * FROM [IO].monitor.log
-- ============================================================================================
*/
CREATE OR ALTER PROCEDURE [monitor].[sp_write_log]
    @ProcId INT,
    @TraceId NVARCHAR(100),
    @Message NVARCHAR(MAX),
    @Severity NVARCHAR(50)
AS
BEGIN

    -- Variables locales
    DECLARE @MessageToRaise NVARCHAR(MAX)
    DECLARE @Source NVARCHAR(200) = COALESCE (
      QUOTENAME(OBJECT_SCHEMA_NAME(@ProcID, DB_ID())) + '.' +
        QUOTENAME(OBJECT_NAME(@ProcID, DB_ID())),
        ERROR_PROCEDURE()
  );

    -- Verifica si se ha proporcionado un origen para el mensaje
  SET @Source = ISNULL(@Source,'')

    -- Construye el mensaje con la marca de tiempo
  SET @MessageToRaise = CONVERT(NVARCHAR, CURRENT_TIMESTAMP, 113) + ' :: ' + @Message

  -- Verifica la severidad del mensaje
  IF @Severity = 'ERROR'
     BEGIN
        -- Reollback si hay una transacción activa o el estado de la transacción es -1
        IF (@@TRANCOUNT > 0 OR XACT_STATE() = -1) ROLLBACK TRANSACTION
        -- Levanta una excepción con el mensaje
        RAISERROR(@MessageToRaise,16,1)
     END
  ELSE
     -- Imprime el mensaje sin levantar una excepción
     BEGIN
        RAISERROR(@MessageToRaise, 0, 1) WITH NOWAIT
     END

    -- Inserta el registro en la tabla de registro
    INSERT INTO monitor.log (created_at, [user], [source], pid, trace_id, [message], severity)
    VALUES (CURRENT_TIMESTAMP, CURRENT_USER, @Source, @@SPID, ISNULL(@TraceId, ''), @Message, @Severity)

END
```

La *Figura C.5* muestra el conjunto de funciones SQL de la BBDD IO.

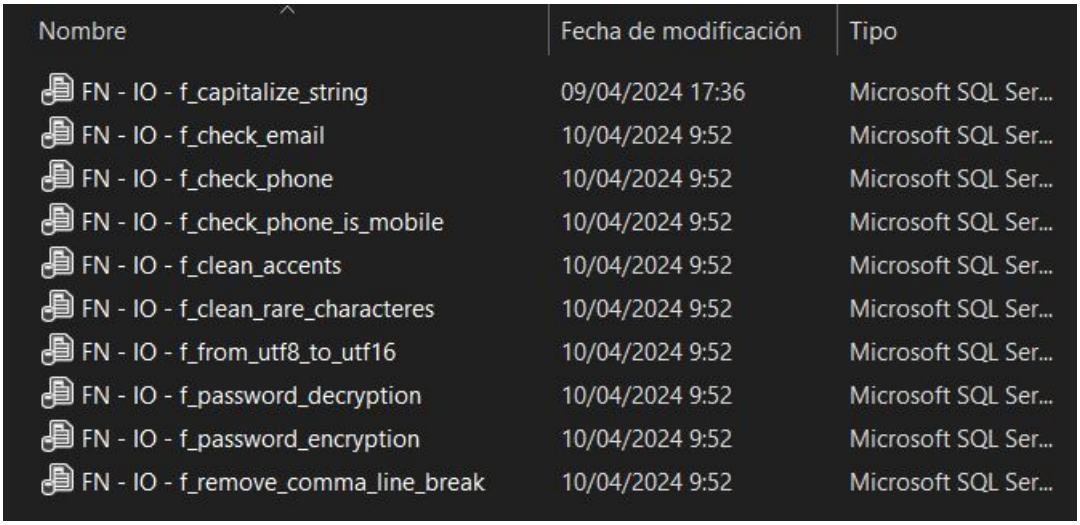


Figura C.5: Estructura de funciones SQL de la BBDD IO

A continuación el *Código C.25-C.34* muestra las funciones SQL de la BBDD IO.

Código C.25: f_capitalize_string.sql

```sql
/*
-- ====================================================================================
Autor: Adrian Maroto
Fecha creacion: 13/03/2024
Descripcion: Capitaliza una cadena.
Cambios:

Entrada:
   @input_string: Cadena a convertir.

Salida:
   RETURN: Cadena convertida.

Ejemplo:
   SELECT [input].[f_capitalize_string]('ADRIÁN MAROTO')
   SELECT [input].[f_capitalize_string]('PRUEBA MAYUSCULA')
   SELECT [input].[f_capitalize_string]('otro test')
   SELECT [input].[f_capitalize_string](name) salida, * FROM VehicleSales.dbo.concessionaire
-- ====================================================================================
*/
CREATE OR ALTER FUNCTION [input].[f_capitalize_string] (@input_string VARCHAR(8000))
RETURNS VARCHAR(8000)
AS
BEGIN
   DECLARE @output_string VARCHAR(8000) = LOWER(@input_string),
      @char CHAR(1),
      @alphanum BIT = 0,
      @len INT = LEN(@input_string),
      @pos INT = 1;

   -- Iterar entre todos los caracteres en la cadena de entrada
   WHILE @pos <= @len
   BEGIN
      -- Obtener el siguiente caracter
      SET @char = SUBSTRING(@input_string, @pos, 1);

      -- Si la posición del caracter es la 1a, o el caracter previo no es alfanumérico
      -- convierte el caracter actual a mayúscula
      IF @pos = 1 OR @alphanum = 0
         SET @output_string = STUFF(@output_string, @pos, 1, UPPER(@char));

      SET @pos = @pos + 1;

      -- Define si el caracter actual es non-alfanumérico
      IF ASCII(@char) <= 47 OR (ASCII(@char) BETWEEN 58 AND 64) OR
      (ASCII(@char) BETWEEN 91 AND 96) OR (ASCII(@char) BETWEEN 123 AND 126)
         SET @alphanum = 0;
      ELSE
         SET @alphanum = 1;
   END

   RETURN @output_string;

END
```

Código C.26: f_check_email.sql

```sql
/*
-- ====================================================================================
Autor: Adrian Maroto
Fecha creacion: 13/03/2024
Descripcion: Función que retorna true si el parámetro de entrada es un correo válido, en caso contrario retorna false.
Cambios:

Entrada:
   @email: Correo a revisar.

Salida:
   RETURN: 1(es correo)/0(no es correo).

Ejemplo:
   SELECT [input].[f_check_email]('amaroto42@alumno.uned.es') salida
   SELECT [input].[f_check_email]('amaroto42alumno.uned.es') salida
   SELECT [input].[f_check_email]('[]:>@gmail.es') salida
   SELECT [input].[f_check_email]('test@g!!mail.es') salida
   SELECT [input].[f_check_email]('test@outlook') salida
   SELECT [input].[f_check_email](email) salida, * FROM VehicleSales.dbo.seller
-- ====================================================================================
*/
CREATE OR ALTER FUNCTION [input].[f_check_email] (@email VARCHAR(1024))
RETURNS BIT
AS
BEGIN

   DECLARE @result BIT;

   SET @email = LTRIM(RTRIM(ISNULL(@email,'')))

   -- Comprueba espacios, caracteres no permitidos, @, dominio, etc.
   SET @result = CASE
      WHEN @email = '' THEN 0
      WHEN @email LIKE '% %' THEN 0
      WHEN @email LIKE ('%["(),:<>\]%') THEN 0
      WHEN SUBSTRING(@email,CHARINDEX('@',@email),LEN(@email)) LIKE ('%[!#$%&*+/=^`{|]%') THEN 0
      WHEN (LEFT(@email,1) LIKE ('[-_+]') OR RIGHT(@email,1) LIKE ('[-_+]')) THEN 0
      WHEN (@email LIKE '%[%' OR @email LIKE '%]%') THEN 0
      WHEN @email NOT LIKE '_%@_%._%' THEN 0
      ELSE 1
   END

   RETURN @result

END
```

Código C.27: f_check_phone_is_mobile.sql

```sql
/*
-- ====================================================================================
Autor: Adrian Maroto
Fecha creacion: 13/03/2024
Descripcion: Retorna true si el parámetro de entrada es un teléfono móvil, en caso contrario retorna false.
Cambios:

Entrada:
   @phone: Teléfono a revisar.

Salida:
   RETURN: 1(es móvil)/0(no es móvil).

Ejemplo:
   SELECT [input].[f_check_phone_is_mobile]('654233144') salida
   SELECT [input].[f_check_phone_is_mobile]('918883322') salida
   SELECT [input].[f_check_phone_is_mobile]('65432109876') salida
   SELECT [input].[f_check_phone_is_mobile](phone) salida, * FROM VehicleSales.dbo.customer
-- ====================================================================================
*/
CREATE OR ALTER FUNCTION [input].[f_check_phone_is_mobile] (@phone VARCHAR(256))
RETURNS BIT
AS
BEGIN

   DECLARE @result BIT = 0;

   -- Si el teléfono empieza por 6-7 y tiene una longitud de 9 dígitos devuelve true
   IF(@phone LIKE '[6-7]%' AND LEN(@phone) = 9)
      SET @result = 1
   ELSE
      SET @result = 0

   RETURN @result

END
```

Código C.28: f_check_phone.sql

```sql
/*
-- ====================================================================================
Autor: Adrian Maroto
Fecha creacion: 13/03/2024
Descripcion: Función que retorna true si el parámetro de entrada es un teléfono válido, en caso contrario retorna false.
```

```
Cambios:

Entrada:
   @phone: Teléfono a revisar.

Salida:
   RETURN: 1(es teléfono)/0(no es teléfono).

Ejemplo:
   SELECT [input].[f_check_phone]('654233144') salida
   SELECT [input].[f_check_phone]('918883322') salida
   SELECT [input].[f_check_phone]('65432109876') salida
   SELECT [input].[f_check_phone](phone) salida, * FROM VehicleSales.dbo.customer
-- ============================================================================================
*/
CREATE OR ALTER FUNCTION [input].[f_check_phone] (@phone VARCHAR(256))
RETURNS BIT
AS
BEGIN

   DECLARE @result BIT = 1;

   SET @phone = REPLACE(@phone,' ','')

   -- Si el teléfono empieza de 6-9 y tiene una longitud de 9 dígitos devuelve true
   SET @result = CASE
      WHEN (@phone < '600000000') THEN 0
      WHEN (@phone > '989000000') THEN 0
      WHEN (LEN(@phone) > 9) THEN 0
      WHEN (LEN(@phone) < 9) THEN 0
      WHEN @phone IS NULL THEN 0
      ELSE @result
   END

   RETURN @result

END
```

Código C.29: f_clean_accents.sql

```
/*
-- ============================================================================================
Autor: Adrian Maroto
Fecha creacion: 13/03/2024
Descripcion: Elimina los acentos de la cadena especificada.
Cambios:

Entrada:
   @input_string: Cadena a convertir.

Salida:
   RETURN: Cadena sin acentos.

Ejemplo:
   SELECT [auxiliar].[f_clean_accents]('ADRIÁN MAROTO')
   SELECT [auxiliar].[f_clean_accents]('Adrián')
   SELECT [auxiliar].[f_clean_accents]('España y ÇÇÇ')
   SELECT [auxiliar].[f_clean_accents](name) salida, * FROM VehicleSales.dbo.customer
-- ============================================================================================
*/
CREATE OR ALTER FUNCTION [auxiliar].[f_clean_accents] (@input_string VARCHAR(MAX))
RETURNS VARCHAR(MAX)
AS
BEGIN

   -- Reemplazamos las vocales acentuadas
   -- áéíóúàèìòùÁÉÍÓÚÀÈÌÒÙ
    RETURN
   REPLACE(REPLACE(REPLACE(REPLACE(REPLACE( /*vocales áéíóú*/
   REPLACE(REPLACE(REPLACE(REPLACE(REPLACE( /*vocales àèìòù*/
   REPLACE(REPLACE(REPLACE(REPLACE(REPLACE( /*vocales ÁÉÍÓÚ*/
   REPLACE(REPLACE(REPLACE(REPLACE(REPLACE( /*vocales ÀÈÌÒÙ*/
   @input_string COLLATE SQL_Latin1_General_CP1_CS_AS
      ,'á','a'),'é','e'),'í','i'),'ó','o'),'ú','u')
      ,'à','a'),'è','e'),'ì','i'),'ò','o'),'ù','u')
      ,'Á','A'),'É','E'),'Í','I'),'Ó','O'),'Ú','U')
      ,'À','A'),'È','E'),'Ì','I'),'Ò','O'),'Ù','U')

END
```

Código C.30: f_clean_rare_characters.sql

```
/*
-- ============================================================================================
Autor: Adrian Maroto
Fecha creacion: 13/03/2024
Descripcion: Elimina los acentos y caracteres especiales (incluyendo Ñ) de la cadena especificada.
Cambios:

Entrada:
   @input_string: Cadena a convertir.

Salida:
   RETURN: Cadena sin acentos y caracteres especiales.

```

```
Ejemplo:
   SELECT [auxiliar].[f_clean_rare_characters]('ADRIÁN MAROTO')
   SELECT [auxiliar].[f_clean_rare_characters]('Adrián')
   SELECT [auxiliar].[f_clean_rare_characters]('España')
   SELECT [auxiliar].[f_clean_rare_characters](name) salida, * FROM VehicleSales.dbo.customer
-- ============================================================================================
*/
CREATE OR ALTER FUNCTION [auxiliar].[f_clean_rare_characters] (@input_string VARCHAR(MAX))
RETURNS VARCHAR(MAX)
AS
BEGIN

   -- Reemplazamos las vocales acentuadas y caracteres especiales
   -- áéíóúàèìòùãõâêîôûäëïöüñÑçÇÁÉÍÓÚÀÈÌÒÙÃÕÂÊÎÔÛÄËÏÖÜ
    RETURN
   REPLACE(REPLACE(REPLACE( /*vocales ÃÕ*/
   REPLACE(REPLACE(REPLACE(REPLACE(REPLACE( /*vocales ÄËÏÖÜ*/
   REPLACE(REPLACE(REPLACE(REPLACE(REPLACE( /*vocales ÂÊÎÔÛ*/
   REPLACE(REPLACE(REPLACE(REPLACE(REPLACE( /*vocales ÀÈÌÒÙ*/
   REPLACE(REPLACE(REPLACE(REPLACE(REPLACE( /*vocales ÁÉÍÓÚ*/
   REPLACE(REPLACE(REPLACE(REPLACE(/*vocales ñÑçÇ*/
   REPLACE(REPLACE(REPLACE(REPLACE(REPLACE( /*vocales äëïöü*/
   REPLACE(REPLACE(REPLACE(REPLACE(REPLACE( /*vocales âêîôû*/
   REPLACE(REPLACE(REPLACE(REPLACE(REPLACE( /*vocales àèìòù*/
   REPLACE(REPLACE(REPLACE(REPLACE( /*vocales áéíóú*/
   @input_string COLLATE SQL_Latin1_General_CP1_CS_AS
      ,'á','a'),'é','e'),'í','i'),'ó','o'),'ú','u')
      ,'à','a'),'è','e'),'ì','i'),'ò','o'),'ù','u')
      ,'â','a'),'ê','e'),'î','i'),'ô','o'),'û','u')
      ,'ä','a'),'ë','e'),'ï','i'),'ö','o'),'ü','u')
      ,'ñ','n'),'Ñ','N'),'ç','c'),'Ç','C')
      ,'Á','A'),'É','E'),'Í','I'),'Ó','O'),'Ú','U')
      ,'À','A'),'È','E'),'Ì','I'),'Ò','O'),'Ù','U')
      ,'Â','A'),'Ê','E'),'Î','I'),'Ô','O'),'Û','U')
      ,'Ä','A'),'Ë','E'),'Ï','I'),'Ö','O'),'Ü','U')
      ,'Ã','A'),'Õ','O')

END
```

Código C.31: f_from_utf8_to_utf16.sql

```
/*
-- ============================================================================================
Autor: Adrian Maroto
Fecha creacion: 13/03/2024
Descripcion: Inserta una cadena de texto en UTF-8 y los convierte en su equivalente en UTF-16.
Cambios:

Entrada:
   @input_string: Cadena a convertir en UTF-8.

Salida:
   RETURN: Cadena convertida a UTF-16.

Ejemplo:
   SELECT [input].[f_from_utf8_to_utf16]('Ejemplo con caracteres especiales: áéíóúñ') salida;
   SELECT [input].[f_from_utf8_to_utf16]('Sin caraceteres especiales') salida;
-- ============================================================================================
*/
CREATE OR ALTER FUNCTION [input].[f_from_utf8_to_utf16] (@input_string VARCHAR(8000))
RETURNS VARCHAR(8000)
AS
BEGIN

   DECLARE @c CHAR,
      @i INT;

    SELECT @i = PATINDEX('%[ÂÃ][!?]%', @input_string COLLATE SQL_Latin1_General_CP1_CI_AS)

   -- Usamos PATINDEX para encontrar la posición del primer carácter especial en la cadena de entrada.
   -- Extraemos el carácter y calculamos su equivalente en UTF-16.
   WHILE @i > 0
   BEGIN
        SELECT  @c = CHAR(((ASCII(SUBSTRING(@input_string, @i, 1)) & 31) * 64)
                        + (ASCII(SUBSTRING(@input_string, @i + 1, 1)) & 63)),
               @input_string = STUFF(@input_string, @i, 2, @c),
               @i = PATINDEX('%[ÂÃ][!?]%', @input_string COLLATE SQL_Latin1_General_CP1_CI_AS)
   END

    RETURN @input_string

END
```

Código C.32: f_password_decryption.sql

```
/*
-- ============================================================================================
Autor: Adrian Maroto
Fecha creacion: 13/03/2024
Descripcion: Desencripta una cadena encriptada. Necesario abrir primero las claves de cifrado.
Cambios:

Entrada:
   @value_to_decrypt: Cadena a desencriptar.

```

```
Salida:
   RETURN: Cadena desencriptada.

Ejemplo:
   -- Apertura de las claves de cifrado, siempre se ha de hacer antes de encriptar o desencriptar
   EXEC [config].[sp_open_io_password_encryption_keys]
   -- Funcion de encriptación, entrada un VARCHAR, salida un VARBINBARY
   SELECT [config].[f_password_encryption]('P4$$WOrd_unico')
   -- Función de desencriptación, entra un VARBINARY, salida un VARCHAR
   SELECT
      [config].[f_password_decryption](0x00BA1DC5F52ACF05CEF7C5F899A696B302000000B16EB65E085C7E2EC39AC7B8C51300496D53D80945989550CE140474318D2116D3F398E8AE95AB406F24A27DD
-- ============================================================================================
*/
CREATE OR ALTER FUNCTION [config].[f_password_decryption] (@value_to_decrypt VARBINARY(256))
RETURNS VARCHAR(MAX)
AS
BEGIN

    DECLARE @result VARCHAR(MAX)

    SET @Result = DECRYPTBYKEY(@value_to_decrypt)

    -- Devuelve el resultado de la función
    RETURN @result

END
```

Código C.33: f_password_encryption.sql

```
/*
-- ============================================================================================
Autor: Adrian Maroto
Fecha creacion: 13/03/2024
Descripcion: Encripta una cadena desencriptada. Necesario abrir primero las claves de cifrado.
Cambios:

Entrada:
   @value_to_encrypt: Cadena a encriptar.

Salida:
   RETURN: Cadena encriptada.

Ejemplo:
   -- Apertura de las claves de cifrado, siempre se ha de hacer antes de encriptar o desencriptar
   EXEC [config].[sp_open_io_password_encryption_keys]
   -- Funcion de encriptación, entrada un VARCHAR, salida un VARBINBARY
   SELECT [config].[f_password_encryption]('P4$$WOrd_unico')
   -- Función de desencriptación, entra un VARBINARY, salida un VARCHAR
   SELECT
      [config].[f_password_decryption](0x00BA1DC5F52ACF05CEF7C5F899A696B302000000B16EB65E085C7E2EC39AC7B8C51300496D53D80945989550CE140474318D2116D3F398E8AE95AB406F24A27DD
-- ============================================================================================
*/
CREATE OR ALTER FUNCTION [config].[f_password_encryption] (@value_to_encrypt VARCHAR(MAX))
RETURNS VARBINARY(256)
AS
BEGIN

    DECLARE @Result VARBINARY(256)

    SET @Result = ENCRYPTBYKEY(KEY_GUID('password_io_simetric_key'), @value_to_encrypt)

   -- Devuelve el resultado de la función
    RETURN @Result
END
```

Código C.34: f_remove_comma_line_break.sql

```
/*
-- ============================================
-- Author:     Data
-- Create date: 2022-03-30
-- Description:   Elimina la coma y el carácter de salto de línea
-- ============================================
*/

/*
-- ============================================================================================
Autor: Adrian Maroto
Fecha creacion: 13/03/2024
Descripcion: Elimina la coma y el carácter de salto de línea.
Cambios:

Entrada:
   @input_string: Cadena a convertir.

Salida:
   RETURN: Cadena sin coma y carácter de salto de línea.

Ejemplo:
   SELECT [auxiliar].[f_remove_comma_line_break]('P,r,u,e,b,a,,,')
   SELECT [auxiliar].[f_remove_comma_line_break]('Salto de linea
   ')
   SELECT [auxiliar].[f_remove_comma_line_break](address) salida, * FROM VehicleSales.dbo.concessionaire
-- ============================================================================================
*/
```

```sql
CREATE OR ALTER FUNCTION [auxiliar].[f_remove_comma_line_break] (@input_string VARCHAR(MAX))
RETURNS VARCHAR(MAX)
AS
BEGIN

      -- Reemplazamos las vocales acentuadas y caracteres especiales
      RETURN
      REPLACE(REPLACE(REPLACE(
      @input_string COLLATE SQL_Latin1_General_CP1_CS_AS
         ,',',',')
         ,CHAR(10),'')
         ,CHAR(13),'')

END
```

La *Figura C.6* muestra el conjunto de scripts de creación SQL de casos de uso del proyecto IO.

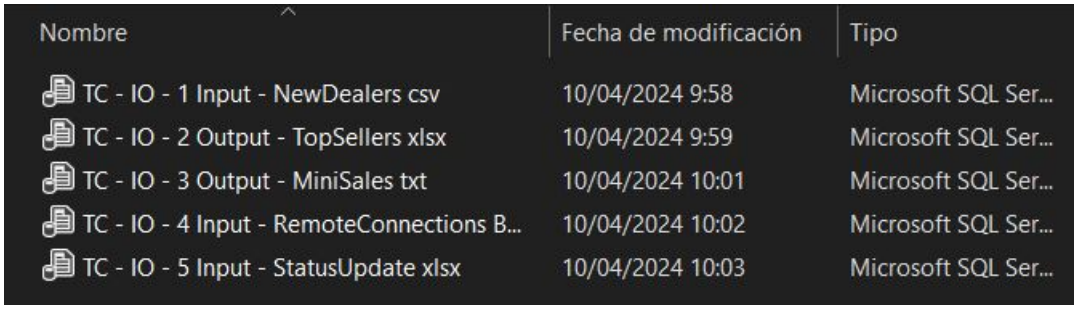


Figura C.6: Estructura de scripts de creación SQL de los casos de uso del proyecto IO

A continuación se muestra el *Código C.35* con la creación SQL para el caso de uso 1 Input NewDealers.

Código C.35: 1 Input NewDealers csv.sql

```sql
/*
IMPORTANTE: Antes de empezar es necesario generar tanto el fichero a tratar como la plantilla
-- ================================
-- ========== Tablas base ==========
-- ================================
*/
USE [IO]
GO

BEGIN TRAN

-- [IO].[config].[master_process_type]
INSERT INTO [IO].[config].[master_process_type]
   ([id],[customer],[name],[description],[active_flag],[ingestion_flag],[transformation_flag],[persist_flag],[output_flag])
VALUES
    (1, 'BMW', 'NewDealers_Input', 'Incorpora los nuevos concesionarios y actualiza los existentes', 1, 1, 1, 1, 0)

-- [IO].[config].[detail_process_type]
-- Apertura de las claves de cifrado, siempre se ha de hacer antes de encriptar o desencriptar
EXEC [IO].config.sp_open_io_password_encryption_keys

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type],[destination_table],[ftp_server],[ftp_port],[ftp_user]
   ,[ftp_pass],[ftp_operation],[ftp_folder],[ftp_file_pattern],[ftp_local_folder])
VALUES
   (100, 1, 'INGESTION', 1, 'insert_BMW_NewDealers', 'Ingesta NewDealers', 'Recoge del FTP y mueve al Input el fichero BMW_NewDealers', 'FTP',
      'FTP_Download', 'data.es', 21, 'ftpuser'
   ,(SELECT [IO].config.f_password_encryption('Usuario2011')), 'DOWNLOAD', '/BMW_Group/Concesionarios/Entrada/', 'BMW_NewDealers*.csv',
      '\\WIN-SER-AMAROTO\IO_Files\Input\BMW_Group\Concesionarios')

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type],[source_file_folder_path],[source_file_pattern],[add_eof]
   ,[skip_final_rows],[skip_initial_rows],[source_file_template_path],[destination_table],[check_ingestion_filename])
VALUES
   (110, 1, 'INGESTION', 2, 'load_file_input_table', 'concessionaire', 'Archivos de concessionaire', 'FILE',
      '\\WIN-SER-AMAROTO\IO_Files\Input\BMW_Group\Concesionarios', 'BMW_NewDealers*.csv', 0
   , 0, 1, '\\WIN-SER-AMAROTO\IO_Files\Template\BMW_Group\Input_NewDealers.fmt', '[IO].[input].[BMW_NewDealers]', 1)

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[source_table_query])
VALUES
   (120, 1, 'DELIVERY', 1, 'new_delivery', 'Alta delivery concessionaire', 'Alta de una nueva delivery de concessionaire por cada marca', 'BBDD'
   , 'EXEC [IO].[input].[sp_BMW_NewDealers_new_delivery] @master_id = ?, @master_log_id = ?')

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[source_table_query])
VALUES
```

```
   (130, 1, 'TRANSFORMATION', 1, 'insert_concessionaire', 'Insert load concessionaire', 'Incorporación de los datos calculando las operaciones', 'BBDD'
   , 'EXEC [IO].[input].[sp_BMW_NewDealers_transformation_insert_concessionaire] @master_process_id = ?')

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[source_table_query])
VALUES
   (140, 1, 'TRANSFORMATION', 2, 'update_concessionaire', 'Update load concessionaire', 'Incorporación de los datos calculando las operaciones', 'BBDD'
   , 'EXEC [IO].[input].[sp_BMW_NewDealers_transformation_update_concessionaire] @master_process_id = ?')

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[source_table_query])
VALUES
   (150, 1, 'PERSISTENCE', 1, 'insert_concessionaire', 'Insert VehicleSales concessionaire', 'Incorporación de los datos en destino insert
      concessionaire', 'BBDD'
   , 'EXEC [IO].[input].[sp_BMW_NewDealers_persistence_insert_concessionaire] @delivery_id = ?, @affected_rows = ? OUTPUT')

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[source_table_query])
VALUES
   (160, 1, 'PERSISTENCE', 2, 'update_concessionaire', 'Update VehicleSales concessionaire', 'Incorporación de los datos en destino update
      concessionaire', 'BBDD'
   , 'EXEC [IO].[input].[sp_BMW_NewDealers_persistence_update_concessionaire] @delivery_id = ?, @affected_rows = ? OUTPUT')

------------------------------------------------------------------------

-- [IO].[load].[concessionaire]
CREATE TABLE [IO].[load].[concessionaire](
   id INT IDENTITY(1,1) NOT NULL,
   delivery_id INT NOT NULL,
   mdb_id INT NULL,
   [name] NVARCHAR(128),
    [address] NVARCHAR (512),
   city NVARCHAR(64),
    country NVARCHAR(32),
   postal_code NVARCHAR(18),
   info NVARCHAR(512),
   active BIT,
   creation_date DATETIME,
   modification_date DATETIME
PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [IO].[load].[concessionaire] WITH CHECK ADD  CONSTRAINT [fk_concessionaire_delivery_id] FOREIGN KEY([delivery_id])
REFERENCES [load].[delivery] ([id])
GO

ALTER TABLE [IO].[load].[concessionaire] CHECK CONSTRAINT [fk_concessionaire_delivery_id]
GO

/*
DELETE [IO].[config].[master_process_type] WHERE id = 1
DELETE [IO].[config].[detail_process_type] WHERE master_process_type_id = 1
DROP TABLE [IO].[load].[concessionaire]
*/
-- COMMIT
-- ROLLBACK

-- master_process_type y detail_process_type
SELECT * FROM IO.config.master_process_type WHERE id = 1
SELECT * FROM IO.config.detail_process_type WHERE master_process_type_id = 1

/*
-- ====================================== LIMPIA PROCESO ============================================================
-- ================================================================================================================
TRUNCATE TABLE [IO].[input].[BMW_NewDealers]
TRUNCATE TABLE [IO].[hist].[BMW_NewDealers]
TRUNCATE TABLE [IO].[load].[concessionaire]

DELETE FROM IO.load.delivery DBCC CHECKIDENT ('IO.load.delivery',RESEED, 0)
TRUNCATE TABLE IO.load.operation
TRUNCATE TABLE IO.monitor.log
TRUNCATE TABLE IO.monitor.detail_process_log
DELETE FROM IO.monitor.master_process_log DBCC CHECKIDENT ('IO.monitor.master_process_log',RESEED, 0)

DELETE FROM [VehicleSales].[dbo].[concessionaire] WHERE id > 5
UPDATE [VehicleSales].[dbo].[concessionaire] SET [address] = NULL, city = NULL, postal_code = NULL, info = NULL, io_id = NULL WHERE id = 3
UPDATE [VehicleSales].[dbo].[concessionaire] SET [address] = 'c/ Entença, 324-326', postal_code = NULL, info = NULL, io_id = NULL WHERE id = 5
DBCC CHECKIDENT ('[VehicleSales].[dbo].[concessionaire]',RESEED, 5)
*/

-- ================================================================================================================
-- ================================================================================================================

SELECT TOP 100 * FROM IO.monitor.log ORDER BY id DESC
SELECT TOP 100 * FROM IO.monitor.master_process_log WHERE master_process_type_id = 1 ORDER BY id DESC
SELECT TOP 100 * FROM IO.monitor.detail_process_log ORDER BY id DESC
SELECT TOP 100 * FROM IO.load.delivery ORDER BY id DESC
SELECT TOP 100 * FROM IO.load.operation

SELECT * FROM [IO].[hist].[BMW_NewDealers]
SELECT * FROM [IO].[input].[BMW_NewDealers]
```

```sql
SELECT * FROM [IO].[load].[concessionaire]
SELECT * FROM [VehicleSales].[dbo].[concessionaire]

-- ======================== MAIN ========================
-- Tarea "start ingestion log" de los procesos para generar la traza en monitor.master_process_log
-- EXEC [monitor].[create_or_update_master_process_log] @p_master_process_id = 1, @p_in_process_log_id = 1, @p_step = 'Beginning'
SELECT * FROM [IO].monitor.master_process_log

-- {EXEC [IO].monitor.get_execution_summary @master_process_type_id = 1, @master_process_log_id = 4}

-- ======================== INGESTA ========================
-- Parámetro User::sql_get_main_config
EXEC config.sp_open_io_password_encryption_keys
SELECT d.[source_type] AS source_type,
   COALESCE(d.[source_file_folder_path], '') AS source_file_folder_path,
   COALESCE(d.[source_file_pattern], '') AS source_file_pattern,
   COALESCE(d.[skip_initial_rows], 0) AS skip_initial_rows,
   COALESCE(d.[source_table_query], '') AS source_table_query,
   COALESCE(d.[destination_table], '') AS destination_table,
   COALESCE(d. [source_file_template_path], '') AS  [source_file_template_path],
   m.[name] AS process_name,
   COALESCE(d.[zip_folder_path], '') AS zip_folder_path,
   COALESCE(d.[zip_file_path], '') AS zip_file_path,
   COALESCE(d.[ftp_server], '') AS ftp_server,
   COALESCE(d.[ftp_port], '') AS ftp_port,
   COALESCE(d.[ftp_user], '') AS ftp_user,
   COALESCE([IO].config.f_password_decryption(d.[ftp_pass]), '') AS ftp_pass,
   COALESCE(d.[ftp_operation], '') AS ftp_operation,
   COALESCE(d.[ftp_folder], '') AS ftp_folder,
   COALESCE(d.[ftp_file_pattern], '') AS ftp_file_pattern,
   COALESCE(d.[ftp_local_folder], '') AS ftp_local_folder,
   COALESCE(d.[ftp_local_file_pattern], '') AS ftp_local_file_pattern,
   COALESCE(d.[skip_final_rows], 0) AS skip_final_rows,
   COALESCE(d.[add_eof], 0) AS add_eof,
   COALESCE(d.[ftp_private_key_path], '') AS ftp_private_key_path,
   COALESCE(d.[ftp_ssh_host_key_fingerprint], '') AS ftp_ssh_host_key_fingerprint,
   COALESCE([IO].config.f_password_decryption(d.[ftp_private_key_passphrase]), '') AS ftp_private_key_passphrase,
   COALESCE(d.[source_connection_server], '') AS source_connection_server,
   COALESCE(d.[source_connection_database], '') AS source_connection_database,
   COALESCE(d.[source_connection_user], '') AS source_connection_user,
   COALESCE([IO].config.f_password_decryption(d.[source_connection_password]), '') AS source_connection_password,
   m.customer AS customer,
   COALESCE(d.check_ingestion_filename, 0) AS check_ingestion_filename,
   COALESCE(m.stop_on_warning, 0) AS stop_on_warning,
   COALESCE(d.ingestion_encoding_file, '') AS ingestion_encoding_file,
   COALESCE([IO].config.f_password_decryption(d.zip_rar_pass), '') AS zip_rar_pass
FROM IO.config.master_process_type m
INNER JOIN IO.config.detail_process_type d ON m.id = d.master_process_type_id
WHERE m.id = 1 AND d.phase = 'INGESTION' ORDER BY d.step ASC

-- Tarea "start detail log" de los procesos para generar la traza en monitor.detail_process_log
-- EXEC [monitor].[create_or_update_detail_process_log] @p_master_log_process_id = 1, @p_step = ?, @p_phase = ?, @p_entity = ?,
      @p_out_detail_process_log_id = ? OUTPUT

-- Tarea "sp_load_file_to_table"
-- EXEC [input].[sp_load_file_to_table] @p_ingest = ?, @p_path = ?, @p_template = ?, @p_skip_initial_rows = ?, @p_skip_final_rows = ?,
      @p_destination_table = ?, @p_order = ?

--Ejemplo de ejecución con formato ",":
EXEC [input].[sp_load_file_to_table]
   @p_ingest = 1
   ,@p_path = '\\WIN-SER-AMAROTO\IO_Files\Input\BMW_Group\Concesionarios\BMW_NewDealers_20240313_1415.csv'
   ,@p_template = '\\WIN-SER-AMAROTO\IO_Files\Template\BMW_Group\Concesionarios.fmt'
   ,@p_skip_initial_rows = 1
   ,@p_skip_final_rows = 0
   ,@p_destination_table = '[IO].[input].[BMW_NewDealers]'
   ,@p_order  = 1

--Ejemplo de ejecución con formato alternativo ';':
-- IMPORTANTE: Depositar antes fichero en la ruta debido a que el registro de la tabla de configuración contiene el formato original
EXEC [input].[sp_load_file_to_table]
   @p_ingest = 1
   ,@p_path = '\\WIN-SER-AMAROTO\IO_Files\Input\BMW_Group\Concesionarios\BMW_NewDealers_20240313_1415_ALT.csv'
   ,@p_template = '\\WIN-SER-AMAROTO\IO_Files\Template\BMW_Group\Concesionarios_ALT.fmt'
   ,@p_skip_initial_rows = 1
   ,@p_skip_final_rows = 0
   ,@p_destination_table = '[IO].[input].[BMW_NewDealers]'
   ,@p_order  = 1

SELECT * FROM [IO].[input].[BMW_NewDealers]
SELECT * FROM [IO].[hist].[BMW_NewDealers] ORDER BY ingestion_date DESC

-- ======================== TRANSFORMACION ========================
-- Tarea "Get delivery config" devuelve:
SELECT dpt.source_table_query AS delivery_config
FROM IO.config.master_process_type mpt
   INNER JOIN IO.config.detail_process_type dpt ON dpt.master_process_type_id = mpt.id
WHERE mpt.id = 1 AND dpt.phase = 'DELIVERY'

-- Tarea "Get transformation config" devuelve:
SELECT dpt.source_table_query AS query, dpt.entity AS entity
FROM IO.config.master_process_type mpt
   INNER JOIN IO.config.detail_process_type dpt ON dpt.master_process_type_id = mpt.id
WHERE mpt.id = 1 AND dpt.phase = 'TRANSFORMATION'
ORDER BY dpt.step ASC

-- Delivery
-- EXEC [IO].[input].[sp_BMW_NewDealers_new_delivery] @master_id = 1, @master_log_id = 1
SELECT * FROM IO.load.delivery ORDER BY id DESC
```

```sql

-- INSERT concessionaire
-- EXEC [IO].[input].[sp_BMW_NewDealers_transformation_insert_concessionaire] @master_process_id = 1
SELECT * FROM [IO].[load].[concessionaire]

-- UPDATE concessionaire
-- EXEC [IO].[input].[sp_BMW_NewDealers_transformation_update_concessionaire] @master_process_id = 1
SELECT * FROM [IO].[load].[concessionaire]

SELECT * FROM IO.load.operation

-- ======================== PERSISTENCIA ========================

-- Valor variable "sql_get_pending_deliveries"
SELECT dv.id as delivery_id, COALESCE(dv.customer_id, '0') as customer_id
FROM [IO].load.delivery dv
   INNER JOIN [IO].monitor.master_process_log mpl ON mpl.id = dv.master_process_log_id
   INNER JOIN [IO].config.master_process_type mpt ON mpt.id = mpl.master_process_type_id
WHERE dv.registration_date is null
   AND mpt.id = 1 ORDER BY dv.id

-- Tarea "Get persistence config" devuelve:
EXEC [IO].config.sp_open_io_password_encryption_keys
SELECT dpt.source_table_query AS query, dpt.entity AS entity, COALESCE(dpt.destination_connection_server, '') AS destination_connection_server,
      COALESCE(dpt.destination_connection_database, '') AS destination_connection_database, COALESCE(dpt.destination_connection_user, '') AS
      destination_connection_user, COALESCE([IO].config.f_password_decryption(dpt.destination_connection_password), '') AS
      destination_connection_password
FROM IO.config.master_process_type mpt
   INNER JOIN IO.config.detail_process_type dpt ON dpt.master_process_type_id = mpt.id
WHERE mpt.id = 1 AND dpt.phase = 'PERSISTENCE' ORDER BY dpt.step ASC

-- INSERT concessionaire
-- EXEC [IO].[input].[sp_BMW_NewDealers_persistence_insert_concessionaire] @delivery_id = 1, @affected_rows = 0
SELECT * FROM [VehicleSales].[dbo].[concessionaire]

-- UPDATE concessionaire
-- EXEC [IO].[input].[sp_BMW_NewDealers_persistence_update_concessionaire] @delivery_id = 1, @affected_rows = 0
SELECT * FROM [VehicleSales].[dbo].[concessionaire]

SELECT * FROM IO.load.operation

--UPDATE [IO].load.delivery SET registration_date = GETDATE() WHERE id = 1
```

A continuación se muestra el *Código C.36* con la creación SQL para el caso de uso 2 Output TopSellers.

Código C.36: 2 Output TopSellers xlsx.sql

```sql
/*
IMPORTANTE: Antes de empezar es necesario generar tanto el fichero a tratar como la plantilla
-- ==================================
-- ========== Tablas base ==========
-- ==================================
*/
USE [IO]
GO

BEGIN TRAN

-- [IO].[config].[master_process_type]
INSERT INTO [IO].[config].[master_process_type]
   ([id],[customer],[name],[description],[active_flag],[ingestion_flag],[transformation_flag],[persist_flag],[output_flag])
VALUES
    (2, 'BMW_Group', 'TopSellers_Output', 'Genera un informe con el TOP 10 de vendedores del grupo', 1, 0, 0, 0, 1)

-- [IO].[config].[detail_process_type]
-- Apertura de las claves de cifrado, siempre se ha de hacer antes de encriptar o desencriptar
EXEC [IO].config.sp_open_io_password_encryption_keys

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[source_table_query],[destination_format],[generate_empty_file],[destination_file_skip_rows],[destination_path]
   ,[destination_file_template_path])
VALUES
   (200, 2, 'OUTPUT', 1, 'TopSellers_Generation', 'TopSellers Exec output procedure', 'Generación del archivo de salida a través de procedimiento
      almacenado', 'BBDD'
   , 'EXEC [IO].[output].[sp_BMWGroup_TopSellers_report]', 'XLSX', 1, 1,
      '\\WIN-SER-AMAROTO\IO_Files\Output\BMW_Group\Vendedores\BMWGroup_TopSellers_yyyymmdd'
   ,'\\WIN-SER-AMAROTO\IO_Files\Template\BMW_Group\Output_TopSellers.fmt')

/*
-- Da problemas con el nombre -> Dejamos solo para cambio de extensiones
INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[file_action],[file_action_source_pattern],[file_action_destination_folder])
VALUES
   (210, 2, 'OUTPUT', 2, 'TopSellers_Rename', 'TopSellers check report name', 'Modifica el nombre del archivo para que incluya BMW Group', 'FILE'
   , 'RENAME', '\\WIN-SER-AMAROTO\IO_Files\Output\BMW_Group\Vendedores\BMW_TopSellers*.*', 'BMWGroup_TopSellers_yyyymmdd.xlsx')
*/

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[zip_folder_path],[zip_file_path],[destination_table]
   ,[zip_rar_pass])
```

```
VALUES
   (210, 2, 'OUTPUT', 2, 'TopSellers_ZIP', 'TopSellers compress report', 'Compresión del archivo con la información .ZIP con contraseña', 'ZIP'
   , '\\WIN-SER-AMAROTO\IO_Files\Output\BMW_Group\Vendedores\*.xlsx',
      '\\WIN-SER-AMAROTO\IO_Files\Output\BMW_Group\Vendedores\BMWGroup_TopSellers_yyyymmdd.zip', 'ZIP_File'
   , (SELECT [IO].config.f_password_encryption('TopSellers')))

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[destination_table],[ftp_server],[ftp_port],[ftp_user],[ftp_pass],[ftp_operation],[ftp_folder]
   ,[ftp_local_folder],[ftp_local_file_pattern])
VALUES
   (220, 2, 'OUTPUT', 3, 'TopSellers_FTP_Upload', 'TopSellers upload report into FTP', 'Subida del report TopSellers al FTP', 'FTP'
   , 'FTP_Upload', 'data.es', 21, 'ftpuser', (SELECT [IO].config.f_password_encryption('Usuario2011')), 'UPLOAD', '/BMW_Group/Informes/Salida/'
   , '\\WIN-SER-AMAROTO\IO_Files\Output\BMW_Group\Vendedores', '*.zip')

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[email_from],[email_to],[email_subject]
   ,[email_body]
   ,[email_has_attachment],[email_attachment_path],[email_attachment_pattern])
VALUES
   (230, 2, 'OUTPUT', 4, 'TopSellers_Email', 'TopSellers business email', 'Envío de email informando a Negocio, incluyendo el report como adjunto',
      'EMAIL'
   , 'server@data.es', 'amaroto@data.es', 'BMW Group - TopSeller report'
   , 'Buenos dias, adjuntamos el fichero mensual de TopSellers. <br> <br> <br> Un cordial saludo. <br> <br> <b> Departamento de Data </b> <br>
      <br><br> <br>'
   , 1, '\\WIN-SER-AMAROTO\IO_Files\Output\BMW_Group\Vendedores', 'BMWGroup_TopSellers*.zip')

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[file_action],[file_action_source_pattern])
VALUES
   (240, 2, 'OUTPUT', 5, 'TopSellers_Delete', 'TopSellers delete source file', 'Eliminar el report de la ruta original para evitar subirlo tras cada
      ejecución', 'FILE'
   , 'DELETE', '\\WIN-SER-AMAROTO\IO_Files\Output\BMW_Group\Vendedores\BMWGroup_TopSellers*.*')

-------------------------------------------------------------------------

/*
DELETE [IO].[config].[master_process_type] WHERE id = 2
DELETE [IO].[config].[detail_process_type] WHERE master_process_type_id = 2
*/
-- COMMIT
-- ROLLBACK

-- master_process_type y detail_process_type
SELECT id, customer, name, description, active_flag, ingestion_flag, transformation_flag, persist_flag, output_flag, stop_on_warning FROM
      IO.config.master_process_type WHERE id = 2
SELECT * FROM IO.config.master_process_type WHERE id = 2
SELECT * FROM IO.config.detail_process_type WHERE master_process_type_id = 2
SELECT * FROM IO.config.detail_process_type WHERE source_type = 'EMAIL'

/*
-- ====================================== LIMPIA PROCESO ================================================================
-- ======================================================================================================================
DELETE FROM IO.load.delivery DBCC CHECKIDENT ('IO.load.delivery',RESEED, 0)
TRUNCATE TABLE IO.monitor.log
TRUNCATE TABLE IO.monitor.detail_process_log
DELETE FROM IO.monitor.master_process_log DBCC CHECKIDENT ('IO.monitor.master_process_log',RESEED, 0)

DROP TABLE IF EXISTS [output].BMWGroup_TopSellers
*/

-- ======================================================================================================================
-- ======================================================================================================================

SELECT TOP 100 * FROM IO.monitor.log ORDER BY id DESC
SELECT TOP 100 * FROM IO.monitor.master_process_log WHERE master_process_type_id = 2 ORDER BY id DESC
SELECT TOP 100 * FROM IO.monitor.detail_process_log ORDER BY id DESC

-- ======================== MAIN ========================
-- Tarea "start ingestion log" de los procesos para generar la traza en monitor.master_process_log
-- EXEC [monitor].[create_or_update_master_process_log] @p_master_process_id = 1, @p_in_process_log_id = 1, @p_step = 'Beginning'
SELECT * FROM [IO].monitor.master_process_log

-- {EXEC [IO].monitor.get_execution_summary @master_process_type_id = 1, @master_process_log_id = 4}

-- ======================== OUTPUT ========================

-- Tarea "Get output config" devuelve:
EXEC [IO].config.sp_open_io_password_encryption_keys
SELECT COALESCE(dpt.source_table_query, '') AS query,
   COALESCE(dpt.entity, '') AS entity,
   COALESCE(dpt.destination_path, '') AS destination_path,
   COALESCE(dpt.destination_file_template_path, '') AS destination_file_template_path,
   COALESCE(dpt.destination_format, '') AS destination_format,
   COALESCE(dpt.destination_file_skip_rows, '') AS destination_file_skip_rows,
   COALESCE(dpt.generate_empty_file, CAST(1 AS BIT)) AS generate_empty_file,
   dpt.source_type AS source_type,
   COALESCE(dpt.[zip_folder_path], '') AS zip_folder_path,
   COALESCE(dpt.[zip_file_path], '') AS zip_file_path,
   COALESCE(dpt.[ftp_server], '') AS ftp_server,
   COALESCE(dpt.[ftp_port], '') AS ftp_port,
   COALESCE(dpt.[ftp_user], '') AS ftp_user,
   COALESCE([IO].config.f_password_decryption(dpt.[ftp_pass]), '') AS ftp_pass,
   COALESCE(dpt.[ftp_operation], '') AS ftp_operation,
   COALESCE(dpt.[ftp_folder], '') AS ftp_folder,
   COALESCE(dpt.[ftp_file_pattern], '') AS ftp_file_pattern,
   COALESCE(dpt.[ftp_local_folder], '') AS ftp_local_folder,
```

```
    COALESCE(dpt.[ftp_local_file_pattern], '') AS ftp_local_file_pattern,
    COALESCE([email_from], '') AS email_from,
    COALESCE([email_to], '') AS email_to,
    COALESCE([email_subject], '') AS email_subject,
    COALESCE([email_body], '') AS email_body,
    COALESCE([email_has_attachment], 0) AS email_has_attachment,
    COALESCE([email_attachment_path], '') AS email_attachment_path,
    COALESCE([email_attachment_pattern], '') AS email_attachment_pattern,
    COALESCE([ftp_private_key_path], '') AS ftp_private_key_path,
    COALESCE([ftp_ssh_host_key_fingerprint], '') AS ftp_ssh_host_key_fingerprint,
    COALESCE([IO].config.f_password_decryption([ftp_private_key_passphrase]), '') AS ftp_private_key_passphrase,
    COALESCE(dpt.[file_action], '') AS file_action,
    COALESCE(dpt.[file_action_source_pattern], '') AS file_action_source_pattern,
    COALESCE(dpt.[file_action_destination_folder], '') AS file_action_destination_folder,
    mpt.customer AS customer,
    mpt.name AS process_name,
    COALESCE([IO].config.f_password_decryption(dpt.zip_rar_pass), '') AS zip_rar_pass
FROM IO.config.master_process_type mpt INNER JOIN IO.config.detail_process_type dpt ON dpt.master_process_type_id = mpt.id
WHERE mpt.id = 2 AND dpt.phase = 'OUTPUT' ORDER BY dpt.step ASC

-- Tarea "Get affected rows" ejecuta:
-- argumentos entrada: query: EXEC [IO].[output].[sp_BMWGroup_TopSellers_report], destination_file_skip_rows: 1
-- argumentos salida: affected_rows: 10

EXEC [IO].[output].[sp_BMWGroup_TopSellers_report]
```

A continuación se muestra el *Código C.37* con la creación SQL para el caso de uso 3 Output MiniSales.

Código C.37: 3 Output MiniSales txt.sql

```
/*
IMPORTANTE: Antes de empezar es necesario generar tanto el fichero a tratar como la plantilla
-- ================================
-- ========== Tablas base ==========
-- ================================
*/
USE [IO]
GO

BEGIN TRAN

-- [IO].[config].[master_process_type]
INSERT INTO [IO].[config].[master_process_type]
   ([id],[customer],[name],[description],[active_flag],[ingestion_flag],[transformation_flag],[persist_flag],[output_flag])
VALUES
    (3, 'Mini', 'MiniSales_Output', 'Genera un informe con los ventas de vehículos de Mini', 1, 0, 0, 0, 1)

-- [IO].[config].[detail_process_type]
-- Apertura de las claves de cifrado, siempre se ha de hacer antes de encriptar o desencriptar
EXEC [IO].config.sp_open_io_password_encryption_keys

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[source_table_query],[destination_format],[generate_empty_file],[destination_file_skip_rows],[destination_path]
   ,[destination_file_template_path])
VALUES
   (300, 3, 'OUTPUT', 1, 'MiniSales_Generation', 'MiniSales Exec output procedure', 'Generación del archivo de salida a través de consulta embebida
      almacenado', 'BBDD'
   --
,'
SET NOCOUNT ON;
DROP TABLE IF EXISTS #auxiliar;

SELECT bt.[name] AS marca, mt.[name] AS modelo, pt.[name] AS tipo_propuesta, ft.[name] AS tipo_modelo, o.offer_amount AS importe_venta
   ,s.sale_date AS fecha_venta, o.offer_date AS fecha_oferta, c.[name] AS cliente, p.proposal_amount AS importe_propuesta
   ,p.proposal_date AS fecha_propuesta, co.[name] AS concesionario, se.[name] AS vendedor, GETDATE() AS creation_date
INTO #auxiliar
FROM [VehicleSales].dbo.sale s
   INNER JOIN [VehicleSales].dbo.offer o ON s.offer_id = o.id
   INNER JOIN [VehicleSales].dbo.proposal p ON p.id = o.proposal_id
   INNER JOIN [VehicleSales].dbo.customer c ON c.id = p.customer_id
   INNER JOIN [VehicleSales].dbo.seller se ON se.id = p.seller_id
   INNER JOIN [VehicleSales].dbo.concessionaire co ON co.id = se.concessionaire_id
   INNER JOIN [VehicleSales].dbo.proposal_type pt ON pt.id = p.proposal_type_id
   INNER JOIN [VehicleSales].dbo.model_type mt ON mt.id = p.model_type_id
   INNER JOIN [VehicleSales].dbo.brand_type bt ON bt.id = mt.brand_type_id
   INNER JOIN [VehicleSales].dbo.fuel_type ft ON ft.id = mt.fuel_type_id
WHERE bt.name = ''Mini'' AND o.active = 1;

SELECT ''Marca'', ''Modelo'', ''Tipo_propuesta'', ''Tipo_modelo'', ''Importe_venta''
   , ''Fecha_venta'', ''Fecha_oferta'', ''Cliente''
   , ''Importe_propuesta'', ''Fecha_propuesta'', ''Concesionario'', ''Vendedor''
UNION ALL
SELECT [marca], [modelo], [tipo_propuesta], [tipo_modelo], CONVERT(NVARCHAR(16), [importe_venta])
   , CONVERT(NVARCHAR(16), [fecha_venta]), CONVERT(NVARCHAR(16), [fecha_oferta]), [cliente]
   , CONVERT(NVARCHAR(16), [importe_propuesta]), CONVERT(NVARCHAR(16), [fecha_propuesta]), [concesionario], [vendedor]
FROM #auxiliar;
'
   --
   , 'TXT', 1, 1, '\\WIN-SER-AMAROTO\IO_Files\Output\BMW_Group\Vehiculos\Mini_MiniSales_yyyymmdd'
   ,'\\WIN-SER-AMAROTO\IO_Files\Template\BMW_Group\Output_MiniSales.fmt')

INSERT INTO [IO].[config].[detail_process_type]
```

```
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[file_action],[file_action_source_pattern],[file_action_destination_folder])
VALUES
   (310, 3, 'OUTPUT', 2, 'MiniSales_Rename', 'MiniSales check report name', 'Modifica la extensión del archivo de TXT a CSV', 'FILE'
   , 'RENAME', '\\WIN-SER-AMAROTO\IO_Files\Output\BMW_Group\Vehiculos\*.txt', '*.csv')

-- Para este ejemplo comprimimos sin contraseña
INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[zip_folder_path],[zip_file_path],[destination_table])
VALUES
   (320, 3, 'OUTPUT', 3, 'MiniSales_ZIP', 'MiniSales compress report', 'Compresión del archivo con la información .ZIP', 'ZIP'
   , '\\WIN-SER-AMAROTO\IO_Files\Output\BMW_Group\Vehiculos\*.csv',
      '\\WIN-SER-AMAROTO\IO_Files\Output\BMW_Group\Vehiculos\Mini_MiniSales_yyyymmdd.zip', 'ZIP_File')

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[destination_table],[ftp_server],[ftp_port],[ftp_user],[ftp_pass],[ftp_operation],[ftp_folder]
   ,[ftp_local_folder],[ftp_local_file_pattern])
VALUES
   (330, 3, 'OUTPUT', 4, 'MiniSales_FTP_Upload', 'MiniSales upload report into FTP', 'Subida del report MiniSales al FTP', 'FTP'
   , 'FTP_Upload', 'data.es', 21, 'ftpuser', (SELECT [IO].config.f_password_encryption('Usuario2011')), 'UPLOAD', '/BMW_Group/Informes/Salida/'
   , '\\WIN-SER-AMAROTO\IO_Files\Output\BMW_Group\Vehiculos', '*.zip')

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[file_action],[file_action_source_pattern])
VALUES
   (340, 3, 'OUTPUT', 5, 'MiniSales_Delete', 'MiniSales delete source file', 'Eliminar el report de la ruta original para evitar subirlo tras cada
      ejecución', 'FILE'
   , 'DELETE', '\\WIN-SER-AMAROTO\IO_Files\Output\BMW_Group\Vehiculos\Mini_MiniSales*.*')

------------------------------------------------------------------------

/*
DELETE [IO].[config].[master_process_type] WHERE id = 3
DELETE [IO].[config].[detail_process_type] WHERE master_process_type_id = 3
*/
-- COMMIT
-- ROLLBACK

-- master_process_type y detail_process_type
SELECT id, customer, name, description, active_flag, ingestion_flag, transformation_flag, persist_flag, output_flag, stop_on_warning FROM
      IO.config.master_process_type WHERE id = 2
SELECT * FROM IO.config.master_process_type WHERE id = 3
SELECT * FROM IO.config.detail_process_type WHERE master_process_type_id = 3
SELECT * FROM IO.config.detail_process_type WHERE source_type = 'BBDD'

/*
-- ======================================== LIMPIA PROCESO ============================================================
-- ====================================================================================================================
TRUNCATE TABLE IO.monitor.log
TRUNCATE TABLE IO.monitor.detail_process_log
DELETE FROM IO.monitor.master_process_log DBCC CHECKIDENT ('IO.monitor.master_process_log',RESEED, 0)
*/

-- ====================================================================================================================
-- ====================================================================================================================

SELECT TOP 100 * FROM IO.monitor.log ORDER BY id DESC
SELECT TOP 100 * FROM IO.monitor.master_process_log WHERE master_process_type_id = 3 ORDER BY id DESC
SELECT TOP 100 * FROM IO.monitor.detail_process_log ORDER BY id DESC

-- ======================== MAIN ========================
-- Tarea "start ingestion log" de los procesos para generar la traza en monitor.master_process_log
-- EXEC [monitor].[create_or_update_master_process_log] @p_master_process_id = 1, @p_in_process_log_id = 1, @p_step = 'Beginning'
SELECT * FROM [IO].monitor.master_process_log

-- {EXEC [IO].monitor.get_execution_summary @master_process_type_id = 1, @master_process_log_id = 4}

-- ======================== OUTPUT ========================

-- Tarea "Get output config" devuelve:
EXEC [IO].config.sp_open_io_password_encryption_keys
SELECT COALESCE(dpt.source_table_query, '') AS query,
   COALESCE(dpt.entity, '') AS entity,
   COALESCE(dpt.destination_path, '') AS destination_path,
   COALESCE(dpt.destination_file_template_path, '') AS destination_file_template_path,
   COALESCE(dpt.destination_format, '') AS destination_format,
   COALESCE(dpt.destination_file_skip_rows, '') AS destination_file_skip_rows,
   COALESCE(dpt.generate_empty_file, CAST(1 AS BIT)) AS generate_empty_file,
   dpt.source_type AS source_type,
   COALESCE(dpt.[zip_folder_path], '') AS zip_folder_path,
   COALESCE(dpt.[zip_file_path], '') AS zip_file_path,
   COALESCE(dpt.[ftp_server], '') AS ftp_server,
   COALESCE(dpt.[ftp_port], '') AS ftp_port,
   COALESCE(dpt.[ftp_user], '') AS ftp_user,
   COALESCE([IO].config.f_password_decryption(dpt.[ftp_pass]), '') AS ftp_pass,
   COALESCE(dpt.[ftp_operation], '') AS ftp_operation,
   COALESCE(dpt.[ftp_folder], '') AS ftp_folder,
   COALESCE(dpt.[ftp_file_pattern], '') AS ftp_file_pattern,
   COALESCE(dpt.[ftp_local_folder], '') AS ftp_local_folder,
   COALESCE(dpt.[ftp_local_file_pattern], '') AS ftp_local_file_pattern,
   COALESCE([email_from], '') AS email_from,
   COALESCE([email_to], '') AS email_to,
   COALESCE([email_subject], '') AS email_subject,
   COALESCE([email_body], '') AS email_body,
   COALESCE([email_has_attachment], 0) AS email_has_attachment,
   COALESCE([email_attachment_path], '') AS email_attachment_path,
```

```
   COALESCE([email_attachment_pattern], '') AS email_attachment_pattern,
   COALESCE([ftp_private_key_path], '') AS ftp_private_key_path,
   COALESCE([ftp_ssh_host_key_fingerprint], '') AS ftp_ssh_host_key_fingerprint,
   COALESCE([IO].config.f_password_decryption([ftp_private_key_passphrase]), '') AS ftp_private_key_passphrase,
   COALESCE(dpt.[file_action], '') AS file_action,
   COALESCE(dpt.[file_action_source_pattern], '') AS file_action_source_pattern,
   COALESCE(dpt.[file_action_destination_folder], '') AS file_action_destination_folder,
   mpt.customer AS customer,
   mpt.name AS process_name,
   COALESCE([IO].config.f_password_decryption(dpt.zip_rar_pass), '') AS zip_rar_pass
FROM IO.config.master_process_type mpt INNER JOIN IO.config.detail_process_type dpt ON dpt.master_process_type_id = mpt.id
WHERE mpt.id = 3 AND dpt.phase = 'OUTPUT' ORDER BY dpt.step ASC

-- Tarea "Get affected rows" ejecuta:
-- argumentos entrada: query: 'QUERY (ver abajo)', destination_file_skip_rows: 1
-- argumentos salida: affected_rows: 5

-- Query
SET NOCOUNT ON;
DROP TABLE IF EXISTS #auxiliar;

SELECT bt.[name] AS marca, mt.[name] AS modelo, pt.[name] AS tipo_propuesta, ft.[name] AS tipo_modelo, o.offer_amount AS importe_venta
   ,s.sale_date AS fecha_venta, o.offer_date AS fecha_oferta, c.[name] AS cliente, p.proposal_amount AS importe_propuesta
   ,p.proposal_date AS fecha_propuesta, co.[name] AS concesionario, se.[name] AS vendedor, GETDATE() AS creation_date
INTO #auxiliar
FROM [VehicleSales].dbo.sale s
   INNER JOIN [VehicleSales].dbo.offer o ON s.offer_id = o.id
   INNER JOIN [VehicleSales].dbo.proposal p ON p.id = o.proposal_id
   INNER JOIN [VehicleSales].dbo.customer c ON c.id = p.customer_id
   INNER JOIN [VehicleSales].dbo.seller se ON se.id = p.seller_id
   INNER JOIN [VehicleSales].dbo.concessionaire co ON co.id = se.concessionaire_id
   INNER JOIN [VehicleSales].dbo.proposal_type pt ON pt.id = p.proposal_type_id
   INNER JOIN [VehicleSales].dbo.model_type mt ON mt.id = p.model_type_id
   INNER JOIN [VehicleSales].dbo.brand_type bt ON bt.id = mt.brand_type_id
   INNER JOIN [VehicleSales].dbo.fuel_type ft ON ft.id = mt.fuel_type_id
WHERE bt.name = 'Mini' AND o.active = 1

SELECT 'Marca', 'Modelo', 'Tipo_propuesta', 'Tipo_modelo', 'Importe_venta'
   , 'Fecha_venta', 'Fecha_oferta', 'Cliente'
   , 'Importe_propuesta', 'Fecha_propuesta', 'Concesionario', 'Vendedor'
UNION ALL
SELECT [marca], [modelo], [tipo_propuesta], [tipo_modelo], CONVERT(NVARCHAR(16), [importe_venta])
   , CONVERT(NVARCHAR(16), [fecha_venta]), CONVERT(NVARCHAR(16), [fecha_oferta]), [cliente]
   , CONVERT(NVARCHAR(16), [importe_propuesta]), CONVERT(NVARCHAR(16), [fecha_propuesta]), [concesionario], [vendedor]
FROM #auxiliar;
```

A continuación se muestra el *Código C.38* con la creación SQL para el caso de uso 4 Input RemoteConnections.

Código C.38: 4 Input RemoteConnections BBDD.sql

```
/*
IMPORTANTE: Antes de empezar es necesario generar tanto el fichero a tratar como la plantilla
Este proceso no requiere transformación, solo contiene ingesta (origen remoto) y persistencia (destino remoto)
-- ================================
-- ========== Tablas base ==========
-- ================================
*/
USE [IO]
GO

BEGIN TRAN

-- [IO].[config].[master_process_type]
INSERT INTO [IO].[config].[master_process_type]
   ([id],[customer],[name],[description],[active_flag],[ingestion_flag],[transformation_flag],[persist_flag],[output_flag])
VALUES
    (4, 'BMW_Group', 'RemoteConnections_Input', 'Test para comprobar la conexión remota de la IO tanto para el origen como para el destino', 1, 1, 1,
      1, 0)

-- [IO].[config].[detail_process_type]
-- Apertura de las claves de cifrado, siempre se ha de hacer antes de encriptar o desencriptar
EXEC [IO].config.sp_open_io_password_encryption_keys

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[source_table_query]
   ,[destination_table],[source_connection_server],[source_connection_database],[source_connection_user],[source_connection_password])
VALUES
   (400, 4, 'INGESTION', 1, 'load_remote_VehicleSales.dbo.sale', 'RemoteConnections load sales', 'Generación de tabla auxiliar con la información de
      ventas en BBDD origen VehicleSales', 'BBDD'
   , 'EXEC [IO].[input].[sp_RemoteConnections_ingestion_VehicleSales_sale] @destination_table = ?, @remote_server = ?, @remote_database = ?,
      @remote_user = ?, @remote_pass = ?'
   , '[IO].auxiliar.sale', 'WIN-SER-AMAROTO', 'VehicleSales', 'etlautomation', (SELECT [IO].config.f_password_encryption('etlautomation')))

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[source_table_query])
VALUES
   (410, 4, 'DELIVERY', 1, 'new_delivery', 'Alta delivery RemoteConnections', 'Alta de una nueva delivery de RemoteConnections_Input', 'BBDD'
   , 'EXEC [IO].[input].[sp_RemoteConnections_new_delivery] @master_id = ?, @master_log_id = ?')

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
```

```
   ,[source_table_query]
   ,[destination_connection_server],[destination_connection_database],[destination_connection_user],[destination_connection_password])
VALUES
   (420, 4, 'PERSISTENCE', 1, 'insert_remote_VehicleAlt.dbo.sale', 'Insert VehicleAlt sale', 'Incorporación de los datos en BBDD destino VehicleSales sale', 'BBDD'
   , 'EXEC [IO].[input].[sp_RemoteConnections_persistence_insert_VehicleAlt_sale] @delivery_id= ?, @remote_server = ?, @remote_database = ?, @remote_user = ?, @remote_pass = ?, @affected_rows = ? OUTPUT'
   , 'WIN-SER-AMAROTO', 'VehicleAlt', 'etlautomation', (SELECT [IO].config.f_password_encryption('etlautomation')))

-----------------------------------------------------------------------

/*
DELETE [IO].[config].[master_process_type] WHERE id = 4
DELETE [IO].[config].[detail_process_type] WHERE master_process_type_id = 4
*/
-- COMMIT
-- ROLLBACK

-- master_process_type y detail_process_type
SELECT * FROM IO.config.master_process_type WHERE id = 4
SELECT * FROM IO.config.detail_process_type WHERE master_process_type_id = 4

/*
-- ======================================= LIMPIA PROCESO ============================================================
-- ===================================================================================================================
DROP TABLE IF EXISTS [IO].auxiliar.sale
TRUNCATE TABLE VehicleAlt.dbo.[sale]

DELETE FROM IO.load.delivery DBCC CHECKIDENT ('IO.load.delivery',RESEED, 0)
TRUNCATE TABLE IO.load.operation
TRUNCATE TABLE IO.monitor.log
TRUNCATE TABLE IO.monitor.detail_process_log
DELETE FROM IO.monitor.master_process_log DBCC CHECKIDENT ('IO.monitor.master_process_log',RESEED, 0)
*/

-- ===================================================================================================================
-- ===================================================================================================================

SELECT TOP 100 * FROM IO.monitor.log ORDER BY id DESC
SELECT TOP 100 * FROM IO.monitor.master_process_log WHERE master_process_type_id = 4 ORDER BY id DESC
SELECT TOP 100 * FROM IO.monitor.detail_process_log ORDER BY id DESC
SELECT TOP 100 * FROM IO.load.delivery ORDER BY id DESC
SELECT TOP 100 * FROM IO.load.operation

SELECT * FROM VehicleSales.dbo.sale
SELECT * FROM [IO].auxiliar.sale
SELECT * FROM VehicleAlt.dbo.sale

-- ======================== MAIN ========================
-- Tarea "start ingestion log" de los procesos para generar la traza en monitor.master_process_log
-- EXEC [monitor].[create_or_update_master_process_log] @p_master_process_id = 4, @p_in_process_log_id = 1, @p_step = 'Beginning'
SELECT * FROM [IO].monitor.master_process_log

-- {EXEC [IO].monitor.get_execution_summary @master_process_type_id = 4, @master_process_log_id = 1}

-- ======================== INGESTA ========================
-- Parámetro User::sql_get_main_config
EXEC config.sp_open_io_password_encryption_keys
SELECT d.[source_type] AS source_type,
   COALESCE(d.[source_file_folder_path], '') AS source_file_folder_path,
   COALESCE(d.[source_file_pattern], '') AS source_file_pattern,
   COALESCE(d.[skip_initial_rows], 0) AS skip_initial_rows,
   COALESCE(d.[source_table_query], '') AS source_table_query,
   COALESCE(d.[destination_table], '') AS destination_table,
   COALESCE(d. [source_file_template_path], '') AS  [source_file_template_path],
   m.[name] AS process_name,
   COALESCE(d.[zip_folder_path], '') AS zip_folder_path,
   COALESCE(d.[zip_file_path], '') AS zip_file_path,
   COALESCE(d.[ftp_server], '') AS ftp_server,
   COALESCE(d.[ftp_port], '') AS ftp_port,
   COALESCE(d.[ftp_user], '') AS ftp_user,
   COALESCE([IO].config.f_password_decryption(d.[ftp_pass]), '') AS ftp_pass,
   COALESCE(d.[ftp_operation], '') AS ftp_operation,
   COALESCE(d.[ftp_folder], '') AS ftp_folder,
   COALESCE(d.[ftp_file_pattern], '') AS ftp_file_pattern,
   COALESCE(d.[ftp_local_folder], '') AS ftp_local_folder,
   COALESCE(d.[ftp_local_file_pattern], '') AS ftp_local_file_pattern,
   COALESCE(d.[skip_final_rows], 0) AS skip_final_rows,
   COALESCE(d.[add_eof], 0) AS add_eof,
   COALESCE(d.[ftp_private_key_path], '') AS ftp_private_key_path,
   COALESCE(d.[ftp_ssh_host_key_fingerprint], '') AS ftp_ssh_host_key_fingerprint,
   COALESCE([IO].config.f_password_decryption(d.[ftp_private_key_passphrase]), '') AS ftp_private_key_passphrase,
   COALESCE(d.[source_connection_server], '') AS source_connection_server,
   COALESCE(d.[source_connection_database], '') AS source_connection_database,
   COALESCE(d.[source_connection_user], '') AS source_connection_user,
   COALESCE([IO].config.f_password_decryption(d.[source_connection_password]), '') AS source_connection_password,
   m.customer AS customer,
   COALESCE(d.check_ingestion_filename, 0) AS check_ingestion_filename,
   COALESCE(m.stop_on_warning, 0) AS stop_on_warning,
   COALESCE(d.ingestion_encoding_file, '') AS ingestion_encoding_file,
   COALESCE([IO].config.f_password_decryption(d.zip_rar_pass), '') AS zip_rar_pass
FROM IO.config.master_process_type m
INNER JOIN IO.config.detail_process_type d ON m.id = d.master_process_type_id
WHERE m.id = 4 AND d.phase = 'INGESTION' ORDER BY d.step ASC

-- Tarea "start detail log" de los procesos para generar la traza en monitor.detail_process_log
-- EXEC [monitor].[create_or_update_detail_process_log] @p_master_log_process_id = 1, @p_step = ?, @p_phase = ?, @p_entity = ?, @p_out_detail_process_log_id = ? OUTPUT

```

```sql
-- Tarea "sp_load_file_to_table"
-- EXEC [input].[sp_load_file_to_table] @p_ingest = ?, @p_path = ?, @p_template = ?, @p_skip_initial_rows = ?, @p_skip_final_rows = ?,
      @p_destination_table = ?, @p_order = ?

-- Ejecución source_table_query (remoto):
EXEC [IO].[input].[sp_RemoteConnections_ingestion_VehicleSales_sale]
   @destination_table = '[IO].auxiliar.sale'
   , @remote_server = 'WIN-SER-AMAROTO'
   , @remote_database = 'VehicleSales'
   , @remote_user = 'etlautomation'
   , @remote_pass = 'etlautomation'

SELECT * FROM VehicleSales.dbo.sale
SELECT * FROM [IO].auxiliar.sale

-- ======================== TRANSFORMACION ========================
-- Tarea "Get delivery config" devuelve:
SELECT dpt.source_table_query AS delivery_config
FROM IO.config.master_process_type mpt
   INNER JOIN IO.config.detail_process_type dpt ON dpt.master_process_type_id = mpt.id
WHERE mpt.id = 4 AND dpt.phase = 'DELIVERY'

-- Tarea "Get transformation config" devuelve:
--SELECT dpt.source_table_query AS query, dpt.entity AS entity
--FROM IO.config.master_process_type mpt
-- INNER JOIN IO.config.detail_process_type dpt ON dpt.master_process_type_id = mpt.id
--WHERE mpt.id = 4 AND dpt.phase = 'TRANSFORMATION'
--ORDER BY dpt.step ASC

-- Delivery
-- EXEC [IO].[input].[sp_RemoteConnections_new_delivery] @master_id = 4, @master_log_id = 1
SELECT * FROM IO.load.delivery ORDER BY id DESC

SELECT * FROM IO.load.operation

-- ======================== PERSISTENCIA ========================

-- Valor variable "sql_get_pending_deliveries"
SELECT dv.id as delivery_id, COALESCE(dv.customer_id, '0') as customer_id
FROM [IO].load.delivery dv
   INNER JOIN [IO].monitor.master_process_log mpl ON mpl.id = dv.master_process_log_id
   INNER JOIN [IO].config.master_process_type mpt ON mpt.id = mpl.master_process_type_id
WHERE dv.registration_date is null
   AND mpt.id = 4 ORDER BY dv.id

-- Tarea "Get persistence config" devuelve:
EXEC [IO].config.sp_open_io_password_encryption_keys
SELECT dpt.source_table_query AS query, dpt.entity AS entity, COALESCE(dpt.destination_connection_server, '') AS destination_connection_server,
      COALESCE(dpt.destination_connection_database, '') AS destination_connection_database, COALESCE(dpt.destination_connection_user, '') AS
      destination_connection_user, COALESCE([IO].config.f_password_decryption(dpt.destination_connection_password), '') AS
      destination_connection_password
FROM IO.config.master_process_type mpt
   INNER JOIN IO.config.detail_process_type dpt ON dpt.master_process_type_id = mpt.id
WHERE mpt.id = 4 AND dpt.phase = 'PERSISTENCE' ORDER BY dpt.step ASC

-- INSERT sale
EXEC [IO].[input].[sp_RemoteConnections_persistence_insert_VehicleAlt_sale]
      @delivery_id = 1
      , @remote_server = 'WIN-SER-AMAROTO'
      , @remote_database = 'VehicleAlt'
      , @remote_user = 'etlautomation'
      , @remote_pass = 'etlautomation'
      , @affected_rows = 0

SELECT * FROM [VehicleAlt].dbo.sale

SELECT * FROM IO.load.operation

--UPDATE [IO].load.delivery SET registration_date = GETDATE() WHERE id = 1
```

A continuación se muestra el *Código C.39* con la de creación SQL para el caso de uso 5 Input StatusUpdate.

Código C.39: 5 Input StatusUpdate xlsx.sql

```sql
/*
IMPORTANTE: Antes de empezar es necesario generar tanto el fichero a tratar como la plantilla
-- ==================================
-- ========== Tablas base ==========
-- ==================================
*/
USE [IO]
GO

BEGIN TRAN

-- [IO].[config].[master_process_type]
INSERT INTO [IO].[config].[master_process_type]
   ([id],[customer],[name],[description],[active_flag],[ingestion_flag],[transformation_flag],[persist_flag],[output_flag])
VALUES
    (5, 'BMW_Group', 'StatusUpdate_Input', 'Incorpora la información relacionada con el proceso de compra de un vehículo', 1, 1, 1, 1, 0)

-- [IO].[config].[detail_process_type]
-- Apertura de las claves de cifrado, siempre se ha de hacer antes de encriptar o desencriptar
```

```
EXEC [IO].config.sp_open_io_password_encryption_keys

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type],[destination_table],[ftp_server],[ftp_port],[ftp_user]
   ,[ftp_pass],[ftp_operation],[ftp_folder],[ftp_file_pattern],[ftp_local_folder])
VALUES
   (500, 5, 'INGESTION', 1, 'download_StatusUpdate', 'Descarga archivo StatusUpdate', 'Recoge del FTP y mueve al Input el fichero StatusUpdate',
      'FTP', 'FTP_Download', 'data.es', 21, 'ftpuser'
   ,(SELECT [IO].config.f_password_encryption('Usuario2011')), 'DOWNLOAD', '/BMW_Group/InformacionCore/Entrada/', '*StatusUpdate*.zip',
      '\\WIN-SER-AMAROTO\IO_Files\Input\BMW_Group\InformacionCore')

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[zip_folder_path],[zip_file_path],[DESTINATION_TABLE]
   ,[ZIP_RAR_PASS])
VALUES
   (505, 5, 'INGESTION', 2, 'unzip_StatusUpdate', 'Descomprime zip StatusUpdate', 'Decompresión del archivo .ZIP con contraseña que contiene los
      ficheros a tratar', 'UNZIP'
   , '\\WIN-SER-AMAROTO\IO_Files\Input\BMW_Group\InformacionCore', '\\WIN-SER-AMAROTO\IO_Files\Input\BMW_Group\InformacionCore\*StatusUpdate*.zip',
      'UNZIP_File'
   , (SELECT [IO].config.f_password_encryption('StatusUpdate')))

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type],[source_file_folder_path]
   ,[source_file_pattern],[add_eof],[skip_final_rows],[skip_initial_rows],[source_file_template_path],[destination_table],[check_ingestion_filename])
VALUES
   (510, 5, 'INGESTION', 3, 'load_file_input_table', 'BMWGroup_StatusUpdate', 'Archivos de propuestas/ofertas/ventas', 'FILE',
      '\\WIN-SER-AMAROTO\IO_Files\Input\BMW_Group\InformacionCore'
   , '*StatusUpdate*.xlsx', 0, 0, 1, '\\WIN-SER-AMAROTO\IO_Files\Template\BMW_Group\Input_StatusUpdate.fmt', '[IO].[input].[BMWGroup_StatusUpdate]', 1)

--
INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[source_table_query])
VALUES
   (515, 5, 'DELIVERY', 1, 'new_delivery', 'Alta delivery StatusUpdate', 'Alta de una nueva delivery de StatusUpdate por cada marca', 'BBDD'
   , 'EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_new_delivery] @master_id = ?, @master_log_id = ?')

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[source_table_query])
VALUES
   (520, 5, 'TRANSFORMATION', 1, 'insert_model_type', 'Insert load model_type', 'Incorporación de los datos de input a load.model_type', 'BBDD'
   , 'EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_transformation_insert_model_type] @master_process_id = ?')

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[source_table_query])
VALUES
   (525, 5, 'TRANSFORMATION', 2, 'insert_seller', 'Insert load seller', 'Incorporación de los datos de input a load.seller', 'BBDD'
   , 'EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_transformation_insert_seller] @master_process_id = ?')

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[source_table_query])
VALUES
   (530, 5, 'TRANSFORMATION', 3, 'insert_customer', 'Insert load customer', 'Incorporación de los datos de input a load.customer', 'BBDD'
   , 'EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_transformation_insert_customer] @master_process_id = ?')

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[source_table_query])
VALUES
   (535, 5, 'TRANSFORMATION', 4, 'insert_proposal', 'Insert load proposal', 'Incorporación de los datos de input a load.proposal', 'BBDD'
   , 'EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_transformation_insert_proposal] @master_process_id = ?')

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[source_table_query])
VALUES
   (540, 5, 'TRANSFORMATION', 5, 'insert_offer', 'Insert load offer', 'Incorporación de los datos de input a load.offer', 'BBDD'
   , 'EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_transformation_insert_offer] @master_process_id = ?')

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[source_table_query])
VALUES
   (545, 5, 'TRANSFORMATION', 6, 'insert_sale', 'Insert load sale', 'Incorporación de los datos de input a load.sale', 'BBDD'
   , 'EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_transformation_insert_sale] @master_process_id = ?')

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[source_table_query])
VALUES
   (550, 5, 'PERSISTENCE', 1, 'insert_model_type', 'Insert VehicleSales model_type', 'Incorporación de los datos de load a VehicleSales.model_type',
      'BBDD'
   , 'EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_persistence_insert_model_type] @delivery_id = ?, @affected_rows = ? OUTPUT')

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[source_table_query])
VALUES
   (555, 5, 'PERSISTENCE', 2, 'insert_seller', 'Insert VehicleSales seller', 'Incorporación de los datos de load a VehicleSales.seller', 'BBDD'
   , 'EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_persistence_insert_seller] @delivery_id = ?, @affected_rows = ? OUTPUT')

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[source_table_query])
VALUES
```

```
   (560, 5, 'PERSISTENCE', 3, 'insert_customer', 'Insert VehicleSales customer', 'Incorporación de los datos de load a VehicleSales.customer', 'BBDD'
   , 'EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_persistence_insert_customer] @delivery_id = ?, @affected_rows = ? OUTPUT')

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[source_table_query])
VALUES
   (565, 5, 'PERSISTENCE', 4, 'insert_proposal', 'Insert VehicleSales proposal', 'Incorporación de los datos de load a VehicleSales.proposal', 'BBDD'
   , 'EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_persistence_insert_proposal] @delivery_id = ?, @affected_rows = ? OUTPUT')

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[source_table_query])
VALUES
   (570, 5, 'PERSISTENCE', 5, 'insert_offer', 'Insert VehicleSales offer', 'Incorporación de los datos de load a VehicleSales.offer', 'BBDD'
   , 'EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_persistence_insert_offer] @delivery_id = ?, @affected_rows = ? OUTPUT')

INSERT INTO [IO].[config].[detail_process_type]
   ([id],[master_process_type_id],[phase],[step],[entity],[name],[description],[source_type]
   ,[source_table_query])
VALUES
   (575, 5, 'PERSISTENCE', 6, 'insert_sale', 'Insert VehicleSales sale', 'Incorporación de los datos de load a VehicleSales.sale', 'BBDD'
   , 'EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_persistence_insert_sale] @delivery_id = ?, @affected_rows = ? OUTPUT')

------------------------------------------------------------------------
-- [IO].[load].[model_type]
CREATE TABLE [IO].[load].[model_type](
   [id] [int] IDENTITY(1,1) NOT NULL,
   delivery_id INT NOT NULL,
   mdb_id INT NULL,
   [name] [nvarchar](128) NULL,
   [brand_type_id] [int] NULL,
   [fuel_type_id] [int] NULL,
   [active] [bit] NULL,
   [creation_date] [datetime] NULL,
   [modification_date] [datetime] NULL,
   [io_id] [int] NULL,
 CONSTRAINT [pk_model_type] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

-- [IO].[load].[seller]
CREATE TABLE [IO].[load].[seller](
   [id] [int] IDENTITY(1,1) NOT NULL,
   delivery_id INT NOT NULL,
   mdb_id INT NULL,
   [name] [nvarchar](128) NULL,
   [phone] [nvarchar](18) NULL,
   [email] [nvarchar](256) NULL,
   [concessionaire_id] [int] NULL,
   [active] [bit] NULL,
   [creation_date] [datetime] NULL,
   [modification_date] [datetime] NULL,
   [io_id] [int] NULL,
 CONSTRAINT [pk_seller] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

-- [IO].[load].[customer]
CREATE TABLE [IO].[load].[customer](
   [id] [int] IDENTITY(1,1) NOT NULL,
   delivery_id INT NOT NULL,
   mdb_id INT NULL,
   [name] [nvarchar](128) NULL,
   [phone] [nvarchar](18) NULL,
   [email] [nvarchar](256) NULL,
   [creation_date] [datetime] NULL,
   [modification_date] [datetime] NULL,
   [io_id] [int] NULL,
 CONSTRAINT [pk_customer] PRIMARY KEY CLUSTERED
(
   [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

-- [IO].[load].[proposal]
CREATE TABLE [IO].[load].[proposal](
   [id] [int] IDENTITY(1,1) NOT NULL,
   delivery_id INT NOT NULL,
   mdb_id INT NULL,
   [customer_id] [int] NULL,
   [cliente] NVARCHAR(256) NULL,
   [seller_id] [int] NULL,
   [vendedor] NVARCHAR(256) NULL,
   [model_type_id] [int] NULL,
   [modelo] NVARCHAR(256) NULL,
   [proposal_date] [date] NULL,
   [proposal_amount] [decimal](10, 2) NULL,
   [proposal_type_id] [int] NULL,
```

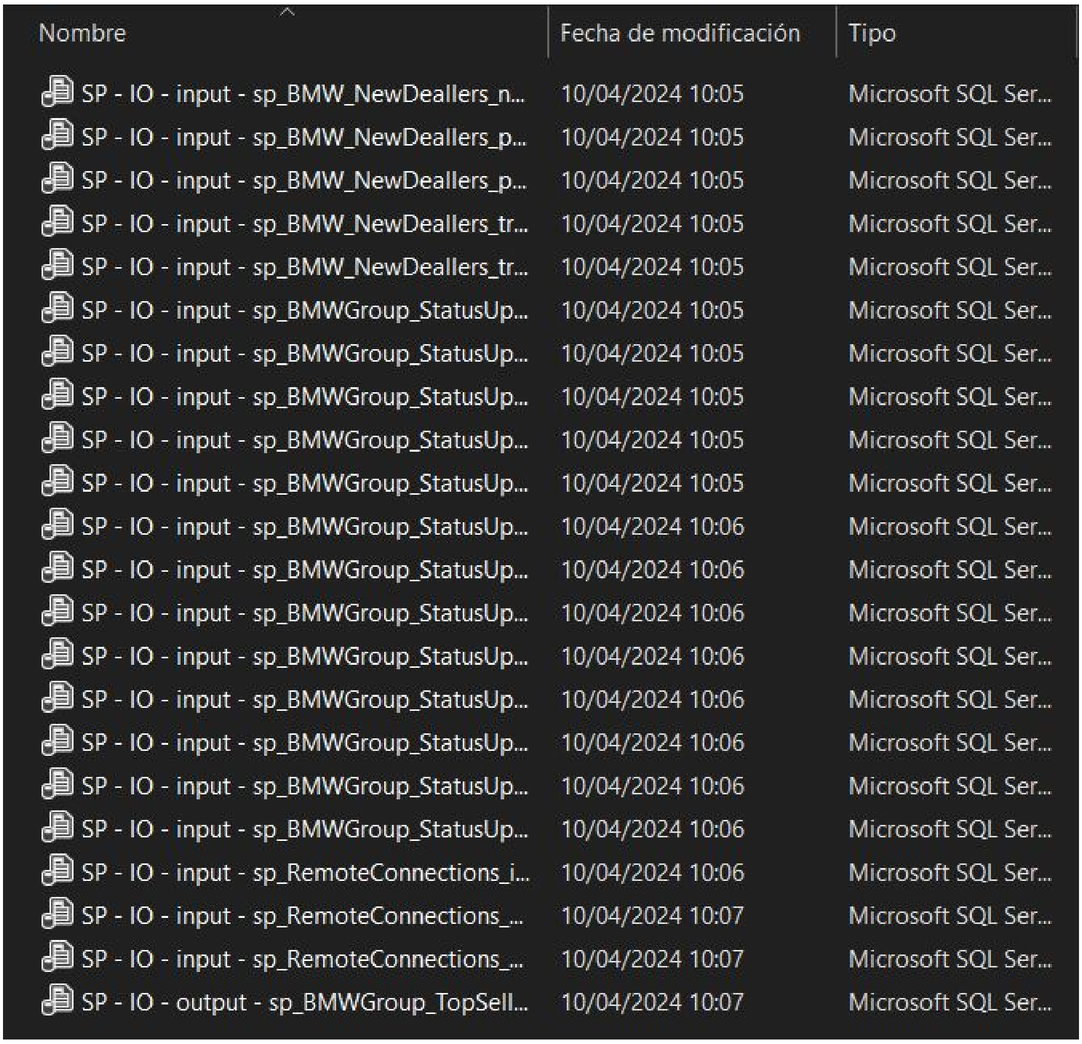
```
    [active] [bit] NULL,
    [creation_date] [datetime] NULL,
    [modification_date] [datetime] NULL,
    [io_id] [int] NULL,
 CONSTRAINT [pk_proposal] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

-- [IO].[load].[offer]
CREATE TABLE [IO].[load].[offer](
    [id] [int] IDENTITY(1,1) NOT NULL,
    delivery_id INT NOT NULL,
    mdb_id INT NULL,
    [proposal_id] [int] NULL,
    [proposal_io_id] [int] NULL,
    [offer_date] [date] NULL,
    [offer_amount] [decimal](10, 2) NULL,
    [active] [bit] NULL,
    [creation_date] [datetime] NULL,
    [modification_date] [datetime] NULL,
    [io_id] [int] NULL,
 CONSTRAINT [pk_offer] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

-- [IO].[load].[sale]
CREATE TABLE [IO].[load].[sale](
    [id] [int] IDENTITY(1,1) NOT NULL,
    delivery_id INT NOT NULL,
    mdb_id INT NULL,
    [offer_id] [int] NULL,
    [offer_io_id] [int] NULL,
    [sale_date] [date] NULL,
    [creation_date] [datetime] NULL,
    [modification_date] [datetime] NULL,
    [io_id] [int] NULL,
 CONSTRAINT [pk_sale] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF, DATA_COMPRESSION = PAGE) ON [PRIMARY]
) ON [PRIMARY]
GO

-- [IO].[auxiliar].[BMWGroup_model_type_mapping]
CREATE TABLE [IO].[auxiliar].[BMWGroup_model_type_mapping](
    [id] [tinyint] IDENTITY(1,1) NOT NULL,
    [model_type_file] NVARCHAR (128) NOT NULL,
    [model_type_table] NVARCHAR(128) NULL,
    [brand_type] NVARCHAR(64) NULL,
    [active] [bit] NOT NULL,
    [creation_date] [datetime] NULL,
    [modification_date] [datetime] NULL,
 CONSTRAINT [pk_BMWGroup_model_type_mapping] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY =
      OFF) ON [PRIMARY]
) ON [PRIMARY]
GO

ALTER TABLE [IO].[auxiliar].[BMWGroup_model_type_mapping] ADD DEFAULT (1) FOR [active]
GO

ALTER TABLE [IO].[auxiliar].[BMWGroup_model_type_mapping] ADD  CONSTRAINT [df_BMWGroup_model_type_mapping_creation_date] DEFAULT (getdate()) FOR
      [creation_date]
GO

-- [IO].[auxiliar].[tr_data_update_model_type]
CREATE TRIGGER [auxiliar].[tr_data_update_model_type] ON [auxiliar].[BMWGroup_model_type_mapping]
   AFTER UPDATE
   AS
   BEGIN
      -- Continua el trigger SOLO si hay resultados
      IF (@@ROWCOUNT = 0) RETURN;

      -- Ejecuta el trigger una sola vez (evita la recursividad)
      IF (TRIGGER_NESTLEVEL() > 1) RETURN;

      UPDATE [auxiliar].BMWGroup_model_type_mapping SET
         modification_date = GETDATE()
      WHERE id IN (SELECT id FROM inserted)
   END
GO

-- Insert mapeos de modelos
INSERT INTO [auxiliar].[BMWGroup_model_type_mapping] ([model_type_file], [model_type_table], [brand_type])
VALUES ('Serie 1', 'Serie 1', 'BMW'),
       ('Serie 2', 'Serie 2', 'BMW'),
       ('Serie 3', 'Serie 3', 'BMW'),
      ('Serie 4', 'Serie 4', 'BMW'),
```

```
       ('Serie 5', 'Serie 5', 'BMW'),
      ('Serie 6', 'Serie 6', 'BMW'),
       ('Serie 7', 'Serie 7', 'BMW'),
      ('Serie 8', 'Serie 8', 'BMW'),
       ('I3', 'I3', 'BMW'),
      ('I8', 'I8', 'BMW'),
       ('X1', 'X1', 'BMW'),
      ('X2', 'X2', 'BMW'),
       ('X3', 'X3', 'BMW'),
      ('X4', 'X4', 'BMW'),
       ('X5', 'X5', 'BMW'),
      ('X6', 'X6', 'BMW'),
       ('X7', 'X7', 'BMW'),
      ('Cooper', 'Cooper', 'Mini'),
       ('Countryman', 'Countryman', 'Mini'),
      ('Clubman', 'Clubman', 'Mini'),
       ('Convertible', 'Convertible', 'Mini'),
       ('Paceman', 'Paceman', 'Mini'),
      ('S3', 'Serie 3', 'BMW'),
      ('I4', 'I4', 'BMW'),
      ('X4 2024', 'X4', 'BMW'),
      ('M2', 'M2', 'BMW'),
      ('M3', 'M3', 'BMW'),
      ('Coper', 'Cooper', 'Mini'),
      ('Cbman', 'Clubman', 'Mini'),
      ('Cabrio', 'Cabrio', 'Mini'),
      ('5 Puertas', '5 Puertas', 'Mini'),
      ('John Coper Works', 'John Coper Works', 'Mini');

/*
DELETE [IO].[config].[master_process_type] WHERE id = 5
DELETE [IO].[config].[detail_process_type] WHERE master_process_type_id = 5
DROP TABLE [IO].[load].[model_type]
DROP TABLE [IO].[load].[seller]
DROP TABLE [IO].[load].[customer]
DROP TABLE [IO].[load].[proposal]
DROP TABLE [IO].[load].[sale]
DROP TABLE [IO].[load].[offer]
DROP TABLE [IO].[load].[proposal]
DROP TABLE [IO].[auxiliar].[BMWGroup_model_type_mapping]
*/
-- COMMIT
-- ROLLBACK

-- master_process_type y detail_process_type
SELECT * FROM IO.config.master_process_type WHERE id = 5
SELECT * FROM IO.config.detail_process_type WHERE master_process_type_id = 5

/*
-- ======================================= LIMPIA PROCESO ============================================================
-- ===================================================================================================================
TRUNCATE TABLE [IO].[input].[BMWGroup_StatusUpdate]
TRUNCATE TABLE [IO].[hist].[BMWGroup_StatusUpdate]
TRUNCATE TABLE [IO].[load].[model_type]
TRUNCATE TABLE [IO].[load].[seller]
TRUNCATE TABLE [IO].[load].[customer]
TRUNCATE TABLE [IO].[load].[proposal]
TRUNCATE TABLE [IO].[load].[offer]
TRUNCATE TABLE [IO].[load].[sale]

DELETE FROM IO.load.delivery DBCC CHECKIDENT ('IO.load.delivery',RESEED, 0)
TRUNCATE TABLE IO.load.operation
TRUNCATE TABLE IO.monitor.log
TRUNCATE TABLE IO.monitor.detail_process_log
DELETE FROM IO.monitor.master_process_log DBCC CHECKIDENT ('IO.monitor.master_process_log',RESEED, 0)

DELETE FROM [VehicleSales].[dbo].[sale] WHERE id > 30
DBCC CHECKIDENT ('[VehicleSales].[dbo].[sale]',RESEED, 30)
DELETE FROM [VehicleSales].[dbo].[offer] WHERE id > 40
DBCC CHECKIDENT ('[VehicleSales].[dbo].[offer]',RESEED, 40)
DELETE FROM [VehicleSales].[dbo].[proposal] WHERE id > 50
DBCC CHECKIDENT ('[VehicleSales].[dbo].[proposal]',RESEED, 50)
DELETE FROM [VehicleSales].[dbo].[customer] WHERE id > 20
DBCC CHECKIDENT ('[VehicleSales].[dbo].[customer]',RESEED, 20)
DELETE FROM [VehicleSales].[dbo].[seller] WHERE id > 20
DBCC CHECKIDENT ('[VehicleSales].[dbo].[seller]',RESEED, 20)
DELETE FROM [VehicleSales].[dbo].[model_type] WHERE id > 22
DBCC CHECKIDENT ('[VehicleSales].[dbo].[model_type]',RESEED, 22)
*/

-- ===================================================================================================================
-- ===================================================================================================================

SELECT TOP 100 * FROM IO.monitor.log ORDER BY id DESC
SELECT TOP 100 * FROM IO.monitor.master_process_log WHERE master_process_type_id = 5 ORDER BY id DESC
SELECT TOP 100 * FROM IO.monitor.detail_process_log ORDER BY id DESC
SELECT TOP 100 * FROM IO.load.delivery ORDER BY id DESC
SELECT TOP 100 * FROM IO.load.operation

SELECT * FROM [IO].[hist].BMWGroup_StatusUpdate
SELECT * FROM [IO].[input].BMWGroup_StatusUpdate
SELECT * FROM [IO].[auxiliar].[BMWGroup_model_type_mapping]
SELECT * FROM [IO].[load].model_type
SELECT * FROM [IO].[load].customer
SELECT * FROM [IO].[load].seller
SELECT * FROM [IO].[load].proposal
SELECT * FROM [IO].[load].offer
SELECT * FROM [IO].[load].sale
SELECT * FROM VehicleSales.dbo.model_type
```

```
SELECT * FROM VehicleSales.dbo.customer
SELECT * FROM VehicleSales.dbo.seller
SELECT * FROM VehicleSales.dbo.proposal
SELECT * FROM VehicleSales.dbo.offer
SELECT * FROM VehicleSales.dbo.sale

/*
SELECT * FROM VehicleSales.dbo.proposal_type
SELECT * FROM VehicleSales.dbo.concessionaire

-- DISTINCT para dar de alta los maestros
*/


-- ======================== MAIN ========================
-- Tarea "start ingestion log" de los procesos para generar la traza en monitor.master_process_log
-- EXEC [monitor].[create_or_update_master_process_log] @p_master_process_id = 1, @p_in_process_log_id = 1, @p_step = 'Beginning'
SELECT * FROM [IO].monitor.master_process_log

-- {EXEC [IO].monitor.get_execution_summary @master_process_type_id = 1, @master_process_log_id = 4}

-- ======================== INGESTA ========================
-- Parámetro User::sql_get_main_config
EXEC config.sp_open_io_password_encryption_keys
SELECT d.[source_type] AS source_type,
   COALESCE(d.[source_file_folder_path], '') AS source_file_folder_path,
   COALESCE(d.[source_file_pattern], '') AS source_file_pattern,
   COALESCE(d.[skip_initial_rows], 0) AS skip_initial_rows,
   COALESCE(d.[source_table_query], '') AS source_table_query,
   COALESCE(d.[destination_table], '') AS destination_table,
   COALESCE(d. [source_file_template_path], '') AS  [source_file_template_path],
   m.[name] AS process_name,
   COALESCE(d.[zip_folder_path], '') AS zip_folder_path,
   COALESCE(d.[zip_file_path], '') AS zip_file_path,
   COALESCE(d.[ftp_server], '') AS ftp_server,
   COALESCE(d.[ftp_port], '') AS ftp_port,
   COALESCE(d.[ftp_user], '') AS ftp_user,
   COALESCE([IO].config.f_password_decryption(d.[ftp_pass]), '') AS ftp_pass,
   COALESCE(d.[ftp_operation], '') AS ftp_operation,
   COALESCE(d.[ftp_folder], '') AS ftp_folder,
   COALESCE(d.[ftp_file_pattern], '') AS ftp_file_pattern,
   COALESCE(d.[ftp_local_folder], '') AS ftp_local_folder,
   COALESCE(d.[ftp_local_file_pattern], '') AS ftp_local_file_pattern,
   COALESCE(d.[skip_final_rows], 0) AS skip_final_rows,
   COALESCE(d.[add_eof], 0) AS add_eof,
   COALESCE(d.[ftp_private_key_path], '') AS ftp_private_key_path,
   COALESCE(d.[ftp_ssh_host_key_fingerprint], '') AS ftp_ssh_host_key_fingerprint,
   COALESCE([IO].config.f_password_decryption(d.[ftp_private_key_passphrase]), '') AS ftp_private_key_passphrase,
   COALESCE(d.[source_connection_server], '') AS source_connection_server,
   COALESCE(d.[source_connection_database], '') AS source_connection_database,
   COALESCE(d.[source_connection_user], '') AS source_connection_user,
   COALESCE([IO].config.f_password_decryption(d.[source_connection_password]), '') AS source_connection_password,
   m.customer AS customer,
   COALESCE(d.check_ingestion_filename, 0) AS check_ingestion_filename,
   COALESCE(m.stop_on_warning, 0) AS stop_on_warning,
   COALESCE(d.ingestion_encoding_file, '') AS ingestion_encoding_file,
   COALESCE([IO].config.f_password_decryption(d.zip_rar_pass), '') AS zip_rar_pass
FROM IO.config.master_process_type m
INNER JOIN IO.config.detail_process_type d ON m.id = d.master_process_type_id
WHERE m.id = 5 AND d.phase = 'INGESTION' ORDER BY d.step ASC

-- Tarea "start detail log" de los procesos para generar la traza en monitor.detail_process_log
-- EXEC [monitor].[create_or_update_detail_process_log] @p_master_log_process_id = 1, @p_step = ?, @p_phase = ?, @p_entity = ?,
      @p_out_detail_process_log_id = ? OUTPUT

-- Tarea "sp_load_file_to_table"
-- EXEC [input].[sp_load_file_to_table] @p_ingest = ?, @p_path = ?, @p_template = ?, @p_skip_initial_rows = ?, @p_skip_final_rows = ?,
      @p_destination_table = ?, @p_order = ?

--Ejemplo de ejecución con formato ";":
EXEC [input].[sp_load_file_to_table]
   @p_ingest = 1
   ,@p_path = '\\WIN-SER-AMAROTO\IO_Files\Input\BMW_Group\InformacionCore\BMW_StatusUpdate_20240322.xlsx'
   ,@p_template = '\\WIN-SER-AMAROTO\IO_Files\Template\BMW_Group\Input_StatusUpdate.fmt'
   ,@p_skip_initial_rows = 1
   ,@p_skip_final_rows = 0
   ,@p_destination_table = '[IO].[input].[BMWGroup_StatusUpdate]'
   ,@p_order  = 1

EXEC [input].[sp_load_file_to_table]
   @p_ingest = 1
   ,@p_path = '\\WIN-SER-AMAROTO\IO_Files\Input\BMW_Group\InformacionCore\Mini_StatusUpdate_20240322.xlsx'
   ,@p_template = '\\WIN-SER-AMAROTO\IO_Files\Template\BMW_Group\Input_StatusUpdate.fmt'
   ,@p_skip_initial_rows = 1
   ,@p_skip_final_rows = 0
   ,@p_destination_table = '[IO].[input].[BMWGroup_StatusUpdate]'
   ,@p_order  = 2

SELECT * FROM [IO].[input].BMWGroup_StatusUpdate
SELECT * FROM [IO].[hist].BMWGroup_StatusUpdate

/*
TRUNCATE TABLE [IO].[input].[BMWGroup_StatusUpdate]
TRUNCATE TABLE [IO].[hist].[BMWGroup_StatusUpdate]
*/

-- ======================== TRANSFORMACION ========================
-- Tarea "Get delivery config" devuelve:
SELECT dpt.source_table_query AS delivery_config
```

```
FROM IO.config.master_process_type mpt
   INNER JOIN IO.config.detail_process_type dpt ON dpt.master_process_type_id = mpt.id
WHERE mpt.id = 5 AND dpt.phase = 'DELIVERY'

-- Delivery
-- EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_new_delivery] @master_id = 5, @master_log_id = 1
SELECT * FROM IO.load.delivery ORDER BY id DESC

-- Tarea "Get transformation config" devuelve:
SELECT dpt.source_table_query AS query, dpt.entity AS entity
FROM IO.config.master_process_type mpt
   INNER JOIN IO.config.detail_process_type dpt ON dpt.master_process_type_id = mpt.id
WHERE mpt.id = 5 AND dpt.phase = 'TRANSFORMATION'
ORDER BY dpt.step ASC

-- INSERT model_type
-- EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_transformation_insert_model_type] @master_process_id = 5
SELECT * FROM [IO].[load].model_type

-- INSERT seller
-- EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_transformation_insert_seller] @master_process_id = 5
SELECT * FROM [IO].[load].seller

-- INSERT customer
-- EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_transformation_insert_customer] @master_process_id = 5
SELECT * FROM [IO].[load].customer

-- INSERT proposal
-- EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_transformation_insert_proposal] @master_process_id = 5
SELECT * FROM [IO].[load].proposal

-- INSERT offer
-- EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_transformation_insert_offer] @master_process_id = 5
SELECT * FROM [IO].[load].offer

-- INSERT sale
-- EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_transformation_insert_sale] @master_process_id = 5
SELECT * FROM [IO].[load].sale

SELECT * FROM IO.load.operation

-- ======================== PERSISTENCIA ========================

-- Valor variable "sql_get_pending_deliveries"
SELECT dv.id as delivery_id, COALESCE(dv.customer_id, '0') as customer_id
FROM [IO].load.delivery dv
   INNER JOIN [IO].monitor.master_process_log mpl ON mpl.id = dv.master_process_log_id
   INNER JOIN [IO].config.master_process_type mpt ON mpt.id = mpl.master_process_type_id
WHERE dv.registration_date is null
   AND mpt.id = 5 ORDER BY dv.id

-- Tarea "Get persistence config" devuelve:
EXEC [IO].config.sp_open_io_password_encryption_keys
SELECT dpt.source_table_query AS query, dpt.entity AS entity, COALESCE(dpt.destination_connection_server, '') AS destination_connection_server,
      COALESCE(dpt.destination_connection_database, '') AS destination_connection_database, COALESCE(dpt.destination_connection_user, '') AS
      destination_connection_user, COALESCE([IO].config.f_password_decryption(dpt.destination_connection_password), '') AS
      destination_connection_password
FROM IO.config.master_process_type mpt
   INNER JOIN IO.config.detail_process_type dpt ON dpt.master_process_type_id = mpt.id
WHERE mpt.id = 5 AND dpt.phase = 'PERSISTENCE' ORDER BY dpt.step ASC

--
-- INSERT model_type
-- EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_persistence_insert_model_type] @delivery_id = 5, @affected_rows = 0
SELECT * FROM [VehicleSales].[dbo].model_type

-- INSERT seller
-- EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_persistence_insert_seller] @delivery_id = 5, @affected_rows = 0
SELECT * FROM [VehicleSales].[dbo].seller

-- INSERT customer
-- EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_persistence_insert_customer] @delivery_id = 5, @affected_rows = 0
SELECT * FROM [VehicleSales].[dbo].customer

-- INSERT proposal
-- EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_persistence_insert_proposal] @delivery_id = 5, @affected_rows = 0
SELECT * FROM [VehicleSales].[dbo].proposal

-- INSERT offer
-- EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_persistence_insert_offer] @delivery_id = 5, @affected_rows = 0
SELECT * FROM [VehicleSales].[dbo].offer

-- INSERT sale
-- EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_persistence_insert_sale] @delivery_id = 5, @affected_rows = 0
SELECT * FROM [VehicleSales].[dbo].sale
--

SELECT * FROM IO.load.operation
--UPDATE IO.load.operation SET operation_finished = 0 WHERE load_table_name = 'proposal'

--UPDATE [IO].load.delivery SET registration_date = GETDATE() WHERE id = 1
```

La *Figura C.7* muestra el conjunto de procedimientos SQL de los casos de uso del proyecto IO.



Nombre	Fecha de modificación	Tipo
SP - IO - input - sp_BMW_NewDeallers_n...	10/04/2024 10:05	Microsoft SQL Ser...
SP - IO - input - sp_BMW_NewDeallers_p...	10/04/2024 10:05	Microsoft SQL Ser...
SP - IO - input - sp_BMW_NewDeallers_p...	10/04/2024 10:05	Microsoft SQL Ser...
SP - IO - input - sp_BMW_NewDeallers_tr...	10/04/2024 10:05	Microsoft SQL Ser...
SP - IO - input - sp_BMW_NewDeallers_tr...	10/04/2024 10:05	Microsoft SQL Ser...
SP - IO - input - sp_BMWGroup_StatusUp...	10/04/2024 10:05	Microsoft SQL Ser...
SP - IO - input - sp_BMWGroup_StatusUp...	10/04/2024 10:05	Microsoft SQL Ser...
SP - IO - input - sp_BMWGroup_StatusUp...	10/04/2024 10:05	Microsoft SQL Ser...
SP - IO - input - sp_BMWGroup_StatusUp...	10/04/2024 10:05	Microsoft SQL Ser...
SP - IO - input - sp_BMWGroup_StatusUp...	10/04/2024 10:05	Microsoft SQL Ser...
SP - IO - input - sp_BMWGroup_StatusUp...	10/04/2024 10:06	Microsoft SQL Ser...
SP - IO - input - sp_BMWGroup_StatusUp...	10/04/2024 10:06	Microsoft SQL Ser...
SP - IO - input - sp_BMWGroup_StatusUp...	10/04/2024 10:06	Microsoft SQL Ser...
SP - IO - input - sp_BMWGroup_StatusUp...	10/04/2024 10:06	Microsoft SQL Ser...
SP - IO - input - sp_BMWGroup_StatusUp...	10/04/2024 10:06	Microsoft SQL Ser...
SP - IO - input - sp_BMWGroup_StatusUp...	10/04/2024 10:06	Microsoft SQL Ser...
SP - IO - input - sp_BMWGroup_StatusUp...	10/04/2024 10:06	Microsoft SQL Ser...
SP - IO - input - sp_BMWGroup_StatusUp...	10/04/2024 10:06	Microsoft SQL Ser...
SP - IO - input - sp_RemoteConnections_i...	10/04/2024 10:06	Microsoft SQL Ser...
SP - IO - input - sp_RemoteConnections_...	10/04/2024 10:07	Microsoft SQL Ser...
SP - IO - input - sp_RemoteConnections_...	10/04/2024 10:07	Microsoft SQL Ser...
SP - IO - output - sp_BMWGroup_TopSell...	10/04/2024 10:07	Microsoft SQL Ser...

Figura C.7: Estructura de procedimientos SQL de los casos de uso del proyecto IO

A continuación el *Código C.40-C.44* muestra los procedimientos SQL para el caso de uso 1 - Input - NewDealers.

Código C.40: input.sp_BMW_NewDealers_new_delivery.sql

```sql
USE [IO];
GO
/*
-- ==========================================================================================
Autor: Adrian Maroto
Fecha creacion: 14/03/2024
Descripcion: Crea una nueva delivery para el proceso BMW NewDealers.
Cambios:

Entrada:
   @master_id: Id configuración maestra del proceso.
   @master_log_id: Id log de la ejecución en curso.

Salida:

Ejemplo:
   EXEC [IO].[monitor].[create_or_update_master_process_log] @p_in_process_log_id = 1, @p_step = 'Beginning'
   EXEC [IO].[input].[sp_BMW_NewDealers_new_delivery] @master_id = 1, @master_log_id = 1

   SELECT * FROM [IO].monitor.master_process_log ORDER BY id DESC
   SELECT * FROM [IO].[load].delivery ORDER BY id DESC
   SELECT * FROM [IO].config.customer_cluster_type
-- ==========================================================================================
*/
CREATE OR ALTER PROCEDURE [input].[sp_BMW_NewDealers_new_delivery]
   @master_id INT,
   @master_log_id INT
```

```sql
AS
BEGIN
   -- Variables de logging en caso de error
   DECLARE @msg VARCHAR(1024)
   DECLARE @error_values VARCHAR(1024)

   -- Eliminación de la tabla temporal en caso de que exista
   DROP TABLE IF EXISTS #incorporation_temp

   -- Se calcula el/los cluster_id basándose en los names que contiene la tabla del archivo ingestado
   SELECT DISTINCT COALESCE(cct.cluster_id, 0) AS customer_id
   INTO #incorporation_temp
   FROM [IO].config.customer_cluster_type cct
   WHERE cluster_name = 'BMW_Group'

   -- Se verifica la existencia de una delivery abierta
   IF EXISTS (
      SELECT * FROM #incorporation_temp tt
      INNER JOIN [IO].[load].delivery dv ON dv.customer_id = tt.customer_id
      INNER JOIN [IO].monitor.master_process_log mpl ON dv.master_process_log_id = mpl.id
      INNER JOIN [IO].config.master_process_type mpt ON mpl.master_process_type_id = mpt.id
      WHERE mpt.id = @master_id AND dv.registration_date IS NULL
   )
      -- En caso afirmativo, se genera el error adecuado sobre el master process id asociado
      BEGIN
         SET @msg = 'There is some pending delivery associated to main process ' + CONVERT(VARCHAR(10), @master_id)
         EXEC [IO].[monitor].[sp_write_log] @ProcId = @@PROCID, @TraceId = '', @Message = @msg , @Severity = 'ERROR'
      END
   ELSE
      -- En caso negativo, se generan las deliveries necesarias según los clientes/clusters
      BEGIN
         INSERT INTO [IO].[load].delivery (customer_id, master_process_log_id)
         SELECT customer_id, @master_log_id FROM #incorporation_temp
      END
END
```

Código C.41: input.sp_BMW_NewDealers_transformation_insert_concessionaire.sql

```sql
USE [IO];
GO
/*
-- ================================================================================
Autor: Adrian Maroto
Fecha creacion: 14/03/2024
Descripcion: Proceso de inserción de concesionarios del fichero BMW_NewDealers en [IO].[load].concessionare.
Cambios:

Entrada:
   @master_process_id: Id configuración maestra del proceso.

Salida:

Ejemplo:
   EXEC [IO].[input].[sp_BMW_NewDealers_transformation_insert_concessionaire] @master_process_id = 1

   SELECT * FROM VehicleSales.dbo.brand_type
   SELECT * FROM VehicleSales.dbo.concessionaire

   SELECT * FROM [IO].config.customer_cluster_type
   SELECT * FROM [IO].[load].delivery ORDER BY id DESC
   SELECT * FROM [IO].[load].operation
   SELECT * FROM [IO].[load].operation_type

   SELECT * FROM [IO].[input].[BMW_NewDealers]
   SELECT * FROM [IO].[load].[concessionaire]
-- ================================================================================
*/
CREATE OR ALTER PROCEDURE [input].[sp_BMW_NewDealers_transformation_insert_concessionaire]
   @master_process_id INT
AS
BEGIN
   -- Se insertan en la tabla concessionare del schema [load] los nuevos concesionarios provenientes del archivo
   -- que no existan ya en la tabla
   INSERT INTO [IO].[load].[concessionaire]
      ([delivery_id], [mdb_id], [name], [address], [city], [country], [postal_code], [info], [active])
   SELECT
      delivery_id,
      mdb_id,
      name,
      address,
      city,
      country,
      postal_code,
      info,
      active
   FROM (
      -- Se obtienen todos los datos de los concesionarios nuevos
      SELECT
         d.id AS delivery_id,
         NULL AS mdb_id,
         nd.Nombre AS [name],
         nd.Direccion AS [address],
         nd.Ciudad AS city,
         nd.Pais AS country,
         nd.CodigoPostal AS postal_code,
         TRIM(nd.Auxiliar1 + ' - ' + nd.Auxiliar2) AS info,
         1 AS active
```

```
        FROM [IO].[input].[BMW_NewDealers] nd
            LEFT JOIN VehicleSales.dbo.concessionaire c ON [auxiliar].[f_clean_rare_characters](nd.[nombre]) =
        [auxiliar].[f_clean_rare_characters](c.[name])
            INNER JOIN [IO].config.customer_cluster_type cct ON 1 = cct.customer_id
            INNER JOIN [IO].[load].delivery d ON d.customer_id = cct.cluster_id
            INNER JOIN [IO].monitor.master_process_log mpl ON d.master_process_log_id = mpl.id
        WHERE
            -- Concesionarios nuevos
            c.id IS NULL
            -- Pertenece al proceso maestro adecuado
            AND mpl.master_process_type_id = @master_process_id
            -- Tiene delivery abierta
            AND d.registration_date IS NULL
    ) sub

    -- Se insertan todas las operaciones INSERT de los registros calculados anteriormente
    INSERT INTO [IO].[load].operation
        ([delivery_id], [load_table_name], [load_reference_id], [master_reference_id], [operation_type_id])
    SELECT DISTINCT
        d.id AS delivery_id,
        'concessionaire' AS load_table_name,
        c.id AS load_reference_id,
        c.mdb_id AS master_reference_id,
        (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Insert') AS operation_type_id
    FROM [IO].[load].[concessionaire] c
        INNER JOIN [IO].[load].delivery d ON c.delivery_id= d.id
        INNER JOIN [IO].monitor.master_process_log mpl ON d.master_process_log_id = mpl.id
    WHERE
        -- Son concesionarios nuevos
        c.mdb_id IS NULL
        -- Pertenece al proceso maestro adecuado
        AND mpl.master_process_type_id = @master_process_id
        -- Tiene delivery abierta
        AND d.registration_date IS NULL

    -- Se devuelve el número de operaciones almacenadas
    SELECT @@ROWCOUNT AS operations_quantity

END
```

Código C.42: input.sp_BMW_NewDealers_transformation_update_concessionaire.sql

```
USE [IO];
GO
/*
-- ============================================================================================
Autor: Adrian Maroto
Fecha creacion: 14/03/2024
Descripcion: Proceso de actualización de concesionarios del fichero BMW_NewDealers en [IO].[load].concessionare.
Cambios:

Entrada:
    @master_process_id: Id configuración maestra del proceso.

Salida:

Ejemplo:
    EXEC [IO].[input].[sp_BMW_NewDealers_transformation_update_concessionaire] @master_process_id = 1

    SELECT * FROM VehicleSales.dbo.brand_type
    SELECT * FROM VehicleSales.dbo.concessionaire

    SELECT * FROM [IO].config.customer_cluster_type
    SELECT * FROM [IO].[load].delivery ORDER BY id DESC
    SELECT * FROM [IO].[load].operation
    SELECT * FROM [IO].[load].operation_type

    SELECT * FROM [IO].[input].[BMW_NewDealers]
    SELECT * FROM [IO].[load].[concessionaire]
-- ============================================================================================
*/
CREATE OR ALTER PROCEDURE [input].[sp_BMW_NewDealers_transformation_update_concessionaire]
    @master_process_id INT
AS
BEGIN
    -- Se insertan en la tabla concessionare del schema [load] los concesionarios existentes provenientes del archivo
    -- que no existan ya en la tabla
    INSERT INTO [IO].[load].[concessionaire]
        ([delivery_id], [mdb_id], [name], [address], [city], [country], [postal_code], [info], [active])
    SELECT
        delivery_id,
        mdb_id,
        name,
        address,
        city,
        country,
        postal_code,
        info,
        active
    FROM (
        -- Se obtienen todos los datos de los concesionarios nuevos
        SELECT
            d.id AS delivery_id,
            c.id AS mdb_id,
            nd.Nombre AS [name],
            nd.Direccion AS [address],
            nd.Ciudad AS city,
```

```
        nd.Pais AS country,
        nd.CodigoPostal AS postal_code,
        TRIM(nd.Auxiliar1 + ' - ' + nd.Auxiliar2) AS info,
        1 AS active
    FROM [IO].[input].[BMW_NewDealers] nd
        INNER JOIN VehicleSales.dbo.concessionaire c ON [auxiliar].[f_clean_rare_characters](nd.[nombre]) =
    [auxiliar].[f_clean_rare_characters](c.[name])
        INNER JOIN [IO].config.customer_cluster_type cct ON 1 = cct.customer_id
        INNER JOIN [IO].[load].delivery d ON d.customer_id = cct.cluster_id
        INNER JOIN [IO].monitor.master_process_log mpl ON d.master_process_log_id = mpl.id
    WHERE
        -- Pertenece al proceso maestro adecuado
        mpl.master_process_type_id = @master_process_id
        -- Tiene delivery abierta
        AND d.registration_date IS NULL
) sub

-- Se insertan todas las operaciones INSERT de los registros calculados anteriormente
INSERT INTO [IO].[load].operation
    ([delivery_id], [load_table_name], [load_reference_id], [master_reference_id], [operation_type_id])
SELECT DISTINCT
    d.id AS delivery_id,
    'concessionaire' AS load_table_name,
    c.id AS load_reference_id,
    c.mdb_id AS master_reference_id,
    (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Update') AS operation_type_id
FROM [IO].[load].[concessionaire] c
    INNER JOIN [IO].[load].delivery d ON c.delivery_id= d.id
    INNER JOIN [IO].monitor.master_process_log mpl ON d.master_process_log_id = mpl.id
WHERE
    -- Son concesionarios existentes
    c.mdb_id IS NOT NULL
    -- Pertenece al proceso maestro adecuado
    AND mpl.master_process_type_id = @master_process_id
    -- Tiene delivery abierta
    AND d.registration_date IS NULL

-- Se devuelve el número de operaciones almacenadas
SELECT @@ROWCOUNT AS operations_quantity

END
```

Código C.43: input.sp_BMW_NewDealers_persistence_insert_concessionaire.sql

```
USE [IO];
GO
/*
-- ================================================================================================
Autor: Adrian Maroto
Fecha creacion: 15/03/2024
Descripcion: Proceso de inserción de concesionarios del fichero BMW_NewDealers en VehicleSales.dbo.concessionare.
Cambios:

Entrada:
    @delivery_id: Id de la delivery abierta en curso.

Salida:
    @affected_rows: Número de filas afectadas por el proceso.

Ejemplo:
    EXEC [IO].[input].[sp_BMW_NewDealers_persistence_insert_concessionaire] @delivery_id = 1, @affected_rows = 0

    SELECT * FROM VehicleSales.dbo.brand_type
    SELECT * FROM VehicleSales.dbo.concessionaire

    SELECT * FROM [IO].config.customer_cluster_type
    SELECT * FROM [IO].[load].delivery ORDER BY id DESC
    SELECT * FROM [IO].[load].operation
    SELECT * FROM [IO].[load].operation_type

    SELECT * FROM [IO].[input].[BMW_NewDealers]
    SELECT * FROM [IO].[load].[concessionaire]
-- ================================================================================================
*/
CREATE OR ALTER PROCEDURE [input].[sp_BMW_NewDealers_persistence_insert_concessionaire]
    @delivery_id INT,
    @affected_rows INT OUTPUT
AS
BEGIN

    -- Se realizan todas las operaciones de INSERT calculadas en el proceso de transformación
    -- [IO].[input].[sp_BMW_NewDealers_transformation_insert_concessionaire]
    INSERT INTO VehicleSales.dbo.concessionaire
        ([name], [address], [city], [country], [postal_code], [info], [active], [io_id])
    SELECT DISTINCT
        c.name,
        c.address,
        c.city,
        c.country,
        c.postal_code,
        c.info,
        c.active,
        c.id AS io_id
    FROM [IO].[load].[concessionaire] c
        INNER JOIN [IO].[load].operation o ON o.delivery_id = c.delivery_id
            AND o.load_reference_id = c.id
    WHERE
```

```
      -- Tabla correcta
      o.load_table_name = 'concessionaire'
      -- Operación de tipo Insert
      AND o.operation_type_id = (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Insert')
      -- Operación sin persistir
      AND o.operation_finished = 0
      -- Para la delivery en curso
      AND o.delivery_id = @delivery_id

   -- Se finalizan (operation_finished = 1) las operaciones realizadas en el paso anterior
   UPDATE o SET operation_finished = 1
   FROM [IO].[load].concessionaire c
      INNER JOIN [IO].[load].operation o ON o.delivery_id = c.delivery_id
         AND o.load_reference_id = c.id
   WHERE
      -- Tabla correcta
      o.load_table_name = 'concessionaire'
      -- Operación de tipo Insert
      AND o.operation_type_id = (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Insert')
      -- Operación sin persistir
      AND o.operation_finished = 0
      -- Para la delivery en curso
      AND o.delivery_id = @delivery_id

   -- Se devuelve el número de operaciones actualizadas
   SELECT @affected_rows = @@ROWCOUNT

END
```

Código C.44: input.sp_BMW_NewDealers_persistence_update_concessionaire.sql

```
USE [IO];
GO
/*
-- ================================================================================
Autor: Adrian Maroto
Fecha creacion: 15/03/2024
Descripcion: Proceso de actualización de concesionarios del fichero BMW_NewDealers en VehicleSales.dbo.concessionare.
Cambios:

Entrada:
   @delivery_id: Id de la delivery abierta en curso.

Salida:
   @affected_rows: Número de filas afectadas por el proceso.

Ejemplo:
   EXEC [IO].[input].[sp_BMW_NewDealers_persistence_update_concessionaire] @delivery_id = 1, @affected_rows = 0

   SELECT * FROM VehicleSales.dbo.brand_type
   SELECT * FROM VehicleSales.dbo.concessionaire

   SELECT * FROM [IO].config.customer_cluster_type
   SELECT * FROM [IO].[load].delivery ORDER BY id DESC
   SELECT * FROM [IO].[load].operation
   SELECT * FROM [IO].[load].operation_type

   SELECT * FROM [IO].[input].[BMW_NewDealers]
   SELECT * FROM [IO].[load].[concessionaire]
-- ================================================================================
*/
CREATE OR ALTER PROCEDURE [input].[sp_BMW_NewDealers_persistence_update_concessionaire]
   @delivery_id INT,
   @affected_rows INT OUTPUT
AS
BEGIN

   -- Se realizan todas las operaciones de UPDATE calculadas en el proceso de transformación
   -- [IO].[input].[sp_BMW_NewDealers_transformation_update_concessionaire]
   UPDATE vc SET
      vc.[address] = c.[address],
      vc.city = c.city,
      vc.country = c.country,
      vc.postal_code = c.postal_code,
      vc.info = c.info,
      vc.active = c.active,
      vc.io_id = c.id
   FROM VehicleSales.dbo.concessionaire vc
      INNER JOIN [IO].[load].[concessionaire] c ON vc.id = c.mdb_id
      INNER JOIN [IO].[load].operation o ON o.delivery_id = c.delivery_id
         AND o.load_reference_id = c.id
   WHERE
      -- Tabla correcta
      o.load_table_name = 'concessionaire'
      -- Operación de tipo Update
      AND o.operation_type_id = (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Update')
      -- Operación sin persistir
      AND o.operation_finished = 0
      -- Para la delivery en curso
      AND o.delivery_id = @delivery_id

   -- Se finalizan (operation_finished = 1) las operaciones realizadas en el paso anterior que existan en destino
   UPDATE o SET operation_finished = 1
   FROM [IO].[load].concessionaire c
      -- Que existan en destino
      INNER JOIN VehicleSales.dbo.concessionaire vc ON vc.id = c.mdb_id
      INNER JOIN [IO].[load].operation o ON o.delivery_id = c.delivery_id
```

```
        AND o.load_reference_id = c.id
   WHERE
      -- Tabla correcta
      o.load_table_name = 'concessionaire'
      -- Operación de tipo Update
      AND o.operation_type_id = (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Update')
      -- Operación sin persistir
      AND o.operation_finished = 0
      -- Para la delivery en curso
      AND o.delivery_id = @delivery_id

   -- Se devuelve el número de operaciones actualizadas
   SELECT @affected_rows = @@ROWCOUNT

END
```

A continuación el *Código C.45-C.48* muestra los procedimientos SQL para el caso de uso 2 - Output - TopSellers.

Código C.45: output.sp_BMWGroup_TopSellers_report.sql

```
USE [IO];
GO
/*
-- ========================================================================================
Autor: Adrian Maroto
Fecha creacion: 18/03/2024
Descripcion: Proceso de generación de report de TopSellers para BMW Group.
Cambios:

Entrada:

Salida:
   - Conjunto de filas con los resultados.

Ejemplo:
   EXEC [IO].[output].[sp_BMWGroup_TopSellers_report]

   SELECT * FROM [VehicleSales].dbo.concessionaire
   SELECT * FROM [VehicleSales].dbo.brand_type
   SELECT * FROM [VehicleSales].dbo.offer
   SELECT * FROM [VehicleSales].dbo.proposal
   SELECT * FROM [VehicleSales].dbo.sale
   SELECT * FROM [VehicleSales].dbo.seller
-- ========================================================================================
*/
CREATE OR ALTER PROCEDURE [output].[sp_BMWGroup_TopSellers_report]
AS
BEGIN

   -- Creacion tabla temporal
   DECLARE @report_table TABLE (
      [concesionario] NVARCHAR(128)
      ,[vendedor] NVARCHAR(128)
      --
      ,[ventas_totales] INT
      ,[importe_ventas_totales] DECIMAL(10,2)
      ,[fecha_ultima_venta] DATE
      ,[ventas_ultimo_mes] INT
      ,[promedio_ventas_mensuales] INT
      ,[promedio_importe_ventas] DECIMAL(10,2)
      --
      ,[ofertas_totales] INT
      ,[fecha_ultima_oferta] DATE
      --
      ,[propuestas_totales] INT
      ,[fecha_ultima_propuesta] DATE
   )

   -- Generación del report e inserción en tabla temporal
   INSERT INTO @report_table
   SELECT

      -- Concesionario
      c.[name] AS [concesionario]
      -- Vendedor
      ,s.[name] AS [vendedor]

      -- Ventas totales
      ,(SELECT COUNT(subs.id) FROM [VehicleSales].dbo.sale subs
         INNER JOIN [VehicleSales].dbo.offer subo ON subs.offer_id = subo.id
         INNER JOIN [VehicleSales].dbo.proposal subp ON subp.id = subo.proposal_id
         WHERE subp.seller_id = s.id) AS [ventas_totales]
      -- Importe ventas totales
      ,(SELECT ROUND(SUM(subo.offer_amount),2) FROM [VehicleSales].dbo.sale subs
         INNER JOIN [VehicleSales].dbo.offer subo ON subs.offer_id = subo.id
         INNER JOIN [VehicleSales].dbo.proposal subp ON subp.id = subo.proposal_id
         WHERE subp.seller_id = s.id) AS [importe_ventas_totales]
      -- Fecha Ultima Venta
      ,(SELECT MAX(subs.sale_date) FROM [VehicleSales].dbo.sale subs
         INNER JOIN [VehicleSales].dbo.offer subo ON subs.offer_id = subo.id
         INNER JOIN [VehicleSales].dbo.proposal subp ON subp.id = subo.proposal_id WHERE subp.seller_id = s.id) AS [fecha_ultima_venta]
      -- Ventas ultimo mes
      ,(SELECT COUNT(subs.id) FROM [VehicleSales].dbo.sale subs
```

```
        INNER JOIN [VehicleSales].dbo.offer subo ON subs.offer_id = subo.id
        INNER JOIN [VehicleSales].dbo.proposal subp ON subp.id = subo.proposal_id
        WHERE MONTH(GETDATE()) = MONTH(subs.sale_date) AND YEAR(GETDATE()) = YEAR(subs.sale_date)
           AND subp.seller_id = s.id) AS [ventas_ultimo_mes]
     -- Promedio ventas mensuales
     ,(SELECT AVG(sub.ventas) FROM
           (
           SELECT subp.seller_id, MONTH(subs.sale_date) AS mes, COUNT(subs.id) AS ventas
           FROM [VehicleSales].dbo.sale subs
              INNER JOIN [VehicleSales].dbo.offer subo ON subs.offer_id = subo.id
              INNER JOIN [VehicleSales].dbo.proposal subp ON subp.id = subo.proposal_id
           GROUP BY subp.seller_id, MONTH(subs.sale_date)
           ) AS sub
        WHERE sub.seller_id = s.id) AS [promedio_ventas_mensuales]
     -- Promedio importe ventas
     ,(SELECT AVG(subo.offer_amount) FROM [VehicleSales].dbo.sale subs
        INNER JOIN [VehicleSales].dbo.offer subo ON subs.offer_id = subo.id
        INNER JOIN [VehicleSales].dbo.proposal subp ON subp.id = subo.proposal_id
        WHERE subp.seller_id = s.id) AS [promedio_importe_ventas]

     -- Ofertas totales (activas)
     ,(SELECT COUNT(subo.id) FROM [VehicleSales].dbo.offer subo
        INNER JOIN [VehicleSales].dbo.proposal subp ON subp.id = subo.proposal_id
        WHERE subp.seller_id = s.id AND subo.active = 1) AS [ofertas_totales]
     -- Fecha ultima Oferta
     ,(SELECT MAX(subo.offer_date) FROM [VehicleSales].dbo.offer subo
        INNER JOIN [VehicleSales].dbo.proposal subp ON subp.id = subo.proposal_id WHERE subp.seller_id = s.id) AS [fecha_ultima_oferta]

     -- Propuestas totales
     ,(SELECT COUNT(subp.id) FROM [VehicleSales].dbo.proposal subp WHERE subp.seller_id = s.id) AS [propuestas_totales]
     -- Fecha ultima propuesta
     ,(SELECT MAX(subp.proposal_date) FROM [VehicleSales].dbo.proposal subp WHERE subp.seller_id = s.id) AS [fecha_ultima_propuesta]

  FROM [VehicleSales].dbo.seller s
     -- Cruce con concesionarios
     INNER JOIN [VehicleSales].dbo.concessionaire c ON c.id = s.concessionaire_id
  WHERE
     -- El vendedor siga trabajando en la compañía
     s.active = 1
     -- El concesionario siga activo
     AND c.active = 1
  GROUP BY c.[name], s.id, s.[name]

  -- Borramos tabla ultima ejecución
  DROP TABLE IF EXISTS [output].BMWGroup_TopSellers

  -- Formato final y creación de tabla final
  SELECT
     -- 10 mejores vendedores
     TOP 10
     -- Incluyo columna Top ordenando por ventas totales
     CONVERT(INT, ROW_NUMBER() OVER(ORDER BY [ventas_totales] DESC)) AS [top]
     ,*
     ,GETDATE() AS creation_date
  INTO [output].BMWGroup_TopSellers
  FROM @report_table

  -- Salida desde tabla final con cabeceras
  SELECT
     'Top','Concesionario','Vendedor'
     ,'Ventas_totales','Importe_ventas_totales','Fecha_ultima_venta'
     ,'Ventas_ultimo_mes','Promedio_ventas_mensuales','Promedio_importe_ventas'
     --
     ,'Ofertas_totales','Fecha_ultima_oferta'
     --
     ,'Propuestas_totales','Fecha_ultima_propuesta'
  UNION ALL
  SELECT
     CONVERT(NVARCHAR(16), [top]), [auxiliar].[f_clean_rare_characters]([concesionario]), [auxiliar].[f_clean_rare_characters]([vendedor])
     ,CONVERT(NVARCHAR(16), [ventas_totales]), CONVERT(NVARCHAR(16), [importe_ventas_totales]), CONVERT(NVARCHAR(16), [fecha_ultima_venta])
     ,CONVERT(NVARCHAR(16), [ventas_ultimo_mes]), CONVERT(NVARCHAR(16), [promedio_ventas_mensuales]), CONVERT(NVARCHAR(16), [promedio_importe_ventas])
     ,CONVERT(NVARCHAR(16), [ofertas_totales]), CONVERT(NVARCHAR(16), [fecha_ultima_oferta])
     ,CONVERT(NVARCHAR(16), [propuestas_totales]), CONVERT(NVARCHAR(16), [fecha_ultima_propuesta])
  FROM [output].BMWGroup_TopSellers

END
```

A continuación el *Código C.49-C.61* muestra los procedimientos SQL para el caso de uso 4 - Input - RemoteConnections.

Código C.46: input.sp_RemoteConnections_ingestion_VehicleSales_sale.sql

```
USE [IO];
GO
/*
-- ================================================================================
Autor: Adrian Maroto
Fecha creacion: 20/03/2024
Descripcion: Proceso de ingesta para copiar la tabla VehicleSales.dbo.sale en la BBDD IO.
Cambios:

Entrada:
   @destination_table = Tabla sobre la que vamos a copiar la informacion.
   @remote_server = Servidor donde se aloja la BBDD remota.
```

```
   @remote_database = Nombre de la BBDD remota.
   @remote_user = Nombre de usuario para conectarnos al servidor remoto.
   @remote_pass = Contraseña de usuario para conectarnos al servidor remoto.

Salida:

Ejemplo:
   EXEC [IO].[input].[sp_RemoteConnections_ingestion_VehicleSales_sale]
      @destination_table = '[IO].auxiliar.sale'
      , @remote_server = 'WIN-SER-AMAROTO'
      , @remote_database = 'VehicleSales'
      , @remote_user = 'etlautomation'
      , @remote_pass = 'etlautomation'

   SELECT * FROM VehicleSales.dbo.sale
   SELECT * FROM [IO].auxiliar.sale
-- ============================================================================================
*/
CREATE OR ALTER PROCEDURE [input].[sp_RemoteConnections_ingestion_VehicleSales_sale]
   @destination_table NVARCHAR(1024),
   @remote_server NVARCHAR(64),
   @remote_database NVARCHAR(64),
   @remote_user NVARCHAR(64),
   @remote_pass NVARCHAR(64)
AS
BEGIN

   DECLARE @query VARCHAR(MAX)

   -- Eliminamos la tabla si existe previamente
   DROP TABLE IF EXISTS [IO].auxiliar.sale

   -- Usamos la funcionalidad OPENROWSET para el acceso remoto
   SET @query =
      'SELECT
         id, CONVERT(INT, NULL) AS delivery_id, offer_id, sale_date, io_id
      INTO ' + @destination_table + '
      FROM OPENROWSET(
         ''SQLOLEDB'',
         ''Server=' + @remote_server + ';uid=' + @remote_user + ';pwd=' + @remote_pass +';database=' + @remote_database + ''',
         ''SET NOCOUNT ON
         SELECT
            id, offer_id, sale_date, io_id
         FROM dbo.sale s;''
      ) vehiclesales'

   --PRINT (@query)
   EXEC (@query)

END
```

Código C.47: input.sp_RemoteConnections_new_delivery.sql

```
USE [IO];
GO
/*
-- ============================================================================================
Autor: Adrian Maroto
Fecha creacion: 20/03/2024
Descripcion: Crea una nueva delivery para el proceso RemoteConnections.
Cambios:

Entrada:
   @master_id: Id configuración maestra del proceso.
   @master_log_id: Id log de la ejecución en curso.

Salida:

Ejemplo:
   EXEC [IO].[monitor].[create_or_update_master_process_log] @p_in_process_log_id = 1, @p_step = 'Beginning'
   EXEC [IO].[input].[sp_RemoteConnections_new_delivery] @master_id = 4, @master_log_id = 1

   SELECT * FROM [IO].monitor.master_process_log ORDER BY id DESC
   SELECT * FROM [IO].[load].delivery ORDER BY id DESC
   SELECT * FROM [IO].config.customer_cluster_type
-- ============================================================================================
*/
CREATE OR ALTER PROCEDURE [input].[sp_RemoteConnections_new_delivery]
   @master_id INT,
   @master_log_id INT
AS
BEGIN
   -- Variables de logging en caso de error
   DECLARE @msg VARCHAR(1024)
   DECLARE @error_values VARCHAR(1024)
   DECLARE @delivery_id INT

   -- Eliminación de la tabla temporal en caso de que exista
   DROP TABLE IF EXISTS #incorporation_temp

   -- Se calcula el/los cluster_id basándose en los names que contiene la tabla del archivo ingestado
   SELECT DISTINCT COALESCE(cct.cluster_id, 0) AS customer_id
   INTO #incorporation_temp
   FROM [IO].config.customer_cluster_type cct
   WHERE cluster_name = 'BMW_Group'

   -- Se verifica la existencia de una delivery abierta
```

```
   IF EXISTS (
      SELECT * FROM #incorporation_temp t
      INNER JOIN [IO].[load].delivery d ON d.customer_id = t.customer_id
      INNER JOIN [IO].monitor.master_process_log mpl ON d.master_process_log_id = mpl.id
      INNER JOIN [IO].config.master_process_type mpt ON mpl.master_process_type_id = mpt.id
      WHERE mpt.id = @master_id AND d.registration_date IS NULL
   )
      -- En caso afirmativo, se genera el error adecuado sobre el master process id asociado
      BEGIN
         SET @msg = 'There is some pending delivery associated to main process ' + CONVERT(VARCHAR(10), @master_id)
         EXEC [IO].[monitor].[sp_write_log] @ProcId = @@PROCID, @TraceId = '', @Message = @msg , @Severity = 'ERROR'
      END
   ELSE
      -- En caso negativo, se generan las deliveries necesarias según los clientes/clusters
      BEGIN
         INSERT INTO [IO].[load].delivery (customer_id, master_process_log_id)
         SELECT customer_id, @master_log_id FROM #incorporation_temp

         -- Almacenamos el id recien insertado para usarlo en los siguientes pasos
         SELECT @delivery_id = MAX(id) FROM [IO].[load].delivery d WHERE d.master_process_log_id = @master_log_id AND d.registration_date IS NULL

         -- Actualizamos la tabla auxiliar con la delivery
         UPDATE [IO].auxiliar.sale SET delivery_id = @delivery_id

         -- Se insertan las operaciones INSERT de los registros de la tabla auxiliar
         INSERT INTO [IO].[load].operation
            ([delivery_id], [load_table_name], [load_reference_id], [master_reference_id], [operation_type_id])
         SELECT DISTINCT
            d.id AS delivery_id,
            'sale' AS load_table_name,
            s.id AS load_reference_id,
            NULL AS master_reference_id,
            (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Insert') AS operation_type_id
         FROM [IO].auxiliar.sale s
            INNER JOIN [IO].[load].delivery d ON d.id = @delivery_id
            INNER JOIN [IO].monitor.master_process_log mpl ON d.master_process_log_id = mpl.id
         WHERE
            -- Pertenece al proceso maestro adecuado
            mpl.master_process_type_id = @master_id
            -- Tiene delivery abierta
            AND d.registration_date IS NULL
      END

END
```

Código C.48: input.sp_RemoteConnections_persistence_insert_VehicleAlt_sale.sql

```
USE [IO];
GO
/*
-- ================================================================================================
Autor: Adrian Maroto
Fecha creacion: 20/03/2024
Descripcion: Proceso de persistencia para insertar las ventas en el destino remoto VehicleAlt.dbo.sale.
Cambios:

Entrada:
   @delivery_id: Id de la delivery abierta en curso.
   @remote_server = Servidor donde se aloja la BBDD remota.
   @remote_database = Nombre de la BBDD remota.
   @remote_user = Nombre de usuario para conectarnos al servidor remoto.
   @remote_pass = Contraseña de usuario para conectarnos al servidor remoto.

Salida:
   @affected_rows: Número de filas afectadas por el proceso.

Ejemplo:
   EXEC [IO].[input].[sp_RemoteConnections_persistence_insert_VehicleAlt_sale]
      @delivery_id = 1
      , @remote_server = 'WIN-SER-AMAROTO'
      , @remote_database = 'VehicleAlt'
      , @remote_user = 'etlautomation'
      , @remote_pass = 'etlautomation'
      , @affected_rows = 0

   SELECT * FROM VehicleSales.dbo.sale
   SELECT * FROM [IO].auxiliar.sale
-- ================================================================================================
*/
CREATE OR ALTER PROCEDURE [input].[sp_RemoteConnections_persistence_insert_VehicleAlt_sale]
   @delivery_id INT,
   @remote_server VARCHAR(64),
   @remote_database VARCHAR(64),
   @remote_user VARCHAR(64),
   @remote_pass VARCHAR(64),
   @affected_rows INT OUTPUT
AS
BEGIN

   DECLARE @query VARCHAR(MAX)

   SET DATEFORMAT dmy

   SET @query =
      'INSERT INTO
         OPENROWSET(
            ''SQLOLEDB'',
```

```
            ''Server=' + @remote_server + ';uid=' + @remote_user + ';pwd=' + @remote_pass +';database=' + @remote_database + ''',
            ''SELECT
               offer_id, sale_date, io_id
            FROM dbo.sale;''
         )
      SELECT
         s.offer_id
         ,s.sale_date
         ,s.id AS io_id
      FROM [IO].[auxiliar].sale s
         INNER JOIN [IO].[load].operation o ON o.delivery_id = s.delivery_id
            AND o.load_reference_id = s.id
      WHERE
         -- Tabla correcta
         o.load_table_name = ''sale''
         -- Operación de tipo Insert
         AND o.operation_type_id = (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = ''Insert'')
         -- Operación sin persistir
         AND o.operation_finished = 0
         -- Para la delivery en curso
         AND o.delivery_id = ' + CAST(@delivery_id AS VARCHAR(10))

   --PRINT (@query)
   EXEC (@query)

   -- Se finalizan (operation_finished = 1) las operaciones realizadas en el paso anterior
   UPDATE o SET operation_finished = 1
   FROM [IO].[auxiliar].sale s
      INNER JOIN [IO].[load].operation o ON o.delivery_id = s.delivery_id
         AND o.load_reference_id = s.id
   WHERE
      -- Tabla correcta
      o.load_table_name = 'sale'
      -- Operación de tipo Insert
      AND o.operation_type_id = (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Insert')
      -- Operación sin persistir
      AND o.operation_finished = 0
      -- Para la delivery en curso
      AND o.delivery_id = @delivery_id

   -- Se devuelve el número de operaciones almacenadas
   SELECT @affected_rows = @@ROWCOUNT

END
```

A continuación se muestran los procedimientos SQL para el caso de uso 5 - Input - StatusUpdate.

Código C.49: input.sp_BMWGroup_StatusUpdate_new_delivery.sql

```
USE [IO];
GO
/*
-- ================================================================================
Autor: Adrian Maroto
Fecha creacion: 22/03/2024
Descripcion: Crea una nueva delivery para el proceso StatusUpdate.
Cambios:

Entrada:
   @master_id: Id configuración maestra del proceso.
   @master_log_id: Id log de la ejecución en curso.

Salida:

Ejemplo:
   EXEC [IO].[monitor].[create_or_update_master_process_log] @p_in_process_log_id = 1, @p_step = 'Beginning'
   EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_new_delivery] @master_id = 1, @master_log_id = 1

   SELECT * FROM [IO].monitor.master_process_log ORDER BY id DESC
   SELECT * FROM [IO].[load].delivery ORDER BY id DESC
   SELECT * FROM [IO].config.customer_cluster_type
-- ================================================================================
*/
CREATE OR ALTER PROCEDURE [input].[sp_BMWGroup_StatusUpdate_new_delivery]
   @master_id INT,
   @master_log_id INT
AS
BEGIN
   -- Variables de logging en caso de error
   DECLARE @msg VARCHAR(1024)
   DECLARE @error_values VARCHAR(1024)

   -- Eliminación de la tabla temporal en caso de que exista
   DROP TABLE IF EXISTS #incorporation_temp

   -- Se calcula el/los cluster_id basándose en los names que contiene la tabla del archivo ingestado
   SELECT DISTINCT COALESCE(cct.cluster_id, 0) AS customer_id
   INTO #incorporation_temp
   FROM [IO].config.customer_cluster_type cct
   WHERE cluster_name = 'BMW_Group'

   -- Se verifica la existencia de una delivery abierta
   IF EXISTS (
```

```
        SELECT * FROM #incorporation_temp tt
        INNER JOIN [IO].[load].delivery dv ON dv.customer_id = tt.customer_id
        INNER JOIN [IO].monitor.master_process_log mpl ON dv.master_process_log_id = mpl.id
        INNER JOIN [IO].config.master_process_type mpt ON mpl.master_process_type_id = mpt.id
        WHERE mpt.id = @master_id AND dv.registration_date IS NULL
    )
        -- En caso afirmativo, se genera el error adecuado sobre el master process id asociado
        BEGIN
            SET @msg = 'There is some pending delivery associated to main process ' + CONVERT(VARCHAR(10), @master_id)
            EXEC [IO].[monitor].[sp_write_log] @ProcId = @@PROCID, @TraceId = '', @Message = @msg , @Severity = 'ERROR'
        END
    ELSE
        -- En caso negativo, se generan las deliveries necesarias según los clientes/clusters
        BEGIN
            INSERT INTO [IO].[load].delivery (customer_id, master_process_log_id)
            SELECT customer_id, @master_log_id FROM #incorporation_temp
        END
END
```

Código C.50: input.sp_BMWGroup_StatusUpdate_transformation_insert_model_type.sql

```
USE [IO];
GO
/*
-- ============================================================================================
Autor: Adrian Maroto
Fecha creacion: 25/03/2024
Descripcion: Proceso de inserción de modelos del fichero StatusUpdate en [IO].[load].model_type.
Cambios:

Entrada:
    @master_process_id: Id configuración maestra del proceso.

Salida:

Ejemplo:
    EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_transformation_insert_model_type] @master_process_id = 1

    SELECT * FROM [IO].[load].delivery ORDER BY id DESC
    SELECT * FROM [IO].[load].operation
    SELECT * FROM [IO].[load].operation_type

    SELECT * FROM [IO].[input].[BMWGroup_StatusUpdate]
    SELECT * FROM [IO].[auxiliar].[BMWGroup_model_type_mapping]
    SELECT * FROM [IO].[load].[model_type]
    SELECT * FROM VehicleSales.dbo.model_type
-- ============================================================================================
*/
CREATE OR ALTER PROCEDURE [input].[sp_BMWGroup_StatusUpdate_transformation_insert_model_type]
    @master_process_id INT
AS
BEGIN

    -- Se insertan en la tabla model_type del schema [load] los nuevos modelos provenientes del archivo
    -- que no existan ya en la tabla
    INSERT INTO [IO].[load].[model_type]
        ([delivery_id], [mdb_id], [name], [brand_type_id], [fuel_type_id], [active])
    SELECT
        delivery_id,
        mdb_id,
        [name],
        brand_type_id,
        NULL,
        active
    FROM (
        -- Se obtienen todos los datos de los modelos nuevos
        SELECT DISTINCT
            d.id AS delivery_id,
            NULL AS mdb_id,
            mtm.model_type_table AS [name],
            CASE
                WHEN mtm.brand_type = 'BMW' THEN 1
                WHEN mtm.brand_type = 'Mini' THEN 1
            END AS brand_type_id,
            1 AS active
        FROM [IO].[input].[BMWGroup_StatusUpdate] su
            -- Cruza contra la tabla de mapeo para distinguir casos nuevos de modelos mal escritos
            INNER JOIN [IO].auxiliar.BMWGroup_model_type_mapping mtm ON [auxiliar].[f_clean_rare_characters](su.modelo) =
        [auxiliar].[f_clean_rare_characters](mtm.model_type_file)
            LEFT JOIN VehicleSales.dbo.model_type mt ON mt.[name] = mtm.model_type_table
            INNER JOIN [IO].config.customer_cluster_type cct ON 1 = cct.customer_id
            INNER JOIN [IO].[load].delivery d ON d.customer_id = cct.cluster_id
            INNER JOIN [IO].monitor.master_process_log mpl ON d.master_process_log_id = mpl.id
        WHERE
            -- Modelos nuevos
            mt.id IS NULL
            -- Pertenece al proceso maestro adecuado
            AND mpl.master_process_type_id = @master_process_id
            -- Tiene delivery abierta
            AND d.registration_date IS NULL
    ) sub

    -- Se insertan todas las operaciones INSERT de los registros calculados anteriormente
    INSERT INTO [IO].[load].operation
        ([delivery_id], [load_table_name], [load_reference_id], [master_reference_id], [operation_type_id])
    SELECT DISTINCT
        d.id AS delivery_id,
```

```
        'model_type' AS load_table_name,
        mt.id AS load_reference_id,
        mt.mdb_id AS master_reference_id,
        (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Insert') AS operation_type_id
    FROM [IO].[load].[model_type] mt
        INNER JOIN [IO].[load].delivery d ON mt.delivery_id= d.id
        INNER JOIN [IO].monitor.master_process_log mpl ON d.master_process_log_id = mpl.id
    WHERE
        -- Son modelos nuevos
        mt.mdb_id IS NULL
        -- Pertenece al proceso maestro adecuado
        AND mpl.master_process_type_id = @master_process_id
        -- Tiene delivery abierta
        AND d.registration_date IS NULL

    -- Se devuelve el número de operaciones almacenadas
    SELECT @@ROWCOUNT AS operations_quantity

END
```

Código C.51: input.sp_BMWGroup_StatusUpdate_transformation_insert_seller.sql

```
USE [IO];
GO
/*
-- ============================================================================================
Autor: Adrian Maroto
Fecha creacion: 25/03/2024
Descripcion: Proceso de inserción de vendedores del fichero StatusUpdate en [IO].[load].seller.
Cambios:

Entrada:
    @master_process_id: Id configuración maestra del proceso.

Salida:

Ejemplo:
    EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_transformation_insert_seller] @master_process_id = 1

    SELECT * FROM [IO].[load].delivery ORDER BY id DESC
    SELECT * FROM [IO].[load].operation
    SELECT * FROM [IO].[load].operation_type

    SELECT * FROM [IO].[input].[BMWGroup_StatusUpdate]
    SELECT * FROM [IO].[load].[seller]
    SELECT * FROM VehicleSales.dbo.seller
-- ============================================================================================
*/
CREATE OR ALTER PROCEDURE [input].[sp_BMWGroup_StatusUpdate_transformation_insert_seller]
    @master_process_id INT
AS
BEGIN

    -- Se insertan en la tabla seller del schema [load] los nuevos vendedores provenientes del archivo
    -- que no existan ya en la tabla
    INSERT INTO [IO].[load].[seller]
        ([delivery_id], [mdb_id], [name], [phone], [email], [concessionaire_id], [active])
    SELECT
        delivery_id,
        mdb_id,
        [name],
        phone,
        email,
        concessionaire_id,
        active
    FROM (
        -- Se obtienen todos los datos de los vendedores nuevos
        SELECT DISTINCT
            d.id AS delivery_id,
            NULL AS mdb_id,
            su.vendedor AS [name],
            -- Comprueba que el telefono sea valido
            IIF(input.f_check_phone(su.tel_vendedor) = 1, su.tel_vendedor, NULL) AS phone,
            -- Comprueba que el correo sea valido
            IIF(input.f_check_email(su.email_vendedor) = 1, su.email_vendedor, NULL) AS email,
            c.id AS concessionaire_id,
            1 AS active
        FROM [IO].[input].[BMWGroup_StatusUpdate] su
            LEFT JOIN VehicleSales.dbo.seller s ON [auxiliar].[f_clean_rare_characters](s.[name]) = [auxiliar].[f_clean_rare_characters](su.vendedor)
            LEFT JOIN VehicleSales.dbo.concessionaire c ON [auxiliar].[f_clean_rare_characters](c.[name]) =
        [auxiliar].[f_clean_rare_characters](su.concesionario)
            INNER JOIN [IO].config.customer_cluster_type cct ON 1 = cct.customer_id
            INNER JOIN [IO].[load].delivery d ON d.customer_id = cct.cluster_id
            INNER JOIN [IO].monitor.master_process_log mpl ON d.master_process_log_id = mpl.id
        WHERE
            -- Vendedores nuevos
            s.id IS NULL
            -- Pertenece al proceso maestro adecuado
            AND mpl.master_process_type_id = @master_process_id
            -- Tiene delivery abierta
            AND d.registration_date IS NULL
    ) sub

    -- Se insertan todas las operaciones INSERT de los registros calculados anteriormente
    INSERT INTO [IO].[load].operation
        ([delivery_id], [load_table_name], [load_reference_id], [master_reference_id], [operation_type_id])
    SELECT DISTINCT
```

```
        d.id AS delivery_id,
        'seller' AS load_table_name,
        s.id AS load_reference_id,
        s.mdb_id AS master_reference_id,
        (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Insert') AS operation_type_id
    FROM [IO].[load].seller s
        INNER JOIN [IO].[load].delivery d ON s.delivery_id= d.id
        INNER JOIN [IO].monitor.master_process_log mpl ON d.master_process_log_id = mpl.id
    WHERE
        -- Son vendedores nuevos
        s.mdb_id IS NULL
        -- Pertenece al proceso maestro adecuado
        AND mpl.master_process_type_id = @master_process_id
        -- Tiene delivery abierta
        AND d.registration_date IS NULL

    -- Se devuelve el número de operaciones almacenadas
    SELECT @@ROWCOUNT AS operations_quantity

END
```

Código C.52: input.sp_BMWGroup_StatusUpdate_transformation_insert_customer.sql

```
USE [IO];
GO
/*
-- ================================================================================
Autor: Adrian Maroto
Fecha creacion: 25/03/2024
Descripcion: Proceso de inserción de clientes del fichero StatusUpdate en [IO].[load].customer.
Cambios:

Entrada:
    @master_process_id: Id configuración maestra del proceso.

Salida:

Ejemplo:
    EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_transformation_insert_customer] @master_process_id = 1

    SELECT * FROM [IO].[load].delivery ORDER BY id DESC
    SELECT * FROM [IO].[load].operation
    SELECT * FROM [IO].[load].operation_type

    SELECT * FROM [IO].[input].[BMWGroup_StatusUpdate]
    SELECT * FROM [IO].[load].[customer]
    SELECT * FROM VehicleSales.dbo.customer
-- ================================================================================
*/
CREATE OR ALTER PROCEDURE [input].[sp_BMWGroup_StatusUpdate_transformation_insert_customer]
    @master_process_id INT
AS
BEGIN

    -- Se insertan en la tabla customer del schema [load] los nuevos clientes provenientes del archivo
    -- que no existan ya en la tabla
    INSERT INTO [IO].[load].[customer]
        ([delivery_id], [mdb_id], [name], [phone], [email])
    SELECT
        delivery_id,
        mdb_id,
        [name],
        phone,
        email
    FROM (
        -- Se obtienen todos los datos de los clientes nuevos
        SELECT DISTINCT
            d.id AS delivery_id,
            NULL AS mdb_id,
            su.cliente AS [name],
            -- Comprueba que el telefono sea valido
            IIF(input.f_check_phone(su.tel_cliente) = 1, su.tel_cliente, NULL) AS phone,
            -- Comprueba que el correo sea valido
            IIF(input.f_check_email(su.email_cliente) = 1, su.email_cliente, NULL) AS email
        FROM [IO].[input].[BMWGroup_StatusUpdate] su
            LEFT JOIN VehicleSales.dbo.customer s ON [auxiliar].[f_clean_rare_characters](s.[name]) = [auxiliar].[f_clean_rare_characters](su.cliente)
            INNER JOIN [IO].config.customer_cluster_type cct ON 1 = cct.customer_id
            INNER JOIN [IO].[load].delivery d ON d.customer_id = cct.cluster_id
            INNER JOIN [IO].monitor.master_process_log mpl ON d.master_process_log_id = mpl.id
        WHERE
            -- Clientes nuevos
            s.id IS NULL
            -- Pertenece al proceso maestro adecuado
            AND mpl.master_process_type_id = @master_process_id
            -- Tiene delivery abierta
            AND d.registration_date IS NULL
    ) sub

    -- Se insertan todas las operaciones INSERT de los registros calculados anteriormente
    INSERT INTO [IO].[load].operation
        ([delivery_id], [load_table_name], [load_reference_id], [master_reference_id], [operation_type_id])
    SELECT DISTINCT
        d.id AS delivery_id,
        'customer' AS load_table_name,
        c.id AS load_reference_id,
        c.mdb_id AS master_reference_id,
        (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Insert') AS operation_type_id
```

```
   FROM [IO].[load].customer c
      INNER JOIN [IO].[load].delivery d ON c.delivery_id= d.id
      INNER JOIN [IO].monitor.master_process_log mpl ON d.master_process_log_id = mpl.id
   WHERE
      -- Son clientes nuevos
      c.mdb_id IS NULL
      -- Pertenece al proceso maestro adecuado
      AND mpl.master_process_type_id = @master_process_id
      -- Tiene delivery abierta
      AND d.registration_date IS NULL

   -- Se devuelve el número de operaciones almacenadas
   SELECT @@ROWCOUNT AS operations_quantity

END
```

Código C.53: input.sp_BMWGroup_StatusUpdate_transformation_insert_proposal.sql

```
USE [IO];
GO
/*
-- ================================================================================
Autor: Adrian Maroto
Fecha creacion: 25/03/2024
Descripcion: Proceso de inserción de propuestas del fichero StatusUpdate en [IO].[load].proposal.
Cambios:

Entrada:
   @master_process_id: Id configuración maestra del proceso.

Salida:

Ejemplo:
   EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_transformation_insert_proposal] @master_process_id = 1

   SELECT * FROM [IO].[load].delivery ORDER BY id DESC
   SELECT * FROM [IO].[load].operation
   SELECT * FROM [IO].[load].operation_type

   SELECT * FROM [IO].[input].[BMWGroup_StatusUpdate]
   SELECT * FROM [IO].[load].[proposal]
   SELECT * FROM VehicleSales.dbo.proposal
-- ================================================================================
*/
CREATE OR ALTER PROCEDURE [input].[sp_BMWGroup_StatusUpdate_transformation_insert_proposal]
   @master_process_id INT
AS
BEGIN

   -- Se insertan en la tabla proposal del schema [load] las nuevas propuestas provenientes del archivo
   -- que no existan ya en la tabla
   INSERT INTO [IO].[load].[proposal]
      ([delivery_id], [mdb_id], [cliente], [vendedor], [modelo],[proposal_date], [proposal_amount], [proposal_type_id], [active])
   SELECT
      delivery_id,
      mdb_id,
      cliente,
      vendedor,
      modelo,
      proposal_date,
      proposal_amount,
      proposal_type_id,
      active
   FROM (
      -- Se obtienen todos los datos de las propuestas nuevas
      SELECT
         d.id AS delivery_id,
         NULL AS mdb_id,
         su.cliente,
         su.vendedor,
         mtm.model_type_table AS modelo,
         su.fecha_propuesta AS proposal_date,
         su.cantidad_propuesta AS proposal_amount,
         pt.id AS proposal_type_id,
         1 AS active
      FROM [IO].[input].[BMWGroup_StatusUpdate] su
         LEFT JOIN VehicleSales.dbo.proposal_type pt ON pt.[name] = su.tipo_propuesta
         LEFT JOIN VehicleSales.dbo.proposal p ON p.proposal_date = su.fecha_propuesta
            AND p.proposal_amount = su.cantidad_propuesta AND pt.id = p.proposal_type_id
         -- Cruza contra la tabla de mapeo para distinguir casos nuevos de modelos mal escritos
         INNER JOIN [IO].auxiliar.BMWGroup_model_type_mapping mtm ON [auxiliar].[f_clean_rare_characters](su.modelo) =
      [auxiliar].[f_clean_rare_characters](mtm.model_type_file)
         -- Info delivery
         INNER JOIN [IO].config.customer_cluster_type cct ON 1 = cct.customer_id
         INNER JOIN [IO].[load].delivery d ON d.customer_id = cct.cluster_id
         INNER JOIN [IO].monitor.master_process_log mpl ON d.master_process_log_id = mpl.id
      WHERE
         -- Propuestas nuevas
         p.id IS NULL
         -- Pertenece al proceso maestro adecuado
         AND mpl.master_process_type_id = @master_process_id
         -- Tiene delivery abierta
         AND d.registration_date IS NULL
   ) sub

   -- Se insertan todas las operaciones INSERT de los registros calculados anteriormente
   INSERT INTO [IO].[load].operation
```

```
        ([delivery_id], [load_table_name], [load_reference_id], [master_reference_id], [operation_type_id])
    SELECT DISTINCT
        d.id AS delivery_id,
        'proposal' AS load_table_name,
        p.id AS load_reference_id,
        p.mdb_id AS master_reference_id,
        (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Insert') AS operation_type_id
    FROM [IO].[load].proposal p
        INNER JOIN [IO].[load].delivery d ON p.delivery_id= d.id
        INNER JOIN [IO].monitor.master_process_log mpl ON d.master_process_log_id = mpl.id
    WHERE
        -- Son propuestas nuevas
        p.mdb_id IS NULL
        -- Pertenece al proceso maestro adecuado
        AND mpl.master_process_type_id = @master_process_id
        -- Tiene delivery abierta
        AND d.registration_date IS NULL

    -- Se devuelve el número de operaciones almacenadas
    SELECT @@ROWCOUNT AS operations_quantity

END
```

Código C.54: input.sp_BMWGroup_StatusUpdate_transformation_insert_offer.sql

```
USE [IO];
GO
/*
-- ================================================================================================
Autor: Adrian Maroto
Fecha creacion: 25/03/2024
Descripcion: Proceso de inserción de ofertas del fichero StatusUpdate en [IO].[load].offer.
Cambios:

Entrada:
    @master_process_id: Id configuración maestra del proceso.

Salida:

Ejemplo:
    EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_transformation_insert_offer] @master_process_id = 1

    SELECT * FROM [IO].[load].delivery ORDER BY id DESC
    SELECT * FROM [IO].[load].operation
    SELECT * FROM [IO].[load].operation_type

    SELECT * FROM [IO].[input].[BMWGroup_StatusUpdate]
    SELECT * FROM [IO].[load].[proposal]
    SELECT * FROM [IO].[load].[offer]
    SELECT * FROM VehicleSales.dbo.offer
-- ================================================================================================
*/
CREATE OR ALTER PROCEDURE [input].[sp_BMWGroup_StatusUpdate_transformation_insert_offer]
    @master_process_id INT
AS
BEGIN

    -- Se insertan en la tabla offer del schema [load] las nuevas ofertas provenientes del archivo
    -- que no existan ya en la tabla
    INSERT INTO [IO].[load].[offer]
        ([delivery_id], [mdb_id], [proposal_io_id], [offer_date], [offer_amount], [active])
    SELECT
        delivery_id,
        mdb_id,
        proposal_io_id,
        offer_date,
        offer_amount,
        active
    FROM (
        -- Se obtienen todos los datos de las ofertas nuevas
        SELECT DISTINCT
            d.id AS delivery_id,
            NULL AS mdb_id,
            p.id AS proposal_io_id,
            su.fecha_oferta AS offer_date,
            su.cantidad_oferta AS offer_amount,
            1 AS active
        FROM [IO].[input].[BMWGroup_StatusUpdate] su
            LEFT JOIN VehicleSales.dbo.offer o ON o.offer_date = su.fecha_oferta
                AND o.offer_amount = su.cantidad_oferta
            -- Cruzamos contra load.proposal para obtener el id insertado en load.proposal
            LEFT JOIN [IO].[load].proposal p ON su.cliente = p.cliente AND su.fecha_propuesta = p.proposal_date AND su.cantidad_propuesta =
        p.proposal_amount
            -- Info delivery
            INNER JOIN [IO].config.customer_cluster_type cct ON 1 = cct.customer_id
            INNER JOIN [IO].[load].delivery d ON d.customer_id = cct.cluster_id
            INNER JOIN [IO].monitor.master_process_log mpl ON d.master_process_log_id = mpl.id
        WHERE
            (su.fecha_oferta IS NOT NULL OR su.cantidad_oferta IS NOT NULL)
            -- Ofertas nuevas
            AND o.id IS NULL
            -- Pertenece al proceso maestro adecuado
            AND mpl.master_process_type_id = @master_process_id
            -- Tiene delivery abierta
            AND d.registration_date IS NULL
    ) sub

```

```
   -- Se insertan todas las operaciones INSERT de los registros calculados anteriormente
   INSERT INTO [IO].[load].operation
      ([delivery_id], [load_table_name], [load_reference_id], [master_reference_id], [operation_type_id])
   SELECT DISTINCT
      d.id AS delivery_id,
      'offer' AS load_table_name,
      o.id AS load_reference_id,
      o.mdb_id AS master_reference_id,
      (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Insert') AS operation_type_id
   FROM [IO].[load].offer o
      INNER JOIN [IO].[load].delivery d ON o.delivery_id= d.id
      INNER JOIN [IO].monitor.master_process_log mpl ON d.master_process_log_id = mpl.id
   WHERE
      -- Son ofertas nuevas
      o.mdb_id IS NULL
      -- Pertenece al proceso maestro adecuado
      AND mpl.master_process_type_id = @master_process_id
      -- Tiene delivery abierta
      AND d.registration_date IS NULL

   -- Se devuelve el número de operaciones almacenadas
   SELECT @@ROWCOUNT AS operations_quantity

END
```

Código C.55: input.sp_BMWGroup_StatusUpdate_transformation_insert_sale.sql

```
USE [IO];
GO
/*
-- ============================================================================================
Autor: Adrian Maroto
Fecha creacion: 25/03/2024
Descripcion: Proceso de inserción de ventas del fichero StatusUpdate en [IO].[load].sale.
Cambios:

Entrada:
   @master_process_id: Id configuración maestra del proceso.

Salida:

Ejemplo:
   EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_transformation_insert_sale] @master_process_id = 1

   SELECT * FROM [IO].[load].delivery ORDER BY id DESC
   SELECT * FROM [IO].[load].operation
   SELECT * FROM [IO].[load].operation_type

   SELECT * FROM [IO].[input].[BMWGroup_StatusUpdate]
   SELECT * FROM [IO].[load].[offer]
   SELECT * FROM [IO].[load].[sale]
   SELECT * FROM VehicleSales.dbo.sale
-- ============================================================================================
*/
CREATE OR ALTER PROCEDURE [input].[sp_BMWGroup_StatusUpdate_transformation_insert_sale]
   @master_process_id INT
AS
BEGIN

   -- Se insertan en la tabla sale del schema [load] las nuevas ventas provenientes del archivo
   -- que no existan ya en la tabla
   INSERT INTO [IO].[load].[sale]
      ([delivery_id], [mdb_id], [offer_io_id], [sale_date])
   SELECT
      delivery_id,
      mdb_id,
      offer_io_id,
      sale_date
   FROM (
      -- Se obtienen todos los datos de las ventas nuevas
      SELECT DISTINCT
         d.id AS delivery_id,
         NULL AS mdb_id,
         o.id AS offer_io_id,
         su.fecha_venta AS sale_date
      FROM [IO].[input].[BMWGroup_StatusUpdate] su
         LEFT JOIN VehicleSales.dbo.sale s ON s.sale_date = su.fecha_venta
         -- Cruzamos contra load.proposal para obtener el id insertado en load.proposal
         LEFT JOIN [IO].[load].offer o ON su.fecha_oferta = o.offer_date AND su.cantidad_oferta = o.offer_amount
         -- Info delivery
         INNER JOIN [IO].config.customer_cluster_type cct ON 1 = cct.customer_id
         INNER JOIN [IO].[load].delivery d ON d.customer_id = cct.cluster_id
         INNER JOIN [IO].monitor.master_process_log mpl ON d.master_process_log_id = mpl.id
      WHERE
         -- Fichero contiene ventas
         su.fecha_venta IS NOT NULL
         -- Ventas nuevas
         AND s.id IS NULL
         -- Pertenece al proceso maestro adecuado
         AND mpl.master_process_type_id = @master_process_id
         -- Tiene delivery abierta
         AND d.registration_date IS NULL
   ) sub

   -- Se insertan todas las operaciones INSERT de los registros calculados anteriormente
   INSERT INTO [IO].[load].operation
      ([delivery_id], [load_table_name], [load_reference_id], [master_reference_id], [operation_type_id])
```

```
    SELECT DISTINCT
        d.id AS delivery_id,
        'sale' AS load_table_name,
        s.id AS load_reference_id,
        s.mdb_id AS master_reference_id,
        (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Insert') AS operation_type_id
    FROM [IO].[load].sale s
        INNER JOIN [IO].[load].delivery d ON s.delivery_id= d.id
        INNER JOIN [IO].monitor.master_process_log mpl ON d.master_process_log_id = mpl.id
    WHERE
        -- Son ventas nuevas
        s.mdb_id IS NULL
        -- Pertenece al proceso maestro adecuado
        AND mpl.master_process_type_id = @master_process_id
        -- Tiene delivery abierta
        AND d.registration_date IS NULL

    -- Se devuelve el número de operaciones almacenadas
    SELECT @@ROWCOUNT AS operations_quantity

END
```

Código C.56: input.sp_BMWGroup_StatusUpdate_persistence_insert_model_type.sql

```
USE [IO];
GO
/*
-- ============================================================================================
Autor: Adrian Maroto
Fecha creacion: 25/03/2024
Descripcion: Proceso de inserción de modelos del fichero StatusUpdate en VehicleSales.dbo.model_type.
Cambios:

Entrada:
    @delivery_id: Id de la delivery abierta en curso.

Salida:
    @affected_rows: Número de filas afectadas por el proceso.

Ejemplo:
    EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_persistence_insert_model_type] @delivery_id = 1, @affected_rows = 0

    SELECT * FROM [IO].[load].delivery ORDER BY id DESC
    SELECT * FROM [IO].[load].operation
    SELECT * FROM [IO].[load].operation_type

    SELECT * FROM [IO].[input].[BMWGroup_StatusUpdate]
    SELECT * FROM [IO].[auxiliar].[BMWGroup_model_type_mapping]
    SELECT * FROM [IO].[load].[model_type]
    SELECT * FROM VehicleSales.dbo.model_type
-- ============================================================================================
*/
CREATE OR ALTER PROCEDURE [input].[sp_BMWGroup_StatusUpdate_persistence_insert_model_type]
    @delivery_id INT,
    @affected_rows INT OUTPUT
AS
BEGIN

    -- Se realizan todas las operaciones de INSERT calculadas en el proceso de transformación
    -- [IO].[input].[sp_BMWGroup_StatusUpdate_transformation_insert_model_type]
    INSERT INTO VehicleSales.dbo.model_type
        ([name], [brand_type_id], [fuel_type_id], [active], [io_id])
    SELECT DISTINCT
        mt.[name],
        mt.brand_type_id,
        mt.fuel_type_id,
        mt.active,
        mt.id AS io_id
    FROM [IO].[load].[model_type] mt
        INNER JOIN [IO].[load].operation o ON o.delivery_id = mt.delivery_id
            AND o.load_reference_id = mt.id
    WHERE
        -- Tabla correcta
        o.load_table_name = 'model_type'
        -- Operación de tipo Insert
        AND o.operation_type_id = (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Insert')
        -- Operación sin persistir
        AND o.operation_finished = 0
        -- Para la delivery en curso
        AND o.delivery_id = @delivery_id

    -- Se finalizan (operation_finished = 1) las operaciones realizadas en el paso anterior
    UPDATE o SET operation_finished = 1
    FROM [IO].[load].model_type mt
        INNER JOIN [IO].[load].operation o ON o.delivery_id = mt.delivery_id
            AND o.load_reference_id = mt.id
    WHERE
        -- Tabla correcta
        o.load_table_name = 'model_type'
        -- Operación de tipo Insert
        AND o.operation_type_id = (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Insert')
        -- Operación sin persistir
        AND o.operation_finished = 0
        -- Para la delivery en curso
        AND o.delivery_id = @delivery_id

    -- Se devuelve el número de operaciones actualizadas
```

```
   SELECT @affected_rows = @@ROWCOUNT

END
```

Código C.57: input.sp_BMWGroup_StatusUpdate_persistence_insert_seller.sql

```
USE [IO];
GO
/*
-- ================================================================================
Autor: Adrian Maroto
Fecha creacion: 25/03/2024
Descripcion: Proceso de inserción de vendedores del fichero StatusUpdate en Vehiclesellers.dbo.seller.
Cambios:

Entrada:
   @delivery_id: Id de la delivery abierta en curso.

Salida:
   @affected_rows: Número de filas afectadas por el proceso.

Ejemplo:
   EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_persistence_insert_seller] @delivery_id = 1, @affected_rows = 0

   SELECT * FROM [IO].[load].delivery ORDER BY id DESC
   SELECT * FROM [IO].[load].operation
   SELECT * FROM [IO].[load].operation_type

   SELECT * FROM [IO].[input].[BMWGroup_StatusUpdate]
   SELECT * FROM [IO].[load].[seller]
   SELECT * FROM Vehiclesellers.dbo.seller
-- ================================================================================
*/
CREATE OR ALTER PROCEDURE [input].[sp_BMWGroup_StatusUpdate_persistence_insert_seller]
   @delivery_id INT,
   @affected_rows INT OUTPUT
AS
BEGIN

   -- Se realizan todas las operaciones de INSERT calculadas en el proceso de transformación
   -- [IO].[input].[sp_BMWGroup_StatusUpdate_persistence_insert_seller]
   INSERT INTO VehicleSales.dbo.seller
      ([name], [phone], [email], [concessionaire_id], [active], [io_id])
   SELECT DISTINCT
      s.[name],
      s.phone,
      s.email,
      s.concessionaire_id,
      s.active,
      s.id AS io_id
   FROM [IO].[load].seller s
      INNER JOIN [IO].[load].operation o ON o.delivery_id = s.delivery_id
         AND o.load_reference_id = s.id
   WHERE
      -- Tabla correcta
      o.load_table_name = 'seller'
      -- Operación de tipo Insert
      AND o.operation_type_id = (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Insert')
      -- Operación sin persistir
      AND o.operation_finished = 0
      -- Para la delivery en curso
      AND o.delivery_id = @delivery_id

   -- Se finalizan (operation_finished = 1) las operaciones realizadas en el paso anterior
   UPDATE o SET operation_finished = 1
   FROM [IO].[load].seller s
      INNER JOIN [IO].[load].operation o ON o.delivery_id = s.delivery_id
         AND o.load_reference_id = s.id
   WHERE
      -- Tabla correcta
      o.load_table_name = 'seller'
      -- Operación de tipo Insert
      AND o.operation_type_id = (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Insert')
      -- Operación sin persistir
      AND o.operation_finished = 0
      -- Para la delivery en curso
      AND o.delivery_id = @delivery_id

   -- Se devuelve el número de operaciones actualizadas
   SELECT @affected_rows = @@ROWCOUNT

END
```

Código C.58: input.sp_BMWGroup_StatusUpdate_persistence_insert_customer.sql

```
USE [IO];
GO
/*
-- ================================================================================
Autor: Adrian Maroto
Fecha creacion: 25/03/2024
Descripcion: Proceso de inserción de clientes del fichero StatusUpdate en VehicleSales.dbo.customer.
Cambios:

Entrada:
```

```
   @delivery_id: Id de la delivery abierta en curso.

Salida:
   @affected_rows: Número de filas afectadas por el proceso.

Ejemplo:
   EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_persistence_insert_customer] @delivery_id = 1, @affected_rows = 0

   SELECT * FROM [IO].[load].delivery ORDER BY id DESC
   SELECT * FROM [IO].[load].operation
   SELECT * FROM [IO].[load].operation_type

   SELECT * FROM [IO].[input].[BMWGroup_StatusUpdate]
   SELECT * FROM [IO].[load].[customer]
   SELECT * FROM VehicleSales.dbo.customer
-- ================================================================================
*/
CREATE OR ALTER PROCEDURE [input].[sp_BMWGroup_StatusUpdate_persistence_insert_customer]
   @delivery_id INT,
   @affected_rows INT OUTPUT
AS
BEGIN

   -- Se realizan todas las operaciones de INSERT calculadas en el proceso de transformación
   -- [IO].[input].[sp_BMWGroup_StatusUpdate_persistence_insert_customer]
   INSERT INTO VehicleSales.dbo.customer
      ([name], [phone], [email], [io_id])
   SELECT DISTINCT
      s.[name],
      s.phone,
      s.email,
      s.id AS io_id
   FROM [IO].[load].customer s
      INNER JOIN [IO].[load].operation o ON o.delivery_id = s.delivery_id
         AND o.load_reference_id = s.id
   WHERE
      -- Tabla correcta
      o.load_table_name = 'customer'
      -- Operación de tipo Insert
      AND o.operation_type_id = (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Insert')
      -- Operación sin persistir
      AND o.operation_finished = 0
      -- Para la delivery en curso
      AND o.delivery_id = @delivery_id

   -- Se finalizan (operation_finished = 1) las operaciones realizadas en el paso anterior
   UPDATE o SET operation_finished = 1
   FROM [IO].[load].customer c
      INNER JOIN [IO].[load].operation o ON o.delivery_id = c.delivery_id
         AND o.load_reference_id = c.id
   WHERE
      -- Tabla correcta
      o.load_table_name = 'customer'
      -- Operación de tipo Insert
      AND o.operation_type_id = (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Insert')
      -- Operación sin persistir
      AND o.operation_finished = 0
      -- Para la delivery en curso
      AND o.delivery_id = @delivery_id

   -- Se devuelve el número de operaciones actualizadas
   SELECT @affected_rows = @@ROWCOUNT

END
```

Código C.59: input.sp_BMWGroup_StatusUpdate_persistence_insert_proposal.sql

```
USE [IO];
GO
/*
-- ================================================================================
Autor: Adrian Maroto
Fecha creacion: 25/03/2024
Descripcion: Proceso de inserción de clientes del fichero StatusUpdate en VehicleSales.dbo.proposal.
Cambios:

Entrada:
   @delivery_id: Id de la delivery abierta en curso.

Salida:
   @affected_rows: Número de filas afectadas por el proceso.

Ejemplo:
   EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_persistence_insert_proposal] @delivery_id = 1, @affected_rows = 0

   SELECT * FROM [IO].[load].delivery ORDER BY id DESC
   SELECT * FROM [IO].[load].operation
   SELECT * FROM [IO].[load].operation_type

   SELECT * FROM [IO].[input].[BMWGroup_StatusUpdate]
   SELECT * FROM [IO].[load].[proposal]
   SELECT * FROM VehicleSales.dbo.proposal

   SELECT * FROM VehicleSales.dbo.customer
   SELECT * FROM VehicleSales.dbo.seller
   SELECT * FROM VehicleSales.dbo.model_type
-- ================================================================================
```

```sql
*/
CREATE OR ALTER PROCEDURE [input].[sp_BMWGroup_StatusUpdate_persistence_insert_proposal]
   @delivery_id INT,
   @affected_rows INT OUTPUT
AS
BEGIN

   -- Se realizan todas las operaciones de INSERT calculadas en el proceso de transformación
   -- [IO].[input].[sp_BMWGroup_StatusUpdate_persistence_insert_proposal]
   INSERT INTO VehicleSales.dbo.proposal
      ([customer_id], [seller_id], [model_type_id], [proposal_date], [proposal_amount], [proposal_type_id], [active], [io_id])
   SELECT
      -- Cruzamos con VehicleSales.dbo.customer para obtener el id
      (SELECT c.id FROM VehicleSales.dbo.customer c WHERE [auxiliar].[f_clean_rare_characters](c.[name]) = [auxiliar].[f_clean_rare_characters]('Mario Torres')) AS customer_id,
      -- Cruzamos con VehicleSales.dbo.seller para obtener el id
      (SELECT s.id FROM VehicleSales.dbo.seller s WHERE [auxiliar].[f_clean_rare_characters](s.[name]) = [auxiliar].[f_clean_rare_characters](p.vendedor)) AS seller_id,
      -- Cruzamos con VehicleSales.dbo.seller para obtener el id
      (SELECT mt.id FROM VehicleSales.dbo.model_type mt WHERE mt.[name] = p.modelo) AS model_type_id,
      p.proposal_date,
      p.proposal_amount,
      p.proposal_type_id,
      p.active,
      p.id AS io_id
   FROM [IO].[load].proposal p
      INNER JOIN [IO].[load].operation o ON o.delivery_id = p.delivery_id
         AND o.load_reference_id = p.id
   WHERE
      -- Tabla correcta
      o.load_table_name = 'proposal'
      -- Operación de tipo Insert
      AND o.operation_type_id = (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Insert')
      -- Operación sin persistir
      AND o.operation_finished = 0
      -- Para la delivery en curso
      AND o.delivery_id = @delivery_id

   -- Se finalizan (operation_finished = 1) las operaciones realizadas en el paso anterior
   UPDATE o SET operation_finished = 1
   FROM [IO].[load].proposal p
      INNER JOIN [IO].[load].operation o ON o.delivery_id = p.delivery_id
         AND o.load_reference_id = p.id
   WHERE
      -- Tabla correcta
      o.load_table_name = 'proposal'
      -- Operación de tipo Insert
      AND o.operation_type_id = (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Insert')
      -- Operación sin persistir
      AND o.operation_finished = 0
      -- Para la delivery en curso
      AND o.delivery_id = @delivery_id

   -- Se devuelve el número de operaciones actualizadas
   SELECT @affected_rows = @@ROWCOUNT

END
```

Código C.60: input.sp_BMWGroup_StatusUpdate_persistence_insert_offer.sql

```sql
USE [IO];
GO
/*
-- ============================================================================================
Autor: Adrian Maroto
Fecha creacion: 25/03/2024
Descripcion: Proceso de inserción de ofertas del fichero StatusUpdate en VehicleSales.dbo.offer.
Cambios:

Entrada:
   @delivery_id: Id de la delivery abierta en curso.

Salida:
   @affected_rows: Número de filas afectadas por el proceso.

Ejemplo:
   EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_persistence_insert_offer] @delivery_id = 1, @affected_rows = 0

   SELECT * FROM [IO].[load].delivery ORDER BY id DESC
   SELECT * FROM [IO].[load].operation
   SELECT * FROM [IO].[load].operation_type

   SELECT * FROM [IO].[input].[BMWGroup_StatusUpdate]
   SELECT * FROM [IO].[load].[offer]
   SELECT * FROM [IO].[load].[proposal]
   SELECT * FROM VehicleSales.dbo.offer
-- ============================================================================================
*/
CREATE OR ALTER PROCEDURE [input].[sp_BMWGroup_StatusUpdate_persistence_insert_offer]
   @delivery_id INT,
   @affected_rows INT OUTPUT
AS
BEGIN

   -- Se realizan todas las operaciones de INSERT calculadas en el proceso de transformación
   -- [IO].[input].[sp_BMWGroup_StatusUpdate_persistence_insert_offer]
   INSERT INTO VehicleSales.dbo.offer
```

```
        ([proposal_id], [offer_date], [offer_amount], [active], [io_id])
    SELECT
        -- Cruzamos con VehicleSales.dbo.proposal para obtener el id
        (SELECT p.id FROM VehicleSales.dbo.proposal p WHERE p.io_id = ofe.proposal_io_id) AS proposal_id,
        ofe.offer_date,
        ofe.offer_amount,
        ofe.active,
        ofe.id AS io_id
    FROM [IO].[load].offer ofe
        INNER JOIN [IO].[load].operation o ON o.delivery_id = ofe.delivery_id
            AND o.load_reference_id = ofe.id
    WHERE
        -- Tabla correcta
        o.load_table_name = 'offer'
        -- Operación de tipo Insert
        AND o.operation_type_id = (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Insert')
        -- Operación sin persistir
        AND o.operation_finished = 0
        -- Para la delivery en curso
        AND o.delivery_id = @delivery_id

    -- Se finalizan (operation_finished = 1) las operaciones realizadas en el paso anterior
    UPDATE o SET operation_finished = 1
    FROM [IO].[load].offer ofe
        INNER JOIN [IO].[load].operation o ON o.delivery_id = ofe.delivery_id
            AND o.load_reference_id = ofe.id
    WHERE
        -- Tabla correcta
        o.load_table_name = 'offer'
        -- Operación de tipo Insert
        AND o.operation_type_id = (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Insert')
        -- Operación sin persistir
        AND o.operation_finished = 0
        -- Para la delivery en curso
        AND o.delivery_id = @delivery_id

    -- Se devuelve el número de operaciones actualizadas
    SELECT @affected_rows = @@ROWCOUNT

END
```

Código C.61: input.sp_BMWGroup_StatusUpdate_persistence_insert_sale.sql

```
USE [IO];
GO
/*
-- ================================================================================================
Autor: Adrian Maroto
Fecha creacion: 25/03/2024
Descripcion: Proceso de inserción de ventas del fichero StatusUpdate en VehicleSales.dbo.sale.
Cambios:

Entrada:
    @delivery_id: Id de la delivery abierta en curso.

Salida:
    @affected_rows: Número de filas afectadas por el proceso.

Ejemplo:
    EXEC [IO].[input].[sp_BMWGroup_StatusUpdate_persistence_insert_sale] @delivery_id = 1, @affected_rows = 0

    SELECT * FROM [IO].[load].delivery ORDER BY id DESC
    SELECT * FROM [IO].[load].operation
    SELECT * FROM [IO].[load].operation_type

    SELECT * FROM [IO].[input].[BMWGroup_StatusUpdate]
    SELECT * FROM [IO].[load].[offer]
    SELECT * FROM [IO].[load].[sale]
    SELECT * FROM VehicleSales.dbo.sale
-- ================================================================================================
*/
CREATE OR ALTER PROCEDURE [input].[sp_BMWGroup_StatusUpdate_persistence_insert_sale]
    @delivery_id INT,
    @affected_rows INT OUTPUT
AS
BEGIN

    -- Se realizan todas las operaciones de INSERT calculadas en el proceso de transformación
    -- [IO].[input].[sp_BMWGroup_StatusUpdate_persistence_insert_sale]
    INSERT INTO VehicleSales.dbo.sale
        ([offer_id], [sale_date], [io_id])
    SELECT
        -- Cruzamos con VehicleSales.dbo.proposal para obtener el id
        (SELECT o.id FROM VehicleSales.dbo.offer o WHERE o.io_id = s.offer_io_id) AS offer_id,
        s.sale_date,
        s.id AS io_id
    FROM [IO].[load].sale s
        INNER JOIN [IO].[load].operation o ON o.delivery_id = s.delivery_id
            AND o.load_reference_id = s.id
    WHERE
        -- Tabla correcta
        o.load_table_name = 'sale'
        -- Operación de tipo Insert
        AND o.operation_type_id = (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Insert')
        -- Operación sin persistir
        AND o.operation_finished = 0
        -- Para la delivery en curso
```

```
      AND o.delivery_id = @delivery_id

   -- Se finalizan (operation_finished = 1) las operaciones realizadas en el paso anterior
   UPDATE o SET operation_finished = 1
   FROM [IO].[load].sale s
      INNER JOIN [IO].[load].operation o ON o.delivery_id = s.delivery_id
         AND o.load_reference_id = s.id
   WHERE
      -- Tabla correcta
      o.load_table_name = 'sale'
      -- Operación de tipo Insert
      AND o.operation_type_id = (SELECT id FROM [IO].[load].operation_type WHERE operation_type_name = 'Insert')
      -- Operación sin persistir
      AND o.operation_finished = 0
      -- Para la delivery en curso
      AND o.delivery_id = @delivery_id

   -- Se devuelve el número de operaciones actualizadas
   SELECT @affected_rows = @@ROWCOUNT

END
```

La *Figura C.8* muestra el conjunto de plantillas de los casos de uso del proyecto IO.

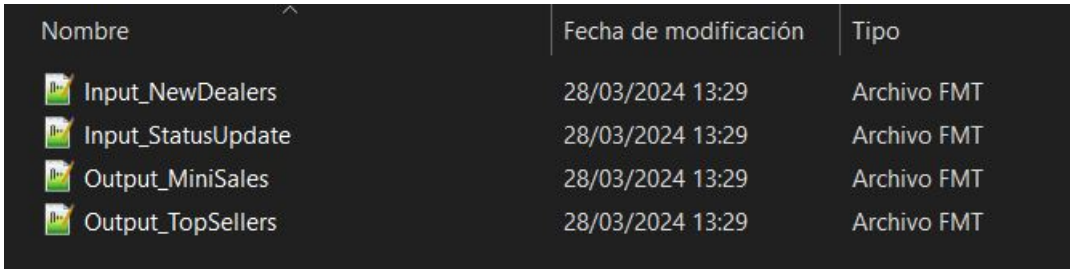


Figura C.8: Estructura de plantillas de los casos de uso del proyecto IO

A continuación el *Código C.62-C.65* muestra las plantillas para los diferentes casos de uso.

Código C.62: Input_NewDealers.fmt

```
12.0
7
1   SQLCHAR    0   0   ",\""   1 Nombre            SQL_Latin1_General_CP1_CI_AS
2   SQLCHAR    0   0   ",\""   2 Direccion         SQL_Latin1_General_CP1_CI_AS
3   SQLCHAR    0   0   ",\""   3 Ciudad            SQL_Latin1_General_CP1_CI_AS
4   SQLCHAR    0   0   ",\""   4 Pais           SQL_Latin1_General_CP1_CI_AS
5   SQLCHAR    0   0   ",\""   5 CodigoPostal      SQL_Latin1_General_CP1_CI_AS
6   SQLCHAR    0   0   ",\""   6 Auxiliar1         SQL_Latin1_General_CP1_CI_AS
7   SQLCHAR    0   0   "\n"    7 Auxiliar2         SQL_Latin1_General_CP1_CI_AS
```

Código C.63: Output_TopSellers.fmt

```
12.0
13
1    SQLCHAR    0   0    ";"   1 top                      SQL_Latin1_General_CP1_CI_AS
2    SQLCHAR    0   0    ";"   2 concesionario            SQL_Latin1_General_CP1_CI_AS
3    SQLCHAR    0   0    ";"   3 vendedor               SQL_Latin1_General_CP1_CI_AS
4    SQLCHAR    0   0    ";"   4 ventas_totales           SQL_Latin1_General_CP1_CI_AS
5    SQLCHAR    0   0    ";"   5 importe_ventas_totales      SQL_Latin1_General_CP1_CI_AS
6    SQLCHAR    0   0    ";"   6 fecha_ultima_venta          SQL_Latin1_General_CP1_CI_AS
7    SQLCHAR    0   0    ";"   7 ventas_ultimo_mes        SQL_Latin1_General_CP1_CI_AS
8    SQLCHAR    0   0    ";"   8 promedio_ventas_mensuales    SQL_Latin1_General_CP1_CI_AS
9    SQLCHAR    0   0    ";"   9 promedio_importe_ventas      SQL_Latin1_General_CP1_CI_AS
10   SQLCHAR    0   0    ";"   10   ofertas_totales             SQL_Latin1_General_CP1_CI_AS
11   SQLCHAR    0   0    ";"   11   fecha_ultima_oferta         SQL_Latin1_General_CP1_CI_AS
12   SQLCHAR    0   0    ";"   12   propuestas_totales          SQL_Latin1_General_CP1_CI_AS
13   SQLCHAR    0   0    "\n"  13   fecha_ultima_propuesta      SQL_Latin1_General_CP1_CI_AS
```

Código C.64: Output_MiniSales.fmt

```
12.0
12
1   SQLCHAR    0   0   ";"   1   marca             SQL_Latin1_General_CP1_CI_AS
2   SQLCHAR    0   0   ";"   2   modelo               SQL_Latin1_General_CP1_CI_AS
3   SQLCHAR    0   0   ";"   3   tipo_propuesta       SQL_Latin1_General_CP1_CI_AS
4   SQLCHAR    0   0   ";"   4   tipo_modelo          SQL_Latin1_General_CP1_CI_AS
5   SQLCHAR    0   0   ";"   5   importe_venta        SQL_Latin1_General_CP1_CI_AS
6   SQLCHAR    0   0   ";"   6   fecha_venta          SQL_Latin1_General_CP1_CI_AS
7   SQLCHAR    0   0   ";"   7   fecha_oferta         SQL_Latin1_General_CP1_CI_AS
8   SQLCHAR    0   0   ";"   8   cliente              SQL_Latin1_General_CP1_CI_AS
9   SQLCHAR    0   0   ";"   9   importe_propuesta    SQL_Latin1_General_CP1_CI_AS
```

```
12  10  SQLCHAR    0   0   ";"    10  fecha_propuesta         SQL_Latin1_General_CP1_CI_AS
13  11  SQLCHAR    0   0   ";"    11  concesionario        SQL_Latin1_General_CP1_CI_AS
14  12  SQLCHAR    0   0   "\n"   12  vendedor          SQL_Latin1_General_CP1_CI_AS
```

Código C.65: Input StatusUpdate.fmt

```
1   12.0
2   14
3   1    SQLCHAR    0   0    ";"    1 modelo                SQL_Latin1_General_CP1_CI_AS
4   2    SQLCHAR    0   0    ";"    2 cliente               SQL_Latin1_General_CP1_CI_AS
5   3    SQLCHAR    0   0    ";"    3 tel_cliente           SQL_Latin1_General_CP1_CI_AS
6   4    SQLCHAR    0   0    ";"    4 email_cliente         SQL_Latin1_General_CP1_CI_AS
7   5    SQLCHAR    0   0    ";"    5 fecha_propuesta          SQL_Latin1_General_CP1_CI_AS
8   6    SQLCHAR    0   0    ";"    6 cantidad_propuesta       SQL_Latin1_General_CP1_CI_AS
9   7    SQLCHAR    0   0    ";"    7 tipo_propuesta        SQL_Latin1_General_CP1_CI_AS
10  8    SQLCHAR    0   0    ";"    8 fecha_oferta          SQL_Latin1_General_CP1_CI_AS
11  9    SQLCHAR    0   0    ";"    9 cantidad_oferta          SQL_Latin1_General_CP1_CI_AS
12  10   SQLCHAR    0   0    ";"    10   fecha_venta           SQL_Latin1_General_CP1_CI_AS
13  11   SQLCHAR    0   0    ";"    11   concesionario          SQL_Latin1_General_CP1_CI_AS
14  12   SQLCHAR    0   0    ";"    12   vendedor            SQL_Latin1_General_CP1_CI_AS
15  13   SQLCHAR    0   0    ";"   13 tel_vendedor           SQL_Latin1_General_CP1_CI_AS
16  14   SQLCHAR    0   0    "\n"   14   email_vendedor          SQL_Latin1_General_CP1_CI_AS
```