UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Proyecto Fin de Grado en Ingeniería Informática

HERRAMIENTA PARA ASISTENCIA AL CÁLCULO DE CARROZADO DE CAMIONES

JUAN MANUEL GÓMEZ GARCÍA

Director: Alfonso Urquía Moraleda

Curso 2022/2023 Septiembre

HERRAMIENTA PARA ASISTENCIA AL CÁLCULO DE CARROZADO DE CAMIONES

Proyecto Fin de Grado de Ingeniería Informática.

Modalidad específica

Realizado por: JUAN MANUEL GÓMEZ GARCÍA

Director: Alfonso Urquía Moraleda

Fecha de lectura y defensa: Septiembre de 2023

Agradecimientos

Me gustaría en este apartado expresar mis agradecimientos por el apoyo, la ayuda o la comprensión de varias personas durante los ocho años que he estado dedicando al grado para el que ahora presento el proyecto.

- A mi esposa Marta, por ser el pilar de la familia, por el esfuerzo adicional que le ha supuesto durante estos ocho años mis horas dedicadas al estudio de este grado, por la generosidad y amor con que lo ha asumido, por sus sonrisas, por sus ánimos constantes en cada examen y por la paciencia que siempre ha tenido conmigo en mis momentos bajos.
- A mis hijos Samuel y Paola, por las horas que he dejado de dedicaros. Por el nerviosismo del que he podido haceros foco y por las sonrisas, besos y abrazos que he recibido de vosotros cada día de este tiempo que han supuesto los mayores fuerza y ánimo posibles.
- A mis padres Manuel y Maria Victoria por su apoyo y amor constante, no sólo en este grado, sino durante toda mi vida y por enseñarme el valor del esfuerzo.
- A mi hermano Ramiro por la ayuda y los conocimientos que me ha prestado sobre el diseño gráfico y por las constantes bromas y risas que genera cada día.
- A mi hermana Victoria por su apoyo, su cariño y los cuidados médicos que seguro se han incrementado con el estrés de estos años.
- Al resto de mi familia por su cariño constante.
- A mis amigos Félix y Zubi, por no echarme en cara las horas que hemos dejado de pasar juntos estos años.
- A mi cuñada Ana por cada mensaje de ánimo antes de cada examen.
- Un agradecimiento especial a mi tutor Alfonso Urquía, por su constante puntualidad en las respuestas, su claridad en las aclaraciones y por la dedicación y profesionalidad que ha mostrado en cada momento que me ha llevado la preparación de este proyecto.

Resumen

La finalidad de este proyecto ha sido generar la herramienta “TruckCalc”, que sirve de soporte para el cálculo de carrozado de camiones. Éste requiere de la presentación ante la Inspección Técnica de Vehículos de un proyecto firmado por un técnico competente que acredite que la estructura del vehículo es capaz de soportar las cargas a las que se verá sometido el vehículo tras la ejecución del trabajo de carrozado correspondiente.

Esta herramienta presentará en un documento un anexo de cálculo requerido en todos los proyectos en el que se recogerán los principales datos que deben analizarse en estos casos como son: reacciones en los ejes de ruedas, momentos flectores, esfuerzos cortantes y tensión máxima en las vigas de la estructura y carga máxima en el King Pin o elemento de anclaje del remolque a la cabeza tractora.

En la mayoría de las ocasiones este tipo de cálculos se realizan utilizando herramientas genéricas que resultan lentas y obligan a disponer de varias herramientas para el mimo propósito que deben ser combinadas.

En los casos en que sí existen herramientas más específicas, éstas suelen centrarse en el cálculo de la carga sobre ejes o suelen ser específicas para Windows por lo que, bien carecen de todos los resultados exigibles o bien los usuarios de otras plataformas o sistemas se ven forzados a tener que utilizar máquinas virtuales o emplear equipos expresamente orientados a poder ejecutar estas herramientas.

En este proyecto se consigue:

- Proporcionar la posibilidad de usar la herramienta en varios sistemas operativos (Windows, Linux, MacOs) empleando la maquina virtual Java en cualquiera de ellos.
- Exportar los datos de los cálculos necesarios para la justificación de la estructura a un único archivo para adjuntar al proyecto a presentar en la Inspección Técnica de Vehículos.
- Utilización de un sistema específicamente diseñado para este propósito que sustituya a sistemas genéricos.
- Utilización de un sistema rápido y eficiente que permita plantear la geometría del camión.
- Utilización de bases de datos específicas de los tipos de camión utilizado.
- Utilización de bases de datos de secciones elementos resistentes de estructura para los camiones.
- Gestión de cargas para el proyecto.

- Selección del tipo de camión y gestión del cálculo acorde al mismo.
- Almacenamiento de los datos del proyecto utilizado.
- Recomendaciones sobre los datos introducidos.
- División e implementación de las distintas entidades de la solución en clases claramente establecidas manteniendo en lo posible la mayor abstracción y cohesión y reduciendo el acoplamiento. Se ha utilizado también, en la medida de lo posible, los mecanismos de herencia y polimorfismo sobre la solución propuesta y su codificación. Además se ha focalizado la solución en la posible ampliación futura con nuevas funcionalidades evitando que pudiese resultar compleja o inadecuada.

Palabras clave

Carrozado, Camión Rígido, Camión Articulado, Cabeza Tractora, Remolque, Sección, Momento de Inercia, Área, Módulo Resistente, ITV, Momento flector, Esfuerzo Cortante, Reacción, Carga Puntual, Carga Distribuida.

Abstract

The goal of this project has been to create a tool to support the truck bodyworks calculation that requires to provide a project to the Technical Inspection of Vehicles which must be signed by an authorized technician that proves that the vehicle structure is able to support the loads that it will be submitted to after the corresponding truck bodyworks execution.

This tool will provide a calculation annex which is required in any project and that will cover the main information that should be analyzed in these cases as: Reaction forces in the wheel axis, Bending moments, Shear forces and Maximum stress in the structure beams. Also maximum load in King Pin or trailer fixing point to the towing vehicle.

In most of the cases these calculations are carried out using generic tools that become slow and make necessary to have several tools for same final purpose.

In those cases where specific tools exist, whether they are focussed in the calculation of the loads at wheel axis or they are specific for Windows Operative System which forces the users of other operative systems to have other virtual machines or teams focussed to execute these toolings.

In this project, next goals are reached:

- Provide the possibility to use several platforms.
- Export the data of the necessary calculations to justify the structure validity to a single file that can be annexed to the project that will be submitted to the Technical Inspection of Vehicles.
- Utilization of a specifically designed system for this purpose that substitute generic systems.
- Utilization of a quick and efficient system to allow the truck geometry introduction .
- Utilization of data bases specific for the used truck types.
- Utilization of data bases for the resistant element sections of the trucks structure.
- Management of the project loads.
- Selection of the truck type and management of the calculation which will be adapted to it.
- Storage of the project information.
- Recommendations while introducing the data.
- Division and implementation of the different entities of the solution in clearly stablished classes keeping the highest abstraction and cohesion and reducing coupling. Using polymorphism and heritage in the solution. Besides, the solution was focussed in enabling future added functions or improvements reducing the complexity of such tasks.

Keywords

Truck Bodyworks, Rigid Truck, Articulated Lorry, Towing Vehicle, Trailer, Section, Inertial Moment, Area, Resistant Modulus, ITV, Bending Moment, Shear Force, Reaction Force, Punctual Load, Distributed Load.

Índice general

Lista de figuras

Lista de tablas

Capítulo 1. Introducción, objetivos y estructura

1.1. Introducción

El actual proyecto se ha planteado para servir de ayuda en la presentación de los cálculos que los proyectistas deben presentar a los colegios profesionales a la hora de conseguir los visados de proyecto para su posterior presentación a la autoridad competente. En este caso, dicha autoridad está representada por la Inspección Técnica de Vehículos (ITV) que realiza la revisión de los mencionados proyectos.

Previamente a la presentación ante la ITV, el proyecto debe entregarse a un colegio profesional apropiado (normalmente el colegio de ingenieros o ingenieros técnicos) quien visará el trabajo comprobando que contiene la información necesaria.

Estos proyectos deberán reflejar la descripción de los trabajos realizados sobre los vehículos, así como las implicaciones estructurales para que, una vez los citados vehículos reciban su autorización, puedan circular en las vías públicas.

En el caso que nos ocupa, el principal problema se refleja en que los distintos modelos de camión se adaptan para ser capaces de transportar distintas tipologías de carga. Además de este hecho, esas tipologías incluyen diferentes elementos que suponen la generación de distintos tipos de carga que ejercerán su acción sobre la estructura del vehículo.

Para que el vehículo no suponga un peligro, la estructura de la que está equipado deberá ser capaz de soportar dichas cargas en condiciones de funcionamiento.

Los puntos más importantes que esta última consideración supone son los siguientes:

a. Las secciones de la perfilería de acero utilizada en el bastidor del camión deben soportar la carga a la que se ve sometida, lo que implica resistir los esfuerzos cortantes y flectores que en este tipo de estructura son predominantes.
b. Las mencionadas cargas generan unas solicitaciones que se transforman en reacciones en los distintos ejes del vehículo. Dichos ejes son capaces de resistir una carga máxima por ficha técnica del vehículo. Bajo ningún concepto deberá sobrepasarse dicha carga máxima en ninguno de los ejes.
c. Debe conocerse la carga sobre el elemento que une el remolque a la cabeza tractora, ya que es objeto de otro tipo de comprobaciones estructurales.
d. Las distintas tipologías de transporte generan distintos repartos de carga sobre el bastidor del camión. Por ejemplo: una cisterna con líquido genera una carga uniformemente repartida sobre el bastidor, mientras que una grúa para remolque genera una carga puntual sobre el punto donde ésta se ancla al bastidor. Así, las cargas de uno y otro caso serán totalmente distintas.
e. Existen dos tipologías básicas de camión: rígido y articulado. En el primer caso el bastidor es rígido, considerándose como tal el que no tiene capacidad de dividirse. Este tipo se trata de una viga continua desde el frente al final del camión. En el segundo caso existe un bastidor para la cabeza tractora, discurre desde el frente de dicha cabeza hasta poco más atrás del eje trasero. Existe además otro bastidor para el remolque, discurre desde el elemento de conexión con la cabeza (comúnmente conocido como King Pin) hasta el final del remolque.

En la actualidad, la manera más común de realizar estos informes y proyectos es de una manera cuasi-artesanal. Se realiza cada cálculo de forma independiente, utilizando software no orientado a este propósito (el carrozado de camiones). Este software no permite la modificación de los datos del camión de una forma sencilla entre ajuste y ajuste de los parámetros del mismo. Por esta razón, el proyectista se obliga a estar “saltando” de aplicación a aplicación para ir obteniendo de cada una de ellas la información necesaria. Este proceso resulta totalmente ineficiente, incómodo y requiere de demasiado tiempo. Este tiempo se ha empleado en realizar una tarea que en la mayoría de las ocasiones puede hacerse de manera mecánica. En las pocas ocasiones en que este software si existe, bien está orientado a una empresa concreta o bien no tiene una versión que pueda trabajar bajo otro sistema operativo que no sea Windows.

1.2. Objetivos

El objetivo principal es desarrollar una herramienta denominada "TruckCalc" para que el proyectista puede obtener rápidamente un anexo técnico, que presente de manera clara la información requerida. Este objetivo principal puede descomponerse en los siguientes objetivos secundarios:

a. Proporcionar una aplicación independiente del sistema operativo sobre el que trabajemos (Windows, Mac o Linux).
b. Proporcionar una interfaz de introducción de los datos del camión que facilite la introducción de los datos geométricos de los camiones a utilizar.
c. Proporcionar un sistema de bases de datos que sirva para ir almacenando tipologías de camión sobre las que realizar las posteriores aplicaciones de cargas.
d. Proporcionar una herramienta de cálculo de propiedades de secciones que sirva para ayudar al proyectista a decidir la sección a utilizar así como para disponer de una referencia sobre sus propiedades mecánicas.
e. Proporcionar un sistema de bases de datos para poder almacenar las mencionadas secciones.
f. Disponer de una interfaz rápida que permita la evaluación de los siguientes elementos de un solo vistazo:
   - Esquema del camión para entender dónde se aplican las cargas.
   - Esquema de cargas cortantes.
   - Esquema de momentos flectores.
   - Lista de cargas introducidas.
   - Reacciones en los ejes así como la posible superación de las cargas máximas.
   - Tensiones en las vigas del bastidor.
g. Disponer de la separación entre los dos tipos de camión (rígido y articulado) a la hora de realizar los cálculos sobre el mismo.
h. Proporcionar un sistema de almacenamiento de proyectos en fichero.
i. Proporcionar un sistema de impresión rápida a pdf para generación del anexo técnico del proyecto.
j. Proporcionar la capacidad de realizar un cambio rápido de unidades en caso necesario tanto en longitud como en carga.

La consecución de los mencionados objetivos requerirá de las siguientes tareas claramente diferenciadas:

a. Obtener de las partes interesadas las necesidades y las mejoras del proceso de trabajo.
b. Establecer un plan de trabajo basado en prototipos funcionales que poder enseñar a las partes interesadas.
c. Estudiar en detalle el proceso de cálculo de estos tipos de estructuras y las tareas críticas y que pueden automatizarse así como la metodología de automatización de las mismas.
d. Diferenciar claramente los tipos de cálculos.
e. Establecer los resultados que debe proveer el sistema.
f. Generar un modelo de dominio del problema y un análisis completo del mismos.
g. Realizar un diseño de la herramienta.
h. Codificar las diferentes clases de la herramienta.
i. Realizar las pruebas sobre la herramienta.

1.3. Estructura

La memoria se ha estructurado en 7 capítulos y anexos que contendrán la siguiente información:

- Capítulo 1. Introducción, Objetivos y Estructura: Este capítulo realiza una breve introducción del proyecto, describirá los objetivos que se pretendían cumplir y expone la estructura de la memoria.
- Capítulo 2. Estado del arte: Este capítulo realiza una introducción inicial sobre cálculos básicos de estructura así como su metodología básica, introduce conceptos específicos de cálculo de camiones y describe los elementos de los mismos, describe las necesidades en un proyectos de carrozado y sus requerimientos en las ITV y expone una breve descripción del estado del arte.
- Capítulo 3. Análisis: Este capítulo describe la fase de análisis del desarrollo del software. Expone también los requerimientos por parte de los dos agentes principales como son el carrocero y la ITV. Expone brevemente la estructura del proyecto para entregar y expone el por qué de la estructura del anexo que el software proveerá.
- Capítulo 4. Diseño: Este capítulo realiza la descripción de la implantación del software. Describe los diagramas de secuencia y las clases que se han utilizado para la generación de la solución describiendo la razón de cada una.

- Capítulo 5. Implementación y Pruebas.: Este capítulo describe el entorno de desarrollo software y hardware. Explica la forma en que se ha implementado la solución mediante el uso de JAVA, Apache Derby SQL y IText7 y finalmente describe las pruebas que se han realizado para la comprobación del funcionamiento de la aplicación.
- Capítulo 6. Planificación y Costes del proyecto: Este capítulo describe las fases que componen el proyecto así como las variaciones sobre el planning inicial y sus razones. Se expone también una estimación del coste de realización del proyecto y las conclusiones sobre estos puntos.
- Capítulo 7. Conclusiones y trabajos futuros: Este capítulo plantea unas conclusiones finales sobre el proyecto y sus objetivos. Asimismo plantea posibles vías de desarrollo futuro sobre el proyecto y sus mejoras.
- Anexo A. Contiene la información del Manual de instalación de la herramienta TruckCalc.
- Anexo B. Contiene el Manual del Usuario de la herramienta TruckCalc.
- Anexo C. Contiene el código de los diferentes paquetes en que se estructura la aplicación.

Capítulo 2. Cálculo del carrozado de vehículos

2.1. Introducción

Se pretende en este capítulo realizar una exposición del estado del arte en este tipo de aplicaciones. Aunque no se tratará en profundidad (puesto que no es el objetivo de este trabajo), se realizará una breve exposición sobre conceptos básicos de cálculo de estructuras que permita entender la información que se está proporcionando a la aplicación y la información que ésta va a proveer al usuario.

Se realizará también una exposición de los puntos requeridos en el cálculo de carrozado para la aprobación por parte de las ITV de los trabajos que sobre el vehículo se realiza.

Finalmente se expondrán las soluciones actualmente disponibles en el mercado y sus posibilidades.

2.2. Conceptos de cálculo de estructuras

Se detallan a continuación algunos conceptos muy básicos utilizados en la ingeniería para el cálculo de estructuras.

Unidades

En primer lugar describimos las unidades más comunes para el cálculo de estructuras, centrándonos únicamente en las que se utilizarán en esta aplicación.

Se utilizan generalmente longitud y carga en este tipo de cálculos. Para la longitud se utilizan comúnmente milímetros, centímetros, decímetros y metros.

Para las fuerzas se utilizan kilogramos-fuerza [kgf] o kilopondio [kp] (de aquí en adelante se hablará de kilogramos por ser lo más comúnmente utilizado, si bien debe entenderse que se habla de kilogramo-fuerza o kilopondio). Otra unidad de fuerza es el Newton siendo 1 kgf equivalente en magnitud a 9,8067 Newtons.

Reacciones

En la realización de un cálculo de estructuras se debe plantear un doble sistema de equilibrio de fuerzas y momentos.

Las fuerzas se caracterizan por tener una dirección y una magnitud. La dirección puede estar orientada en cualquiera de las tres dimensiones para el caso de estructuras tridimensionales.

Los momentos generados por una fuerza sobre un punto origen se definen como el producto de una fuerza y su distancia perpendicular hasta ese origen desde la dirección de la fuerza. Por ejemplo, una fuerza de 1 kgf, a una distancia perpendicular a la dirección de la fuerza de 1 metro sobre nuestro origen para los momentos, generaría un momento de 1 kgf·m.

Como se ha mencionado, cuando se plantea un sistema que necesitamos calcular, se debe plantear un doble sistema de ecuaciones. En el primer sistema la suma de las fuerzas netas considerando cargas y reacciones debe ser cero. Asimismo el segundo sistema debe garantizar que la suma de los momentos generados por todas las fuerzas y reacciones desde un origen determinado sea cero.

Esfuerzo normal

Los esfuerzos normales sobre una viga se producen en dirección perpendicular a la sección de la misma, de manera que puede generarse una carga de tracción o compresión en función de si este esfuerzo tiene a estirar o comprimir la viga. El resultado de este esfuerzo sobre una sección es una tensión normal representada por la letra griega σ (sigma).

La unidad en este caso sería la fuerza por unidad de superficie. Por ejemplo: una posible unidad sería kg/m^2.

Se presenta en la Figura 2.1 y Figura 2.2 una viga teórica sometida a un esfuerzo normal con las dos casuísticas de tracción y compresión:

Figura 2.1. Esfuerzo normal de tracción

Figura 2.2. Esfuerzo normal de compresión

Esfuerzo cortante

Los esfuerzos cortantes sobre una viga se producen en la dirección del plano de la sección, generando tensiones cortantes que suelen representarse por la letra griega τ (tau). La unidad de medida es la fuerza por unidad de superficie. Por ejemplo: una posible unidad sería kg/m^2. La Figura 2.3 muestra la aplicación de un esfuerzo cortante sobre un elemento diferencial de viga.

Figura 2.3. Esfuerzo cortante sobre un elemento diferencial de viga

Momento flector

El momento flector se produce como consecuencia de la aplicación de una carga perpendicular a una sección a una distancia dada de los apoyos de la misma.

En la Figura 2.4 se muestra una viga sometida a un momento flector sobre una viga biapoyada. El cálculo del momento flector se obtiene como el producto de la distancia de la fuerza a un origen de momentos. Por ejemplo: En la figura 2.4, llamaremos F a la fuerza representada en la parte superior. La fuerza se encuentra en la mitad de la longitud de la viga azul (llamando L a la longitud de esta

viga). Si tomásemos origen en el extremo izquierdo , el momento generado por la fuerza F desde ese origen es:

$$M = F\frac{L}{2}$$

Las unidades por tanto serían las de fuerza por longitud y una posible unidad podría ser kilogramos·metro.

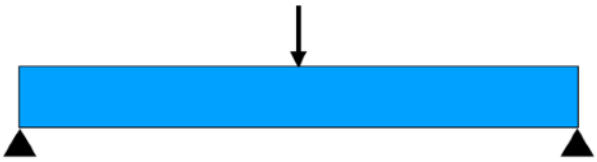


Figura 2.4. Momento flector sobre una viga biapoyada

Sección Resistente

La sección o área resistente comprende el área encerrada por la sección de la viga o elemento estructural que se va a considerar. En el caso de este proyecto, se están considerando secciones uniformes que no varían a lo largo de la longitud de la viga a considerar como simplificación comúnmente aceptada. La sección resistente se mide en longitud al cuadrado. Por ejemplo: una posible unidad serían metros al cuadrado.

Momento de inercia

El momento de inercia (al que se suele llamar inercia en este tipo de cálculos para acortar) es una propiedad de las secciones de los elementos estructurales. El cálculo depende de la geometría de la sección del elemento estructural. No entramos a definir su cálculo y bastará decir que es utilizada para calcular tensiones y deformaciones sobre las mismas. Sus valores son comúnmente proporcionados en prontuarios de tipologías de vigas para ser directamente aplicados en el cálculo o bien calculados mediante software específico. En los prontuarios de secciones, la unidad utilizada es la de longitud elevada a la cuarta potencia. Una posible unidad es mm^4.

Módulo Resistente

De igual forma que en el punto anterior, tampoco entramos a definir como se calcula el modulo resistente y baste decir que su valor está relacionado con la tensión que soporta el elemento estructural. En los prontuarios de secciones, la unidad con la que se le representa es longitud al cubo. Una posible unidad podría ser mm^3.

Límite elástico

Los elementos estructurales utilizados en construcción son normalmente calculados bajos unas condiciones específicas, donde no debe sobrepasarse un punto tensional en el que las deformaciones no se recuperen. Esto implica que, una vez se retira el efecto de las cargas sobre nuestra estructura, ésta debería recuperarse de la deformación que se hubiese impuesto bajo el efecto de las mismas sin mantener ningún tipo de deformación adicional. El límite elástico es una propiedad medida normalmente mediante ensayos en máquinas de tracción universal sobre una probeta de la misma composición del material a determinar. Se obtiene un diagrama de carga-deformación, en la que puede verse cuál es la tensión máxima que el elemento puede soportar sin mantener deformación plástica (o permanente) (Cervera y Blanco, 2015).

A la hora de realizar una estructura, el proyectista debe seleccionar un valor de límite elástico con el que trabajar posteriormente y asegurarse de que las cargas no sobrepasarán este valor. En caso de no ser posible eliminar cargas deberá incrementar sección o bien seleccionar un material de mayor límite elástico.

Los límites elásticos más comunes suelen estar recogidos en normativa específica para aceros conformados en frío o laminados en caliente, con una gran variedad de tipologías que permiten la selección de la más adecuada para la aplicación en cuestión.

En Europa, el límite elástico de un determinado acero se suele representar por su valor en Newton/ mm2 al que se añade un prefijo y un sufijo que indica las propiedades del mismo. Los límites más comúnmente utilizados son 235, 275, 355 y 420. Estos valores representan, como se indicaba, la tensión máxima que el elemento puede soportar de forma que, una vez se libere la carga, vuelva a su forma original sin sufrir deformaciones permanentes.

Tensión Normal

La tensión normal tiene normalmente dos componentes: una parte que proviene de los esfuerzos normales de tracción y compresión y otra parte que proviene del momento flector en los elementos resistentes.

Su calculo es normalmente realizado mediante la siguiente ecuación:

$$\sigma = \frac{N}{A} + \frac{M}{W}$$

Sus unidades son la fracción unidad de fuerza partido por unidad de longitud al cuadrado. Una posible unidad sería Newton/mm^2.

donde:

N = Esfuerzo Normal medido en unidades de fuerza. Ejemplo: Newton.

A = Sección Resistente medida en unidades de longitud al cuadrado. Ejemplo: mm^2.

M = Momento Flector medido por el producto de fuerza por distancia. Ejemplo: N·mm.

W = Módulo Resistente medido en unidades de longitud al cubo. Ejemplo: mm^3.

La flexión del elemento resistente puede producirse en dos planos, si bien para los efectos del cálculo de carrozado puede simplificarse y considerar únicamente una estructura bidimensional por lo que uno de ellos desaparece. Así, en la Figura 2.5 se aprecia un chasis compuesto por dos vigas (representadas en azul). La flexión de estas vigas en la dirección 'Z' (representada en verde) no se considera por ser irrelevante en comparación con la flexión en la dirección 'Y' (representada en rojo) por lo que solo en necesario calcular en el plano de la flexión en la dirección 'Y' (Cervera y Blanco, 2015).

Tensión Cortante

La tensión cortante se calcula mediante la siguiente ecuación:

$$\tau = \frac{V}{A}$$

Sus unidades son la fracción unidad de fuerza partido por unidad de longitud al cuadrado. Una posible unidad sería Newton/mm^2.

donde:

V = Esfuerzo Cortante medido en unidades de fuerza. Ejemplo: Newton.

A = Sección Transversal medida en unidades de longitud al cuadrado. Ejemplo: mm^2.

Tensión de Von Mises

Las tensiones normal y cortante tienen consideraciones distintas y no entraremos a exponerlas aquí por su complejidad. No obstante, para calcular la tensión total del elemento deben considerarse ambas, por lo que se puede aplicar la ecuación de Von Mises que permite realizar el mencionado efecto de ambas simultáneamente. Dicha tensión viene dada por la siguiente ecuación:

$$\sigma_{VM} = \sqrt{\sigma^2 + 3\tau^2}$$

Sus unidades son la fracción unidad de fuerza partido por unidad de longitud al cuadrado. Una posible unidad sería Newton/mm^2 (Cervera y Blanco, 2015).

donde:

σ es la tensión normal, medido en unidad de fuerza por unidad de longitud al cuadrado.

τ es la tensión cortante, medido en unidad de fuerza por unidad de longitud al cuadrado.

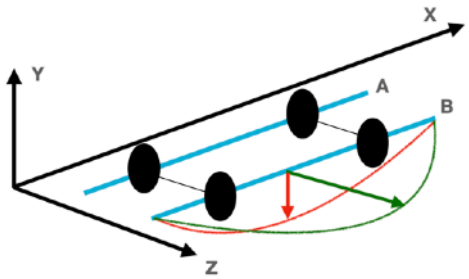


Figura 2.5. Flexiones a las que la estructura se ve sometida.

Apoyos y simplificaciones

Realizaremos un cálculo de la estructura considerándola bidimensional, es decir descartaremos las fuerzas en uno de los ejes del espacio. La razón es que el vehículo consta normalmente de dos vigas situadas en paralelo a lo largo del vehículo y cada una de ellas puede considerarse como un elemento lineal.

A la hora del cálculo se considera una de estas vigas y se considera la carga repartida en cada una de ellas, lo que permitirá una simplificación comúnmente aceptada en este tipo de proyecto.

Desde este momento hablaremos, por tanto, del caso en el que se considera la estructura como bidimensional (en el plano XY de la figura 2.5) cuando hablemos de las consideraciones a realizar cuando se tome uno u otro tipo de apoyo.

Existen distintos tipos de apoyo:

- Empotrado: Contiene todos sus grados de libertad restringidos. Es decir, no se permiten desplazamientos en ninguno de los 2 ejes. Asimismo se restringen los giros. Si tenemos un apoyo de este tipo tendremos 3 incógnitas (la reacción en el eje X, la reacción en el eje Y y el giro alrededor de un eje Z perpendicular a los dos ejes anteriores). Un símil a este tipo de apoyo sería el poste de una farola anclado al suelo. No podemos desplazarlo verticalmente, ni horizontalmente, ni tampoco girarlo en su punto de anclaje.
- Articulado: Contiene todos sus grados de libertad lineales restringidos. Es decir, no se permiten desplazamientos en ninguno de los 2 ejes. Sin embargo, en este caso, se permite el giro alrededor de uno de sus ejes. En este caso tenemos 2 incógnitas (reacciones en X y en Y). Un símil a este tipo de apoyo sería una puerta. No podemos desplazarla verticalmente ni horizontalmente pero sí podemos girarla sobre el eje de la misma.
- Deslizante: Contiene un grado de libertad lineal restringido, permitiendo el desplazamiento en el otro. Se permite asimismo el giro alrededor de uno de sus ejes. En este caso tenemos una sola incógnita (reacción en X o en Y). Un símil a este tipo de apoyo sería un conector ‘jack’ dentro de su conexión. Podemos mover el conector en dirección del eje del mismo o girarlo, pero no desplazarlo lateralmente.

La Figura 2.6 clarifica los tipos de apoyo con su representación. Se pueden apreciar las reacciones que genera cada uno de los tres tipos. Así, las flechas representan la dirección de las reacciones producidas bajo la carga en el apoyo siendo estas vertical, horizontal o un momento.

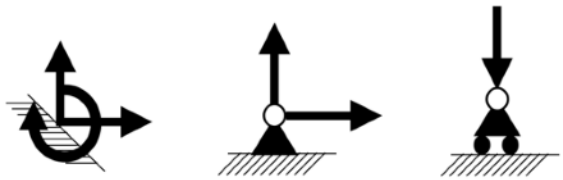


Figura 2.6. Representación de los tipos de apoyo. De izquierda a derecha: empotrado, articulado y deslizante.

Cada vez que introducimos un apoyo introducimos las incógnitas correspondientes a su tipología. Así, si introducimos un apoyo empotrado, nos encontraremos que tenemos que averiguar qué reacción vertical, horizontal y momento (o restricción del giro) se genera en el mismo.

En nuestro caso, introduciremos dos apoyos correspondientes a la rueda delantera y trasera, realizando el planteamiento de forma que se supone el primer apoyo como articulado (rueda que estaría además frenada) y el otro apoyo deslizante (la otra rueda que, en este caso no estaría frenada).

Al hacer esto, estamos introduciendo tres incógnitas: la reacción vertical del apoyo articulado, la reacción horizontal del apoyo articulado y la reacción vertical del apoyo deslizante.

Podemos suponer en este tipo de cálculo que la reacción horizontal del apoyo articulado es cero, ya que no se introducen cargas en esta dirección. Además, en caso de introducirlas, la suma de estas cargas horizontales resultaría equivalente a dicha reacción horizontal del apoyo articulado al no existir otra reacción horizontal en ningún otro apoyo, por lo que esta incógnita estaría resuelta.

Quedarían dos reacciones verticales que se calcularían con un sistema de dos ecuaciones. En la primera el sumatorio de cargas debe ser cero y en la segunda la suma de momentos debe ser cero. La resolución de este sistema arrojaría el valor de las reacciones verticales del apoyo articulado y del apoyo deslizante, obteniendo la resolución del problema.

Por otra parte, existe una complicación adicional que proviene del hecho de que la mayoría de camiones contienen tándem de rodadura en vez de ejes simple. Es decir, cada apoyo tiene 2 o 3 ejes en lugar de uno sólo, por lo que se incluirían apoyos adicionales que complicarían aún más el cálculo.

La solución pasa por simplificar cada tándem considerándolo como un único apoyo. De esta forma conseguimos reducir la complejidad del problema. Por ejemplo, un tándem de tres ejes produciría incluir en nuestro cálculo tres apoyos. En estas condiciones el número de incógnitas resulta muy

elevado para resolver el cálculo de una manera simplificada y habría que usar métodos más complejos.

Dado que, en condiciones normales, la carga que se llevará cada eje de este tándem sería cercana a un tercio del total que soporta el tándem completo, podemos realizar la simplificación de incluir un solo apoyo por cada tándem y después de realizar el cálculo correspondiente, dividir la carga obtenida entre el número de ejes del mismo.

Así por ejemplo, en el caso de un camión con un eje delantero y un tándem trasero de 3 ejes, se sustituiría el tándem trasero por un único apoyo para el cálculo. Una vez obtenida la reacción tras el cálculo, dividiríamos la misma entre 3 ruedas para obtener la carga en cada uno.

De esta forma podemos simplificar el cálculo del mencionado camión mediante la simplificación expuesta en la Figura 2.7.

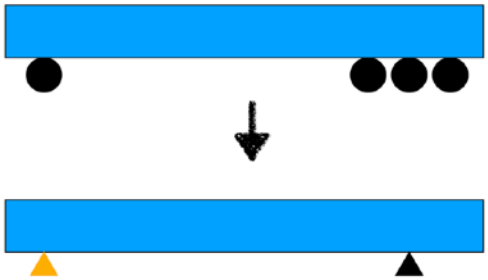


Figura 2.7. Simplificación de un camión con eje delantero y tándem triple trasero a un apoyo deslizante delantero y articulado trasero

2.3. Cargas sobre las estructuras

Se detallan a continuación las cargas que pueden aplicarse de forma más usual sobre las estructuras y la forma de representarlas.

Se entiende por “carga” el efecto de la gravedad sobre la masa de los distintos elementos que componen la estructura del camión y por otro lado, la masa de los elementos que transporta el camión(Michelin 2021). Esta carga se mide en Newtons, kilogramos-fuerza o kilopondios.

En el caso de la carga de la estructura para el carrozado, se consideran las cargas en dirección vertical, es decir, aplicadas en la dirección del eje de coordenadas Y.

Carga puntual

La carga puntual tiene la particularidad de que considera la acción de la fuerza en un único punto de la estructura. Normalmente el punto es una simplificación ya que no se tratará de un punto como tal pero la longitud sobre la que se aplica la carga será muy pequeña en comparación con el total de la longitud de la estructura.

Pertenece a esta categoría de carga las cargas como el conductor, el tanque de combustible, el motor, etc. También puede considerarse dentro de este tipo otras cargas distribuidas en una dirección vertical como podría ser el caso de un transportador sinfín orientado verticalmente.

La forma de representar este tipo de cargas es mediante una linea con punta de flecha orientada en la dirección de la misma y con la punta de flecha en el lado del sentido de la carga como se aprecia en la Figura 2.8.

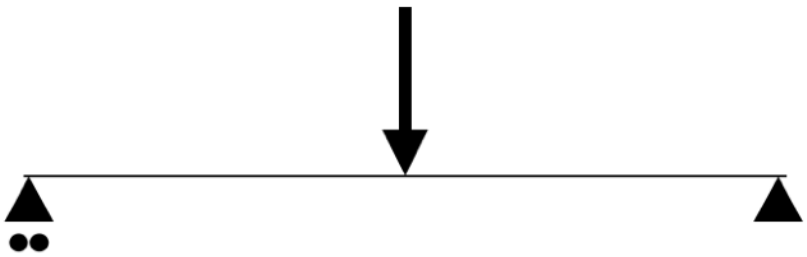


Figura 2.8. Representación de carga puntual en medio de una viga con un apoyo deslizante en la izquierda y un articulado en la derecha

Carga distribuida

La carga distribuida, al contrario que la puntual, abarca una longitud mayor y por tanto la simplificación de considerarla en un único punto no es posible. Pertenece a este tipo de cargas una cisterna con líquido o pulverulentos, un sinfín orientado horizontalmente, la propia estructura del camión, etc. Este tipo de carga está compuesta por dos parámetros fundamentales que son un valor neto y una longitud sobre la que se debe aplicar esa misma carga.

La forma de representar este tipo de cargas es mediante múltiples flechas cubriendo la longitud sobre la que la carga se aplica como se ve en la Figura 2.9.

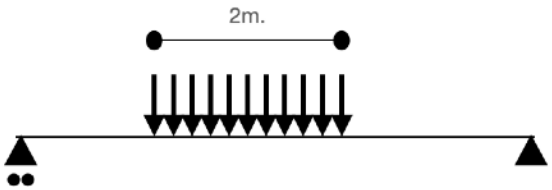


Figura 2.9. Representación de carga distribuida en una longitud de 2 metros, sobre una viga biapoyada con un apoyo delantero deslizante y un apoyo trasero articulado.

2.4. Conceptos de cálculo de camiones y elementos del mismo

En este apartado se van a exponer las particularidades de los proyectos de carrozado de camiones especialmente orientadas a los requerimientos que se exigen para el cálculo de sus estructuras y sus elementos.

TARA

La TARA es el peso máximo del vehículo sin contar las cargas, es decir, sin tener en cuenta conductores, pasajeros, carga, combustible, agua, ni herramientas u otros accesorios (Michelin, 2021).

M.M.A.

La MMA o Peso Maximo Autorizado, es el peso máximo del camión contabilizando además las cargas que puede transportar (Michelin 2021).

Carga útil

La carga útil es la diferencia entre la Masa Máxima Autorizada y la TARA del mismo (Michelin 2021).

Carga máxima por eje

En España, el Ministerio de Transportes, Movilidad y Agenda Urbana establece un peso máximo por eje en función de diferentes parámetros y de si se trata de un eje simple o doble. La Tabla 2.1 expone los valores máximos permitidos (Real Decreto 2822/1998). Estos pesos pueden variar por país o evolucionar con nuevas actualizaciones de las normativas existentes.

		Toneladas
Eje simple	Eje motor	11,5
	Eje motor de los vehículos de la clase I (autobuses urbanos), según la clasificación de la Directiva 2001/85/CE, de 20 de noviembre	13
	Eje motor de los vehículos de las clases II y III (autobuses interurbanos), según la clasificación de la Directiva 2001/85/CE de 20 de noviembre	12,6
	Eje no motor	10
Eje Tandem	Si la separación «d» de dos ejes es inferior a 1,00 metros (d < 1,00 m)	11,5
	Si es igual o superior a 1,00 metros e inferior a 1,30 metros (1,00 m ≤ d < 1,30 m)	16
	Si es igual o superior a 1,30 metros e inferior a 1,80 metros (1,30 m d < 1,80 m)	18
	En el caso anterior si el eje motor va equipado con neumáticos dobles y suspensión neumática o reconocida como equivalente a escala comunitaria, o cuando cada eje motor esté equipado con neumáticos dobles y la masa máxima de cada eje no excede de las 9,5 toneladas	19
	Eje tándem de los remolques o semirremolques	
	Si la separación «d» de los ejes es inferior a 1,00 metros (d < 1,00 m)	11
	Si es igual o superior a 1,00 metros e inferior a 1,30 metros (1,00 ≤ d < 1,30 m)	16
	Si es igual o superior a 1,30 metros e inferior a 1,80 metros (1,30 m ≤ d < 1,80 m) (1)	18
	Si es igual o superior a 1,80 metros (1,80 m ≤ d)	20
	Tándem triaxial de los remolques o semirremolques	
	Si la distancia es igual o inferior a 1,30 metros (d ≤ 1,30 m)	21
	Si la distancia es superior a 1,30 metros e inferior o igual a 1,40 metros (1,30 < d ≤ 1,40 m)	24

Tabla 2.1. Pesos de eje máximos autorizados por el ministerio de transportes, movilidad y agenda urbana

La masa por eje es la masa que gravita sobre el suelo transmitida por la totalidad de las ruedas del mencionado eje. Este concepto se aplica normalmente en el caso de los camiones, debido a que estos son los que presentan una mayor actividad en el tráfico de mercancía pesada.

La limitación de la masa máxima por eje es importante ya que superar esas cifras puede suponer tanto un peligro para el propio vehículo, al rebasar la capacidad portante del eje, como al estado del firme de las carreteras.

Cada país define sus propias restricciones en cuanto a los límites de masa máxima por eje (onroad 2023).

Estructura del vehículo

La estructura portante el vehículo de transporte se compone normalmente de dos vigas paralelas, con una sección de tipo C o tipo I, sobre las que apoyan y se acoplan la carrocería y diferentes elementos del camión. Estas dos vigas apoyan sobre los ejes en los distintos puntos donde estos se distribuyen, como se muestra en la Figura 2.10.

Estas dos estructuras tienen, como su nombre indica, forma de 'C' o de 'I'. En el primer caso, el alma o zona vertical de la viga está descentrada sobre las alas (o partes horizontales de la viga) inferior y superior, mientras que en el caso de la 'I', el alma cae en el centro de la sección. Ambos tipos de sección se muestran en el Capítulo 5, al comentar como se calculan dichas secciones.

Figura 2.10. Chasis de un camión donde se representan las dos vigas en C apoyadas sobre un eje simple y un tándem doble (Imagen libre de copyright de la web (pngwing).

Simplificaciones comúnmente aceptadas en la ITV para aprobación

La estructura portante del vehículo de transporte se compone normalmente de dos vigas paralelas conectadas por viguetas transversales.

Cuando se necesita realizar el carrozado del camión, se necesita justificar en una ITV que en ningún caso se está superando la carga máxima autorizada de cada uno de los ejes. Asimismo, se debe justificar que la altura y espesor de la viga sobre la que se apoyarán los elementos del carrozado será capaz de soportar el peso correspondiente sin sufrir deformación ni alcanzar una situación donde se supere el límite elástico del acero elegido.

Para ello se realiza la siguiente simplificación sobre la estructura del chasis para poder conseguir aplicar los cálculos correspondientes:

1) Se pasa de un sistema tridimensional a un sistema bidimensional. Para ello se utiliza una única viga para el cálculo en vez de dos. Posteriormente de realizar los cálculos, cuando obtengamos la tensión final sobre la viga utilizada, dividiremos dicha tensión entre dos para tener en cuenta este hecho.
2) Se sustituyen los ejes y tándems por apoyos. Veremos más adelante las variaciones que se aplican al pasar de un camión rígido a un camión articulado. Cuando obtengamos las reacciones sobre cada uno podremos dividir entre el número de ejes para cada tándem la carga.
3) Se define la sección de las vigas a utilizar para calcular su momento de inercia, sección y módulo resistente.
4) Se utiliza un apoyo articulado y un apoyo deslizante para tener solo 3 incógnitas que resolver. Dichas incógnitas son las dos reacciones verticales (una de cada apoyo) y la reacción horizontal del apoyo articulado.
5) Se aplican las cargas puntuales sobre la estructura en los puntos donde se encontrarán ubicadas.
6) Se aplican las cargas distribuidas sobre la estructura en los puntos donde se encontrarán ubicadas.
7) Se calculan las reacciones sobre los ejes y se comprueba que no se ha producido en ningún caso un exceso sobre las máximas autorizadas.
8) Se calcula con ellos el diagrama de esfuerzos cortantes y se encuentra el máximo esfuerzo cortante para calcular la tensión cortante.

9) Se calcula el diagrama de momentos flectores y se encuentra el máximo momento flector para calcular la tensión normal. En este caso se ignora la carga de tracción pura ya que no se consideran cargas aplicadas en la dirección de las vigas sino perpendicularmente a ellas.
10) Se calculan las tensiones de Von Mises para comprobar que no se ha sobrepasado el límite elástico sobre el material seleccionado y que por tanto el sistema es capaz de soportar las cargas sin llegar a deformación plástica o permanente.

Una vez terminados estos pasos puede emitirse un informe a la ITV en el que se indica la idoneidad del vehículos para realizar las funciones que debe acometer. La ITV revisará las informaciones suministradas así como el cálculo y validará en su caso las modificaciones que se van a aplicar sobre la estructura al realizarse el carrozado.

Cálculo de camiones rígidos

La realización de un cálculo de camiones rígidos es más sencilla que en el camión articulado, ya que en el caso de camiones rígidos contemplamos únicamente una estructura rígida que normalmente consta de un eje o tándem que se convertirá en un apoyo delantero y otro eje o tándem que se convertirán en un apoyo trasero.

El método simplificado explicado en el punto anterior es por tanto, aplicable directamente y no necesita de operaciones adicionales. Se expone en la Figura 2.11 un caso de cálculo en esta sección de un camión rígido. En él se aprecia una carga de 1000 kgf (kilogramos-fuerza) situada a 1000 mm del frente del camión.

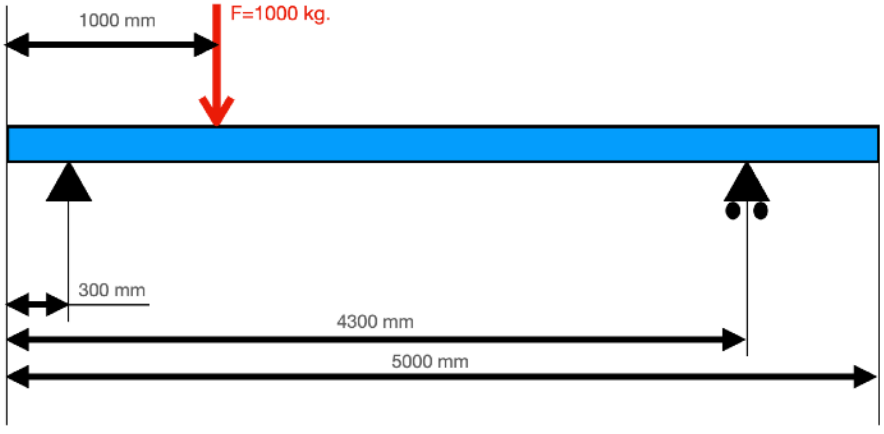


Figura 2.11. Calculo de un camión de 5000 mm con una carga de 1000 kgf a 1000 mm del frente.

Para el cálculo, se realiza la simplificación por medio de la sustitución de los apoyos mencionados anteriormente por las reacciones que generan, obteniéndose el sistema de la Figura 2.12.

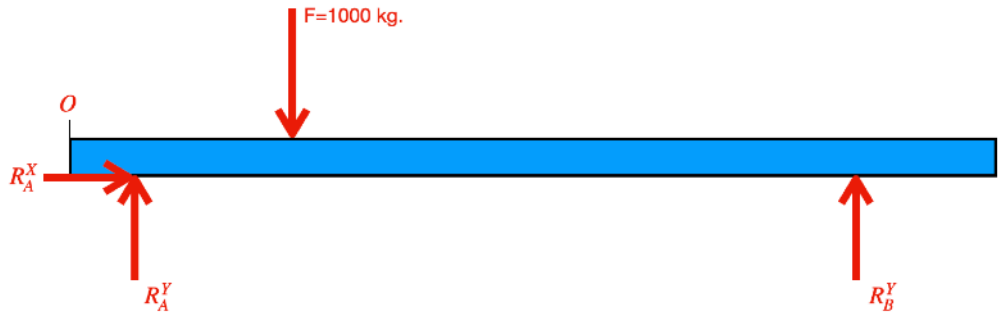


Figura 2.12. Inclusión de las reacciones

A continuación, se realiza la preparación y resolución de los sistemas de ecuaciones (sumatorios de fuerzas y sumatorio de momentos desde el origen "O", como se expone en la Figura 2.13.

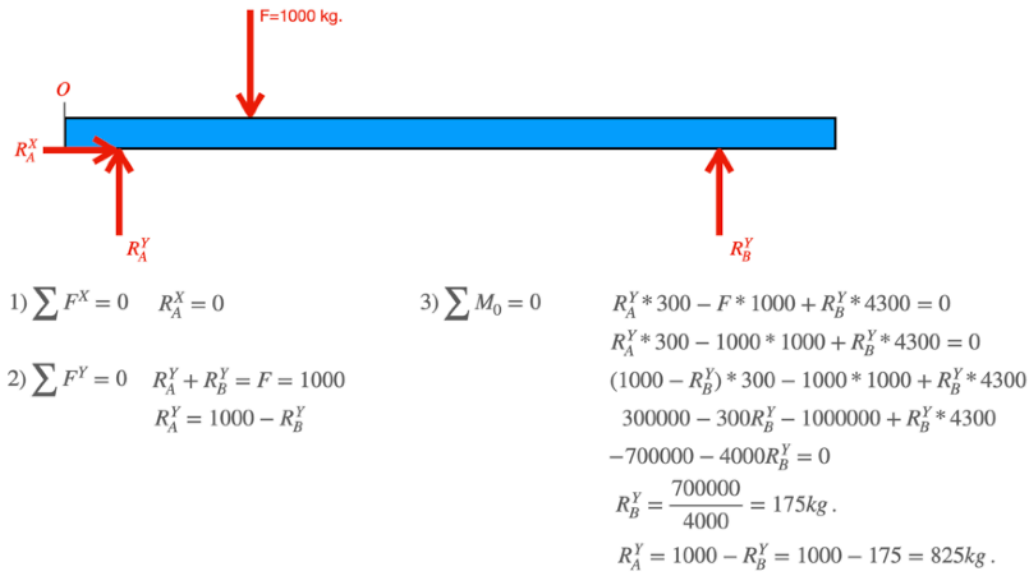


Figura 2.13. Resolución del problema.

Obteniendo las reacciones en X y en Y de A (la reacción en el eje X será 0) y la reacción en Y de B.

Cálculo de camiones articulados

La realización de un cálculo de camiones articulados revierte algo más de complejidad si bien el proceso es parecido y únicamente se necesita aplicar alguna simplificación adicional.

Un camión articulado consta de una cabeza tractora (que a efectos de cálculo podría considerarse como un camión rígido) al que se engancha un remolque que es arrastrado por la misma.

La cabeza tractora dispone de una zona específica en la que se debe realizar la conexión del remolque por medio del King Pin. Éste elemento es un vástago que se encaja en dicha zona de la cabeza tractora.

A la hora de realizar el cálculo, hablamos prácticamente de dos cálculos de camión rígido, el primero sería el correspondiente a la cabeza tractora y el segundo el correspondiente al remolque.

La simplificación para realizar estos cálculos es la siguiente:

1) Se realiza el cálculo del remolque con las cargas que le apliquen, realizando un apoyo trasero correspondiente al eje o tándem de la parte trasera del remolque y un apoyo delantero correspondiente al King Pin.
2) Se realiza el cálculo sobre la cabeza tractora con las cargas que le apliquen, realizando los apoyos como en el caso del camión rígido pero incluyendo una carga puntual más, que corresponderá a la reacción del King Pin que se ha calculado en la simplificación anterior.

De esta forma, podemos realizar el cálculo de un elemento que, considerado en su conjunto, implicaría hasta 6 apoyos cada uno con hasta 3 grados de libertad, por lo que el sistema se volvería hiperestático y requeriría una complejidad mucho mayor de lo requerido para su validación en el organismo competente.

Se muestra un ejemplo en la Figura 2.14. La barra azul representa a la cabeza tractora mientras que la barra roja representa al remolque. La pieza amarilla representa el king pin.

Se procede primero con la parte del remolque, sustituyendo el king pin por un apoyo articulado y obteniendo el sistema de la figura 2.15. A continuación resolvemos la parte del remolque como se expone en la Figura 2.16.

Una vez hemos obtenido la reacción del king pin, aplicamos su carga sobre la cabeza tractora tal como se muestra en la Figura 2.17. Finalmente resolvemos el sistema generado, como se expone en la Figura 2.18.

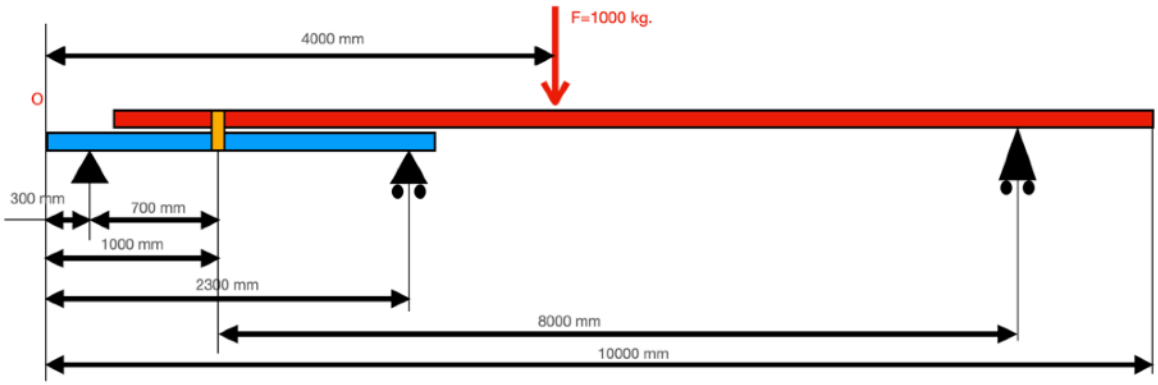


Figura 2.14. Representación de un camión articulado de 10000 mm de longitud con cabeza tractora (azul) y remolque(rojo) unidos por un king pin(amarillo) con una carga de 1000 kgf a 4000 mm del frente.

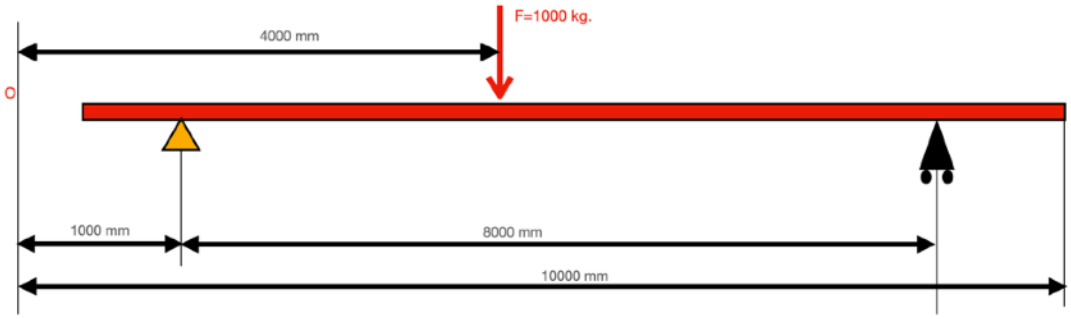
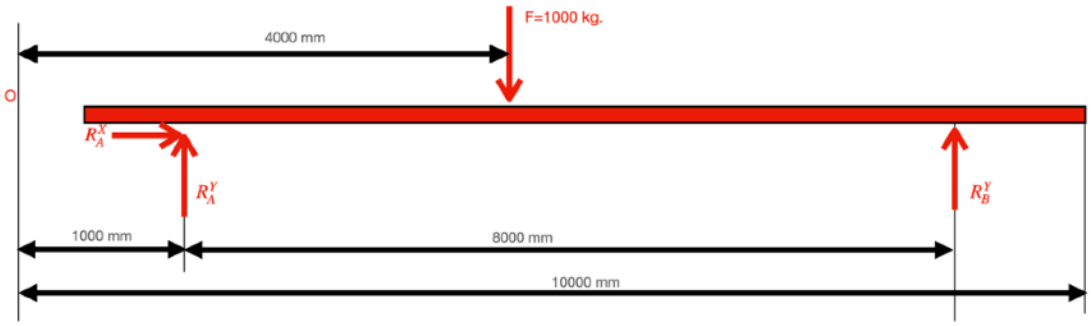


Figura 2.15. Esquema de apoyos de la parte del remolque



$$1)\ \sum F^X = 0 \quad R_A^X = 0$$

$$2)\ \sum F^Y = 0 \quad R_A^Y + R_B^Y = F = 1000$$
$$R_A^Y = 1000 - R_B^Y$$

$$3)\ \sum M_0 = 0 \quad R_A^Y * 1000 - F * 4000 + R_B^Y * 9000 = 0$$
$$R_A^Y * 1000 - 1000 * 4000 + R_B^Y * 9000 = 0$$
$$(1000 - R_B^Y) * 1000 - 1000 * 4000 + R_B^Y * 9000$$
$$1000000 - 1000R_B^Y - 4000000 + R_B^Y * 9000$$
$$-30000000 + 8000R_B^Y = 0$$
$$R_B^Y = \frac{3000000}{8000} = 375kg\,.$$
$$R_A^Y = 1000 - R_B^Y = 1000 - 375 = 625kg\,.$$

Figura 2.16. Resolución del sistema de ecuaciones y obtención de la carga en B (Eje del remolque) y en A (king pin).

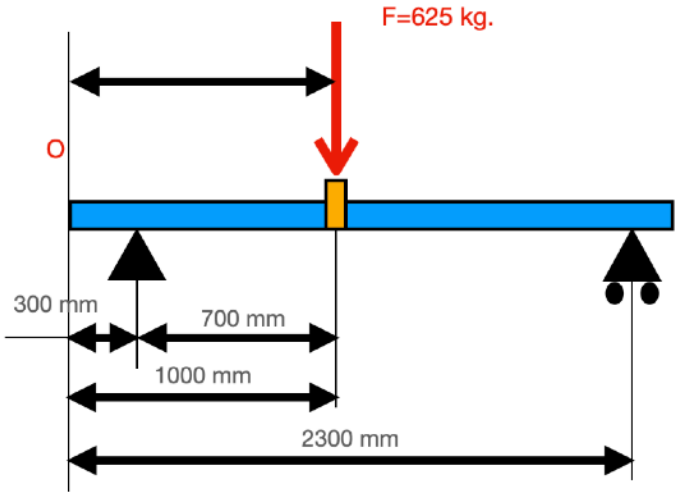
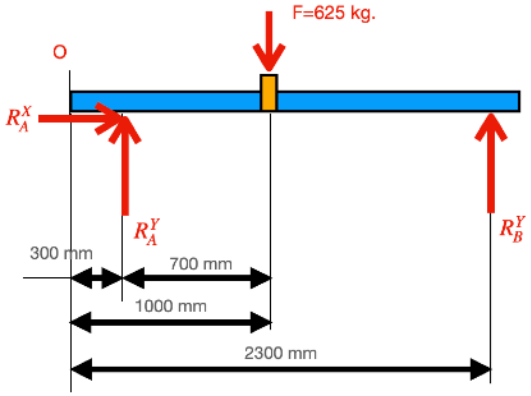


Figura 2.17. Aplicación de la carga del king pin sobre la cabeza tractora.



$$1)\ \sum F^X = 0 \quad R_A^X = 0$$

$$2)\ \sum F^Y = 0 \quad R_A^Y + R_B^Y = F = 625$$

$$R_A^Y = 625 - R_B^Y$$

$$3)\ \sum M_0 = 0 \quad R_A^Y * 300 - F * 1000 + R_B^Y * 2300 = 0$$

$$R_A^Y * 300 - 625 * 1000 + R_B^Y * 2300 = 0$$

$$(625 - R_B^Y) * 300 - 625 * 1000 + R_B^Y * 2300$$

$$187500 - 300R_B^Y - 625000 + R_B^Y * 2300$$

$$-437500 + 2000R_B^Y = 0$$

$$R_B^Y = \frac{437500}{2000} = 218{,}75kg\,.$$

$$R_A^Y = 625 - R_B^Y = 625 - 218{,}75 = 406{,}25kg\,.$$

Figura 2.18. Resolución del sistema de ecuaciones y obtención de cargas en los ejes.

2.5. Soluciones existentes

El cálculo de este tipo de elementos es algo que suele ser muy específico para el país, aplicación, marca de camión, etc. No existen, por esta razón, demasiadas soluciones expresamente

desarrolladas para este propósito y cuando lo están, suelen estar orientadas a la captación de clientes que posteriormente faciliten la contratación del propio trabajo de carrozado en sus propias instalaciones.

Por otra parte, las soluciones existentes suelen estar orientadas a vehículos comerciales, por lo que particulares y pequeñas empresas suelen tener dificultades en encontrar algo adecuado a sus necesidades.

A continuación se exponen algunas de las que se han encontrado disponibles para este propósito.

NOVAB

Tal como se indicaba anteriormente, Nooteboom es el caso de una empresa que realiza carrozados y que, por tanto, ofrece a sus clientes el servicio adicional de poder realizar cálculos o evaluaciones.

Según la información de la propia empresa en su página web: "Nooteboom Trailers B.V., un negocio familiar fundado en 1881, es una organización internacional que ofrece a sus clientes soluciones completas en el campo del transporte especial. Nooteboom diseña y construye remolques de la más alta calidad con una carga útil de 20 a 200 toneladas. Nooteboom se esfuerza continuamente por mejorar mediante una continua inversión en las mejores soluciones para sus clientes. Pensando y trabajando realmente de una manera pionera. Ya sea en lo relativo a la calidad de sus productos, el soporte ofrecido con el concepto de servicio para el ciclo de vida o las muchas soluciones innovadoras con las que Nooteboom marca tendencia una y otra vez" (Noteboom 2003).

Según indican en su página web, ha desarrollado un software para el cálculo de la carga máxima en ejes y según la empresa "el programa calcula con precisión la forma óptima de colocar las cargas en todas las combinaciones de vehículos" (Noteboom 2003). Por otra parte "Los usuarios del programa pueden elegir entre una gran variedad de vehículos predefinidos tales como camiones con caja, cabezas tractoras, remolques con lanza y semirremolques" (Noteboom 2003).

La empresa indica también "Los usuarios pueden optar por una carga predefinida o introducir los datos de una carga. Basándose en toda esta información, NoVAB calcula la carga por eje y la mejor distribución posible de la carga por eje" (Noteboom 2003).

En cuanto a las ventajas que la propia empresa indica: "NoVAB 3.0 tiene de una extensa biblioteca de camiones, cabezas tractoras, remolques y cargas a su disposición" (Noteboom 2003).

Esta aplicación tiene, desde el punto de vista de partes interesadas, las siguientes posibles ventajas:

- Multiplataforma: Ordenadores de sobremesa, portátiles, tabletas y teléfonos inteligentes.
- Las combinaciones de vehículos y las cargas se actualizan continuamente.

- Multilingüe: holandés, inglés, alemán, francés, español, polaco, italiano.

En cuanto a los inconvenientes de la aplicación, se mencionan los siguientes:

- Se ciñe a vehículos predefinidos por la empresa y por tanto no existe posibilidad de adaptar totalmente a las configuraciones de algunos usuarios.
- Su utilización es web, por lo que no es posible su uso en caso de no disponer de conexión.

TRAILERWIN

TrailerWin es un software creado por la empresa finlandesa Trailer Consultation, expresamente focalizada en la realización de software para el soporte al transporte. Pertenece a una colección de herramientas entre las que se encuentran las siguientes opciones:

- TrailerWin: Software para construcción y carrozado de camiones.
- CraneWin: Software para el cálculo de la estabilidad de las gruas en camiones y trailers.
- FrameWin: Software para el cálculo de la subestructura y calculo de tensión para las grúas.
- CornerWin: Software para el cálculo del giro.
- BrakeWin: Software para el cálculo teórico del frenado en trailers.
- DrivelineWin: Software para el cálculo de velocidad y tiro en distintos tipos de motores, cajas de reducción y neumáticos.
- BusWin: Software de cálculo de giros y simulación para autobuses.

Este software está operativo desde 1982 y sus primeras versiones estaban realizadas para el Commodore 64 (TrailerWin 2023).

Esta aplicación tiene, desde el punto de vista de partes interesadas, las siguientes posibles ventajas:

- 40 años de experiencia sobre el sector.
- Varias sub-aplicaciones, que conforman un abanico de posibilidades amplio para la industria del carrozado.
- Exportación de planos a software CAD. Este punto es especialmente importante a la hora de realizar la construcción del carrozado.

En cuanto a los inconvenientes de la aplicación, se mencionan los siguientes:

- Sólo se encuentra la versión bajo Sistema Operativo Windows y en idioma inglés.
- La estética de la aplicación, así como la facilidad de uso, resulta bastante descuidada con respecto a lo que a día de hoy suele utilizarse.

2.6. Conclusión

El carrozado de camiones es un campo donde el número de modificaciones que pueden llegarse a hacer sobre el elemento de transporte es ilimitado. Cualquiera de estas modificaciones debe implicar seguridad tanto para el conductor como para los usuarios de las vías de transporte, por lo que su diseño debe estar completamente analizado y testado.

Los profesionales implicados en el cálculo del carrozado son en su mayoría ingenieros, ya que entender los datos a introducir y los resultados de los cálculos necesitan de estos conocimientos.

En este capítulo se ha expuesto una parte mínima de la necesaria para el dominio del cálculo de estructuras. Se ha tenido muy en cuenta para el desarrollo de la aplicación todos los conceptos aquí introducidos y se ha prestado especial atención a los cálculos que la aplicación debe realizar y a qué resultados son más importantes a la hora de su presentación.

Tal como se ha comentado en este capítulo, se ha hablado especialmente de masa máxima autorizada sobre ejes y de tensión sobre las vigas que conforman la estructura, por lo que estos dos resultados han sido cuidadosamente analizados en el desarrollo.

Se ha analizado la disponibilidad de aplicaciones y se han revisado las carencias existentes actualmente en las mismas, siendo en su mayoría relativas a la falta de adaptación a vehículos específicos del cliente, ausencia de disponibilidad de Sistema Operativo o facilidad de uso.

Capítulo 3. Análisis

3.1. Introducción

Se pretende en este capítulo realizar el análisis de los requerimientos. Se utilizarán los casos de uso que describirán la forma de interacción de los actores con el sistema y permitirán analizar las funciones que el sistema deberá proveer para un correcto funcionamiento.

Los requerimientos tendrán también en cuenta las simplificaciones y consideraciones que se mencionaron en el Capítulo 2 para poder ser congruentes con los requerimientos de otras partes interesadas como puede ser la ITV o los fabricantes de vehículos y carroceros.

3.2. Casos de uso

Se describen a continuación los casos de uso que describen la interacción de los actores con el sistema (Pressman y Maxim, 2020).

CU1: Introducción de datos

Actor: Usuario de la aplicación.

- El usuario abre la aplicación.
- El sistema muestra la pantalla con los datos a introducir.
- El usuario introduce los datos genéricos del camión como cliente, bastidor, matrícula.
- El usuario selecciona el tipo de camión a utilizar (rigido/articulado) y el sub-tipo (ejes por tándem).

- El sistema presenta la información específica del tipo de camión seleccionado.
- El usuario rellena los datos del camión.
- El usuario valida los datos introducidos.
- El sistema valida los datos o muestra un mensaje de error en caso de no ser correctos.

CU2: Importa camión de base de datos

Actor: Usuario de la aplicación.

- El usuario selecciona importar de base de datos.
- El sistema muestra la lista de camiones existentes en la base de datos.
- El usuario selecciona el camión a importar.
- El usuario confirma la importación.
- El sistema importa los datos del camión y rellena los campos necesarios.
- El sistema cierra la ventana de importación.

CU3: Exporta camión a base de datos

Actor: Usuario de la aplicación.

- El usuario selecciona exportar a base de datos.
- El sistema exporta los datos del camión a la base de datos.

CU4: Modifica camión de base de datos

Actor: Usuario de la aplicación.

- El usuario selecciona la base de datos.
- El sistema muestra la ventana de la base de datos.
- El usuario selecciona un parámetro del camión.
- El sistema muestra el parámetro en una casilla.
- El usuario modifica el valor en la casilla.
- El usuario confirma la modificación.
- El sistema actualiza la tabla.
- El sistema actualiza la base de datos.

CU5: Cambio de unidades

Actor: Usuario de la aplicación.

- El usuario selecciona cambiar de unidades.
- El sistema muestra las posibilidades de unidades para seleccionar.
- El usuario selecciona las unidades a utilizar.
- El sistema actualiza la información para usar las unidades seleccionadas.

CU6: Información

Actor: Usuario de la aplicación.

- El usuario solicita ayuda específica sobre la aplicación.
- El sistema presenta las posibles informaciones disponibles.
- El usuario solicita la información deseada.
- El sistema presenta la información.

CU7: Determinar sección de viga

Actor: Usuario de la aplicación.

- El usuario selecciona la opción de trabajar con la sección.
- El sistema muestra una ventana específica para el trabajo con la sección.
- El usuario introduce los datos necesarios para la descripción de la sección.
- El usuario solicita las propiedades para los datos introducidos.
- El sistema muestra la información de los datos calculados.
- El usuario asigna la sección a la viga del camión.
- El sistema confirma la asignación o muestra un mensaje de error.

CU8: Importa sección de viga

Actor: Usuario de la aplicación.

- El usuario selecciona la opción de trabajar con la sección.
- El sistema muestra una ventana específica para el trabajo con la sección.
- El usuario selecciona importar de Base de Datos.

- El sistema muestra las posibles secciones de la base de datos.
- El usuario selecciona la viga a importar.
- El usuario confirma la importación.
- El sistema importa la viga y rellena los datos correspondientes.
- El sistema cierra la ventana de base de datos.

CU9: Modifica sección de viga

Actor: Usuario de la aplicación.

- El usuario selecciona la opción de trabajar con la sección.
- El sistema muestra una ventana específica para el trabajo con la sección.
- El usuario selecciona bases de datos.
- El sistema muestra una ventana con las vigas existentes.
- El usuario selecciona una propiedad de una viga.
- El sistema muestra el valor en una casilla.
- El usuario modifica el valor en la casilla.
- El usuario valida el nuevo valor.
- El sistema actualiza la información en la tabla.
- El sistema actualiza la información en la base de datos.

CU10: Introducción de cargas

Actor: Usuario de la aplicación.

- El usuario selecciona el trabajo con cargas si no está en el menú.
- El sistema muestra la ventana específica de cálculo con la información esquemática del camión.
- El usuario selecciona introducir carga.
- El sistema presenta la información necesaria para la introducción de las cargas.
- El usuario rellena la información y valida la carga.
- El sistema comprueba que el nombre de la carga no sea duplicado y valida la introducción.
- El sistema presenta la carga en la lista de cargas.
- El sistema presenta la carga en el esquema del camión.
- El sistema presenta la variación de los esfuerzos cortantes.
- El sistema presenta la variación de los momentos flectores.

- El sistema evalúa las reacciones en los ejes y las presenta avisando si se sobrepasan los máximos.
- El sistema presenta la tensión máxima de la viga.

CU11: Visualización de cargas

Actor: Usuario de la aplicación.

- El usuario solicita el trabajo con cargas si no está en el menú.
- El usuario solicita visualizar cargas.
- El sistema presenta un menú para seleccionar las cargas que se quieren visualizar.
- El usuario selecciona la carga a visualizar o a ocultar.
- El sistema presenta u oculta la carga en el esquema del camión.

CU12: Eliminación de cargas

Actor: Usuario de la aplicación.

- El usuario solicita el trabajo con cargas si no está en el menú.
- El usuario solicita eliminar cargas.
- El sistema presenta un menú para seleccionar las cargas que se desean eliminar.
- El usuario selecciona la carga a eliminar.
- El sistema elimina la carga en la lista de cargas.
- El sistema elimina la carga en el esquema del camión.
- El sistema presenta la variación de los esfuerzos cortantes.
- El sistema presenta la variación de los momentos flectores.
- El sistema evalúa las reacciones en los ejes y las presenta avisando si se sobrepasan los máximos.
- El sistema presenta la tensión máxima de la viga.

CU13: Visualizar zona de cálculo

Actor: Usuario de la aplicación.

- El usuario solicita el trabajo con cargas si no está en el menú.
- El sistema desbloquea la opción de visualizar cabeza tractora o remolque si es un vehículo articulado.
- El usuario selecciona la parte con la que se desea trabajar.

- El sistema actualiza el esquema.
- El sistema presenta la variación de los esfuerzos cortantes para la zona seleccionada.
- El sistema presenta la variación de los momentos flectores para la zona seleccionada.
- El sistema evalúa las reacciones en los ejes y las presenta avisando si se sobrepasan los máximos.
- El sistema presenta la tensión máxima de la viga.

CU14: Guardar archivo

Actor: Usuario de la aplicación.

- El usuario selecciona la opción de guardar archivo de datos del proyecto.
- El sistema muestra la información del sistema de archivos.
- El usuario selecciona la ubicación de guardado.
- El usuario introduce un nombre para el archivo.
- El usuario confirma el guardado.
- El sistema guarda la información del archivo con el nombre dado en la ubicación seleccionada.

CU15: Importar archivo

Actor: Usuario de la aplicación.

- El usuario selecciona la opción de cargar archivo.
- El sistema muestra la información del sistema de archivos.
- El usuario selecciona la ubicación del archivo a cargar.
- El usuario selecciona el archivo a cargar.
- El usuario confirma la carga.
- El sistema elimina la información existente en el sistema.
- El sistema carga la información y la presenta en la pantalla principal.

CU16: Elimina información

Actor: Usuario de la aplicación.

- El usuario selecciona borrar datos.
- El sistema elimina todos los datos del proyecto.

CU17: Exportar a PDF los resultados

Actor: Usuario de la aplicación.

- El usuario selecciona exportar los datos a un anexo PDF para su presentación a la ITV.
- El sistema muestra una ventana con el nombre a utilizar.
- El usuario introduce el nombre del archivo pdf.
- El sistema prepara el archivo pdf y lo guarda con el nombre dado por el usuario.

3.3. Modelado de clases

A continuación se muestran las clases potenciales derivadas del análisis de los requisitos (Gómez 2019).

CP1: Camión

Descripción: La clase Camión se encargaría de recoger la información sobre el camión. Incluye informaciones generales del camión que permiten tanto la descripción geométrica como valores necesarios para el cálculo del mismo y sus resultados.

Atributos:

- Matrícula: Referencia la matricula que identifica el camión.
- Bastidor: Referencia el bastidor que identifica la estructura del camión.
- Dimensiones: Datos geométricos del camión.
- Sección de viga: Sección utilizada para el camión.
- Tensión: Considera la tensión que el proyecto ha calculado.
- Cargas: Recoge la lista de cargas que se han aplicado para el proyecto.
- Tipo: Se definen dos tipos
  - Rígido.
  - Articulado.

CP2: Carga

Descripción: La clase Carga se encargaría de recoger la información sobre las características de las cargas puntuales o distribuidas que se aplican sobre el camión.

Atributos:

- Nombre: Hace referencia al nombre que se le da a la carga para describir su origen.
- Tipo: Recoge el tipo de carga que es.
  - Puntual.
  - Distribuida
- Punto: Contendrá el punto donde se aplicará la carga.
- Valor: Contendrá la magnitud de la carga a aplicar.

CP3: Sección

Descripción: La clase Sección se encarga de las secciones resistentes.

Atributos:

- Area: Contiene la información del Area.
- Inercia: Contiene la información de la Inercia.
- Tipo: Contiene la tipología de la sección.
  - Tipo I.
  - Tipo C.

CP4: Esquema

Descripción: La clase Esquema se encargará de la gestión del esquema del camión y su presentación al usuario.

Atributos:

- Escala: Contendrá la información de la escala a la que se debe presentar el camión para una visualización adecuada.
- Dimensiones: Contendrá las dimensiones adaptadas a la escala.

CP5: Cortantes

Descripción: La clase cortantes se encargaría del cálculo de las cargas cortantes y de la presentación de las mismas al usuario.

Atributos:

- Escala: Contendrá la información de la escala a la que se debe presentar las gráficas.
- Dimensiones: Contendrá la información de las dimensiones adaptadas a la escala.
- Puntos cálculo: Contendrá una serie de puntos en los que se deberá calcular el cortante para poder presentar una gráfica adecuada.
- Carga máxima: Contendrá la información de la carga cortante máxima sobre la viga.

CP6: Momentos flectores

Descripción: La clase momentos flectores se encargaría del cálculo de los momentos y de la presentación de los mismos al usuario.

Atributos:

- Escala: Contendrá la información de la escala a la que se debe presentar las gráficas.
- Dimensiones: Contendrá la información de las dimensiones adaptadas a la escala.
- Puntos cálculo: Contendrá una serie de puntos en los que se deberá calcular el momento para poder presentar una gráfica adecuada.
- Carga máxima: Contendrá la información del momento máximo sobre la viga.

CP7: Reacciones

Descripción: La clase Reacciones se encargaría del cálculo de las reacciones y de la presentación de las mismos al usuario.

Atributos:

- Reacción A: Contendrá el valor de la reacción en el lado A.
- Reacción B: Contendrá el valor de la reacción en el lado B.

CP8: Base de datos de camiones

Descripción: Esta clase se encargará de la gestión de la base de datos de camiones.

Atributos:

- Driver: Tipo de base de datos a utilizar.
- Conexión: Conexión a utilizar para la gestión.
- Nombre: Nombre de base de datos.

CP9: Base de datos de secciones

Descripción: Esta clase se encargará de la gestión de la base de datos de secciones.

Atributos:

- Driver: Tipo de base de datos a utilizar.
- Conexión: Conexión a utilizar para la gestión.
- Nombre: Nombre de base de datos.

3.4. Diagrama de modelo de dominio

Se adjunta en la Figura 3.1 el Diagrama de modelo de dominio (Microsoft 2009).

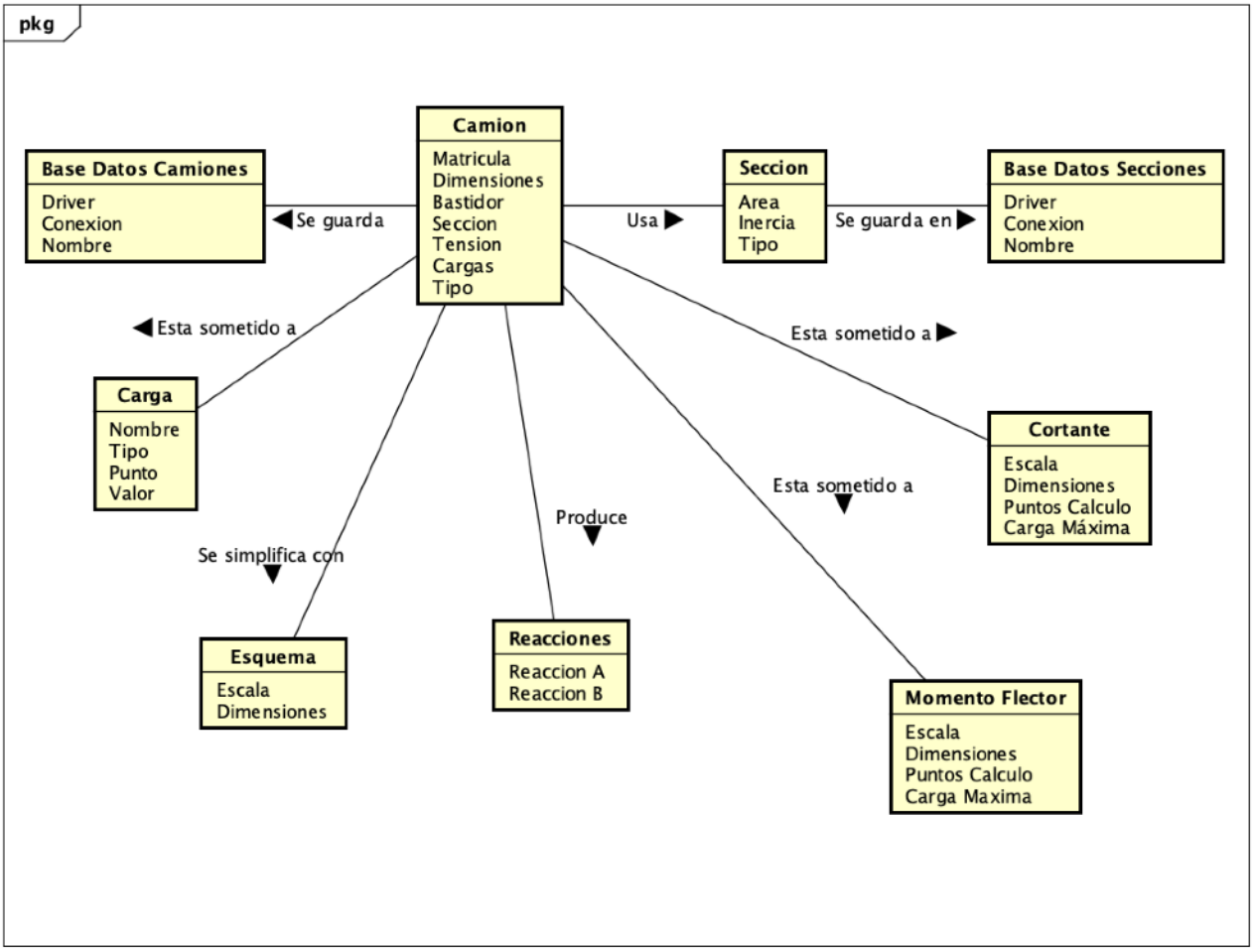


Figura 3.1. Diagrama de dominio.

3.5. Definición de requisitos

Mediante el estudio de los requisitos definiremos las especificaciones mínimas para las que el sistema debe ser capaz de proveer un cumplimiento. Estos requisitos constituyen las necesidades mínimas para las partes interesadas, por lo que deberán ser cuidadosamente recopilados para posteriormente servir de base para el diseño de la aplicación.

3.5.1. Requisitos funcionales

Se realiza a continuación una exposición de los requisitos funcionales (Gómez 2019):

- **Requisito Funcional 1. Gestión de proyectos.**
  - RF1.1. El sistema debe permitir el control de la información fundamental para la descripción de un proyecto.
  - RF1.2. El sistema debe ser capaz de gestionar los datos geométricos del camión.
  - RF1.3. El sistema debe ser capaz de distinguir las casuística de tipologías de camión.
  - RF1.4. El sistema debe ser capaz de adaptar las unidades necesarias.
- **Requisito Funcional 2. Almacenamiento de proyectos.**
  - RF2.1. El sistema debe ser capaz de almacenar y recuperar información de proyectos.
- **Requisito Funcional 3. Gestión de bases de datos.**
  - RF3.1. El sistema debe ser capaz de almacenar información de camiones en una base de datos específica de camiones.
  - RF3.2. El sistema debe ser capaz de recuperar información de camiones desde una base de datos de datos específica de camiones.
  - RF3.3. El sistema debe ser capaz de modificar la información existente de camiones de la base de datos específica de camiones.
  - RF3.4. El sistema debe ser capaz de borrar la información existente de camiones de la base de datos específica de camiones.
  - RF3.5. El sistema debe ser capaz de almacenar información de secciones de acero en una base de datos específica de secciones de acero.
  - RF3.6. El sistema debe ser capaz de recuperar información de secciones de acero desde una base de datos de datos específica de secciones de acero.

  - RF3.7. El sistema debe ser capaz de modificar la información existente de secciones de acero de la base de datos específica de secciones de acero.
  - RF3.8. El sistema debe ser capaz de borrar la información existente de secciones de acero de la base de datos específica de secciones de acero.
- **Requisito Funcional 4. Ayuda con secciones.**
  - RF4.1. El sistema debe proporcionar ayuda para el cálculo de secciones resistentes.
- **Requisito Funcional 5. Gestion del cálculo.**
  - RF5.1. El sistema debe ser capaz de obtener un esquema simplificado del camión a partir de los datos obtenidos.
  - RF5.2. El sistema debe ser capaz de proporcionar un diagrama de esfuerzos cortantes.
  - RF5.3. El sistema debe ser capaz de proporcionar un diagrama de momentos flectores.
  - RF5.4. El sistema debe ser capaz de proporcionar un cálculo de reacciones adaptada a la tipología de camión.
  - RF5.5. El sistema debe ser capaz de indicar las cargas con las que se está trabajando.
  - RF5.6. El sistema debe ser capaz de mostrar las tensiones en las vigas.
  - RF5.7. El sistema debe ser capaz de mostrar avisos si se sobrepasan las cargas máximas de los ejes.
  - RF5.8. El sistema debe ser capaz de dar sugerencias para la definición del camión adaptada a no sobrepasar cargas.
- **Requisito Funcional 6.** Generación de información de resultados.
  - RF6.1. El sistema debe ser capaz de proveer un anexo técnico que pueda incluirse en un informe técnico para su presentación en la ITV en formato pdf.

3.5.2. Requisitos no funcionales

Se realiza a continuación una exposición de los requisitos no funcionales de la aplicación (Gómez 2019).

- **Requisito No Funcional 1.** El sistema debe tener una interfaz sencilla que permita la ejecución rápida del trabajo.
- **Requisito No Funcional 2.** El sistema debe presentar toda la información de cálculo lo más concentrada posible para su mejor y más rápido análisis.
- **Requisito No Funcional 3.** El sistema debe tener una interfaz agradable para el uso.

- **Requisito No Funcional 4.** El sistema debe cuidar la usabilidad para facilitar la rapidez en generación de resultados.
- **Requisito No Funcional 5.** sistema debe presentar claramente los datos finales y dar avisos claros en los casos necesarios.
- **Requisito No Funcional 6.** El sistema debe realizar los cálculos y actualizaciones de forma sencilla.
- **Requisito No Funcional 7.** El sistema debe proporcionar claridad cuando la complejidad del cálculo se produzca.

3.5.3. Actores

Existe para la aplicación un único actor que se presenta a continuación.

- **Actor 1.** Usuario de la aplicación. El usuario de la aplicación es el encargado del análisis del carrozado y quién expondrá los datos para que ésta proporcione el resultado de validez o invalidez del cálculo así como el anexo técnico.

3.5.4. Casos de uso

Se definen los siguientes casos de uso como resultado del análisis de los requisitos funcionales anteriormente analizados (Pressman y Maxim 2020). Estos casos se muestran en el Diagrama de Casos de Uso de la Figura 3.2.

- **Caso de Uso 1.** Crear Camion.

Actor: Usuario.

Escenario principal:

1. El Caso de Uso comienza cuando, una vez abierta la aplicación, el usuario selecciona un tipo de camión.
2. El sistema muestra la información de los datos a introducir en pantalla.
3. El usuario introduce los datos principales del camión en la interfaz mostrada en pantalla.
4. El usuario pulsa guardar datos.
5. El sistema almacena los datos en el camión.

Escenarios alternativos:

4b. El sistema detecta que los datos introducidos no son correctos.

- El sistema muestra un aviso solicitando la corrección de los mismos.

5b. El sistema encuentra algún problema a la hora de almacenar los datos.

- El sistema muestra un aviso sobre el error.

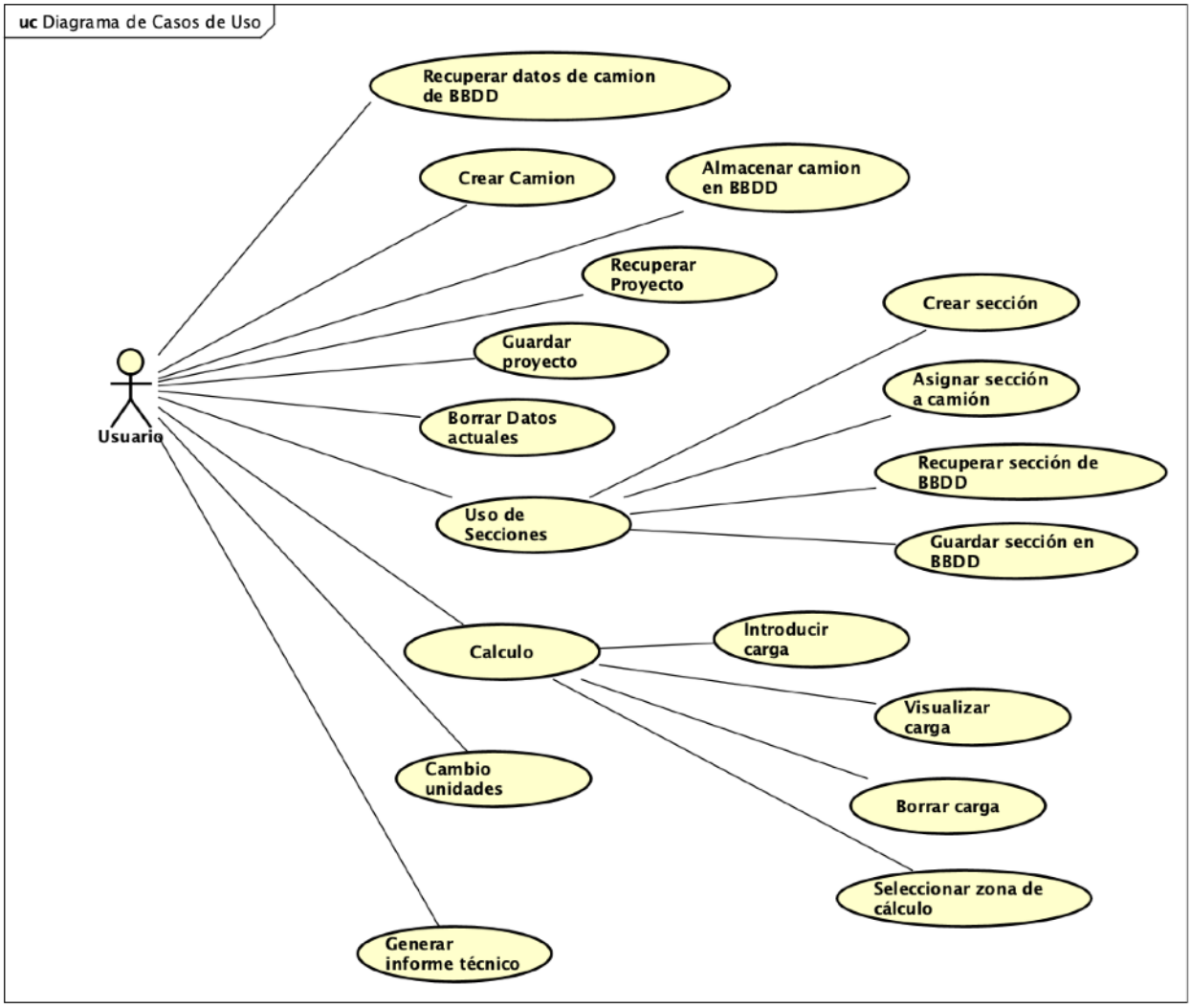


Figura 3.2. Diagrama de casos de uso

• **Caso de Uso 2.** Recuperar datos de BBDD de Camión.

Actor: Usuario.

Escenario principal:

1. El Caso de Uso comienza cuando, una vez abierta la aplicación, el usuario selecciona importar datos de un camión.
2. El sistema muestra la lista de camiones en la base de datos.

3. El usuario selecciona el camión a cargar.
4. El sistema carga los datos del camión en las casillas correspondientes.

Escenarios alternativos:

2b. El sistema detecta un problema con la base de datos.

- El sistema muestra un aviso indicando que existe un problema al cargar la base de datos.

2c. El sistema muestra la base de datos pero no existen camiones para importar.

- El usuario cierra la ventana de base de datos.

• **Caso de Uso 3.** Almacenar datos en BBDD de Camión.

Actor: Usuario.

Escenario principal:

1. El Caso de Uso comienza cuando, una vez abierta la aplicación, el usuario selecciona almacenar datos de un camión en BBDD.
2. El sistema exporta los datos a la BBDD de camiones.

Escenarios alternativos:

2b. El sistema detecta un problema con la base de datos.

- El sistema muestra un aviso indicando que existe un problema al guardar en la base de datos.

2c. El sistema detecta un problema con los datos a exportar.

- El sistema muestra un aviso indicando que existe un problema al exportar los datos a la base de datos.

• **Caso de Uso 4.** Recuperar proyecto.

Actor: Usuario.

Escenario principal:

1. El Caso de Uso comienza cuando, una vez abierta la aplicación, el usuario selecciona cargar un proyecto existente.
2. El sistema muestra un menú con el directorio actual.
3. El usuario selecciona entre los diversos directorios el que contiene el archivo a utilizar.
4. El usuario selecciona el archivo a cargar.
5. El sistema carga los datos y rellena las casillas correspondientes en la aplicación.

Escenarios alternativos:

3b. El usuario decide cancelar.

- El sistema cierra el directorio de archivos.
- El sistema muestra un aviso indicando que no se ha cargado ningún proyecto.

4b. El usuario decide cancelar.

- El sistema cierra el directorio de archivos.
- El sistema muestra un aviso indicando que no se ha cargado ningún proyecto.

• **Caso de Uso 5.** Guardar proyecto.

Actor: Usuario.

Escenario principal:

1. El Caso de Uso comienza cuando el usuario selecciona guardar un proyecto existente.
2. El sistema muestra un menú con el directorio actual.
3. El usuario selecciona entre los diversos directorios el que contendrá el archivo a guardar.
4. El usuario introduce un nombre para el archivo.
5. El sistema genera un archivo en la ubicación determinada.

Escenarios alternativos:

3b. El usuario decide cancelar.

- El sistema cierra el directorio de archivos.

4b. El usuario decide cancelar.

- El sistema cierra el directorio de archivos.

5b. El sistema detecta algún problema con el directorio de archivos.

- El sistema muestra un aviso indicando la imposibilidad de guardar la información,

• **Caso de Uso 6.** Borrar datos actuales.

Actor: Usuario.

Escenario principal:

1. El Caso de Uso comienza cuando, el usuario selecciona borrar los datos actuales del proyecto en curso.
2. El sistema muestra un aviso indicando que los datos se borrarán.
3. El usuario confirma que quiere borrar.
4. El sistema destruye las instancias de proyecto con las que estaba trabajando desde el camión, cargas, etc.

Escenarios alternativos:

3b. El usuario cancela la operación.

- El sistema cancela la eliminación.

• **Caso de Uso 7.** Crear secciones.

Actor: Usuario.

Escenario principal:

1. El Caso de Uso comienza cuando, el usuario selecciona el menú de secciones.
2. El sistema muestra una nueva ventana con la información relativa a secciones.
3. El usuario introduce la información de la sección que desea crear.
4. El sistema almacena la sección.

Escenarios alternativos:

3b. El usuario comprueba las características de la sección pulsando en dibujar sección.

- El sistema muestra una imagen acotada de las dimensiones introducida y muestra la información de las propiedades de la sección introducida.

4b. El sistema detecta errores en los datos introducidos.

- El sistema muestra un aviso al usuario para que corrija los datos incorrectos.

• **Caso de Uso 8.** Asignar secciones al camión.

Actor: Usuario.

Escenario principal:

1. El Caso de Uso comienza cuando, el usuario selecciona el menú de secciones.
2. El sistema muestra una nueva ventana con la información relativa a secciones.
3. El usuario selecciona asignar sección al camión.

Escenarios alternativos:

3b. El sistema detecta errores en la asignación y muestra un aviso de error.

• **Caso de Uso 9.** Recuperar sección de BBDD de secciones.

Actor: Usuario.

Escenario principal:

1. El Caso de Uso comienza cuando, el usuario selecciona el menú de secciones.
2. El sistema muestra una nueva ventana con la información relativa a secciones.
3. El usuario selecciona importar de la base de datos de secciones.

4. El sistema muestra una ventana con las secciones existentes.
5. El usuario selecciona la sección a importar.
6. El sistema rellena los datos de la sección importada en las casillas correspondientes.

Escenarios alternativos:

4b. El sistema encuentra errores con la base de datos.

- El sistema muestra un aviso indicando errores con la base de datos.

6b. El sistema detecta errores en la importación.

- El sistema muestra un aviso de error.

• **Caso de Uso 10.** Guardar sección en BBDD de secciones.

Actor: Usuario.

Escenario principal:

1. El Caso de Uso comienza cuando, el usuario selecciona el menú de secciones.
2. El sistema muestra una nueva ventana con la información relativa a secciones.
3. El usuario selecciona exportar a la base de datos de secciones.
4. El sistema exporta los datos a la base de datos de secciones.

Escenarios alternativos:

3b. El sistema encuentra errores con la base de datos.

- El sistema muestra un aviso indicando errores con la base de datos.

4b. El sistema detecta errores en la exportación.

- El sistema muestra un aviso de error.

• **Caso de Uso 11.** Introducir carga.

Actor: Usuario.

Escenario principal:

1. El Caso de Uso comienza cuando, el usuario selecciona el menú de cálculo.
2. El sistema muestra una nueva ventana con la información relativa a los cálculos.
3. El usuario selecciona crear una nueva carga.
4. El sistema muestra una nueva ventana para la introducción de cargas.
5. El usuario introduce la información de la carga.
6. El sistema guarda la carga.
7. El sistema actualiza el esquema del camión.

8. El sistema actualiza el diagrama de esfuerzos cortantes.
9. El sistema actualiza el diagrama de momentos flectores.
10. El sistema actualiza las reacciones en los ejes.
11. El sistema calcula las tensiones sobre la sección.
12. El sistema actualiza la lista de cargas.

Escenarios alternativos:

6b. El sistema encuentra errores con los datos introducidos.

- El sistema muestra un aviso indicando errores en la introducción de datos.

13. El sistema detecta que la nueva carga produce una sobrecarga en los ejes.

- El sistema muestra un aviso indicando que los ejes exceden su carga máxima.

• **Caso de Uso 12.** Visualizar carga.

Actor: Usuario.

Escenario principal:

1. El Caso de Uso comienza cuando, el usuario selecciona el menú de cálculo.
2. El sistema muestra una nueva ventana con la información relativa a los cálculos.
3. El usuario selecciona visualizar una nueva carga.
4. El sistema muestra una nueva ventana para la visualización de cargas.
5. El usuario selecciona la carga a visualizar.
6. El usuario selecciona visualizar.
7. El sistema actualiza el esquema del camión.

Escenarios alternativos:

6b. El usuario selecciona ocultar.

- El sistema actualiza el esquema.

• **Caso de Uso 13.** Eliminar carga.

Actor: Usuario.

Escenario principal:

1. El Caso de Uso comienza cuando, el usuario selecciona el menú de cálculo.
2. El sistema muestra una nueva ventana con la información relativa a los cálculos.
3. El usuario selecciona eliminar una nueva carga.
4. El sistema muestra una nueva ventana para la eliminación de cargas.

5. El usuario selecciona la carga a eliminar.
6. El usuario selecciona eliminar.
7. El sistema actualiza el esquema del camión.
8. El sistema actualiza el diagrama de esfuerzos cortantes.
9. El sistema actualiza el diagrama de momentos flectores.
10. El sistema actualiza las reacciones en los ejes.
11. El sistema calcula las tensiones sobre la sección.
12. El sistema actualiza la lista de cargas.

Escenarios alternativos:

13. El sistema comprueba si eliminando la carga se sobrepasa la carga máxima en los ejes.

- El sistema muestra un aviso indicando que los ejes exceden su carga máxima.

• **Caso de Uso 14.** Seleccionar zona de cálculo.

Actor: Usuario.

Escenario principal:

1. El Caso de Uso comienza cuando, el usuario selecciona el menú de cálculo.
2. El sistema muestra una nueva ventana con la información relativa a los cálculos.
3. El usuario selecciona la zona del camión articulado con la que quiere trabajar.
4. El sistema actualiza el esquema del camión.
5. El sistema actualiza el diagrama de esfuerzos cortantes.
6. El sistema actualiza el diagrama de momentos flectores.
7. El sistema actualiza las reacciones en los ejes.
8. El sistema calcula las tensiones sobre la sección.
9. El sistema actualiza la lista de cargas.

Escenarios alternativos:

3b. El camión es rigido.

- El sistema deshabilita la posibilidad de trabajar con zonas.

• **Caso de Uso 15.** Cambiar unidades.

Actor: Usuario.

Escenario principal:

1. El Caso de Uso comienza cuando, el usuario accede a la pantalla principal.

2. El usuario selecciona las unidades en las que quiere trabajar.
3. El sistema actualizar las unidades en las casillas con los datos correspondientes.

• **Caso de Uso 16.** Generar informe técnico.

Actor: Usuario.

Escenario principal:

1. El Caso de Uso comienza cuando, el usuario selecciona el menú de informe.
2. El sistema muestra una nueva ventana para que el usuario introduzca el nombre del informe.
3. El usuario introduce el nombre.
4. El sistema crea un pdf con el nombre provisto con la información requerida en formato pdf.
5. El sistema cierra la ventana de generación de informe.

Escenarios alternativos:

3b. El usuario cancela la generación.

- El sistema cierra la ventana de generación de archivo.

4b. El sistema detecta un problema con la generación del archivo.

- El sistema muestra un aviso con la imposibilidad de crear el archivo.

3.6. Conclusiones

Mediante el estudio de los requisitos definiremos las especificaciones mínimas para las que el sistema debe ser capaz de proveer una solución lo más optimizada y adecuada posible.

A lo largo de este capítulo se ha realizado un estudio de los puntos críticos en todo análisis del problema. Se han revisado los siguientes puntos críticos para nuestro problema:

- **Modelo de dominio:** Se han mostrado en esta parte del análisis las clases potenciales más importantes que contemplará el sistema.
- **Requisitos del sistema:** Se han descrito en esta parte del análisis los requisitos funcionales que debe contemplar el sistema. Adicionalmente, se han expuesto un conjunto de requisitos no funcionales que podrán completar los requerimientos que el sistema deberá cumplir.
- **Actores principales:** Se ha mostrado el único actor que interviene en la aplicación, siendo este un usuario que por lo general será un técnico competente para el análisis de los datos introducidos y mostrados posteriormente al cálculo.

- **Casos de Uso:** Se han expuesto los casos de uso principales para la aplicación, incluyendo los principales casos para la gestión de los datos, para el cálculo, para la preparación de los elementos constructivos y para la generación del informe de resultados que el usuario espera obtener para completar la documentación a presentar a la autoridad competente.

Capítulo 4. Diseño

4.1. Introducción

Hasta este punto se ha estado realizando un análisis que incluye entre otros puntos los requisitos técnicos, los actores principales y los casos de uso necesarios para entender el problema y poder diseñar una solución que resuelva de la forma más ajustada a estos requisitos el problema planteado. Una vez realizado dicho análisis, se pretende en este capítulo describir las distintas facetas del diseño que se ha decidido adoptar.

El diseño que se aborda ha tenido en cuenta los siguientes aspectos.

- Maximizar la cohesión: Se intenta que cada clase sea responsable de una tarea específica o de manejar su propio concepto.
- Minimizar el acoplamiento. Se intenta que se produzca una dependencia mínima entre las unidades funcionales, si bien un acoplamiento inexistente es imposible (Wikipedia 2022).
- Maximizar la reutilización. Se intenta que las unidades funcionales que constituirán la solución puedan ser empleadas en otros problemas parecidos.
- Permitir la portabilidad. Se utilizará un lenguaje Java, que permitirá que el ejecutable sea funcional en cualquier sistema operativo que implemente la máquina virtual de Java.Facilitar la flexibilidad y el mantenimiento en el desarrollo. Se intentará en todo caso poder mejorar la flexibilidad sobre las unidades funcionales para facilitar el trabajo a la hora de tener que realizar modificaciones y mantenimiento sobre el software.

4.2. Modelo de Diseño

El modelo de diseño se basará en el modelo del análisis para describir la estructura del sistema y la forma en que se han ido implementando los requisitos expuestos anteriormente (IBM 2021).

4.2.1. Diagramas de secuencia

Las figuras 4.1 a 4.16 exponen los diagramas de secuencia para los casos de uso expuestos anteriormente.

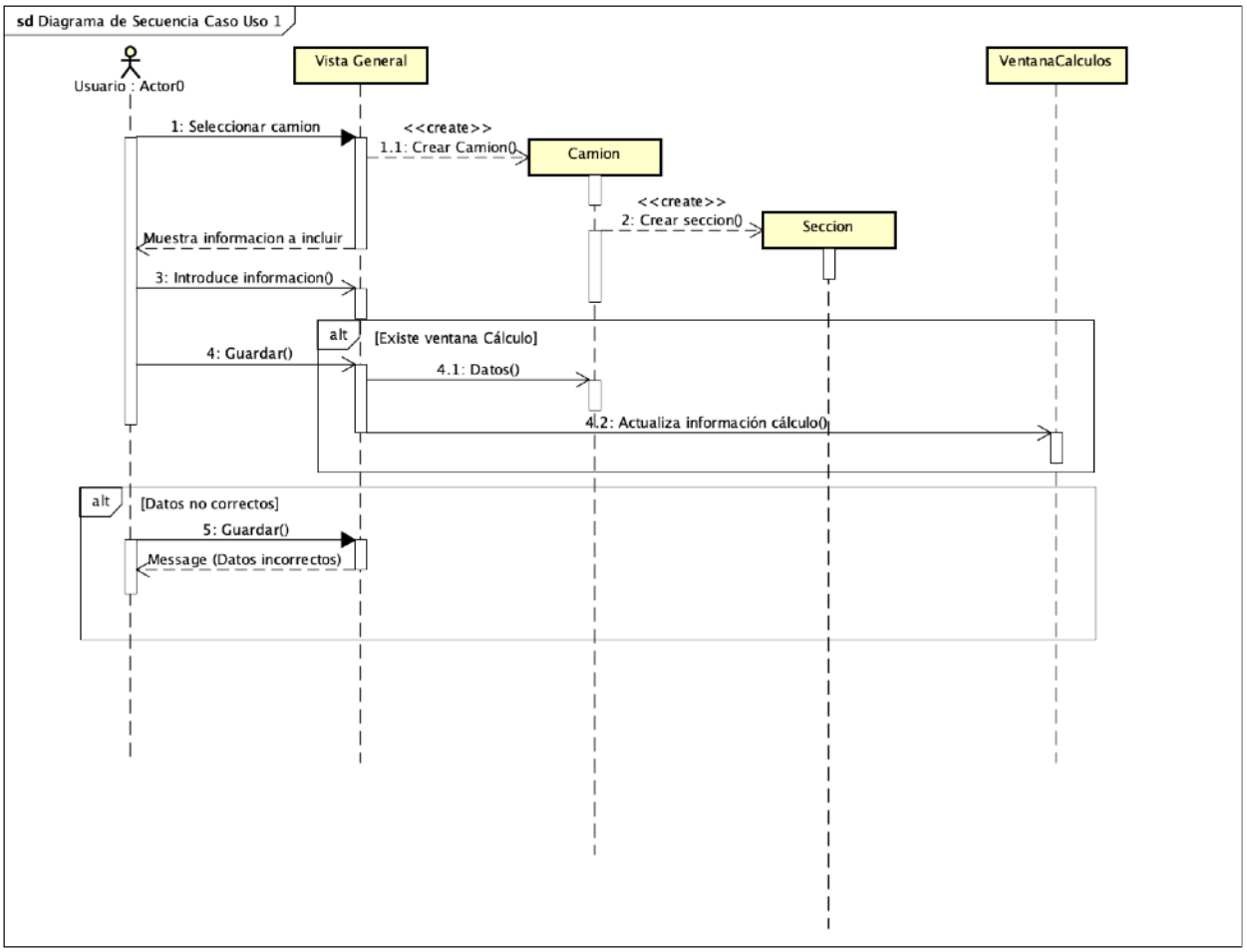


Figura 4.1. Diagrama de secuencia caso de uso 1.

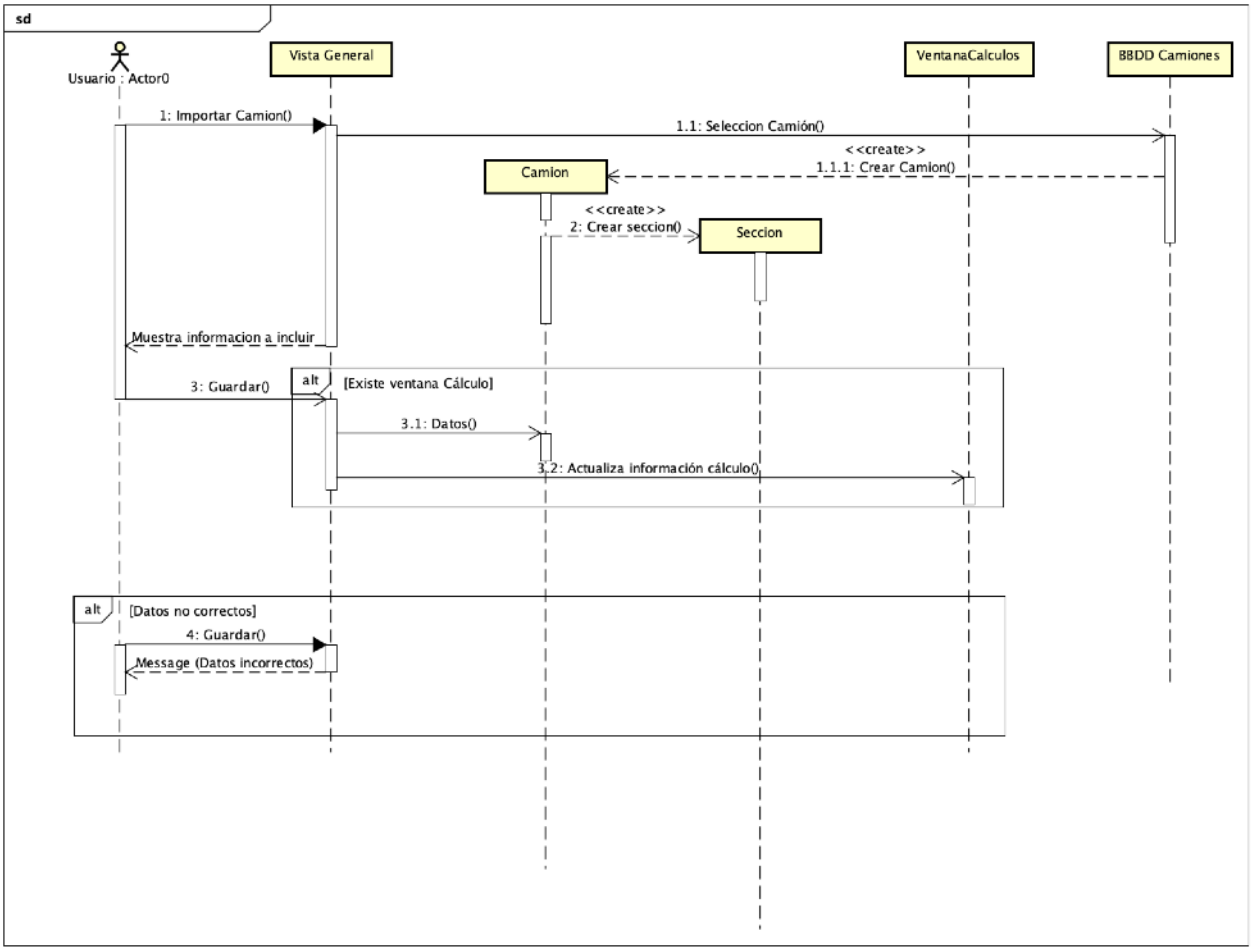


Figura 4.2. Diagrama de secuencia caso de uso 2.

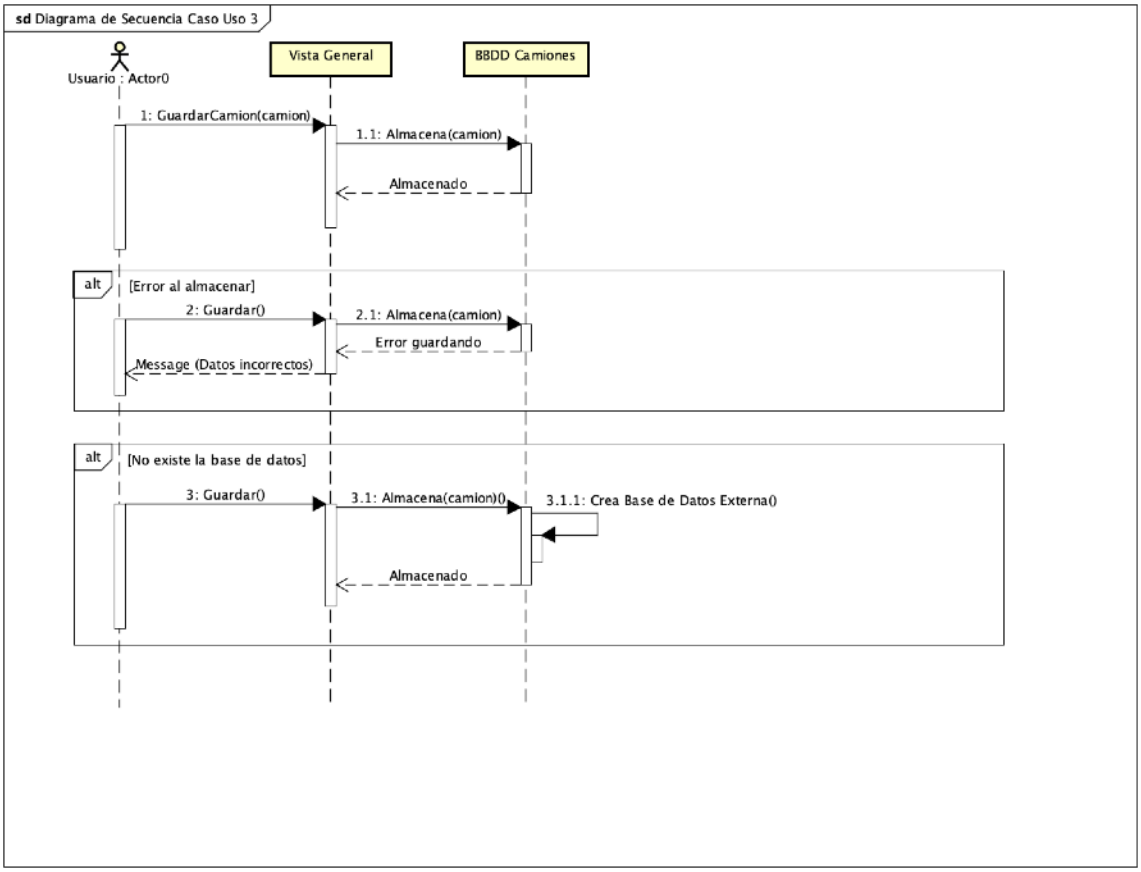


Figura 4.3. Diagrama de secuencia caso de uso 3.

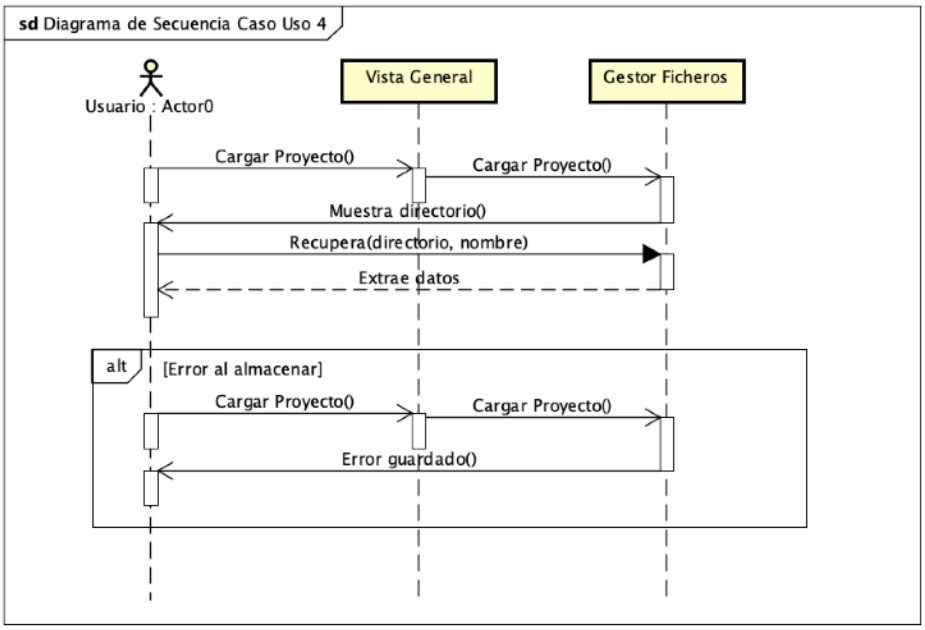


Figura 4.4. Diagrama de secuencia caso de uso 4.

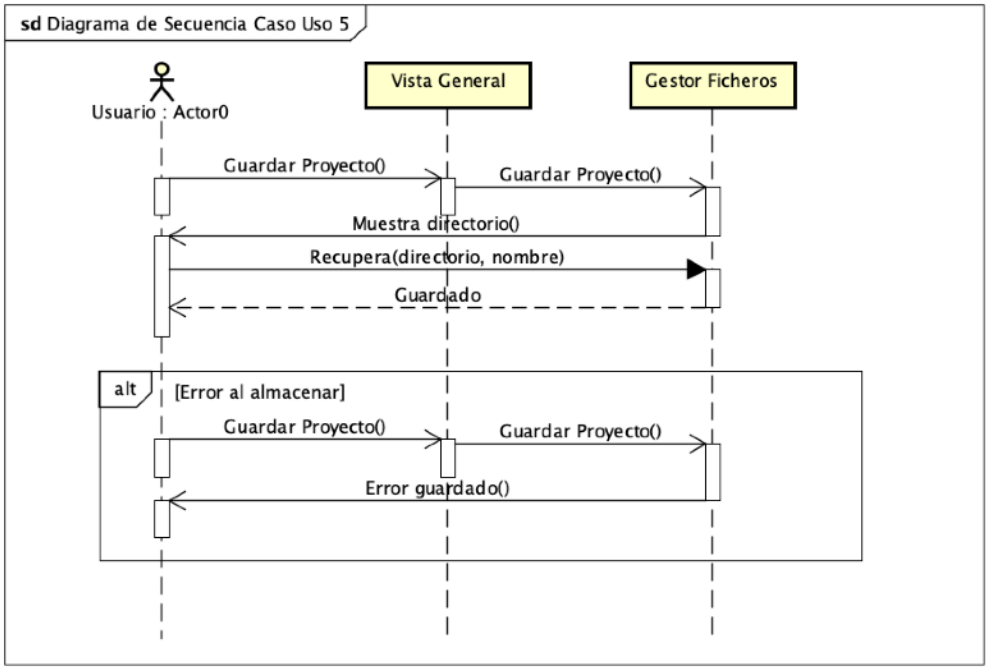


Figura 4.5. Diagrama de secuencia caso de uso 5.

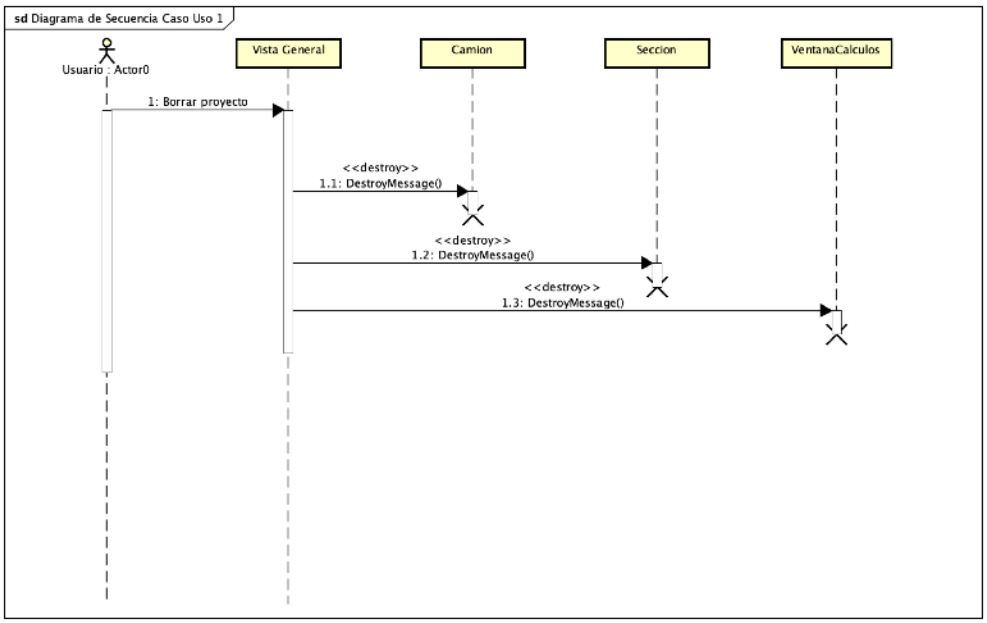


Figura 4.6. Diagrama de secuencia caso de uso 6.

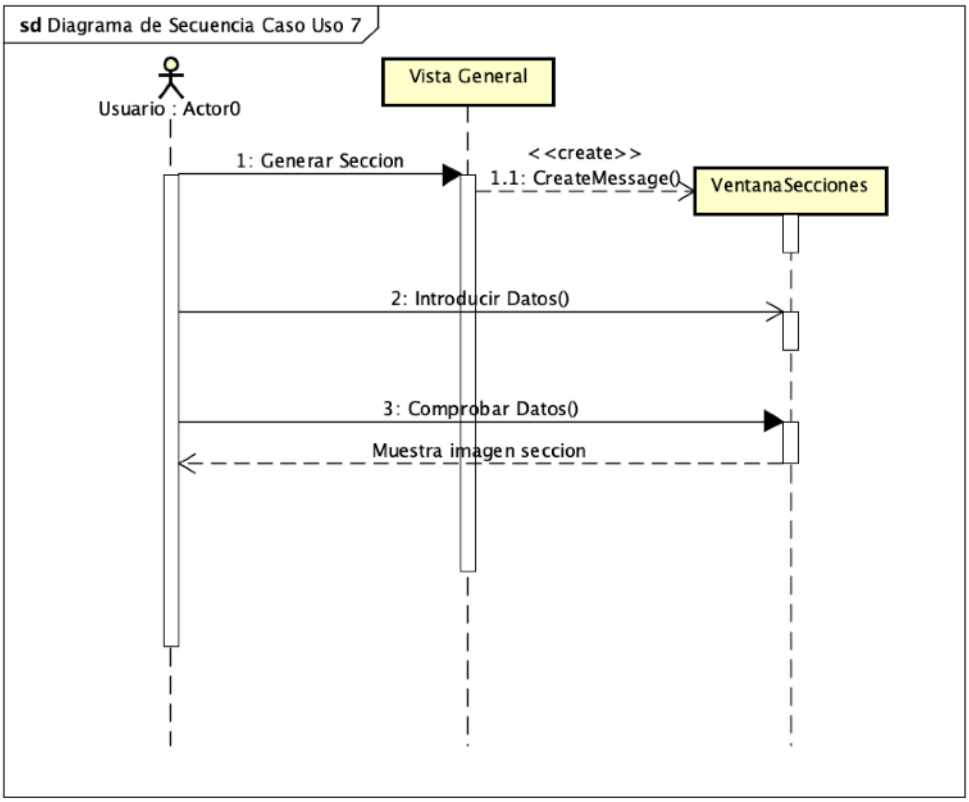


Figura 4.7. Diagrama de secuencia caso de uso 7.

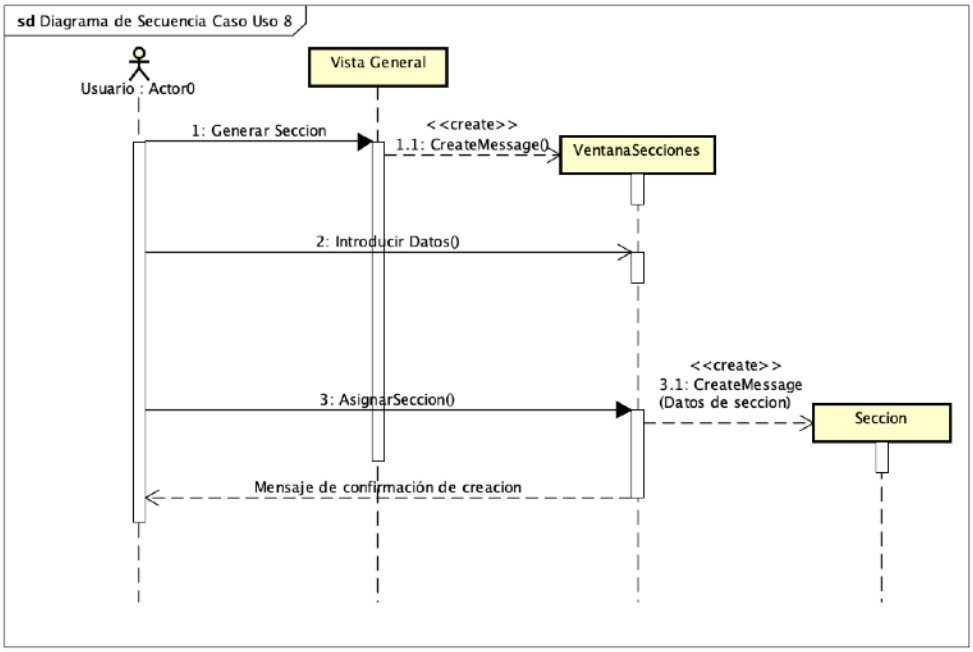


Figura 4.8. Diagrama de secuencia caso de uso 8.

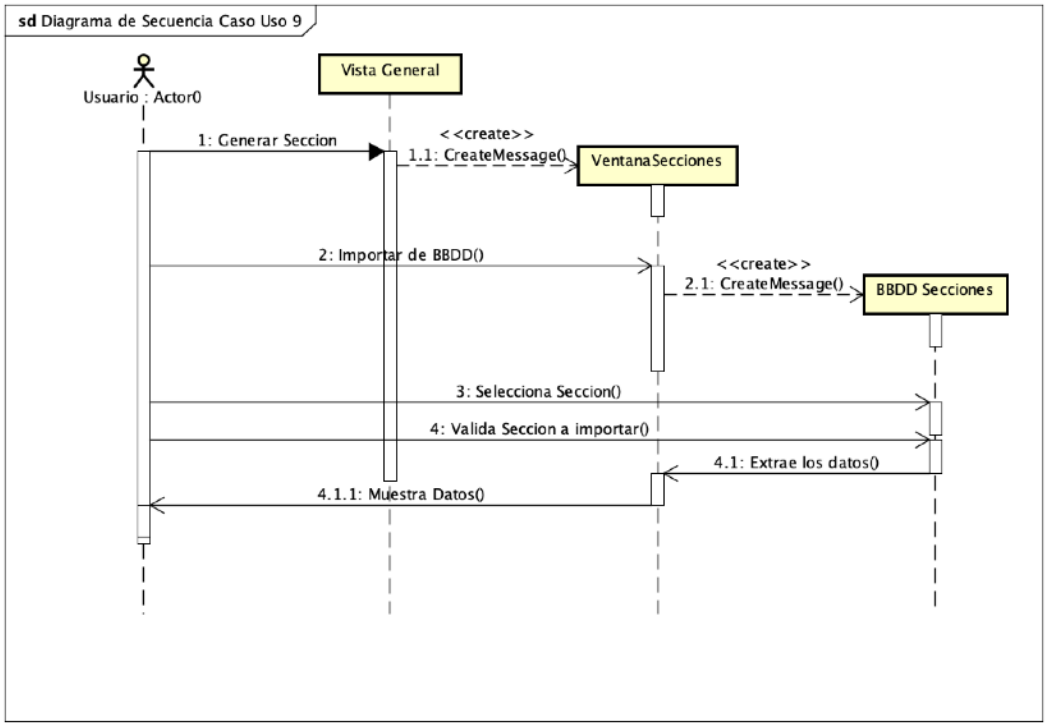


Figura 4.9. Diagrama de secuencia caso de uso 9.

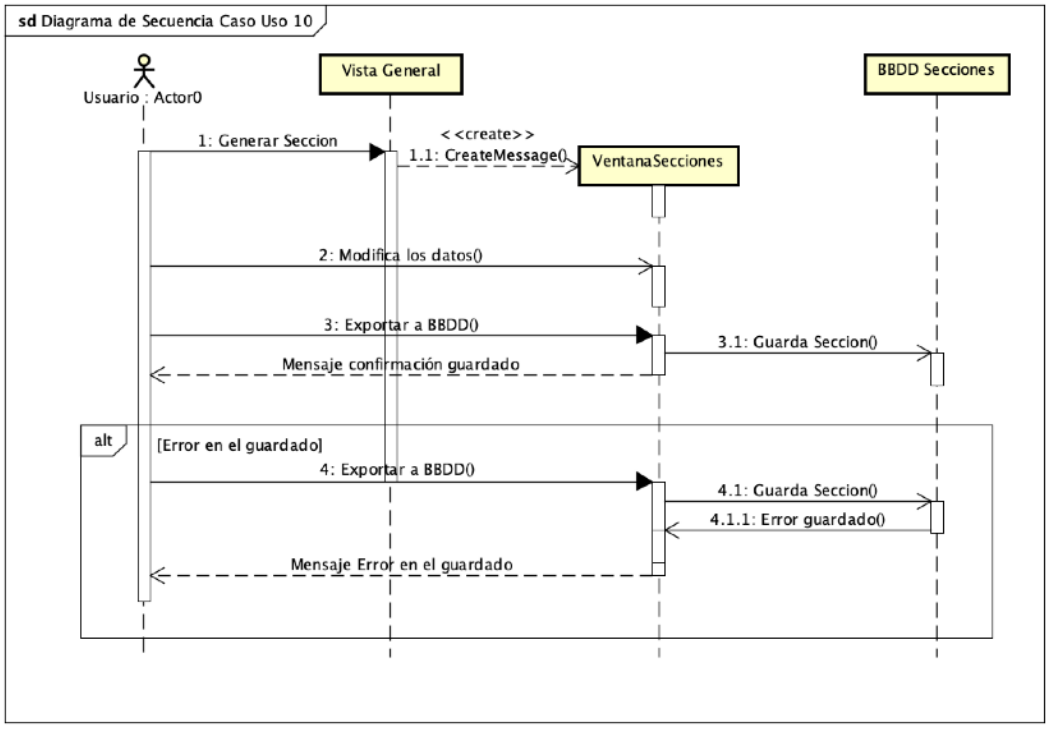


Figura 4.10. Diagrama de secuencia caso de uso 10.

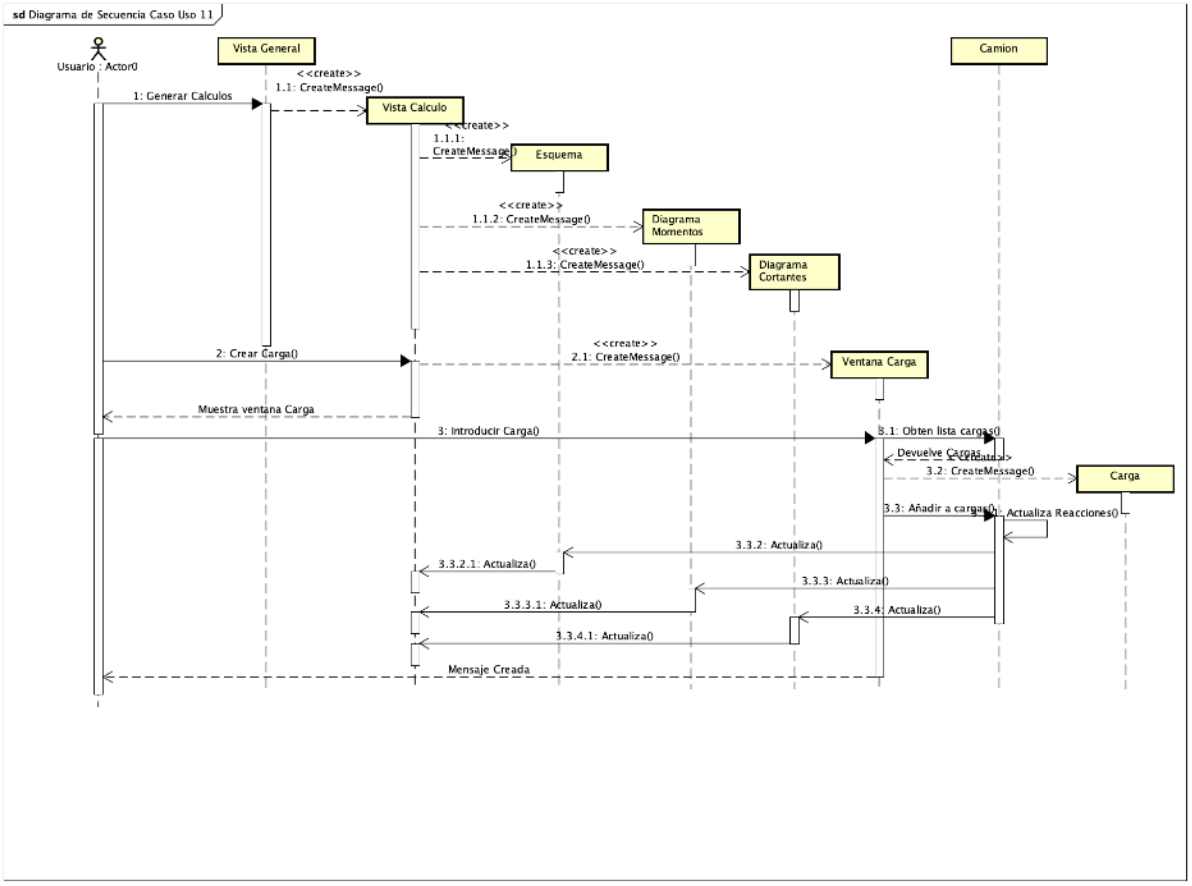


Figura 4.11. Diagrama de secuencia caso de uso 11.

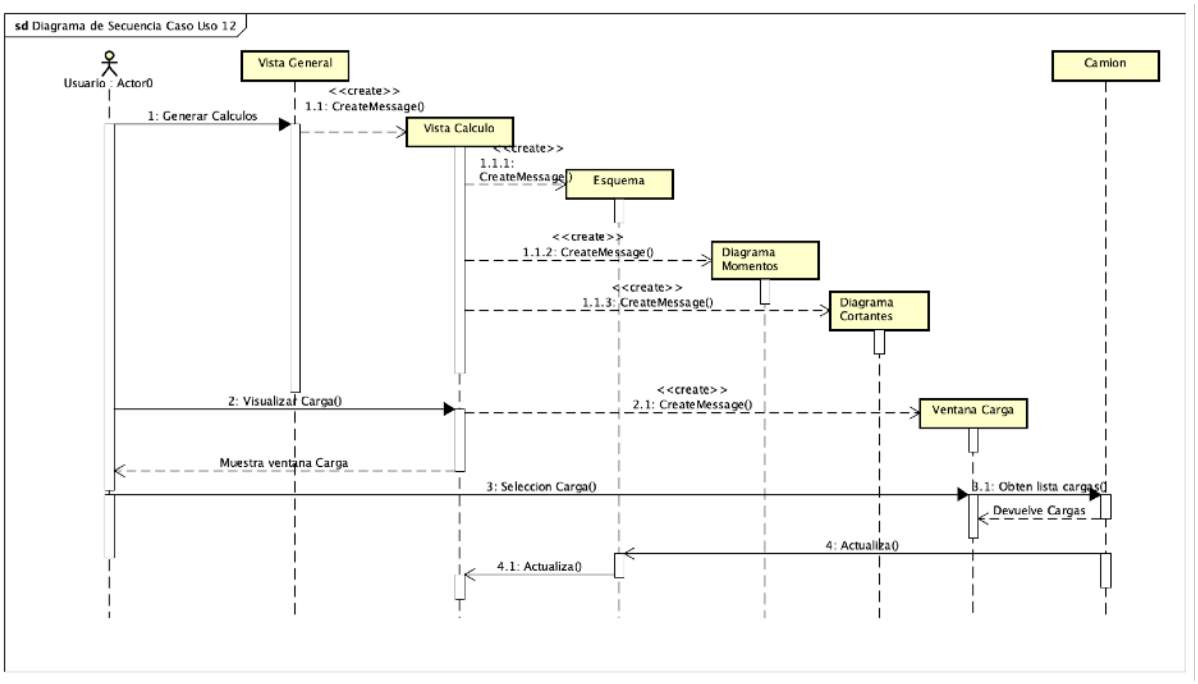


Figura 4.12. Diagrama de secuencia caso de uso 12.

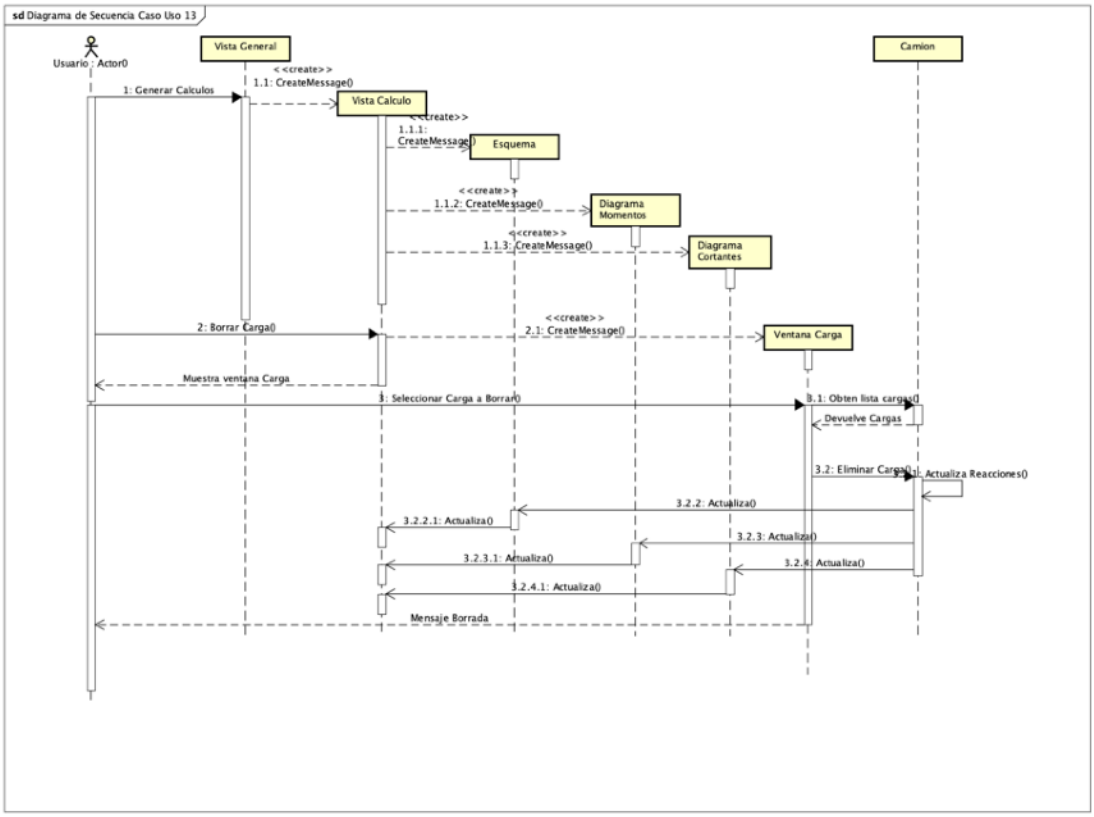


Figura 4.13. Diagrama de secuencia caso de uso 13.

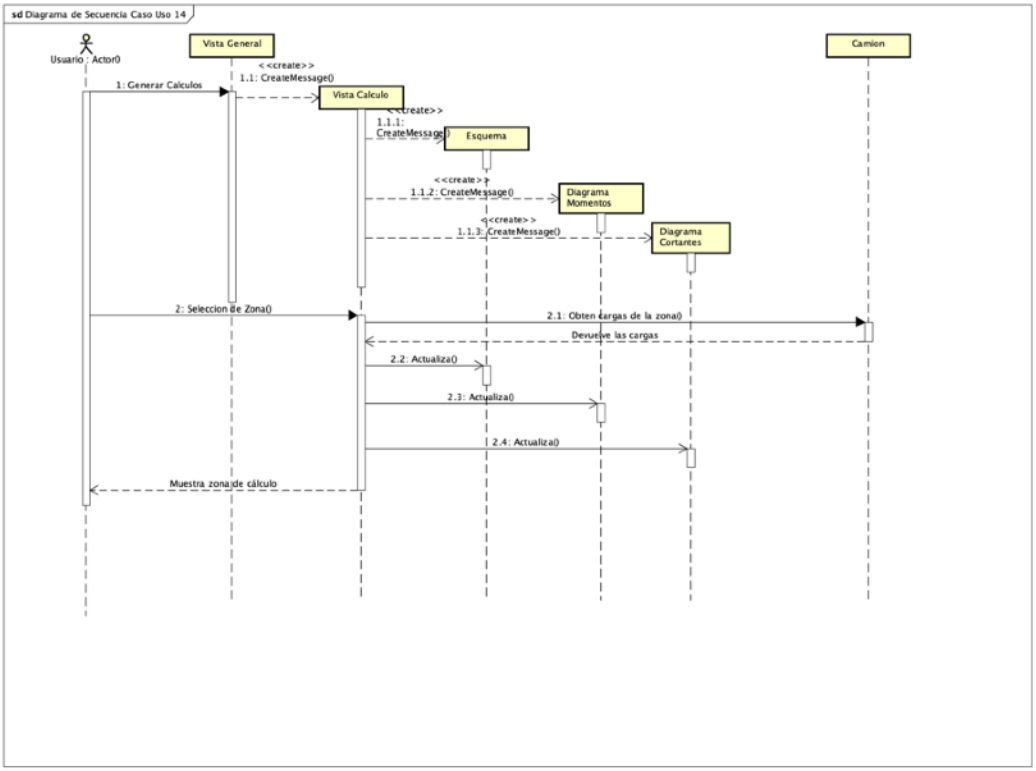


Figura 4.14. Diagrama de secuencia caso de uso 14.

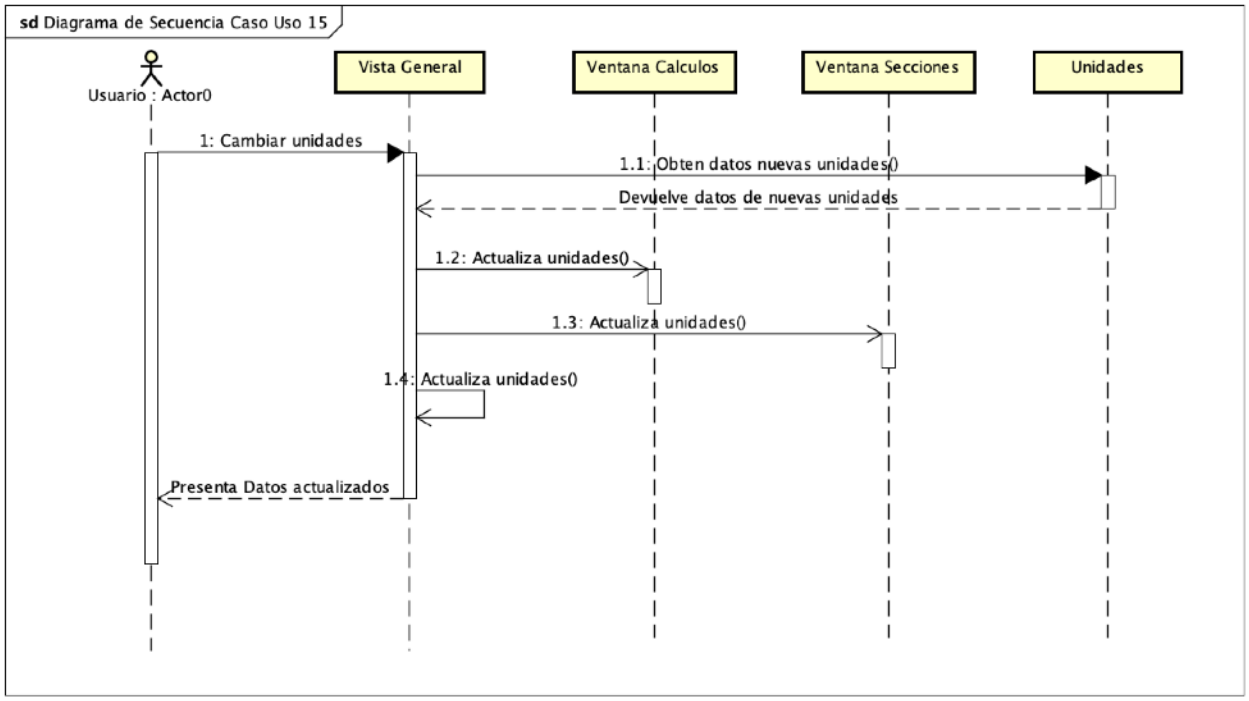


Figura 4.15. Diagrama de secuencia caso de uso 15.

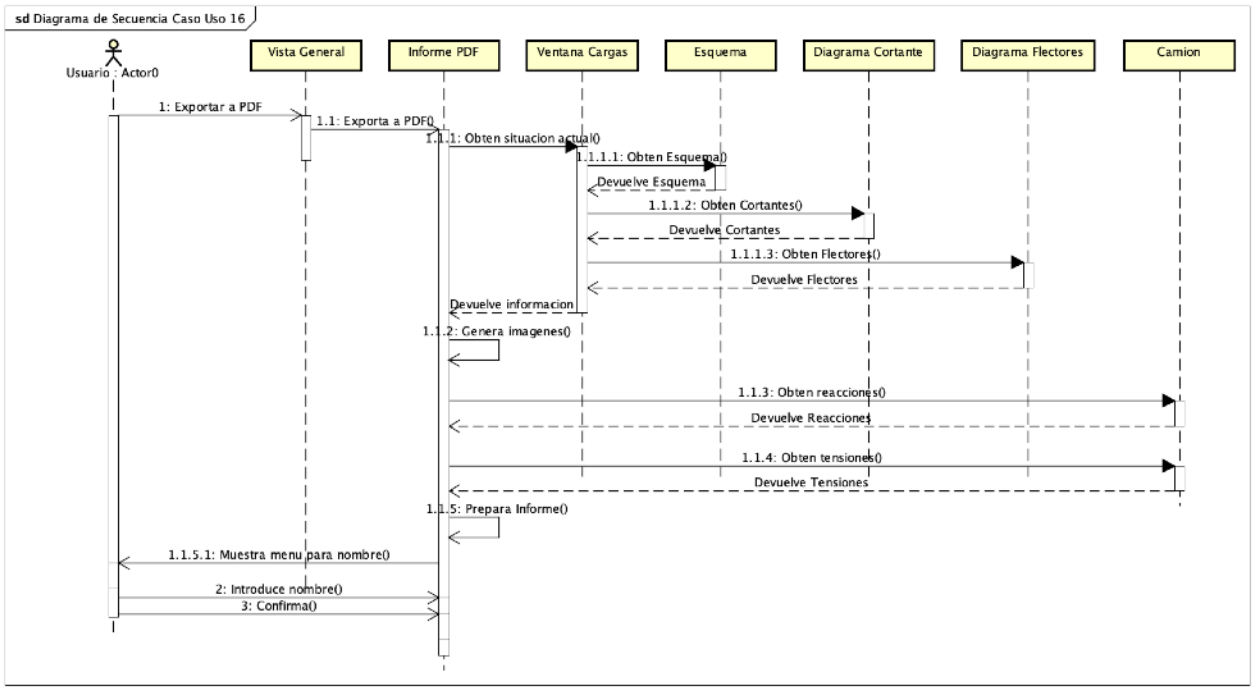


Figura 4.16. Diagrama de secuencia caso de uso 16.

4.2.2. Arquitectura lógica

"La arquitectura del software identifica los elementos estructurales y las interfaces de un sistema junto con el comportamiento de los componentes y los subsistemas individuales" (Kruchten, Booch, Bittner y Reitman 2009).

Es importante *"hacer énfasis en el rol de los componentes del software" (Pressman y Maxim)*. En nuestro caso, debo además incluir el comentario de que el desarrollo de la presente aplicación se ha realizado con modelos ágiles. El planteamiento ha sido el de realizar un desarrollo basado en un modelo incremental de prototipos ágil, mediante el cual se han ido incluyendo distintas funcionalidades a cada versión de prototipo. Por esta razón, la arquitectura ha ido ajustándose a las nuevas necesidades detectadas tras cada fase de prototipo.

A continuación se muestra la arquitectura lógica del sistema en la Figura 4.17.

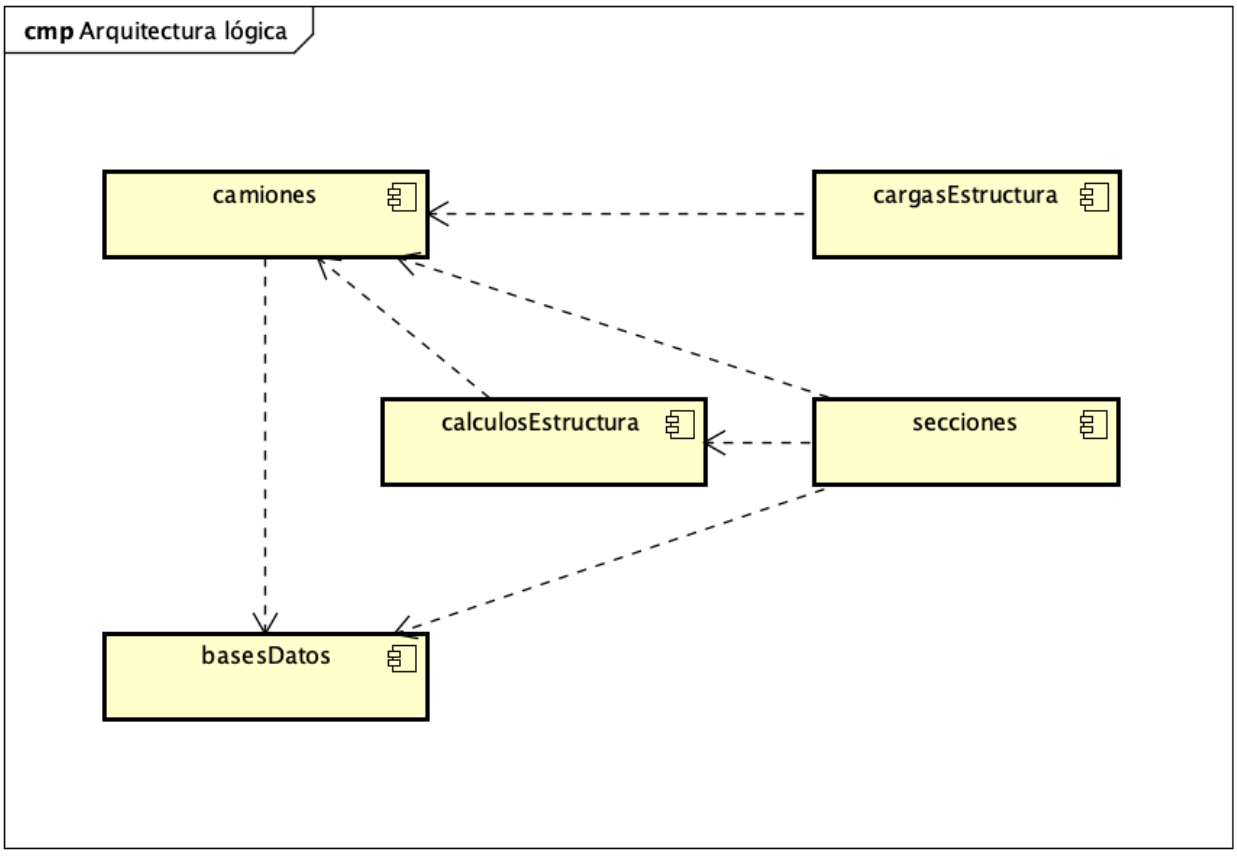


Figura 4.17. Arquitectura lógica del sistema

Se ha planteado una división en 6 paquetes, si bien en la figura solo se muestran cinco de ellos, ya que el sexto contiene clases auxiliares para diversos propósitos que se explicarán más adelante.

A continuación se exponen los distintos paquetes y sus clases así como su funcionalidad.

Paquete Camiones

El paquete camiones tiene la finalidad principal de gestionar toda la información referente a los camiones y vehículos de cálculo.

Permitirá establecer y almacenar toda la información relativa a dimensiones, pesos, tensiones, datos de cliente, etc. para la gestión de cada proyecto.

Se realiza dentro de la división en clases una distinción en tipologías de camión ya que, en función de que el vehículo sea rígido o articulado, nos encontraremos con dos formas de proceder con los cálculos y presentación de resultados.

Las clases introducidas en este paquete tienen las siguientes funciones:

- Establecer una división de tipologías clara en los camiones.
- Facilitar la herencia de una clase común de camión que permita especificar otros datos importantes en cada caso y ampliar, en caso necesario para futuro, con nuevas tipologías como autobuses, caravanas, etc.
- Contener la información técnica del proyecto: Reacciones, tensiones, cargas, etc.
- Contener la información genérica del proyecto: Cliente, Modelo, Bastidor, etc.
- Facilitar el almacenamiento de la información en ficheros, al estar recogida en un objeto específico que puede serializarse mediante la implementación de la interfaz Serializable y permita facilitar su gestión posterior.
- Definir un objeto claro sobre el que trabajar a la hora de establecer la metodología de validación.

La Tabla 4.1 recoge el paquete Camiones junto con sus componentes.

Paquete Secciones

El paquete Secciones se ha creado para facilitar el trabajo con las mismas, al ser esta una de las propiedades del camión más importantes y el centro de la justificación del cálculo que debe prepararse para las autoridades técnicas.

Paquete	Componente	Tipo	Proposito
Camiones	Camion	Clase	Permite gestionar la información general de los camiones. Permite además la herencia sobre los dos tipos de camiones.
	Camion Articulado	Clase	Especifica la información de los tipos de camión articulados.
	Camion Rigido	Clase	Especifica la información de los tipos de camión rígidos.
	Proyecto	Clase	Permite especificar los datos del proyecto específico.
	TipoCamion	Clase	Facilita la gestión de los dos tipos de camión en la implementación del software.
	TruckCalc	Clase	Permite arrancar la aplicación y establecer el proyecto sobre el que se trabajará.
	VistaGeneral	Clase	Permite establecer una interfaz visual adecuada para la gestión de los datos de camión y el acceso a otras funcionalidades.

Tabla 4.1. Componentes del paquete Camiones.

El paquete secciones y sus clases tienen las siguientes funcionalidades:

- Definir una interfaz de Sección que todas las secciones que se utilicen sean capaces de utilizar.
- Establecer una sección básica (la rectangular) sobre la que trabajar para combinarlas y obtener otras secciones como las tipo I y las tipo C de forma que el cálculo de las propiedades de estas últimas sea fácilmente obtenible.
- Establecer una interfaz visual que permita al usuario poder entender qué propiedades y dimensiones se están asignando al perfil utilizado para minimizar los posibles errores de escritura de los datos.

La Tabla 4.2 muestra los componentes del paquete Secciones:

Paquete	Componente	Tipo	Proposito
Secciones	Seccion	Interfaz	Define una interfaz que puedan implementar todas las secciones.
	CalculoSeccion	Clase	Se encarga de los cálculos de las diferentes propiedades de las secciones.
	SeccionRectangular	Clase	Especifica una sección básica con la que realizar los cálculos del resto de secciones.
	VentanaSeccion	Clase	Permite establecer una interfaz visual con el usuario para trabajar con las propiedades de las secciones.

Tabla 4.2. Componentes del paquete Secciones.

Paquete CargasEstructura

El paquete CargasEstructura se ha creado para poder definir claramente tipologías de carga comúnmente utilizadas en las estructuras, de forma que el trabajo con cada una de ellas sea lo más sencillo posible, tanto desde el punto de vista del usuario como desde el punto de vista del programador que tenga que gestionar los datos de cada tipo.

Se persiguen las siguientes funcionalidades:

- Establecer una interfaz que toda carga deberá implementar para facilitar su gestión.
- Establecer una clase con las propiedades básicas de carga de la que posteriormente se pueda heredar.
- Permitir ampliar las tipologías de carga en caso necesario si en un futuro se desease. Se podría pensar en introducir momentos torsores, cargas axiales, etc.
- Facilitar la distinción de las tipologías de cargas a la hora de tenerlas en cuenta para el cálculo.

La Tabla 4.3 muestra el paquete CargasEstructuras junto con sus componentes:

Paquete	Componente	Tipo	Proposito
CargasEstructura	Carga	Clase	Implementa la interfaz CargasInterface y define una clase que contiene las propiedades básicas de una carga.
	CargaDistribuida	Clase	Hereda de Carga y se especifica para cargas distribuidas.
	CargaPuntual	Clase	Hereda de Carga y se especifica para cargas puntuales.
	CargasInterface	Interfaz	Establece una interfaz que deberá cumplir toda clase relacionada con las cargas.
	TipoCarga	Clase	Facilita la división de tipologías de cargas a la hora de su gestión en el software.

Tabla 4.3. Componentes del paquete CargasEstructura.

Paquete CalculosEstructura

Este paquete contiene la parte más estudiada y depurada. Se trata posiblemente del componente más importante en el sistema debido a que los resultados provistos por los componentes de este paquete definirán cuál es la estructura que el camión tendrá que incluir.

Se persiguen las siguientes funcionalidades:

- Implementar una metodología rápida de cálculo de cargas en cada punto de la estructura para la generación de los diagramas necesarios para presentar en las ITV.
- Implementar una metodología rápida de cálculo de cargas en el punto de máxima carga para el correcto dimensionamiento de la viga.
- Facilitar las modificaciones y actualizaciones de valores a la hora de la introducción de cargas.
- Permitir dar al usuario una visualización completa del estado estructural del vehículo, de manera que pueda decidir si algún cambio es necesario.
- Facilitar la distinta metodología de cálculo en los casos de camión rígido y Articulado.

A continuación se muestra el paquete CalculosEstructura con sus componentes en la Tabla 4.4:

Paquete	Componente	Tipo	Proposito
CalculosEstructura	CalculoReaccionesArticulado	Clase	Se encarga de realizar los cálculos de las reacciones en camiones articulados
	CalculoReaccionesRigidos	Clase	Se encarga de realizar los cálculos de las reacciones en camiones rígidos.
	LeyesCortantes	Clase	Se encarga de calcular todo lo relacionado con los esfuerzos cortantes.
	LeyesMomentos	Clase	Se encarga de calcular todo lo relacionado con los momentos flectores.
	PanelEsquema	Clase	Se encarga de generar el esquema simplificado para el cálculo.
	PanelCortantes	Clase	Se encarga de generar la representación de las leyes de cortantes.
	PanelMomentos	Clase	Se encarga de generar la representación de las leyes de momentos.
	PuntosCalculo	Clase	Define las propiedades de cada punto de cálculo.
	VentanaCalculos	Clase	Establece la interfaz gráfica donde se trabajará con los cálculos.
	VentanaEliminarCarga	Clase	Establece la interfaz gráfica donde se eliminaran cargas.
	VentanaIntroduceCarga	Clase	Establece la interfaz gráfica donde se añadirán cargas.
	VentanaVerCarga	Clase	Establece la interfaz gráfica donde se visualizarán cargas.
	VentanaSeleccionParteArticulado	Clase	Establece la forma de trabajar con la cabeza tractora o con el remolque en vehículos articulados.

Tabla 4.4. Componentes del paquete CalculosEstructura.

Paquete BasesDatos

Este paquete contiene las clases necesarias para la gestión de la información contenida en las bases de datos de camiones y secciones.

Se persiguen las siguientes funcionalidades:

- Permitir gestionar fácilmente la creación de bases de datos.
- Permitir gestionar la presentación de la información mediante tablas.
- Permitir una organización adecuada de la información en las bases de datos.
- Permitir importar, exportar, modificar y borrar datos de las mismas mediante SQL embebido en Java.

Se muestra el paquete BasesDatos con los componentes incluidos en el mismo en la Tabla 4.5.

Paquete	Componente	Tipo	Proposito
BasesDatos	BaseDatosCamiones	Clase	Se encarga de la gestión de la base de datos de camiones.
	BaseDatosSecciones	Clase	Se encarga de la gestión de la base de datos de secciones.
	VentanaBaseDatosCamiones	Clase	Establece una interfaz amigable al usuario para trabajar con los datos de la base de datos de camiones.
	VentanaBaseDatosSecciones	Clase	Establece una interfaz amigable al usuario para trabajar con los datos de la base de datos de secciones.

Tabla 4.5. Componentes del paquete BasesDatos.

Paquete Auxiliares

Este paquete contiene las clases auxiliares que se emplean para distintos propósitos no específicamente relacionados con ninguna de las funcionalidades descritas en los anteriores paquetes descritos.

Se persiguen las siguientes funcionalidades:

- Gestionar los avisos y errores.

- Gestionar los formatos de una manera rápida, permitiendo los cambios visuales de manera eficiente.
- Permitir la adecuación de las unidades a las necesidades del usuario.
- Permitir generar el informe técnico de manera rápida una vez se ha terminado con el cálculo.
- Gestionar la carga y almacenamiento de datos en ficheros.
- Contener información común para todos los paquetes de la aplicación para poder realizar su gestión de forma eficiente.

Se muestra el paquete Auxiliares con los componentes incluidos en el mismo en la Tabla 4.6.

Paquete	Elemento	Tipo	Proposito
Auxiliares	CheckDatos	Clase	Comprueba los datos introducidos para dar avisos.
	CogeImagen	Clase	Sirve para acceder a los recursos de imágenes.
	CreaPDF	Clase	Se encarga de la generación del documento PDF.
	DatosGenericos	Clase	Se encarga de gestionar datos que resultan ser genéricos y constantes en la aplicación.
	FicheroUsado	Clase	Se encarga de la gestión de ficheros.
	Formato	Clase	Se encarga del formato del tipo de letra, tamaño de letra, colores, etc.
	FormatoCeldaTabla	Clase	Se encarga del formato de las celdas genéricas en las tablas.
	FormatoCeldaTablaSeleccionada	Clase	Se encarga del formato de las celdas seleccionadas en las tablas.
	FormatoEncabezadoTabla	Clase	Se encarga del formato de las celdas del encabezado de las tablas.
	IconoAjustado	Clase	Se encarga de ajustar los iconos a los tamaños necesarios.
	PanelConImagen	Clase	Se encarga de generar un JPanel con imagen de fondo.
	Unidades	Clase	Se encarga de la gestión de las unidades.
	VentanaIntroduceNombrePDF	Clase	Se encarga de la gestión de la ventana para la introducción del nombre del PDF a generar.
	VentanaProceso	Clase	Se encarga de gestionar la información que se muestra sobre el proceso realizado por el usuario

Tabla 4.6. Componentes del paquete Auxiliares.

4.2.3. Diagramas de clase

Las figuras 4.18 a 4.23 muestran los diagramas de clase para los paquetes expuestos anteriormente.

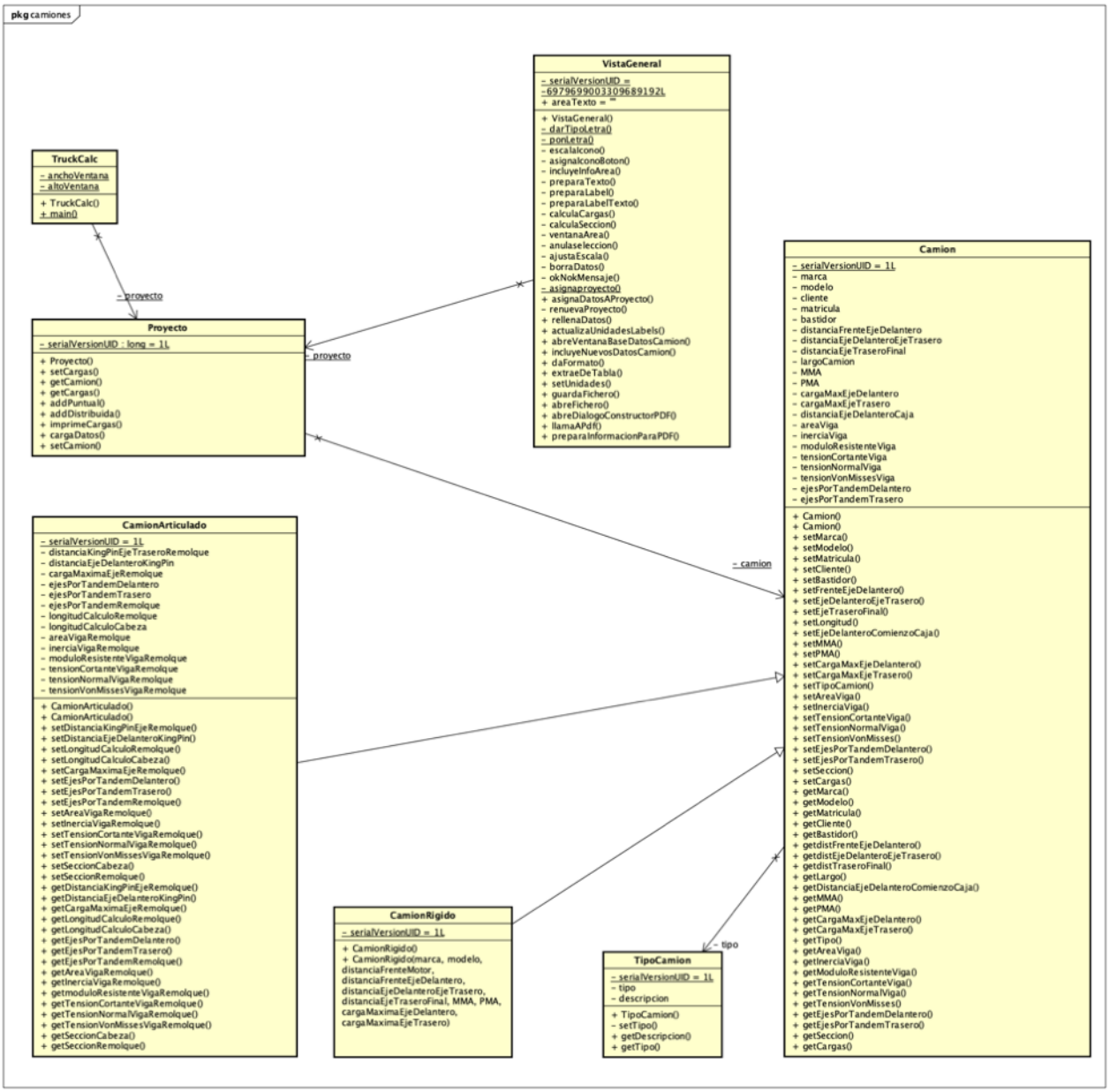


Figura 4.18. Diagrama de clases del paquete camiones.

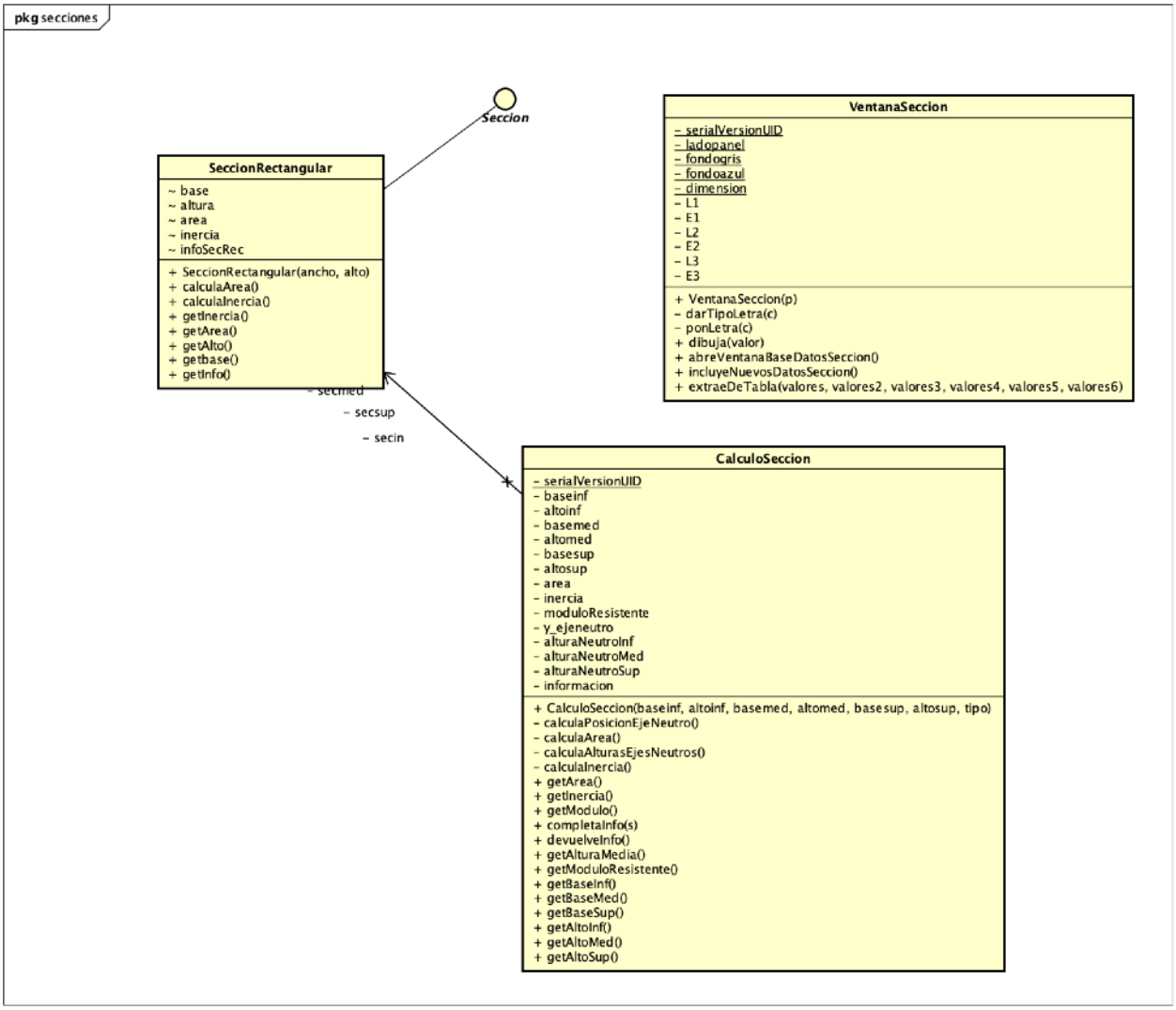


Figura 4.19. Diagrama de clases del paquete secciones.

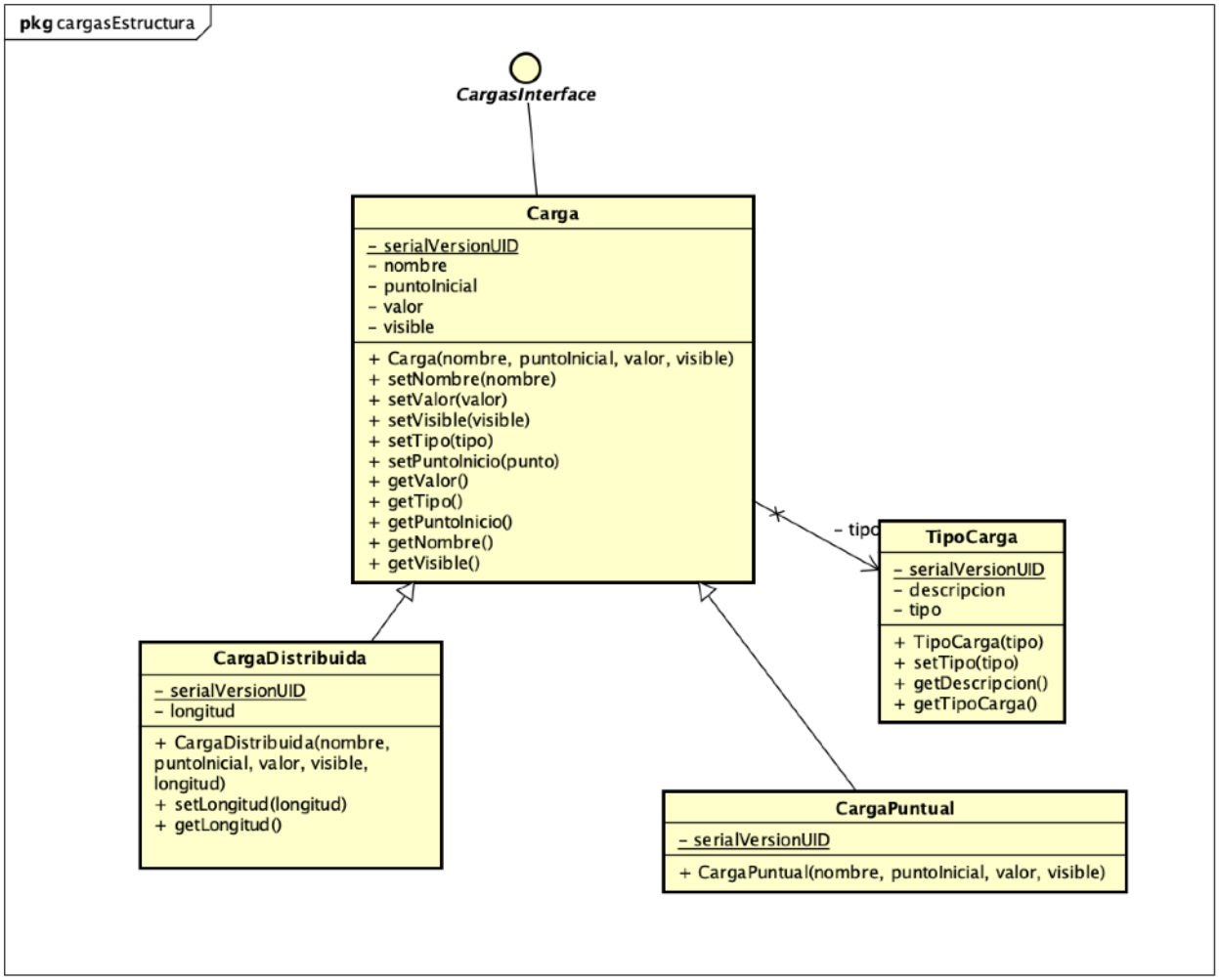


Figura 4.20. Diagrama de clases del paquete cargasEstructura.

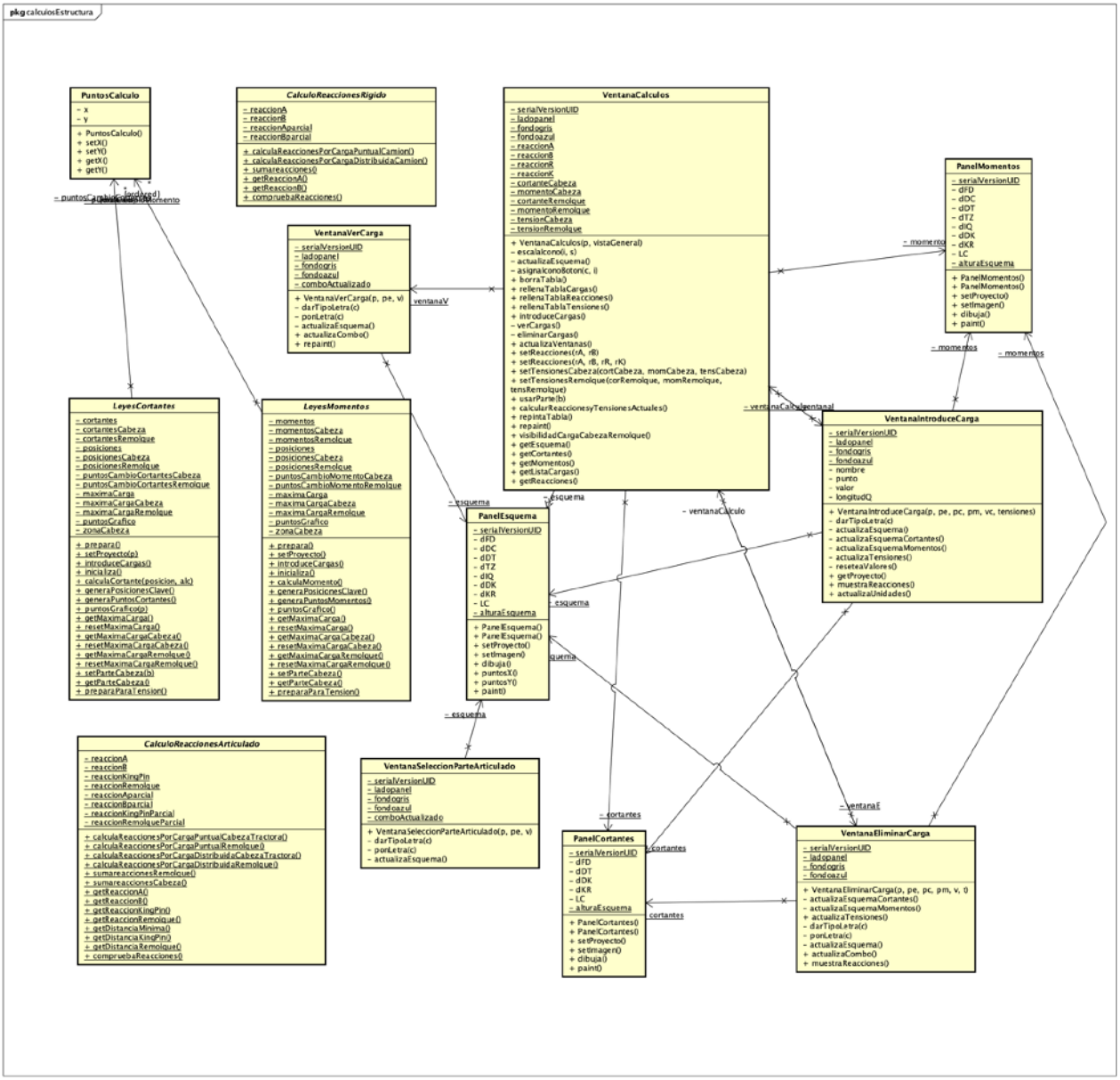


Figura 4.21. Diagrama de clases del paquete cargasEstructuras.

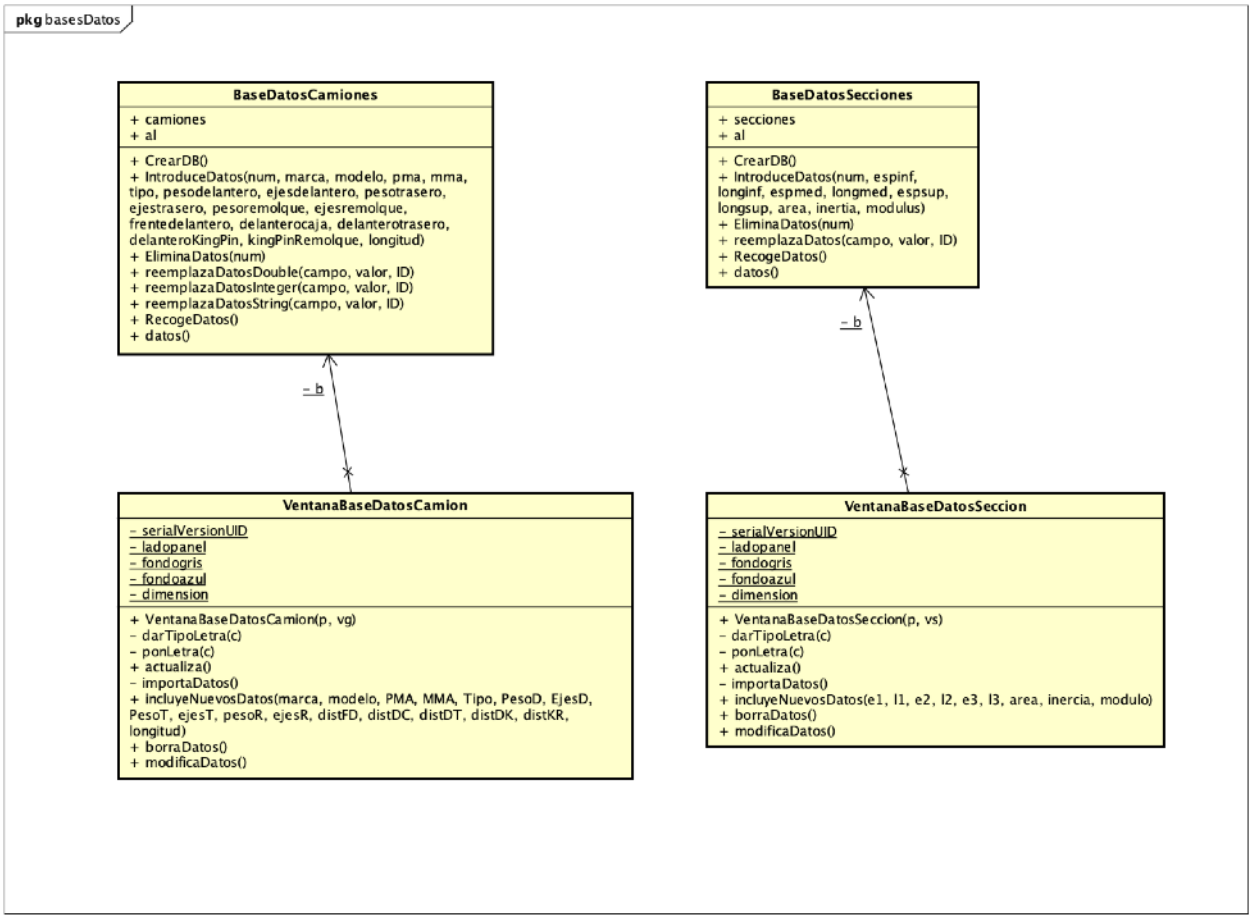


Figura 4.22. Diagrama de clases del paquete basesDatos.

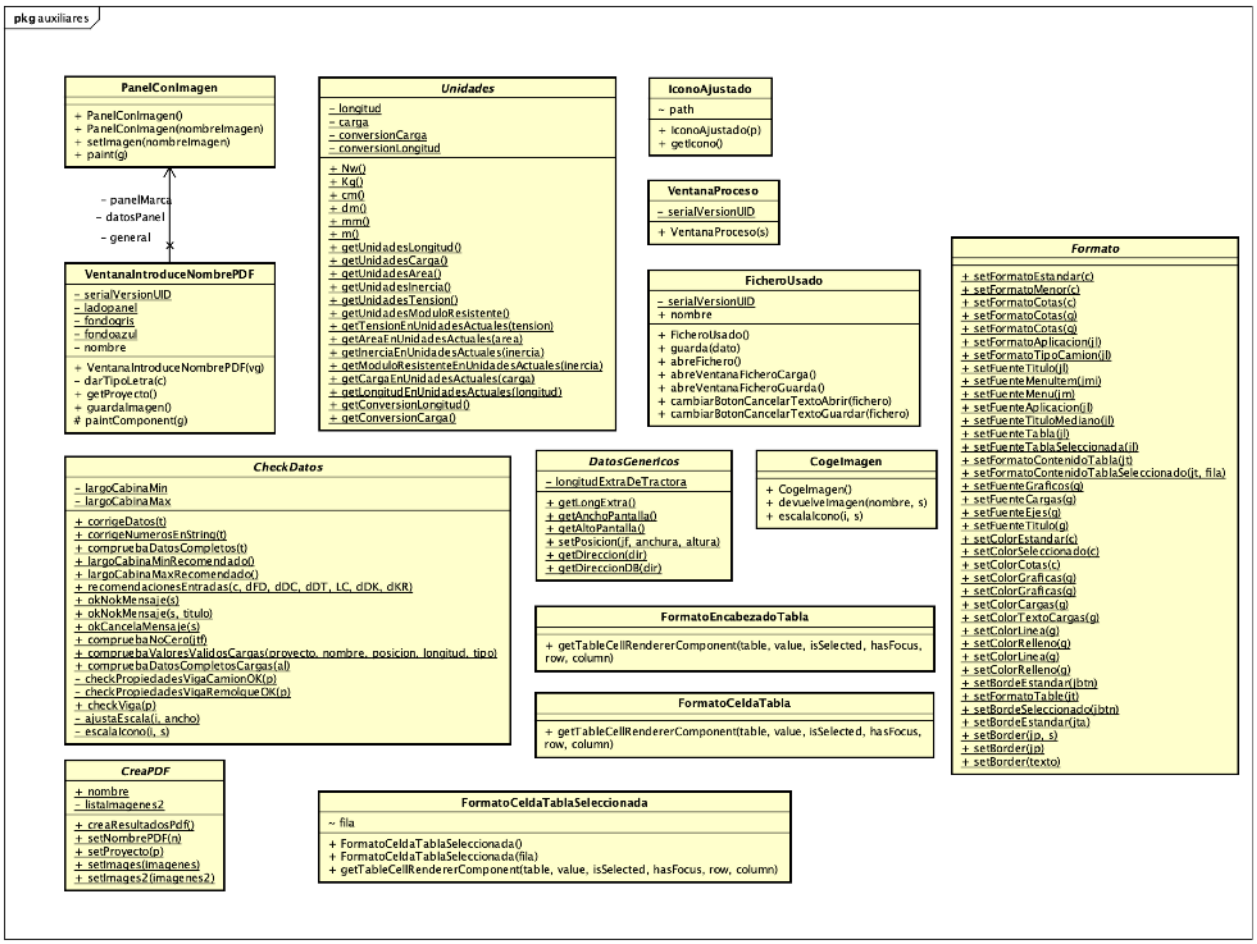


Figura 4.23. Diagrama de clases del paquete auxiliares.

4.3. Conclusiones

En este capítulo se han expuesto las decisiones constructivas adoptadas en el diseño:

- Diagramas de secuencia: Se han expuesto los diagramas de secuencia de los distintos casos de uso que exponen la interacción entre los distintos objetos que intervienen.
- Diagrama de arquitectura lógica: Se ha expuesto en él el diferente rol que desempeña cada uno de los paquetes, describiendo la funcionalidad de las clases incluidas en cada uno de ellos.
- Diagramas de clases: Se han expuesto en ellos las relaciones, herencias, implementación de interfaces, etc. que tienen implementadas las distintas clases de los diferentes paquetes mencionados en la arquitectura del sistema.

A partir de esta información se ha establecido un marco en el que organizar el código necesario para el desarrollo de la aplicación.

Capítulo 5. Implementación y pruebas

5.1. Introducción

Se describirá en lo sucesivo, dentro de este capítulo, el proceso de implementación de la solución adoptada. Esta descripción de la implementación incluirá la descripción del entorno software y hardware utilizado, así como diferentes apartados que contendrán justificaciones, descripciones o argumentaciones de las soluciones especialmente relevantes utilizadas para el desarrollo del código necesario para la aplicación.

De igual manera, se incluirán las pertinentes pruebas realizadas para comprobar la corrección del producto, incluyendo estas pruebas las unitarias, de integración y de validación.

5.2. Entorno de desarrollo

Se procede a continuación a realizar una descripción somera, pero suficientemente detallada para poder exponer los aspectos mas importantes acerca del entorno software, el entorno hardware y el lenguaje de programación utilizado.

Entorno Hardware

Se han utilizado tres computadoras distintas para la elaboración del desarrollo y pruebas de la herramienta generada.

- **MacBook Air:** Utilizado para el desarrollo principal de la herramienta y para las pruebas principales.
  - Sistema operativo MacOS Monterey v12.2.
  - Memoria RAM: 8 GB.
  - Chip Apple M1, 2020.
  - 500 GB. Almacenamiento flash.
  - Pantalla 13,3 pulgadas.

- **iMac (21,5 pulgadas, 2017):** Utilizado para la comprobación del funcionamiento y visualización en otras pantallas de mayor tamaño así como en otro tipo de configuración.
  - Sistema operativo MacOS Monterey v12.0.1.
  - Procesador Intel Core i5 2,3 GHz. de doble núcleo.
  - Memoria: RAM 8GB 2133 MHz. DDR4.
  - 251 GB. Almacenamiento flash.
  - Pantalla 21,5 pulgadas.

- **Huawei MateBook D15:** Utilizado para la comprobación del funcionamiento de la herramienta en un sistema operativo distinto al utilizado durante el desarrollo de la misma.
  - Sistema Operativo Windows 11 Home 64 bits.
  - Memoria RAM: 16 GB.
  - Chip Intel Core 1135G7 de generación 11 a 2,4GHz. y 2,42 GHz.
  - 500 GB. Almacenamiento.
  - Pantalla 15,6 pulgadas.

Entorno Software

El desarrollo de la herramienta objeto de este documento ha requerido de las siguientes herramientas para poder realizarse:

- **Eclipse IDE:** Se ha utilizado Eclipse como entorno integrado de desarrollo durante el transcurso del trabajo y del desarrollo de los sucesivos prototipos de la herramienta a generar. *"The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Maven and Gradle integration"* (Traducido directamente del inglés: La herramienta esencial para cada desarrollador

- Java, incluye un IDE Java un cliente Git, Editor XML, integración Maven y Gradle) (The Eclipse Foundation).

- **Apache Maven:** Se ha utilizado Maven para la integración de itext7 y de apache Derby en el proyecto. *"Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information"* (Traducido directamente del inglés: Apache Maven es un proyecto de gestión de proyecto y una herramienta de comprensión. Basado en el concepto de un objeto modelo de proyecto(POM). Maven puede gestionar la construcción, informe y documentación de un proyecto desde una porción central de información). (Apache Maven Project 2023).

- **Itext:** Itext se ha utilizado para la programación de las necesarias ordenes que se usan para la gestión de los archivos pdf cuando se quiere generar el anexo de resultados mencionado en capítulos anteriores. *"iText Suite refers to the complete line of products comprising the open-source iText Core PDF library and its add-ons. The iText Suite is a fully-featured SDK for PDF development that allows you to seamlessly embed extensive PDF functionality into your software or workflows"* (Traducido directamente del inglés: iText Suite es la completa linea de productos que comprenden la librería PDF y sus añadidos. iText Suite es un SDK completamente equipado para el desarrollo de PDF que permite introducir funcionalidad PDF en tu software o tus flujos de trabajo) (IText Suite by aprise).

- **Apache Derby:** Se ha utilizado Apache Derby para la gestión de la base de datos de la aplicación. Este tipo de base de datos se encuentra embebida en el proyecto y no necesita una conexión específica a un servidor, si no que el propio usuario es quien la gestiona en su aplicación. "*Apache Derby, an Apache DB subproject, is an open source relational database implemented entirely in Java and available under the Apache License, Version 2.0" (The Apache DB project)*
  - Derby no ocupa demasiado espacio. unas 3.5 megabytes para el motor base y el driver JDBC.
  - Derby es basado en Java, JDBC y estándar SQL.
  - Derby incluye un driver JDBC que permite embeber Derby en cualquier solución Java.

- **Astah:** Se ha utilizado la aplicación para la generación de los diagramas UML de la aplicación, empleando para ello la licencia de estudiante concedida por el desarrollador. *"Astah Professional is a system design tool that supports UML (Unified Modeling Language) 2.x (partly), UML1.4, Flowchart, Data Flow Diagram, ER diagram, CRUD, Requirement diagram and Mind Map*. (Traducido directamente del inglés: Astah Professional es una herramienta de diseño que soporta UML (Unified Modeling Language) 2.0 (parcialmente), UML1.4, Diagramas de flujo, Diagrama de flujo de datos, Diagrama de Entidad Relación, CRUD, Diagrama de requerimientos y Mapas mentales. (Change Vision Inc. 2009-2022)

- **Pages:** Utilizado para el desarrollo del presente documento. "*Pages es un potente procesador de textos con todo lo necesario para crear documentos tan espectaculares que entran por los ojos, y viene de serie en casi todos los dispositivos Apple" (Apple 2023).*

Lenguaje de programación

Para las distintas funcionalidades de la aplicación y la gestión del control de la misma se han utilizado durante el desarrollo:

- **Java:** Se ha utilizado Java para la programación del código de las clases java que se emplean en el desarrollo de la aplicación, suponiendo el mayor porcentaje de trabajo sobre la misma. *"Java is a programming language and computing platform first released by Sun Microsystems in 1995"* . (Traducido directamente del inglés: Java es un lenguaje de programación y una plataforma de computación publicada inicialmente por Sun Microsystems en 1995) (Java 2023). Ha evolucionado para permitir compartir gran parte del mundo digital de hoy, proveyendo la plataforma fiable que muchos servidores y aplicaciones usan como base. Mientras que la mayoría de las aplicaciones modernas combinan el entorno Java y la aplicación conjuntamente, hay muchas aplicaciones e incluso algunas páginas web que no funcionarán a menos que tengan un escritorio Java instalado (Java 2023).

- **XML:** Se ha utilizado XML para la gestión del POM (project object model) que se emplea para la gestión del proyecto Maven, utilizado para la implementación de Itext y de Apache Derby. *"XML es un lenguaje pensado para el intercambio de información. Consiste en un lenguaje de etiquetas*

mediante el cual se puede modelizar cualquier tipo de datos tan complejo como sea necesario". (desarrolloweb 2023) *"puede servir para el tránsito de los datos mediante redes como Internet y es compatible con cualquier lenguaje de programación que requiera usar esos datos"*. (desarrolloweb 2023) *"la información se puede entender fácilmente en la lectura por parte de humanos. XML es además un lenguaje muy potente, ya que a partir de él se pueden definir múltiples lenguajes, para cualquier ámbito de aplicaciones (desarrolloweb 2023).*

- **SQL:** Se ha empleado SQL para la programación de las ordenes necesarias para la gestión de la creación de las bases de datos, inclusión de nuevos registros y datos, eliminación y modificación de datos de la misma. En nuestro caso, se ha realizado un embebido total en Java y utilizado el driver correspondiente siendo el uso de SQL es totalmente ajeno al usuario. El propio código java realiza la conversión de las ordenes del usuario sobre la herramienta a lenguaje SQL en los casos en que es necesario interactuar con las bases de datos. *"SQL es un lenguaje de computación para trabajar con conjuntos de datos y las relaciones entre ellos. Los programas de bases de datos relacionales, como Microsoft Office Access, usan SQL para trabajar con datos. A diferencia de muchos lenguajes de computación, SQL no es difícil de leer y entender, incluso para un usuario inexperto. Al igual que muchos lenguajes de computación, SQL es un estándar internacional reconocido por organismos de estándares como ISO y ANSI.SQL se usa para describir conjuntos de datos que pueden ayudarle a responder preguntas. Al usar SQL, debe usar la sintaxis correcta. La sintaxis es el conjunto de reglas mediante las que se combinan correctamente los elementos de un idioma. La sintaxis SQL se basa en la sintaxis del idioma inglés y usa muchos de los mismos elementos que la sintaxis de Visual Basic para Aplicaciones (VBA)" (Microsoft 2023).*

Planteamiento de la solución con Apache Derby

Como ya se ha comentado en el punto del entorno software, Apache Derby es el gestor de base de datos relacional que se ha utilizado para el desarrollo de la herramienta objeto del presente documento.

Las razones fundamentales para la elección de Apache Derby son las siguientes:

- Está basada en Java. Es importante que nuestra gestión de bases de datos esté basada en el lenguaje de programación que se ha elegido para el desarrollo de la herramienta.

- El núcleo ocupa poco espacio. La inclusión de la gestión de la base de datos no debería suponer un requerimiento de espacio de almacenamiento muy grande en la herramienta.
- Es libre. Su utilización no implica un desembolso económico ni una inversión importante.
- Es compatible con SQL. La intención ha sido la de utilizar el lenguaje SQL a la hora del desarrollo de los statements que se utilizarán a través de Java para establecer las instrucciones que se darán en la gestión de las bases de datos utilizadas.
- Se puede embeber en la aplicación. Dada la necesidad de los usuarios de tener un acceso a bases de datos de camiones y secciones, se ha incluido la gestión de este tipo de elementos. Sin embargo, se da la circunstancia de que cada carrozado y solución es distinta en función del profesional que quiera realizar la adaptación del vehículo a sus propias necesidades. Esto hace que la base de datos utilizada sea muy específica para cada cliente, por lo que embeber la misma en la aplicación es la solución más adecuada para que cada cliente pueda particularizarla convenientemente.

Preparación e instalación

La utilización de Apache Derby es sencilla y su configuración no requiere especial complejidad. Sin embargo, en nuestro caso, previamente a su utilización es necesario preparar el proyecto para su uso. Esta preparación requiere de tres sencillos pasos que preparan nuestro IDE y MAVEN para su utilización directamente en las clases Java. Los pasos son los siguientes:

A. Importación de las librerías necesarias en el proyecto java. Para la importación se debe descargar de la página web de Apache Derby las librerías necesarias. En nuestro caso descargamos la versión db-derby-10.15.2.0-lib.
B. El siguiente paso es su inclusión en el proyecto a través del IDE Eclipse. Este paso es relativamente sencillo y no tiene más complicación que abrir la página de propiedades del proyecto haciendo click con el botón derecho sobre el mismo y seleccionando “properties”.
   Una vez abiertas las propiedades seleccionamos “Java Build Path” para posteriormente añadir la librería descargada mediante la opción “add external jar”.
   Iremos a la localización donde se encuentra nuestra librería descargada de Apache Derby y la incluiremos en nuestro proyecto. Se muestra esta información en la Figura 5.1.
C. Una vez terminado el paso anterior, en nuestro caso ha sido necesario establecer las propiedades del fichero POM de Maven para que pueda gestionar el funcionamiento de Apache Derby. Esta

gestión se realiza directamente sobre el fichero XML donde incluiremos la lineas mostradas en la Figura 5.2.

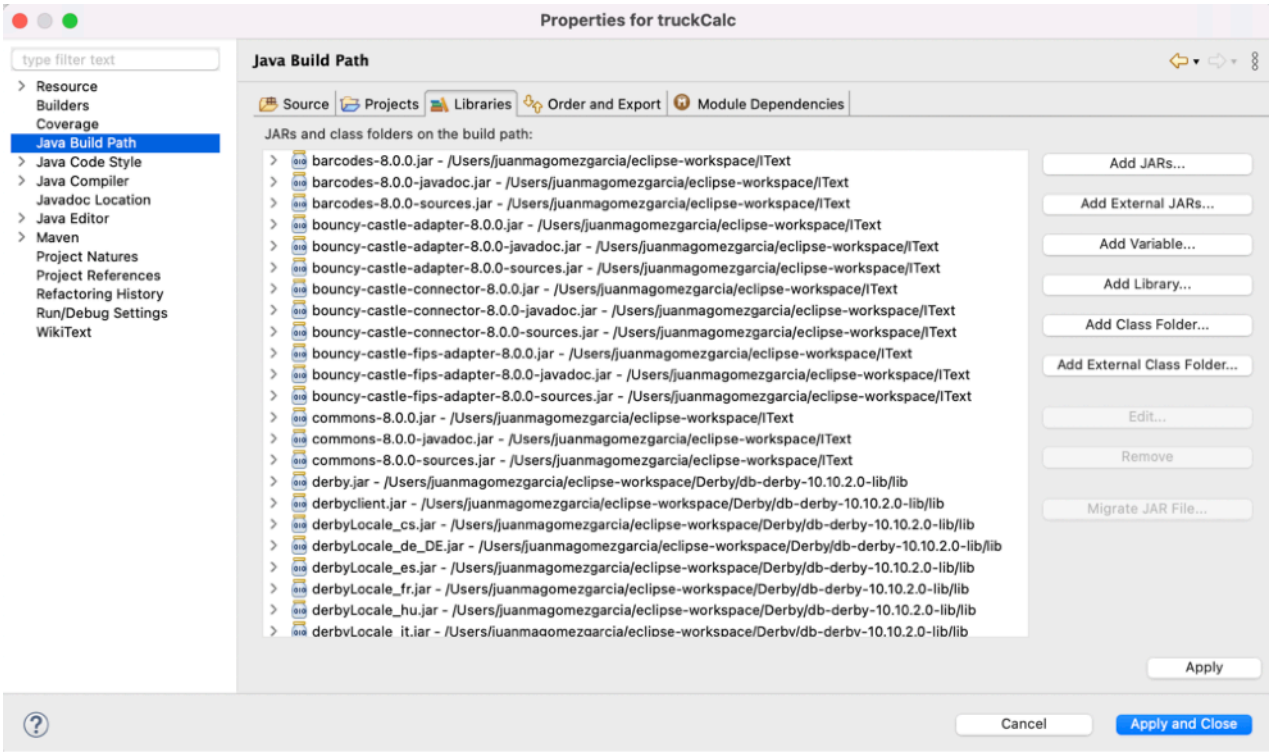


Figura 5.1. Adición de la librería de Derby Apache al proyecto

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-jar-plugin</artifactId>
            <version>3.3.0</version>
            <configuration>
                <archive>
                    <manifest>
                        <addClasspath>true</addClasspath>
                        <mainClass>camiones.TruckCalc</mainClass>
                    </manifest>
                </archive>
            </configuration>
        </plugin>
    </plugins>
</build>
```

Figura 5.2. Establecimiento de Apache en el fichero POM de Maven

Utilización

Realizados los pasos anteriores podemos pasar a utilizar Apache Derby sobre nuestra aplicación.

Lo primero que hemos realizado es la importación de las clases necesarias para la gestión de la base de datos. Dicha importación se lleva a cabo mediante las siguientes ordenes.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.Statement;
```

- La primera orden importará las clases java necesarias para la gestión de la conexión con la base de datos Apache.
- La segunda orden importa las clases encargadas de la gestión del Driver específico JDBC para nuestra base de datos. Hay que tener en cuenta que cada gestor de bases de datos tendrá su propio driver encargado de la comunicación con el tipo de base de datos que estemos utilizando. Así existe un driver para MySQL, otro driver para Oracle, etc.
- La tercera orden importa las clases encargadas de gestionar las ordenes SQL que pueden prepararse de una forma genérica para posteriormente poder modificar valores y actualizar dicha orden conteniendo los nuevos valores indicados.
- La cuarta orden importa Statement que se encargara de la gestión de ordenes SQL.

Para el uso de la base de datos exponemos aquí el método encargado de la creación de la base de datos en caso de no existir.

Como vemos en la Figura 5.3, lo primero que se hace es crear una conexión mediante la orden:

```
Connection con;
```

Obtenemos la dirección donde estará ubicada la base de datos utilizando la orden:

```
File url=new File(proyecto);
```

Indicamos que utilizaremos el driver para base de datos embebida de apache Derby mediante la orden:

```
Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
```

```
public void CrearDB() {
    //Preparamos la conexion
    Connection con;
    //Usamos el separador de archivos
    String proyecto = DatosGenericos.getDireccionDB("Vehiculos");
    File url = new File(proyecto);

    //Si no existe la creamos
    if(!url.exists()) {
        try {
            //Usamos el driver embebido de Apache Derby
            Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
            //Creamos la base de datos si no existe en la ubicacion
            String db = "jdbc:derby:"+proyecto+";create=true";
            //Generamos la conexion
            con=DriverManager.getConnection(db);

            //Preparamos el string que contendrá el statement
            String tabla = "create table Camiones( Id INT PRIMARY KEY, MARCA VARCHAR(50), MODELO VARCHAR(50), PMA DOUBLE,"
                    + "MMA DOUBLE, TIPO VARCHAR(1), PESODELANTERO DOUBLE, EJESDELANTERO INT, PESOTRASERO DOUBLE, EJESTRASERO INT,"
                    + "PESOREMOLQUE DOUBLE, EJESREMOLQUE INT, FRENTEDELANTERO DOUBLE, DELANTEROCAJA DOUBLE,"
                    + "DELANTEROTRASERO DOUBLE, DELANTKINGPIN DOUBLE, KINGPINREM DOUBLE, LONGITUD DOUBLE)";
            //Generamos el statement
            PreparedStatement ps = con.prepareStatement(tabla);
            //Ejecutamos y cerramos
            ps.execute();
            ps.close();
            CheckDatos.okCancelaMensaje("Base de Datos Creada");
        }catch (Exception ex) {
            CheckDatos.okCancelaMensaje("Error. "+ex.toString());
        }
    }
}
```

Figura 5.3. Método encargado de la creación de la base de datos en caso de no existir.

Finalmente realizamos la conexión mediante

```
con=DriverManager.getConnection(db);
```

A partir de este momento podemos generar un Statement en el que incluiremos ordenes en lenguaje SQL para la gestión de la base de datos. En el caso de la figura 5.3, el statement que pasamos es el que se muestra a continuación:

```
String tabla = "create table Camiones( Id INT PRIMARY KEY, MARCA
VARCHAR(50), MODELO VARCHAR(50), PMA DOUBLE,"
+ "MMA DOUBLE, TIPO VARCHAR(1), PESODELANTERO DOUBLE,
EJESDELANTERO INT, PESOTRASERO DOUBLE, EJESTRASERO INT,"
+ "PESOREMOLQUE DOUBLE, EJESREMOLQUE INT, FRENTEDELANTERO DOUBLE,
DELANTEROCAJA DOUBLE,"
+"DELANTEROTRASERO DOUBLE, DELANTKINGPIN DOUBLE, KINGPINREM
  DOUBLE, LONGITUD DOUBLE)";
```

En este caso generamos la orden SQL necesaria para crear la tabla Camiones que tendrá los siguientes campos:

- ID, de tipo entero y que representa la clave principal.
- MARCA, de tipo cadena de caracteres de longitud 50.
- MODELO, de tipo cadena de caracteres de longitud 50.
- PMA, de tipo double.
- MMA, de tipo double.
- TIPO, de tipo cadena de caracteres de longitud 1.
- PESODELANTERO de tipo double.
- EJESDELANTERO de tipo int.
- PESOTRASERO de tipo double.
- EJESTRASERO de tipo int.
- PESOREMOLQUE de tipo double.
- EJESREMOLQUE de tipo int.
- FRENTEDELANTERO de tipo double.
- DELANTEROCAJA de tipo double.
- DELANTEROTRASERO de tipo double.
- DELANTKINGPIN de tipo double.
- KINGPINREM de tipo double.
- LONGITUD de tipo double.

Dicha orden se genera en un objeto de tipo String, que se pasa a otro objeto de tipo PreparedStatement como argumento del método prepareStatement del objeto conexión creado anteriormente.
Se ejecutará la orden SQL mediante la instrucción:

```
ps.execute();
```

Para finalmente terminar la conexión mediante la orden:

```
ps.close();
```

Como se puede apreciar, la utilización de Apache Derby es sencilla y rápida y permite adaptaciones de las ordenes java utilizando la información que puede pasarse desde otras clases, como ocurre en el método mostrado en la Figura 5.4 donde se pasan como argumentos distintos valores, que posteriormente se introducen en la base de datos de Camiones.

```
public void IntroduceDatos(int num,String marca, String modelo, double pma, double mma, char tipo, double pesodelantero,
        int ejesdelantero, double pesotrasero, int ejestrasero, double pesoremolque, int ejesremolque, double frentedelantero,
        double delanterocaja, double delanterotrasero, double delanteroKingPin, double kingPinRemolque, double longitud) {
    //Preparamos la conexion
    Connection con;
    //Usamos el separador de archivos
    String proyecto = DatosGenericos.getDireccionDB("Vehiculos");
    File url = new File(proyecto);

    if(url.exists()) {
        try {
            //Usamos el driver embebido
            Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
            String db = "jdbc:derby:"+proyecto;
            con=DriverManager.getConnection(db);

            //Generamos el string que contendra el statement para SQL
            String tabla = "INSERT INTO Camiones VALUES ("+num+",'"+marca+"','"+modelo+"',"+
                    pma+","+mma+",'"+tipo+"',"+pesodelantero+","+ejesdelantero+","+pesotrasero+","+ejestrasero+","+
                    pesoremolque+","+ejesremolque+","+frentedelantero+","+delanterocaja+","+delanterotrasero+","+
                    delanteroKingPin+","+kingPinRemolque+","+longitud+")";
            //Generamos el statement
            PreparedStatement ps = con.prepareStatement(tabla);
            //Ejecutamos y cerramos el statement
            ps.execute();
            ps.close();
        }catch (Exception ex) {
            CheckDatos.okCancelaMensaje("No se pudieron introducir los datos. "+ex.toString());
        }
    }
}
```

Figura 5.4. Método encargado de introducir datos en la base de datos mediante valores pasados como argumentos al propio método.

En resumen, la utilización de Apache Derby, tal como se indicaba al principio de este apartado resulta muy sencilla y facilita un manejo eficiente del almacenamiento debido a su reducido tamaño.

Planteamiento de la solución con IText

Otro de los puntos especiales del proyecto, que queda fuera del uso general de Java y sus librerías por defecto es la solución propuesta por la librería IText.

La librería IText contiene las clases necesarias para nuestra herramienta a la hora de la elaboración del anexo técnico de cálculo necesario, tanto por la necesidad de generarlo en un documento pdf como por la versatilidad de la solución para la generación del pdf y la inclusión de imágenes en el mismo, aspecto éste, que resulta fundamental para la inclusión de los esquemas, diagramas de momentos flectores y diagramas esfuerzos cortantes en el documento, puesto que es requerido para la justificación del estado tensional de la estructura del vehículo.

Preparación e instalación

Al igual que ocurría con Apache Derby, Itext requiere de una instalación previa para su utilización.

Hemos de decir aquí que la instalación de las librerías tiene una complejidad algo mayor que en el caso de Apache, fundamentalmente por la preparación del fichero POM.
Itext requiere de la generación de un proyecto tipo Maven que necesita de un fichero POM para la gestión de la información contenida en el mismo. La realidad es que existen diferentes propuestas para la configuración del fichero POM y su preparación para el correcto funcionamiento de Itext, si bien, debemos decir que encontrar la propuesta adecuada no ha sido una tarea inmediata y llevó varios intentos, con distintas variantes, hasta que se llegó a la opción correcta y que finalmente ha funcionado.
En muchas ocasiones, también se encontraron problemas por las versiones de Java, Itext y otras librerías incluidas en el proyecto, por lo que se tuvo que modificar el compilador y las versiones de Itext hasta encontrar la combinación que permitía trabajar adecuadamente.
El fichero POM finalmente propuesto es el que se muestra en la imagen de la Figura 5.5, si bien en ésta imagen se ha eliminado la parte correspondiente a la configuración de Apache Derby para una mejor claridad.

Utilización

Al igual que en el caso de Apache Derby, una vez hemos terminado la instalación y la importación, podemos pasar a utilizar Itext sobre nuestra aplicación.
Una vez más, lo primero que hemos realizado es la importación de las clases necesarias como se aprecia en las siguientes instrucciones.

```
import com.itextpdf.io.image.ImageData;
import com.itextpdf.io.image.ImageDataFactory;
import com.itextpdf.kernel.pdf.PdfDocument;
import com.itextpdf.kernel.pdf.PdfWriter;
import com.itextpdf.layout.Document;
import com.itextpdf.layout.element.AreaBreak;
import com.itextpdf.layout.element.Image;
import com.itextpdf.layout.element.Paragraph;
import com.itextpdf.layout.element.Text;
```

La importación anterior permite la gestión de la salidas de imágenes, la gestión de documentos pdf, la escritura de documentos pdf, la gestión de documentos generales, la inclusión de áreas, imágenes, párrafos y Texto respectivamente.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://
maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.itextpdf</groupId>
  <artifactId>truckCalc</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>iText8 truckCalc</name>
  <description>TruckCalc carrozados</description>
  <properties> <maven.compiler.source>15</maven.compiler.source>
<maven.compiler.target>15</maven.compiler.target>

   <itext.version>8.0.0</itext.version>
</properties>
<dependencies>
      <!-- always needed -->
    <dependency>
        <groupId>com.itextpdf</groupId>
        <artifactId>kernel</artifactId>
        <version>${itext.version}</version>
    </dependency>
      <!-- always needed -->
    <dependency>
        <groupId>com.itextpdf</groupId>
        <artifactId>io</artifactId>
        <version>${itext.version}</version>
    </dependency>
      <!-- always needed -->
    <dependency>
        <groupId>com.itextpdf</groupId>
        <artifactId>layout</artifactId>
        <version>${itext.version}</version>
    </dependency>
      <!-- only needed for forms -->
    <dependency>
        <groupId>com.itextpdf</groupId>
        <artifactId>forms</artifactId>
        <version>${itext.version}</version>
    </dependency>
      <!-- only needed for PDF/A -->
    <dependency>
        <groupId>com.itextpdf</groupId>
        <artifactId>pdfa</artifactId>
        <version>${itext.version}</version>
    </dependency>
   <dependency>
       <groupId>org.slf4j</groupId>
       <artifactId>slf4j-log4j12</artifactId>
       <version>1.7.5</version>
   </dependency>
</dependencies>
</project>
```

Figura 5.5. Método encargado de introducir datos en la base de datos mediante valores pasados como argumentos al propio método.

En la Figura 5.6, se muestran diferentes ordenes del método encargado de la generación de los documentos pdf de resultados:

```
public static void creaResultadosPdf() {
    String path=nombre+".pdf";
    try {
        PdfWriter pdfw = new PdfWriter(path);
        PdfDocument pdfdoc=new PdfDocument(pdfw);
        pdfdoc.addNewPage();
        Document document = new Document(pdfdoc);

        try {
            CogeImagen ci=new CogeImagen();
            ImageData imageninic = ImageDataFactory.create(ci.devuelveImagen(DatosGenericos.getDireccion("Imagenes/pantallaprincipal/
            Image im1 = new Image(imageninic);
            document.add(im1);
        } catch (Exception e) {
            CheckDatos.okCancelaMensaje("Fallo al cargar el logo");
        };


        Paragraph p = new Paragraph("");
        document.add(p);

        //Añadimos informacion del proyecto
        //TITULO
        Text texto = new Text("ANEXO TECNICO PROYECTO CARROZADO");
        texto.setBold();
        texto.setUnderline();
        texto.setFontSize(20);
        p = new Paragraph(texto);
        document.add(p);
```

Figura 5.6. Método encargado de generar el fichero de resultados

Describimos a continuación lo que realiza el método:

- Inicialmente se genera una ruta para el fichero a la que se añade la extensión “pdf”.
- Creamos un objeto PdfWriter al que le pasamos la ruta en la que escribir.
- Generamos un nuevo objeto PdfDocument al que pasamos el argumento PdfWriter creado.
- Invocamos al método addNewPage() sobre el objeto PdfDocument mediante el que generaremos una nueva página.
- Generamos un nuevo objeto Document al que pasamos el objeto PdfDocument como argumento.

Las siguientes ordenes son representativas de la forma de incluir distinta información sobre el archivo pdf que estamos generando.

- Imagenes

En primer lugar creamos un objeto CogeImagen que se encargará de la preparación de la imagen que querremos incluir en el archivo pdf.

Generamos un objeto ImageData, utilizando la imagen preparada por el método CogeImagen, con la ruta pasada.

Generamos un objeto Image con el objeto ImageData anterior.
Añadimos la imagen mediante el método add del objeto Document.

- Párrafos

Generamos un nuevo párrafo sin más que crear un objeto Paragraph, al que pasaremos la información en texto que queremos que incluya. En nuestro el primer objeto Paragraph se usa para generar un salto de párrafo y dejar espacios en el documento.
Añadimos el párrafo mediante el método add del objeto Document.

- Texto

Generamos un objeto Texto al que pasamos como argumento la cadena que debe contener y posteriormente realizamos ajustes de estilo sobre el mismo.
Generamos un objeto párrafo, pero en esta ocasión añadimos el objeto Texto que creamos para que lo incluya en este párrafo.
Añadimos el párrafo mediante el método add del objeto Document.

- Diagramas de la aplicación

Esta es, como decíamos, la parte más importante del documento y también la parte que más ha costado incluir en el documento pdf, ya que se producían no pocos errores durante la transmisión de la información y también requería de bastante código adicional si queríamos incluirlo sin usar la ventana de cálculos.
Finalmente se decidió, por eficiencia, reutilizar el código de la ventana de cálculos para generar las imágenes. Para ello generaremos un objeto BufferedImage que obtendrá la información de la ventana que contiene los esquemas correspondientes y lo incluirá en el archivo.
La Figura 5.7 muestra la parte de código utilizada para preparar las imágenes. En este caso se genera, para cada diagrama, un JPanel obtenido del correspondiente de la ventana cálculos usada en la aplicación. A este JPanel se le cambia el fondo a un color blanco para mayor claridad en el documento pdf y un borde para una visibilidad más agradable por quién necesite revisarlo. Posteriormente, se obtiene de este JPanel la imagen y se pasa como argumento al método de generación de pdf, que se encargará de incluir cada una como en el caso anteriormente explicado para las imágenes.

```
//Preparamos el panelEsquema
PanelEsquema pe = ventCalc.getEsquema();
pe.setImagen(DatosGenericos.getDireccion("Imagenes/fondos/fondo-blanco.jpg"));
pe.setBorder(javax.swing.BorderFactory.createLineBorder(Color.BLACK));
pe.setVisible(true);
BufferedImage imageE = new BufferedImage(pe.getWidth(), pe.getHeight(), BufferedImage.TYPE_INT_RGB);
Graphics2D graphics2De = imageE.createGraphics();
pe.paint(graphics2De);

//Preparamos el panelCortantes
PanelCortantes pc = ventCalc.getCortantes();
pc.setImagen(DatosGenericos.getDireccion("Imagenes/fondos/fondo-blanco.jpg"));
pc.setBorder(javax.swing.BorderFactory.createLineBorder(Color.BLACK));
pc.setVisible(true);
BufferedImage imageC = new BufferedImage(pc.getWidth(), pc.getHeight(), BufferedImage.TYPE_INT_RGB);
Graphics2D graphics2Dc = imageC.createGraphics();
pc.paint(graphics2Dc);

//Preparamos el panelMomentos
PanelMomentos pm = ventCalc.getMomentos();
pm.setImagen(DatosGenericos.getDireccion("Imagenes/fondos/fondo-blanco.jpg"));
pm.setBorder(javax.swing.BorderFactory.createLineBorder(Color.BLACK));
pm.setVisible(true);
BufferedImage imageM = new BufferedImage(pm.getWidth(), pm.getHeight(), BufferedImage.TYPE_INT_RGB);
Graphics2D graphics2Dm = imageM.createGraphics();
pm.paint(graphics2Dm);

ArrayList<BufferedImage> albi = new ArrayList<BufferedImage>();
albi.add(imageE);
albi.add(imageC);
albi.add(imageM);
```

Figura 5.7. Método encargado de obtener los diagramas para el fichero de resultados

Compilación y generación del ejecutable

Es remarcable que, en este caso y dado que se está utilizando un proyecto Maven, la compilación y generación de los archivos *.jar debe realizarse mediante otra secuencia de acciones que permitan un proceso correcto.

El proceso es el siguiente:

- Se pulsa sobre el botón derecho sobre el proyecto y se selecciona la opción “Run As” como se expone en la Figura 5.8.
- Se selecciona “Maven clean”. Esta opción eliminará las configuraciones utilizadas hasta el momento para preparar la nueva compilación.
- Se selecciona “Maven install”. Esta opción instalará la información necesaria para gestionar las librerías de Apache Derby, IText y otras necesarias.
- Se selecciona “Maven build…” y a continuación se escribe “compile” en la ventana emergente y se acepta.

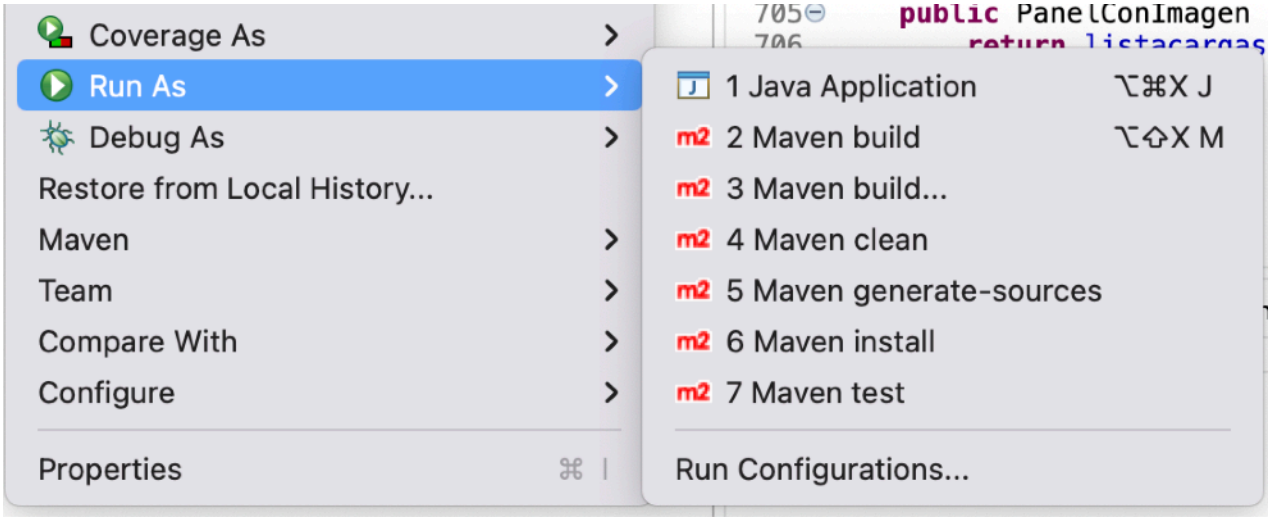


Figura 5.8. Compilación del proyecto

Una vez realizados estos pasos ya podemos ejecutar el proyecto.

Si necesitamos realizar un proyecto jar ejecutable, deberemos añadir un paso más a esta secuencia. La forma de hacerlo es la siguiente:

- Se pulsa sobre el proyecto con el botón derecho y se selecciona “Export”.
- Aparece el menú de la Figura 5.9 en el que se seleccionará “Runnable JAR file” y se designará el nombre con el que queremos que se guarde.

5.3. Implementación de la solución

La herramienta se ha planteado en varias etapas evolutivas, en las que se han ido obteniendo sucesivos prototipos funcionales. Dichos prototipos han permitido revisiones del diseño al analizar los resultados obtenidos y comparar con las especificaciones previstas, realizar las modificaciones necesarias, rediseñar y evaluar las pruebas necesarias para su validación posterior.

Los sucesivos prototipos realizados tienen que ver con las siguientes etapas:

Introducción de datos genéricos del problema a analizar

Como se mencionaba en la etapa de requerimientos, una de las características fundamentales a tener en cuenta ha sido el hecho de que el uso de la herramienta debería ser sencillo y rápido pero debería añadirse a la solución una interfaz agradable.

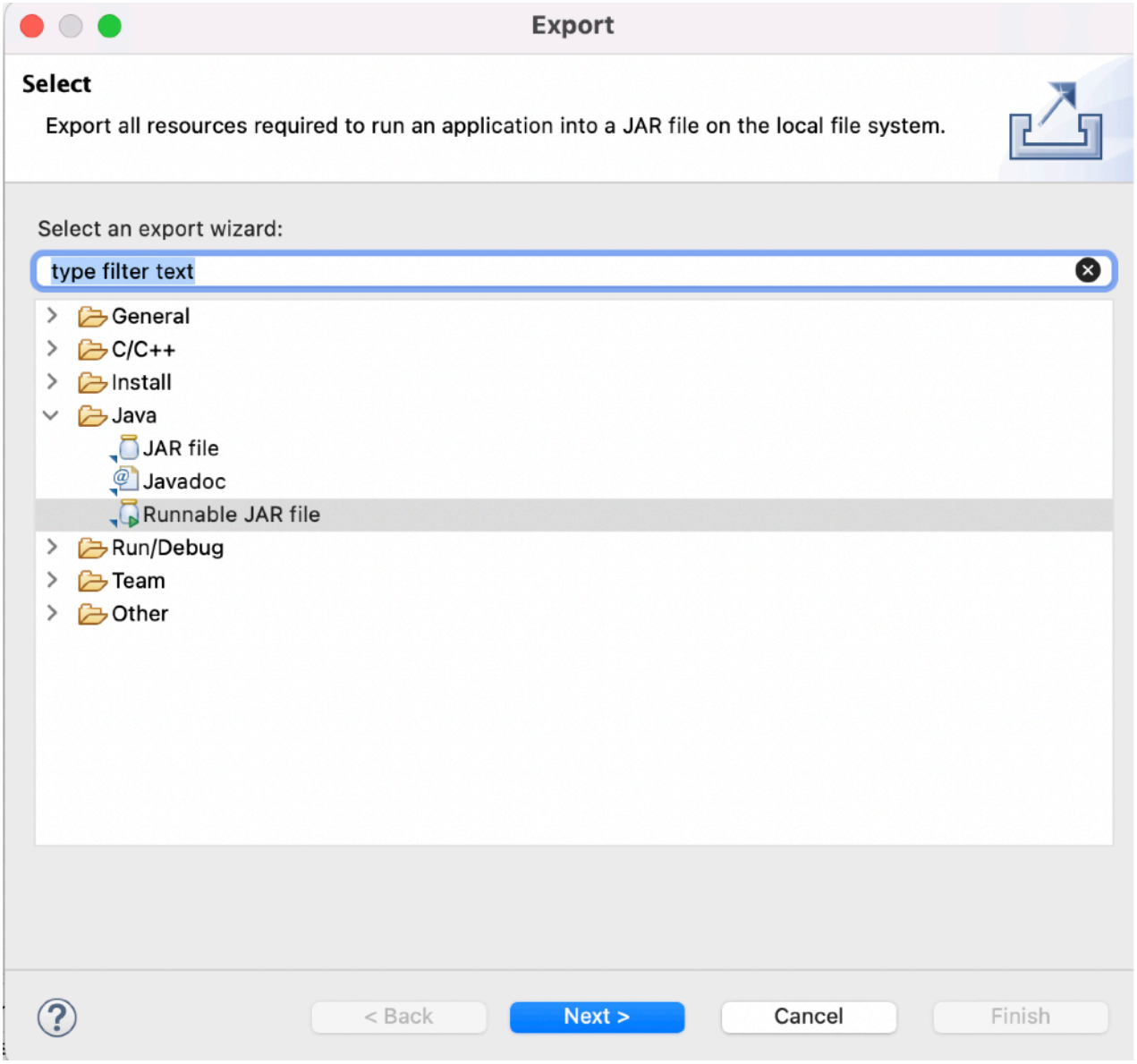


Figura 5.9. Generación del jar ejecutable

La etapa de introducción de datos es, con seguridad, la que requiere de un planteamiento estético y de usabilidad mas importante, debido a la necesidad de introducir la información de los vehículos que se van a evaluar. Esta información requiere de datos genéricos, datos dimensionales, datos de cargas, datos de tipología de vehículo, etc. Por lo que su planteamiento ha requerido de un diseño cuidado de la ubicación de los elementos a rellenar sobre la misma.

Este hecho ha llevado realizar varias etapas de diseño de la interfaz y la etapa de introducción de los datos desde un formato papel al diseño final que se empleará en la herramienta.

Las principales consideraciones sobre el diseño de la aplicación en este caso han sido:

- La introducción de los datos debería hacerse de izquierda a derecha y de arriba abajo.
- La división de los datos debería ser por bloques, conteniendo un bloque para la información general, un bloque para la información dimensional, un bloque para la información de tipología de vehículo y un bloque para otras informaciones.
- El uso de colores para la aplicación debería ser constante y no suponer una carga visual al usuario durante su uso, por lo que se han usado colores fríos.
- La aplicación debería inicialmente dar algún detalle de lo que se está haciendo por si perdiésemos en algún momento la conciencia de la información que se ha introducido y lo que hemos hecho. Es decir, un histórico de instrucciones. Aunque inicialmente este histórico no es importante, en sucesivas versiones sí podrían introducirse opciones que requieran de un seguimiento mayor.
- El acceso a los sucesivos pasos de la aplicación debería ser "guiado" por la aplicación cuando se accede a la Interfaz Gráfica de Usuario.
- El diseño de la presentación debería requerir de pocas modificaciones en caso de que posteriormente deseemos realizar versiones para móvil o tableta, por lo que la mayoría de la información debería presentarse en forma de una pequeña colección de iconos visuales con un tamaño suficientemente amplio y dejar la mínima cantidad posible en la barra de menús.

Una vez analizadas estas consideraciones se ha realizado un diseño inicial como el que puede observarse en la Figura 5.10.

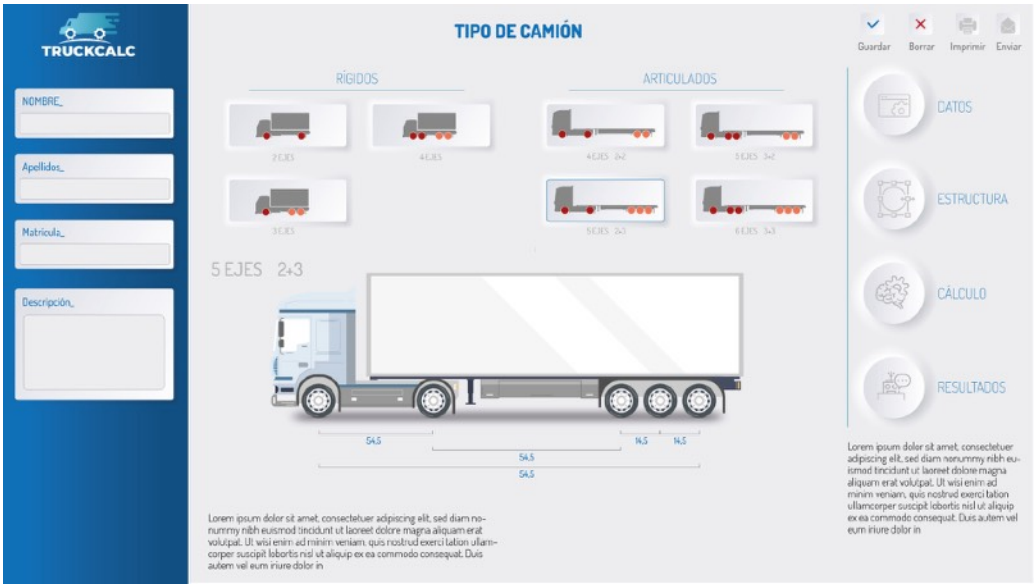


Figura 5.10. Diseño inicial de la interfaz gráfica de usuario de la herramienta

Sobre este diseño podemos comentar lo siguiente:

- A la izquierda aparecen los datos generales del vehículo ordenados de arriba a abajo.
- En el centro, en la parte superior, aparece la clasificación de posibles vehículos ordenada en dos bloques (rígidos y articulados) que permiten elegir la tipología así como el tipo de vehículo, con iconos de tamaño mediano que representan de manera sencilla la solución que implica cada uno para una guía más rápida al usuario de la opción a utilizar.
- En el centro, en la parte inferior, aparece una imagen de la solución con la que se va a trabajar cuando se selecciona el vehículo necesario.
- A la derecha aparecen, en la parte superior, opciones más generales. En la parte inferior, ordenados de arriba a abajo, se encuentran los pasos que el usuario va a tener que seguir para la realización del proyecto.

Sobre este diseño (se trata de una imagen previa a la generación de la aplicación, no de la GUI real) se han realizado modificaciones posteriores para adaptarnos más aún a lo que se requiere en la herramienta.

La Figura 5.11 muestra la implementación final de esta parte de la herramienta:

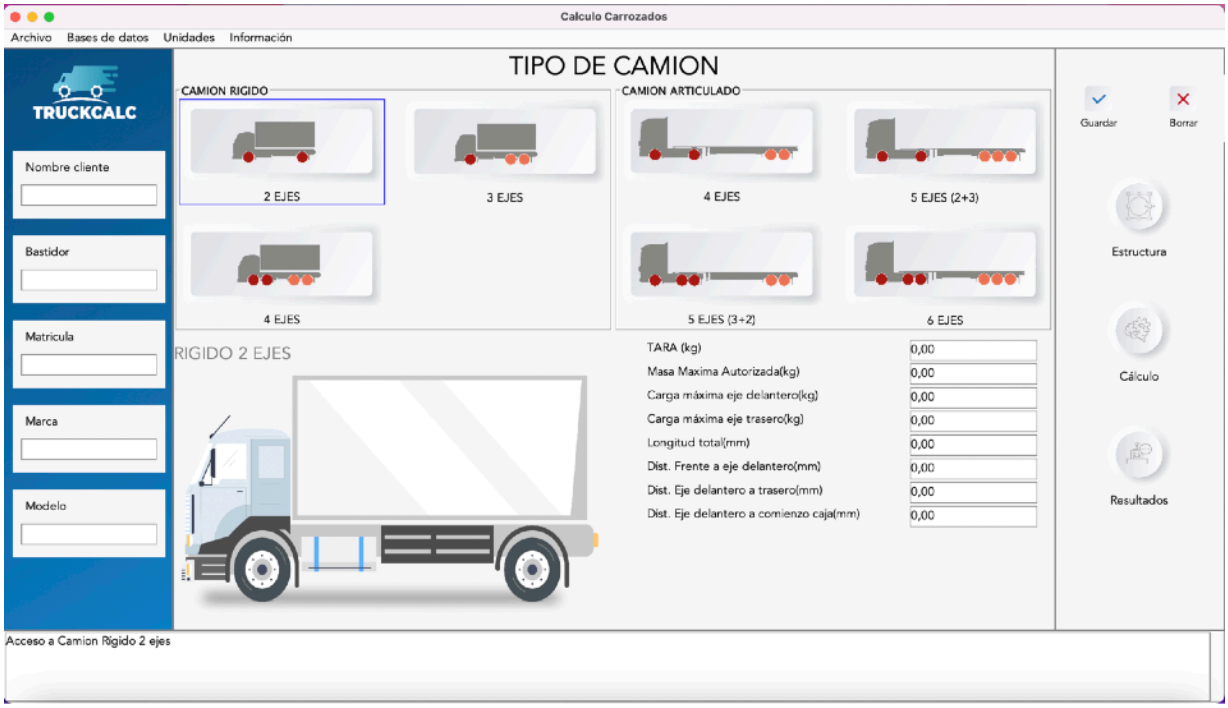


Figura 5.11. Implementación de la fase de introducción de datos de la herramienta

Como vemos se han producido algunos cambios que se reflejan a continuación:

- La zona de la izquierda se completa con datos más específicos para el proyecto, como el número de bastidor o la marca y modelo del camión, desapareciendo la descripción que se ha considerado innecesaria para los propósitos de la herramienta.
- La zona central inferior se desdobla en dos partes. Una parte izquierda contiene la imagen del camión y una parte derecha permite la introducción de los datos del camión.
- En la parte inferior de la pantalla se ha introducido un área de información para contener el histórico comentado anteriormente.
- La zona de la derecha elimina las opciones de enviar o imprimir así como el botón de datos, ya que éstos se han introducido en la parte central para poder facilitar referirse al camión seleccionado.
- Se introducen determinados menús con funciones adicionales en la zona superior.

Una vez se han rellenado los datos y se presiona el botón guardar, el sistema almacena la información necesaria para proseguir con la gestión de la información.

Generación y utilización de secciones

La segunda parte importante de la herramienta, es la gestión de secciones para su utilización en la aplicación, su cálculo y su administración para posible utilizaciones posteriores. La parte de la herramienta encargada de esta tarea es la de estructura.

La Figura 5.12 expone la gestión de las secciones de trabajo con la vigas.

En la figura podemos ver las siguientes áreas:

- En la zona izquierda la zona de introducción de datos. Esta zona se utiliza para definir las medidas de la sección a utilizar.
- La zona derecha superior contiene un esquema gráfico de la sección introducida.
- La zona inferior provee la información técnica de Area, Inercia y Modulo Resistente.
- La zona superior provee un área de menú para otras tareas.

Cálculo de cargas

La parte de la herramienta dedicada al cálculo de cargas supone el corazón de la misma. En primer lugar, porque el cálculo de la estructura es crítica para la seguridad de las vías públicas y en

segundo lugar porque es crítica para la seguridad el propio vehículo, por lo que el correcto análisis del estado tensional.

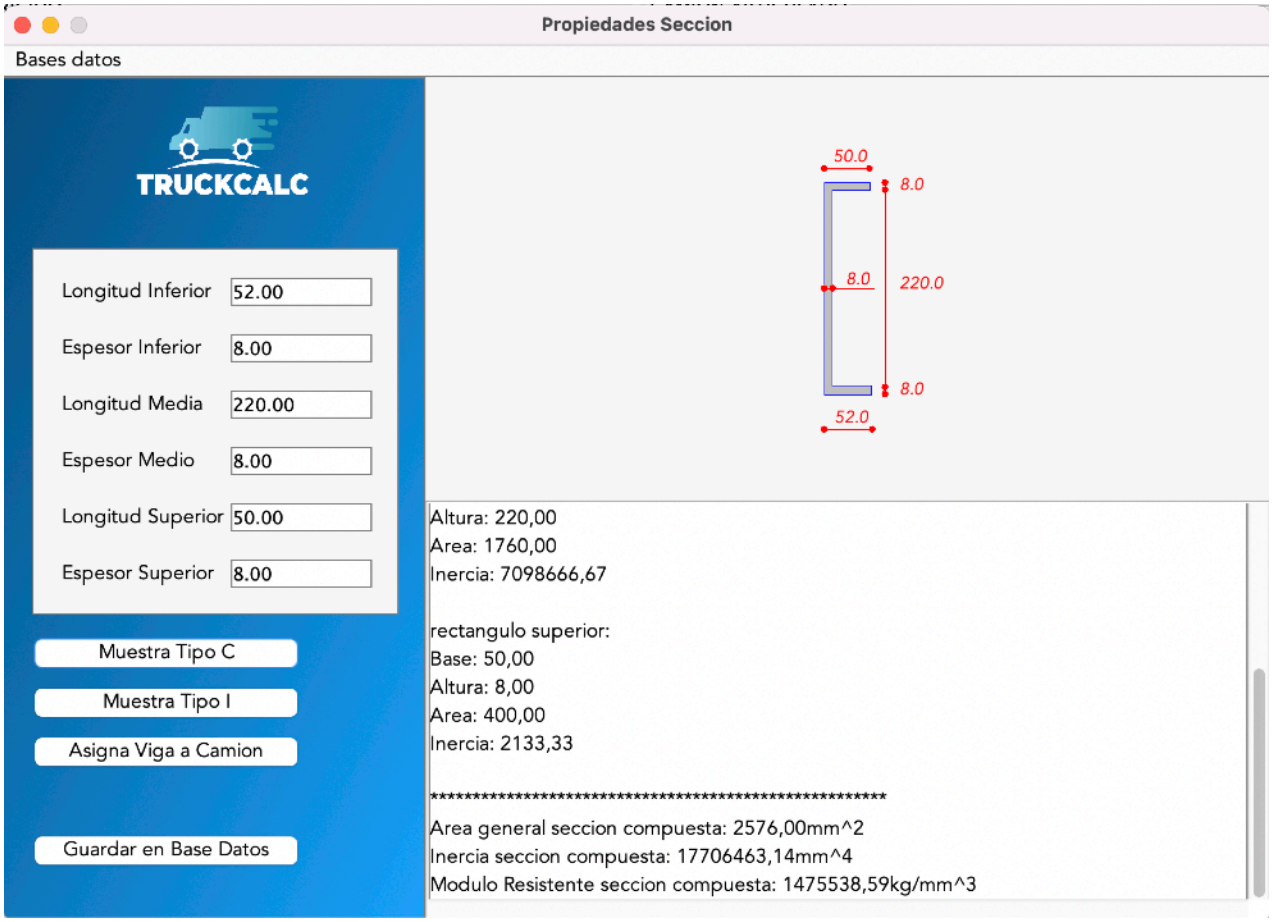


Figura 5.12. Implementación de la fase de definición de la estructura resistente

Tal como se requería en las especificaciones, se ha preparado una interfaz gráfica donde el usuario pueda tener acceso visual a todos los resultados de un único vistazo. Se puede apreciar la misma en la Figura 5.13.

Tenemos una pantalla dividida en una zona derecha e izquierda y una pequeña zona de actuación con las cargas en la zona central superior.

- La zona izquierda superior incluye el esquema del cálculo a realizar, dispone las dimensiones del camión acotadas convenientemente para que el usuario pueda ver si la información es correcta y coincide con la que se desea calcular

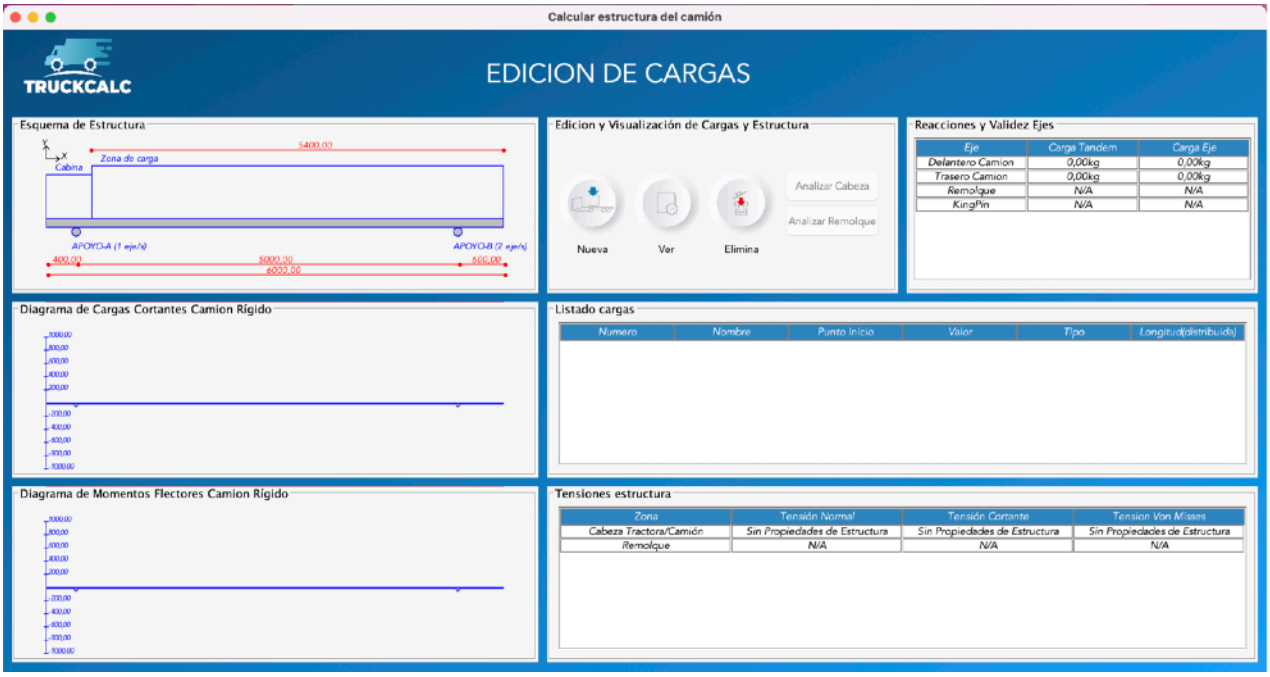


Figura 5.13. Implementación de la fase de cálculos

- La zona izquierda central incluye el diagrama de esfuerzos cortantes necesario para evaluar las cargas y tensiones totales sobre las vigas.
- La zona izquierda inferior incluye el diagrama de momentos flectores, necesario para evaluar las cargas y tensiones totales sobre las vigas.
- La zona derecha incluye una zona superior dividida en dos mitades. La parte izquierda contiene una zona de interacción, con los esquemas y cargas de manera que permite la introducción, eliminación y visualización de cargas mientras que la parte derecha recoge las cargas máximas sobre ejes, que también deberán controlarse en el proyecto de carrozado.
- La zona derecha central contiene la lista de cargas introducidas para una mejor compresión y revisión de la información gestionada.
- La zona derecha inferior contiene el resultado de tensión total sobre las vigas del vehículo.

De esta forma, conseguimos tener en pantalla toda la información necesaria para poder comprobar si la estructura del vehículo es capaz de soportar las cargas a las que se verá sometida, así como qué zona del vehículo es la que más está sufriendo en términos de tensión para considerar reforzarla.

Generación del archivo de resultados

Los resultados de los cálculos deben adjuntarse en un anexo técnico en la memoria del proyecto de carrozado a la ITV para su posterior evaluación y aprobación.

El anexo deberá contener la información básica del vehículo, así como los resultados de los diagramas de momentos y las cargas en ejes.

Esta parte del trabajo no debería suponer más que una recopilación de la información ya gestionada y calculada, por lo que sólo incluye un sencillo menú para asignación de un nombre y preparar el archivo pdf correspondiente.

El comienzo del documento se encarga de presentar la información del vehículo dividiéndola en tres partes. La primera parte contiene tal como nombre del cliente, matrícula, número de bastidor, modelo, marca y tipo de camión.

La segunda parte incluye información relativa a las dimensiones del camión.

La tercera parte incluye información de la carga máxima admitida en los ejes del camión.

La cuarta parte incluye información de la carga máxima del camión.

Tras estas informaciones se incluyen los resultados que se dividen en:

- Cargas máximas sobre los ejes del camión.
- Esquemas y Diagramas de resultados.

En la Figura 5.14, Figura 5.15 y Figura 5.16 se muestra cada una de las zonas del fichero.

Bases de datos

En las zonas donde se realiza gestión con las bases de datos se ha introducido un menú superior para el acceso a las mismas.

La presentación de los datos se realiza utilizando tablas que permitan la gestión de la información de una manera rápida y clara.

Las bases de datos generadas son dos:

- Base de datos de camiones.
- Base de datos de secciones.

En ambos casos la organización de los datos, presentación, gestión de inserción, eliminación o inserción de datos se ha desarrollado de idéntica forma para permitir una utilización más sencilla al usuario.

ANEXO TECNICO PROYECTO CARROZADO

Datos del Proyecto

Nombre:

Matrícula:

Bastidor:

Marca: IvecoNaranja

Modelo: DEXTERCLAYMORE2

Tipo de Camion: Camion Rigido

Datos Generales de Dimensiones del Camion

Distancia Frente - Eje delantero: 400.0mm.

Distancia Eje Delantero - Comienzo de Caja: 600.0mm.

Distancia Eje Delantero - Eje Trasero: 5000.0mm.

Longitud Total del Camión: 6000.0mm.

Datos Generales de Cargas en Ejes del Camión

Carga Maxima Tandem Delantero: 7500.0Kg.

Ejes en Tandem Delantero: 1

Carga Maxima Tandem Trasero: 19500.0Kg.

Ejes en Tandem Trasero: 2

Figura 5.14. Fichero de resultados. Información del vehículo. Partes primera, segunda y tercera

Datos Generales de Cargas Maximas del Camión

Masa Maxima Autorizada: 19505.0Kg.

Peso Maximo Autorizado: 20010.0Kg.

Lista de Cargas del Proyecto

Numero	Nombre	Punto Inicio	Valor	Tipo	Longitud(distribuida)
1	CARGA1	2000,00mm.	1000,00kg.	Carga Puntual	N/A
2	CARGA2	3000,00mm.	1,00kg/mm.	Carga Distribuida	1000,00mm.

Resultados del proyecto

Reacciones del Proyecto

Eje	Carga Tandem	Carga Eje
Delantero Camion	1060,00kg	1060,00kg
Trasero Camion	940,00kg	470,00kg
Remolque	N/A	N/A
KingPin	N/A	N/A

Figura 5.15. Fichero de resultados. Información del vehículo. Parte cuarta y resultados

ESQUEMAS Y DIAGRAMAS

Del Camión

Esquema del Camion

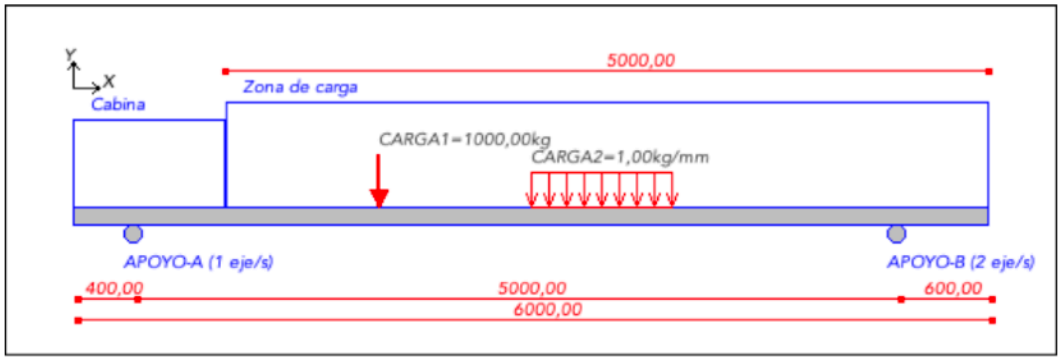


Diagrama de Cortantes del Camión

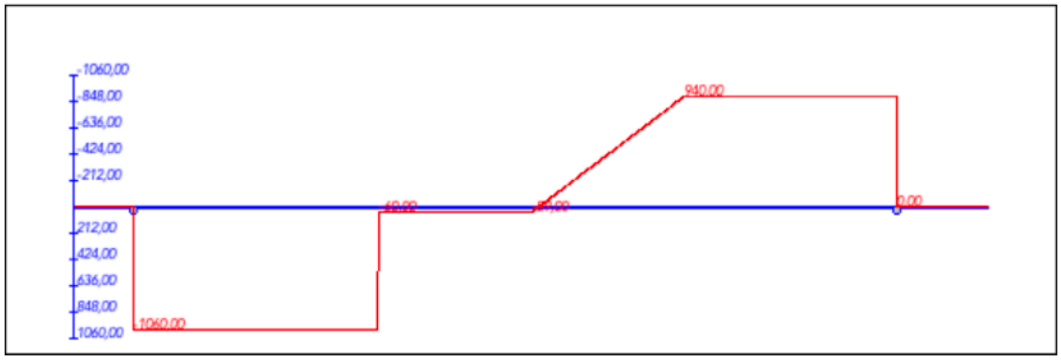


Diagrama de Momentos del Camión

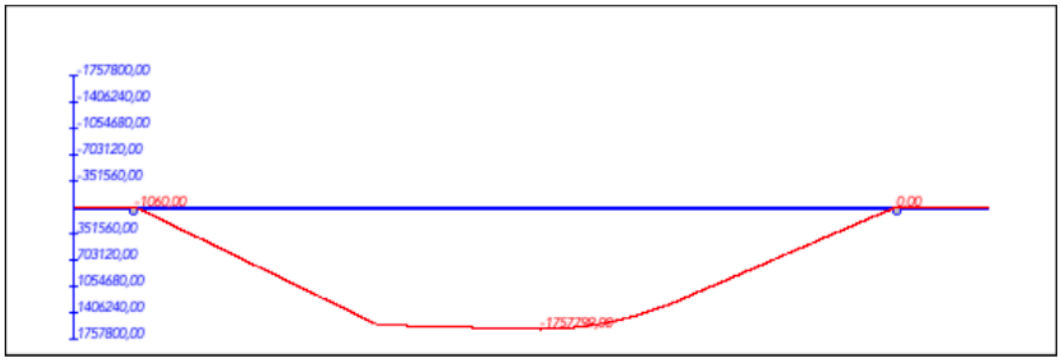


Figura 5.16. Fichero de resultados. Información del vehículo. Esquemas y diagramas

A continuación se muestra en la Figura 5.17 la de camiones y en la Figura 5.18 la de secciones.

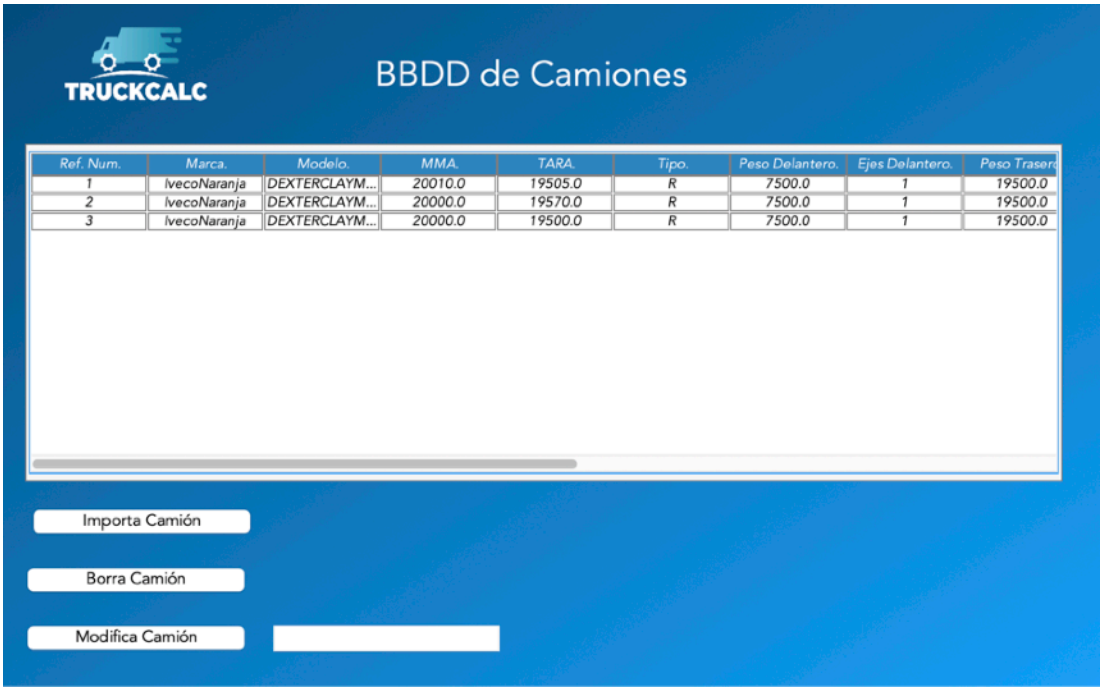


Figura 5.17. Base de datos de camiones

Figura 5.18. Base de datos de secciones

5.4. Soluciones críticas durante la codificación

Se presenta en este punto la información relativa a las soluciones de código específicamente aplicadas en esta herramienta y que tienen una importancia especialmente alta para el correcto o eficiente funcionamiento de la misma.

Se plantean aquí dos soluciones importantes adoptadas durante la fase de codificación:

- Generación del cálculo de los esfuerzos cortantes y los momentos flectores.
- Cálculo de secciones compuestas.

5.4.1. Generación del cálculo de los esfuerzos cortantes y los momentos flectores

La generación del cálculo de los esfuerzos cortantes y momentos flectores sigue una metodología muy claramente establecida en el cálculo de estructuras.

Reacciones.

El primer paso es la resolución de las ecuaciones de equilibrio de fuerzas y momentos de cuyo resultado se obtendrán las reacciones en los ejes. Estos valores de reacciones en los ejes también deberán utilizarse en el cálculo de los esfuerzos cortantes y momentos.

El cálculo de las reacciones puede realizarse de varias formas pero resultaría innecesariamente complejo aplicar formulaciones y teorías complejas para la situación en la que nos encontramos. Resulta más sencillo resolver este punto aplicando el principio de superposición, que es posible cuando el comportamiento de la viga es lineal, es decir, cuando la deformación es proporcional a la fuerza.

En este caso, si tenemos un conjunto de cargas o acciones aplicados sobre una estructura, el efecto de todas ellas es equivalente a la suma de los efectos de cada acción por separado.

De esta forma podemos plantear el cálculo como una iteración sobre una lista de cargas, encontrando para cada una el resultado de la reacción y sumar dichos resultados.

Si además, aplicamos la simplificación que se indicó en capítulos anteriores donde consideramos únicamente dos apoyos, obtendremos el resultado para las dos reacciones.

En efecto, el planteamiento realizado calculará la reacción parcial sobre la viga producido por cada carga como se ve a continuación en la Figura 5.19.

```java
//Calculamos para una carga puntual dada las reacciones en camion
public static void calculaReaccionesPorCargaPuntualCamion(Carga q, Proyecto proyecto) {
    double valor=q.getValor();
    double posicion=q.getPuntoInicio();
    double posicionA=proyecto.getCamion().getdistFrenteEjeDelantero();
    double posicionB=proyecto.getCamion().getdistFrenteEjeDelantero()+proyecto.getCamion().getdistEjeDelanteroEjeTrasero();

    //Hacemos sumatorio de cargas verticales=0.
    //Supongo la direccion de la carga como sentido negativo y las reacciones dirigidas hacia el sentido positivo
    //-valor+reaccionA+reaccionB=0
    //valor=reaccionA+reaccionB

    //Hacemos sumatorio de momentos=0
    //-valor*posicion+reaccionA*dFD+reaccionB*(dFD+dFT)=0

    //Despejando
    reaccionBparcial=valor*(posicion-posicionA)/(posicionB-posicionA);
    reaccionAparcial=valor-reaccionBparcial;
}
```

Figura 5.19. Método simplificado de cálculo de los efectos sobre una sola carga sobre la estructura.

Como vemos en los comentarios, realizamos la resolución del sistema de ecuaciones para las fuerzas y los momentos para conseguir el valor parcial de la carga.

Posteriormente realizaremos el cálculo de las reacciones sin más que realizar la suma de todos los efectos de cada una de las cargas como se muestra en la Figura 5.20.

```java
//Aplicando el principio de superposicion vamos sumando las reacciones parciales para obtener la total.
public static void sumareacciones(Proyecto proyecto) {
    reaccionA=0;
    reaccionB=0;
    ArrayList<Carga> al= proyecto.getCargas();
    Iterator<Carga> it = al.iterator();
    Carga q=null;
    while(it.hasNext()){
        q=it.next();
        if(q.getTipo().getTipoCarga()=='P') {
            calculaReaccionesPorCargaPuntualCamion(q,proyecto);
            reaccionA=reaccionAparcial+reaccionA;
            reaccionB=reaccionBparcial+reaccionB;
        } else {
            calculaReaccionesPorCargaDistribuidaCamion(q,proyecto);
            reaccionA=reaccionAparcial+reaccionA;
            reaccionB=reaccionBparcial+reaccionB;
        }
    }
}
```

Figura 5.20. Método de iteración por cada una de las cargas para la obtención de la reacción total.

Aplicamos el principio de superposición para obtener cada uno de los efectos de cada una de las cargas teniendo en cuenta su tipología (puntual o distribuida).

Conseguimos, de esta manera, una resolución completa sin necesidad de introducir la complejidad de cálculo matricial, que habría requerido más complicación al tener que trabajar con matrices de rigidez y deformación.

Esfuerzos Cortantes

El segundo paso sería el cálculo de los esfuerzos cortantes. En esta fase se va a preparar también la información necesaria para el posterior dibujo de los diagramas correspondientes.

El cálculo de los cortantes, se plantea igualmente simplificado dada la bidimensionalidad de la viga de la estructura planteada (se calcula para una viga lineal y no como una estructura tridimensional).

En este caso el calculo suele plantearse dividiendo la viga en tramos para los cuales se iría calculando el efecto acumulado de los esfuerzos cortantes.

Como ejemplo, en la viga de la Figura 5.21, se realizarían las evaluaciones de las situación de esfuerzos cortantes de manera progresiva avanzando hasta las cargas y reacciones correspondientes. Así, inicialmente se calcularía el valor de los cortantes en el tramo A. Posteriormente se plantearía la longitud B. A continuación se llegaría hasta la carga marcada por la longitud C para continuar con el planteamiento en la longitud D y terminaría considerando la viga completa.

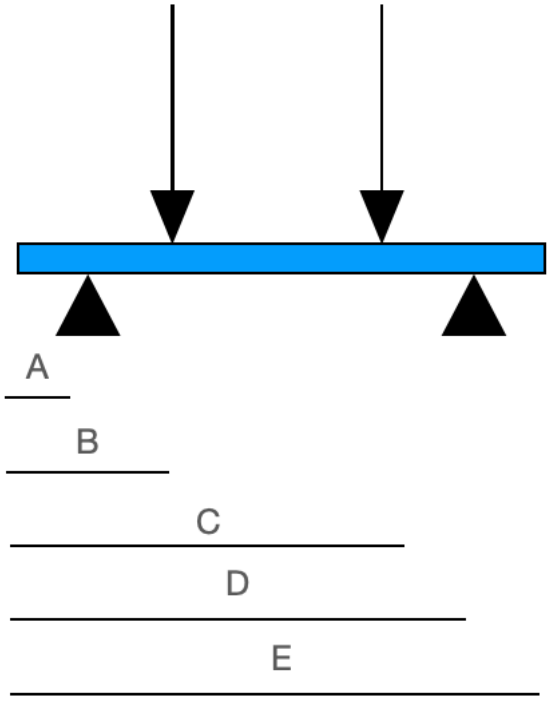


Figura 5.21. Planteamiento del cálculo de cortantes

El método de la Figura 5.22 debe introducir las reacciones calculadas en la viga una vez calculadas. Además introduce una carga 0 en el comienzo de la viga y al final de la misma que serán las posiciones desde la que calculará y hasta la que calculará.

Esta introducción de cargas 0 al inicio y final es necesaria porque de no haber cargas en estos puntos no se considerarían los tramos inicial y final:

```
//Este metodo inicializa los cortantes en los puntos representatios
public static void inicializa() {
    if(proyecto.getCamion().getTipo().getTipo()=='R') {
        cortantes=new ArrayList<Carga>();
        cortantes.add(new CargaPuntual("Inicio",0,0,false));
        cortantes.add(new CargaPuntual("Ra",proyecto.getCamion().getdistFrenteEjeDelantero(),-CalculoReaccionesRigido.getReaccionA(),false));
        cortantes.add(new CargaPuntual("Rb",proyecto.getCamion().getdistFrenteEjeDelantero()+proyecto.getCamion().getdistEjeDelanteroEjeTrasero(),
                -CalculoReaccionesRigido.getReaccionB(),false));
        cortantes.add(new CargaPuntual("Fin",proyecto.getCamion().getLargo(),0,false));
    }
```

Figura 5.22. Método de inicialización de cortantes

El método de la Figura 5.23 prepara las cargas para analizarlas en orden de posición de izquierda a derecha ya que el orden de introducción del usuario puede no ser ese para el usuario.

```
//Este método genera las posiciones clave en la viga del camion
public static void generaPosicionesClave() {
    if(proyecto.getCamion().getTipo().getTipo()=='R') {
        posiciones=new ArrayList<Double>();
        Iterator<Carga> it = cortantes.iterator();
        while(it.hasNext()) {
            Carga q=it.next();
            if(q.getTipo().getTipoCarga()=='P') {
                //Metemos un punto extra antes para poder ver como evoluciona en el cambio
                if(q.getPuntoInicio()!=0) posiciones.add(q.getPuntoInicio()-0.1);
                posiciones.add(q.getPuntoInicio());
                //Metemos un punto extra despues para poder ver como evoluciona en el cambio
                if(q.getPuntoInicio()!=proyecto.getCamion().getLargo()) posiciones.add(q.getPuntoInicio()+0.1);
            } else {
                CargaDistribuida cd=(CargaDistribuida)q;
                //Añadimos mas puntos para ver el cambio con la longitud
                double tramoCarga=cd.getLongitud()/20;
                for(int tramos=1;tramos<20;tramos++) {
                    posiciones.add(cd.getPuntoInicio()+(tramoCarga*tramos));
                }
                //Metemos el punto inicial y final
                posiciones.add(cd.getPuntoInicio());
                posiciones.add(cd.getPuntoInicio()+cd.getLongitud());
            }
        }
        //Ordenamos el ArrayList
        posiciones.sort(Comparator.naturalOrder());
    }
```

Figura 5.23. Método de organización de las cargas por posición

Así vemos que se prepara un ArrayList de posiciones donde iremos introduciendo cada posición de cada carga para al final ordenar dicha lista.

Vemos que en el método hemos introducido un punto extra antes y después. La finalidad de este punto anterior y posterior es para facilitar el dibujo de los diagramas.

Incluir en los diagramas los valores en sus zonas suele ser de ayuda. Sin embargo, es necesario saber en qué zonas el valor es importante y en cuales no. De lo contrario, todo el diagrama aparecería plagado de valores y sería imposible poder leer ninguno.

Con la introducción del punto anterior y posterior podemos decidir si el valor ha llegado a un máximo o a un mínimo y por tanto introducir o no el valor en el dibujo del diagrama.

Posteriormente, se realiza la preparación de los puntos de dibujo de los cortantes tal como se muestra en el método de la Figura 5.24. Se ha planteado esta solución como un objeto de tipo Punto que contendrá una posición (que representará su posición en el eje X) y un valor de carga (que representará su posición en el eje Y). Cada uno de estos puntos se incluye en una lista que se podrá utilizar posteriormente para la generación del diagrama.

```
//Este metodo genera los pares (posicion,cortante)
public static void generaPuntosCortantes() {
    if(proyecto.getCamion().getTipo().getTipo()=='R') {
        double carga=0;
        Iterator<Double> itpos=posiciones.iterator();
        puntosCambioCortantes=new ArrayList<PuntosCalculo>();
        while(itpos.hasNext()) {
            Double punto = itpos.next();
            carga=calculaCortante(punto,cortantes);
            puntosCambioCortantes.add(new PuntosCalculo(punto,carga));
        }
    }
```

Figura 5.24. Método de generación de los pares (posición, carga)

Se calcula el valor del cortante en cada punto importante de la viga. Como se puede apreciar, ese cálculo consiste sin más en realizar una suma del valor en caso de ser una carga localizada en un punto.

Si se trata de una carga distribuida, tal como se expone en la Figura 5.25, se realiza el producto de la longitud por la carga por unidad para obtener el valor total. Es necesario indicar que en este tipo de cargas se introducen dos posiciones en la lista de puntos, una para el inicio y otra para el final.

```java
//Este metodo devuele la carga cortante en una posicion del torque
public static double calculaCortante(double posicion,ArrayList<Carga> alc) {
    double carga=0;
    Iterator<Carga> it = alc.iterator();
    while(it.hasNext()) {
        Carga q=it.next();
        if(q.getTipo().getTipoCarga()=='P') {
            if(q.getPuntoInicio()<=posicion) {
                carga=carga+q.getValor();
            }
        } else {
            CargaDistribuida cd=(CargaDistribuida)q;
            double valorCarga=cd.getValor();
            if((posicion>=cd.getPuntoInicio())&&(posicion<=(cd.getLongitud()+cd.getPuntoInicio()))) {
                carga=carga+valorCarga*(posicion-cd.getPuntoInicio());
            } else if(posicion>(cd.getPuntoInicio()+cd.getLongitud())){
                carga=carga+valorCarga*cd.getLongitud();
            }
        }
    }
    carga=(double) carga;
    return carga;
}
```

Figura 5.25. Método de cálculo del valor por posición

Por último, el método de la Figura 5.26 se encarga de evaluar si el valor que se ha calculado para el esfuerzo cortante supone un valor máximo de carga y por tanto tenemos que actualizar el valor de tensión sobre la viga o no.

```java
//Devuelve el camion por trozos iguales y saca los valores para cada punto devolviendolos
public static ArrayList<PuntosCalculo> puntosGrafico(Proyecto p) {
    proyecto=p;
    ArrayList<PuntosCalculo> listaPuntos = new ArrayList<PuntosCalculo>();
    //Reseteamos el cortante que nos sirve para escalar el gráfico
    maximaCarga=0;
    maximaCargaCabeza=0;
    maximaCargaRemolque=0;
    prepara();
    double carga=0;
    int i=0;
    if(proyecto.getCamion().getTipo().getTipo()=='R') {
        while (i<proyecto.getCamion().getLargo()){
            carga=carga+calculaCortante(i,cortantes);
            if(Math.abs(carga)>Math.abs(maximaCarga)) {
                maximaCarga=carga;
                proyecto.getCamion().setTensionCortanteViga(maximaCarga);
                proyecto.getCamion().setTensionVonMisses();
            }
            PuntosCalculo puntoC=new PuntosCalculo(i,carga);
            i=i+puntosGrafico;
            carga=0;
            listaPuntos.add(puntoC);
        }
        return listaPuntos;
```

Figura 5.26. Método de cálculo de la máxima carga cortante

En caso de comprobar que el valor es el máximo de la carga se actualiza la tensión cortante con esa carga máxima y posteriormente se actualiza la tensión Von Misses del camión.

NOTA: Es importante remarcar que la información que se ha expuesto en los métodos corresponde al cálculo de la estructura del camión rigido. En todos ellos se ha omitido la parte del articulado porque únicamente implica la división del camión en dos partes: cabeza tractora y remolque y aplicar esta misma metodología por cada parte.

Momentos flectores

El cálculo de los momentos flectores no se va a exponer aquí, ya que resulta exactamente igual al de los esfuerzos cortantes con la única diferencia de la fórmula necesaria para calcular el valor que sí se muestra a continuación.

Podemos ver en la Figura 5.27 que en este caso el cálculo impone el producto de la carga por la distancia a la que se encuentra.

```
//Este metodo devuele la carga cortante en una posicion del torque
public static double calculaMomento(double posicion,ArrayList<Carga> alc) {
    double carga=0;
    Iterator<Carga> it = alc.iterator();
    while(it.hasNext()) {
        Carga q=it.next();
        if(q.getTipo().getTipoCarga()=='P') {
            if(q.getPuntoInicio()<=posicion) {
                carga=carga+q.getValor()*(posicion-q.getPuntoInicio());
            }
        } else {
            CargaDistribuida cd=(CargaDistribuida)q;
            double valorCarga=cd.getValor();
            if((posicion>=cd.getPuntoInicio())&&(posicion<=(cd.getLongitud()+cd.getPuntoInicio()))) {
                carga=carga+valorCarga*(posicion-cd.getPuntoInicio())*(posicion-cd.getPuntoInicio())/2;
            } else if(posicion>(cd.getPuntoInicio()+cd.getLongitud())){
                carga=carga+valorCarga*cd.getLongitud()*(posicion-(cd.getPuntoInicio()+cd.getLongitud()/2));
            }
        }
    }
    carga=(double) carga;
    return carga;
}
```

Figura 5.27. Método de cálculo de los momentos flectores

Dibujo de los diagramas

A la hora de realizar el diagrama se consideran los puntos y se realiza un escalado del dibujo a las dimensiones del panel, utilizando para ello las dimensiones máximas del camión.

En el caso de los diagramas de cortantes y momentos, realizamos un escalado en función del valor máximo calculado y asignamos a los ejes de dichos diagramas unas divisiones y valores correspondientes con dicho escalado.

5.4.2. Cálculo de las secciones compuestas

Para el cálculo de las secciones resistentes se han aplicado las propiedades del cálculo de secciones compuestas. Para ello, se realiza en cada caso el cálculo del área, inercia y modulo resistente de una sección rectangular y mediante la combinación de varias de ellas se ha realizado el cálculo de las propiedades de secciones más complejas.

En la Figura 5.28 puede apreciarse como una sección tipo C puede descomponerse en tres secciones rectangulares para ser capaces de realizar el cálculo de una forma simplificada.

Como comentario, se indica que este tipo de secciones suelen tener redondeos en las zonas de unión que no se consideran cuando se realiza un cálculo aproximado, como es el caso. El resultado obtenido de esta forma no se desvía de la realidad y el tratamiento de las bases de datos permite la modificación del valor en caso necesario, una vez calculadas las propiedades, si podemos obtener los valores de un prontuario.

Figura 5.28. Ejemplo de cálculo de una sección tipo C

Así pues se ha realizado una clase específica de sección rectangular que se expone en la Figura 5.29.

```
//Esta clase calcula propiedades físicas de una seccion rectangular. Se usa como una clase auxiliar para los calculos de secciones
public class SeccionRectangular implements Seccion, Serializable {
        double base;
        double altura;
        double area;
        double inercia;
        //Generamos una variable para almacenar la información en el area de Texto"
        String infoSecRec="";

        public SeccionRectangular(double ancho, double alto) {
                this.base = ancho;
                this.altura = alto;
                //Completamos la informacion de la seccion rectangular
                //Formateamos los numeros para que tengan 2 decimales
                DecimalFormat df = new DecimalFormat("0.00");
                //Pasamos a la variable infoSecRec la información a presentar en pantalla
                infoSecRec=infoSecRec+"Base: "+df.format(Unidades.getLongitudEnUnidadesActuales(this.base))
                +"\nAltura: "+df.format(Unidades.getLongitudEnUnidadesActuales(this.altura))
                +"\nArea: "+df.format(Unidades.getAreaEnUnidadesActuales(calculaArea()))
                +"\nInercia: "+df.format(Unidades.getInerciaEnUnidadesActuales(calculaInercia()))
                +"\n";
        }
        @Override
        public double calculaArea() {
                this.area=base*altura;
                return this.area;
        }

        @Override
        public double calculaInercia() {
                this.inercia=base*Math.pow(altura,3)/12;
                return this.inercia;
        }
```

Figura 5.29. Clase de sección rectangular

Sobre ella podemos apreciar la simplicidad del cálculo de las propiedades y del manejo de la misma y una vez tenemos esta clase, podemos producir infinidad de secciones mediante la combinación de varias de ellas. Se expone el cálculo de una sección compuesta en la Figura 5.30.

```
public CalculoSeccion(double baseinf, double altoinf, double basemed,double altomed,double basesup,double altosup,char tipo){
    //Inicializamos los 3 rectángulos de la seccion
    //Actualizamos las variables de clase y uso las unidades por defecto para el calculo que son milimetros.
    this.baseinf=Unidades.getConversionLongitud()*baseinf;
    this.altoinf=Unidades.getConversionLongitud()*altoinf;
    //Como el rectangulo intermedio de la seccion esta vertical en vez de en horizontal cambio el orden de los valores.
    this.basemed=Unidades.getConversionLongitud()*altomed;
    this.altomed=Unidades.getConversionLongitud()*basemed;
    this.basesup=Unidades.getConversionLongitud()*basesup;
    this.altosup=Unidades.getConversionLongitud()*altosup;
    //Completamos la informacion de referencia de ayuda y calculamos propiedades de cada rectangulo de la sección
    //inferior:
    completaInfo("rectangulo inferior:");
    this.secin = new SeccionRectangular(this.baseinf,this.altoinf);
    completaInfo(this.secin.getInfo());
    //medio
    completaInfo("rectangulo medio:");
    this.secmed = new SeccionRectangular(this.basemed,this.altomed);
    completaInfo(this.secmed.getInfo());
    //superior
    completaInfo("rectangulo superior:");
    this.secsup = new SeccionRectangular(this.basesup,this.altosup);
    completaInfo(this.secsup.getInfo());
    //Calculamos las propiedades de la sección compuesta
    calculaArea(); //Calcula el Area
    calculaAlturasEjesNeutros(); //Calcula la cota y del eje neutro de cada rectangulo de la seccion para calcular el eje neutro
    calculaPosicionEjeNeutro(); //Calcula la posicion del eje neutro de la seccion C completa
    calculaInercia(); //Calculamos la inercia de la sección compuesta
    //Completamos la información para el area de ayuda
    DecimalFormat df = new DecimalFormat("0.00");
    completaInfo("**************************************************");
    completaInfo("Area general seccion compuesta: "+df.format(Unidades.getAreaEnUnidadesActuales(this.getArea()))+Unidades.getUnidadesArea());
    completaInfo("Inercia seccion compuesta: "+df.format(Unidades.getInerciaEnUnidadesActuales(this.getInercia()))+Unidades.getUnidadesInercia());
    completaInfo("Modulo Resistente seccion compuesta: "+df.format(Unidades.getModuloResistenteEnUnidadesActuales(this.getModulo()))+Unidades.getUnida
}
```

Figura 5.30. Ejemplo de cálculo de una sección compuesta

En este caso, es importante en primer lugar calcular el eje neutro para la viga compuesta, donde puede considerarse el centro de masas de la sección, ya que las propiedades deben medirse con respecto al eje neutro. En la Figura 5.31 se muestra el método que realiza el cálculo de la posición del eje neutro.

En primer lugar calcularemos el área total de todas las subsecciones incluidas en la sección compuesta.

Calculamos entonces las posiciones de los ejes neutros de las subsecciones que estará en su punto medio.

El nuevo eje neutro estará localizado en el punto que resulta de sumar los productos de las áreas de cada subsección por la distancia de su eje neutro a la cota cero.

Posteriormente dividiremos dicho producto entre las sumas de las áreas de las subsecciones. Con esto hemos conseguido aportar una solución de cálculo al usuario que permita trabajar con muchos tipos de sección.

```
//Se calcula la altura desde la base inferior del eje neutro
private void calculaPosicionEjeNeutro() {
            //Calcula el sumatorio de cada Area de cada rectángulo por la altura a sus respectivos ejes neutros.
            double sumaperfiles = this.alturaNeutroInf*this.secin.getArea()+this.alturaNeutroMed*this.secmed.getArea()+
                    this.alturaNeutroSup*this.secsup.getArea();
            //Obtenemos el eje neutro del perfil C completo
            this.y_ejeneutro=sumaperfiles/this.getArea();
}

//Calculamos el area de la seccion compuesta
private void calculaArea() {
            this.area = secin.getArea()+secmed.getArea()+secsup.getArea();
}

//Calcula la altura del eje Neutro de cada rectangulo
private void calculaAlturasEjesNeutros() {
            //Calculamos la altura del eje neutro del rectangulo en la base (inferior)
            this.alturaNeutroInf = this.secin.getAlto()/2;
            //Calculamos la altura del eje neutro del rectangulo intermedio (medio)
            this.alturaNeutroMed = this.secin.getAlto()+this.secmed.getAlto()/2;
            //Calculamos la altura del eje neutro del rectangulo más elevado (superior)
            this.alturaNeutroSup = this.secin.getAlto()+this.secmed.getAlto()+this.secsup.getAlto()/2;
}

//Calcula la Inercia sobre el eje neutro de la sección completa
private void calculaInercia() {
            double inerciaInf = this.secin.getInercia()+this.secin.getArea()*Math.pow(this.alturaNeutroInf, 2);
            double inerciaMed = this.secmed.getInercia()+this.secmed.getArea()*Math.pow(this.alturaNeutroMed, 2);
            double inerciaSup = this.secsup.getInercia()+this.secsup.getArea()*Math.pow(this.alturaNeutroSup, 2);
            double inerciaCEjeBase = inerciaInf+inerciaMed+inerciaSup;
            this.inercia = inerciaCEjeBase-this.area*Math.pow(this.y_ejeneutro, 2);
            this.moduloResistente=this.getInercia()/this.getAlturaMedia();
}
```

Figura 5.31. Gestión de los nuevos ejes neutros de la viga

NOTA: Para esta versión de la aplicación sólo se han considerado las vigas en C o en I, ya que son las más utilizadas en este tipo de trabajos. Por otra parte, hay que remarcar que para el cálculo de las tensiones solo se necesitará la inercia para la flexión de la viga sobre su eje X-X en las tipo C o I (al ser un cálculo bidimensional). Por fortuna, el valor sería el mismo para ambas en este sentido.
Si tuviésemos en cuenta las cargas tridimensionalmente, las vigas C e I sí tienen una diferencia en el eje de flexión Y-Y. Los ejes aquí comentados se muestran en la Figura 5.32.

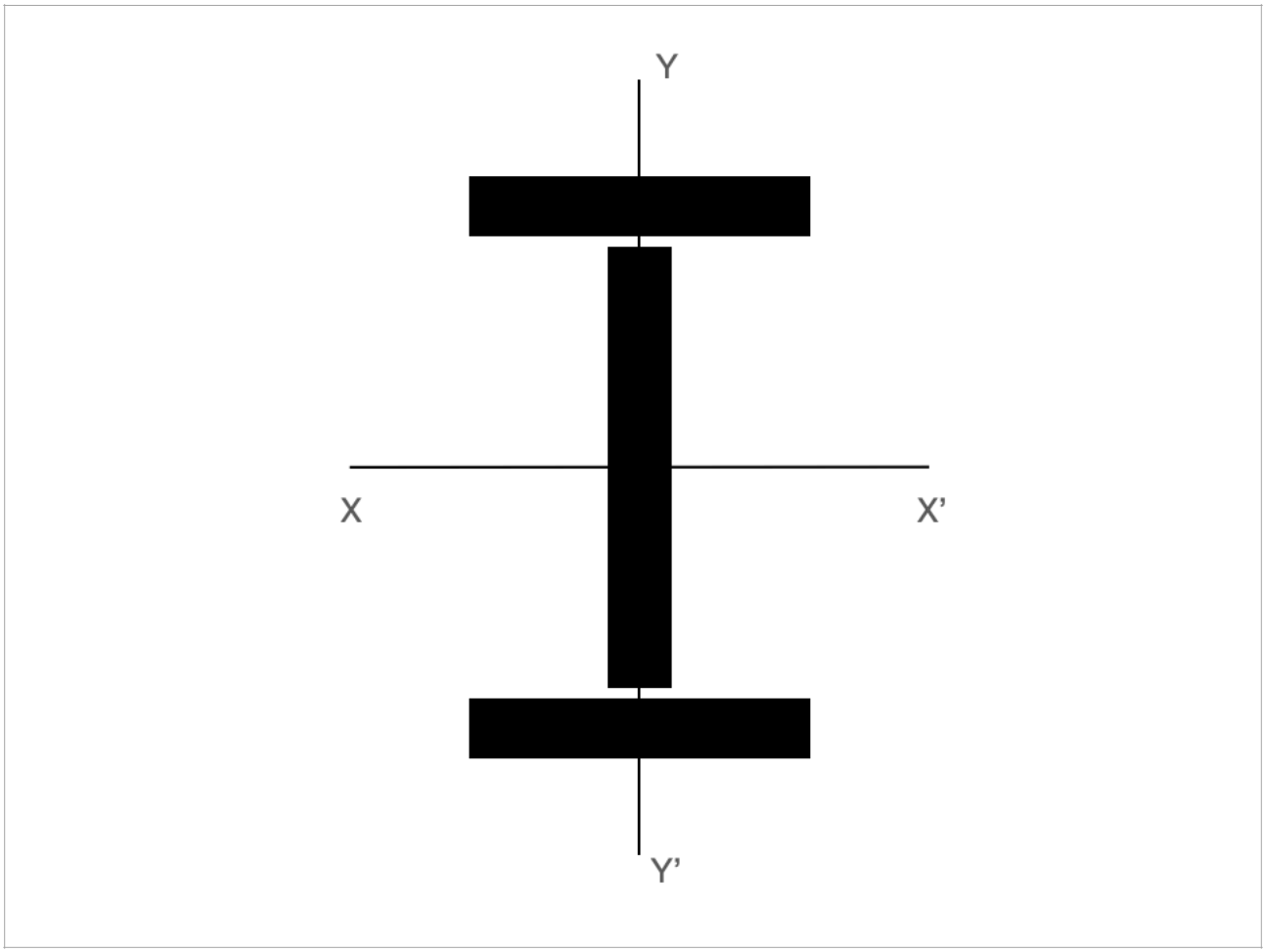


Figura 5.32. Ejes para flexión en una viga tipo I

5.5. Pruebas

Durante el desarrollo de la aplicación y de acuerdo a la estrategia evolutiva de los sucesivos prototipos obtenidos se ha planteado una estrategia de pruebas orientadas a las necesidades de recursos y tiempos de los que se ha dispuestos.

“Una buena prueba es la que tiene una alta probabilidad de encontrar un error” (Pressman y Maxim 2020, p. 396) .

“Las restricciones de tiempo y recursos pueden dictar la ejecución de solo aquellas pruebas que tengan la mayor probabilidad de descubrir toda una clase de errores” (Pressman y Maxim 2020, p. 396).

Como resultado se ha focalizado el esfuerzo de las pruebas principalmente en la fases de introducción de datos para evitar errores que puedan afectar a los resultados y para acelerar la corrección de los mismos en dicha fase. De los resultados, se ha ido generando con los sucesivos prototipos una clase denominada CheckDatos donde se han introducido métodos específicos para la introducción de los mencionados datos de una forma guiada y evitando posibles errores.

Durante cada fase de prototipos, se ha ido testando el funcionamiento de los distintos métodos para comprobar el correcto funcionamiento de los mismos.

Paralelamente y con cada prototipo, se han realizado pruebas de integración de los distintos avances que se están realizando sobre la herramienta.

5.5.1. Pruebas de introducción de datos

Se han realizado pruebas de introducción de datos en cada área de la aplicación donde se debe introducir información: Datos principales, sección o cálculo. Las pruebas han cubierto la introducción de datos de distintos tipos así como la ausencia de alguno de ellos.

Ausencia de algún dato: La aplicación indica la ausencia de algún dato tal como muestra la Figura 5.33.

Figura 5.33. Error por ausencia de datos

Introducción de valores negativos o no numéricos: Gestiona la información de los valores negativos tal como muestra la Figura 5.34.

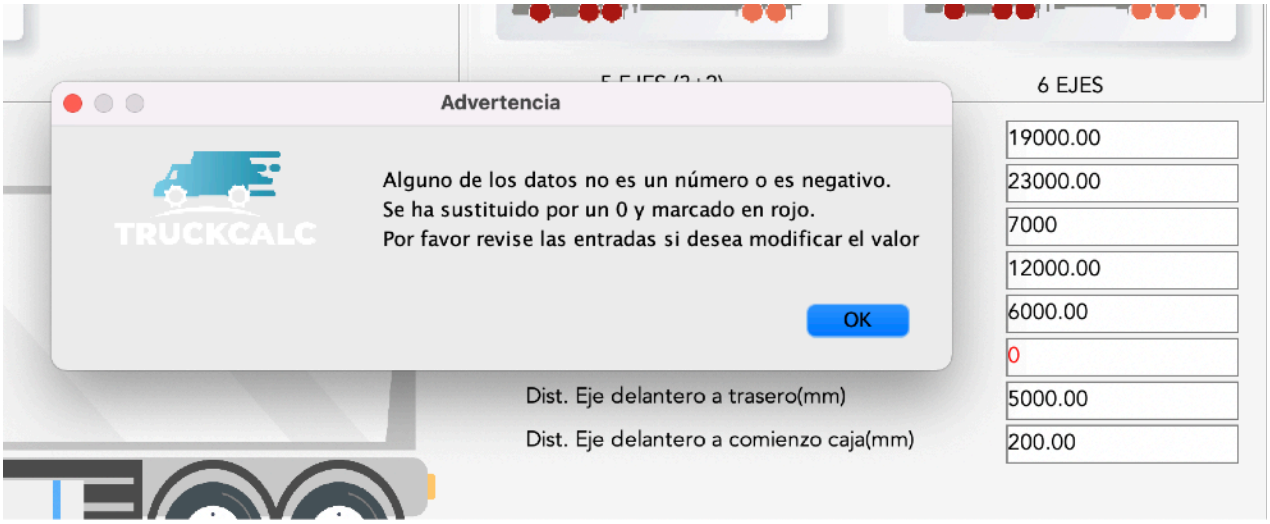


Figura 5.34. Gestión de errores en datos negativos o no numéricos

Introducción de valores no correctos para la solución: La aplicación indica las recomendaciones posibles tal como se muestra en la Figura 5.35.

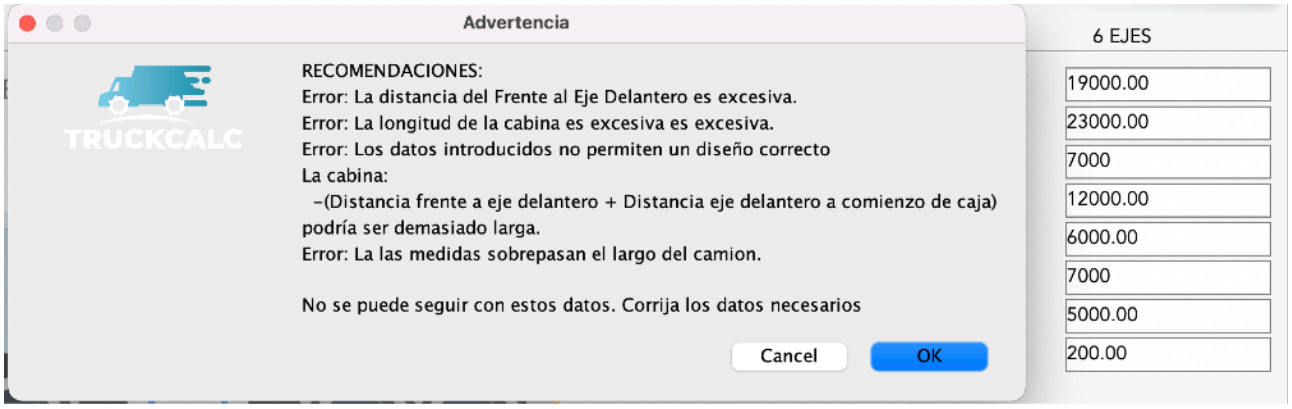


Figura 5.35. Gestión de errores en datos negativos o no numéricos

Hay que remarcar que estas recomendaciones varían con los valores introducidos y el mensaje puede incluir más o menos información en función de la corrección o incorrección de la información introducida.

5.5.2. Pruebas de validación

PRUEBAS CON SISTEMAS EXTERNOS

Los resultados de la prueba de validación deben realizarse mediante el uso de software específicamente diseñado para el cálculo de estructuras, si bien, dado que su uso no está orientado a este tipo de aplicación, deben adecuarse los modelos de cálculo para poder obtener un sistema completamente equivalente a la hora de la comparación.

Para la validación de las pruebas debemos comentar lo siguiente.

El cálculo de la herramienta se realiza aplicando el principio de superposición, como se indicaba en puntos anteriores. Esto implica que la validación puede realizarse de manera sencilla y sin realizar una batería de pruebas compleja. La razón es que simplemente se realiza una suma de estados para cada una de las cargas.

En nuestro caso hemos realizado un modelo comparativo utilizando el software SkyCiv que se encuentra de manera gratuita para utilización en web.

Se plantea el caso de prueba mostrado en la Figura 5.36:

- Camión rígido de 6000 mm de longitud.

- Apoyo delantero a 400 mm del frente del camión.
- Apoyo trasero a 5000 mm del apoyo delantero (5400 mm del frente).
- Carga sobre el camión a 3000 mm del frente con un valor de 1000 kgf.

Se introducen los datos en el software de la página web para el cálculo de vigas.

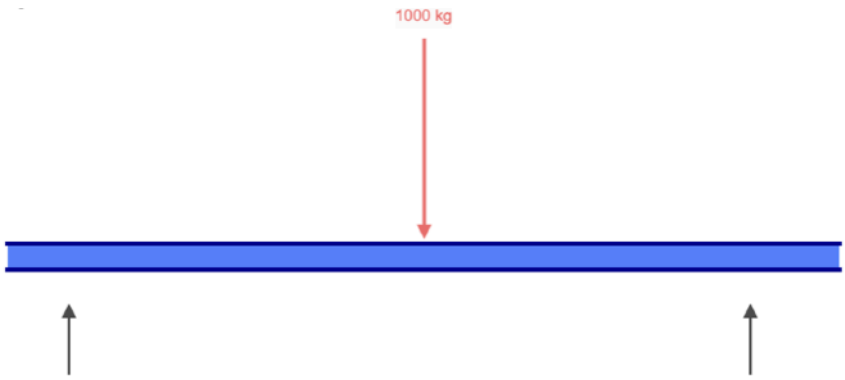


Figura 5.36. Caso de prueba en SkyCiv (SkyCiv).

Una vez solicitamos la resolución de este caso obtenemos los siguientes valores:

Reacciones.

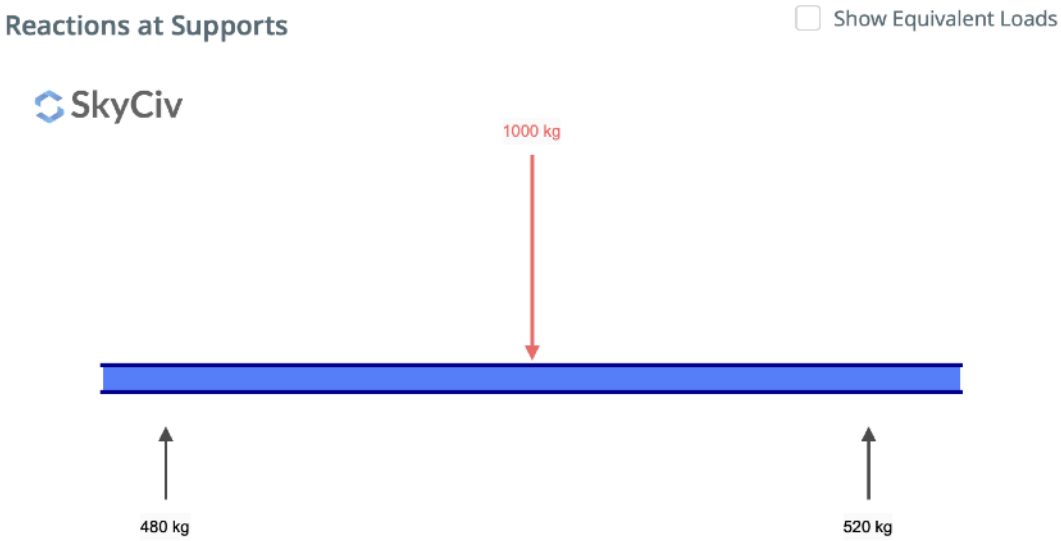


Figura 5.37. Reacciones calculadas (SkyCiv).

Diagramas de esfuerzos cortantes y momentos flectores.

Figura 5.38. Diagramas de momentos y cortantes (SkyCiv)

Resultados de fuerzas extremas.

Results

Reactions

Support at	X	Y	Mx
400	0 kg	480 kg	0 kN-m
5400	0 kg	520 kg	0 kN-m

Force Extremes

Result	Max	Min
Bending Moment	12.239 kN-m	0 kN-m
Shear	480 kg	-520 kg

Figura 5.39. Valores de resultados (SkyCiv)

Utilizamos los resultados de las reacciones de la Figura 5.37, los diagramas de cortantes y flectores de la Figura 5.38 y los resultados de la Figura 5.39 para comprobar la validez de los resultados que arroja nuestra herramienta empezando por introducir los valores como muestra la Figura 5.40.

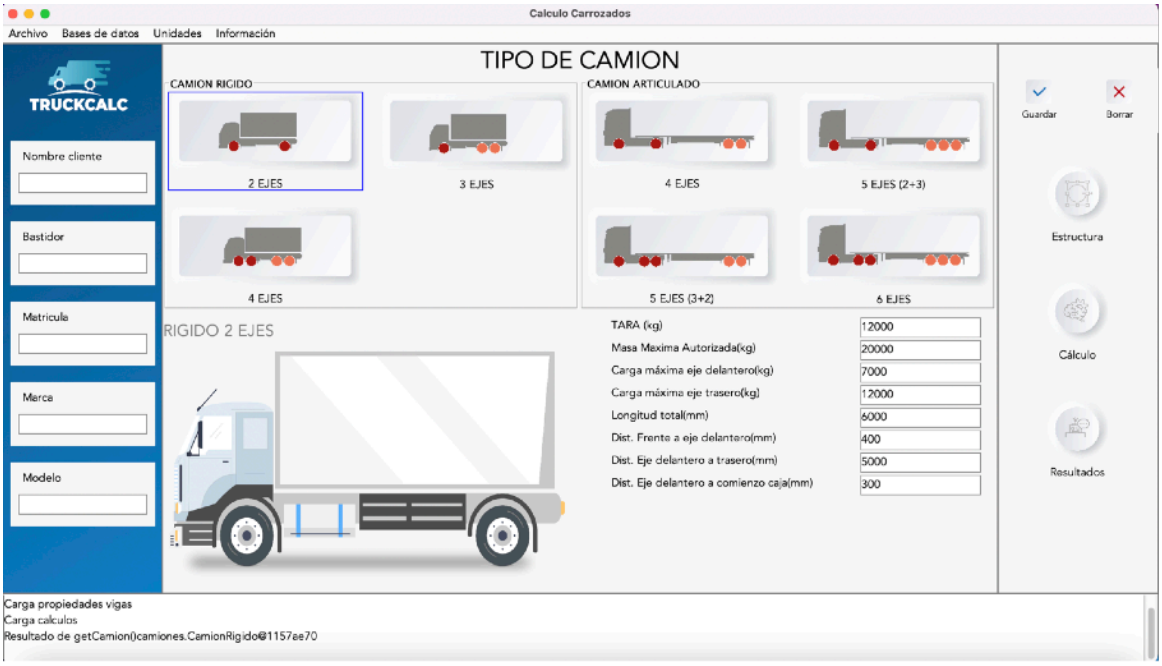


Figura 5.40. Introducción de valores en la herramienta

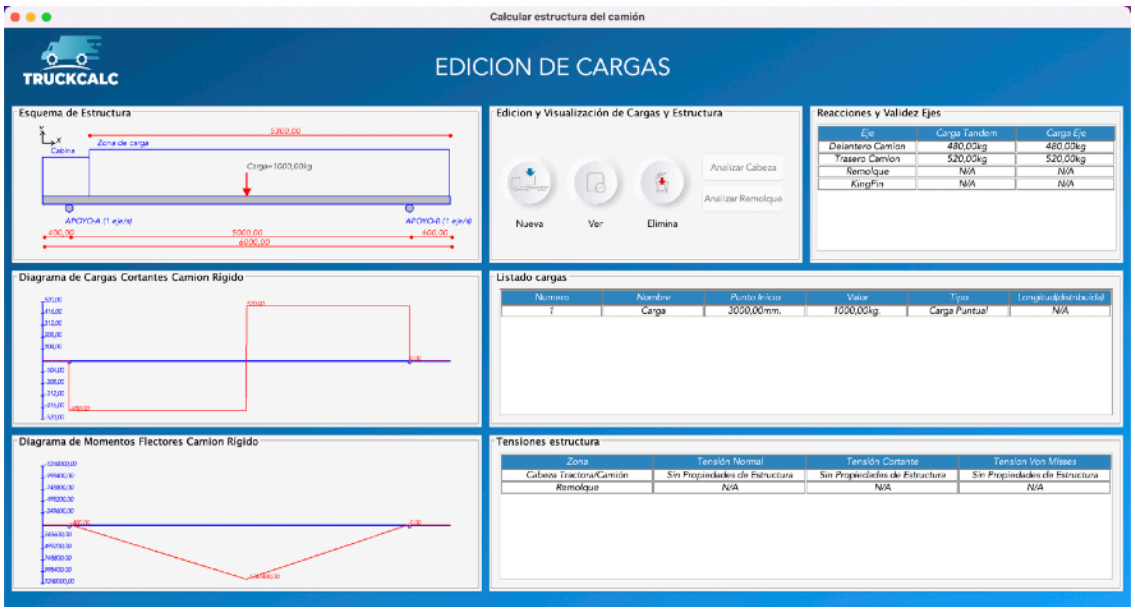


Figura 5.41. Introducción de carga

Introducimos la carga sobre el camión y comprobamos resultados como se muestra en la Figura 5.41. Comprobamos también los resultados de las reacciones que obtenemos y que se han mostrado en la figura 5.42 comprobándose que existe coincidencia con los obtenidos con SkyCiv.

Reacciones y Validez Ejes

Eje	Carga Tandem	Carga Eje
Delantero Camion	480,00kg	480,00kg
Trasero Camion	520,00kg	520,00kg
Remolque	N/A	N/A
KingPin	N/A	N/A

Figura 5.42. Reacciones sobre el caso de prueba

A continuación comprobamos el diagrama de cortantes obtenido y mostrado en la Figura 5.43.

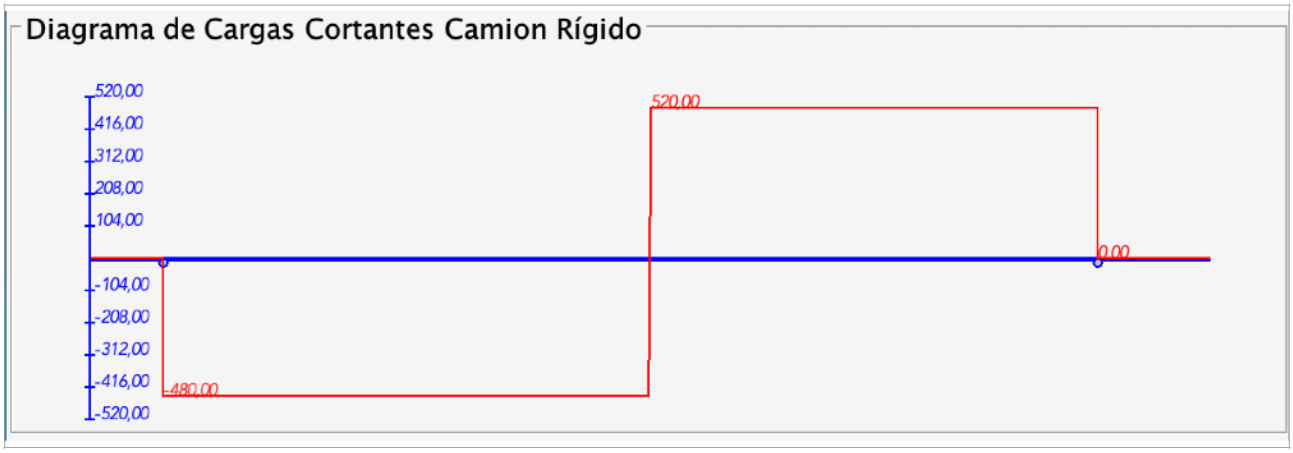


Figura 5.43. Cortantes sobre el caso de prueba

Vemos que los cortantes coinciden con los de la aplicación, comenzando en cero hasta llegar a los apoyos, incrementando su valor hasta el cortante en cada sección y terminando en cero al final de la viga desde el apoyo trasero.

NOTA: La dirección de los diagramas tienen que ver con la consideración realizada por la aplicación y por la herramienta en cuanto al sentido positivo.

Comprobamos los momentos flectores, que se muestran en la Figura 5.44.

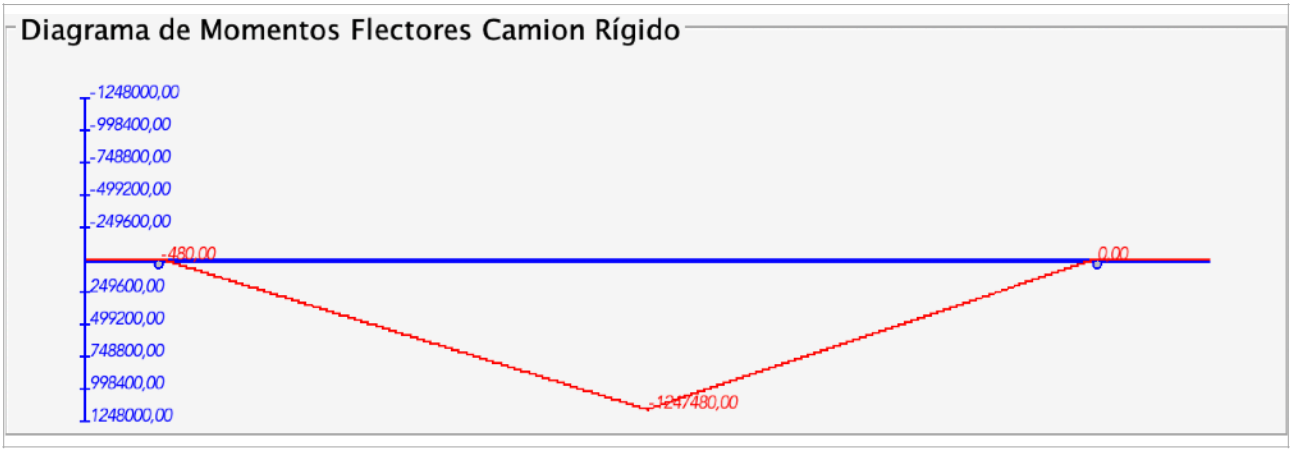


Figura 5.44. Momentos flectores sobre el caso de prueba

Vemos que los momentos flectores coinciden con los de la aplicación comenzando en cero hasta llegar a los apoyos, incrementando su valor hasta el máximo en el punto de la fuerza y terminando en cero al final de la viga desde el apoyo trasero.

NOTA: La dirección de los diagramas tienen que ver con la consideración realizada por la aplicación y por la herramienta en cuanto al sentido positivo.

Podemos ver también que el valor máximo calculado por ambas es el mismo.

De esta forma, podemos considerar que la aplicación está realizando un cálculo correcto sobre las consideraciones y simplificaciones consideradas y que, por lo mencionado al comienzo de este apartado en relación al principio de superposición, podemos validar su funcionamiento.

PRUEBAS CON CALCULOS MANUALES

Se comprueban los resultados obtenidos en los ejemplos de cálculo de camión rígido y articulado del Capítulo 2.

Caso Rígido

Los valores manuales obtenidos fueron:

Reacción en delantero: 825 kgf.

Reacción en eje trasero: 175 kgf.

Figura 5.45. Comprobación resultados manuales de camión rígido del Capítulo 2

Se comprueba que los valores obtenidos en el cálculo con la herramienta y mostrados en la Figura 5.45 coinciden con los obtenidos manualmente, por lo que se considera correcto el comportamiento de la herramienta.

Caso Articulado

Los valores manuales obtenidos fueron:

<u>En el remolque:</u>

Reacción en eje remolque: 375 kgf.

Reacción en king pin: 625 kgf.

<u>En la cabeza tractora</u>

Reacción en eje delantero: 406,25 kgf.

Reacción en eje trasero: 218.75 kgf.

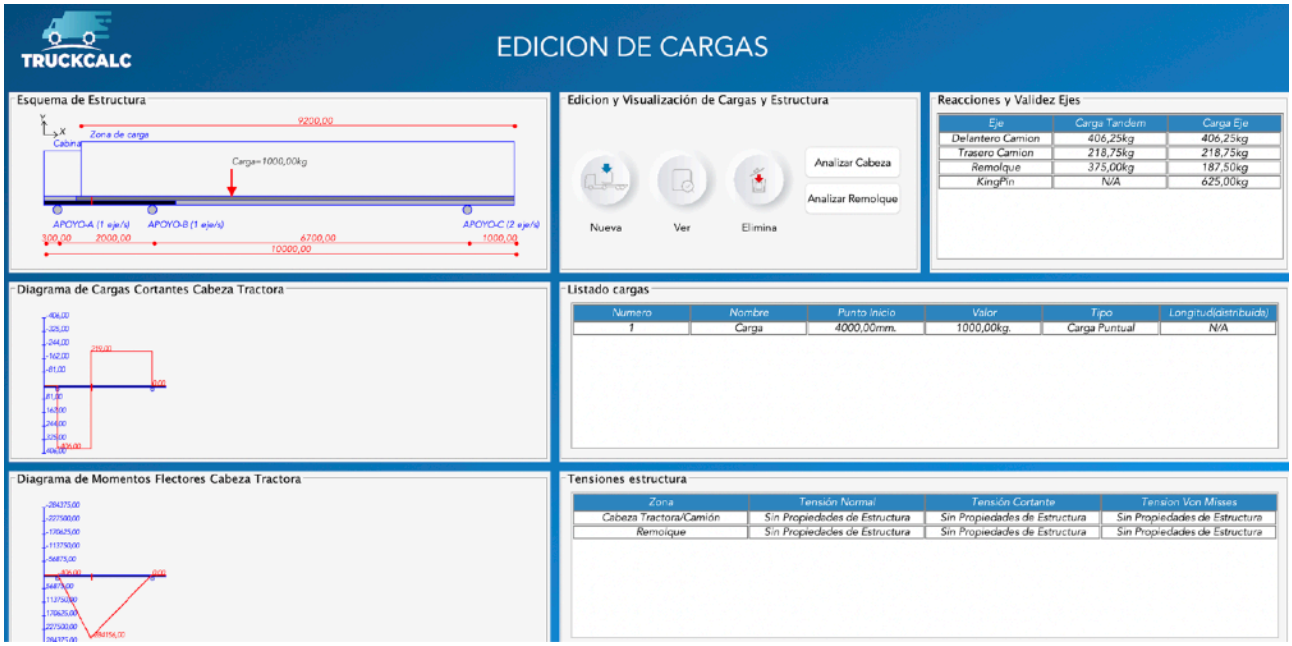


Figura 5.46. Comprobación resultados manuales de camión articulado del Capítulo 2

Se comprueba los valores obtenidos con la herramienta, mostrados en la Figura 5.46, coinciden con lo lo obtenido en el cálculo manual validándose, por tanto, el comportamiento de la herramienta.

5.6. Conclusiones

A lo largo del presente capítulo se han expuesto las diferentes herramientas software y hardware adoptadas así como el entorno de desarrollo que se ha utilizado.

Se ha hecho especial hincapié en la descripción del uso de Apache Derby y de Itext al no resultar ambas herramientas estándar incluidas con Java.

Se ha presentado la forma en que se han implementado varios de los requerimientos presentados con los casos de uso y se han descrito las peculiaridades de cada uno. Se ha prestado especial atención a la descripción de la introducción de los datos, generación y utilización de secciones, cálculo de cargas, generación del archivo de resultados y utilización de las bases de datos.

Se han presentado algunas soluciones críticas que han supuesto bien especial dificultad o especial eficiencia en la implementación de la herramienta.

Para terminar se han descrito las pruebas realizadas con especial atención a las de introducción de datos y las de validación.

Capítulo 6. Planificación y costes del proyecto

6.1. Introducción

La realización del presente proyecto ha supuesto el seguimiento de un plan de desarrollo adaptado a las particularidades de cada una de las fases. Se exponen aquí las diferentes fases junto con los resultados obtenidos en cada una de ellas.

Asimismo, los tiempos y recursos expuestos en la realización del proyecto representan un coste asociado al mismo. Este coste necesita ser considerado especialmente durante la fase inicial para la evaluación de su precio final y para planificar los desembolsos que se deberán realizar.

También es necesario el control de dicho coste previsto durante la fase de desarrollo. Deberán tenerse en cuenta las desviaciones, considerar medidas alternativas y adoptar medidas correctoras para evitar un incremento que conlleve la falta de rentabilidad del proyecto.

6.2. Fases y prototipos sucesivos

La realización del presente proyecto ha estado basada en una estrategia de obtención de prototipos sucesivos que han ido completando la funcionalidad del proyecto.

Antes del comienzo del trabajo con el primer prototipo, se ha realizado un análisis de los requerimientos de cada una de las fases y de como los resultados y objetos obtenidos en cada una de las fases influirá en las entradas de los siguientes prototipos para mantener el control de los tiempos y costes.
Se expone inicialmente el plan previsto y se indican las desviaciones que se han producido durante la realización del mismo y sus causas.
Posteriormente describen las fases para cada prototipo y la funcionalidad que se pretende con cada una de ellas.

Se han descrito cuatro fases de desarrollo:

- **FASE 1. INTERFAZ DE ENTRADA**. Esta fase tiene que ver con el diseño de las diferentes GUI que el usuario utilizará en el uso de la herramienta. Se estudian diferentes alternativas, diseños o metodologías de introducción de los datos.
- **FASE 2. ESTRUCTURA DE CÁLCULO.** Esta fase recoge el estudio de la metodología de cálculo de estructuras utilizada en este tipo de trabajos. Recoge también la búsqueda de la metodología más eficiente que, cumpliendo con los requisitos de seguridad y de presentación de resultados comúnmente aceptados, sea capaz de mejorar la eficiencia en la generación del código. Finalmente recoge la estructuración del código en diferentes métodos para la realización del cálculo y su comprobación.
- **FASE 3. ALMACENAMIENTO DE LA INFORMACIÓN.** Esta fase recoge el trabajo realizado para la gestión, presentación en GUI y almacenamiento de los datos tanto de las bases de datos como de los ficheros de proyecto.
- **FASE 4. PRESENTACIÓN DE RESULTADOS.** Esta fase recoge el trabajo realizado para la generación del informe anexo técnico de cálculos que representa el entregable de todo proyecto realizado.

La Figura 6.1 muestra el diagrama de GANTT presentado como planteamiento inicial del desarrollo del proyecto.

Figura 6.1. Diagrama inicial de Gantt

Se han producido algunas variaciones en el desarrollo de la aplicación y su plan que exponemos a continuación.

La Figura 6.2 expone el plan finalmente seguido en la que se aprecian algunas barras en color rojo y algunas barras en color naranja. Las barras rojas implican un retraso no controlable y las naranjas un incremento del plazo estimado para completar la tarea correspondiente.

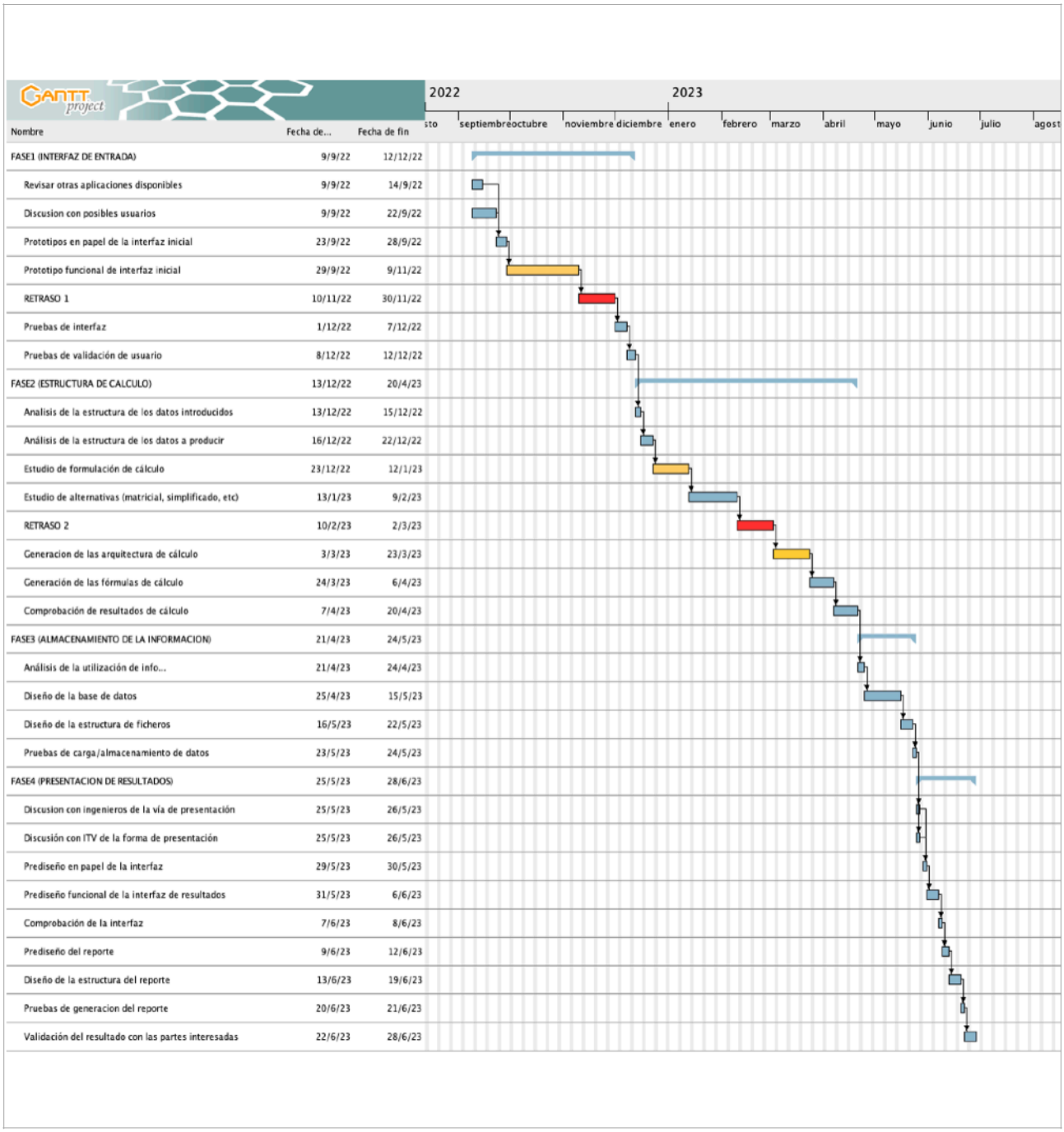


Figura 6.2. Diagrama final de Gantt

Se exponen dos barras rojas correspondientes a retrasos sobre el plan previsto:

- RETRASO 1.
- RETRASO 2.

Se exponen tres barras naranjas:

- Prototipo funcional de interfaz inicial.
- Estudio de formulación de cálculo.

- Generación de arquitectura de cálculo.

Donde los plazos han sido mayores de lo esperado. La razón en todos los casos ha sido la búsqueda de una codificación más eficiente y que requiriese menos esfuerzo en caso de extensiones así como la comprobación de los resultados del cálculo de cada método.

6.3. Costes del proyecto

Se recogen aquí, sobre la Tabla 6.1, los costes aproximados del proyecto. No se trata de un coste exacto porque resulta difícil estimar algunas partidas como pueden ser la electricidad consumida, el pago la conexión a internet, uso de material de oficina como tinta para impresión de documentación, etc. Sin embargo, se ha hecho una estimación aproximada de los mismos y se recoge en la tabla mostrada a continuación.

No se consideran las licencias utilizadas ya que el software empleado ha recogido licencias libres o para estudiantes.

Concepto	Unidades	Coste/unidad (€)	Coste total (€)
Horas	661	60	39660
Electricidad	N/A	24	24
Conexión a internet	N/A	30	30
Material de oficina (varios)	N/A	47	47
Amortizaciones de equipos	N/A	182	182
Licencias	N/A	0	0
		TOTAL	39943

Tabla 6.1. Costes del proyecto

Capítulo 7. Conclusiones y trabajos futuros

7.1. Introducción

Se exponen en este capítulo unas conclusiones finales sobre el trabajo realizado y algunas mejoras o extensiones posibles al proyecto.

Se pretende por tanto indicar el grado de cumplimiento con las expectativas buscadas al inicio, así como definir nuevas vías de desarrollo del proyecto que pudiesen completar la funcionalidad, bien por no estar incluidas en el alcance inicial del proyecto o bien por haberse encontrado posibilidades de mejora durante el trabajo realizado.

7.2. Conclusiones

Se han realizado durante el presente proyecto las siguientes tareas:

- **Definición de la metodología de trabajo:** Se han revisado las distintas formas de actuación para la consecución del proyecto, llegándose a una estrategia incremental organizada en etapas. En una primera etapa se ha planteado el diseño de la interfaz. Posteriormente, se ha trabajado en el refine de la metodología de cálculo, se han introducido las capacidades de gestión de bases de datos, almacenamiento en fichero de resultados y generación de los archivos pdf.

- **Definición de la metodología de cálculo y estado del arte:** Se han analizado los conceptos básicos de cálculo de estructuras. Posteriormente, se han analizado las simplificaciones posibles y se han revisado las posibilidades de implementación de dichas metodologías en la codificación. Se ha consultado a la ITV y a partes interesadas sobre el proceso más común de gestión de estos proyectos y se han investigado los resultados que se deben obtener de estos cálculos para la presentación del informe en los organismos técnicos.
- **Análisis del problema:** Se han revisado los Casos de Uso y modelado las clases correspondientes, definido el modelo de dominio y se han obtenido las definiciones de los requisitos funcionales y no funcionales.
- **Diseño de la herramienta:** Se ha considerado el diseño y planteado los diagramas de secuencia, la arquitectura lógica y los diagramas de clase.
- **Implementación y pruebas:** Se han realizado la implementación de los requerimientos y funcionalidades. Se ha optimizado el código lo posible para cumplir con bajo acoplamiento y alta cohesión así como futuras ampliaciones de funcionalidades y se han testado los resultados para comprobar con el cumplimiento de los requerimientos. También se han generado el manual de instalación y de uso de la herramienta.

Otras conclusiones sobre el trabajo realizado se exponen a continuación.

Cumplimiento con requerimientos técnicos iniciales

La aplicación debía responder a los siguientes puntos:

- Interfaz clara y agradable.
- Estructura adecuada al cálculo que procese los datos introducidos.
- Permitir el cálculo del king pin.
- Generar una interfaz de presentación de resultados.
- Generar bases de datos para las cabezas tractoras y tipo de carrozado así como para secciones resistentes.
- Generar un informe de resultados.
- Almacenamiento en ficheros del cálculo y las cabezas tractoras.

Se considera que se ha cumplido con los requerimientos iniciales que han sido recogidos realizando reuniones y comunicaciones con personal de ITV's así como con empresarios del carrozado y posteriormente habiendo trabajado en las siguientes funcionalidades:

- Diseño de interfaz.

- Introducción de los datos y gestión de los mismos para ayudar al usuario. Se han realizado clasificaciones por tipo de camión que realiza el cálculo en función de su tipología.
- Aporte de bases de datos específicas. Se han incluido además mejoras en la gestión de la modificación de los valores así como en la eliminación de registros.
- Generación de secciones incluyendo su dibujo, acotación y cálculo de propiedades mecánicas. Se dibuja y se acotado la sección permitiendo una mejor comprensión por el usuario.
- Almacenamiento de proyectos en ficheros específicos de la aplicación.
- Generación de esquemas de camión incluyendo su acotación. Se dibuja un esquema correspondiente a las medidas del camión y se acotan para la visualización del problema por parte del usuario.
- Edición de cargas de forma gráfica con la representación específica correspondiente. Se ha utilizado la representación gráfica comúnmente aceptada para todos los tipos de carga incluyendo la posibilidad de introducción de cargas distribuidas para cargas no puntuales.
- Realización de algoritmos simplificados de cálculo. Se ha realizado una investigación de las distintas posibilidades de cálculo que pueden emplearse, estudiando posibilidades de utilización y generando algoritmos simplificados.
- Elaboración de diagramas de cortantes.
- Elaboración de diagramas de momentos.
- Cálculo de las cargas sobre cada eje y tándem. Se ha incluido la carga por tándem y eje.
- Cálculo de la tensión sobre las vigas del bastidor.
- Avisos sobre exceso de cargas en los ejes. El programa emite avisos en caso de sobrepasar las cargas máximas por eje
- Generación en PDF del informe de anexo técnico para la presentación en la ITV. La información se presenta (además de en pantalla) en un archivo PDF que permite al usuario incluirlo directamente en su informe.

Con todo, resulta una herramienta completa y que da respuesta a cada uno de los puntos necesarios con un grado mayor al esperado.

La herramienta resulta ser un avance importante al planteamiento de los cálculos sobre los carrozados, ya que permite pasar de herramientas demasiado generalistas de cálculo de estructuras, que requieren realizar “apaños” sobre el modelo, a calcular utilizando un sistema específicamente diseñado para estas, que permite una rapidez importante en la introducción de datos y en la obtención de resultados.

No resulta menos importante, en mi opinión, el hecho de la mejora importante en la gestión de modificaciones en las cargas o posiciones de los apoyos que, para otras herramientas genéricas, supone en muchas ocasiones tener que remodelar completamente el sistema o incluso realizar varios modelos. En estos sistemas genéricos esto implica empezar la introducción de datos desde el comienzo resultando en tiempo innecesario.

Cumplimiento con plazos previstos

Se han respetado la mayoría de los plazos estipulados, si bien existen retrasos en algunas de las fases. En tareas de las fases expuestas en el diagrama Gantt se aprecian retrasos que tienen que ver con un mayor estudio sobre la metodología de cálculo que puede mejorar el algoritmo de cálculo y las mejoras futuras..

Se debe tener en cuenta que las metodologías de cálculo de estructuras son muy variadas y más o menos complejas en función de los resultados o el tipo de cálculo que se quiere realizar.

Al comenzar las fases relativas al cálculo, me he ido dando cuenta de que ciertas metodologías de cálculo eran inapropiadas por la complejidad y otras por la inexactitud, por lo que una parte importante de esta tarea ha sido la de analizar cuál es el sistema más eficiente.

Conclusiones generales

Dada la finalidad del proyecto, se considera que el punto de requerimientos y mejoras resulta el más importante del proyecto, restando importancia a los posibles retrasos. Dicha mejora sobre los requisitos esperados supone un extra de una importancia mucho mayor que las desviaciones en el planning inicial.

Se considera que el proyecto en conjunto ha sido realizado además de eficientemente y eficazmente, de manera que ha permitido a la herramienta resultar atractiva al usuario.

7.3. Trabajos futuros

Se exponen en este apartado posibles mejoras realizables sobre la herramienta.

Inclusión de otro tipo de vehículos

La herramienta está específicamente desarrollada para camiones rígidos y articulados. Sin embargo, dada la manera de manejar el cálculo y la introducción de los datos, otro tipo de vehículos podrían ser introducidos sin mucha complicación. Algunos de estos tipos de vehículos podrían ser:

- Autobuses.
- Autocaravanas.
- Automóviles con modificaciones especiales.
- Furgonetas.
- etc.

En todos estos casos se podría realizar un tratamiento similar al de los camiones rígidos.

Inclusión de otro tipo de cálculos

Se podrían incluir en la herramienta cálculos de otra naturaleza:

- Fijaciones, que anclan los elementos de transporte al bastidor.
- Cálculos a vuelco en giros, en los que se incluiría una variable adicional de altura de la carga para controlar que el vehículo no vuelque en los giros.
- Cálculos del bastidor en frenada, en los que se tienen en cuenta las cargas longitudinales de las vigas del bastidor.

Inclusión de otro tipo de ayudas auxiliares

Otro punto posible de mejora sobre la aplicación podría ser la de incluir más ayudas a la hora de gestionar determinadas informaciones.

- Posicionamiento en el gráfico de elementos de carga mediante un sistema drag & drop.
- Calculo de pesos y cargas en función de la utilidad del vehículo, en el que se diferenciarían casos como el transporte de líquidos, gases, vehículos, etc.
- Importaciones de bibliotecas de fabricantes de camiones.
- Gestión de la validación del proyecto completo de carrozado con la ITV.
- Inclusión de requerimientos legales para evitar su incumplimiento antes de llegar a la ITV.

A pesar de parecer un sector relativamente pequeño, debe tenerse en cuenta que cualquier modificación realizada a un vehículo debe cumplir con una serie de requerimientos, entre los que está soportar correctamente la carga para la que se va a modificar, por lo que el abanico de posibles partes interesadas en este tipo de aplicación es muy amplio. Este proyecto recoge una parte amplia

de ese abanico y se considera que con la realización del mismo se ha dado respuesta a este requerimiento en el sector mejorando algunos de los aspectos ya existentes.

Bibliografía

- Apache DB Project. **Apache Derby**.
  https://db.apache.org/derby
- Apache Maven Project (July 2023). **Welcome to Maven.**
  https://maven.apache.org
- Apple.com (2023). **Pages.**
  https://www.apple.com/es/pages/
- Aprise (2023). **Itext Suite.**
  https://itextpdf.com/products/itext-7
- Cervera Ruiz, Miguel; Blanco Díaz, Elena (febrero 2015). **Resistencia de materiales.**
  Escuela técnica superior de caminos canales y puertos de Barcelona.
  http://cervera.rmee.upc.edu/libros/Resistencia%20de%20Materiales.pdf
- Change Vision Inc. (2009-2022). **Astah Reference manual version 9.0.**
- desarrolloweb (2023). **Introducción a XML.**
  https://desarrolloweb.com/manuales/18
- Eclipse Foundation (June 2023). **Eclipse Installer.**
  https://www.eclipse.org/downloads/packages/
- Gómez, Roi (2019). **Un sistema de apoyo a la gestión de ambulancias en un centro urbano.**
  UNED. Proyecto Fin de Grado
- IBM (marzo 2021). **El modelo de diseño.**
  https://www.ibm.com/docs/es/rsm/7.5.0?topic=model-design
- Java (2023). **Help Resources.**
  https://www.java.com/en/download/help/whatis_java.html
- Michelin (diciembre 2021). **Aprende a calcular la TARA de tus vehículos.**
  https://connectedfleet.michelin.com/es/blog/que-es-la-tara-de-un-vehiculo-y-como-se-calcula
- Microsoft (2023). **Access SQL: Conceptos básicos, vocabulario y sintaxis.**
  https://support.microsoft.com/es-es/office/access-sql-conceptos-básicos-vocabulario-y-sintaxis-444d0303-cde1-424e-9a74-e8dc3e460671

- Microsoft. **Microsoft Patterns&Practices Team, Microsoft Application Architecture Guide, 2a. ed.**

  Microsoft Press (2009)
- Ministerio de transportes, movilidad y agenda urbana. BOE núm. 22, de 26/01/1999. Real Decreto 2822/1998 de 23 de diciembre,. **Masas máximas por eje permitidas.**

  https://www.mitma.gob.es/transporte-terrestre/inspeccion-y-seguridad-en-el-transporte/pesos-y-dimensiones/pesos/pesos-por-eje
- Noteboom (2003). **Programa de cálculo de carga por eje NOVAB 3.0.**

  https://www.nooteboom.com/es/asistencia/programa-de-calculo-de-carga-por-eje-novab-3-0/
- Onroad. **La masa por eje de un vehículo.**

  https://www.onroad.to/practico/aprender-conducir/carga-vehiculo/masa-eje
- PngWing. **Rueda de chasis de coche neumático, coche, coche, modo de transporte, vehículo png.**

  https://www.pngwing.com/es/free-png-ppdcd/download
- Pressman, Roger; Maxim Bruce R.(2020). **Ingeniería de Software. Un enfoque práctico.**

  McGraw Hill.
- TrailerWin. **Trailer consultation. TrailerWin.**

  https://www.trailerwin.com
- SkyCiv Engineering (2023). **SkyCiv**

  https://skyciv.com/structural-software/beam-analysis-software/
- Wikipedia (marzo 2022). **Acoplamiento(Informática).**

  https://es.wikipedia.org/wiki/Acoplamiento_(informática)#Bajo_acoplamiento

.

Glosario

- ITV: Inspección técnica de vehículos.
- GUI: Graphic User Interface.
- JVM: Java Virtual Machine.
- SQL: Structured Query Language.
- kgf: kilogramo-fuerza.
- N: Newton.
- mm: milímetro.
- cm: centímetro.
- dm: decímetro.
- m: metro.
- kgf·m: kilogramo-fuerza metro.
- σ: Tensión normal.
- N: Esfuerzo normal.
- A: Sección resistente.
- M: Momento flector.
- W: Módulo resistente.
- τ: Tensión cortante.
- V: Esfuerzo cortante.
- σ_{VM}: Tensión de Von Mises.
- TARA: Masa vacía del vehículo.
- M.M.A: Masa máxima autorizada.
- CAD: Computer Aided Drawing.

Anexo A. Manual de instalación

A.1. Instalación inicial

La herramienta se provee en una carpeta comprimida. Para su instalación es necesario disponer de la máquina virtual de Java instalada en el ordenador en el que se quiere utilizar la aplicación. Dicha máquina virtual puede descargarse de la página web de Oracle.
Una vez instalada la JVM, sólo es necesario descomprimirla en la ubicación deseada. Se extraerán un conjunto de carpetas necesarias para su funcionamiento tal como se muestra en la Figura A-1.

- La carpeta “Derby” contiene la información necesaria para la gestión de las bases de datos.
- La carpeta “Imágenes” contiene los recursos necesarios para la aplicación, como las imágenes de fondo o los iconos de los logos.
- La carpeta “IText” contiene la información necesaria para la generación de los documentos pdf.
- El fichero ejecutable “TruckCalc.jar” contiene el ejecutable principal de la herramienta.

Una vez se realice doble click sobre “TruckCalc.jar “ se lanzará la aplicación.

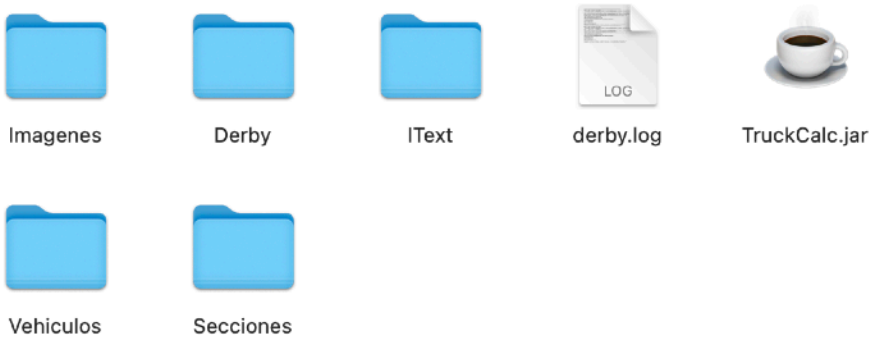


Figura A-1. Carpetas y ejecutable de la aplicación una vez descomprimido el fichero.

A.2. Notas sobre la instalación inicial

En la primera instalación de la herramienta no se dispone de ninguna información relativa a las secciones y camiones que se contienen en las bases de datos.

Una vez se exporte el primer camión o la primera sección a la base de datos, se crearán las bases de datos de camiones y de secciones y se generarán sendas carpetas denominadas “Vehículos” y “Secciones” así como un fichero denominado “Derby.log” que se utilizarán para la gestión de las informaciones de la base de datos.

Si, a posteriori, se desease trasladar la aplicación a otra ubicación, sería necesario trasladar todas las carpetas indicadas en el punto 1 anterior así como las carpetas y ficheros comentadas en este punto 2 a la nueva ubicación para no perder la información contenida en las bases de datos creadas.

Anexo B. Manual de uso

B.1. Ventana Principal

Para el arranque de la herramienta sólo es necesario realizar doble click sobre el archivo "TruckCalc.jar". Una vez realizada esta acción, aparecerá en pantalla unos breves segundos una portada inicial con información inicial sobre la misma.

Tras estos segundos iniciales aparecerá la ventana principal de la aplicación que se divide en las siguientes zonas:

- **Zona de entrada de datos:** Esta zona está preparada para recoger la información principal del cliente y proyecto. Sólo es necesario hacer click en cada una de las cajas de texto y rellenar la información correspondiente.
- **Zona informativa:** Muestra información relativa a las acciones del usuario para informar de los pasos que ha seguido por si éste tuviese alguna duda en la introducción de los datos.
- **Barra de menús:** Esta parte es la encargada de gestionar los menús. Contiene el acceso a los siguientes menús:
  - **Archivo:** Accedemos con ello a la recuperación o almacenamiento de archivos de proyecto.
  - **Base de Datos:** Accedemos con él a la gestión de las bases de datos.
  - **Unidades:** Podemos realizar el cambio de unidades en éste menú.
  - **Información:** Provee información genérica de la aplicación.
- **Zona de selección:** Contiene una zona izquierda para trabajar con vehículos rígidos y otra derecha para trabajar con articulados. En ambos casos, permite seleccionar el número de ejes del vehículo y la configuración del mismo. Para ello, únicamente es necesario hacer click en el vehículo necesario.

- **Zona de proceso:** Da acceso al menú de eliminación de datos, de validación de los datos introducidos y de acceso a otras funcionalidades de la herramienta como la gestión de estructura, cálculo de la estructura y obtención del fichero de resultados.

Se muestra en la Figura B-1 la ventana principal de la aplicación con las distintas áreas que la componen.

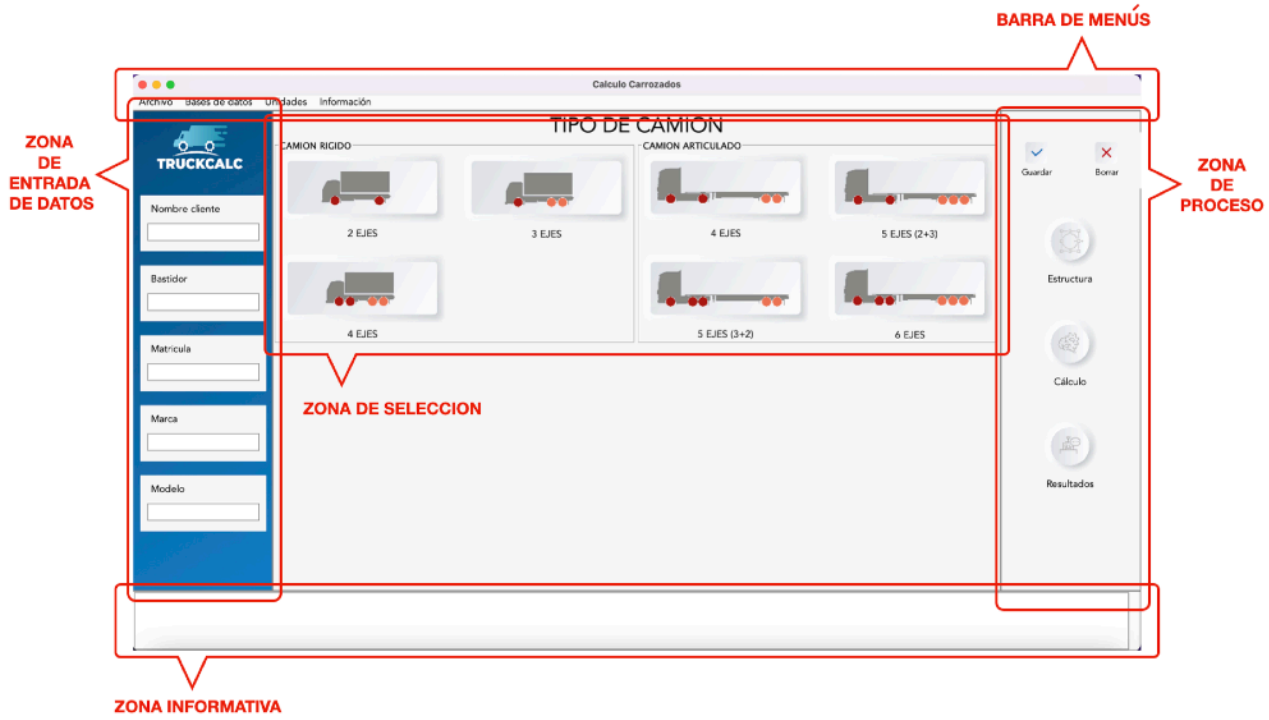


Figura B-1. Zonas de ventana principal

La herramienta informará en cualquier caso de la información que pueda faltar si intentamos trabajar en un orden incorrecto. No obstante, el orden natural para realizar la gestión completa de un proyecto es la siguiente.

Para comenzar a generar un proyecto comenzaremos a trabajar con la zona de entrada de datos introduciendo los datos de nuestro proyecto.

Una vez introducidos los datos, nos moveremos a la zona de selección para escoger el tipo de camión con el que vamos a trabajar.

En el momento de la selección, la herramienta mostrará ese icono como opción marcada, mostrará una vista algo más clara de la selección y requerirá para ella la información a rellenar relativa a los datos dimensionales del proyecto.

La figura B-2 muestra la información que se requiere al pulsar camión rígido, si bien la cantidad de información a introducir varía de la opción camión rígido a la opción camión articulado.

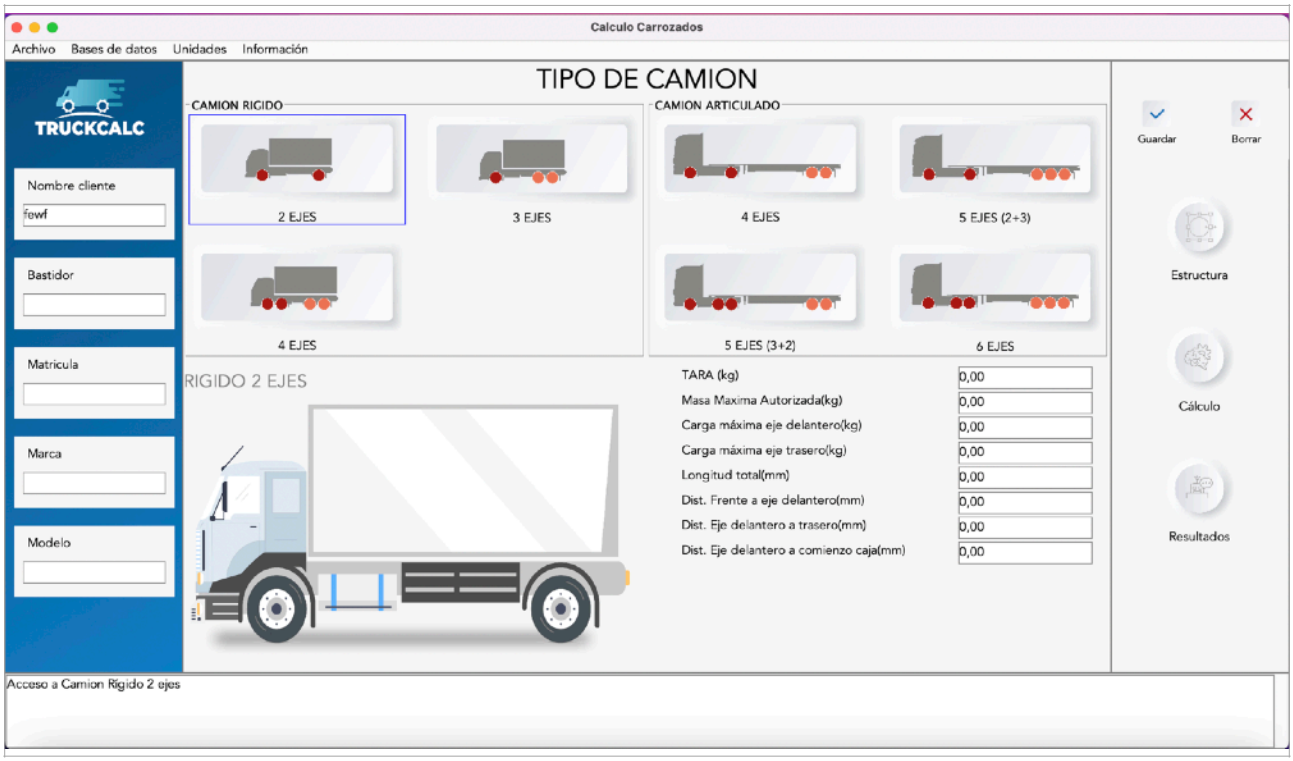


Figura B-2. Información mostrada al pulsar en el icono

Opciones Guardar/Borrar

Una vez introduzcamos la información, pulsaremos el botón “Guardar” marcado con una V de color verde y los datos quedarán registrados en la herramienta (nota: Esta acción no implica en ningún caso que la información se almacene definitivamente en el disco duro sino que supone un almacenamiento temporal de la información hasta guardarla posteriormente.

Si al presionar el botón “Guardar”, la aplicación detecta cualquier tipo de error en los datos introducidos, se mostrará un mensaje de aviso para corregir la información incorrecta y no permitirá continuar con ella. También puede ocurrir que la información sea correcta pero no recomendable o

adecuada a las prácticas comunes. En tal caso se mostrará una advertencia pero el sistema sí dejará continuar si no se corrigen los datos.

En caso de que la información no sea correcta y quiera comenzarse desde cero puede pulsarse el botón "Borrar". En este caso la aplicación mostrará un mensaje de aviso para confirmar si se desean borrar los datos y en caso de confirmar, se eliminará la información introducida.

NOTA: Si el usuario desea continuar trabajando con el proyecto entrando al cálculo estructural pero no presiona el botón "Guardar", la aplicación simplemente guardará por defecto la información que se ha introducido debido a que no es posible comenzar a trabajar sin ella.

MENÚ ARCHIVO

El menú "Archivo" permite la carga y almacenamiento de proyectos realizados.

Para almacenar la información existente en un archivo de proyecto se pulsa sobre el menú "Archivo" y se selecciona la opción "Guardar".

Aparecerá un menú de directorios donde el usuario puede seleccionar la ubicación en la que desea almacenar la información como se ve a continuación, en la Figura B-3.

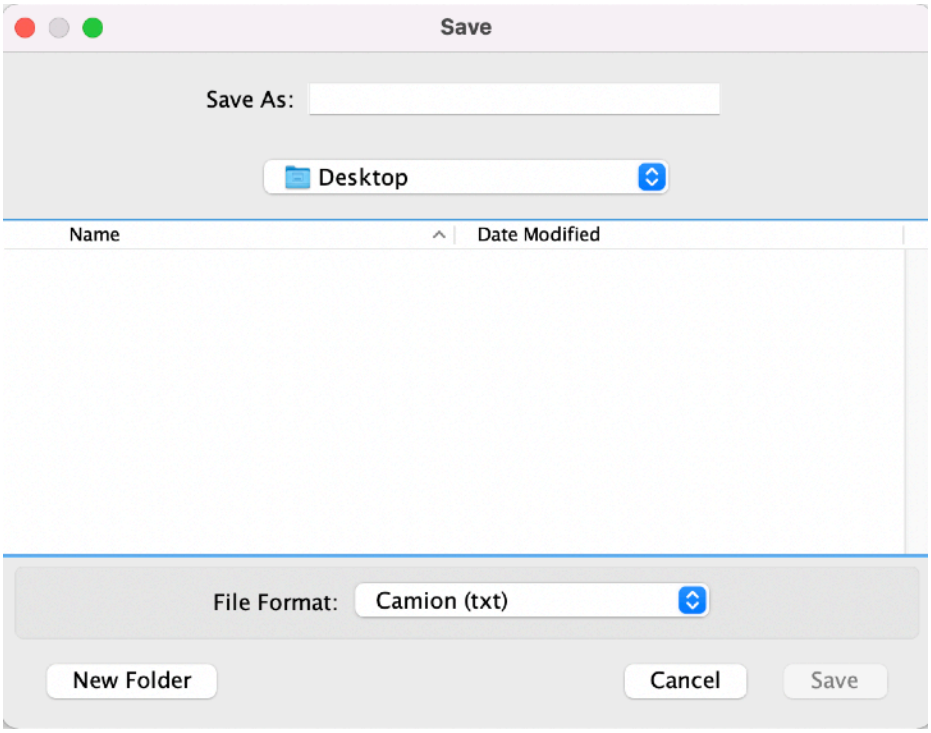


Figura B-3. Menú archivo. Opción guardar.

- Introducimos el nombre a guardar en la caja de texto “Save As”.
- También es posible seleccionar el directorio en el que realizar el almacenamiento del archivo.
- Podemos seleccionar la extensión del archivo mediante la opción “File Format”.
- Finalmente podemos crear un nuevo directorio mediante la opción “New Folder”

Una vez escogemos nuestras preferencias hacemos click en “Save” para guardar el archivo o “Cancel” para cancelar la operación.

NOTA: Por defecto los archivos de proyecto de TruckCalc tienen extensión *.txt.

Para cargar un archivo haremos click en el menú “Archivo” y pulsaremos sobre “Abrir”. Se mostrará el menú de la Figura B-4.

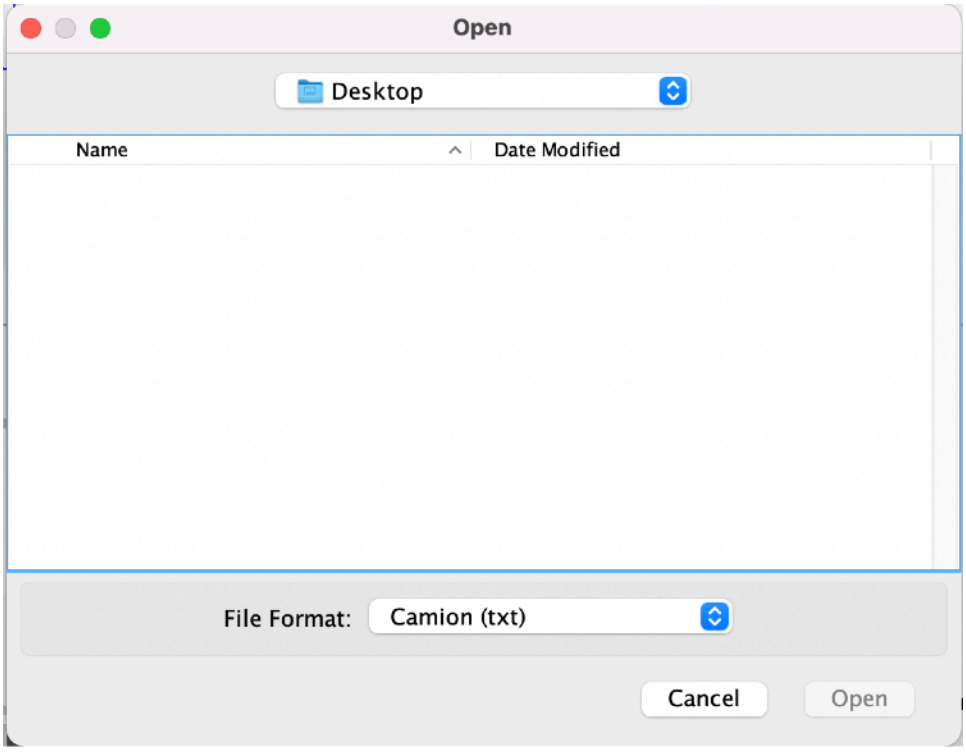


Figura B-4. Menú archivo. Opción abrir

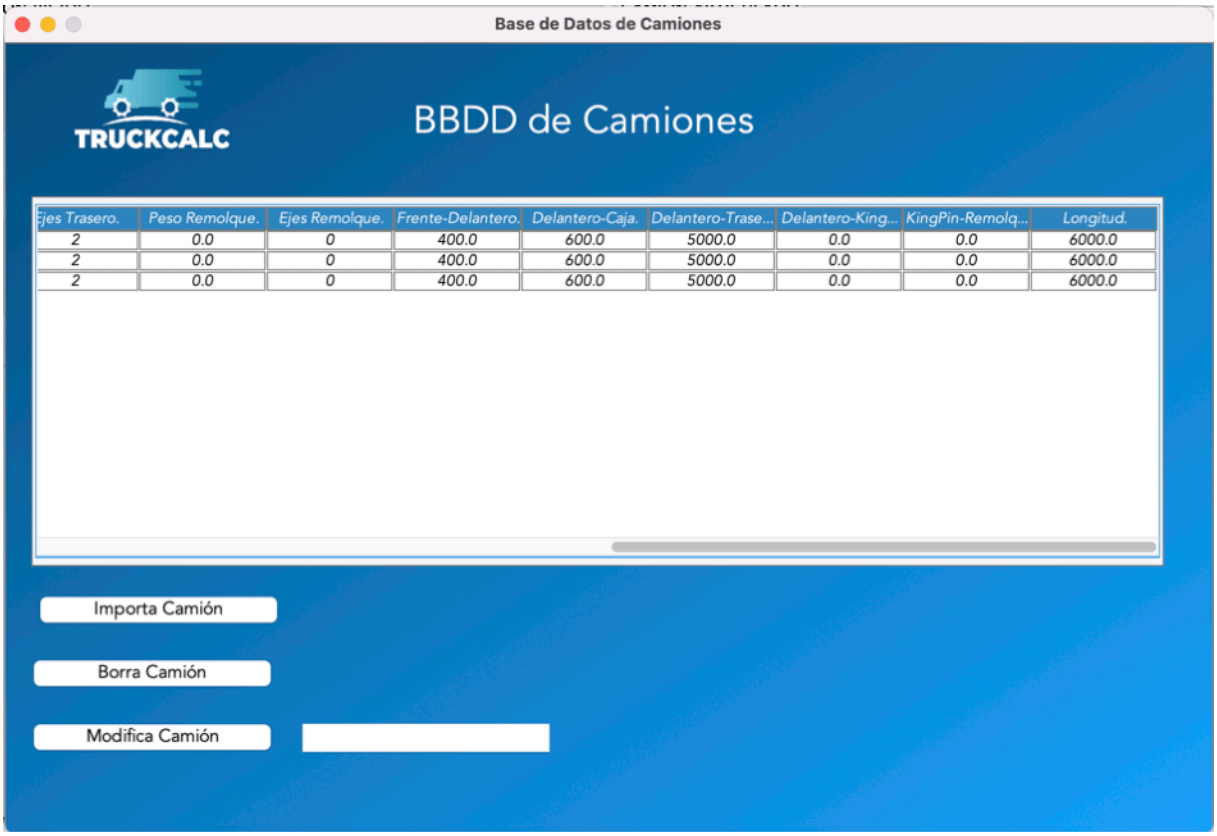
- Seleccionamos el directorio.
- Selecionamos el formato.
- Pinchamos sobre el fichero a cargar.

- Pulsamos sobre “Open” en caso de querer abrirlo o “Cancel” si queremos cancelar la operación.

MENÚ BASE DE DATOS

El menú “Base de Datos” gestiona en esta pantalla la importación de camiones almacenados en la base de datos o bien la exportación de los datos introducidos a la base de datos como un nuevo camión.

Para importar de la base de datos, iremos a la barra de menús y seleccionaremos “Importar Camión de Base de Datos”. La herramienta mostrará la ventana de la Figura B-5 en pantalla:



Ejes Trasero.	Peso Remolque.	Ejes Remolque.	Frente-Delantero.	Delantero-Caja.	Delantero-Trase...	Delantero-King...	KingPin-Remolq...	Longitud.
2	0.0	0	400.0	600.0	5000.0	0.0	0.0	6000.0
2	0.0	0	400.0	600.0	5000.0	0.0	0.0	6000.0
2	0.0	0	400.0	600.0	5000.0	0.0	0.0	6000.0

Figura B-5. Menu bases de datos. Importar Camión

Se muestran ordenadas por filas las diferentes opciones disponibles en la base de datos.

En dicha ventana disponemos de tres botones:

- Importa Camión.
- Borra Camión.
- Modifica Camión.

y de una caja de texto junto al botón “Modifica Camión”.

Importar

Para importar un camión se selecciona la opción deseada de la base de datos haciendo click en cualquiera de los valores de la misma. La opción seleccionada se marcará colocando todos sus valores en color rojo.

Una vez la opción ha cambiado a rojo se pulsa sobre el botón "Importa Camión" y los datos del camión se irán rellenando en las diferentes casillas de la pantalla principal.

Borrar Camión

Para borrar un camión se selecciona la opción deseada de la base de datos haciendo click en cualquiera de los valores de la misma. La opción seleccionada se marcará colocando todos sus valores en color rojo.

Una vez la opción ha cambiado a rojo se pulsa sobre el botón Borra Camión.

La aplicación eliminará el camión seleccionado.

NOTA: Es importante remarcar que la eliminación se produce de manera definitiva y no sólo en la tabla mostrada en la ventana de base de datos, es decir, la opción eliminada no volverá a estar disponible a menos que vuelva a exportarse a la base de datos.

Modificar Camión

La modificación de cualquier camión almacenado se realiza valor por valor.

Para modificar un valor de la base de datos se hace click sobre el valor a modificar. El camión seleccionado marcará colocando todos sus valores en color rojo y además el valor sobre el que hemos hecho click aparecerá en la caja de texto junto a "Modifica Camión".

Escribiremos en la caja de texto el valor que queremos modificar y pulsaremos sobre "Modifica Camión". La base de datos actualizará el valor de ese campo con el nuevo valor introducido para el registro indicado.

Guardar un nuevo camión en la base de datos

Para almacenar un nuevo camión en la base de datos partiremos de la ventana principal de introducción de datos de los camiones.

Una vez rellenados los datos haremos click sobre el menú “Bases de Datos” y seleccionaremos “Exportar Camión a Base de Datos”. El camión se almacenará como un nuevo registro en la base de datos.

NOTA: Cada registro de camión de la base de datos tiene asignado un número de registro único. Cuando se produce una eliminación en la base de datos, los números de referencia de los registros eliminados serán reutilizados por la aplicación al introducir nuevos valores previamente a la creación de nuevos valores para evitar desperdigar dichos números y mantener un orden sobre los mismos.

MENÚ UNIDADES

El menú unidades permite cambiar las unidades con las que se trabaja en la aplicación para adaptarlas al requerimiento o preferencia del usuario.

Existen dos magnitudes fundamentales en la aplicación: Longitud y Fuerza.

- **Longitud.** Permite seleccionar las siguientes unidades:
  - metro (mm).
  - decímetro (dm).
  - centímetro (cm).
  - milímetro (mm).
- **Peso.** Permite seleccionar las siguientes unidades:
  - kilogramo (kg) (se entiende que se refiere a kilogramo-fuerza o kilopondio).
  - Newton (N).

Para adaptar las unidades a cualquiera de las unidades indicadas se selecciona unidades y se despliega un sub-menú con dos alternativas “Unidades Longitud” y “Unidades Carga”. Cada uno de ellos contiene las alternativas anteriormente indicadas por lo que solo se necesita hacer click sobre la opción requerida. Las dimensiones y cargas introducidas se adaptarán a las unidades que se seleccionen.

MENÚ INFORMACION

El menú información contiene una única opción denominada “Acerca de TruckCalc”. Dicha opción muestra la información de la Figura B-6 en pantalla.

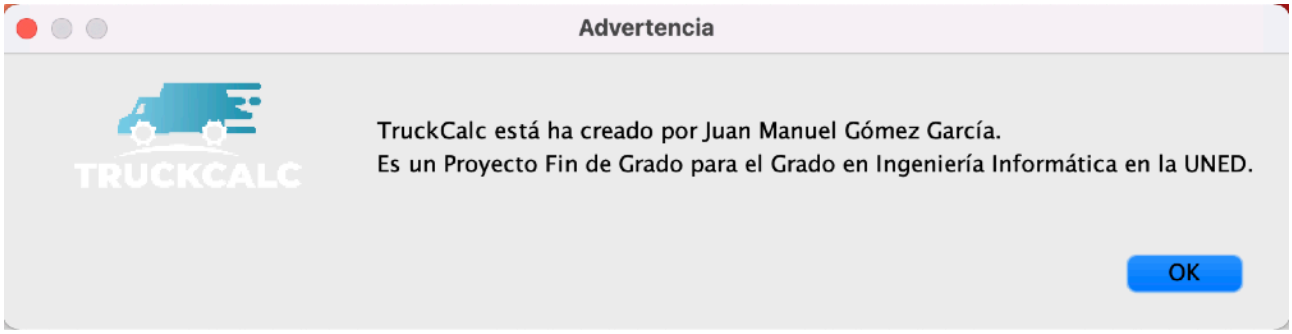


Figura B-6. Menu ayuda. Información mostrada

B.2. Ventana Estructura

Previamente al acceso a la Pantalla de Estructura es necesario seleccionar un tipo de camión. De no haberse seleccionado, la aplicación mostrará un mensaje de aviso para que el usuario realice esta acción.

La ventana Estructura se encarga de la gestión de las secciones que se utilizarán en el cálculo de la estructura del camión. Está dividida en las siguientes zonas de la Figura B-7.

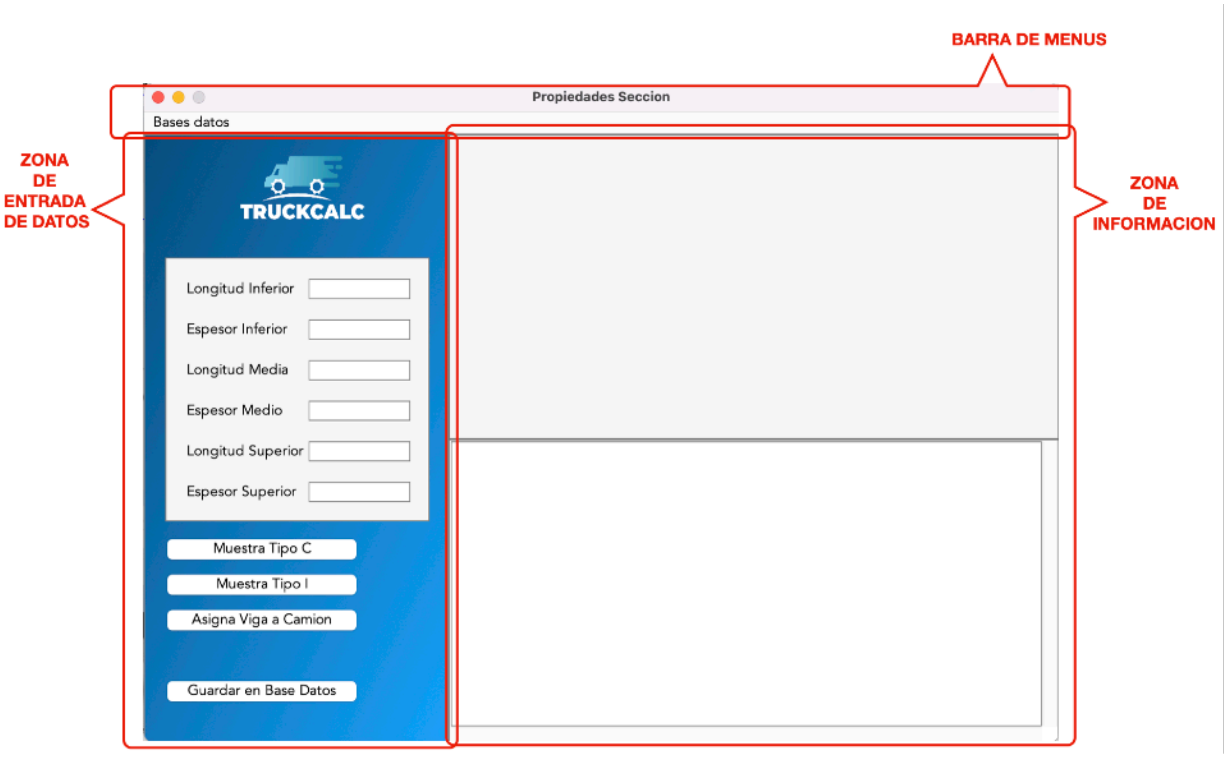


Figura B-7. Zonas de ventana estructura

- **Zona de entrada de datos:** Esta zona está preparada para recoger la información principal de las secciones así como para poder requerir una consulta sobre la información introducida, asignar la viga al proyecto y camión y exportar la sección a la base de datos.
- **Zona de información:** Muestra información relativa a las secciones cuando el usuario lo requiere. La parte superior muestra un boceto de la sección con las cotas de la misma y la inferior hace un resumen del cálculo de las propiedades mecánicas que tiene la sección introducida.
- **Barra de menús:** Esta parte es la encargada de gestionar los menús. Contiene el acceso a los siguientes menús:
  - **Base de Datos:** Accedemos con él a la gestión de las bases de datos de secciones.

Creación de una sección

El proceso de creación de una sección parte de la introducción de los datos de la misma. A efectos de la aplicación, una sección puede tener una geometría en C o en I. En ambos casos se requiere la longitud y espesor de los rectángulos que forman la misma.

Una vez introducidos estos datos, el usuario puede pulsar en "Muestra tipo C" o "Muestra tipo I" si tiene dudas sobre los datos introducidos. Al pulsar cualquiera de los botones, se mostrará la geometría creada con sus cotas así como las propiedades de la sección introducida como se muestra en la Figura B-8, que recoge la información de una sección en la que todas sus partes tienen una longitud de 50 mm y un espesor de 3 mm.

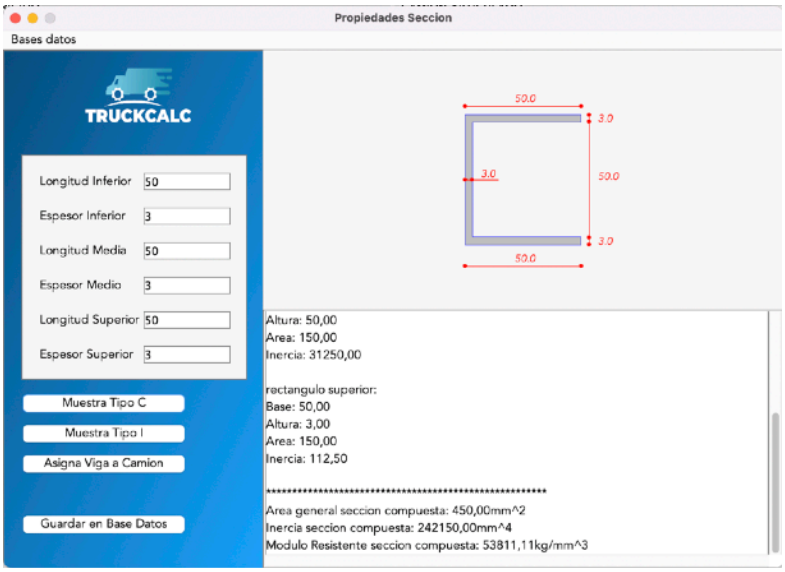


Figura B-8. Ejemplo generación de una sección tipo C

Una vez el usuario ha encontrado el perfil necesario para la estructura puede hacer click en el botón "Asigna Viga a Camión" para que el camión la adopte.

Guardado de una sección en la base de datos

Una vez creada una sección, el usuario puede guardarla en la base de datos de secciones mediante la pulsación del botón "Guardar en base de datos"

MENÚ BASE DE DATOS

El menú "Base de Datos" gestiona en esta pantalla la importación de camiones almacenados en la base de datos o bien la exportación de los datos introducidos a la base de datos como un nuevo camión.

Para importar de la base de datos iremos a la barra de menús y seleccionaremos "Importar Camión de Base de Datos". La herramienta mostrará la ventana de la Figura B-9 en pantalla.

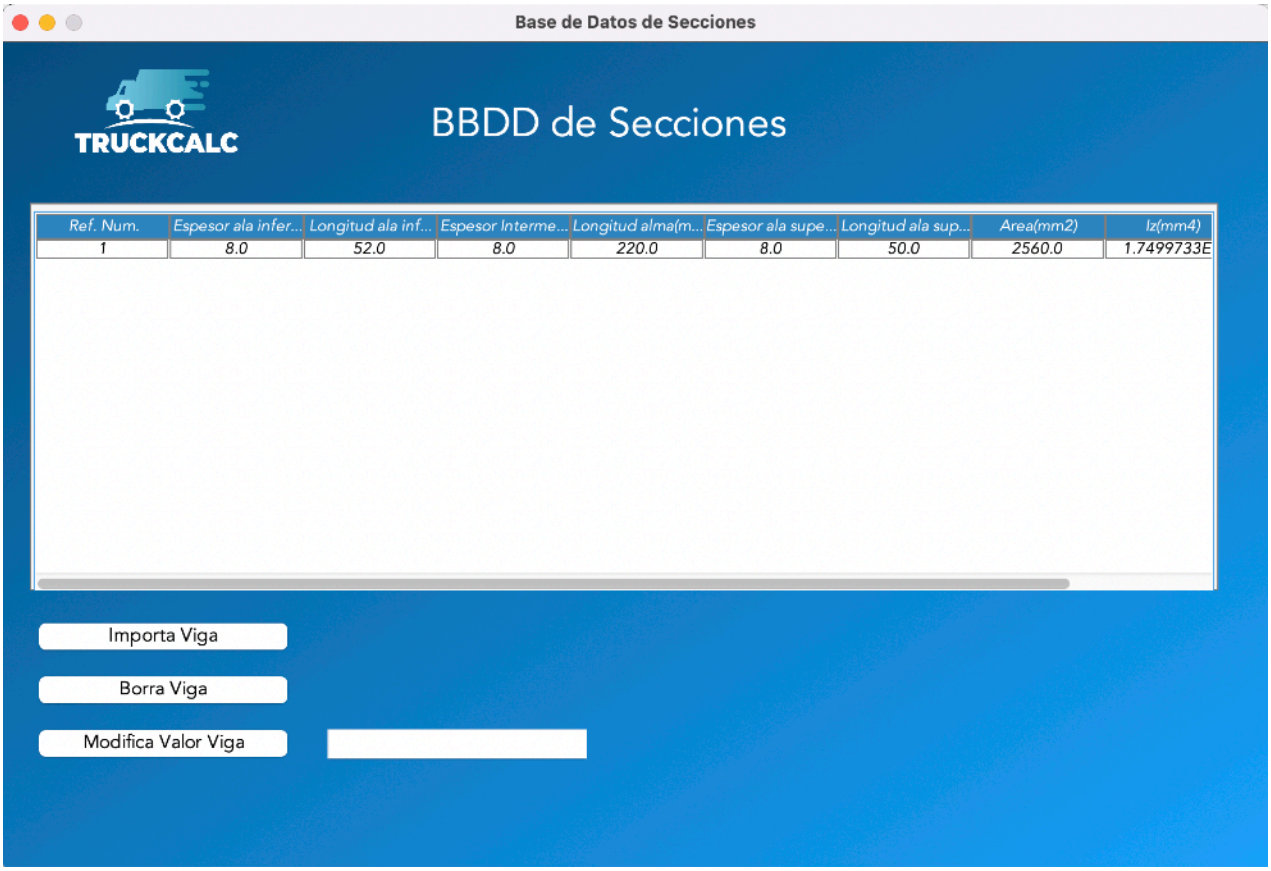


Figura B-9. Menú base de datos secciones.

Se muestran ordenadas por filas las diferentes opciones disponibles en la base de datos.
En dicha ventana disponemos de tres botones:

- Importa Viga.
- Borra Viga.
- Modifica Valor Viga.

y de una caja de texto junto al botón “Modifica Valor Viga”.

Importar Viga

Para importar una sección se selecciona la opción deseada de la base de datos haciendo click en cualquiera de los valores de la misma. La opción seleccionada se marcará colocando todos sus valores en color rojo.
Una vez la opción ha cambiado a rojo se pulsa sobre el botón “Importa Viga” y los datos de la viga se irán rellenando en las diferentes casillas de la pantalla principal.

Borrar Viga

Para borrar una viga se selecciona la opción deseada de la base de datos haciendo click en cualquiera de los valores de la misma. La opción seleccionada se marcará colocando todos sus valores en color rojo.
Una vez la opción ha cambiado a rojo se pulsa sobre el botón Borra Viga.
La aplicación eliminará la viga seleccionada.

NOTA: Es importante remarcar que la eliminación se produce de manera definitiva y no sólo en la tabla mostrada en la ventana de base de datos, es decir, la opción eliminada no volverá a estar disponible a menos que vuelva a exportarse a la base de datos.

Modifica Valor Viga

La modificación de cualquier viga almacenada se realiza valor por valor.
Para modificar un valor de la base de datos se hace click sobre el valor a modificar. La viga seleccionada se marcará colocando todos sus valores en color rojo y además el valor sobre el que hemos hecho click aparecerá en la caja de texto junto a “Modifica Valor Viga”.

Escribiremos en la caja de texto el valor que queremos modificar y pulsaremos sobre “Modifica Valor Viga”. La base de datos actualizará el valor de ese campo con el nuevo valor introducido para el registro indicado.

NOTA: Cada registro de vigas de la base de datos tiene asignado un número de registro único. Cuando se produce una eliminación en la base de datos, los números de referencia de los registros eliminados serán reutilizados por la aplicación al introducir nuevos valores previamente a la creación de nuevos valores para evitar desperdigar dichos números y mantener un orden sobre los mismos.

B.3. Ventana Cálculo

La ventana cálculo da acceso a la información específica de cálculo que se incluirá posteriormente en el anexo técnico.
La ventana tiene las siguientes zonas, mostradas en la Figura B-10.

Figura B-10. Ventana cálculo. Zonas.

- **Zona de esquemas:** Esta zona está dividida verticalmente en tres áreas que muestran la siguiente información respectivamente:

- Esquema del camión. Contiene las dimensiones generales del camión, sus posiciones de los tándem y el número de ejes en cada tándem.
- Diagrama de cortantes. Contiene un gráfico en el que se muestra en el eje horizontal la posición dentro de la longitud de camión y en el eje vertical el valor de su respectivo esfuerzo cortante. Indica el valor puntual en máximos o mínimos.
- Diagrama de momentos flectores. Contiene un gráfico en el que se muestra en el eje horizontal la posición dentro de la longitud de camión y en el eje vertical el valor de su respectivo momento flector. Indica el valor puntual en máximos o mínimos.

- **Zona Cargas Ejes:** Muestra información relativa a las reacciones producidas sobre cada eje en la situación actual de cargas introducidas.
- **Zona Información Cargas:** Muestra una lista de las cargas introducidas indicando su nombre, tipología y los datos más importantes de cada una de ellas.
- **Zona de Proceso:** Muestra herramientas para la gestión de la visualización sobre el camión así como para la introducción, modificación y eliminación de cargas.

Crear Carga

Para crear una carga pulsaremos sobre el icono “Nueva” de la zona de proceso.

Se mostrará una ventana como la de la Figura B-11.

Figura B-11. Ventana introducir carga

Para crear la carga, simplemente necesitamos rellenar la información solicitada:

- Nombre de la misma.
- Punto de aplicación desde el extremo izquierdo.
- Valor de la carga (en las unidades mostradas en el texto superior). Para cargas puntuales el valor será el de fuerza, para cargas distribuidas el de fuerza por unidad de longitud.
- Tipo de carga. Puede solicitarse carga puntual o carga distribuida. En caso de solicitar carga distribuida, se requerirá también la longitud desde el punto de aplicación en la que se aplica la carga.

Una vez introducidos los datos se pulsa sobre el botón "Guardar" y se crea la carga. Mostrándose en su posición y valor en el esquema y actualizando el esquema de cortantes y de momentos flectores con la nueva carga.

La Figura B-12 muestra como ejemplo una carga puntual de 1000 kgf a 3000 mm y una carga distribuida de 1 kgf/mm a 4000 mm y con una longitud de 1000 mm.

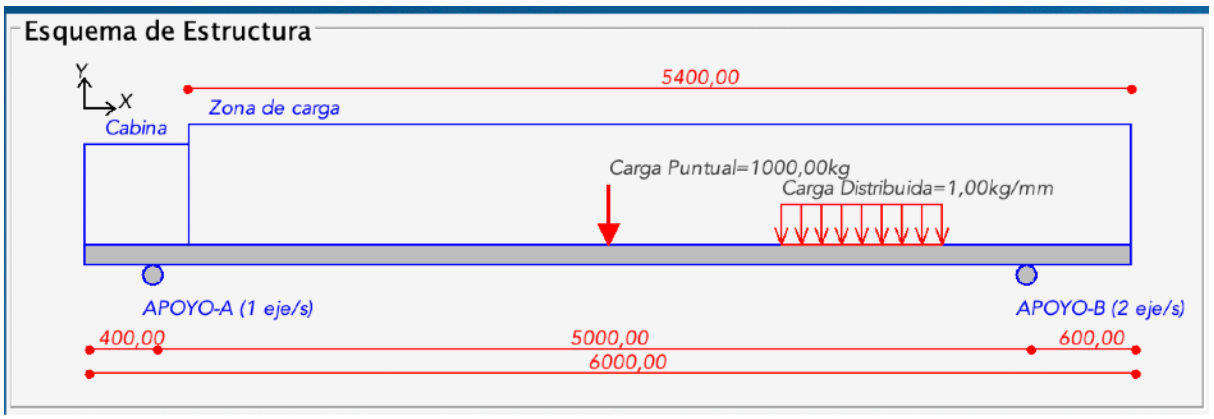


Figura B-12. Cargas mostradas en el esquema

Borrar Carga

Para eliminar una carga creada sólo es necesario pulsar sobre el icono "Elimina" y seleccionar la carga en el desplegable de la ventana emergente mostrada.

Una vez seleccionada la carga, se pulsará el botón "Eliminar" y la carga se eliminará, desapareciendo del esquema de estructura y actualizando los diagramas de cortantes y de momentos.

Figura B-13. Ventana eliminar carga

La Figura B-13 muestra la ventana "Eliminar Carga" con el desplegable de cargas en el que se seleccionará la deseada para eliminar.

Ver

El botón "Ver" nos permite modificar la visualización para poder tener un esquema más claro cuando se juntan demasiadas cargas en él.

Al pulsarlo se mostrará una ventana emergente en la que una vez más se dispondrá de un menú desplegable con las cargas introducidas. El usuario puede seleccionar en él la carga con la que desea operar. Se muestra la ventana en la Figura B-14.

Bajo el desplegable se encuentran tres botones con tres funcionalidades distintas:

- Ver: Muestra la carga seleccionada conjuntamente con la información que ya se está visualizando.
- Ocultar: Oculta la carga seleccionada de la información que se está visualizando.
- Aislar: Muestra únicamente la carga y ninguna otra carga adicional.

Figura B-14. Ventana visualizar carga

Analizar Cabeza / Analizar Remolque

Estos dos botones alternan la parte que se quiere analizar en el caso de un camión articulado. En el caso de pulsar en el botón "Analizar Cabeza", el diagrama de cortantes y el diagrama de momentos flectores acortarán la distancia analizada a la de la cabeza tractora. De esta manera se trabajará con una escala adecuada de gráfica para la zona requerida.

De igual forma, si se pulsa el botón "Analizar Remolque", los diagramas de cortantes y de momentos flectores se adaptarán para mostrar la zona de remolques.

Se muestra a continuación, en la Figura B-15, el análisis de un camión articulado y la adaptación de la zona de esquemas cuando se pulsa cada uno de los botones. Se puede apreciar como los diagramas se ciñen a la zona solicitada prescindiendo de la zona que no es de interés.

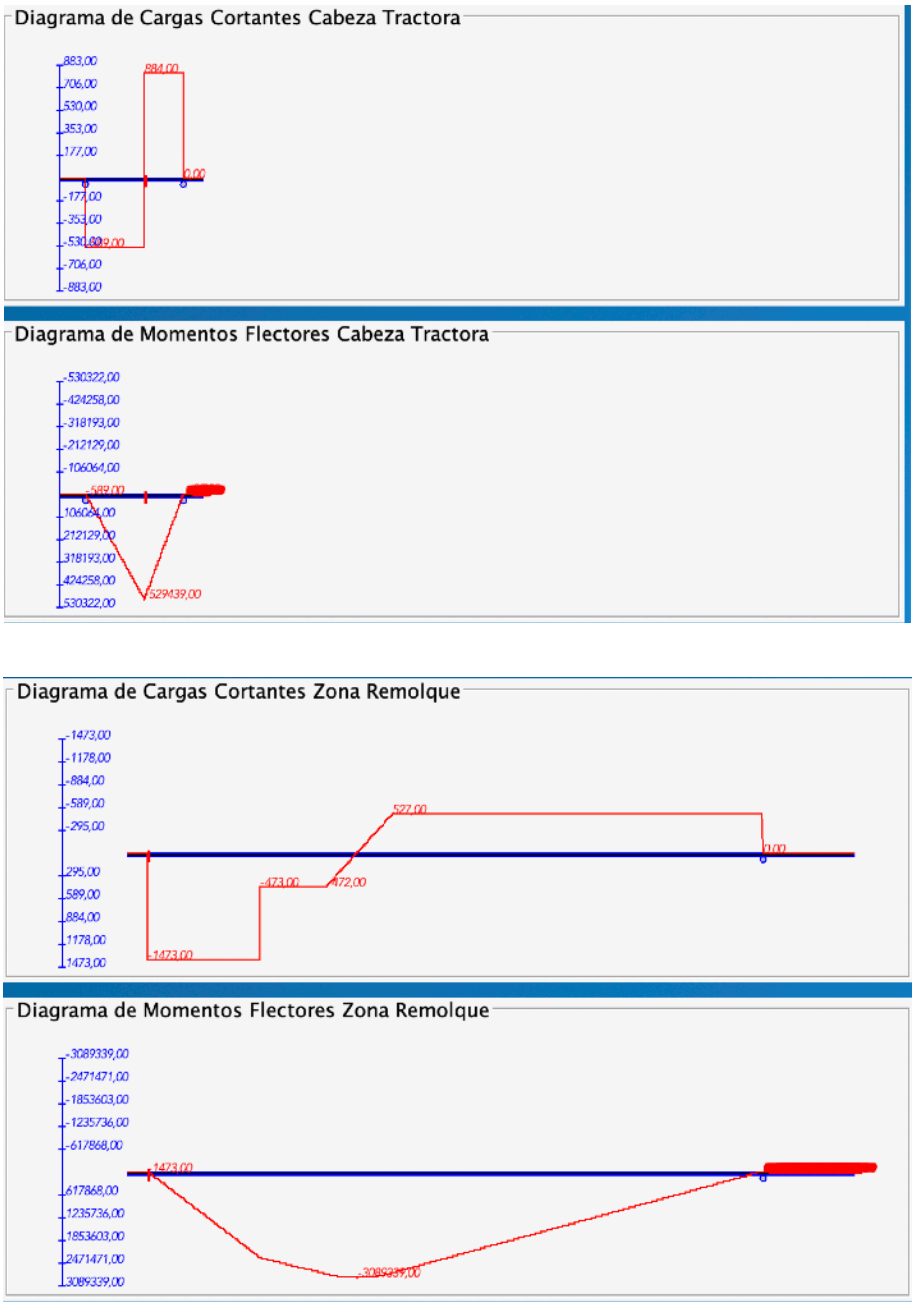


Figura B-15. Comparación “Analizar cabeza” / “Analizar remolque”.

B.4. Resultados

Una vez se ha finalizado el cálculo de la estructura, volveremos a la ventana principal indicada en el punto 1 y allí pulsaremos sobre el icono “Resultados” para generar el archivo *.pdf de resultados que cumplimentarán el anexo técnico.

Una vez se pulsa el botón “Resultados” se mostrará una ventana emergente como la de la Figura B-16 en la que se requerirá el nombre para dicho archivo *.pdf.

Figura B-16. Ventana para generar el anexo técnico

Una vez introduzcamos el nombre y pulsemos sobre el botón “Guardar”, se generará el archivo con el nombre requerido.

B.5. Ejemplos de uso

Vamos a resolver ahora con la herramienta los casos de cálculo a mano expuestos en el Capítulo 2.

CASO RÍGIDO

Se procede con el ejemplo del Capítulo 2. Para este cálculo no es necesario rellenar propiedades de la viga ya que únicamente se necesitan reacciones.

Recordemos las características de dicho ejemplo en la Figura B-17.

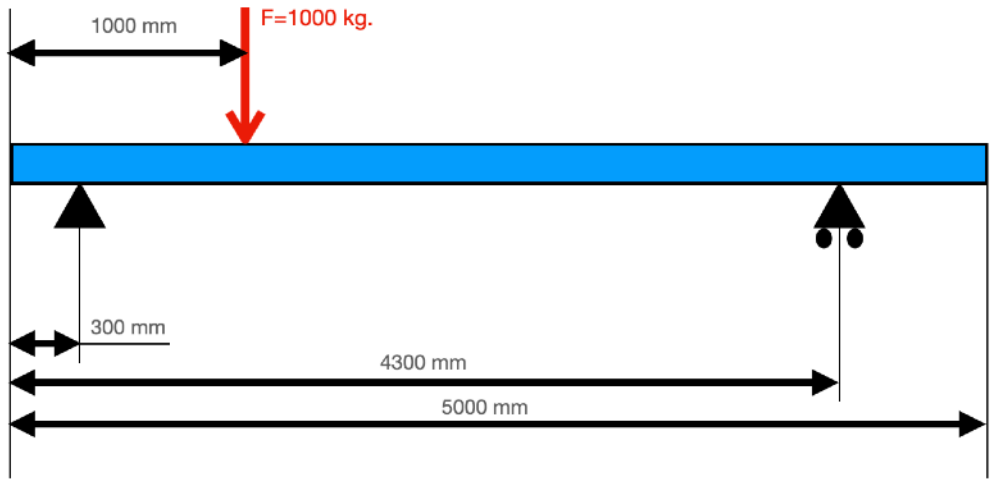


Figura B-17. Ejemplo caso rígido

Se procede a la introducción de los datos. Para ello primero generamos un nuevo proyecto y seleccionamos el tipo de camión (en este caso el numero de ejes por apoyo se va a realizar como simple. Es decir, cada tándem contiene un sólo eje. Seleccionamos pues el tipo "Rígido" y la variante "2 ejes" del mismo como se ve en la Figura B-18.

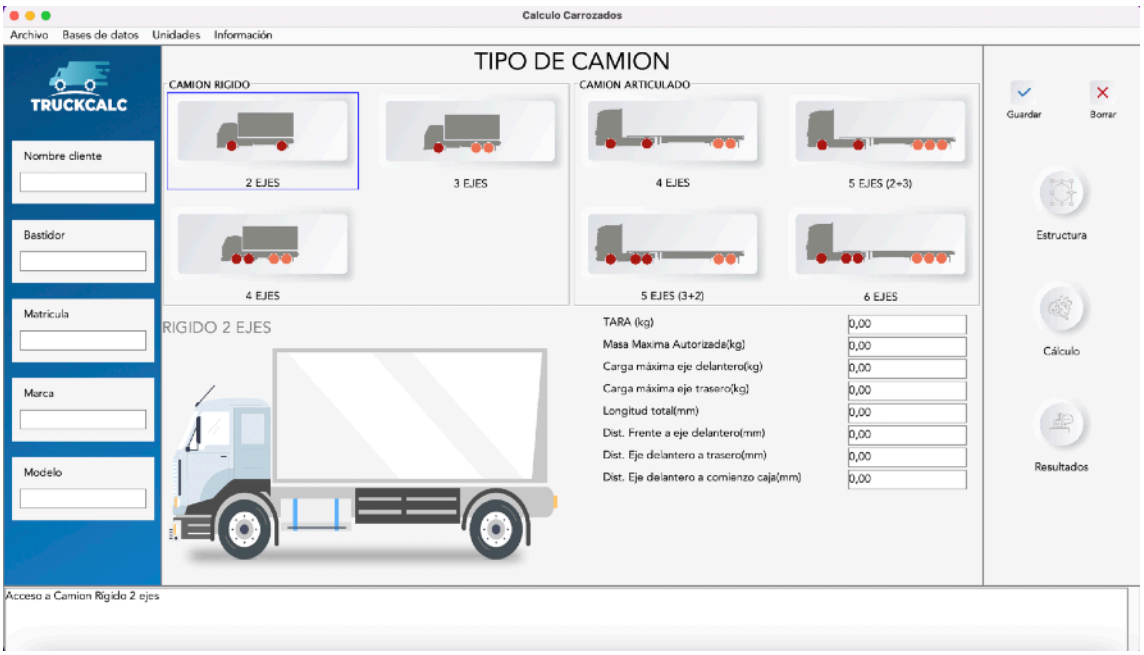


Figura B-18. Selección del tipo de camión

Rellenamos los datos del proyecto. En este caso los valores que nos interesan son los de longitud total, Dist. Frente a eje delantero y Dist. Eje delantero a trasero. Los datos introducidos se muestran en la Figura B-19.

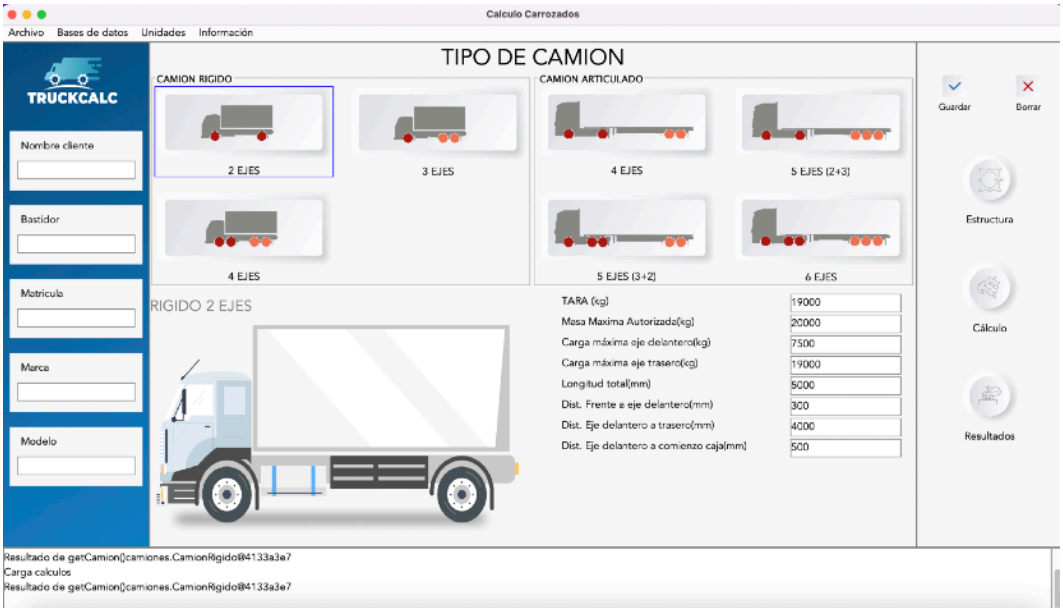


Figura B-19. Introducción de los datos

Entramos en el botón “Cálculo” e introducimos la carga de 1000 kgf como carga puntual situada a 1000 mm del frente. La situación generada será la que se muestra en la Figura B-20.

Figura B-20. Introducción de la carga

Introducida la carga, ésta aparece en el listado de cargas y el resultado ya aparece en la ventana "Reacciones y Validez Ejes".

Si deseásemos obtener el informe con estos datos, cerraríamos esta ventana y pulsaríamos sobre resultados y obteniendo el informe tras introducir el nombre con el que queremos guardarlo.

CASO ARTICULADO

Se procede con el ejemplo del Capítulo 2 de igual forma.

El caso a resolver se muestra en la Figura B-21.

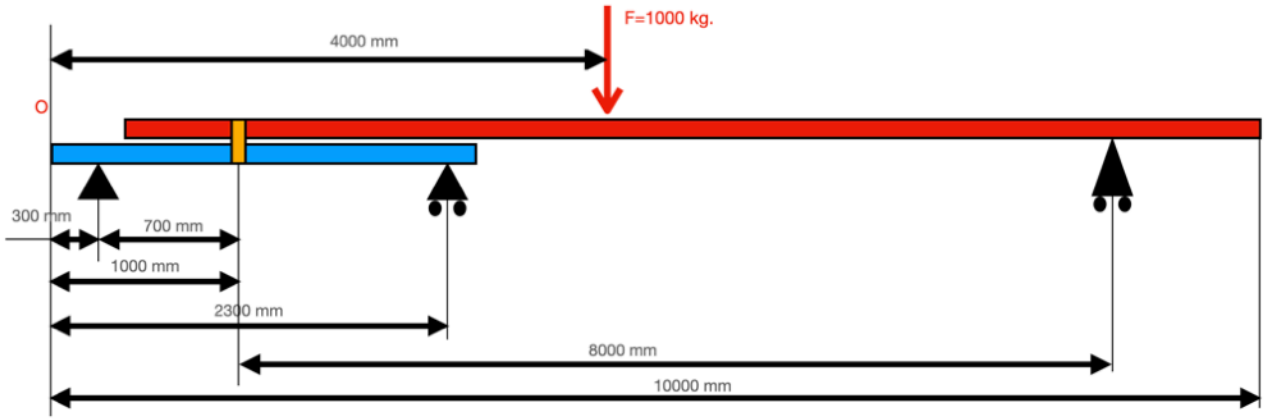


Figura B-21. Caso de ejemplo a resolver.

Se procede a la introducción de los datos. Para ello primero generamos un nuevo proyecto y seleccionamos el tipo de camión (en este caso el numero de ejes por apoyo se va a realizar como simple. Es decir, cada tándem contiene un sólo eje salvo el del remolque. Seleccionamos pues el tipo "Articulado" y la variante "4 ejes" del mismo como se ve a continuación y rellenamos los datos correspondientes. La Figura B-22 muestra los datos introducidos en la herramienta.

Al igual que antes pulsamos en "Cálculo" e introducimos una carga puntual a 4000 mm del frente con un valor de 1000 kgf.

De igual forma se obtendrá automáticamente el valor de las reacciones buscadas en el ejemplo del Capítulo 2 tal como se muestra en la Figura B-23.

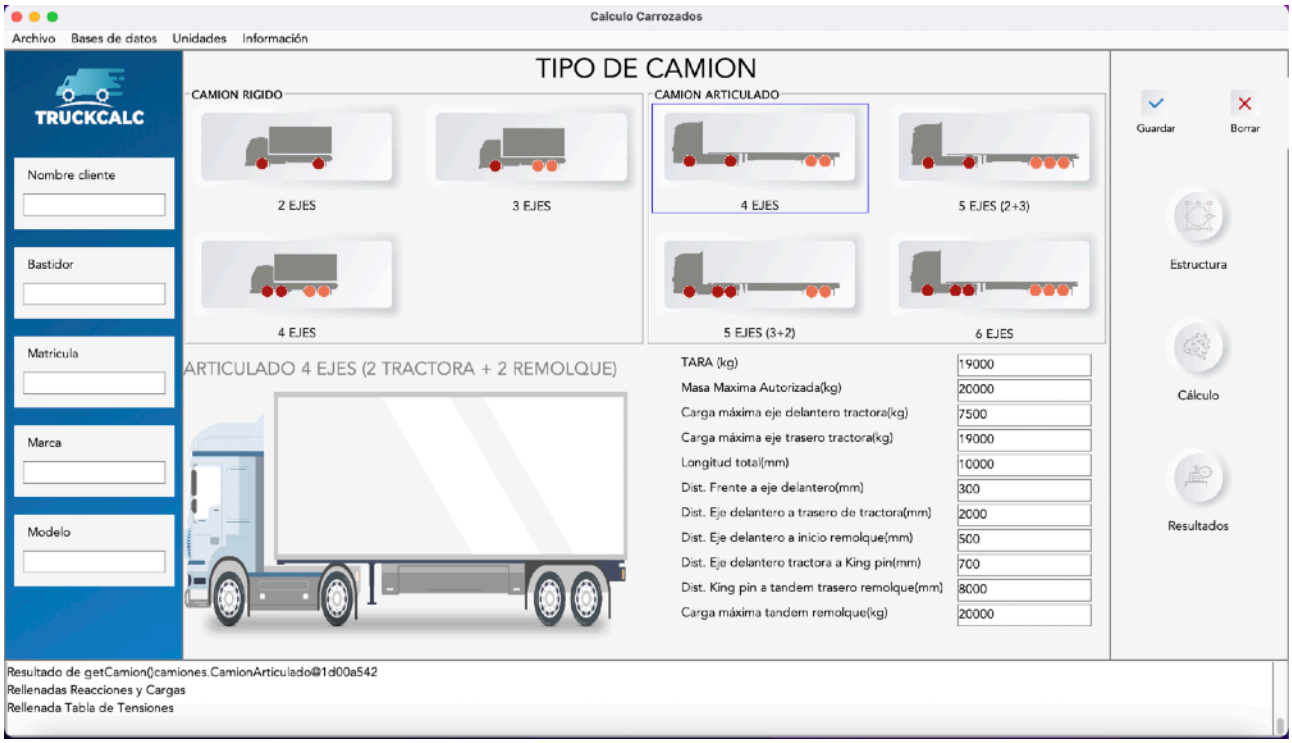


Figura B-22. Datos para el proyecto ya introducidos

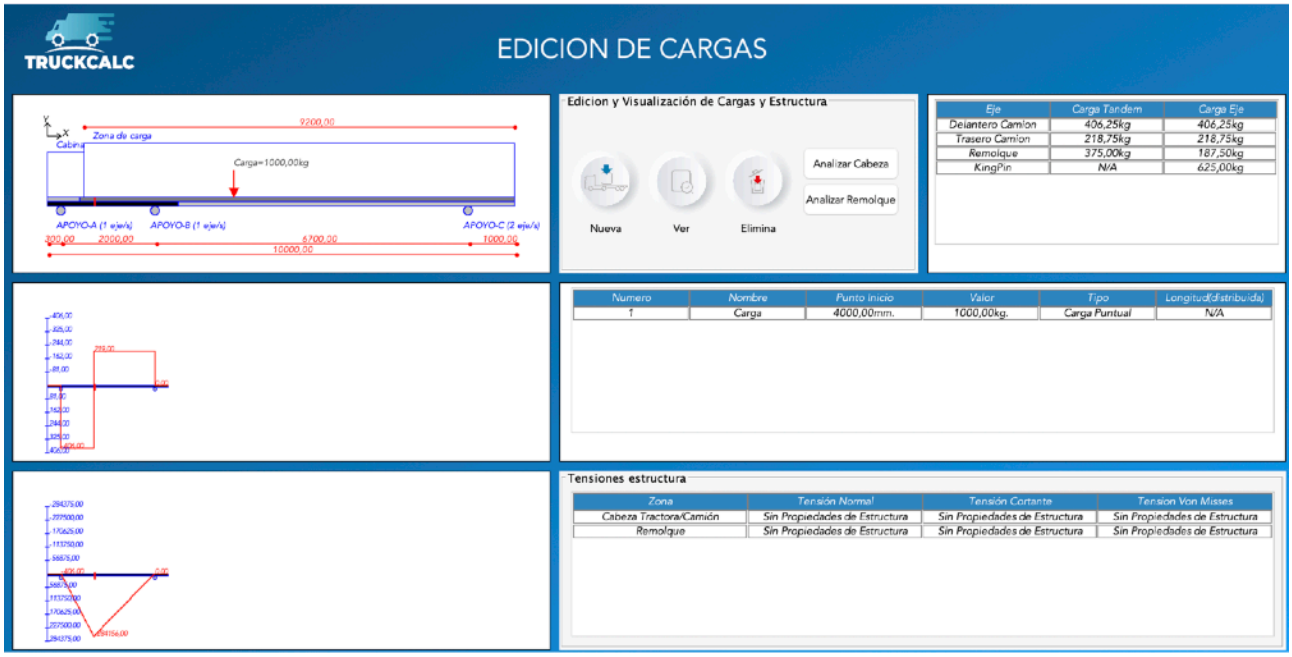


Figura B-23. Datos para el proyecto ya introducidos

Obtenemos pues los resultados y de igual forma podemos cerrar ya esta ventana y pulsar en “Resultados” para guardar nuestro informe en formato pdf.

Anexo C. Código Fuente

C.1. Paquete camiones

Se expone a continuación el código de las clases del paquete camiones que contiene la codificación relativa a los objetos camión

CLASE Camion

```
package camiones;

import java.io.Serializable;
import java.util.ArrayList;

import cargasEstructura.Carga;
import secciones.CalculoSeccion;

public class Camion implements Serializable{
	private static final long serialVersionUID = 1L;
	private String marca;
	private String modelo;
	private String cliente;
	private String matricula;
	private String bastidor;
	private TipoCamion tipo;
	private double distanciaFrenteEjeDelantero;
	private double distanciaEjeDelanteroEjeTrasero;
	private double distanciaEjeTraseroFinal;
	private double largoCamion;
	private double MMA;
	private double PMA;
	private double cargaMaxEjeDelantero;
	private double cargaMaxEjeTrasero;
	private double distanciaEjeDelanteroCaja;
	private double areaViga;
	private double inerciaViga;
	private double moduloResistenteViga;
	private double tensionCortanteViga;
	private double tensionNormalViga;
	private double tensionVonMissesViga;
	private int ejesPorTandemDelantero;
	private int ejesPorTandemTrasero;
	private CalculoSeccion seccion;
	private ArrayList<Carga> cargas;

	//Constructor sin datos
	public Camion() {
	}

	//Constructor con datos
	public Camion(String marca,String modelo,double distanciaFrenteMotor,
			double distanciaFrenteEjeDelantero,double
distanciaEjeDelanteroEjeTrasero,
			double longitud, double MMA,double PMA, double cargaMaxEjeDelantero,
double cargaMaxEjeTrasero) {
		setMarca(marca);
		setModelo(modelo);
		setFrenteEjeDelantero(distanciaFrenteEjeDelantero);
		setEjeDelanteroEjeTrasero(distanciaEjeDelanteroEjeTrasero);
		setMMA(MMA);
		setPMA(PMA);
		setCargaMaxEjeDelantero(cargaMaxEjeDelantero);
		setCargaMaxEjeTrasero(cargaMaxEjeTrasero);
		setLongitud(longitud);
	}

	//Incluimos los setters
	//Para la marca
	public void setMarca(String marca) {
		this.marca=marca;
	}
	//Para el modelo
	public void setModelo(String modelo) {
		this.modelo=modelo;
```

```
}
//Para la matricula
public void setMatricula(String matricula) {
	this.matricula=matricula;
}
//Para el cliente
public void setCliente(String cliente) {
	this.cliente=cliente;
}
//Para el bastidor
public void setBastidor(String bastidor) {
	this.bastidor=bastidor;
}
//Para la distancia del frente al eje delantero
public void setFrenteEjeDelantero(double dFD) {
	this.distanciaFrenteEjeDelantero=dFD;
}
//Para la distancia del Eje Delantero al Eje Trasero
public void setEjeDelanteroEjeTrasero(double dDT) {
	this.distanciaEjeDelanteroEjeTrasero=dDT;
}
//Para la distancia del Eje Trasero al Final
public void setEjeTraseroFinal(double dTF) {
	this.distanciaEjeTraseroFinal=dTF;
}
//Para la longitud total
public void setLongitud(double longitud) {
	this.largoCamion=longitud;
}
//Para la distnacia del Eje delantero al comienzo de la caja
public void setEjeDelanteroComienzoCaja(double dDC) {
	this.distanciaEjeDelanteroCaja=dDC;
}
//Para la Masa Maxima Autorizada
public void setMMA(double MMA) {
	this.MMA=MMA;
}
//Para el Peso Maximo Autorizado
public void setPMA(double PMA) {
	this.PMA=PMA;
}
//Para la Carga Maxima en Eje Delantero
public void setCargaMaxEjeDelantero(double cargaMaximaEjeDelantero) {
	this.cargaMaxEjeDelantero=cargaMaximaEjeDelantero;
}
//Para la Carga Maxima en Eje Trasero
public void setCargaMaxEjeTrasero(double cargaMaximaEjeTrasero) {
	this.cargaMaxEjeTrasero=cargaMaximaEjeTrasero;
}
//Para el tipo de camion
public void setTipoCamion(TipoCamion tipo) {
	this.tipo=tipo;
}
//Para el area
public void setAreaViga(double area) {
	this.areaViga=area;
}
//Para la Inercia
public void setInerciaViga(double inercia) {
	this.inerciaViga=inercia;
	this.moduloResistenteViga=this.getSeccion().getModuloResistente();
}
//Para la tension cortante
public void setTensionCortanteViga(double cortante) {
	//Dividimos por 2 porque son 2 vigas por camion
	this.tensionCortanteViga=Math.abs(cortante/(this.areaViga))/2;
}
```

```
	//Para la tension normal
	public void setTensionNormalViga(double momento) {
		if(this.getSeccion()==null){
			this.moduloResistenteViga=1;
			this.tensionNormalViga=1;
		} else {
			this.moduloResistenteViga=this.getSeccion().getModuloResistente();
			//Dividimos por 2 porque son 2 vigas por camion
			this.tensionNormalViga=Math.abs(momento/this.moduloResistenteViga)/2;
		}
	}
	//Para la tension total
	public void setTensionVonMisses() {

this.tensionVonMissesViga=Math.sqrt(Math.pow(this.tensionNormalViga,2)+3*Math.pow(this.te
nsionCortanteViga,2));
	}
	//Para los ejes por tandem delantero
	public void setEjesPorTandemDelantero(int i) {
		this.ejesPorTandemDelantero=i;
	}
	//Para los ejes por tandem trasero
	public void setEjesPorTandemTrasero(int i) {
		this.ejesPorTandemTrasero=i;
	}
	//Para la seccion
	public void setSeccion(CalculoSeccion cs) {
		this.seccion=cs;
	}
	//Para las cargas
	public void setCargas(ArrayList<Carga> alc) {
		this.cargas=alc;
	}


	//Incluimos los getters
	//Para la marca
	public String getMarca() {
		return this.marca;
	}
	//Para el modelo
	public String getModelo() {
		return this.modelo;
	}
	//Para la matricula
	public String getMatricula() {
		return this.matricula;
	}
	//Para el cliente
	public String getCliente() {
		return this.cliente;
	}
	//Para el bastidor
	public String getBastidor() {
		return this.bastidor;
	}
	//Para la distancia del frente al eje delantero
	public double getdistFrenteEjeDelantero() {
		return this.distanciaFrenteEjeDelantero;
	}
	//Para la distancia del eje Delantero al Trasero
	public double getdistEjeDelanteroEjeTrasero() {
		return this.distanciaEjeDelanteroEjeTrasero;
	}
	//Para la distancia del eje trasero al final del camion
	public double getdistTraseroFinal() {
		return this.distanciaEjeTraseroFinal;
```

```
}
//Para la longitud total
public double getLargo() {
	return this.largoCamion;
}
//Para la distancia del eje delantero al comienzo de caja
public double getDistanciaEjeDelanteroComienzoCaja() {
	return this.distanciaEjeDelanteroCaja;
}
//Para la Masa Maxima Autorizada
public double getMMA() {
	return this.MMA;
}
//Para el Peso Maximo Autorizado
public double getPMA() {
	return this.PMA;
}
//Para la carga maxima en eje delantero
public double getCargaMaxEjeDelantero() {
	return this.cargaMaxEjeDelantero;
}
//Para la carga maxima en eje trasero
public double getCargaMaxEjeTrasero() {
	return this.cargaMaxEjeTrasero;
}
//Para el tipo de camion
public TipoCamion getTipo() {
	return this.tipo;
}
//Para el area
public double getAreaViga() {
	return this.areaViga;
}
//Para la Inercia
public double getInerciaViga() {
	return this.inerciaViga;
}
//Para el modulo
public double getModuloResistenteViga() {
	return this.moduloResistenteViga;
}
//Para la tension cortante
public double getTensionCortanteViga() {
	return tensionCortanteViga;
}
//Para la tension normal
public double getTensionNormalViga() {
	return tensionNormalViga;
}
//Para la tension de Von Misses
public double getTensionVonMisses() {
	return tensionVonMissesViga;
}
//Para los ejes por tandem delantero
public int getEjesPorTandemDelantero() {
	return ejesPorTandemDelantero;
}
//Para los ejes por tandem trasero
public int getEjesPorTandemTrasero() {
	return ejesPorTandemTrasero;
}
//Para la seccion
public CalculoSeccion getSeccion() {
	return seccion;
}
//Para las cargas
public ArrayList<Carga> getCargas() {
```

```
                return cargas;
        }
}
```

CLASE CamionArticulado

```
package camiones;

import java.io.Serializable;

import secciones.CalculoSeccion;

public class CamionArticulado extends Camion implements Serializable{

	private static final long serialVersionUID = 1L;
	//Generamos las variables par
	private double distanciaKingPinEjeTraseroRemolque;
	private double distanciaEjeDelanteroKingPin;
	private double cargaMaximaEjeRemolque;
	//Las siguientes contendran el numero de ejes por tandem
	private int ejesPorTandemDelantero;
	private int ejesPorTandemTrasero;
	private int ejesPorTandemRemolque;
	private double longitudCalculoRemolque;
	private double longitudCalculoCabeza;
	private double areaVigaRemolque;
	private double inerciaVigaRemolque;
	private double moduloResistenteVigaRemolque;
	private double tensionCortanteVigaRemolque;
	private double tensionNormalVigaRemolque;
	private double tensionVonMissesVigaRemolque;
	private CalculoSeccion seccionCabeza,seccionRemolque;


	public CamionArticulado() {
		super();
		//Ajustamos el tipo a Articulado
		this.setTipoCamion(new TipoCamion('A'));
	}

	public CamionArticulado(String marca,
			String modelo,
			double distanciaFrenteMotor,
			double distanciaFrenteEjeDelantero,
			double distanciaEjeDelanteroEjeTrasero,
			double distanciaEjeTraseroFinal,
			double MMA,
			double PMA,
			double cargaMaximaEjeDelantero,
			double cargaMaximaEjeTrasero){
		super(marca,
				modelo,
				distanciaFrenteMotor,
				distanciaFrenteEjeDelantero,
				distanciaEjeDelanteroEjeTrasero,
				distanciaEjeTraseroFinal,
				MMA,
				PMA,
				cargaMaximaEjeDelantero,
				cargaMaximaEjeTrasero);
		// TODO Auto-generated constructor stub
		//Ajustamos el tipo a Articulado
		this.setTipoCamion(new TipoCamion('A'));
	}

	//GENERAMOS LOS SETTERS
	public void setDistanciaKingPinEjeRemolque(double dKR) {
		distanciaKingPinEjeTraseroRemolque=dKR;
	}
	public void setDistanciaEjeDelanteroKingPin(double dDK) {
		distanciaEjeDelanteroKingPin=dDK;
```

```
	}
	public void setLongitudCalculoRemolque(double LCR) {
		longitudCalculoRemolque=LCR;
	}
	public void setLongitudCalculoCabeza(double LCC) {
		longitudCalculoCabeza=LCC;
	}
	public void setCargaMaximaEjeRemolque(double qER) {
		cargaMaximaEjeRemolque=qER;
	}
	public void setEjesPorTandemDelantero(int ejes) {
		ejesPorTandemDelantero=ejes;
	}
	public void setEjesPorTandemTrasero(int ejes) {
		ejesPorTandemTrasero=ejes;
	}
	public void setEjesPorTandemRemolque(int ejes) {
		ejesPorTandemRemolque=ejes;
	}
	public void setAreaVigaRemolque(double area) {
		areaVigaRemolque=area;
	}
	public void setInerciaVigaRemolque(double inercia) {
		if(this.getSeccion()==null) {
			this.moduloResistenteVigaRemolque=1;
			inerciaVigaRemolque=1;
		} else {

this.moduloResistenteVigaRemolque=this.getSeccion().getModuloResistente();
			inerciaVigaRemolque=inercia;
		}
	}
	public void setTensionCortanteVigaRemolque(double Cortante) {
		//Dividimos por 2 porque son 2 vigas por camion
		tensionCortanteVigaRemolque=Cortante/(this.areaVigaRemolque)/2;
	}
	public void setTensionNormalVigaRemolque(double Momento) {
		//Dividimos por 2 porque son 2 vigas por camion
		tensionNormalVigaRemolque=Momento/this.moduloResistenteVigaRemolque/2;
	}
	public void setTensionVonMissesVigaRemolque() {

tensionVonMissesVigaRemolque=Math.sqrt(Math.pow(tensionNormalVigaRemolque,2)+3*Math.pow(t
ensionCortanteVigaRemolque, 2));
	}
	public void setSeccionCabeza(CalculoSeccion cs) {
		seccionCabeza=cs;
	}
	public void setSeccionRemolque(CalculoSeccion cs) {
		seccionRemolque=cs;
	}

	//GENERAMOS LOS GETTERS
	public double getDistanciaKingPinEjeRemolque() {
		return distanciaKingPinEjeTraseroRemolque;
	}
	public double getDistanciaEjeDelanteroKingPin() {
		return distanciaEjeDelanteroKingPin;
	}
	public double getCargaMaximaEjeRemolque() {
		return cargaMaximaEjeRemolque;
	}
	public double getLongitudCalculoRemolque() {
		return longitudCalculoRemolque;
	}
	public double getLongitudCalculoCabeza() {
		return longitudCalculoCabeza;
```

```
	}
	public int getEjesPorTandemDelantero() {
		return ejesPorTandemDelantero;
	}
	public int getEjesPorTandemTrasero() {
		return ejesPorTandemTrasero;
	}
	public int getEjesPorTandemRemolque() {
		return ejesPorTandemRemolque;
	}
	public double getAreaVigaRemolque() {
		return areaVigaRemolque;
	}
	public double getInerciaVigaRemolque() {
		return inerciaVigaRemolque;
	}
	public double getmoduloResistenteVigaRemolque() {
		return moduloResistenteVigaRemolque;
	}
	public double getTensionCortanteVigaRemolque() {
		return Math.abs(tensionCortanteVigaRemolque);
	}
	public double getTensionNormalVigaRemolque() {
		return Math.abs(tensionNormalVigaRemolque);
	}
	public double getTensionVonMissesVigaRemolque() {
		return Math.abs(tensionVonMissesVigaRemolque);
	}
	public CalculoSeccion getSeccionCabeza() {
		return seccionCabeza;
	}
	public CalculoSeccion getSeccionRemolque() {
		return seccionRemolque;
	}
}
```

CLASE CamionRigido

```
package camiones;

import java.io.Serializable;

public class CamionRigido extends Camion implements Serializable{

	private static final long serialVersionUID = 1L;
	public CamionRigido() {
		super();
		//Ajustamos el tipo a Rigido
		this.setTipoCamion(new TipoCamion('R'));
	}
	public CamionRigido(String marca,
			String modelo,
			double distanciaFrenteMotor,
			double distanciaFrenteEjeDelantero,
			double distanciaEjeDelanteroEjeTrasero,
			double distanciaEjeTraseroFinal,
			double MMA,
			double PMA,
			double cargaMaximaEjeDelantero,
			double cargaMaximaEjeTrasero) {
		super(marca,
				modelo,
				distanciaFrenteMotor,
				distanciaFrenteEjeDelantero,
				distanciaEjeDelanteroEjeTrasero,
				distanciaEjeTraseroFinal,
				MMA,
				PMA,
				cargaMaximaEjeDelantero,
				cargaMaximaEjeTrasero);
		// TODO Auto-generated constructor stub
		//Ajustamos el tipo a Rigido
		this.setTipoCamion(new TipoCamion('R'));
	}

}
```

CLASE Proyecto

```
package camiones;

import java.io.Serializable;
import java.util.ArrayList;
import java.util.Iterator;

import cargasEstructura.Carga;
import cargasEstructura.CargaDistribuida;
import cargasEstructura.CargaPuntual;
import cargasEstructura.TipoCarga;

public class Proyecto implements Serializable{
private static final long serialVersionUID = 1L;
private static ArrayList<Carga> cargas;
private static Camion camion;

	public Proyecto() {
	//Inicializo las cargas
		cargas=new ArrayList<Carga>();
		//Inicializo el camion
		camion = new Camion();
	}

	//Para las cargas
	public void setCargas(ArrayList<Carga> q) {
		cargas=q;
	}

	//Generamos los GETTERS
	//Para el camion
	public Camion getCamion() {
		return camion;
	}

	//Para las cargas
	public ArrayList<Carga> getCargas(){
		return cargas;
	}

	//Generamos un metodo para añadir una carga puntual
	public void addPuntual(String nombre,double puntoInicial,double valor,boolean
visible) {
		cargas.add(new CargaPuntual(nombre,puntoInicial,valor,visible));
	}

	//Generamos un metodo para añadir una carga distribuida
	public void addDistribuida(String nombre,double puntoInicial,double valor,boolean
visible,double longitud) {
		cargas.add(new
CargaDistribuida(nombre,puntoInicial,valor,visible,longitud));
	}

	//Este metodo sirve como comprobacion por consola de las cargas que he metido
	public void imprimeCargas() {
		ArrayList<Carga> al = getCargas();
		Iterator<Carga> it = al.iterator();
		Carga q;
		while(it.hasNext()) {
			q=it.next();
			TipoCarga t=q.getTipo();
			System.out.println("---Nombre:"+q.getNombre()+"\n---
Valor:"+q.getValor()+"\n---Punto Inicio:"+q.getPuntoInicio()+"\n---
Tipo:"+t.getTipoCarga());
		}
```

```
	}

	//carga los datos del camion
	public void cargaDatos(double mma, double pma, double cargadelantero,double
cargatrasero, double longitud, double distfrentedelantero
			,double distdelanterotrasero,double distdelanteroiniciocaja) {
		camion.setMMA(mma);
		camion.setPMA(pma);
		camion.setCargaMaxEjeDelantero(cargadelantero);
		camion.setCargaMaxEjeTrasero(cargatrasero);
		camion.setLongitud(longitud);
		camion.setEjeDelanteroEjeTrasero(distdelanterotrasero);
		camion.setFrenteEjeDelantero(distfrentedelantero);
		camion.setEjeDelanteroComienzoCaja(distdelanteroiniciocaja);
	}

	//crea el camion
	public void setCamion(Camion c) {
		camion=c;
	}


}
```

CLASE TipoCamion

```
package camiones;

import java.io.Serializable;

public class TipoCamion implements Serializable{
       private static final long serialVersionUID = 1L;
       private char tipo;
       private String descripcion;

    public TipoCamion(char tipo) {
              setTipo(tipo);
       }

       //Este método asigna el tipo de camion
    private void setTipo(char tipo) {
              if(tipo=='R') {
                     this.tipo='R';
                     this.descripcion="Camión Rigido";
              }
              else if(tipo=='A') {
                     this.tipo='A';
                     this.descripcion="Camión Articulado";
              } else {
                     this.tipo='N';
                     this.descripcion="Camión No Valido";
              }
    }

    //Devuelve la descripcion
    public String getDescripcion() {
       return descripcion;
    }

    //Devuelve el tipo
    public char getTipo() {
       return tipo;
    }
}
```

CLASE TruckCalc

```
package camiones;

import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.Toolkit;
import java.util.concurrent.TimeUnit;

import javax.swing.ImageIcon;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.WindowConstants;

import auxiliares.CheckDatos;
import auxiliares.DatosGenericos;
import auxiliares.PanelConImagen;

public class TruckCalc extends JFrame{
       private static Proyecto proyecto;
       private static int anchoVentana=900;
       private static int altoVentana=600;
       public TruckCalc() {
       }
    //Metodo principal para arrancar
    public static void main(String args[]) {
       TruckCalc tc = new TruckCalc();
       try{
                  proyecto=new Proyecto();
                  PanelConImagen ve = new PanelConImagen();
                  ve.setImagen(DatosGenericos.getDireccion("Imagenes/pantallaprincipal/
portada.png"));

                  tc.add(ve);
                  //Quitamos los botones de cierre, maximizar y minimizar
                  tc.setUndecorated(true);
                  //Le damos transparencia
                  tc.setOpacity((float)0.8);
                  //Lo hacemos visible
                  tc.setVisible(true);
                  //Lo centramos en la ventana
                  //Cogemos las dimensiones de pantalla
                  DatosGenericos.setPosicion(tc,anchoVentana,altoVentana);
                  tc.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);

                  try {
                     //Paramos 3 segundos para mostrar la ventana
                     TimeUnit.SECONDS.sleep(2);
                  } catch(Exception e) {

                  }
                  tc.dispose();
                  //Generamos una vista general
                     VistaGeneral carga = new VistaGeneral(proyecto);
       } catch (Exception e) {
              CheckDatos.okNokMensaje("Se ha producido un error: "+e.toString());
       }
    }
}
```

CLASE VistaGeneral

```
package camiones;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.image.BufferedImage;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Iterator;
import java.awt.*;
import javax.swing.*;
import javax.swing.text.JTextComponent;

import auxiliares.CheckDatos;
import auxiliares.CreaPDF;
import auxiliares.DatosGenericos;
import auxiliares.FicheroUsado;
import auxiliares.Formato;
import auxiliares.PanelConImagen;
import auxiliares.Unidades;
import auxiliares.VentanaIntroduceNombrePDF;
import auxiliares.VentanaProceso;
import basesDatos.VentanaBaseDatosCamion;
import calculosEstructura.CalculoReaccionesArticulado;
import calculosEstructura.CalculoReaccionesRigido;
import calculosEstructura.LeyesCortantes;
import calculosEstructura.LeyesMomentos;
import calculosEstructura.PanelCortantes;
import calculosEstructura.PanelEsquema;
import calculosEstructura.PanelMomentos;
import calculosEstructura.VentanaCalculos;
import secciones.CalculoSeccion;
import secciones.VentanaSeccion;

public class VistaGeneral extends JFrame{
              private static final long serialVersionUID = -6979699003309689192L;
              public String areaTexto="";
              private JTextArea area;
              //Definimos los valores comunes para todos los componentes y elementos
        private ArrayList<JButton> camionseleccion = new ArrayList<JButton>(); //Usado
para la usabilidad
        private ArrayList<JTextComponent> grupoInfo = new
ArrayList<JTextComponent>();  //Usado para eliminar la información
        private static VentanaCalculos ventCalc;
        private static Proyecto proyecto;
        private JTextField nombretext,bastidortext ,matriculatext, marcatext, modelotext,
mmatext, pmatext,cargaejedelanterotext,cargaejetraserotext,

cargaejeremolquetext,largotext,frenteprimerejetext,distdelanterotraserotext,comienzocajat
ext,distprimerejekingpintext,
        distkingpintandemremolquetext;
        private JButton calculo,r2ejes,r3ejes,r4ejes,a4ejes, a5ejes23,a5ejes32,a6ejes;
        private JLabel
mma,pma,cargaejedelantero,cargaejetrasero,cargaejeremolque,largototal,distfrenteprimereje
,entreejes,comienzocaja,distprimerejekingpin,distkingpintandemremolque;
        private static VentanaSeccion ventSec;
       private static VentanaBaseDatosCamion panelBaseDatos;
       private FicheroUsado fichero;
       private static VentanaIntroduceNombrePDF vinp;

              public VistaGeneral(Proyecto proyecto){

                     //Asignamos las unidades que queremos mostrar junto a los datos y que
podremos cambiar al modificar las unidades
```

```
                Unidades.Kg();
                Unidades.mm();

                //Inicializamos el fichero
            fichero=new FicheroUsado();

             //Generamos un ArrayList de componentes para darles formato
             ArrayList<Component> al = new ArrayList<Component>();

             //Trabajamos con el proyecto
             asignaproyecto(proyecto);

            //Generamos los BOTONES y los añadimos al ArrayList
           JButton viga = new JButton("Estructura");
           calculo = new JButton("Cálculo");
           JButton resultados = new JButton("Resultados");
           r2ejes= new JButton("2 EJES");
           r3ejes= new JButton("3 EJES");
           r4ejes= new JButton("4 EJES");
           a4ejes= new JButton("4 EJES");
           a5ejes23= new JButton("5 EJES (2+3)");
           a5ejes32= new JButton("5 EJES (3+2)");
           a6ejes= new JButton("6 EJES");
           JButton graba=new JButton("Guardar");
           JButton borra=new JButton("Borrar");
           //Preparamos las imagenes de los botones secundarios. Seleccionamos los
iconos y los asignamos al boton
            ImageIcon iconobtnviga = escalaIcono(new
ImageIcon(getClass().getResource(DatosGenericos.getDireccion("Imagenes/pantallaprincipal/
estructura.png"))),1);
            ImageIcon iconobtncalculo = escalaIcono(new
ImageIcon(getClass().getResource(DatosGenericos.getDireccion("Imagenes/pantallaprincipal/
calculo.png"))),1);
            ImageIcon iconobtnresultados = escalaIcono(new
ImageIcon(getClass().getResource(DatosGenericos.getDireccion("Imagenes/pantallaprincipal/
resultados.png"))),1);
            ImageIcon iconobtnr2ejes = escalaIcono(new
ImageIcon(getClass().getResource(DatosGenericos.getDireccion("Imagenes/pantallaprincipal/
r2ejes.png"))),3);
            ImageIcon iconobtnr3ejes = escalaIcono(new
ImageIcon(getClass().getResource(DatosGenericos.getDireccion("Imagenes/pantallaprincipal/
r3ejes.png"))),3);
            ImageIcon iconobtnr4ejes = escalaIcono(new
ImageIcon(getClass().getResource(DatosGenericos.getDireccion("Imagenes/pantallaprincipal/
r4ejes.png"))),3);
            ImageIcon iconobtna4ejes = escalaIcono(new
ImageIcon(getClass().getResource(DatosGenericos.getDireccion("Imagenes/pantallaprincipal/
a4ejes-2-2.png"))),3);
            ImageIcon iconobtna5ejes23 = escalaIcono(new
ImageIcon(getClass().getResource(DatosGenericos.getDireccion("Imagenes/pantallaprincipal/
a5ejes-2-3.png"))),3);
            ImageIcon iconobtna5ejes32 = escalaIcono(new
ImageIcon(getClass().getResource(DatosGenericos.getDireccion("Imagenes/pantallaprincipal/
a5ejes-3-2.png"))),3);
            ImageIcon iconobtna6ejes = escalaIcono(new
ImageIcon(getClass().getResource(DatosGenericos.getDireccion("Imagenes/pantallaprincipal/
a6ejes-3-3.png"))),3);
            ImageIcon iconobtnguardar = escalaIcono(new
ImageIcon(getClass().getResource(DatosGenericos.getDireccion("Imagenes/pantallaprincipal/
guardar.png"))),1);
            ImageIcon iconobtnborrar = escalaIcono(new
ImageIcon(getClass().getResource(DatosGenericos.getDireccion("Imagenes/pantallaprincipal/
borrar.png"))),1);
            //Asignamos los iconos a los botones
            asignaIconoBoton(viga,iconobtnviga);
            asignaIconoBoton(calculo,iconobtncalculo);
            asignaIconoBoton(resultados,iconobtnresultados);
```

```
             asignaIconoBoton(r2ejes,iconobtnr2ejes);
             asignaIconoBoton(r3ejes,iconobtnr3ejes);
             asignaIconoBoton(r4ejes,iconobtnr4ejes);
             asignaIconoBoton(a4ejes,iconobtna4ejes);
             asignaIconoBoton(a5ejes23,iconobtna5ejes23);
             asignaIconoBoton(a5ejes32,iconobtna5ejes32);
             asignaIconoBoton(a6ejes,iconobtna6ejes);
             asignaIconoBoton(graba,ajustaEscala(iconobtnguardar,32));
             asignaIconoBoton(borra,ajustaEscala(iconobtnborrar,32));
             //Metemos los camiones para su control de usabilidad
             camionseleccion.add(r2ejes);
             camionseleccion.add(r3ejes);
             camionseleccion.add(r4ejes);
             camionseleccion.add(a4ejes);
             camionseleccion.add(a5ejes23);
             camionseleccion.add(a5ejes32);
             camionseleccion.add(a6ejes);


            //Generamos los CAMPOS Y LABELS para recoger los datos del proyecto
            JLabel nombre = new JLabel("Nombre cliente");
            nombretext = new JTextField();
            JLabel bastidor = new JLabel("Bastidor");
            bastidortext = new JTextField();
            JLabel matricula = new JLabel("Matrícula");
            matriculatext=new JTextField();
            JLabel marca = new JLabel("Marca");
            marcatext=new JTextField();
            JLabel modelo = new JLabel("Modelo");
            modelotext=new JTextField();


            mma = new JLabel("TARA("+Unidades.getUnidadesCarga()+")");
            mmatext=new JTextField();
            pma = new JLabel("Masa Maxima Autorizada("+Unidades.getUnidadesCarga()+")");
            pmatext=new JTextField();
            cargaejedelantero = new JLabel("Carga máxima tándem
delantero("+Unidades.getUnidadesCarga()+")");
            cargaejedelanterotext=new JTextField();
            cargaejetrasero = new JLabel("Carga máxima tándem
trasero("+Unidades.getUnidadesCarga()+")");
            cargaejetraserotext=new JTextField();
            cargaejeremolque = new JLabel("Carga máxima tándem
remolque("+Unidades.getUnidadesCarga()+")");
            cargaejeremolquetext=new JTextField();
            largototal = new JLabel("Longitud total("+Unidades.getUnidadesLongitud()
+")");
            largotext = new JTextField();
            distfrenteprimereje = new JLabel("Dist. Frente a tándem
delantero("+Unidades.getUnidadesLongitud()+")");
            frenteprimerejetext = new JTextField();
            entreejes = new JLabel("Dist. Tándem delantero a tándem
trasero("+Unidades.getUnidadesLongitud()+")");
            distdelanterotraserotext = new JTextField();
            comienzocaja = new JLabel("Dist. Tándem delantero a comienzo
caja("+Unidades.getUnidadesLongitud()+")");
            comienzocajatext = new JTextField();
            distprimerejekingpin = new JLabel("Dist. Eje delantero tractora a King
pin("+Unidades.getUnidadesLongitud()+")");
            distprimerejekingpintext = new JTextField();
            distkingpintandemremolque = new JLabel("Dist. King pin a tándem trasero
remolque("+Unidades.getUnidadesLongitud()+")");
            distkingpintandemremolquetext = new JTextField();
            JLabel tituloapp = new JLabel("TIPO DE CAMIÓN");
            JLabel tipocamion = new JLabel("");
            //Damos formato a los tipos de aplicacion
            Formato.setFormatoAplicacion(tituloapp);
```

```
            Formato.setFormatoTipoCamion(tipocamion);


            //Rellenamos los Jtext de forma inicial
            rellenaDatos();


            //Generamos un AREA PARA MENSAJES del programa
            area = new JTextArea(4,1);
            area.setEnabled(true);
            //Le damos formato
            Formato.setBordeEstandar(area);
            area.setText(areaTexto); //Le damos al area de mensajes algo más de espacio
para que no sea solo una linea.
            area.setEditable(false);
            //Generamos un scroll para el area y lo incluimos en el area de texto
            JScrollPane areaScroll = new JScrollPane(area,

JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,JScrollPane.HORIZONTAL_SCROLLBAR_NEVER);


            //Generamos los JMENU
             /* Creamos el JMenuBar y lo asociamos con el JFrame */
             JMenuBar menuBar=new JMenuBar();
             setJMenuBar(menuBar);

             /* Creamos los JMenu y lo pasamos como parámetro al JMenuBar mediante el
método add */
             JMenu menuFichero=new JMenu("Archivo");
                    JMenuItem menuGuardaFichero=new JMenuItem("Guardar");
                    JMenuItem menuCargaFichero=new JMenuItem("Abrir");
             JMenu menuOpciones=new JMenu("Información");
             //AcercaDe será un submenu del menu
                    JMenuItem menuAcerca=new JMenuItem("Acerca de TruckCalc");
             JMenu menuBBDD=new JMenu("Bases de datos");
             //Importa y guarda BBDD serán submenus de BBDD
                    JMenuItem importaBBDD=new JMenuItem("Importar camión de base de
datos");
                    JMenuItem guardaBBDD=new JMenuItem("Guarda camión en base de datos");
             JMenu menuUnidades=new JMenu("Unidades");
             //Longitud y Carga serán submenus de Unidades
                    JMenu menuLongitud=new JMenu("Unidades Longitud");
                    //M, dm, cm y mm serán submenus de Longitud
                           JMenuItem menuMm=new JMenuItem("mm");
                           JMenuItem menuCm=new JMenuItem("cm");
                           JMenuItem menuDm=new JMenuItem("dm");
                           JMenuItem menuM=new JMenuItem("m");
                    JMenu menuCarga=new JMenu("Unidades Carga");
                    //Kg y NW serán submenus de Carga
                           JMenuItem menuKg=new JMenuItem("kg");
                           JMenuItem menuNw=new JMenuItem("N");

                //Metemos cada submenu en el correspondiente
             menuBar.add(menuFichero);
             menuBar.add(menuBBDD);
             menuBar.add(menuUnidades);
             menuBar.add(menuOpciones);
             menuFichero.add(menuCargaFichero);
             menuFichero.add(menuGuardaFichero);
             menuOpciones.add(menuAcerca);
             menuBBDD.add(importaBBDD);
             menuBBDD.add(guardaBBDD);
             menuUnidades.add(menuLongitud);
             menuUnidades.add(menuCarga);
             menuLongitud.add(menuMm);
             menuLongitud.add(menuCm);
             menuLongitud.add(menuDm);
```

```
 menuLongitud.add(menuM);
 menuCarga.add(menuKg);
 menuCarga.add(menuNw);
 //Damos formato a los menus
 Formato.setFuenteMenu(menuFichero);
 Formato.setFuenteMenu(menuOpciones);
 Formato.setFuenteMenu(menuBBDD);
 Formato.setFuenteMenu(menuUnidades);
 Formato.setFuenteMenu(menuLongitud);
 Formato.setFuenteMenu(menuCarga);
 Formato.setFuenteMenuItem(menuGuardaFichero);
 Formato.setFuenteMenuItem(menuCargaFichero);
 Formato.setFuenteMenuItem(menuAcerca);
 Formato.setFuenteMenuItem(importaBBDD);
 Formato.setFuenteMenuItem(guardaBBDD);
 Formato.setFuenteMenuItem(menuUnidades);
 Formato.setFuenteMenuItem(menuMm);
 Formato.setFuenteMenuItem(menuCm);
 Formato.setFuenteMenuItem(menuDm);
 Formato.setFuenteMenuItem(menuM);
 Formato.setFuenteMenuItem(menuKg);
 Formato.setFuenteMenuItem(menuNw);


//ADJUNTAMOS los elementos al ArrayList para darles formato y se lo damos
al.add(viga);
al.add(calculo);
al.add(resultados);
al.add(r2ejes);
al.add(r3ejes);
al.add(r4ejes);
al.add(a4ejes);
al.add(a5ejes23);
al.add(a5ejes32);
al.add(a6ejes);
al.add(graba);
al.add(borra);
al.add(nombre);
al.add(bastidor);
al.add(nombretext);
al.add(bastidortext);
al.add(matricula);
al.add(matriculatext);
al.add(marca);
al.add(marcatext);
al.add(modelo);
al.add(modelotext);
al.add(largototal);
al.add(largotext);
al.add(distfrenteprimereje);
al.add(frenteprimerejetext);
al.add(entreejes);
al.add(distdelanterotraserotext);
al.add(comienzocaja);
al.add(comienzocajatext);
al.add(distprimerejekingpin);
al.add(distprimerejekingpintext);
al.add(distkingpintandemremolque);
al.add(distkingpintandemremolquetext);
al.add(mma);
al.add(mmatext);
al.add(pma);
al.add(pmatext);
al.add(cargaejedelantero);
al.add(cargaejedelanterotext);
al.add(cargaejetrasero);
al.add(cargaejetraserotext);
```

```
            al.add(cargaejeremolque);
            al.add(distkingpintandemremolquetext);
            al.add(area);
            //FORMATEAMOS EL ESTILO de los elementos componentes.
            darTipoLetra(al);


            //Generamos el grupo de información a comprobar
            grupoInfo.add(area);
            grupoInfo.add(nombretext);
            grupoInfo.add(matriculatext);
            grupoInfo.add(bastidortext);
            grupoInfo.add(largotext);
            grupoInfo.add(marcatext);
            grupoInfo.add(modelotext);
            grupoInfo.add(frenteprimerejetext);
            grupoInfo.add(distdelanterotraserotext);
            grupoInfo.add(comienzocajatext)    ;
            grupoInfo.add(distprimerejekingpintext)   ;
            grupoInfo.add(distkingpintandemremolquetext)    ;


            //Generamos los distintos PANELES DE CONTENIDO
            PanelConImagen panelInfo = new
PanelConImagen(DatosGenericos.getDireccion("Imagenes/fondos/fondo2.jpg"));//Panel de los
botones
            PanelConImagen imagenCentral = new
PanelConImagen(DatosGenericos.getDireccion("Imagenes/fondos/fondo-gris.png")); //Imagen
del producto
            PanelConImagen botonesSecundario = new
PanelConImagen(DatosGenericos.getDireccion("Imagenes/fondos/fondo-gris.png")); //
Funciones secundarias
            PanelConImagen mensajesCamion = new PanelConImagen(); //Area de mensajes
            PanelConImagen camionesCompleto = new
PanelConImagen(DatosGenericos.getDireccion("Imagenes/fondos/fondo-gris.png")); //Este
panel contendra el título y la imagen del camion
            PanelConImagen camiones = new
PanelConImagen(DatosGenericos.getDireccion("Imagenes/fondos/fondo-gris.png")); //Este
panel contiene la información del camion
            PanelConImagen camionesimagen = new
PanelConImagen(DatosGenericos.getDireccion("Imagenes/fondos/fondo-gris.png")); //Este
panel carga la imagen del camion
            PanelConImagen datosCamion= new
PanelConImagen(DatosGenericos.getDireccion("Imagenes/fondos/fondo-gris.png"));//Panel de
los datos del camion


            //definimos los subpaneles CONTENEDORES DE DESCRIPCION Y RECOGIDA DE TEXTO
            PanelConImagen fondoTexto1=new
PanelConImagen(DatosGenericos.getDireccion("Imagenes/fondos/fondo-gris.png"));//contiene
el texto "nombre" y el campo para recogerlo
            PanelConImagen fondoTexto2=new
PanelConImagen(DatosGenericos.getDireccion("Imagenes/fondos/fondo-gris.png"));//contiene
el texto "bastidor" y el campo para recogerlo
            PanelConImagen fondoTexto3=new
PanelConImagen(DatosGenericos.getDireccion("Imagenes/fondos/fondo-gris.png"));//contiene
el texto "matricula" y el campo para recogerlo
            PanelConImagen fondoTexto4=new
PanelConImagen(DatosGenericos.getDireccion("Imagenes/fondos/fondo-gris.png"));//contiene
el texto "matricula" y el campo para recogerlo
            PanelConImagen fondoTexto5=new
PanelConImagen(DatosGenericos.getDireccion("Imagenes/fondos/fondo-gris.png"));//contiene
el texto "matricula" y el campo para recogerlo
            PanelConImagen panelMarca=new
PanelConImagen(DatosGenericos.getDireccion("Imagenes/pantallaprincipal/recurso.png")); //
contiene el logo de la aplicación
            panelMarca.setPreferredSize(new Dimension(150,80));
```

```
            //Con el metodo preparaLabelTexto generamos el conjunto de componentes del
subpanel
            preparaLabelTexto(fondoTexto1,180,80,nombre,nombretext);
            preparaLabelTexto(fondoTexto2,180,80,bastidor,bastidortext);
            preparaLabelTexto(fondoTexto3,180,80,matricula,matriculatext);
            preparaLabelTexto(fondoTexto4,180,80,marca,marcatext);
            preparaLabelTexto(fondoTexto5,180,80,modelo,modelotext);


            //Definimos los subpaneles CONTENEDORES DE LAS IMAGENES MINIATURA DE LOS
TIPOS DE CAMION
            //Este panel contendra el titulo
            PanelConImagen panelTitulo = new
PanelConImagen(DatosGenericos.getDireccion("Imagenes/fondos/fondo-gris.png"));
            //Damos el layout al titulo y lo completamos con el titulo
            panelTitulo.setLayout(new GridLayout(1,0,0,0));
            panelTitulo.add(tituloapp);
            panelTitulo.setAlignmentX(SwingConstants.CENTER);
            //el contenedorRigidosyArticulados contendra el panel de rigidos a la
izquierda y de articulado a la derecha
            PanelConImagen contenedorRigidosyArticulados = new
PanelConImagen(DatosGenericos.getDireccion("Imagenes/fondos/fondo-gris.png"));
            PanelConImagen panelrigidos = new
PanelConImagen(DatosGenericos.getDireccion("Imagenes/fondos/fondo-gris.png"));
            PanelConImagen panelarticulados = new
PanelConImagen(DatosGenericos.getDireccion("Imagenes/fondos/fondo-gris.png"));
            //Damos el layout al contenedor de rigidos y articulados y lo completamos
            contenedorRigidosyArticulados.setLayout(new GridLayout(1,2,0,0));
            panelrigidos.setLayout(new GridLayout(2,2,20,20));
            //Damos formato al borde
            Formato.setBorder(panelrigidos, "CAMIÓN RÍGIDO");
            panelrigidos.add(r2ejes);
            panelrigidos.add(r3ejes);
            panelrigidos.add(r4ejes);
            panelarticulados.setLayout(new GridLayout(2,2,20,20));
            //Damos formateo al borde
            Formato.setBorder(panelarticulados,"CAMIÓN ARTICULADO");
            panelarticulados.add(a4ejes);
            panelarticulados.add(a5ejes23);
            panelarticulados.add(a5ejes32);
            panelarticulados.add(a6ejes);
            contenedorRigidosyArticulados.add(panelrigidos);
            contenedorRigidosyArticulados.add(panelarticulados);
            contenedorRigidosyArticulados.setPreferredSize(new Dimension(1000,300));;
            contenedorRigidosyArticulados.setMaximumSize(getMaximumSize());

            //Preparamos el panel camiones completo
            camionesCompleto.setLayout(new BorderLayout());
            camionesCompleto.setPreferredSize(new Dimension(1000,350));
            camiones.setLayout(new BorderLayout());

            camionesimagen.setPreferredSize(new Dimension(500,200));

            camiones.add(tipocamion,BorderLayout.NORTH);
            camiones.add(camionesimagen,BorderLayout.WEST);

            camionesCompleto.add(camiones,BorderLayout.CENTER);
            //Preparamos el panel imagenCentral que contendra todo
            imagenCentral.setLayout(new BorderLayout());
            imagenCentral.add(panelTitulo, BorderLayout.NORTH);
            imagenCentral.add(contenedorRigidosyArticulados, BorderLayout.CENTER);
            imagenCentral.add(camionesCompleto,BorderLayout.SOUTH);
            Formato.setBorder(imagenCentral);


            //Definimos los subpaneles de grabado, borrado e impresion
```

```
            PanelConImagen panelcomandos = new
PanelConImagen(DatosGenericos.getDireccion("Imagenes/fondos/fondo-gris.png"));
            panelcomandos.setLayout(new GridLayout(1,2,0,0));
            panelcomandos.setPreferredSize(new Dimension(200,80));
            //Damos formato
            Formato.setFormatoMenor(graba);
            Formato.setFormatoMenor(borra);
            panelcomandos.add(graba);
            panelcomandos.add(borra);


            //Definimos la ORGANIZACION DE LA VENTANA PRINCIPAL adjuntando los
contenedores.
            setTitle("Calculo Carrozados");
            setBounds(500,500,400,400);
            add(panelInfo, BorderLayout.WEST);
            add(botonesSecundario, BorderLayout.EAST);
            add(imagenCentral,BorderLayout.CENTER);
            add(mensajesCamion,BorderLayout.SOUTH);


            //Introducimos los PANELES DE TEXTOS EN EL PANEL CONTENEDOR DE INFORMACION DE
TEXTO
            //Usamos un SpringLayout para dejar los componentes exactamente a mi gusto
entre ellos
            panelInfo.setPreferredSize(new Dimension(200,800));
            Formato.setBorder(panelInfo);
            int gapX=10;
            int gapY=20;
            SpringLayout layout = new SpringLayout();
            layout.putConstraint(SpringLayout.WEST, panelMarca, 20, SpringLayout.WEST,
panelInfo);
            layout.putConstraint(SpringLayout.WEST, fondoTexto1,gapX,SpringLayout.WEST,
panelInfo);
            layout.putConstraint(SpringLayout.WEST, fondoTexto2,gapX,SpringLayout.WEST,
panelInfo);
            layout.putConstraint(SpringLayout.WEST, fondoTexto3,gapX,SpringLayout.WEST,
panelInfo);
            layout.putConstraint(SpringLayout.WEST, fondoTexto4,gapX,SpringLayout.WEST,
panelInfo);
            layout.putConstraint(SpringLayout.WEST, fondoTexto5,gapX,SpringLayout.WEST,
panelInfo);
            layout.putConstraint(SpringLayout.NORTH, panelMarca,gapY,SpringLayout.NORTH,
panelInfo);
            layout.putConstraint(SpringLayout.NORTH, fondoTexto1,gapY,SpringLayout.SOUTH,
panelMarca);
            layout.putConstraint(SpringLayout.NORTH, fondoTexto2,gapY,SpringLayout.SOUTH,
fondoTexto1);
            layout.putConstraint(SpringLayout.NORTH, fondoTexto3,gapY,SpringLayout.SOUTH,
fondoTexto2);
            layout.putConstraint(SpringLayout.NORTH, fondoTexto4,gapY,SpringLayout.SOUTH,
fondoTexto3);
            layout.putConstraint(SpringLayout.NORTH, fondoTexto5,gapY,SpringLayout.SOUTH,
fondoTexto4);
                    panelInfo.setLayout(layout);
                    panelInfo.add(panelMarca);
            panelInfo.add(fondoTexto1);
            panelInfo.add(fondoTexto2);
            panelInfo.add(fondoTexto3);
            panelInfo.add(fondoTexto4);
            panelInfo.add(fondoTexto5);


            //Definimos la ORGANIZACION BOTONES SECUNDARIO (EL LATERAL DERECHO)
            //Usamos un SpringLayout para dejar los componentes exactamente a mi gusto
entre ellos
            botonesSecundario.setPreferredSize(new Dimension(200,800));
```

```
            Formato.setBorder(botonesSecundario);
            int gapX2=50;
            int gapY2=30;
            SpringLayout layout2 = new SpringLayout();
            layout2.putConstraint(SpringLayout.WEST, panelcomandos, 0, SpringLayout.WEST,
botonesSecundario);
            layout2.putConstraint(SpringLayout.WEST, viga, gapX2, SpringLayout.WEST,
botonesSecundario);
            layout2.putConstraint(SpringLayout.WEST, calculo,gapX2,SpringLayout.WEST,
botonesSecundario);
            layout2.putConstraint(SpringLayout.WEST, resultados,gapX2,SpringLayout.WEST,
botonesSecundario);
            layout2.putConstraint(SpringLayout.NORTH,
panelcomandos,gapY2,SpringLayout.NORTH, botonesSecundario);
            layout2.putConstraint(SpringLayout.NORTH, viga,gapY2,SpringLayout.SOUTH,
panelcomandos);
            layout2.putConstraint(SpringLayout.NORTH, calculo,gapY2,SpringLayout.SOUTH,
viga);
            layout2.putConstraint(SpringLayout.NORTH,
resultados,gapY2,SpringLayout.SOUTH, calculo);
            botonesSecundario.setLayout(layout2);
            botonesSecundario.add(panelcomandos);
            botonesSecundario.add(viga);
            botonesSecundario.add(calculo);
            botonesSecundario.add(resultados);


            //Defino la ORGANIZACION MENSAJES DE TEXTO
            mensajesCamion.setLayout(new GridLayout(0,1,0,0));
            mensajesCamion.add(areaScroll);
            //Formato.setBorder(mensajesCamion);

            //Defino la ORGANIZACION DEL PANEL DE DATOS DEL CAMION
              //Generamos los LAYOUTS A PANELES
              SpringLayout layoutdatos = new SpringLayout();
              int gapX3=20;
              int gapY3=8;
              //Asignamos los constraints
              layoutdatos.putConstraint(SpringLayout.WEST, largototal, gapX3,
SpringLayout.WEST, datosCamion);
              layoutdatos.putConstraint(SpringLayout.EAST, largotext, -gapX3,
SpringLayout.EAST, datosCamion);
              layoutdatos.putConstraint(SpringLayout.WEST, distfrenteprimereje, gapX3,
SpringLayout.WEST, datosCamion);
              layoutdatos.putConstraint(SpringLayout.EAST, frenteprimerejetext, -gapX3,
SpringLayout.EAST,datosCamion);
              layoutdatos.putConstraint(SpringLayout.WEST, entreejes, gapX3,
SpringLayout.WEST,datosCamion);
              layoutdatos.putConstraint(SpringLayout.EAST, distdelanterotraserotext,
-gapX3, SpringLayout.EAST, datosCamion);
              layoutdatos.putConstraint(SpringLayout.WEST, comienzocaja, gapX3,
SpringLayout.WEST, datosCamion);
              layoutdatos.putConstraint(SpringLayout.EAST, comienzocajatext, -gapX3,
SpringLayout.EAST, datosCamion);
              layoutdatos.putConstraint(SpringLayout.WEST, distprimerejekingpin, gapX3,
SpringLayout.WEST,datosCamion);
              layoutdatos.putConstraint(SpringLayout.EAST, distprimerejekingpintext,
-gapX3, SpringLayout.EAST, datosCamion);
              layoutdatos.putConstraint(SpringLayout.WEST, distkingpintandemremolque,
gapX3, SpringLayout.WEST,datosCamion);
              layoutdatos.putConstraint(SpringLayout.EAST, distkingpintandemremolquetext,
-gapX3, SpringLayout.EAST, datosCamion);
              layoutdatos.putConstraint(SpringLayout.WEST, mma, gapX3,
SpringLayout.WEST,datosCamion);
              layoutdatos.putConstraint(SpringLayout.EAST, mmatext, -gapX3,
SpringLayout.EAST, datosCamion);
```

```
            layoutdatos.putConstraint(SpringLayout.WEST, pma, gapX3,
SpringLayout.WEST,datosCamion);
            layoutdatos.putConstraint(SpringLayout.EAST, pmatext, -gapX3,
SpringLayout.EAST, datosCamion);
            layoutdatos.putConstraint(SpringLayout.WEST, cargaejedelantero, gapX3,
SpringLayout.WEST,datosCamion);
            layoutdatos.putConstraint(SpringLayout.EAST, cargaejedelanterotext, -gapX3,
SpringLayout.EAST, datosCamion);
            layoutdatos.putConstraint(SpringLayout.WEST, cargaejetrasero, gapX3,
SpringLayout.WEST,datosCamion);
            layoutdatos.putConstraint(SpringLayout.EAST, cargaejetraserotext, -gapX3,
SpringLayout.EAST, datosCamion);
            layoutdatos.putConstraint(SpringLayout.WEST, cargaejeremolque, gapX3,
SpringLayout.WEST,datosCamion);
            layoutdatos.putConstraint(SpringLayout.EAST, cargaejeremolquetext, -gapX3,
SpringLayout.EAST, datosCamion);
            //Incluimos los elementos en los paneles las restricciones en Y
            layoutdatos.putConstraint(SpringLayout.NORTH, mma, gapY3,
SpringLayout.NORTH,datosCamion);
            layoutdatos.putConstraint(SpringLayout.NORTH, mmatext, gapY3,
SpringLayout.NORTH, datosCamion);
            layoutdatos.putConstraint(SpringLayout.NORTH, pma, gapY3,
SpringLayout.SOUTH,mma);
            layoutdatos.putConstraint(SpringLayout.NORTH, pmatext, gapY3,
SpringLayout.SOUTH, mma);
            layoutdatos.putConstraint(SpringLayout.NORTH, cargaejedelantero, gapY3,
SpringLayout.SOUTH,pma);
            layoutdatos.putConstraint(SpringLayout.NORTH, cargaejedelanterotext, gapY3,
SpringLayout.SOUTH, pma);
            layoutdatos.putConstraint(SpringLayout.NORTH, cargaejetrasero, gapY3,
SpringLayout.SOUTH,cargaejedelantero);
            layoutdatos.putConstraint(SpringLayout.NORTH, cargaejetraserotext, gapY3,
SpringLayout.SOUTH, cargaejedelantero);
            layoutdatos.putConstraint(SpringLayout.NORTH, largototal, gapY3,
SpringLayout.SOUTH, cargaejetrasero);
            layoutdatos.putConstraint(SpringLayout.NORTH, largotext, gapY3,
SpringLayout.SOUTH, cargaejetrasero);
            layoutdatos.putConstraint(SpringLayout.NORTH, distfrenteprimereje, gapY3,
SpringLayout.SOUTH, largototal);
            layoutdatos.putConstraint(SpringLayout.NORTH, frenteprimerejetext, gapY3,
SpringLayout.SOUTH, largototal);
            layoutdatos.putConstraint(SpringLayout.NORTH, entreejes, gapY3,
SpringLayout.SOUTH, distfrenteprimereje);
            layoutdatos.putConstraint(SpringLayout.NORTH, distdelanterotraserotext,
gapY3, SpringLayout.SOUTH, distfrenteprimereje);
            layoutdatos.putConstraint(SpringLayout.NORTH, comienzocaja, gapY3,
SpringLayout.SOUTH,entreejes);
            layoutdatos.putConstraint(SpringLayout.NORTH, comienzocajatext, gapY3,
SpringLayout.SOUTH, entreejes);
            layoutdatos.putConstraint(SpringLayout.NORTH, distprimerejekingpin, gapY3,
SpringLayout.SOUTH,comienzocaja);
            layoutdatos.putConstraint(SpringLayout.NORTH, distprimerejekingpintext,
gapY3, SpringLayout.SOUTH, comienzocaja);
            layoutdatos.putConstraint(SpringLayout.NORTH, distkingpintandemremolque,
gapY3, SpringLayout.SOUTH,distprimerejekingpin);
            layoutdatos.putConstraint(SpringLayout.NORTH,
distkingpintandemremolquetext, gapY3, SpringLayout.SOUTH, distprimerejekingpin);
            layoutdatos.putConstraint(SpringLayout.NORTH, cargaejeremolque, gapY3,
SpringLayout.SOUTH,distkingpintandemremolque);
            layoutdatos.putConstraint(SpringLayout.NORTH, cargaejeremolquetext, gapY3,
SpringLayout.SOUTH, distkingpintandemremolque);
           //largo total del camion
            datosCamion.setLayout(layoutdatos);
           datosCamion.add(largototal);
           preparaTexto(largotext,150);
           datosCamion.add(largotext);
           //distancia de frente al primer eje o tandem
```

```
            datosCamion.add(distfrenteprimereje);
            preparaTexto(frenteprimerejetext,150);
            datosCamion.add(frenteprimerejetext);
            //distancia entre ejes
            datosCamion.add(entreejes);
            preparaTexto(distdelanterotraserotext,150);
            datosCamion.add(distdelanterotraserotext);
            //distancia de primer eje a comienzo caja
            datosCamion.add(comienzocaja);
            preparaTexto(comienzocajatext,150);
            datosCamion.add(comienzocajatext);
            //distancia de primer eje al King Pin
            datosCamion.add(distprimerejekingpin);
            preparaTexto(distprimerejekingpintext,150);
            datosCamion.add(distprimerejekingpintext);
            //distancia de king pin a tandem trasero
            datosCamion.add(distkingpintandemremolque);
            preparaTexto(distkingpintandemremolquetext,150);
            datosCamion.add(distkingpintandemremolquetext);
            datosCamion.add(mma);
            preparaTexto(mmatext,150);
            datosCamion.add(mmatext);
            datosCamion.add(pma);
            preparaTexto(pmatext,150);
            datosCamion.add(pmatext);
            datosCamion.add(cargaejedelantero);
            preparaTexto(cargaejedelanterotext,150);
            datosCamion.add(cargaejedelanterotext);
            datosCamion.add(cargaejetrasero);
            preparaTexto(cargaejetraserotext,150);
            datosCamion.add(cargaejetraserotext);
            datosCamion.add(cargaejeremolque);
            preparaTexto(cargaejeremolquetext,150);
            datosCamion.add(cargaejeremolquetext);
            datosCamion.setPreferredSize(new Dimension(500,300));
            datosCamion.setVisible(false);
            camionesCompleto.add(datosCamion,BorderLayout.EAST);


            //CREAMOS LOS ACTIONLISTENERS
             //ActionListener estructura
            ActionListener accionbtnviga=new ActionListener() {
             public void actionPerformed(ActionEvent e) {
                    //El siguiente booleano recoge la respuesta de seguir o no ante
recomendaciones sobre los datos
                    boolean seguir=true;
                    //Recogemos la accion en el historicoi
                                incluyeInfoArea("Carga propiedades vigas",area);
                                //Revisamos los datos antes de pasar al calculo por
medio del metodo de la clase abstracta Chequea datos
                                Camion c = proyecto.getCamion();
                                if (c.getTipo()==null) {
                                       seguir=false;
                                       CheckDatos.okNokMensaje("Debe seleccionarse un
tipo de camión");
                                }
                                //Pasamos a la ventana de calculo de vigas
                                if(seguir) calculaSeccion();
                         }
             };
             //ActionListener calculos
            ActionListener accionbtncalculo=new ActionListener() {
             public void actionPerformed(ActionEvent e) {
                    try {
                    //El siguiente booleano recoge la respuesta de seguir o no ante
recomendaciones sobre los datos
                    boolean seguir=true;
```

```
                    //Recogemos la accion en el historico
                                incluyeInfoArea("Carga cálculos",area);
                                //Revisamos los datos antes de pasar al calculo por
medio del metodo de la clase abstracta Chequea datos
                                //Primero genero un arraylist con los campos de texto
                                ArrayList<JTextField> jtf = new
ArrayList<JTextField>();
                                jtf.add(mmatext);
                                jtf.add(pmatext);
                                jtf.add(cargaejedelanterotext);
                                jtf.add(cargaejetraserotext);
                                jtf.add(largotext);
                                jtf.add(distdelanterotraserotext);
                                jtf.add(frenteprimerejetext);
                                jtf.add(comienzocajatext);
                                //Si calculamos con un articulado metemos mas campos a
comprobar
                                boolean resultado=true;
                                Camion c = proyecto.getCamion();
                                incluyeInfoArea("Resultado de getCamion()"+c,area);
                                if((c.getTipo()!=null)&&(c.getTipo().getTipo()=='A')) {
                                       jtf.add(cargaejeremolquetext);
                                       jtf.add(distprimerejekingpintext);
                                       jtf.add(distkingpintandemremolquetext);
                                } else if (c.getTipo()==null) {
                                       resultado=false;
                                       CheckDatos.okNokMensaje("Debe seleccionarse un
tipo de camion");
                                }
                                //Los paso a comprobar
                                if(resultado==true)
resultado=resultado&&CheckDatos.corrigeDatos(jtf);
                                if (resultado==true)
resultado=resultado&&CheckDatos.compruebaDatosCompletos(jtf);
                                //Cogemos los valores si todo está correcto
                                if (resultado) {
                                       double
distFrenteEjeDelantero=Double.parseDouble(frenteprimerejetext.getText());
                                       double
distEjeDelanteroFrenteCaja=Double.parseDouble(comienzocajatext.getText());
                                       double
distEjeDelanteroEjeTrasero=Double.parseDouble(distdelanterotraserotext.getText());
                                       double
largo=Double.parseDouble(largotext.getText());
                                       double distprimerEjeKingPin=0;
                                       double distKingPinEjeRemolque=0;
                                       if(proyecto.getCamion().getTipo().getTipo()=='A')
{

distprimerEjeKingPin=Double.parseDouble(distprimerejekingpintext.getText());

distKingPinEjeRemolque=Double.parseDouble(distkingpintandemremolquetext.getText());
                                       }
                                       //Damos avisos sobre los datos
                                       String aviso="";
                                       if (resultado==true) {

aviso=CheckDatos.recomendacionesEntradas(c,distFrenteEjeDelantero,distEjeDelanteroFrenteC
aja,distEjeDelanteroEjeTrasero,largo,distprimerEjeKingPin,
                                                                    distKingPinEjeRemolque);
                                              //Si hay sugerencias las muestro y recojo
si quiero sequir
                                              if (!aviso.equals("RECOMENDACIONES:
\n\n¿Proseguir?")) {

seguir=CheckDatos.okCancelaMensaje(aviso);
                                              }
```

```
                                        }
                                        //Pasamos a la ventana de calculo de cargas
                                        if(resultado&&seguir) {
                                              if(aviso.contains("Se corrige la distancia
del eje de remolque para que quede dentro de los límites del camión")) {
                                                    distKingPinEjeRemolque=(largo-
distFrenteEjeDelantero-distprimerEjeKingPin);

distkingpintandemremolquetext.setText(Double.toString(distKingPinEjeRemolque));
                                              }
                                              if(!aviso.contains("Error")) {
                                                    calculaCargas();
                                              }
                                        }
                                  }
                        } catch (Exception e2) {
                              incluyeInfoArea(e2.toString(),area);
                              incluyeInfoArea(e2.getMessage(),area);
                              incluyeInfoArea(e2.getStackTrace().toString(),area);
                        }
                  }
                  };
                  //ActionListener resultados
                 ActionListener accionbtnresultados=new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        //Recogemos la accion
                        try {
                                        incluyeInfoArea("Acceso a Resultados",area);
                                        CreaPDF.setProyecto(proyecto);
                                        abreDialogoConstructorPDF();
                                        incluyeInfoArea("Fichero de resultados
creado",area);
                        } catch (Exception e2) {
                              CheckDatos.okCancelaMensaje("Ha habido problemas durante la
generación del archivo de resultados");
                        }
                              }
                  };
                  //ActionListener rigido2ejes
                 ActionListener accionbtnr2ejes=new ActionListener() {
                  public void actionPerformed(ActionEvent e) {
                        //Recogemos la accion
                        //Hacemos que el camion sea de tipo Rigido
                        CalculoSeccion cs=null;
                        if(proyecto.getCamion()!=null) {
                              cs=proyecto.getCamion().getSeccion();
                        }
                        CamionRigido c = new CamionRigido();
                        c.setSeccion(cs);
                        c.setEjesPorTandemDelantero(1);
                        c.setEjesPorTandemTrasero(1);
                        proyecto.setCamion(c);
                                    incluyeInfoArea("Acceso a camión rígido 2 ejes",area);
                                    tipocamion.setText("RÍGIDO 2 EJES");
                                    entreejes.setText("Dist. Eje delantero a
trasero("+Unidades.getUnidadesLongitud()+")");
                                    cargaejedelantero.setText("Carga máxima eje
delantero("+Unidades.getUnidadesCarga()+")");
                                    cargaejetrasero.setText("Carga máxima eje
trasero("+Unidades.getUnidadesCarga()+")");
                                    distfrenteprimereje.setText("Dist. Frente a eje
delantero("+Unidades.getUnidadesLongitud()+")");
                                    comienzocaja.setText("Dist. Eje delantero a comienzo
caja("+Unidades.getUnidadesLongitud()+")");
                                    cargaejeremolque.setVisible(false);
                                    cargaejeremolquetext.setVisible(false);
                                    distprimerejekingpin.setVisible(false);
```

```
                                distprimerejekingpintext.setVisible(false);
                                distkingpintandemremolque.setVisible(false);
                                distkingpintandemremolquetext.setVisible(false);
                                datosCamion.setVisible(true);

camionesimagen.setImagen(DatosGenericos.getDireccion("Imagenes/pantallaprincipal/
rigido2.png"));
                                anulaseleccion();
                                Formato.setBordeSeleccionado(r2ejes);
                                r2ejes.setBorderPainted(true);
                        }
             };
             //ActionListener rigido3ejes
            ActionListener accionbtnr3ejes=new ActionListener() {
             public void actionPerformed(ActionEvent e) {
                    //Recogemos la accion
                    CalculoSeccion cs=null;
                    if(proyecto.getCamion()!=null) {
                           cs=proyecto.getCamion().getSeccion();
                    }
                    CamionRigido c = new CamionRigido();
                    c.setSeccion(cs);
                    c.setEjesPorTandemDelantero(1);
                    c.setEjesPorTandemTrasero(2);
                    proyecto.setCamion(c);
                                incluyeInfoArea("Acceso a camión rígido 3 ejes",area);
                                tipocamion.setText("RÍGIDO 3 EJES");
                                entreejes.setText("Dist. Eje delantero a
trasero("+Unidades.getUnidadesLongitud()+")");
                                cargaejedelantero.setText("Carga máxima eje
delantero("+Unidades.getUnidadesCarga()+")");
                                cargaejetrasero.setText("Carga máxima tandem
trasero("+Unidades.getUnidadesCarga()+")");
                                distfrenteprimereje.setText("Dist. Frente a eje
delantero("+Unidades.getUnidadesLongitud()+")");
                                comienzocaja.setText("Dist. Eje delantero a comienzo
caja("+Unidades.getUnidadesLongitud()+")");
                                cargaejeremolque.setVisible(false);
                                cargaejeremolquetext.setVisible(false);
                                distprimerejekingpin.setVisible(false);
                                distprimerejekingpintext.setVisible(false);
                                distkingpintandemremolque.setVisible(false);
                                distkingpintandemremolquetext.setVisible(false);
                                datosCamion.setVisible(true);

camionesimagen.setImagen(DatosGenericos.getDireccion("Imagenes/pantallaprincipal/
rigido3.png"));
                                anulaseleccion();
                                Formato.setBordeSeleccionado(r3ejes);
                                r3ejes.setBorderPainted(true);
                        }
             };
             //ActionListener rigido4ejes
            ActionListener accionbtnr4ejes=new ActionListener() {
             public void actionPerformed(ActionEvent e) {
                    //Recogemos la accionCamionRigido c = new CamionRigido();
                    CalculoSeccion cs=null;
                    if(proyecto.getCamion()!=null) {
                           cs=proyecto.getCamion().getSeccion();
                    }
                    CamionRigido c = new CamionRigido();
                    c.setSeccion(cs);
                    c.setEjesPorTandemDelantero(2);
                    c.setEjesPorTandemTrasero(2);
                    proyecto.setCamion(c);
                                incluyeInfoArea("Acceso a camión rígido 4 ejes",area);
                                tipocamion.setText("RÍGIDO 4 EJES");
```

```
                                    entreejes.setText("Dist. Eje delantero a
trasero("+Unidades.getUnidadesLongitud()+")");
                                    cargaejedelantero.setText("Carga máxima tandem
delantero("+Unidades.getUnidadesCarga()+")");
                                    cargaejetrasero.setText("Carga máxima tandem
trasero("+Unidades.getUnidadesCarga()+")");
                                    distfrenteprimereje.setText("Dist. Frente a trandem
delantero("+Unidades.getUnidadesLongitud()+")");
                                    comienzocaja.setText("Dist. Eje delantero a comienzo
caja("+Unidades.getUnidadesLongitud()+")");
                                    cargaejeremolque.setVisible(false);
                                    cargaejeremolquetext.setVisible(false);
                                    distprimerejekingpin.setVisible(false);
                                    distprimerejekingpintext.setVisible(false);
                                    distkingpintandemremolque.setVisible(false);
                                    distkingpintandemremolquetext.setVisible(false);
                                    datosCamion.setVisible(true);

camionesimagen.setImagen(DatosGenericos.getDireccion("Imagenes/pantallaprincipal/
rigido4.png"));
                                    anulaseleccion();
                                    Formato.setBordeSeleccionado(r4ejes);
                                    r4ejes.setBorderPainted(true);
                            }
                };
                //ActionListener articulado 4 ejes
               ActionListener accionbtna4ejes=new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                        //Recogemos la accion
                        CalculoSeccion cs=null;
                        if(proyecto.getCamion()!=null) {
                                cs=proyecto.getCamion().getSeccion();
                        }
                        CamionArticulado c = new CamionArticulado();
                        c.setSeccion(cs);
                        c.setEjesPorTandemDelantero(1);
                        c.setEjesPorTandemTrasero(1);
                        c.setEjesPorTandemRemolque(2);
                        proyecto.setCamion(c);
                                    incluyeInfoArea("Acceso a camión articulado 4 ejes, 2
en tractora y 2 en remolque",area);
                                    tipocamion.setText("ARTICULADO 4 EJES (2 TRACTORA + 2
REMOLQUE)");
                                    entreejes.setText("Dist. Eje delantero a trasero de
tractora("+Unidades.getUnidadesLongitud()+")");
                                    cargaejedelantero.setText("Carga máxima eje delantero
tractora("+Unidades.getUnidadesCarga()+")");
                                    cargaejetrasero.setText("Carga máxima eje trasero
tractora("+Unidades.getUnidadesCarga()+")");
                                    distfrenteprimereje.setText("Dist. Frente a eje
delantero("+Unidades.getUnidadesLongitud()+")");
                                    comienzocaja.setText("Dist. Eje delantero a inicio
remolque("+Unidades.getUnidadesLongitud()+")");
                                    cargaejeremolque.setVisible(true);
                                    cargaejeremolquetext.setVisible(true);
                                    distprimerejekingpin.setVisible(true);
                                    distprimerejekingpintext.setVisible(true);
                                    distkingpintandemremolque.setVisible(true);
                                    distkingpintandemremolquetext.setVisible(true);
                                    datosCamion.setVisible(true);

camionesimagen.setImagen(DatosGenericos.getDireccion("Imagenes/pantallaprincipal/
articulado2-2.png"));
                                    anulaseleccion();
                                    Formato.setBordeSeleccionado(a4ejes);
                                    a4ejes.setBorderPainted(true);
                            }
```

```
            };
            //ActionListener articulado 5 ejes 2+3
           ActionListener accionbtna5ejes23=new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                  //Recogemos la accion
                  CalculoSeccion cs=null;
                  if(proyecto.getCamion()!=null) {
                        cs=proyecto.getCamion().getSeccion();
                  }
                  CamionArticulado c = new CamionArticulado();
                  c.setSeccion(cs);
                  c.setEjesPorTandemDelantero(1);
                  c.setEjesPorTandemTrasero(1);
                  c.setEjesPorTandemRemolque(3);
                  proyecto.setCamion(c);
                              incluyeInfoArea("Acceso a camión articulado 5 ejes, 2
en tractora y 3 en remolque",area);
                              tipocamion.setText("ARTICULADO 5 EJES (2 TRACTORA + 3
REMOLQUE)");
                              entreejes.setText("Dist. Eje delantero a trasero de
tractora("+Unidades.getUnidadesLongitud()+")");
                              cargaejedelantero.setText("Carga máxima eje delantero
tractora("+Unidades.getUnidadesCarga()+")");
                              cargaejetrasero.setText("Carga máxima eje trasero
tractora("+Unidades.getUnidadesCarga()+")");
                              distfrenteprimereje.setText("Dist. Frente a eje
delantero("+Unidades.getUnidadesLongitud()+")");
                              comienzocaja.setText("Dist. Eje delantero a inicio de
remolque("+Unidades.getUnidadesLongitud()+")");
                              cargaejeremolque.setVisible(true);
                              cargaejeremolquetext.setVisible(true);
                              distprimerejekingpin.setVisible(true);
                              distprimerejekingpintext.setVisible(true);
                              distkingpintandemremolque.setVisible(true);
                              distkingpintandemremolquetext.setVisible(true);
                              datosCamion.setVisible(true);

camionesimagen.setImagen(DatosGenericos.getDireccion("Imagenes/pantallaprincipal/
articulado2-3.png"));
                              anulaseleccion();
                              Formato.setBordeSeleccionado(a5ejes23);
                              a5ejes23.setBorderPainted(true);
                        }
            };
            //ActionListener articulado 5 ejes 3+2
           ActionListener accionbtna5ejes32=new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                  //Recogemos la accion
                  CalculoSeccion cs=null;
                  if(proyecto.getCamion()!=null) {
                        cs=proyecto.getCamion().getSeccion();
                  }
                  CamionArticulado c = new CamionArticulado();
                  c.setSeccion(cs);
                  c.setEjesPorTandemDelantero(1);
                  c.setEjesPorTandemTrasero(2);
                  c.setEjesPorTandemRemolque(2);
                  proyecto.setCamion(c);
                              incluyeInfoArea("Acceso a camión articulado 5 ejes, 3
en tractora y 2 en remolque",area);
                              tipocamion.setText("ARTICULADO 5 EJES (3 TRACTORA + 2
REMOLQUE)");
                              entreejes.setText("Dist. Eje delantero a trasero de
tractora("+Unidades.getUnidadesLongitud()+")");
                              cargaejedelantero.setText("Carga máxima eje delantero
tractora("+Unidades.getUnidadesCarga()+")");
```

```
                                cargaejetrasero.setText("Carga máxima tandem trasero
tractora("+Unidades.getUnidadesCarga()+")");
                                distfrenteprimereje.setText("Dist. Frente a tandem
delantero("+Unidades.getUnidadesLongitud()+")");
                                comienzocaja.setText("Dist. Eje delantero a inicio de
remolque("+Unidades.getUnidadesLongitud()+")");
                                cargaejeremolque.setVisible(true);
                                cargaejeremolquetext.setVisible(true);
                                distprimerejekingpin.setVisible(true);
                                distprimerejekingpintext.setVisible(true);
                                distkingpintandemremolque.setVisible(true);
                                distkingpintandemremolquetext.setVisible(true);
                                datosCamion.setVisible(true);

camionesimagen.setImagen(DatosGenericos.getDireccion("Imagenes/pantallaprincipal/
articulado3-2.png"));
                                anulaseleccion();
                                Formato.setBordeSeleccionado(a5ejes32);
                                a5ejes32.setBorderPainted(true);
                        }
              };
              //ActionListener articulado 6 ejes
             ActionListener accionbtna6ejes=new ActionListener() {
              public void actionPerformed(ActionEvent e) {
                     //Recogemos la accion
                     CalculoSeccion cs=null;
                     if(proyecto.getCamion()!=null) {
                            cs=proyecto.getCamion().getSeccion();
                     }
                     CamionArticulado c = new CamionArticulado();
                     c.setSeccion(cs);
                     c.setEjesPorTandemDelantero(1);
                     c.setEjesPorTandemTrasero(2);
                     c.setEjesPorTandemRemolque(3);
                     proyecto.setCamion(c);
                                incluyeInfoArea("Acceso a camión articulado 6 ejes, 3
en tractora y 3 en remolque",area);
                                tipocamion.setText("ARTICULADO 6 EJES (3 TRACTORA + 3
REMOLQUE)");
                                entreejes.setText("Dist. Eje delantero a trasero de
tractora("+Unidades.getUnidadesLongitud()+")");
                                cargaejedelantero.setText("Carga máxima eje delantero
tractora("+Unidades.getUnidadesCarga()+")");
                                cargaejetrasero.setText("Carga máxima tandem trasero
tractora("+Unidades.getUnidadesCarga()+")");
                                comienzocaja.setText("Dist. Eje delantero a inicio de
remolque("+Unidades.getUnidadesLongitud()+")");
                                cargaejeremolque.setVisible(true);
                                cargaejeremolquetext.setVisible(true);
                                distprimerejekingpin.setVisible(true);
                                distprimerejekingpintext.setVisible(true);
                                distkingpintandemremolque.setVisible(true);
                                distkingpintandemremolquetext.setVisible(true);
                                datosCamion.setVisible(true);

camionesimagen.setImagen(DatosGenericos.getDireccion("Imagenes/pantallaprincipal/
articulado3-3.png"));
                                anulaseleccion();
                                Formato.setBordeSeleccionado(a6ejes);
                                a6ejes.setBorderPainted(true);
                        }
              };
              //ActionListener boton graba
             ActionListener accionBtnGraba=new ActionListener() {
              public void actionPerformed(ActionEvent e) {
                     //Recogemos la accion de grabar
                     asignaDatosAProyecto();
```

```
                        }
              };
              //ActionListener boton borra
             ActionListener accionBtnBorra=new ActionListener() {
              public void actionPerformed(ActionEvent e) {
                     if(okNokMensaje("Todos los datos del proyecto introducidos se
eliminarán.\nLos ficheros guardados no serán eliminados\n¿Está seguro?")==0) {
                     borraDatos(grupoInfo);
                     renuevaProyecto();
                     datosCamion.setVisible(false);
                     camionesimagen.setImagen(DatosGenericos.getDireccion("Imagenes/
fondos/fondo-gris.png"));
                     tipocamion.setText("");
                     }
                            }
              };
              //ActionListener menu Acerca de TruckCalc
             ActionListener accionMenuAcerca=new ActionListener() {
              public void actionPerformed(ActionEvent e) {
                     //Recogemos la accion de grabar
                     CheckDatos.okNokMensaje("TruckCalc está creado por Juan Manuel Gómez
García.\n"
                                   +"Es un Proyecto Fin de Grado para el Grado en
Ingeniería Informática en la UNED.\n"
                                   ,"Información sobre TruckCalc");
                            }
              };
              //ActionListener menu Importa de Base de datos
              ActionListener accionImportadeBBDD=new ActionListener() {
              public void actionPerformed(ActionEvent e) {
                     abreVentanaBaseDatosCamion();

                     //La siguiente llamada a asignaDatosAProyecto se ha cambiado a la
base VentanaBaseDatosCamion.
                     //Si hubiese problemas cambiarlo aquí
                     //asignaDatosAProyecto();
                     }
              };

              ActionListener accionGuardaEnBBDD=new ActionListener() {
              public void actionPerformed(ActionEvent e) {
                     incluyeNuevosDatosCamion();
                     }
              };

                     ActionListener accionGuardaEnFichero=new ActionListener() {
                     public void actionPerformed(ActionEvent e) {
                            guardaFichero();
                            }
                     };

                     ActionListener accionCargaDeFichero=new ActionListener() {
                     public void actionPerformed(ActionEvent e) {
                            abreFichero();
                            //RecogeCalculos
                            calculaCargas();
                            ventCalc.setVisible(false);
                            }
                     };
             ActionListener cambiaUnidades=new ActionListener() {
              public void actionPerformed(ActionEvent e) {
                     if (e.getSource()==menuM) {
                         setUnidades("m",Unidades.getUnidadesCarga());
                     }
                     if (e.getSource()==menuDm) {
                         setUnidades("dm",Unidades.getUnidadesCarga());
                     }
```

```
        if (e.getSource()==menuCm) {
            setUnidades("cm",Unidades.getUnidadesCarga());
        }
        if (e.getSource()==menuMm) {
            setUnidades("mm",Unidades.getUnidadesCarga());
        }
        if (e.getSource()==menuKg) {
            setUnidades(Unidades.getUnidadesLongitud(),"Kg");
        }
        if (e.getSource()==menuNw) {
            setUnidades(Unidades.getUnidadesLongitud(),"N");
        }
    }
 };


 //ASIGNAMOS ACTIONLISTENERS A ELEMENTOS
viga.addActionListener(accionbtnviga);
calculo.addActionListener(accionbtncalculo);
resultados.addActionListener(accionbtnresultados);
r2ejes.addActionListener(accionbtnr2ejes);
r3ejes.addActionListener(accionbtnr3ejes);
r4ejes.addActionListener(accionbtnr4ejes);
a4ejes.addActionListener(accionbtna4ejes);
a5ejes23.addActionListener(accionbtna5ejes23);
a5ejes32.addActionListener(accionbtna5ejes32);
a6ejes.addActionListener(accionbtna6ejes);
graba.addActionListener(accionBtnGraba);
borra.addActionListener(accionBtnBorra);
menuAcerca.addActionListener(accionMenuAcerca);
importaBBDD.addActionListener(accionImportadeBBDD);
guardaBBDD.addActionListener(accionGuardaEnBBDD);
 menuCm.addActionListener(cambiaUnidades);
 menuDm.addActionListener(cambiaUnidades);
 menuMm.addActionListener(cambiaUnidades);
 menuM.addActionListener(cambiaUnidades);
 menuKg.addActionListener(cambiaUnidades);
 menuNw.addActionListener(cambiaUnidades);
 menuGuardaFichero.addActionListener(accionGuardaEnFichero);
 menuCargaFichero.addActionListener(accionCargaDeFichero);


//ASIGNAMOS MOUSELISTENERS A ELEMENTOS
area.addMouseListener(new MouseListener() {
                @Override
                public void mouseClicked(MouseEvent e) {
                       // TODO Auto-generated method stub
                       if (e.getClickCount() == 2 && !e.isConsumed()) {
                e.consume();
                ventanaArea();
      }
                }
                @Override
                public void mousePressed(MouseEvent e) {
                       // TODO Auto-generated method stub

                }
                @Override
                public void mouseReleased(MouseEvent e) {
                       // TODO Auto-generated method stub

                }
                @Override
                public void mouseEntered(MouseEvent e) {
                       // TODO Auto-generated method stub

                }
```

```
                        @Override
                        public void mouseExited(MouseEvent e) {
                               // TODO Auto-generated method stub

                        }});


            //Hacemos visible la ventana
            setResizable(true);
            setExtendedState(MAXIMIZED_BOTH);
            pack();
            setVisible(true);
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        }


        //Este método asigna a todos los componentes de la ventana el mismo tipo de texto
        private static void darTipoLetra(ArrayList<Component> c) {
               Iterator<Component> iter = c.iterator();
               while(iter.hasNext()){
                     ponLetra(iter.next());
               }
        }


        //Este método da el tipo letra a cada componente igual
        private static void ponLetra(Component c) {
             Formato.setFormatoEstandar(c);
        }


        //Este método escala los iconos de los botones con un coeficiente de escala para
evitar problemas por tamaños
        private ImageIcon escalaIcono(ImageIcon i, int s) {
             return new ImageIcon(i.getImage().getScaledInstance(80*s, -1,
java.awt.Image.SCALE_DEFAULT));
        }


        //Este método asigna los iconos a los botones disponiendo el texto y la imagen en
la posicion deseada
        private void asignaIconoBoton(JButton c,ImageIcon i) {
             c.setFocusPainted(false);
             c.setBorderPainted(false);
             c.setContentAreaFilled(false);
             c.setIcon(i);
            c.setIconTextGap(5);
            c.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
            c.setVerticalAlignment(javax.swing.SwingConstants.CENTER);
            c.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
            c.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
        }


        //Vamos guardando la información del area de texto con las acciones enviamos la
nueva info y devuelve todo
        private void incluyeInfoArea(String s, JTextArea jta) {
             this.areaTexto=areaTexto+s+"\n";
             jta.setText(areaTexto);
             jta.setSize(0, 30);
        }


        private void preparaTexto(JTextField texto, int ancho) {
             texto.setPreferredSize(new Dimension(ancho,25));
             Formato.setFormatoEstandar(texto);
             Formato.setBorder(texto);
```

```
        }


        private void preparaLabel(JLabel label, int ancho){
              label.setPreferredSize(new Dimension(150,30));
              Formato.setFormatoEstandar(label);
              ponLetra(label);
              label.setHorizontalAlignment(SwingConstants.LEFT);
        }


        //Metodo para generar las areas de texto con los label y campo de texto para
entrada de informacion
        private void preparaLabelTexto(JPanel panel, int ancho, int alto, JLabel label,
JTextField texto) {
              panel.setPreferredSize(new Dimension(ancho,alto));
             preparaTexto(texto, (ancho-20));
             preparaLabel(label, (ancho-10));
             panel.add(label);
             panel.add(texto);
        }


        //Metodo para genera una ventana de Cálculo.
        private void calculaCargas() {
              //Metemos los datos del camion en el proyecto
              asignaDatosAProyecto();
              if(ventCalc!=null) {
                           ventCalc.reseteaVentanas();
                     ventCalc.rellenaTablaReacciones();
                     ventCalc.rellenaTablaCargas();
                           incluyeInfoArea("Rellenadas reacciones y cargas",area);
                     ventCalc.rellenaTablaTensiones();
                           incluyeInfoArea("Rellenada tabla de tensiones",area);
                     ventCalc.repaint();
                     ventCalc.setVisible(true);
                     ventCalc.repaint();
              }
              else {
                           ventCalc = new VentanaCalculos(proyecto,this);
                           //Recalculamos las reacciones para poderlas pintar llamando al
método calcularReaccionesActuales.
                           ventCalc.calcularReaccionesyTensionesActuales();
                           ventCalc.setVisible(true);
              }
              if(ventSec!=null) ventSec.setVisible(false);
        }


        //Metodo para generar una sección determinada.
        private void calculaSeccion() {
                     ventSec = new VentanaSeccion(proyecto);
        }


        //Este método permite extraer la información de la ventana de procesos a una
ventana mayor para una lectura más comoda
        private void ventanaArea() {
              VentanaProceso va = new VentanaProceso(areaTexto);
        }

        //Este metodo sirve para dejar todos los botones de tipo de camion como no
marcados una vez se borran los datos
        private void anulaseleccion() {
              JButton b;
              Iterator<JButton> iter = camionseleccion.iterator();
               while(iter.hasNext()){
```

```
                    b=iter.next();
                    Formato.setBordeEstandar(b);
                          b.setBorderPainted(false);
              }
        }


        //Este metodo ajusta la escala del icono al tamaño deseado
        private ImageIcon ajustaEscala(ImageIcon i, int ancho) {
              int alto = -1; // alto (para que conserve la proporcion pasamos -1)
              ImageIcon iconoOK = new ImageIcon(i.getImage().getScaledInstance(ancho,
alto, java.awt.Image.SCALE_DEFAULT));
              return iconoOK;
        }


        //Este metodo borra los datos de los campos de texto una vez se selecciona borrar
        private void borraDatos(ArrayList<JTextComponent> c) {
              Iterator<JTextComponent> iter = c.iterator();
               while(iter.hasNext()){
                    iter.next().setText("");
               }
               anulaseleccion();
        }


        //Este método muestra un mensaje de sistema xcon el icono de la aplicacion y el
string pasado
        private int okNokMensaje(String s) {
              ImageIcon iconoAjustable = new
ImageIcon(DatosGenericos.getDireccion("Imagenes/pantallaprincipal/recurso.png"));
              ImageIcon icon=ajustaEscala(iconoAjustable,120);
              return
JOptionPane.showOptionDialog(null,s,"Advertencia",0,1,icon,null,null);
        }

        private static void asignaproyecto(Proyecto p) {
              proyecto=p;
        }

        //Este método asigna los datos del camion al proyecto
        public void asignaDatosAProyecto() {
              //Al asignar nos aseguramos de hacerlo en las unidades por defecto que son
Kg y mm
              //Solo se guardan si son distintos a los valores que ya habia
              //Usamos los metodos de unidades convertLong y convertCarga
              Camion c=proyecto.getCamion();
              c.setCliente(nombretext.getText());
              c.setBastidor(bastidortext.getText());
              c.setMatricula(matriculatext.getText());
              c.setMarca(marcatext.getText());
              c.setModelo(modelotext.getText());
              //Asignamos los datos pasados a las unidades por defecto
              double comienzocaja=c.getDistanciaEjeDelanteroComienzoCaja();
              double cargadelantero=c.getCargaMaxEjeDelantero();
              double cargatrasero=c.getCargaMaxEjeTrasero();
              double distanciadelanterotrasero=c.getdistEjeDelanteroEjeTrasero();
              double distanciafrenteprimereje=c.getdistFrenteEjeDelantero();
              double longitud=c.getLargo();
              double masamax=c.getMMA();
              double pesomax=c.getPMA();

              //Primero comprobamos los datos generales (valores cero, etc.., aquí no
entramos dar consejos sobre las medidas pero si en ver si
              //los datos son coherentes (son numeros, no hay dobles puntos, etc...)
              //Usamos un booleano para ver si el resultado es correcto
                    boolean resultado=true;
```

```
                //Meteremos los campos a comprobar en un ArrayList
            ArrayList<JTextField> jtf = new ArrayList<JTextField>();
            //Metemos los campos
                jtf.add(mmatext);
                jtf.add(pmatext);
                jtf.add(cargaejedelanterotext);
                jtf.add(cargaejetraserotext);
                jtf.add(largotext);
                jtf.add(distdelanterotraserotext);
                jtf.add(frenteprimerejetext);
                jtf.add(comienzocajatext);
                //Si calculamos con un articulado metemos mas campos a comprobar
                if((c.getTipo()!=null)&&(c.getTipo().getTipo()=='A')) {
                        jtf.add(cargaejeremolquetext);
                        jtf.add(distprimerejekingpintext);
                        jtf.add(distkingpintandemremolquetext);
                } else if (c.getTipo()==null) {
                        resultado=false;
                        CheckDatos.okNokMensaje("Debe seleccionarse un tipo de
camión");
                }
                //Los paso a comprobar
                if (resultado==true)
resultado=resultado&&CheckDatos.compruebaDatosCompletos(jtf);
                if(resultado==true)
resultado=resultado&&CheckDatos.corrigeDatos(jtf);
            //Solo asigna valores si han cambiado
            //Lo intentamos en un try catch por si hay problemas de formato
                if(resultado) {
                try {
                        if(comienzocaja!
=Unidades.getConversionLongitud()*Double.parseDouble(comienzocajatext.getText().replace('
,', '.')))
c.setEjeDelanteroComienzoCaja(Unidades.getConversionLongitud()*Double.parseDouble(comienz
ocajatext.getText().replace(',', '.')));
                        if(cargadelantero!
=Unidades.getConversionCarga()*Double.parseDouble(cargaejedelanterotext.getText().replace
(',', '.')))
c.setCargaMaxEjeDelantero(Unidades.getConversionCarga()*Double.parseDouble(cargaejedelant
erotext.getText().replace(',', '.')));
                        if(cargatrasero!
=Unidades.getConversionCarga()*Double.parseDouble(cargaejetraserotext.getText().replace('
,', '.')))
c.setCargaMaxEjeTrasero(Unidades.getConversionCarga()*Double.parseDouble(cargaejetraserot
ext.getText().replace(',', '.')));
                        if(distanciadelanterotrasero!
=Unidades.getConversionLongitud()*Double.parseDouble(distdelanterotraserotext.getText().r
eplace(',', '.')))
c.setEjeDelanteroEjeTrasero(Unidades.getConversionLongitud()*Double.parseDouble(distdelan
terotraserotext.getText().replace(',', '.')));
                        if(distanciafrenteprimereje!
=Unidades.getConversionLongitud()*Double.parseDouble(frenteprimerejetext.getText().replac
e(',', '.')))
c.setFrenteEjeDelantero(Unidades.getConversionLongitud()*Double.parseDouble(frenteprimere
jetext.getText().replace(',', '.')));
                        if(longitud!
=Unidades.getConversionLongitud()*Double.parseDouble(largotext.getText().replace(',',
'.')))
c.setLongitud(Unidades.getConversionLongitud()*Double.parseDouble(largotext.getText().rep
lace(',', '.')));
                        if(masamax!
=Unidades.getConversionCarga()*Double.parseDouble(mmatext.getText().replace(',', '.')))
c.setMMA(Unidades.getConversionCarga()*Double.parseDouble(mmatext.getText().replace(',',
'.')));
                        if(pesomax!
=Unidades.getConversionCarga()*Double.parseDouble(pmatext.getText().replace(',',
```

```
'.')))c.setPMA(Unidades.getConversionCarga()*Double.parseDouble(pmatext.getText().replace
(',', '.')));

                              //Si es Articulado incluimos mas datos
                              if((c.getTipo()!=null)&&(c.getTipo().getTipo()=='A')) {
                                    CamionArticulado ca = (CamionArticulado)c;
                                    double cargaremolque=ca.getCargaMaximaEjeRemolque();
                                    double
distanciaprimerejekingpin=ca.getDistanciaEjeDelanteroKingPin();
                                    double
distanciakingpinejeremolque=ca.getDistanciaKingPinEjeRemolque();
                                    //Solo asigna valores si han cambiado
                                    if(cargaremolque!
=Double.parseDouble(cargaejeremolquetext.getText().replace(',', '.')))
ca.setCargaMaximaEjeRemolque(Unidades.getConversionCarga()*Double.parseDouble(cargaejerem
olquetext.getText().replace(',', '.')));
                                    if(distanciaprimerejekingpin!
=Double.parseDouble(distprimerejekingpintext.getText().replace(',', '.')))
ca.setDistanciaEjeDelanteroKingPin(Unidades.getConversionLongitud()*Double.parseDouble(di
stprimerejekingpintext.getText().replace(',', '.')));
                                    if(distanciakingpinejeremolque!
=Double.parseDouble(distkingpintandemremolquetext.getText().replace(',', '.')))
ca.setDistanciaKingPinEjeRemolque(Unidades.getConversionLongitud()*Double.parseDouble(dis
tkingpintandemremolquetext.getText().replace(',', '.')));
                                    //Comprobamos otros valores de calculo
                                    double
largo=Unidades.getConversionLongitud()*Double.parseDouble(largotext.getText().replace(','
, '.'));
                                    double
frenteEjeDelantero=Unidades.getConversionLongitud()*Double.parseDouble(frenteprimerejetex
t.getText().replace(',', '.'));
                                    double
delanteroKingPin=Unidades.getConversionLongitud()*Double.parseDouble(distprimerejekingpin
text.getText().replace(',', '.'));
                                    double
delanteroTrasero=Unidades.getConversionLongitud()*Double.parseDouble(distdelanterotrasero
text.getText().replace(',', '.'));
                                    //Cogemos los valores guardados
                                    double longitudguardada=ca.getLargo();
                                    double
distanciafrentedelanteroguardada=ca.getdistFrenteEjeDelantero();
                                    double
distanciaejedelanterokingpinguardada=ca.getDistanciaEjeDelanteroKingPin();
                                    double
distanciadelanterotraseroguardada=ca.getdistEjeDelanteroEjeTrasero();
                                    //Solo los guardamos si han cambiado
                                    if((largo!=longitudguardada)||(frenteEjeDelantero!
=distanciafrentedelanteroguardada)||(delanteroKingPin!
=distanciaejedelanterokingpinguardada))

ca.setLongitudCalculoRemolque(Unidades.getConversionLongitud()*(largo-frenteEjeDelantero-
delanteroKingPin));
                                    if((frenteEjeDelantero!
=distanciafrentedelanteroguardada)||(delanteroTrasero!
=distanciadelanterotraseroguardada))

ca.setLongitudCalculoCabeza(Unidades.getConversionLongitud()*(frenteEjeDelantero+delanter
oTrasero+Unidades.getConversionCarga()*500));

                              }
                        } catch (Exception e){
                              CheckDatos.okNokMensaje("Se ha producido un error al intentar
guardar los datos.\n Por favor, revise los datos");
                        }
                        }

            daFormato();
```

```
        }

        private void renuevaProyecto() {
              proyecto=new Proyecto();
              //BORRA ESTA LLAMADA A rellenaDatosPrueba
              rellenaDatos();
        }

        //Este metodo sirve para actualizar los valores cuando camvian en las etiquetas.
        public void rellenaDatos() {
              //En funcion del tipo de camion hacemos click en su boton para ejecutar el
codigo correspondiente
              //Actualizamos a las nuevas unidades las etiquetas
              actualizaUnidadesLabels();
              //Formatearemos los datos con 2 decimales
              DecimalFormat df=new DecimalFormat("0.00");
              if(proyecto.getCamion()!=null) {
                    //Primero los datos generales
                    Camion c=proyecto.getCamion();
                    marcatext.setText(c.getMarca());
                    nombretext.setText(c.getCliente());
                    matriculatext.setText(c.getMatricula());
                    bastidortext.setText(c.getBastidor());
                    modelotext.setText(c.getModelo());

                    //Ahora los datos técnicos

mmatext.setText(df.format(Unidades.getCargaEnUnidadesActuales(c.getMMA())));

pmatext.setText(df.format(Unidades.getCargaEnUnidadesActuales(c.getPMA())));

cargaejedelanterotext.setText(df.format(Unidades.getCargaEnUnidadesActuales(c.getCargaMax
EjeDelantero())));

cargaejetraserotext.setText(df.format(Unidades.getCargaEnUnidadesActuales(c.getCargaMaxEj
eTrasero())));

largotext.setText(df.format(Unidades.getLongitudEnUnidadesActuales(c.getLargo())));

frenteprimerejetext.setText(df.format(Unidades.getLongitudEnUnidadesActuales(c.getdistFre
nteEjeDelantero())));

distdelanterotraserotext.setText(df.format(Unidades.getLongitudEnUnidadesActuales(c.getdi
stEjeDelanteroEjeTrasero())));

comienzocajatext.setText(df.format(Unidades.getLongitudEnUnidadesActuales(c.getDistanciaE
jeDelanteroComienzoCaja())));
                    if((c.getTipo()!=null)&&(c.getTipo().getTipo()=='A')) {
                          CamionArticulado ca=(CamionArticulado)c;

distprimerejekingpintext.setText(df.format(Unidades.getLongitudEnUnidadesActuales(ca.getD
istanciaEjeDelanteroKingPin())));

distkingpintandemremolquetext.setText(df.format(Unidades.getLongitudEnUnidadesActuales(ca
.getDistanciaKingPinEjeRemolque())));

cargaejeremolquetext.setText(df.format(Unidades.getCargaEnUnidadesActuales(ca.getCargaMax
imaEjeRemolque())));
                     } else {
                   cargaejeremolquetext.setText("0,00");
                   distprimerejekingpintext.setText("0,00");
                   distkingpintandemremolquetext.setText("0,00");
                    }
              } else {
                   mmatext.setText("0,00");
                   pmatext.setText("0,00");
                   cargaejedelanterotext.setText("0,00");
```

```
                cargaejetraserotext.setText("0,00");
                largotext.setText("0,00");
                frenteprimerejetext.setText("0,00");
                distdelanterotraserotext.setText("0,00");
                comienzocajatext.setText("0,00");
                cargaejeremolquetext.setText("0,00");
                distprimerejekingpintext.setText("0,00");
                distkingpintandemremolquetext.setText("0,00");
            }

             if(proyecto.getCamion().getTipo()!=null) {
                    switch(proyecto.getCamion().getTipo().getTipo()) {
                           case 'R':{
                                  String ejesRig="";
                                  String
ejesD=Integer.toString(((CamionRigido)proyecto.getCamion()).getEjesPorTandemDelantero());
                                  String
ejesT=Integer.toString(((CamionRigido)proyecto.getCamion()).getEjesPorTandemTrasero());
                                  ejesRig=ejesD+ejesT;
                                  //Vemos los ejes que tiene para ver el boton que
clickar
                                  switch(ejesRig) {
                                         case "11": r2ejes.doClick(); break;
                                         case "12": r3ejes.doClick(); break;
                                         case "22": r4ejes.doClick(); break;
                                         default: break;
                                  }
                           }; break;
                           case 'A':{
                                  String ejesArt="";
                                  String
ejesD=Integer.toString(((CamionArticulado)proyecto.getCamion()).getEjesPorTandemDelantero
());
                                  String
ejesT=Integer.toString(((CamionArticulado)proyecto.getCamion()).getEjesPorTandemTrasero()
);
                                  String
ejesR=Integer.toString(((CamionArticulado)proyecto.getCamion()).getEjesPorTandemRemolque(
));
                                  ejesArt=ejesD+ejesT+ejesR;
                                  //Vemos los ejes que tiene para ver el boton que
clickar
                                         switch(ejesArt) {
                                                case "112": a4ejes.doClick(); break;
                                                case "113": a5ejes23.doClick(); break;
                                                case "122": a5ejes32.doClick(); break;
                                                case "123": a6ejes.doClick(); break;
                                                default: break;
                                         }
                           }; break;
                           default: break;
                    }
             }
        }

        //Actualizamos unidades a las que sean las actuales
        public void actualizaUnidadesLabels() {
              //Cambiamos las unidades que indican las etiquetas a las actuales
            mma.setText("TARA ("+Unidades.getUnidadesCarga()+")");
            pma.setText("Masa Maxima Autorizada("+Unidades.getUnidadesCarga()+")");
            cargaejedelantero.setText("Carga máxima tándem
delantero("+Unidades.getUnidadesCarga()+")");
            cargaejetrasero.setText("Carga máxima tándem
trasero("+Unidades.getUnidadesCarga()+")");
            cargaejeremolque.setText("Carga máxima tándem
remolque("+Unidades.getUnidadesCarga()+")");
            largototal.setText("Longitud total("+Unidades.getUnidadesLongitud()+")");
```

```
            distfrenteprimereje.setText("Dist. Frente a tándem
delantero("+Unidades.getUnidadesLongitud()+")");
            entreejes.setText("Dist. Delantero a trasero("+Unidades.getUnidadesLongitud()
+")");
            comienzocaja.setText("Dist. Tandem delantero a comienzo
caja("+Unidades.getUnidadesLongitud()+")");
            distprimerejekingpin.setText("Dist. Eje delantero tractora a King
pin("+Unidades.getUnidadesLongitud()+")");
            distkingpintandemremolque.setText("Dist. King pin a tándem trasero
remolque("+Unidades.getUnidadesLongitud()+")");
        }

        //Abrimos la ventana de Base de Datos del camion y si no existe la creamos
        public void abreVentanaBaseDatosCamion() {
            if (panelBaseDatos!=null) {
                panelBaseDatos.actualiza();
                panelBaseDatos.setVisible(true);
            } else panelBaseDatos=new VentanaBaseDatosCamion(proyecto,this);
      }


      //Incluimos nuevos datos en la Base de Datos
      public void incluyeNuevosDatosCamion() {
            double PMA=0;
                double MMA=0;
                double PesoD=0;
                double PesoT=0;
                double PesoR=0;
                double distFrenteEjeDelantero=0;
                double distEjeDelanteroFrenteCaja=0;
                double distEjeDelanteroEjeTrasero=0;
                double largo=0;
                double distprimerEjeKingPin=0;
                double distKingPinEjeRemolque=0;
            //Comprobamos los resultados
            //El siguiente booleano recoge la respuesta de seguir o no ante
recomendaciones sobre los datos
            boolean seguir=true;
                ArrayList<JTextField> jtf = new ArrayList<JTextField>();
                jtf.add(mmatext);
                jtf.add(pmatext);
                jtf.add(cargaejedelanterotext);
                jtf.add(cargaejetraserotext);
                jtf.add(largotext);
                jtf.add(distdelanterotraserotext);
                jtf.add(frenteprimerejetext);
                jtf.add(comienzocajatext);
                //Si calculamos con un articulado metemos mas campos a comprobar
                boolean resultado=true;
                Camion c = proyecto.getCamion();
                if((c.getTipo()!=null)&&(c.getTipo().getTipo()=='A')) {
                    jtf.add(cargaejeremolquetext);
                    jtf.add(distprimerejekingpintext);
                    jtf.add(distkingpintandemremolquetext);
                } else if (c.getTipo()==null) {
                    resultado=false;
                    CheckDatos.okNokMensaje("Debe seleccionarse un tipo de
camión");
                }
                //Los paso a comprobar
                if (resultado==true)
resultado=resultado&&CheckDatos.compruebaDatosCompletos(jtf);
                if(resultado==true)
resultado=resultado&&CheckDatos.corrigeDatos(jtf);

                //Usamos un formateo de los datos
                DecimalFormat df = new DecimalFormat("0.00");
```

```
                    //Cogemos los valores.
                    //Lo incluimos en un try para evitar problemas de excepciones por el
formato de los datos
                    try {
                          if (resultado) {

PMA=Unidades.getConversionCarga()*Double.parseDouble(pmatext.getText());

MMA=Unidades.getConversionCarga()*Double.parseDouble(mmatext.getText());

PesoD=Unidades.getConversionCarga()*Double.parseDouble(cargaejedelanterotext.getText());

PesoT=Unidades.getConversionCarga()*Double.parseDouble(cargaejetraserotext.getText());
                                    PesoR=0;

distFrenteEjeDelantero=Unidades.getConversionLongitud()*Double.parseDouble(frenteprimerej
etext.getText());

distEjeDelanteroFrenteCaja=Unidades.getConversionLongitud()*Double.parseDouble(comienzoca
jatext.getText());

distEjeDelanteroEjeTrasero=Unidades.getConversionLongitud()*Double.parseDouble(distdelant
erotraserotext.getText());

largo=Unidades.getConversionLongitud()*Double.parseDouble(largotext.getText());
                                    distprimerEjeKingPin=0;
                                    distKingPinEjeRemolque=0;
                                    if(proyecto.getCamion().getTipo().getTipo()=='A') {

distprimerEjeKingPin=Unidades.getConversionLongitud()*Double.parseDouble(distprimerejekin
gpintext.getText());

distKingPinEjeRemolque=Unidades.getConversionLongitud()*Double.parseDouble(distkingpintan
demremolquetext.getText());

PesoR=Unidades.getConversionCarga()*Double.parseDouble(cargaejeremolquetext.getText());
                                    }
                                    //Damos avisos sobre los datos
                                    String aviso="";
                                    if (resultado==true) {

aviso=CheckDatos.recomendacionesEntradas(c,distFrenteEjeDelantero,distEjeDelanteroFrenteC
aja,distEjeDelanteroEjeTrasero,largo,distprimerEjeKingPin,
                                                      distKingPinEjeRemolque);
                                          //Si hay sugerencias las muestro y recojo si
quiero sequir
                                          if (!aviso.equals("RECOMENDACIONES:
\n\n¿Proseguir?")) {
                                                seguir=CheckDatos.okCancelaMensaje(aviso);
                                          }
                                    }
                                          //Pasamos a la ventana de calculo de cargas
                                    if(resultado&&seguir) {
                                          if(aviso.contains("Se corrige la distancia del
eje de remolque para que quede dentro de los límites del camión")) {
                                                distKingPinEjeRemolque=(largo-
distFrenteEjeDelantero-distprimerEjeKingPin);

distkingpintandemremolquetext.setText(Double.toString(distKingPinEjeRemolque));
                                          }
                                          if(!aviso.contains("Error")) {
                                                Camion camion=proyecto.getCamion();
                                                String
tipocam=String.valueOf(camion.getTipo().getTipo());
                                                int
ejesD=camion.getEjesPorTandemDelantero();
```

```
                                                int
ejesT=camion.getEjesPorTandemTrasero();
                                                int ejesR;
                                                if(camion.getTipo().getTipo()=='A') {
                                                      CamionArticulado
camiontipo=(CamionArticulado)camion;

ejesR=camiontipo.getEjesPorTandemRemolque();
                                                } else {
                                                      ejesR=0;
                                                }
                                          abreVentanaBaseDatosCamion();

panelBaseDatos.incluyeNuevosDatos(marcatext.getText(),modelotext.getText(),PMA,MMA,tipoca
m,PesoD,ejesD,PesoT,ejesT,PesoR,ejesR,distFrenteEjeDelantero,

distEjeDelanteroFrenteCaja,distEjeDelanteroEjeTrasero,distprimerEjeKingPin,distKingPinEje
Remolque,largo);
                                          daFormato();
                                          }
                                    }
                              }
                        } catch (Exception e) {
                        CheckDatos.okNokMensaje("Error en la exportación de los datos del
camión seleccionado");
                  }
      }

      //Este método da formato a los
      public void daFormato() {
            //Cogemos los valores
            mmatext.setText(mmatext.getText().replace('.',','));
            pmatext.setText(pmatext.getText().replace('.',','));

cargaejedelanterotext.setText(cargaejedelanterotext.getText().replace('.',','));

cargaejetraserotext.setText(cargaejetraserotext.getText().replace('.',','));

cargaejeremolquetext.setText(cargaejeremolquetext.getText().replace('.',','));
            largotext.setText(largotext.getText().replace('.',','));

frenteprimerejetext.setText(frenteprimerejetext.getText().replace('.',','));

distdelanterotraserotext.setText(distdelanterotraserotext.getText().replace('.',','));
            comienzocajatext.setText(comienzocajatext.getText().replace('.',','));

distprimerejekingpintext.setText(distprimerejekingpintext.getText().replace('.',','));

distkingpintandemremolquetext.setText(distkingpintandemremolquetext.getText().replace('.'
,','));
      }


      //Llama desde base de datos y convierte en texto los valores de seccion usada
      public void extraeDeTabla(String marca, String modelo,String PMA,String MMA,String
tipocam,String PesoD,String ejesD,String PesoT,String ejesT,String  PesoR,
                  String ejesR,String distFrenteEjeDelantero,String
distEjeDelanteroFrenteCaja,String distEjeDelanteroEjeTrasero,
                  String distPrimerEjeKingPin,String distKingPinEjeRemolque,String
largo) {

            ArrayList<String> s = new ArrayList<String>();
            s.add(PMA);
            s.add(MMA);
            s.add(PesoD);
            s.add(ejesD);
            s.add(PesoT);
```

```
            s.add(ejesT);
            s.add(PesoR);
            s.add(ejesR);
            s.add(distFrenteEjeDelantero);
            s.add(distEjeDelanteroFrenteCaja);
            s.add(distEjeDelanteroEjeTrasero);
            s.add(distPrimerEjeKingPin);
            s.add(distKingPinEjeRemolque);
            s.add(largo);

            //Informamos en el Area de que hemos procedido a cargar el camion
            incluyeInfoArea("Carga de camión desde BBDD",area);
            //Dejamos un formato decimal para los datos
            DecimalFormat df = new DecimalFormat("0.00");
            //Asignamos los valores pasados
            marcatext.setText(marca);
            modelotext.setText(modelo);
            //En funcion del tipo de camion hacemos click en su boton para ejecutar el
codigo correspondiente
            //Convertimos en char para ver la comparacion
            char tipo = tipocam.toCharArray()[0];
            //Vemos si es rigido o articulado (R o A)
            switch(tipo) {
                  case 'R':{
                        String ejesRig="";
                        ejesRig=ejesD+ejesT;
                        //Vemos los ejes que tiene para ver el boton que clickar
                        switch(ejesRig) {
                              case "11": r2ejes.doClick(); break;
                              case "12": r3ejes.doClick(); break;
                              case "22": r4ejes.doClick(); break;
                              default: break;
                        }
                  }; break;
                  case 'A':{
                        String ejesArt="";
                        ejesArt=ejesD+ejesT+ejesR;
                        //Vemos los ejes que tiene para ver el boton que clickar
                              switch(ejesArt) {
                                    case "112": a4ejes.doClick(); break;
                                    case "113": a5ejes23.doClick(); break;
                                    case "122": a5ejes32.doClick(); break;
                                    case "123": a6ejes.doClick(); break;
                                    default: break;
                              }
                  }; break;
                  default: break;
            }
            //Cogemos los valores
                  //Lo incluimos en un try para evitar problemas de excepciones por el
formato de los datos
            try {

mmatext.setText(df.format(Unidades.getCargaEnUnidadesActuales(Double.valueOf(MMA.replace(
',', '.')))));

pmatext.setText(df.format(Unidades.getCargaEnUnidadesActuales(Double.valueOf(PMA.replace(
',', '.')))));

cargaejedelanterotext.setText(df.format(Unidades.getCargaEnUnidadesActuales(Double.valueO
f(PesoD.replace(',', '.')))));

cargaejetraserotext.setText(df.format(Unidades.getCargaEnUnidadesActuales(Double.valueOf(
PesoT.replace(',', '.')))));

cargaejeremolquetext.setText(df.format(Unidades.getCargaEnUnidadesActuales(Double.valueOf
(PesoR.replace(',', '.')))));
```

```
largotext.setText(df.format(Unidades.getLongitudEnUnidadesActuales(Double.valueOf(largo.r
eplace(',', '.')))));

frenteprimerejetext.setText(df.format(Unidades.getLongitudEnUnidadesActuales(Double.value
Of(distFrenteEjeDelantero.replace(',', '.')))));

distdelanterotraserotext.setText(df.format(Unidades.getLongitudEnUnidadesActuales(Double.
valueOf(distEjeDelanteroEjeTrasero.replace(',', '.')))));

comienzocajatext.setText(df.format(Unidades.getLongitudEnUnidadesActuales(Double.valueOf(
distEjeDelanteroFrenteCaja.replace(',', '.')))));

distprimerejekingpintext.setText(df.format(Unidades.getLongitudEnUnidadesActuales(Double.
valueOf(distPrimerEjeKingPin.replace(',', '.')))));

distkingpintandemremolquetext.setText(df.format(Unidades.getLongitudEnUnidadesActuales(Do
uble.valueOf(distKingPinEjeRemolque.replace(',', '.')))));
                    //Damos formato
                    //daFormato();
             } catch (Exception e) {
                    CheckDatos.okNokMensaje("Error en la carga de datos del camión
seleccionado");
             }
             //Volvemos a hacer esta ventana visible si no lo estaba.
             setVisible(true);
             //La repintamos
             repaint();
       }

        public void setUnidades(String longitud, String carga) {
             asignaDatosAProyecto();
             switch(longitud) {
             case "m": Unidades.m();
             break;
             case "dm": Unidades.dm();
             break;
             case "cm": Unidades.cm();
             break;
             case "mm": Unidades.mm();
             break;
             default: Unidades.mm();
             break;
             }
             switch(carga) {
             case "Kg": Unidades.Kg();
             break;
             case "N": Unidades.Nw();
             break;
             default: Unidades.Kg();
             break;
             }
             repaint();
             this.repaint();
             this.rellenaDatos();
        }

        public void guardaFichero() {
             try {
                    asignaDatosAProyecto();
                    Camion c = proyecto.getCamion();
                    c.setCargas(proyecto.getCargas());
                    fichero.guarda(proyecto.getCamion());
             } catch (Exception e) {
             }
        }
```

```
        public void abreFichero() {
              try {
              if (ventCalc != null) ventCalc.dispose();
              } catch(Exception e) {
                    CheckDatos.okNokMensaje(e.toString());
              }
              try{
                    LeyesCortantes.setProyecto(proyecto);
                    LeyesMomentos.setProyecto(proyecto);
                     Object c=(Camion)fichero.abreFichero();

                     if(((Camion)c).getTipo().getTipo()=='R') {
                           CamionRigido cr=(CamionRigido)c;

                           //Asignamos el camion y las cargas al proyecto.
                           proyecto.setCamion((CamionRigido)cr);
                            proyecto.setCargas(((Camion)cr).getCargas());

                            //Calculamos las reacciones.
                           CalculoReaccionesRigido.sumareacciones(proyecto);

                     } else {
                           CamionArticulado ca=(CamionArticulado)c;

                           //Asignamos camion y cargas al proyecto.
                            proyecto.setCamion((CamionArticulado)ca);
                            proyecto.setCargas(((Camion)ca).getCargas());


                           //Calculamos Reacciones en Articulado
                            CalculoReaccionesArticulado.sumareaccionesCabeza(proyecto);
                            CalculoReaccionesArticulado.sumareaccionesRemolque(proyecto);

                     }
                     ventCalc=new VentanaCalculos(proyecto,this);
                     ventCalc.reseteaVentanas();
                     ventCalc.setVisible(false);
                     try {
                     rellenaDatos();
                     } catch (Exception e) {
                           CheckDatos.okCancelaMensaje("Se ha producido un error en el
rellenado de datos. ");
                     }
                     repaint();
              } catch (Exception e2) {
                    CheckDatos.okCancelaMensaje("No se ha podido cargar ningún
fichero.");
              }
        }

        //Este metodo abre el dialogo para crear el PDF de resultados
        public void abreDialogoConstructorPDF() {
                    vinp=new VentanaIntroduceNombrePDF(this);
        }

        //Este metodo prepara la informacion tecnica para pasarla al PDF
        public void llamaAPdf(String n) {
              //Recogemos el nombre del fichero
                    String nombre = n;

                    //Preparamos la ventana con la informacion
                    calculaCargas();
                    ventCalc.setVisible(true);

                    //Preparamos el panelEsquema
                    PanelEsquema pe = ventCalc.getEsquema();
```

```
                    pe.setImagen(DatosGenericos.getDireccion("Imagenes/fondos/fondo-
blanco.jpg"));

pe.setBorder(javax.swing.BorderFactory.createLineBorder(Color.BLACK));
                    pe.setVisible(true);
                    BufferedImage imageE = new BufferedImage(pe.getWidth(),
pe.getHeight(), BufferedImage.TYPE_INT_RGB);
            Graphics2D graphics2De = imageE.createGraphics();
            pe.paint(graphics2De);

                    //Preparamos el panelCortantes
                    PanelCortantes pc = ventCalc.getCortantes();
                    pc.setImagen(DatosGenericos.getDireccion("Imagenes/fondos/fondo-
blanco.jpg"));

pc.setBorder(javax.swing.BorderFactory.createLineBorder(Color.BLACK));
                    pc.setVisible(true);
                    BufferedImage imageC = new BufferedImage(pc.getWidth(),
pc.getHeight(), BufferedImage.TYPE_INT_RGB);
            Graphics2D graphics2Dc = imageC.createGraphics();
            pc.paint(graphics2Dc);

                    //Preparamos el panelMomentos
                    PanelMomentos pm = ventCalc.getMomentos();
                    pm.setImagen(DatosGenericos.getDireccion("Imagenes/fondos/fondo-
blanco.jpg"));

pm.setBorder(javax.swing.BorderFactory.createLineBorder(Color.BLACK));
                    pm.setVisible(true);
                    BufferedImage imageM = new BufferedImage(pm.getWidth(),
pm.getHeight(), BufferedImage.TYPE_INT_RGB);
            Graphics2D graphics2Dm = imageM.createGraphics();
            pm.paint(graphics2Dm);

            ArrayList<BufferedImage> albi = new ArrayList<BufferedImage>();
            albi.add(imageE);
            albi.add(imageC);
            albi.add(imageM);

            //Si el camion es articulado necesitamos tambien cambiar al calculo del
remolque
            if(proyecto.getCamion().getTipo().getTipo()=='A') {

             ventCalc.usarParte(false);

                        //Preparamos el panelEsquema del Remolque
                        PanelEsquema peR = ventCalc.getEsquema();
                        peR.setImagen(DatosGenericos.getDireccion("Imagenes/fondos/
fondo-blanco.jpg"));

peR.setBorder(javax.swing.BorderFactory.createLineBorder(Color.BLACK));
                        peR.setVisible(true);
                        BufferedImage imageER = new BufferedImage(peR.getWidth(),
peR.getHeight(), BufferedImage.TYPE_INT_RGB);
                  Graphics2D graphics2DeR = imageER.createGraphics();
                  peR.paint(graphics2DeR);

                        //Preparamos el panelCortantes del Remolque
                        PanelCortantes pcR = ventCalc.getCortantes();
                        pcR.setImagen(DatosGenericos.getDireccion("Imagenes/fondos/
fondo-blanco.jpg"));

pcR.setBorder(javax.swing.BorderFactory.createLineBorder(Color.BLACK));
                        pcR.setVisible(true);
                        BufferedImage imageCR = new BufferedImage(pcR.getWidth(),
pcR.getHeight(), BufferedImage.TYPE_INT_RGB);
                  Graphics2D graphics2DcR = imageCR.createGraphics();
```

```
                    pcR.paint(graphics2DcR);

                            //Preparamos el panelMomentos del Remolque
                            PanelMomentos pmR = ventCalc.getMomentos();
                            pmR.setImagen(DatosGenericos.getDireccion("Imagenes/fondos/
fondo-blanco.jpg"));

pmR.setBorder(javax.swing.BorderFactory.createLineBorder(Color.BLACK));
                            pmR.setVisible(true);
                            BufferedImage imageMR = new BufferedImage(pmR.getWidth(),
pmR.getHeight(), BufferedImage.TYPE_INT_RGB);
                    Graphics2D graphics2DmR = imageMR.createGraphics();
                    pmR.paint(graphics2DmR);

                    albi.add(imageER);
                    albi.add(imageCR);
                    albi.add(imageMR);

            }

            //Preparamos otro ArrayList con las tablas de reacciones
            ArrayList<BufferedImage> albi2 = new ArrayList<BufferedImage>();

                     //Preparamos la lista de Cargas
                     PanelConImagen lc = ventCalc.getListaCargas();
                     lc.setImagen(DatosGenericos.getDireccion("Imagenes/fondos/fondo-
blanco.jpg"));

lc.setBorder(javax.swing.BorderFactory.createLineBorder(Color.BLACK));
                     lc.setVisible(true);
                     BufferedImage imageLC = new BufferedImage(lc.getWidth(),
lc.getHeight(), BufferedImage.TYPE_INT_RGB);
            Graphics2D graphics2DLC = imageLC.createGraphics();
            lc.paint(graphics2DLC);

                     //Preparamos las Reacciones
                     PanelConImagen pr = ventCalc.getReacciones();
                     pr.setImagen(DatosGenericos.getDireccion("Imagenes/fondos/fondo-
blanco.jpg"));

pr.setBorder(javax.swing.BorderFactory.createLineBorder(Color.BLACK));
                     pr.setVisible(true);
                     BufferedImage imagePR = new BufferedImage(pr.getWidth(),
pr.getHeight(), BufferedImage.TYPE_INT_RGB);
            Graphics2D graphics2DPR = imagePR.createGraphics();
            pr.paint(graphics2DPR);

            //Metemos las imagenes
            albi2.add(imageLC);
            albi2.add(imagePR);


            try {
                   //Mandamos el nombre a CreaPDF
                    CreaPDF.setNombrePDF(nombre);

                    //Mandamos la imagen
                    CreaPDF.setImages(albi);

                    //Mandamos las tablas en imagen
                    CreaPDF.setImages2(albi2);

                    //Creamos el PDF
                    CreaPDF.creaResultadosPdf();

                    //Ocultamos las ventanas usadas
                    ventCalc.setVisible(false);
```

```
            } catch (Exception exc) {
             CheckDatos.okCancelaMensaje("Error al crear el PDF.
"+exc.getStackTrace().toString());
            }
        }

        //Este metodo prepara las imagenes de los paneles
        public void preparaInformacionParaPDF(JPanel p) {

        }
}
```

C.2. Paquete secciones

Se expone a continuación el código de las clases del paquete secciones que contiene la codificación para la gestión y creación de objetos de tipo sección.

CLASE CalculoSeccion

```
package secciones;

import java.io.Serializable;
import java.text.DecimalFormat;

import auxiliares.Unidades;

//Calcula el momento de inercia y el Area de una seccion tipo C o I
//secin es la sección rectangular inferior (rectangulo horizontal)
//secmed es la sección rectangular media (rectangulo horizontal)
//secsup es la sección rectangular superior (rectangulo horizontal)
public class CalculoSeccion implements Serializable{

          private static final long serialVersionUID = 1L;
          //Definimos secin como el rectangulo inferior de la sección en C o I, secmed el
intermedio y secsup el superior
          private SeccionRectangular secin,secmed,secsup;
          private double baseinf,altoinf,basemed,altomed,basesup,altosup; //Medidas de
secciones (primero dimension horizontal del rectangulo y luego vertical)
          //Definimos la inercia de la sección en C
          private double area,inercia,moduloResistente;
          //Usamos una variable para calcular la altura del eje Y neutro.
          private double y_ejeneutro;
          //Guardamos las alturas de los ejes neutros de cada seccion rectangular ya que
las usaremos
          private double alturaNeutroInf, alturaNeutroMed, alturaNeutroSup;
          private String informacion="";

          public CalculoSeccion(double baseinf, double altoinf, double basemed,double
altomed,double basesup,double altosup,char tipo){
                //Inicializamos los 3 rectángulos de la seccion
               //Actualizamos las variables de clase y uso las unidades por defecto para
el calculo que son milimetros.
                     this.baseinf=Unidades.getConversionLongitud()*baseinf;
                     this.altoinf=Unidades.getConversionLongitud()*altoinf;
                     //Como el rectangulo intermedio de la seccion esta vertical en vez
de en horizontal cambio el orden de los valores.
                     this.basemed=Unidades.getConversionLongitud()*altomed;
                     this.altomed=Unidades.getConversionLongitud()*basemed;
                     this.basesup=Unidades.getConversionLongitud()*basesup;
                     this.altosup=Unidades.getConversionLongitud()*altosup;
                     //Completamos la informacion de referencia de ayuda y calculamos
propiedades de cada rectangulo de la sección
                     //inferior:
                completaInfo("rectangulo inferior:");
                this.secin = new SeccionRectangular(this.baseinf,this.altoinf);
                completaInfo(this.secin.getInfo());
                //medio
                completaInfo("rectangulo medio:");
                this.secmed = new SeccionRectangular(this.basemed,this.altomed);
                completaInfo(this.secmed.getInfo());
                //superior
                completaInfo("rectangulo superior:");
                this.secsup = new SeccionRectangular(this.basesup,this.altosup);
                completaInfo(this.secsup.getInfo());
                //Calculamos las propiedades de la sección compuesta
                calculaArea(); //Calcula el Area
                calculaAlturasEjesNeutros(); //Calcula la cota y del eje neutro de cada
rectangulo de la seccion para calcular el eje neutro
                calculaPosicionEjeNeutro(); //Calcula la posicion del eje neutro de la
seccion C completa
                calculaInercia(); //Calculamos la inercia de la sección compuesta
                //Completamos la información para el area de ayuda
                DecimalFormat df = new DecimalFormat("0.00");
                completaInfo("*****************************************************");
```

```
                completaInfo("Área general sección compuesta:
"+df.format(Unidades.getAreaEnUnidadesActuales(this.getArea()))
+Unidades.getUnidadesArea());
                completaInfo("Inercia sección compuesta:
"+df.format(Unidades.getInerciaEnUnidadesActuales(this.getInercia()))
+Unidades.getUnidadesInercia());
                completaInfo("Módulo resistente sección compuesta:
"+df.format(Unidades.getModuloResistenteEnUnidadesActuales(this.getModulo()))
+Unidades.getUnidadesModuloResistente());
          }

          //Se calcula la altura desde la base inferior del eje neutro
          private void calculaPosicionEjeNeutro() {
                    //Calcula el sumatorio de cada Area de cada rectángulo por la
altura a sus respectivos ejes neutros.
                    double sumaperfiles = this.alturaNeutroInf*this.secin.getArea()
+this.alturaNeutroMed*this.secmed.getArea()+this.alturaNeutroSup*this.secsup.getArea();
                    //Obtenemos el eje neutro del perfil C completo
                    this.y_ejeneutro=sumaperfiles/this.getArea();
          }

          //Calculamos el area de la seccion compuesta
          private void calculaArea() {
                    this.area = secin.getArea()+secmed.getArea()+secsup.getArea();
          }

          //Calcula la altura del eje Neutro de cada rectangulo
          private void calculaAlturasEjesNeutros() {
                    //Calculamos la altura del eje neutro del rectangulo en la base
(inferior)
                    this.alturaNeutroInf = this.secin.getAlto()/2;
                    //Calculamos la altura del eje neutro del rectangulo intermedio
(medio)
                    this.alturaNeutroMed = this.secin.getAlto()+this.secmed.getAlto()/
2;
                    //Calculamos la altura del eje neutro del rectangulo más elevado
(superior)
                    this.alturaNeutroSup = this.secin.getAlto()+this.secmed.getAlto()
+this.secsup.getAlto()/2;
          }

          //Calcula la Inercia sobre el eje neutro de la sección completa
          private void calculaInercia() {
                    double inerciaInf = this.secin.getInercia()
+this.secin.getArea()*Math.pow(this.alturaNeutroInf, 2);
                    double inerciaMed = this.secmed.getInercia()
+this.secmed.getArea()*Math.pow(this.alturaNeutroMed, 2);
                    double inerciaSup = this.secsup.getInercia()
+this.secsup.getArea()*Math.pow(this.alturaNeutroSup, 2);
                    double inerciaCEjeBase = inerciaInf+inerciaMed+inerciaSup;
                    this.inercia = inerciaCEjeBase-this.area*Math.pow(this.y_ejeneutro,
2);
                    this.moduloResistente=this.getInercia()/this.getAlturaMedia();
          }

          //Ponemos los getters
          //area
          public double getArea() {
               return this.area;
          }
          //Inercia
          public double getInercia() {
              return this.inercia;
          }
          //Modulo
          public double getModulo() {
              return this.moduloResistente;
```

```
	}

	//Generamos metodos para la gestión de la información del area de ayuda
	public void completaInfo(String s) {
		this.informacion=this.informacion+"\n"+s;
	}
	//Devuelve la información
	public String devuelveInfo() {
		return ("RESULTADOS:\n")+this.informacion;
	}
	//Devuelve altura media
	public double getAlturaMedia() {
		return (this.altoinf+this.basemed+this.altosup)/2;
	}
	//Devuelve el modulo
	public double getModuloResistente() {
		return this.moduloResistente;
	}
	//Devuelve la base del rectangulo inferior
	public double getBaseInf() {
		return this.baseinf;
	}
	//Devuelve la base del rectangulo medio
	public double getBaseMed() {
		//Como este rectangulo esta en vertical y hemos cambiado el orden de los
valores se devuelve el de la altura
		return this.altomed;
	}
	//Devuelve la base del rectangulo superior
	public double getBaseSup() {
		return this.basesup;
	}
	//Devuelve la altura del rectangulo inferior
	public double getAltoInf() {
		return this.altoinf;
	}
	//Devuelve la altura del rectangulo medio
	public double getAltoMed() {
		//Como este rectangulo esta en vertical y hemos cambiado el orden de los
valores se devuelve el de la base
		return this.basemed;
	}
	//Devuelve la altura del rectangulo superior
	public double getAltoSup() {
		return this.altosup;
	}

}
```

INTERFACE Seccion

```
package secciones;

public interface Seccion{
            public double calculaArea();
            public double calculaInercia();
}
```

CLASE SeccionRectangular

```
package secciones;

import java.io.Serializable;
import java.text.DecimalFormat;

import auxiliares.Unidades;

//Esta clase calcula propiedades físicas de una seccion rectangular. Se usa como una
clase auxiliar para los calculos de secciones
public class SeccionRectangular implements Seccion, Serializable {
          double base;
          double altura;
          double area;
          double inercia;
          //Generamos una variable para almacenar la información en el area de Texto"
          String infoSecRec="";

          public SeccionRectangular(double ancho, double alto) {
                    this.base = ancho;
                    this.altura = alto;
                    //Completamos la informacion de la seccion rectangular
                    //Formateamos los numeros para que tengan 2 decimales
                    DecimalFormat df = new DecimalFormat("0.00");
                    //Pasamos a la variable infoSecRec la información a presentar en
pantalla
                    infoSecRec=infoSecRec+"Base:
"+df.format(Unidades.getLongitudEnUnidadesActuales(this.base))
                    +"\nAltura:
"+df.format(Unidades.getLongitudEnUnidadesActuales(this.altura))
                    +"\nÁrea:
"+df.format(Unidades.getAreaEnUnidadesActuales(calculaArea()))
                    +"\nInercia:
"+df.format(Unidades.getInerciaEnUnidadesActuales(calculaInercia()))
                    +"\n";
          }
          @Override
          public double calculaArea() {
                    this.area=base*altura;
                    return this.area;
          }

          @Override
          public double calculaInercia() {
                    this.inercia=base*Math.pow(altura,3)/12;
                    return this.inercia;
          }

          public double getInercia() {
                    return inercia;
          }

          public double getArea() {
                    return area;
          }

          public double getAlto() {
                    return altura;
          }

          public double getbase() {
                    return base;
          }
          public String getInfo() {
               return this.infoSecRec;
          }
```

```
}
```

CLASE VentanaSeccion

```
package secciones;

import java.awt.*;
import javax.swing.*;
import javax.swing.border.LineBorder;
import camiones.*;

import auxiliares.CheckDatos;
import auxiliares.DatosGenericos;
import auxiliares.Formato;
import auxiliares.PanelConImagen;
import auxiliares.Unidades;
import basesDatos.VentanaBaseDatosSeccion;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Iterator;

import static java.awt.GridBagConstraints.*;


@SuppressWarnings("unused")
public class VentanaSeccion extends JFrame{

	private static final long serialVersionUID = -7418029428404409927L;
	//Definimos los valores comunes para todos los componentes y elementos
	private static int ladopanel=300;
	private static String fondogris=DatosGenericos.getDireccion("Imagenes/fondos/
fondo-gris.png");
	private static String fondoazul=DatosGenericos.getDireccion("Imagenes/fondos/
fondo.jpg");
	private Image fondo=new ImageIcon(getClass().getResource(fondogris)).getImage();
	private static ArrayList<Component> c = new ArrayList<Component>();
	private static ArrayList<JTextField> t = new ArrayList<JTextField>();
	private PanelConImagen general,dibujo,infoPanel,panelMarca,datosPanel; //Generamos
los paneles contenedores de los datos, el dibujo y la info
	private JLabel lbll1,lble1,lbll2,lble2,lbll3,lble3; //Generamos las etiquetas
	private JTextField tfl1,tfe1,tfl2,tfe2,tfl3,tfe3; //Generamos los campos de texto
para recogida de los datos
	private JTextArea info; //Generamos el area de texto para los calculos
	private JButton
calcula,calculaI,asignaACamion,asignaARemolque,guardaEnBaseDatos; //Generamos el boton
para calcular y los botones de asignación a camion y remolque y almacenamiento en Base de
Datos
	private static int dimension=800; //Generamos una dimension de ventana para
cambiarla si nos interesa
	private double L1,E1,L2,E2,L3,E3; //Generamos las variables que contendran los
datos para el calculo
	private GridBagConstraints constraints; //Generamos las restriccioones del
GridBagLayout
	private static Proyecto proyecto;
	private static VentanaBaseDatosSeccion panelBaseDatos;


	public VentanaSeccion(Proyecto p) {
		super("Propiedades Sección");
		this.setResizable(false);
		proyecto=p;

		//Preparamos los PANELES
		//datos panel contiene los elementos de entrada de datos
```

```
            datosPanel=new PanelConImagen(fondogris);
            //general contiene los datos de interaccion
            general=new PanelConImagen(fondoazul);
            //Infopanel contiene los datos de informacion de ingenieria
            infoPanel = new PanelConImagen(fondogris);
            //dibujo contiene el panel donde se dibuja la seccion
            dibujo=new PanelConImagen(fondogris);
            panelMarca=new PanelConImagen(DatosGenericos.getDireccion("Imagenes/
pantallaprincipal/recurso.png")); //contiene el logo de la aplicación
            panelMarca.setPreferredSize(new Dimension(150,80));


            //Damos formato al PANEL GENERAL
            SpringLayout layout = new SpringLayout();
            SpringLayout layoutDatosPanel = new SpringLayout();
            datosPanel.setPreferredSize(new Dimension(ladopanel-40,(ladopanel-40)));
            //Damos fomato
            Formato.setBorder(datosPanel);
            general.setPreferredSize(new Dimension(ladopanel,ladopanel*2));
            //Damos formato
            Formato.setBorder(general);
        int gapX=20;
        int gapY=20;


            // Preparamos las ETIQUETAS
            lbll1 = new JLabel("Longitud inferior");
            lble1 = new JLabel("Espesor inferior");
            lble2 = new JLabel("Espesor medio");
            lbll2 = new JLabel("Longitud media");
            lbll3 = new JLabel("Longitud superior");
            lble3 = new JLabel("Espesor superior");


            //Preparamos los TEXTOS
            tfl1=new JTextField();
            tfl1.setPreferredSize(new Dimension(100,20));
            Formato.setBorder(tfl1);
            tfe1=new JTextField();
            tfe1.setPreferredSize(new Dimension(100,20));
            Formato.setBorder(tfe1);
            tfl2=new JTextField();
            tfl2.setPreferredSize(new Dimension(100,20));
            Formato.setBorder(tfl2);
            tfe2=new JTextField();
            tfe2.setPreferredSize(new Dimension(100,20));
            Formato.setBorder(tfe2);
            tfl3=new JTextField();
            tfl3.setPreferredSize(new Dimension(100,20));
            Formato.setBorder(tfl3);
            tfe3=new JTextField();
            tfe3.setPreferredSize(new Dimension(100,20));
            Formato.setBorder(tfe3);

            // Si ya se han introducido datos los metemos en los valores los metidos.
            if(proyecto.getCamion().getSeccion()!=null) {
                  DecimalFormat df = new DecimalFormat("0.00");
                  CalculoSeccion cs=proyecto.getCamion().getSeccion();

tfl1.setText(df.format(Unidades.getLongitudEnUnidadesActuales(cs.getBaseInf())));

tfe1.setText(df.format(Unidades.getLongitudEnUnidadesActuales(cs.getAltoInf())));

tfl2.setText(df.format(Unidades.getLongitudEnUnidadesActuales(cs.getBaseMed())));

tfe2.setText(df.format(Unidades.getLongitudEnUnidadesActuales(cs.getAltoMed())));
```

```
tfl3.setText(df.format(Unidades.getLongitudEnUnidadesActuales(cs.getBaseSup())));

tfe3.setText(df.format(Unidades.getLongitudEnUnidadesActuales(cs.getAltoSup())));
            }

            //Preparamos los BOTONES DE CALCULO
            //Generamos los botones
            calcula = new JButton("Muestra tipo C");
            calcula.setPreferredSize(new Dimension(200,30));
            calculaI = new JButton("Muestra tipo I");
            calculaI.setPreferredSize(new Dimension(200,30));
            asignaACamion = new JButton("");
            asignaACamion.setPreferredSize(new Dimension(200,30));
            asignaARemolque = new JButton("Asigna viga a remolque");
            asignaARemolque.setPreferredSize(new Dimension(200,30));
            if(proyecto.getCamion().getTipo().getTipo()=='R') {
                  asignaACamion.setText("Asigna viga a camión");
                  asignaARemolque.setVisible(false);
            }
            if(proyecto.getCamion().getTipo().getTipo()=='A') {
                  asignaACamion.setText("Asigna viga a cabeza");
                  asignaARemolque.setVisible(true);
            }
            guardaEnBaseDatos= new JButton("Guardar en base datos");
            guardaEnBaseDatos.setPreferredSize(new Dimension(200,30));


            //Generamos los JMENU
            /* Creamos el JMenuBar y lo asociamos con el JFrame */
            JMenuBar menuBar=new JMenuBar();
            setJMenuBar(menuBar);
            /* Creamos el primer JMenu y lo pasamos como parámetro al JMenuBar mediante
el método add */
            JMenu menuBBDD=new JMenu("Bases datos");
            menuBar.add(menuBBDD);
            Formato.setFuenteMenu(menuBBDD);
            /* Creamos JMenuItem */
            JMenuItem menuImportarSeccion=new JMenuItem("Importar sección de base de
datos");
            menuBBDD.add(menuImportarSeccion);
            Formato.setFuenteMenuItem(menuImportarSeccion);


            //Generamos los ActionListeners
        ActionListener accionbtncalcula=new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                  //Obtenemos los valores introducidos
                  //Metemos todas las etiquetas en un ArrayList para comprobar y que
tengan formato adecuado
                  boolean resultadosOK=true;
                  t.clear();
                  t.add(tfe1);
                  t.add(tfe2);
                  t.add(tfe3);
                  t.add(tfl1);
                  t.add(tfl2);
                  t.add(tfl3);
                  resultadosOK=CheckDatos.corrigeDatos(t);
                  if(resultadosOK) resultadosOK=CheckDatos.compruebaDatosCompletos(t);
                  if(resultadosOK) {
                        //Metemos los valores
                        L1=Double.parseDouble(tfl1.getText());
                        E1=Double.parseDouble(tfe1.getText());
                        L2=Double.parseDouble(tfl2.getText());
                        E2=Double.parseDouble(tfe2.getText());
                        L3=Double.parseDouble(tfl3.getText());
```

```
                        E3=Double.parseDouble(tfe3.getText());
                        //Imprimimos los datos del calculo de la seccion en el area de
texto
                        CalculoSeccion calculo = new
CalculoSeccion(L1,E1,L2,E2,L3,E3,'C');
                        info.setText(calculo.devuelveInfo());
                        //dibujamos la seccion con el metodo dibuja pasando una C para
que sepa que es esta seccion
                        dibuja('C');
                  }
                  }
      };
       ActionListener accionbtncalculaI=new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                  //Obtenemos los valores introducidos
                  //Metemos todas las etiquetas en un ArrayList para comprobar y que
tengan formato adecuado
                  boolean resultadosOK=true;
                  t.clear();
                  t.add(tfe1);
                  t.add(tfe2);
                  t.add(tfe3);
                  t.add(tfl1);
                  t.add(tfl2);
                  t.add(tfl3);
                  resultadosOK=CheckDatos.corrigeDatos(t);
                  if(resultadosOK) resultadosOK=CheckDatos.compruebaDatosCompletos(t);
                  if(resultadosOK) {
                        //Metemos los valores
                        L1=Double.parseDouble(tfl1.getText());
                        E1=Double.parseDouble(tfe1.getText());
                        L2=Double.parseDouble(tfl2.getText());
                        E2=Double.parseDouble(tfe2.getText());
                        L3=Double.parseDouble(tfl3.getText());
                        E3=Double.parseDouble(tfe3.getText());
                        //Imprimimos los datos del calculo de la seccion en el area de
texto
                        CalculoSeccion calculo = new
CalculoSeccion(L1,E1,L2,E2,L3,E3,'I');
                        info.setText(calculo.devuelveInfo());
                        //dibujamos la seccion con el metodo dibuja pasando una I para
que sepa que es esta seccion
                        dibuja('I');
                  }
                  }
      };
      ActionListener accionbtnasignaACamion=new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                  //Obtenemos los valores introducidos
                  //Metemos todas las etiquetas en un ArrayList para comprobar y que
tengan formato adecuado
                  boolean resultadosOK=true;
                  t.clear();
                  t.add(tfe1);
                  t.add(tfe2);
                  t.add(tfe3);
                  t.add(tfl1);
                  t.add(tfl2);
                  t.add(tfl3);
                  resultadosOK=CheckDatos.corrigeDatos(t);
                  if(resultadosOK) resultadosOK=CheckDatos.compruebaDatosCompletos(t);
                  if(resultadosOK) {
                        //Metemos los valores
                        L1=Double.parseDouble(tfl1.getText());
                        E1=Double.parseDouble(tfe1.getText());
                        L2=Double.parseDouble(tfl2.getText());
                        E2=Double.parseDouble(tfe2.getText());
```

```
                        L3=Double.parseDouble(tfl3.getText());
                        E3=Double.parseDouble(tfe3.getText());
                        //Imprimimos los datos del calculo de la seccion en el area de
texto
                        CalculoSeccion calculo = new
CalculoSeccion(L1,E1,L2,E2,L3,E3,'I');
                        Camion c=proyecto.getCamion();
                        c.setSeccion(calculo);
                        c.setAreaViga(calculo.getArea());
                        c.setInerciaViga(calculo.getInercia());
                        DecimalFormat df = new DecimalFormat("0.00");
                        CheckDatos.okNokMensaje("Propiedades asignadas
(Área="+df.format(calculo.getArea())+", Inercia="+df.format(calculo.getInercia())+")");
                  }
                  }
      };
      ActionListener accionbtnasignaARemolque=new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                  //Obtenemos los valores introducidos
                  //Metemos todas las etiquetas en un ArrayList para comprobar y que
tengan formato adecuado
                  boolean resultadosOK=true;
                  t.clear();
                  t.add(tfe1);
                  t.add(tfe2);
                  t.add(tfe3);
                  t.add(tfl1);
                  t.add(tfl2);
                  t.add(tfl3);
                  resultadosOK=CheckDatos.corrigeDatos(t);
                  if(resultadosOK) resultadosOK=CheckDatos.compruebaDatosCompletos(t);
                  if(resultadosOK) {
                        L1=Double.parseDouble(tfl1.getText());
                        E1=Double.parseDouble(tfe1.getText());
                        L2=Double.parseDouble(tfl2.getText());
                        E2=Double.parseDouble(tfe2.getText());
                        L3=Double.parseDouble(tfl3.getText());
                        E3=Double.parseDouble(tfe3.getText());
                        //Imprimimos los datos del calculo de la seccion en el area de
texto
                        CalculoSeccion calculo = new
CalculoSeccion(L1,E1,L2,E2,L3,E3,'I');
                        //No hace falta que comprobemos si es articulado porque el
boton solo sale si lo es.
                        CamionArticulado c=(CamionArticulado)proyecto.getCamion();
                        c.setAreaVigaRemolque(calculo.getArea());
                        c.setInerciaVigaRemolque(calculo.getInercia());
                        DecimalFormat df = new DecimalFormat("0.00");
                        CheckDatos.okNokMensaje("Propiedades asignadas
(Área="+df.format(calculo.getArea())+", Inercia="+df.format(calculo.getInercia())+")");
                  }
                  }
      };
      ActionListener accionImportadeBBDD=new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                  abreVentanaBaseDatosSeccion();
                  }
      };

      ActionListener accionGuardaEnBBDD=new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                  incluyeNuevosDatosSeccion();
                  }
      };
      //Asignamos el actionListener a los botones
      calcula.addActionListener(accionbtncalcula);
      calculaI.addActionListener(accionbtncalculaI);
```

```
        asignaACamion.addActionListener(accionbtnasignaACamion);
        asignaARemolque.addActionListener(accionbtnasignaARemolque);
        menuImportarSeccion.addActionListener(accionImportadeBBDD);
        guardaEnBaseDatos.addActionListener(accionGuardaEnBBDD);


        //Marcamos las RESTRICCIONES DE COLOCACION SPRINGLAYOUT EN EL PANEL
        //Incluimos los elementos en los paneles las restricciones en X
        layoutDatosPanel.putConstraint(SpringLayout.WEST, lbll1, gapX, SpringLayout.WEST,
datosPanel);
        layoutDatosPanel.putConstraint(SpringLayout.WEST, lble1, gapX, SpringLayout.WEST,
datosPanel);
        layoutDatosPanel.putConstraint(SpringLayout.WEST, lbll2, gapX, SpringLayout.WEST,
datosPanel);
        layoutDatosPanel.putConstraint(SpringLayout.WEST, lble2, gapX, SpringLayout.WEST,
datosPanel);
        layoutDatosPanel.putConstraint(SpringLayout.WEST, lbll3, gapX, SpringLayout.WEST,
datosPanel);
        layoutDatosPanel.putConstraint(SpringLayout.WEST, lble3, gapX, SpringLayout.WEST,
datosPanel);
        layoutDatosPanel.putConstraint(SpringLayout.WEST, tfl1, gapX*7, SpringLayout.WEST,
datosPanel);
        layoutDatosPanel.putConstraint(SpringLayout.WEST, tfe1, gapX*7, SpringLayout.WEST,
datosPanel);
        layoutDatosPanel.putConstraint(SpringLayout.WEST, tfl2, gapX*7, SpringLayout.WEST,
datosPanel);
        layoutDatosPanel.putConstraint(SpringLayout.WEST, tfe2, gapX*7, SpringLayout.WEST,
datosPanel);
        layoutDatosPanel.putConstraint(SpringLayout.WEST, tfl3, gapX*7, SpringLayout.WEST,
datosPanel);
        layoutDatosPanel.putConstraint(SpringLayout.WEST, tfe3, gapX*7, SpringLayout.WEST,
datosPanel);
        layout.putConstraint(SpringLayout.WEST, panelMarca, (gapX*4), SpringLayout.WEST,
general);
        layout.putConstraint(SpringLayout.WEST,
datosPanel,gapX,SpringLayout.WEST,general);
        layout.putConstraint(SpringLayout.WEST, calcula, gapX-5, SpringLayout.WEST,
general);
        layout.putConstraint(SpringLayout.WEST, calculaI, gapX-5, SpringLayout.WEST,
general);
        layout.putConstraint(SpringLayout.WEST, asignaACamion, gapX-5, SpringLayout.WEST,
general);
        layout.putConstraint(SpringLayout.WEST, asignaARemolque, gapX-5,
SpringLayout.WEST, general);
        layout.putConstraint(SpringLayout.WEST, guardaEnBaseDatos, gapX-5,
SpringLayout.WEST, general);
        //Incluimos los elementos en los paneles las restricciones en Y
        layoutDatosPanel.putConstraint(SpringLayout.NORTH, lbll1, gapY,
SpringLayout.NORTH, datosPanel);
        layoutDatosPanel.putConstraint(SpringLayout.NORTH, lble1, gapY,
SpringLayout.SOUTH, lbll1);
        layoutDatosPanel.putConstraint(SpringLayout.NORTH, lbll2, gapY,
SpringLayout.SOUTH, lble1);
        layoutDatosPanel.putConstraint(SpringLayout.NORTH, lble2, gapY,
SpringLayout.SOUTH, lbll2);
        layoutDatosPanel.putConstraint(SpringLayout.NORTH, lbll3, gapY,
SpringLayout.SOUTH, lble2);
        layoutDatosPanel.putConstraint(SpringLayout.NORTH, lble3, gapY,
SpringLayout.SOUTH, lbll3);
        layoutDatosPanel.putConstraint(SpringLayout.NORTH, tfl1, gapY, SpringLayout.NORTH,
datosPanel);
        layoutDatosPanel.putConstraint(SpringLayout.NORTH, tfe1, gapY, SpringLayout.SOUTH,
lbll1);
        layoutDatosPanel.putConstraint(SpringLayout.NORTH, tfl2, gapY, SpringLayout.SOUTH,
lble1);
        layoutDatosPanel.putConstraint(SpringLayout.NORTH, tfe2, gapY, SpringLayout.SOUTH,
lbll2);
```

```
	layoutDatosPanel.putConstraint(SpringLayout.NORTH, tfl3, gapY, SpringLayout.SOUTH,
lble2);
	layoutDatosPanel.putConstraint(SpringLayout.NORTH, tfe3, gapY, SpringLayout.SOUTH,
lbll3);
	layout.putConstraint(SpringLayout.NORTH, panelMarca, gapY, SpringLayout.NORTH,
general);
	layout.putConstraint(SpringLayout.NORTH, datosPanel, gapY, SpringLayout.SOUTH,
panelMarca);
	layout.putConstraint(SpringLayout.NORTH, calcula, gapY-7, SpringLayout.SOUTH,
datosPanel);
	layout.putConstraint(SpringLayout.NORTH, calculaI, gapY-15, SpringLayout.SOUTH,
calcula);
	layout.putConstraint(SpringLayout.NORTH, asignaACamion, gapY-15,
SpringLayout.SOUTH, calculaI);
	layout.putConstraint(SpringLayout.NORTH, asignaARemolque, gapY-15,
SpringLayout.SOUTH, asignaACamion);
	layout.putConstraint(SpringLayout.NORTH, guardaEnBaseDatos, gapY-15,
SpringLayout.SOUTH, asignaARemolque);


	//Metemos los componentes en los PANELES GENERAL y de RECOGIDA DE INFORMACION
	datosPanel.setLayout(layoutDatosPanel);
	general.setLayout(layout);
	general.add(panelMarca);
	datosPanel.add(lbll1);
	datosPanel.add(lble1);
	datosPanel.add(lbll2);
	datosPanel.add(lble2);
	datosPanel.add(lbll3);
	datosPanel.add(lble3);
	datosPanel.add(tfl1);
	datosPanel.add(tfe1);
	datosPanel.add(tfl2);
	datosPanel.add(tfe2);
	datosPanel.add(tfl3);
	datosPanel.add(tfe3);
	general.add(datosPanel);
	general.add(calcula);
	general.add(calculaI);
	general.add(asignaACamion);
	general.add(asignaARemolque);
	general.add(guardaEnBaseDatos);


		//Generamos el PANEL DE INFORMACION
		infoPanel.setLayout(new GridLayout(2,0));
		//damos las dimensiones al panel
		infoPanel.setPreferredSize(new Dimension(ladopanel*2,ladopanel));
		//Introducimos el area de información en texto
		info=new JTextArea();
		Formato.setBordeEstandar(info);
		JScrollPane infoScroll = new
JScrollPane(info,JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,JScrollPane.HORIZONTAL_SCROLLBAR_A
LWAYS);


		//Introducimos el AREA DE DIBUJO
		Formato.setBorder(dibujo);
		//damos las dimensiones al panel
		dibujo.setPreferredSize(new Dimension(ladopanel,ladopanel));
		//Incluimos ambos paneles en el panel de informacion
		infoPanel.add(dibujo);
		infoPanel.add(infoScroll);
		infoPanel.setPreferredSize(new Dimension(ladopanel*2,ladopanel*2));


		//Incluimos todo en la VENTANA GENERAL.
```

```
            SpringLayout layout2 = new SpringLayout();
      //Incluimos los elementos en el panel las restricciones en X
      layout2.putConstraint(SpringLayout.WEST, general, 0, SpringLayout.WEST, this);
      layout2.putConstraint(SpringLayout.WEST, infoPanel, 0, SpringLayout.EAST,
general);
      //Incluimos los elementos en el panel las restricciones en Y
      layout2.putConstraint(SpringLayout.NORTH, general, 0, SpringLayout.NORTH, this);
      layout2.putConstraint(SpringLayout.NORTH, infoPanel, 0, SpringLayout.NORTH, this);
            setLayout(layout2);
            add(general);
            add(infoPanel);
            setPreferredSize(new Dimension(ladopanel*3,ladopanel*2+50));
            DatosGenericos.setPosicion(this, ladopanel*3, ladopanel*2+50);
            pack();
            setVisible(true);

            //damos formato a todos los componentes
            c.add(lbll1);
            c.add(lbll2);
            c.add(lbll3);
            c.add(lble1);
            c.add(lble2);
            c.add(lble3);
            c.add(calcula);
            c.add(calculaI);
            c.add(asignaACamion);
            c.add(asignaARemolque);
            c.add(guardaEnBaseDatos);
            c.add(info);
            darTipoLetra(c);
      }

   //Este método asigna a todos los componentes de la ventana el mismo tipo de texto
   private void darTipoLetra(ArrayList<Component> c) {
       Iterator<Component> iter = c.iterator();
       while(iter.hasNext()){
            ponLetra(iter.next());
       }
   }


   //Este método da el tipo letra a cada componente igual
   private void ponLetra(Component c) {
      Formato.setFormatoEstandar(c);
   }


   //Este metodo dibuja la viga que queremos calcular
      public void dibuja(char valor) {
            //Tomamos la medida del area a dibujar ancho y alto
            char tipoviga=valor;
            double ancho = dibujo.getSize().width;
            double alto = dibujo.getSize().height;

            //Medimos el area de nuestra viga
            double altoViga=this.E1+this.L2+this.E3;
            double anchoViga=Math.max(this.L3, Math.max(this.L1, this.E2));

            //Calculamos la escala para que ocupe más o menos la mitad del area
            double anchoPreferido=ancho/2;
            double altoPreferido=alto/2;
            double escalaAncho=anchoViga/anchoPreferido;
            double escalaAlto=altoViga/altoPreferido;
            double escala=Math.max(escalaAncho,escalaAlto);

            //Calculamos las nuevas medidas
            double L1esc=L1/escala;
```

```
double L2esc=L2/escala;
double L3esc=L3/escala;
double E1esc=E1/escala;
double E2esc=E2/escala;
double E3esc=E3/escala;

//Calculamos el alto efectivo y ancho efectivo del dibujo
double altoefectivo=E1esc+L2esc+E3esc;
double anchoefectivo=anchoViga/escala;

//Seleccionamos el punto para dibujar
double x=(ancho-anchoefectivo)/2;
double y=(alto-altoefectivo)/2;

//Encontramos los puntos para los rectangulos
//Superior
double xinic1;
double yinic1;
double xfin1;
double yfin1;
if (tipoviga=='C') {
       xinic1=x; //x inicial del superior
       yinic1=y; //y inicial del superior
       xfin1=x+L3esc; //x final del superior
       yfin1=y+E3esc; //y final del superior
} else {
       xinic1=ancho/2-L3esc/2; //x inicial del superior
       yinic1=y; //y inicial del superior
       xfin1=xinic1+L3esc; //x final del superior
       yfin1=y+E3esc; //y final del superior
}
//Intermedio
double xinic2;
double yinic2;
double xfin2;
double yfin2;
if (tipoviga=='C') {
       xinic2=x; //x inicial del medio
       yinic2=y+E3esc; //y inicial del medio
       xfin2=x+E2esc; //x final del medio
       yfin2=yinic2+L2esc; //y final del medio
} else {
       xinic2=ancho/2-E2esc/2; //x inicial del medio
       xfin2=xinic2+E2esc; //x final del medio
       yfin2=y+E3esc+L2esc; //y final del medio
       yinic2=yfin2-L2esc; //y inicial del medio
}
//Inferior
double xinic3;
double yinic3;
double xfin3;
double yfin3;
if (tipoviga=='C') {
       xinic3=x; // x inicial del inferior
       yinic3=y+E3esc+L2esc; // y inicial del inferior
       xfin3=x+L1esc; // x final del inferior
       yfin3=yinic3+E1esc; //y final del inferior
} else {
       xinic3=ancho/2-L1esc/2; // x inicial del inferior
       yinic3=y+E3esc+L2esc; // y inicial del inferior
       xfin3=xinic3+L1esc; // x final del inferior
       yfin3=yinic3+E1esc; //y final del inferior
}

//dibujamos el GRAFICO
Graphics2D g = (Graphics2D) dibujo.getGraphics();
//Borramos la imagen anterior
```

```
            g.clearRect(0, 0, dibujo.getSize().width, dibujo.getSize().height);
            //Dibujamos la nueva imagen(primero el fondo)
            g.drawImage(fondo, 0, 0, getWidth(), getHeight(),this);
            //Dibujamos la nueva imagen(ahora el exterior)
            //Damos formato a la linea externa
            Formato.setColorLinea(g);
            g.drawRect((int)xinic1,(int)yinic1,(int)xfin1-(int)xinic1,(int)yfin1-
(int)yinic1);
            g.drawRect((int)xinic2,(int)yinic2,(int)xfin2-(int)xinic2,(int)yfin2-
(int)yinic2);
            g.drawRect((int)xinic3,(int)yinic3,(int)xfin3-(int)xinic3,(int)yfin3-
(int)yinic3);
            //Damos formato al relleno
            Formato.setColorRelleno(g);
            g.fillRect((int)xinic1,(int)yinic1,(int)xfin1-(int)xinic1,(int)yfin1-
(int)yinic1);
            g.fillRect((int)xinic2,(int)yinic2,(int)xfin2-(int)xinic2,(int)yfin2-
(int)yinic2);
            g.fillRect((int)xinic3,(int)yinic3,(int)xfin3-(int)xinic3,(int)yfin3-
(int)yinic3);
            //Dibujamos las cotas
            //Damos el formato de cotas
            Formato.setFormatoCotas(g);
            //Cotas horizontales
            //cotasuperior
            g.drawLine((int)xinic1,(int)yinic1-10,(int)xfin1,(int)yinic1-10);
            g.fillOval((int)(xinic1-2.5),(int)(yinic1-12.5),5,5);
            g.fillOval((int)(xfin1-2.5),(int)(yinic1-12.5),5,5);
            //Para que la posicion del texto quede centrada en la posición le resto 10
que es aprox la mitad de lo que ocupa el texto
            g.drawString(Double.toString(L3),(int)(xinic1+((xfin1-xinic1)/2)-10),
(int)yinic1-15);
            //cotaintermedias
            g.drawLine((int)xinic2,(int)(yinic2+(yfin2-yinic2)/2),(int)xfin2+30,(int)
(yinic2+(yfin2-yinic2)/2));
            g.fillOval((int)(xinic2-2.5),(int)(yinic2+(yfin2-yinic2)/2-2.5),5,5);
            g.fillOval((int)(xfin2-2.5),(int)(yinic2+(yfin2-yinic2)/2-2.5),5,5);
            g.drawString(Double.toString(E2),(int)xfin2+10,(int)(yinic2+(yfin2-yinic2)/
2-2.5));
            //cotainferior
            g.drawLine((int)xinic3,(int)yfin3+25,(int)xfin3,(int)yfin3+25);
            g.fillOval((int)(xinic3-2.5),(int)(yfin3+22.5),5,5);
            g.fillOval((int)(xfin3-2.5),(int)(yfin3+22.5),5,5);
            //Para que la posicion del texto quede centrada en la posición le resto 10
que es aprox la mitad de lo que ocupa el texto
            g.drawString(Double.toString(L1),(int)(xinic3+((xfin3-xinic3)/2)-10),
(int)yfin3+20);
            //Cotas verticales
            //Calculo el ancho mayor para colocar todas las cotas verticales alineadas
en la misma cota x
            int maximaDistancia=Math.max((int)xfin1+10,(int)xfin2+20);
            maximaDistancia=Math.max(maximaDistancia,(int)xfin3+10);
            //cota superior
            g.drawLine(maximaDistancia,(int)yinic1,maximaDistancia,(int)yfin1);
            g.fillOval((int)(maximaDistancia-2.5),(int)(yinic1-2.5),5,5);
            g.fillOval((int)(maximaDistancia-2.5),(int)(yinic2-2.5),5,5);
            g.drawString(Double.toString(E3),(int)(maximaDistancia+10),(int)(yinic1+
(yinic2-yinic1)));
            //Intermedia
            g.drawLine(maximaDistancia,(int)yinic2,maximaDistancia,(int)yfin2);
            g.fillOval((int)(maximaDistancia-2.5),(int)(yinic2-2.5),5,5);
            g.fillOval((int)(maximaDistancia-2.5),(int)(yinic2-2.5),5,5);
            g.drawString(Double.toString(L2),(int)(maximaDistancia+10),(int)(yinic2+
(yfin2-yinic2)/2));
            //cota inferior
            g.drawLine(maximaDistancia,(int)yinic3,maximaDistancia,(int)yfin3);
            g.fillOval((int)(maximaDistancia-2.5),(int)(yinic3-2.5),5,5);
```

```
            g.fillOval((int)(maximaDistancia-2.5),(int)(yfin3-2.5),5,5);
            g.drawString(Double.toString(E1),(int)(maximaDistancia+10),(int)(yinic3+
(yfin3-yinic3)));
      }

      //Llamamos a una nueva VentanaBaseDatosSeccion o la creamos si no existe
      public void abreVentanaBaseDatosSeccion() {
            if (panelBaseDatos!=null) {
                  panelBaseDatos.actualiza();
                  panelBaseDatos.setVisible(true);
            } else panelBaseDatos=new VentanaBaseDatosSeccion(proyecto,this);
      }

      //Incluimos nuevos datos en la Base de Datos
      public void incluyeNuevosDatosSeccion() {
            abreVentanaBaseDatosSeccion();
            //Borramos el arraylist de comprobación y lo rellenamos con los datos a
comprobar
            t.clear();
            t.add(tfe1);
            t.add(tfe2);
            t.add(tfe3);
            t.add(tfl1);
            t.add(tfl2);
            t.add(tfl3);
            boolean resultadosOK=CheckDatos.corrigeDatos(t);
            if(resultadosOK) resultadosOK=CheckDatos.compruebaDatosCompletos(t);
            if(resultadosOK) {
                  L1=Unidades.getConversionCarga()*Double.parseDouble(tfl1.getText());
                  E1=Unidades.getConversionCarga()*Double.parseDouble(tfe1.getText());
                  L2=Unidades.getConversionCarga()*Double.parseDouble(tfl2.getText());
                  E2=Unidades.getConversionCarga()*Double.parseDouble(tfe2.getText());
                  L3=Unidades.getConversionCarga()*Double.parseDouble(tfl3.getText());
                  E3=Unidades.getConversionCarga()*Double.parseDouble(tfe3.getText());
                  //Imprimimos los datos del calculo de la seccion en el area de texto,
metemos una I ya que es sólo para calcular
                  //inercia y en ambos casos C o I es igual en el sentido Z-Z
                  CalculoSeccion calculo = new CalculoSeccion(L1,E1,L2,E2,L3,E3,'I');
                  double area = Math.round(calculo.getArea()*100/100);
                  double inercia= Math.round(calculo.getInercia()*100/100);
                  double moduloResistente = Math.round(calculo.getModulo()*100/100);

panelBaseDatos.incluyeNuevosDatos(E1,L1,E2,L2,E3,L3,area,inercia,moduloResistente);
            }
      }

      //Llama desde base de datos y convierte en texto los valores de seccion usada
      public void extraeDeTabla(String valores, String valores2, String valores3, String
valores4, String valores5, String valores6) {
            //Dejamos un formato decimal para los datos
            DecimalFormat df = new DecimalFormat("0.00");
            //Eliminamos la información que quedaba en el panel de información
            info.setText("");
            infoPanel.repaint();
            //Asignamos los valores pasados

tfe1.setText(df.format(Unidades.getLongitudEnUnidadesActuales(Double.valueOf(valores))));

tfl1.setText(df.format(Unidades.getLongitudEnUnidadesActuales(Double.valueOf(valores2))))
;

tfe2.setText(df.format(Unidades.getLongitudEnUnidadesActuales(Double.valueOf(valores3))))
;

tfl2.setText(df.format(Unidades.getLongitudEnUnidadesActuales(Double.valueOf(valores4))))
;
```

```
tfe3.setText(df.format(Unidades.getLongitudEnUnidadesActuales(Double.valueOf(valores5))))
;

tfl3.setText(df.format(Unidades.getLongitudEnUnidadesActuales(Double.valueOf(valores6))))
;
              //Volvemos a hacer esta ventana visible si no lo estaba.
              setVisible(true);
              //La repintamos
              repaint();
       }
}
```

C.3. Paquete cargasEstructura

Se expone a continuación el código de las clases del paquete cargasEstructura que dispone el código para la gestión y creación de cargas para introducir en la estructura

CLASE Cargas

```
package cargasEstructura;

import java.awt.Color;
import java.io.Serializable;
import camiones.*;

public class Carga implements CargasInterface, Serializable{
	private static final long serialVersionUID = 1L;
	private String nombre="";
	private TipoCarga tipo;
	private double puntoInicial=0;
	private double valor=1;
	private boolean visible=true;

	public Carga(String nombre,double puntoInicial,double valor,boolean visible) {
		this.nombre=nombre;
		this.puntoInicial=puntoInicial;
		this.valor=valor;
		this.visible=visible;
	}

	//GENERAMOS LOS SETTERS
	//Asigna el nombre de la carga
	@Override
	public void setNombre(String nombre) {
		this.nombre=nombre;
	}

	//Asigna el valor de la carga
	@Override
	public void setValor(double valor) {
		this.valor=valor;
	}

	//Asigna la visibilidad de la carga
	@Override
	public void setVisible(boolean visible) {
		this.visible=visible;
	}

	//Asigna el tipo de Carga
	@Override
	public void setTipo(TipoCarga tipo) {
		this.tipo=tipo;
	}

	//Asigna el punto inicial de la carga
	@Override
	public void setPuntoInicio(double punto) {
		this.puntoInicial=punto;
	}

	//GENERAMOS LOS GETTERS
	//Obtiene el valor de la carga
	@Override
	public double getValor() {
		return valor;
	}

	//Obtiene el tipo de la carga
	@Override
	public TipoCarga getTipo() {
		return tipo;
	}
```

```
	//Obtiene el punto inicial de la carga
	@Override
	public double getPuntoInicio() {
		return puntoInicial;
	}

	//Obtiene el nombre de la carga
	@Override
	public String getNombre() {
		return nombre;
	}

	//Obtiene la visibilidad de la carga
	@Override
	public boolean getVisible() {
		return visible;
	}

}
```

CLASE CargaDistribuida

```
package cargasEstructura;

import java.io.Serializable;

public class CargaDistribuida extends Carga implements Serializable {
	private static final long serialVersionUID = 1L;
	private double longitud=0;
	public CargaDistribuida(String nombre,double puntoInicial,double valor,boolean
visible,double longitud) {
		super(nombre,puntoInicial,valor,visible);
		this.setTipo(new TipoCarga('D'));
		setLongitud(longitud);
	}

	//Generamos un método para asignar la longitud
	public void setLongitud(double longitud) {
		this.longitud=longitud;
	}

	//Generamos un método para obtener la longitud
	public double getLongitud() {
		return this.longitud;
	}

}
```

CLASE CargaPuntual

```
package cargasEstructura;

import java.io.Serializable;

public class CargaPuntual extends Carga implements Serializable{
	private static final long serialVersionUID = 1L;

	public CargaPuntual(String nombre,double puntoInicial,double valor,boolean
visible) {
		super(nombre,puntoInicial,valor,visible);
		this.setTipo(new TipoCarga('P'));
	}
}
```

INTERFACE CargasInterface

```
package cargasEstructura;

import java.awt.Color;

public interface CargasInterface {

	//Definimos los metodos que implementarán las clases
	//Para el valor de la carga
	public double getValor();
	public void setValor(double valor);
	//Para el tipo de carga
	public TipoCarga getTipo();
	public void setTipo(TipoCarga tipo);
	//Para definir el punto donde se aplica
	public double getPuntoInicio();
	public void setPuntoInicio(double punto);
	//Para definir el nombre de la carga
	public String getNombre();
	public void setNombre(String nombre);
	//Para definir si es o no visible
	public boolean getVisible();
	public void setVisible(boolean visible);
}
```

CLASE TipoCarga

```
package cargasEstructura;

import java.io.Serializable;

import javax.swing.ImageIcon;
import javax.swing.JOptionPane;

public class TipoCarga implements Serializable {
       private static final long serialVersionUID = 1L;
       private String descripcion;
       private char tipo;
       public TipoCarga(char tipo) {
              setTipo(tipo);
       }

    //Este método asigna el tipo de carga
    public void setTipo(char tipo) {
              if(tipo=='P') {
                     this.tipo='P';
                     this.descripcion="Carga puntual";
              }
              else if(tipo=='D') {
                     this.tipo='D';
                     this.descripcion="Carga distribuida";
              } else {
                     this.tipo='N';
                     this.descripcion="Carga no valida";
              }
    }

    //Devuelve la descripcion
    public String getDescripcion() {
       return descripcion;
    }

    //Devuelve el tipo
    public char getTipoCarga() {
       return tipo;
    }
}
```

C.4. Paquete calculosEstructura

Se expone a continuación el contenido del paquete calculosEstructura que gestiona todo el código relativo a la manipulación de los cálculos que deben realizarse con la estructura del vehículo.

CLASE CalculoReaccionesArticulado

```
package calculosEstructura;

import java.util.ArrayList;
import java.util.Iterator;

import auxiliares.CheckDatos;
import camiones.*;
import cargasEstructura.Carga;
import cargasEstructura.CargaDistribuida;
import cargasEstructura.CargaPuntual;

public abstract class CalculoReaccionesArticulado {
	private static double reaccionA=0;
	private static double reaccionB=0;
	private static double reaccionKingPin=0;
	private static double reaccionRemolque=0;
	private static double reaccionAparcial=0;
	private static double reaccionBparcial=0;
	private static double reaccionKingPinParcial=0;
	private static double reaccionRemolqueParcial=0;
	private static Proyecto proyecto;

	//Calculamos para una carga puntual dada las reacciones en camion
	public static void calculaReaccionesPorCargaPuntualCabezaTractora(Carga q,
Proyecto p) {
		proyecto=p;
		double valor=q.getValor();
		double posicion=q.getPuntoInicio();
		double posicionA=proyecto.getCamion().getdistFrenteEjeDelantero();
		double posicionB=proyecto.getCamion().getdistFrenteEjeDelantero()
+proyecto.getCamion().getdistEjeDelanteroEjeTrasero();

		//Hacemos sumatorio de cargas verticales=0.
		//Supongo la direccion de la carga como sentido negativo y las reacciones
dirigidas hacia el sentido positivo
		//-valor+reaccionA+reaccionB=0
		//valor=reaccionA+reaccionB

		//Hacemos sumatorio de momentos=0
		//-valor*posicion+reaccionA*dFD+reaccionB*(dFD+dFT)=0

		//Despejando
		reaccionBparcial=valor*(posicion-posicionA)/(posicionB-posicionA);
		reaccionAparcial=valor-reaccionBparcial;
	}

	//Calculamos para una carga puntual dada las reacciones en camion
	public static void calculaReaccionesPorCargaPuntualRemolque(Carga q, Proyecto p) {
		proyecto=p;
		double valor=q.getValor();
		//En este caso cambian los ejes que consideramos
		double posicion=q.getPuntoInicio();
		double posicionA=getDistanciaKingPin(proyecto);
		double posicionB=posicionA+
((CamionArticulado)proyecto.getCamion()).getDistanciaKingPinEjeRemolque();

		//Hacemos sumatorio de cargas verticales=0.
		//Supongo la direccion de la carga como sentido negativo y las reacciones
dirigidas hacia el sentido positivo
		//-valor+reaccionA+reaccionB=0
		//valor=reaccionA+reaccionB

		//Hacemos sumatorio de momentos=0
		//-valor*posicion+reaccionA*dFD+reaccionB*(dFD+dFT)=0
```

```
			//Despejando
			reaccionRemolqueParcial=valor*(posicion-posicionA)/(posicionB-posicionA);
			reaccionKingPinParcial=valor-reaccionRemolqueParcial;
		}

		//Calculamos para una carga distribuida dada las reacciones en camion
		public static void calculaReaccionesPorCargaDistribuidaCabezaTractora(Carga q,
Proyecto p) {
			proyecto=p;
			CargaDistribuida qd=(CargaDistribuida)q;
			double longitud=qd.getLongitud();
			double valor=q.getValor();
			double posicion=q.getPuntoInicio();
			double posicionA=proyecto.getCamion().getdistFrenteEjeDelantero();
			double posicionB=proyecto.getCamion().getdistFrenteEjeDelantero()
+proyecto.getCamion().getdistEjeDelanteroEjeTrasero();

			//Hacemos sumatorio de cargas verticales=0.
			//Supongo la direccion de la carga como sentido negativo y las reacciones
dirigidas hacia el sentido positivo
			//-valor+reaccionA+reaccionB=0
			//valor=reaccionA+reaccionB

			//Hacemos sumatorio de momentos=0
			//-valor*posicion+reaccionA*dFD+reaccionB*(dFD+dFT)=0

			//Despejando
			reaccionBparcial=((valor*longitud*(posicion+(longitud/2)))-
(valor*longitud)*posicionA)/(posicionB-posicionA);
			reaccionAparcial=(valor*longitud)-reaccionBparcial;
		}

		//Calculamos para una carga distribuida dada las reacciones en camion
		public static void calculaReaccionesPorCargaDistribuidaRemolque(Carga q, Proyecto
p) {
			proyecto=p;
			CargaDistribuida qd=(CargaDistribuida)q;
			double longitud=qd.getLongitud();
			double valor=q.getValor();
			double posicion=q.getPuntoInicio();
			double posicionA=proyecto.getCamion().getdistFrenteEjeDelantero()+
((CamionArticulado)proyecto.getCamion()).getDistanciaEjeDelanteroKingPin();
			double posicionB=posicionA+
((CamionArticulado)proyecto.getCamion()).getDistanciaKingPinEjeRemolque();

			//Hacemos sumatorio de cargas verticales=0.
			//Supongo la direccion de la carga como sentido negativo y las reacciones
dirigidas hacia el sentido positivo
			//-valor+reaccionA+reaccionB=0
			//valor=reaccionA+reaccionB

			//Hacemos sumatorio de momentos=0
			//-valor*posicion+reaccionA*dFD+reaccionB*(dFD+dFT)=0

			//Despejando
			reaccionRemolqueParcial=((valor*longitud*(posicion+(longitud/2)))-
(valor*longitud)*posicionA)/(posicionB-posicionA);
			reaccionKingPinParcial=(valor*longitud)-reaccionRemolqueParcial;
		}

		//Aplicando el principio de superposicion vamos sumando las reacciones parciales
para obtener la total.
		public static void sumareaccionesRemolque(Proyecto p) {
			reaccionKingPin=0;
			reaccionRemolque=0;
			proyecto=p;
```

```
			//Marcamos la distancia a partir de la que consideraremos que las cargas
pertenecen al remolque
			double distanciaParaRemolque = getDistanciaMinima(proyecto);
			ArrayList<Carga> al= proyecto.getCargas();
			Iterator<Carga> it = al.iterator();
			Carga q=null;
			while(it.hasNext()){
				q=it.next();
				if(q.getPuntoInicio()>=distanciaParaRemolque) {
					if(q.getTipo().getTipoCarga()=='P') {
						calculaReaccionesPorCargaPuntualRemolque(q,proyecto);
						reaccionKingPin=reaccionKingPinParcial+reaccionKingPin;

reaccionRemolque=reaccionRemolqueParcial+reaccionRemolque;
					} else {

calculaReaccionesPorCargaDistribuidaRemolque(q,proyecto);
						reaccionKingPin=reaccionKingPinParcial+reaccionKingPin;

reaccionRemolque=reaccionRemolqueParcial+reaccionRemolque;
					}
				}
			}
		}

		public static void sumareaccionesCabeza(Proyecto p) {
			reaccionA=0;
			reaccionB=0;
			proyecto=p;
			ArrayList<Carga> al= proyecto.getCargas();
			//Marcamos la distanciaMinimahasta la que consideraremos que las cargas
pertenecen a la cabeza tractora
			double distanciaParaCabeza = getDistanciaMinima(proyecto);
			//Iteramos con las cargas que queden en la zona de la cabeza
			Iterator<Carga> it = al.iterator();
			Carga q=null;
			while(it.hasNext()){
				q=it.next();
				if((q.getPuntoInicio()<distanciaParaCabeza)) {
					if(q.getTipo().getTipoCarga()=='P') {

calculaReaccionesPorCargaPuntualCabezaTractora(q,proyecto);
						reaccionA=reaccionAparcial+reaccionA;
						reaccionB=reaccionBparcial+reaccionB;
					} else {

calculaReaccionesPorCargaDistribuidaCabezaTractora(q,proyecto);
						reaccionA=reaccionAparcial+reaccionA;
						reaccionB=reaccionBparcial+reaccionB;
					}
				}
			}
			//Creamos una nueva Carga que será la reaccion del King Pin sobre la cabeza
tractora
			double distanciaKingPin=getDistanciaKingPin(proyecto);
			Carga kingPinCarga=new
CargaPuntual("KingPin",distanciaKingPin,reaccionKingPin,true);
			calculaReaccionesPorCargaPuntualCabezaTractora(kingPinCarga,proyecto);
			reaccionA=reaccionAparcial+reaccionA;
			reaccionB=reaccionBparcial+reaccionB;
		}

		public static double getReaccionA() {
			return reaccionA;
		}

		public static double getReaccionB() {
```

```
		return reaccionB;
	}

	public static double getReaccionKingPin() {
		return reaccionKingPin;
	}

	public static double getReaccionRemolque() {
		return reaccionRemolque;
	}
	public static double getDistanciaMinima(Proyecto p) {
		proyecto=p;
		//Creamos una distancia que será hasta la que consideramos las cargas en
cabeza a partir de ella serán consideradas en el remolque
		double dCFC =
((CamionArticulado)proyecto.getCamion()).getdistFrenteEjeDelantero()+
((CamionArticulado)proyecto.getCamion()).getDistanciaEjeDelanteroComienzoCaja();
		return dCFC;
	}

	public static double getDistanciaKingPin(Proyecto p) {
		proyecto=p;
		//Creamos una nueva Carga que será la posicion del KingPin
		double dKP =
((CamionArticulado)proyecto.getCamion()).getdistFrenteEjeDelantero()+
((CamionArticulado)proyecto.getCamion()).getDistanciaEjeDelanteroKingPin();
		return dKP;
	}

	public static double getDistanciaRemolque(Proyecto p) {
		proyecto=p;
		//Creamos una nueva Carga que será la posicion del KingPin
		double dR = getDistanciaKingPin(proyecto)+
((CamionArticulado)proyecto.getCamion()).getDistanciaKingPinEjeRemolque();
		return dR;
	}

    //Muestra las reacciones y avisa
    public static void compruebaReacciones(Proyecto proyecto) {
       boolean errorValores=false;
       if(proyecto.getCamion().getTipo().getTipo()=='R') {
                    if(CalculoReaccionesArticulado.getReaccionA()>
proyecto.getCamion().getCargaMaxEjeDelantero()) errorValores=true;

if(CalculoReaccionesArticulado.getReaccionB()>proyecto.getCamion().getCargaMaxEjeTrasero(
)) errorValores=true;

if(CalculoReaccionesArticulado.getReaccionRemolque()>((CamionArticulado)proyecto.getCamio
n()).getCargaMaximaEjeRemolque()) errorValores=true;
                    if (errorValores) CheckDatos.okNokMensaje("La carga sobrepasa las
admisibles en los ejes");
	}

    }

}
```

CLASE CalculoReaccionesRigido

```
package calculosEstructura;

import java.util.ArrayList;
import java.util.Iterator;

import auxiliares.CheckDatos;
import camiones.*;
import cargasEstructura.Carga;
import cargasEstructura.CargaDistribuida;

public abstract class CalculoReaccionesRigido {
	private static double reaccionA=0;
	private static double reaccionB=0;
	private static double reaccionAparcial=0;
	private static double reaccionBparcial=0;

	//Calculamos para una carga puntual dada las reacciones en camion
	public static void calculaReaccionesPorCargaPuntualCamion(Carga q, Proyecto
proyecto) {
		double valor=q.getValor();
		double posicion=q.getPuntoInicio();
		double posicionA=proyecto.getCamion().getdistFrenteEjeDelantero();
		double posicionB=proyecto.getCamion().getdistFrenteEjeDelantero()
+proyecto.getCamion().getdistEjeDelanteroEjeTrasero();

		//Hacemos sumatorio de cargas verticales=0.
		//Supongo la direccion de la carga como sentido negativo y las reacciones
dirigidas hacia el sentido positivo
		//-valor+reaccionA+reaccionB=0
		//valor=reaccionA+reaccionB

		//Hacemos sumatorio de momentos=0
		//-valor*posicion+reaccionA*dFD+reaccionB*(dFD+dFT)=0

		//Despejando
		reaccionBparcial=valor*(posicion-posicionA)/(posicionB-posicionA);
		reaccionAparcial=valor-reaccionBparcial;
	}

	//Calculamos para una carga distribuida dada las reacciones en camion
	public static void calculaReaccionesPorCargaDistribuidaCamion(Carga q, Proyecto
proyecto) {
		CargaDistribuida qd=(CargaDistribuida)q;
		double longitud=qd.getLongitud();
		double valor=q.getValor();
		double posicion=q.getPuntoInicio();
		double posicionA=proyecto.getCamion().getdistFrenteEjeDelantero();
		double posicionB=proyecto.getCamion().getdistFrenteEjeDelantero()
+proyecto.getCamion().getdistEjeDelanteroEjeTrasero();

		//Hacemos sumatorio de cargas verticales=0.
		//Supongo la direccion de la carga como sentido negativo y las reacciones
dirigidas hacia el sentido positivo
		//-valor+reaccionA+reaccionB=0
		//valor=reaccionA+reaccionB

		//Hacemos sumatorio de momentos=0
		//-valor*posicion+reaccionA*dFD+reaccionB*(dFD+dFT)=0

		//Despejando
		reaccionBparcial=((valor*longitud*(posicion+(longitud/2)))-
(valor*longitud)*posicionA)/(posicionB-posicionA);
		reaccionAparcial=(valor*longitud)-reaccionBparcial;
	}
```

```
      //Aplicando el principio de superposicion vamos sumando las reacciones parciales
para obtener la total.
      public static void sumareacciones(Proyecto proyecto) {
            reaccionA=0;
            reaccionB=0;
            ArrayList<Carga> al= proyecto.getCargas();
            Iterator<Carga> it = al.iterator();
            Carga q=null;
            while(it.hasNext()){
                  q=it.next();
                  if(q.getTipo().getTipoCarga()=='P') {
                        calculaReaccionesPorCargaPuntualCamion(q,proyecto);
                        reaccionA=reaccionAparcial+reaccionA;
                        reaccionB=reaccionBparcial+reaccionB;
                  } else {
                        calculaReaccionesPorCargaDistribuidaCamion(q,proyecto);
                        reaccionA=reaccionAparcial+reaccionA;
                        reaccionB=reaccionBparcial+reaccionB;
                  }
            }
      }

      public static double getReaccionA() {
            return reaccionA;
      }

      public static double getReaccionB() {
            return reaccionB;
      }

    //Muestra las reacciones y avisa
    public static void compruebaReacciones(Proyecto proyecto) {
      boolean errorValores=false;
      if(proyecto.getCamion().getTipo().getTipo()=='R') {
                  if(CalculoReaccionesRigido.getReaccionA()>
proyecto.getCamion().getCargaMaxEjeDelantero()) errorValores=true;

if(CalculoReaccionesRigido.getReaccionB()>proyecto.getCamion().getCargaMaxEjeTrasero())
errorValores=true;
                  if (errorValores) CheckDatos.okNokMensaje("La carga sobrepasa las
admisibles en los ejes");
      }

    }

}
```

CLASE LeyesCortantes

```
package calculosEstructura;

import java.util.ArrayList;
import java.util.Comparator;
import java.util.Iterator;

import auxiliares.DatosGenericos;
import camiones.*;
import cargasEstructura.Carga;
import cargasEstructura.CargaDistribuida;
import cargasEstructura.CargaPuntual;

public abstract class LeyesCortantes {
	private static ArrayList<Carga>
cargas,cortantes,cortantesCabeza,cortantesRemolque;
	private static ArrayList<Double> posiciones,posicionesCabeza,posicionesRemolque;
	private static ArrayList<PuntosCalculo>
puntosCambioCortantes,puntosCambioCortantesCabeza,puntosCambioCortantesRemolque;
	private static Proyecto proyecto;
	//Esta variable guarda la maxima carga que se usará para definir la escala al
dibujar
	private static double maximaCarga=0;
	private static double maximaCargaCabeza=0;
	private static double maximaCargaRemolque=0;
	//Esta variable indica cada cuantos puntos tomo uno para dibujar. Un valor mas
alto implica mas precision pero mas tiempo de procesamiento
	private static int puntosGrafico=1;
	private static boolean zonaCabeza=true;

	//Preparamos todo llamando a los distintos métodos
	public static void prepara() {
		introduceCargas();
		generaPuntosCortantes();
	}

	//Este metodo asigna el proyecto
	public static void setProyecto(Proyecto p) {
		proyecto=p;
	}

	//Este método introduce todas las cargas una a una en el proyecto
	public static void introduceCargas() {
		if(proyecto.getCamion().getTipo().getTipo()=='R') {
			cargas=proyecto.getCargas();
			//Introducimos el origen, las reacciones y el final
			inicializa();
			Iterator<Carga> it = cargas.iterator();
			while(it.hasNext()) {
				Carga q=it.next();
				cortantes.add(q);
			}
			generaPosicionesClave();
		}
		if(proyecto.getCamion().getTipo().getTipo()=='A') {
			cargas=proyecto.getCargas();
			//Introducimos el origen, las reacciones y el final
			double
posicionCaja=CalculoReaccionesArticulado.getDistanciaMinima(proyecto);
			inicializa();
			Iterator<Carga> it = cargas.iterator();
			while(it.hasNext()) {
				Carga q=it.next();
				if(q.getPuntoInicio()<posicionCaja)
					cortantesCabeza.add(q);
				else
```

```
                                    cortantesRemolque.add(q);
                        }
                        generaPosicionesClave();
                }
        }

        //Este metodo inicializa los cortantes en los puntos representatios
        public static void inicializa() {
                if(proyecto.getCamion().getTipo().getTipo()=='R') {
                        cortantes=new ArrayList<Carga>();
                        cortantes.add(new CargaPuntual("Inicio",0,0,false));
                        cortantes.add(new
CargaPuntual("Ra",proyecto.getCamion().getdistFrenteEjeDelantero(),-
CalculoReaccionesRigido.getReaccionA(),false));
                        cortantes.add(new
CargaPuntual("Rb",proyecto.getCamion().getdistFrenteEjeDelantero()
+proyecto.getCamion().getdistEjeDelanteroEjeTrasero(),-
CalculoReaccionesRigido.getReaccionB(),false));
                        cortantes.add(new
CargaPuntual("Fin",proyecto.getCamion().getLargo(),0,false));
                }
                //Si es articulado lo preparamos metemos las reacciones y cargas extra
                if(proyecto.getCamion().getTipo().getTipo()=='A') {cortantes=new
ArrayList<Carga>();
                        double
distanciaKingPin=CalculoReaccionesArticulado.getDistanciaKingPin(proyecto);
                        double cargaKingPin=CalculoReaccionesArticulado.getReaccionKingPin();
                        double
distanciaRemolque=CalculoReaccionesArticulado.getDistanciaRemolque(proyecto);
                        double
cargaRemolque=CalculoReaccionesArticulado.getReaccionRemolque();
                        cortantesCabeza=new ArrayList<Carga>();
                        cortantesRemolque=new ArrayList<Carga>();
                        cortantesCabeza.add(new CargaPuntual("Inicio",0,0,false));
                        cortantesCabeza.add(new
CargaPuntual("Ra",proyecto.getCamion().getdistFrenteEjeDelantero(),-
CalculoReaccionesArticulado.getReaccionA(),false));
                        cortantesCabeza.add(new
CargaPuntual("KingPin",distanciaKingPin,cargaKingPin,false));
                        cortantesCabeza.add(new
CargaPuntual("Rb",proyecto.getCamion().getdistFrenteEjeDelantero()
+proyecto.getCamion().getdistEjeDelanteroEjeTrasero(),-
CalculoReaccionesArticulado.getReaccionB(),false));
                        //Vamos a considerar que la cabeza tractora termina 500 mm detras del
King Pin
                        cortantes.add(new
CargaPuntual("Fin",proyecto.getCamion().getdistFrenteEjeDelantero()
+proyecto.getCamion().getdistEjeDelanteroEjeTrasero(),0,false));
                        cortantesRemolque.add(new
CargaPuntual("Inicio",CalculoReaccionesArticulado.getDistanciaMinima(proyecto),0,false));
                        cortantesRemolque.add(new CargaPuntual("KingPin",distanciaKingPin,-
cargaKingPin,false));
                        cortantesRemolque.add(new CargaPuntual("Rc",distanciaRemolque,-
cargaRemolque,false));
                        cortantesRemolque.add(new
CargaPuntual("Fin",proyecto.getCamion().getLargo(),0,false));
                }
        }

        //Este metodo devuele la carga cortante en una posicion del torque
        public static double calculaCortante(double posicion,ArrayList<Carga> alc) {
                double carga=0;
                Iterator<Carga> it = alc.iterator();
                while(it.hasNext()) {
                        Carga q=it.next();
                        if(q.getTipo().getTipoCarga()=='P') {
                                if(q.getPuntoInicio()<=posicion) {
```

```
                                carga=carga+q.getValor();
                        }
                    } else {
                        CargaDistribuida cd=(CargaDistribuida)q;
                        double valorCarga=cd.getValor();

if((posicion>=cd.getPuntoInicio())&&(posicion<=(cd.getLongitud()+cd.getPuntoInicio()))) {
                                carga=carga+valorCarga*(posicion-cd.getPuntoInicio());
                        } else if(posicion>(cd.getPuntoInicio()+cd.getLongitud())){
                                carga=carga+valorCarga*cd.getLongitud();
                        }
                    }
                }
                carga=(double) carga;
                return carga;
        }

        //Este método genera las posiciones clave en la viga del camion
        public static void generaPosicionesClave() {
                if(proyecto.getCamion().getTipo().getTipo()=='R') {
                    posiciones=new ArrayList<Double>();
                    Iterator<Carga> it = cortantes.iterator();
                    while(it.hasNext()) {
                        Carga q=it.next();
                        if(q.getTipo().getTipoCarga()=='P') {
                                //Metemos un punto extra antes para poder ver como
evoluciona en el cambio
                                if(q.getPuntoInicio()!=0)
posiciones.add(q.getPuntoInicio()-0.1);
                                posiciones.add(q.getPuntoInicio());
                                //Metemos un punto extra despues para poder ver como
evoluciona en el cambio
                                if(q.getPuntoInicio()!=proyecto.getCamion().getLargo())
posiciones.add(q.getPuntoInicio()+0.1);
                        } else {
                                CargaDistribuida cd=(CargaDistribuida)q;
                                //Añadimos mas puntos para ver el cambio con la
longitud
                                double tramoCarga=cd.getLongitud()/20;
                                for(int tramos=1;tramos<20;tramos++) {
                                        posiciones.add(cd.getPuntoInicio()+
(tramoCarga*tramos));
                                }
                                //Metemos el punto inicial y final
                                posiciones.add(cd.getPuntoInicio());
                                posiciones.add(cd.getPuntoInicio()+cd.getLongitud());
                        }
                    }
                    //Ordenamos el ArrayList
                    posiciones.sort(Comparator.naturalOrder());
                }
                if(proyecto.getCamion().getTipo().getTipo()=='A') {
                    posicionesCabeza=new ArrayList<Double>();
                    Iterator<Carga> it = cortantesCabeza.iterator();
                    while(it.hasNext()) {
                        Carga q=it.next();
                        if(q.getTipo().getTipoCarga()=='P') {
                                //Metemos un punto extra antes para poder ver como
evoluciona en el cambio
                                if(q.getPuntoInicio()!=0)
posicionesCabeza.add(q.getPuntoInicio()-0.1);
                                posicionesCabeza.add(q.getPuntoInicio());
                                //Metemos un punto extra despues para poder ver como
evoluciona en el cambio
                                if(q.getPuntoInicio()!=proyecto.getCamion().getLargo())
posicionesCabeza.add(q.getPuntoInicio()+0.1);
                        } else {
```

```
                                CargaDistribuida cd=(CargaDistribuida)q;
                                //Añadimos mas puntos para ver el cambio con la
longitud
                                double tramoCarga=cd.getLongitud()/20;
                                for(int tramos=1;tramos<20;tramos++) {
                                      posicionesCabeza.add(cd.getPuntoInicio()+
(tramoCarga*tramos));
                                }
                                //Metemos el punto inicial y final
                                posicionesCabeza.add(cd.getPuntoInicio());
                                posicionesCabeza.add(cd.getPuntoInicio()
+cd.getLongitud());
                          }
                    }
                    //Ordenamos el ArrayList
                    posicionesCabeza.sort(Comparator.naturalOrder());

                    posicionesRemolque=new ArrayList<Double>();
                    Iterator<Carga> itr = cortantesRemolque.iterator();
                    while(itr.hasNext()) {
                          Carga q=itr.next();
                          if(q.getTipo().getTipoCarga()=='P') {
                                //Metemos un punto extra antes para poder ver como
evoluciona en el cambio
                                if(q.getPuntoInicio()!=0)
posicionesRemolque.add(q.getPuntoInicio()-0.1);
                                posicionesRemolque.add(q.getPuntoInicio());
                                //Metemos un punto extra despues para poder ver como
evoluciona en el cambio
                                if(q.getPuntoInicio()!=proyecto.getCamion().getLargo())
posicionesRemolque.add(q.getPuntoInicio()+0.1);
                          } else {
                                CargaDistribuida cd=(CargaDistribuida)q;
                                //Añadimos mas puntos para ver el cambio con la
longitud
                                double tramoCarga=cd.getLongitud()/20;
                                for(int tramos=1;tramos<20;tramos++) {
                                      posicionesRemolque.add(cd.getPuntoInicio()+
(tramoCarga*tramos));
                                }
                                //Metemos el punto inicial y final
                                posicionesRemolque.add(cd.getPuntoInicio());
                                posicionesRemolque.add(cd.getPuntoInicio()
+cd.getLongitud());
                          }
                    }
                    //Ordenamos el ArrayList
                    posicionesRemolque.sort(Comparator.naturalOrder());
              }
        }

        //Este metodo genera los pares (posicion,cortante)
        public static void generaPuntosCortantes() {
              if(proyecto.getCamion().getTipo().getTipo()=='R') {
                    double carga=0;
                    Iterator<Double> itpos=posiciones.iterator();
                    puntosCambioCortantes=new ArrayList<PuntosCalculo>();
                    while(itpos.hasNext()) {
                          Double punto = itpos.next();
                          carga=calculaCortante(punto,cortantes);
                          puntosCambioCortantes.add(new PuntosCalculo(punto,carga));
                    }
              }
              if(proyecto.getCamion().getTipo().getTipo()=='A') {
                    double cargaCabeza=0;
                    Iterator<Double> itposcab=posicionesCabeza.iterator();
                    puntosCambioCortantesCabeza=new ArrayList<PuntosCalculo>();
```

```
                    while(itposcab.hasNext()) {
                          Double punto = itposcab.next();
                          cargaCabeza=calculaCortante(punto,cortantesCabeza);
                          puntosCambioCortantesCabeza.add(new
PuntosCalculo(punto,cargaCabeza));
                    }
                    double cargaRemolque=0;
                    Iterator<Double> itposrem=posicionesRemolque.iterator();
                    puntosCambioCortantesRemolque=new ArrayList<PuntosCalculo>();
                    while(itposrem.hasNext()) {
                          Double punto = itposrem.next();
                          cargaRemolque=calculaCortante(punto,cortantesRemolque);
                          puntosCambioCortantesRemolque.add(new
PuntosCalculo(punto,cargaRemolque));
                    }
             }
      }

      //Devuelve el camion por trozos iguales y saca los valores para cada punto
devolviendolos
      public static ArrayList<PuntosCalculo> puntosGrafico(Proyecto p) {
             proyecto=p;
             ArrayList<PuntosCalculo> listaPuntos = new ArrayList<PuntosCalculo>();
             //Reseteamos el cortante que nos sirve para escalar el gráfico
             maximaCarga=0;
             maximaCargaCabeza=0;
             maximaCargaRemolque=0;
             prepara();
             double carga=0;
             int i=0;
             if(proyecto.getCamion().getTipo().getTipo()=='R') {
                    while (i<proyecto.getCamion().getLargo()){
                          carga=carga+calculaCortante(i,cortantes);
                          if(Math.abs(carga)>Math.abs(maximaCarga)) {
                                maximaCarga=carga;

proyecto.getCamion().setTensionCortanteViga(maximaCarga);
                                proyecto.getCamion().setTensionVonMisses();
                          }
                          PuntosCalculo puntoC=new PuntosCalculo(i,carga);
                          i=i+puntosGrafico;
                          carga=0;
                          listaPuntos.add(puntoC);
                    }
                    return listaPuntos;
             } else {
                    //Obtenemos cuanta longitud consideramos despues del eje trasero
                    double parteExtra=DatosGenericos.getLongExtra();
                    if(zonaCabeza) {
                          double
distanciaAConsiderar=((CamionArticulado)proyecto.getCamion()).getdistFrenteEjeDelantero()
+

((CamionArticulado)proyecto.getCamion()).getdistEjeDelanteroEjeTrasero()+parteExtra;
                          while (i<=((CamionArticulado)proyecto.getCamion()).getLargo())
{
                                if((i<=distanciaAConsiderar)&&(i>=0)){
                                      carga=carga+calculaCortante(i,cortantesCabeza);
                                      if(Math.abs(carga)>Math.abs(maximaCargaCabeza)) {
                                            maximaCargaCabeza=carga;

((CamionArticulado)proyecto.getCamion()).setTensionCortanteViga(maximaCargaCabeza);

((CamionArticulado)proyecto.getCamion()).setTensionVonMisses();
                                      }
                                      PuntosCalculo puntoC=new PuntosCalculo(i,carga);
                                      carga=0;
```

```
                                        listaPuntos.add(puntoC);
                                  }
                                  i=i+puntosGrafico;
                            }
                            return listaPuntos;
                      } else {
                            double
distanciaAConsiderar=proyecto.getCamion().getdistFrenteEjeDelantero()+
((CamionArticulado)proyecto.getCamion()).getDistanciaEjeDelanteroKingPin()-parteExtra;
                            while (i<=proyecto.getCamion().getLargo()){
                                  if(i>distanciaAConsiderar) {
                                        carga=carga+calculaCortante(i,cortantesRemolque);
                                        if(Math.abs(carga)>Math.abs(maximaCargaRemolque))
{
                                              maximaCargaRemolque=carga;

((CamionArticulado)proyecto.getCamion()).setTensionCortanteVigaRemolque(maximaCargaRemolq
ue);

((CamionArticulado)proyecto.getCamion()).setTensionVonMissesVigaRemolque();
                                        }
                                        PuntosCalculo puntoC=new PuntosCalculo(i,carga);
                                        carga=0;
                                        listaPuntos.add(puntoC);
                                  }
                                  i=i+puntosGrafico;
                            }
                            return listaPuntos;
                      }
                }
          }

          //Obtenemos la maxima carga de camion
          public static double getMaximaCarga() {
                return maximaCarga;
          }

          //Reseteamos la maxima carga de camion
          public static void resetMaximaCarga() {
                maximaCarga=0;
          }

          //Obtenemos la maxima carga de cabeza
          public static double getMaximaCargaCabeza() {
                return maximaCargaCabeza;
          }

          //Reseteamos la maxima carga de cabeza
          public static void resetMaximaCargaCabeza() {
                maximaCargaCabeza=0;
          }

          //Obtenemos la maxima carga de remolque
          public static double getMaximaCargaRemolque() {
                return maximaCargaRemolque;
          }

          //Reseteamos la maxima carga de cabeza
          public static void resetMaximaCargaRemolque() {
                maximaCargaRemolque=0;
          }

          //marcamos la zona del camion donde queremos calcular en el articulado
          public static void setParteCabeza(boolean b) {
                zonaCabeza=b;
          }
```

```
	//devolvemos la parte con la que estamos trabajando
	public static boolean getParteCabeza() {
		return zonaCabeza;
	}
	//Este metodo calcula los valores de tensiones y para ello usa tanto la cabeza
como el remolque
	public static void preparaParaTension() {
		//Primero guardamos el valor actual
		boolean zona=zonaCabeza;
		//Hacemos una primera pasada para obtener las reacciones en la zona de
cabeza tractora
		zonaCabeza=true;
		puntosGrafico(proyecto);
		//Hacemos una segunda pasada para obtener las reacciones en la zona de
remolque
		zonaCabeza=false;
		puntosGrafico(proyecto);
		//Reasignamos el valor que tenía zonaCabeza.
		zonaCabeza=zona;
	}
}
```

CLASE LeyesMomentos

```
package calculosEstructura;

import java.util.ArrayList;
import java.util.Comparator;
import java.util.Iterator;

import auxiliares.DatosGenericos;
import camiones.*;
import cargasEstructura.Carga;
import cargasEstructura.CargaDistribuida;
import cargasEstructura.CargaPuntual;

public abstract class LeyesMomentos {
	private static ArrayList<Carga> cargas,momentos,momentosCabeza,momentosRemolque;
	private static ArrayList<Double> posiciones,posicionesCabeza,posicionesRemolque;
	private static ArrayList<PuntosCalculo>
puntosCambioMomento,puntosCambioMomentoCabeza,puntosCambioMomentoRemolque;
	private static Proyecto proyecto;
	//Esta variable guarda la maxima carga que se usará para definir la escala al
dibujar
	private static double maximaCarga=0;
	private static double maximaCargaCabeza=0;
	private static double maximaCargaRemolque=0;
	//Esta variable indica cada cuantos puntos tomo uno para dibujar. Un valor mas
alto implica mas precision pero mas tiempo de procesamiento
	private static int puntosGrafico=1;
	private static boolean zonaCabeza=true;

	//Preparamos todo llamando a los distintos métodos
	public static void prepara() {
		introduceCargas();
		generaPuntosMomentos();
	}

	//Este metodo asigna el proyecto
	public static void setProyecto(Proyecto p) {
		proyecto=p;
	}

	//Este método introduce todas las cargas una a una en el proyecto
	public static void introduceCargas() {
		if(proyecto!=null) {
		if(proyecto.getCamion().getTipo().getTipo()=='R') {
			cargas=proyecto.getCargas();
			//Introducimos el origen, las reacciones y el final
			inicializa();
			Iterator<Carga> it = cargas.iterator();
			while(it.hasNext()) {
				Carga q=it.next();
				momentos.add(q);
			}
			generaPosicionesClave();
		}
		if(proyecto.getCamion().getTipo().getTipo()=='A') {
			cargas=proyecto.getCargas();
			//Introducimos el origen, las reacciones y el final
			double
posicionCaja=CalculoReaccionesArticulado.getDistanciaMinima(proyecto);
			inicializa();
			Iterator<Carga> it = cargas.iterator();
			while(it.hasNext()) {
				Carga q=it.next();
				if(q.getPuntoInicio()<posicionCaja)
					momentosCabeza.add(q);
				else
```

```
                                momentosRemolque.add(q);
                        }
                        generaPosicionesClave();
                }
                }
        }

        //Este metodo inicializa los cortantes en los puntos representatios
        public static void inicializa() {
                if(proyecto.getCamion().getTipo().getTipo()=='R') {
                        momentos=new ArrayList<Carga>();
                        momentos.add(new CargaPuntual("Inicio",0,0,false));
                        momentos.add(new
CargaPuntual("Ra",proyecto.getCamion().getdistFrenteEjeDelantero(),-
CalculoReaccionesRigido.getReaccionA(),false));
                        momentos.add(new
CargaPuntual("Rb",proyecto.getCamion().getdistFrenteEjeDelantero()
+proyecto.getCamion().getdistEjeDelanteroEjeTrasero(),-
CalculoReaccionesRigido.getReaccionB(),false));
                        momentos.add(new
CargaPuntual("Fin",proyecto.getCamion().getLargo(),0,false));
                }
                //Si es articulado lo preparamos metemos las reacciones y cargas extra
                if(proyecto.getCamion().getTipo().getTipo()=='A') {
                        double
distanciaKingPin=CalculoReaccionesArticulado.getDistanciaKingPin(proyecto);
                        double cargaKingPin=CalculoReaccionesArticulado.getReaccionKingPin();
                        double
distanciaRemolque=CalculoReaccionesArticulado.getDistanciaRemolque(proyecto);
                        double
cargaRemolque=CalculoReaccionesArticulado.getReaccionRemolque();
                        momentosCabeza=new ArrayList<Carga>();
                        momentosRemolque=new ArrayList<Carga>();
                        momentosCabeza.add(new CargaPuntual("Inicio",0,0,false));
                        momentosCabeza.add(new
CargaPuntual("Ra",proyecto.getCamion().getdistFrenteEjeDelantero(),-
CalculoReaccionesArticulado.getReaccionA(),false));
                        momentosCabeza.add(new
CargaPuntual("KingPin",distanciaKingPin,cargaKingPin,false));
                        momentosCabeza.add(new
CargaPuntual("Rb",proyecto.getCamion().getdistFrenteEjeDelantero()
+proyecto.getCamion().getdistEjeDelanteroEjeTrasero(),-
CalculoReaccionesArticulado.getReaccionB(),false));
                        //Vamos a considerar que la cabeza tractora termina 500 mm detras del
King Pin
                        momentosCabeza.add(new
CargaPuntual("Fin",proyecto.getCamion().getdistFrenteEjeDelantero()
+proyecto.getCamion().getdistEjeDelanteroEjeTrasero()+500,0,false));
                        momentosRemolque.add(new
CargaPuntual("Inicio",CalculoReaccionesArticulado.getDistanciaMinima(proyecto),0,false));
                        momentosRemolque.add(new CargaPuntual("KingPin",distanciaKingPin,-
cargaKingPin,false));
                        momentosRemolque.add(new CargaPuntual("Rc",distanciaRemolque,-
cargaRemolque,false));
                        momentosRemolque.add(new
CargaPuntual("Fin",proyecto.getCamion().getLargo(),0,false));
                }
        }

        //Este metodo devuele la carga cortante en una posicion del torque
        public static double calculaMomento(double posicion,ArrayList<Carga> alc) {
                double carga=0;
                Iterator<Carga> it = alc.iterator();
                while(it.hasNext()) {
                        Carga q=it.next();
                        if(q.getTipo().getTipoCarga()=='P') {
                                if(q.getPuntoInicio()<=posicion) {
```

```
                                carga=carga+q.getValor()*(posicion-q.getPuntoInicio());
                        }
                    } else {
                        CargaDistribuida cd=(CargaDistribuida)q;
                        double valorCarga=cd.getValor();

if((posicion>=cd.getPuntoInicio())&&(posicion<=(cd.getLongitud()+cd.getPuntoInicio()))) {
                                carga=carga+valorCarga*(posicion-
cd.getPuntoInicio())*(posicion-cd.getPuntoInicio())/2;
                        } else if(posicion>(cd.getPuntoInicio()+cd.getLongitud())){
                                carga=carga+valorCarga*cd.getLongitud()*(posicion-
(cd.getPuntoInicio()+cd.getLongitud()/2));
                        }
                    }
                }
                carga=(double) carga;
                return carga;
        }

        //Este método genera las posiciones clave en la viga del camion
        public static void generaPosicionesClave() {
                if(proyecto.getCamion().getTipo().getTipo()=='R') {
                        posiciones=new ArrayList<Double>();
                        Iterator<Carga> it = momentos.iterator();
                        while(it.hasNext()) {
                                Carga q=it.next();
                                if(q.getTipo().getTipoCarga()=='P') {
                                        //Metemos un punto extra antes para poder ver como
evoluciona en el cambio
                                        if(q.getPuntoInicio()!=0)
posiciones.add(q.getPuntoInicio()-0.1);
                                        posiciones.add(q.getPuntoInicio());
                                        //Metemos un punto extra despues para poder ver como
evoluciona en el cambio
                                        if(q.getPuntoInicio()!=proyecto.getCamion().getLargo())
posiciones.add(q.getPuntoInicio()+0.1);
                                } else {
                                        CargaDistribuida cd=(CargaDistribuida)q;
                                        //Añadimos mas puntos para ver el cambio con la
longitud
                                        double tramoCarga=cd.getLongitud()/20;
                                        for(int tramos=1;tramos<20;tramos++) {
                                                posiciones.add(cd.getPuntoInicio()+
(tramoCarga*tramos));
                                        }
                                        //Metemos el punto inicial y final
                                        posiciones.add(cd.getPuntoInicio());
                                        posiciones.add(cd.getPuntoInicio()+cd.getLongitud());
                                }
                        }
                        //Ordenamos el ArrayList
                        posiciones.sort(Comparator.naturalOrder());
                }
                if(proyecto.getCamion().getTipo().getTipo()=='A') {
                        posicionesCabeza=new ArrayList<Double>();
                        Iterator<Carga> it = momentosCabeza.iterator();
                        while(it.hasNext()) {
                                Carga q=it.next();
                                if(q.getTipo().getTipoCarga()=='P') {
                                        //Metemos un punto extra antes para poder ver como
evoluciona en el cambio
                                        if(q.getPuntoInicio()!=0)
posicionesCabeza.add(q.getPuntoInicio()-0.1);
                                        posicionesCabeza.add(q.getPuntoInicio());
                                        //Metemos un punto extra despues para poder ver como
evoluciona en el cambio
```

```
                                if(q.getPuntoInicio()!=proyecto.getCamion().getLargo())
posicionesCabeza.add(q.getPuntoInicio()+0.1);
                        } else {
                                CargaDistribuida cd=(CargaDistribuida)q;
                                //Añadimos mas puntos para ver el cambio con la
longitud
                                double tramoCarga=cd.getLongitud()/20;
                                for(int tramos=1;tramos<20;tramos++) {
                                        posicionesCabeza.add(cd.getPuntoInicio()+
(tramoCarga*tramos));
                                }
                                //Metemos el punto inicial y final
                                posicionesCabeza.add(cd.getPuntoInicio());
                                posicionesCabeza.add(cd.getPuntoInicio()
+cd.getLongitud());
                        }
                }
                //Ordenamos el ArrayList
                posicionesCabeza.sort(Comparator.naturalOrder());

                posicionesRemolque=new ArrayList<Double>();
                Iterator<Carga> itr = momentosRemolque.iterator();
                while(itr.hasNext()) {
                        Carga q=itr.next();
                        if(q.getTipo().getTipoCarga()=='P') {
                                //Metemos un punto extra antes para poder ver como
evoluciona en el cambio
                                if(q.getPuntoInicio()!=0)
posicionesRemolque.add(q.getPuntoInicio()-0.1);
                                posicionesRemolque.add(q.getPuntoInicio());
                                //Metemos un punto extra despues para poder ver como
evoluciona en el cambio
                                if(q.getPuntoInicio()!=proyecto.getCamion().getLargo())
posicionesRemolque.add(q.getPuntoInicio()+0.1);
                        } else {
                                CargaDistribuida cd=(CargaDistribuida)q;
                                //Añadimos mas puntos para ver el cambio con la
longitud
                                double tramoCarga=cd.getLongitud()/20;
                                for(int tramos=1;tramos<20;tramos++) {
                                        posicionesRemolque.add(cd.getPuntoInicio()+
(tramoCarga*tramos));
                                }
                                //Metemos el punto inicial y final
                                posicionesRemolque.add(cd.getPuntoInicio());
                                posicionesRemolque.add(cd.getPuntoInicio()
+cd.getLongitud());
                        }
                }
                //Ordenamos el ArrayList
                posicionesRemolque.sort(Comparator.naturalOrder());
        }
    }

    //Este metodo genera los pares (posicion,cortante)
    public static void generaPuntosMomentos() {
        if(proyecto.getCamion().getTipo().getTipo()=='R') {
                double carga=0;
                Iterator<Double> itpos=posiciones.iterator();
                puntosCambioMomento=new ArrayList<PuntosCalculo>();
                while(itpos.hasNext()) {
                        Double punto = itpos.next();
                        carga=calculaMomento(punto,momentos);
                        puntosCambioMomento.add(new PuntosCalculo(punto,carga));
                }
        }
        if(proyecto.getCamion().getTipo().getTipo()=='A') {
```

```
                    double cargaCabeza=0;
                    Iterator<Double> itposcab=posicionesCabeza.iterator();
                    puntosCambioMomentoCabeza=new ArrayList<PuntosCalculo>();
                    while(itposcab.hasNext()) {
                           Double punto = itposcab.next();
                           cargaCabeza=calculaMomento(punto,momentosCabeza);
                           puntosCambioMomentoCabeza.add(new
PuntosCalculo(punto,cargaCabeza));
                    }
                    double cargaRemolque=0;
                    Iterator<Double> itposrem=posicionesRemolque.iterator();
                    puntosCambioMomentoRemolque=new ArrayList<PuntosCalculo>();
                    while(itposrem.hasNext()) {
                           Double punto = itposrem.next();
                           cargaRemolque=calculaMomento(punto,momentosRemolque);
                           puntosCambioMomentoRemolque.add(new
PuntosCalculo(punto,cargaRemolque));
                    }
             }
      }

      //Devuelve el camion por trozos iguales y saca los valores para cada punto
devolviendolos
      public static ArrayList<PuntosCalculo> puntosGrafico(Proyecto p) {
             proyecto=p;
             ArrayList<PuntosCalculo> listaPuntos = new ArrayList<PuntosCalculo>();
             //Reseteamos el momento que nos sirve para escalar el gráfico
             maximaCarga=0;
             maximaCargaCabeza=0;
             maximaCargaRemolque=0;
             prepara();
             double carga=0;
             int i=0;
             if(proyecto.getCamion().getTipo().getTipo()=='R') {
                    while (i<=proyecto.getCamion().getLargo()){
                           carga=carga+calculaMomento(i,momentos);
                           proyecto.getCamion().setTensionNormalViga(carga);
                           if(Math.abs(carga)>Math.abs(maximaCarga))
                                  maximaCarga=carga;
                                  proyecto.getCamion().setTensionNormalViga(maximaCarga);
                                  proyecto.getCamion().setTensionVonMisses();
                           PuntosCalculo puntoC=new PuntosCalculo(i,carga);
                           i=i+puntosGrafico;
                           carga=0;
                           listaPuntos.add(puntoC);
                    }
                    return listaPuntos;
             } else {
                    //Obtenemos cuanta longitud consideramos despues del eje trasero
                    double parteExtra=DatosGenericos.getLongExtra();
                    if(zonaCabeza) {
                           double
distanciaAConsiderar=((CamionArticulado)proyecto.getCamion()).getdistFrenteEjeDelantero()
+

((CamionArticulado)proyecto.getCamion()).getdistEjeDelanteroEjeTrasero()+parteExtra;
                           while (i<=((CamionArticulado)proyecto.getCamion()).getLargo())
{
                                  if((i<=distanciaAConsiderar)&&(i>=0)){
                                         carga=carga+calculaMomento(i,momentosCabeza);
                                         if(Math.abs(carga)>Math.abs(maximaCargaCabeza)) {
                                                maximaCargaCabeza=carga;

((CamionArticulado)proyecto.getCamion()).setTensionNormalViga(maximaCargaCabeza);

((CamionArticulado)proyecto.getCamion()).setTensionVonMisses();
                                         }
```

```
                                        PuntosCalculo puntoC=new PuntosCalculo(i,carga);
                                        carga=0;
                                        listaPuntos.add(puntoC);
                                  }
                                  i=i+puntosGrafico;
                            }
                            return listaPuntos;
                      } else {
                            double
distanciaAConsiderar=proyecto.getCamion().getdistFrenteEjeDelantero()+
((CamionArticulado)proyecto.getCamion()).getDistanciaEjeDelanteroKingPin()-parteExtra;
                            while (i<=proyecto.getCamion().getLargo()){
                                  if(i>distanciaAConsiderar) {
                                        carga=carga+calculaMomento(i,momentosRemolque);
                                        if(Math.abs(carga)>Math.abs(maximaCargaRemolque))
{
                                              maximaCargaRemolque=carga;

((CamionArticulado)proyecto.getCamion()).setTensionNormalVigaRemolque(maximaCargaRemolque
);

((CamionArticulado)proyecto.getCamion()).setTensionVonMissesVigaRemolque();

                                        }
                                        PuntosCalculo puntoC=new PuntosCalculo(i,carga);
                                        carga=0;
                                        listaPuntos.add(puntoC);
                                  }
                                  i=i+puntosGrafico;
                            }
                            return listaPuntos;
                      }
                }
          }

          //Obtenemos la maxima carga de camion
          public static double getMaximaCarga() {
                return maximaCarga;
          }

          //Reseteamos la maxima carga de camion
          public static void resetMaximaCarga() {
                maximaCarga=0;
          }

          //Obtenemos la maxima carga de cabeza
          public static double getMaximaCargaCabeza() {
                return maximaCargaCabeza;
          }

          //Reseteamos la maxima carga de cabeza
          public static void resetMaximaCargaCabeza() {
                maximaCargaCabeza=0;
          }

          //Obtenemos la maxima carga de remolque
          public static double getMaximaCargaRemolque() {
                return maximaCargaRemolque;
          }

          //Reseteamos la maxima carga de cabeza
          public static void resetMaximaCargaRemolque() {
                maximaCargaRemolque=0;
          }

          //marcamos la zona del camion donde queremos calcular en el articulado
          public static void setParteCabeza(boolean b) {
```

```
			zonaCabeza=b;
		}

		//devolvemos la parte con la que estamos trabajando
		public static boolean getParteCabeza() {
			return zonaCabeza;
		}
		//Este metodo calcula los valores de tensiones y para ello usa tanto la cabeza
como el remolque
		public static void preparaParaTension() {
			//Primero guardamos el valor actual
			boolean zona=zonaCabeza;
			//Hacemos una primera pasada para obtener las reacciones en la zona de
cabeza tractora
			zonaCabeza=true;
			puntosGrafico(proyecto);
			//Hacemos una segunda pasada para obtener las reacciones en la zona de
remolque
			zonaCabeza=false;
			puntosGrafico(proyecto);
			//Reasignamos el valor que tenía zonaCabeza.
			zonaCabeza=zona;
		}
}
```

CLASE PanelCortantes

```
package calculosEstructura;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.Image;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Iterator;
import camiones.*;

import javax.swing.ImageIcon;
import javax.swing.JPanel;

import auxiliares.DatosGenericos;
import auxiliares.Formato;
import auxiliares.Unidades;

public class PanelCortantes extends JPanel {

    private static final long serialVersionUID = 1L;
       private Image imagen;
    //Definimos variables para
    // distancia de frontal-del-camion a eje-delantero=dFD
       //  distancia del eje delantero a comienzo de carga=dDC
    // distancia de eje-delantero a eje-trasero=dDT
    // distancia de eje-trasero a finalcamion=dTZ
    // distancia de iniciocamion a iniciocarga=dIQ
       //     distancia eje delantero tractora a King Pin=dDK
       //  distancia de kingpin a eje de remolque=dKR
    // Longitud-camion=LC
    private double dFD,dDT,dDK,dKR,LC;
    private static Proyecto proyecto;
    private static int alturaEsquema=115;

    //Generamos el constructor por defecto
    public PanelCortantes() {
    }

    //Generamos el constructor con imagen de Jpanel y con el proyecto que estamos usando
    public PanelCortantes(String nombreImagen,Proyecto p) {
        if (nombreImagen != null) {
            imagen = new ImageIcon(
                            getClass().getResource(nombreImagen)
                            ).getImage();
        }
        //Usamos el proyecto como estatico
        proyecto=p;
    }

    //Este metodo asigna el proyecto
    public void setProyecto(Proyecto p) {
       proyecto=p;
    }

    //Este metodo asigna a la variable imagen el icono
    public void setImagen(String nombreImagen) {
        if (nombreImagen != null) {
            imagen = new ImageIcon(
                   getClass().getResource(nombreImagen)
                   ).getImage();
        } else {
            imagen = null;
        }
        repaint();
    }
```

```
    //Este metodo dibuja el camion en esquema
    public void dibuja(Graphics g) {
       //dibujamos el GRAFICO

        //Obtenemos el camion y sus propiedades
        Camion cam = proyecto.getCamion();
        //Asignamos los valores de las dimensiones del camion para el dibujo
        LC=cam.getLargo();
        dFD=cam.getdistFrenteEjeDelantero();
        dDT=cam.getdistEjeDelanteroEjeTrasero();
        //Si es articulado cogemos tambien las medidas necesarias
        if(cam.getTipo().getTipo()=='A') {
              CamionArticulado ca = (CamionArticulado)cam;
              dDK=ca.getDistanciaEjeDelanteroKingPin();
              dKR=ca.getDistanciaKingPinEjeRemolque();
        }
        else {
              dDK=0;
              dKR=0;
        }
       //Tomamos la medida del area a dibujar ancho y alto
              double largo = this.getSize().width;
              //Usamos el siguiente valor de char para saber que tipo de camion tengo que
dibujar. Articulado o Rigido
              char c = (proyecto.getCamion().getTipo()).getTipo();

              //Calculamos la escala que queremos para que la imagen sea más pequeña que
el JPanel
              //Haremos que la imagen sea el tamaño del JPanel entre 1.5
              double largoPreferido=largo/1.15;

              //Calculamos la escala
              double largoCamion=LC;
              double escala=largoCamion/largoPreferido;

              //Calculamos las nuevas medidas para dibujarlo correctamente
              double dFDesc=dFD/escala;
              double dDTesc=dDT/escala;
              double dDKesc=dDK/escala;
              double dKResc=dKR/escala;
              double largoesc=largoCamion/escala;


              //Seleccionamos el punto para dibujar
              double x=(largo-largoesc)/2;

              //Generamos un formateador para los Strings
              DecimalFormat df = new DecimalFormat("0.00");

              //Dividimos entre articulado o rigido
              //RIGIDO
              if (c=='R') {
              //Dibujamos la VIGA ESTRUCTURA
               Formato.setFormatoCotas(g);
                     Formato.setColorRelleno(g);
               g.fillRect((int)x,alturaEsquema,(int)(largoesc),1);
               Formato.setColorLinea(g);
               g.drawRect((int)x,alturaEsquema,(int)(largoesc),1);
               //Dibujamos los APOYOS de la viga
                     Formato.setColorRelleno(g);
               g.fillOval((int)(x+dFDesc-2), alturaEsquema, 4, 4);
               g.fillOval((int)(x+dFDesc+dDTesc-2), alturaEsquema, 4, 4);
               Formato.setColorLinea(g);
               g.drawOval((int)(x+dFDesc-2), alturaEsquema, 4, 4);
               g.drawOval((int)(x+dFDesc+dDTesc-2), alturaEsquema, 4, 4);
```

```
                //Dibujamos LAS LEYES DE ESFUERZOS CORTANTES de la VIGA de la estructura
                Formato.setFormatoCotas(g);
                Formato.setFuenteCargas(g);
                Formato.setFuenteEjes(g);
                //Obtenemos los puntos para dibujar
                ArrayList<PuntosCalculo>
listaPuntos=LeyesCortantes.puntosGrafico(proyecto);
                //Obtenemos la máxima carga para poder escalar convenientemente el gráfico
en eje Y
                double maxCarga=LeyesCortantes.getMaximaCarga();
                //Ajustamos la escala al tamaño del gráfico
                double escalaY=70/Math.abs(maxCarga);
                //Dibujamos cada punto
                //Primero usamos un booleano para que nos ayude a mostrar el valor sólo
cuando cambie la tendencia (de incremento a decremento y viceversa)
                boolean ascendente=false;
                boolean descendente=false;
                boolean mantenido=false;
                boolean tendenciaascendente=false;
                boolean tendenciadescendente=false;
                boolean tendenciamantenida=false;
                Iterator<PuntosCalculo> itp=listaPuntos.iterator();
                PuntosCalculo p1=null;
                if(itp.hasNext()) p1=itp.next();
                while(itp.hasNext()) {
                      PuntosCalculo p2=itp.next();
                      if(p2.getY()>p1.getY()) {
                            if(ascendente) {
                                  tendenciaascendente=true;
                                  tendenciadescendente=false;
                                  tendenciamantenida=false;
                            }
                            ascendente=true;
                            mantenido=false;
                            descendente=false;
                      }
                      if(p2.getY()<p1.getY()) {
                            if(descendente) {
                                  tendenciadescendente=true;
                                  tendenciaascendente=false;
                                  tendenciamantenida=false;
                            }
                            ascendente=false;
                            mantenido=false;
                            descendente=true;
                      }
                      if(p2.getY()==p1.getY()) {
                            if(mantenido) {
                                  tendenciadescendente=false;
                                  tendenciaascendente=false;
                                  tendenciamantenida=true;
                            }
                            ascendente=false;
                            descendente=false;
                            mantenido=true;
                      }
                      if((tendenciadescendente&&(ascendente||mantenido))||
(tendenciaascendente&&(descendente||mantenido))||(tendenciamantenida&&(ascendente||
descendente))) {
                            Integer v=(int)Math.round(p2.getY());
                            v=(int)Unidades.getCargaEnUnidadesActuales(v);
                            g.drawString(df.format(v),(int)(x+(p2.getX()/escala)),(int)
(alturaEsquema-(p2.getY()*escalaY)));
                      }
                      g.drawLine((int)(x+(p1.getX()/escala)), (int)(alturaEsquema-
(p1.getY()*escalaY)), (int)(x+(p2.getX()/escala)),(int)(alturaEsquema-
(p2.getY()*escalaY)));
```

```
                    p1=p2;
                }

                //Dibujamos los ejes
                Formato.setFormatoCotas(g);
                Formato.setColorLinea(g);
                Formato.setFuenteEjes(g);
                double gap=40;
                double puntoEjeSup=gap;
                double puntoEjeInf=2*alturaEsquema-gap;
                g.drawLine((int)x,(int)puntoEjeInf,(int)x,(int)puntoEjeSup);
                //Marcamos las divisiones por eje y con ese valor la longitud entre marcas
                double marcasEje=5;
                double longitudsup=(alturaEsquema-puntoEjeSup)/marcasEje;
                double longitudinf=(puntoEjeInf-alturaEsquema)/marcasEje;
                //Para que inicialmente ponga valores, si no hay cargas ponemos el valor
de 1000
                if(maxCarga==0) maxCarga=1000;
                double valor=(Math.round(maxCarga*100)/100)/marcasEje;
                //Dibujamos 5 puntos en cada lado de corte
                for(int i=1;i<=marcasEje;i++) {
                      g.drawLine((int)(x-2),(int)(alturaEsquema-longitudsup*i),(int)
(x+2),(int)(alturaEsquema-longitudsup*i));
                      g.drawLine((int)(x-2),(int)(alturaEsquema+longitudinf*i),(int)
(x+2),(int)(alturaEsquema+longitudinf*i));
                      g.drawString(df.format((int)
(Math.round(Unidades.getCargaEnUnidadesActuales(valor*i)))),(int)(x+2),(int)
(alturaEsquema-longitudsup*i));
                      g.drawString(df.format((int)
(Math.round(Unidades.getCargaEnUnidadesActuales(-valor*i)))),(int)(x+2),(int)
(alturaEsquema+longitudinf*i));
                }

             }
          //ARTICULADO
             else if(c=='A') {
                     //Vamos a ver con que parte trabajamos
                    boolean zonaCabeza=LeyesCortantes.getParteCabeza();
                    //Obtenemos la parte extra a añadir a la cabeza tractora tras el eje
trasero
                    int parteExtra=(int) DatosGenericos.getLongExtra();


                    //Dibujamos la VIGA ESTRUCTURA EN FUNCION DE LA ZONA
                    if(zonaCabeza) {
                           //Dibujamos la viga de cabeza
                     Formato.setFormatoCotas(g);
                           Formato.setColorRelleno(g);
                     g.setColor(Color.BLACK);
                     g.fillRect((int)x, alturaEsquema,(int)(dFDesc+dDTesc+parteExtra/
escala), 2);
                     g.setColor(Color.GRAY);
                     Formato.setColorLinea(g);
                     g.drawRect((int)x, alturaEsquema,(int)(dFDesc+dDTesc+parteExtra/
escala), 2);
                     //Dibujamos el KingPin
                     g.setColor(Color.RED);
                     g.fillRect((int)(x+dFDesc+dDKesc),alturaEsquema-2,2,8);
                     //Dibujamos los APOYOS de la viga
                           Formato.setColorRelleno(g);
                     g.fillOval((int)(x+dFDesc-2), alturaEsquema+2, 4, 4);
                     g.fillOval((int)(x+dFDesc+dDTesc-2), alturaEsquema+2, 4, 4);
                     Formato.setColorLinea(g);
                     g.drawOval((int)(x+dFDesc-2), alturaEsquema+2, 4, 4);
                     g.drawOval((int)(x+dFDesc+dDTesc-2), alturaEsquema+2, 4, 4);
                    } else {
                           //Dibujamos la viga del remolque
```

```
                    g.fillRect((int)(x+dFDesc+dDKesc-parteExtra/escala), alturaEsquema,
(int)(largoesc-dFDesc-dDKesc+parteExtra/escala), 2);
                    Formato.setColorLinea(g);
                    g.drawRect((int)(x+dFDesc+dDKesc-parteExtra/escala), alturaEsquema,
(int)(largoesc-dFDesc-dDKesc+parteExtra/escala), 2);
                    Formato.setColorLinea(g);
                    //Dibujamos el KingPin
                    g.setColor(Color.RED);
                    g.fillRect((int)(x+dFDesc+dDKesc),alturaEsquema-2,2,8);
                    //Dibujamos los APOYOS de la viga
                          Formato.setColorRelleno(g);
                    g.fillOval((int)(x+dFDesc+dDKesc+dKResc-2), alturaEsquema+2, 4, 4);
                    Formato.setColorLinea(g);
                    g.drawOval((int)(x+dFDesc+dDKesc+dKResc-2), alturaEsquema+2, 4, 4);
                   }

                   //Dibujamos LAS LEYES DE ESFUERZOS CORTANTES de la VIGA de la
estructura
              Formato.setFormatoCotas(g);
              Formato.setFuenteCargas(g);
              Formato.setFuenteEjes(g);
              //Obtenemos los puntos para dibujar
              ArrayList<PuntosCalculo>
listaPuntos=LeyesCortantes.puntosGrafico(proyecto);
              //Obtenemos la máxima carga para poder escalar convenientemente el gráfico
en eje Y
              double maxCarga;
              if(zonaCabeza) maxCarga=LeyesCortantes.getMaximaCargaCabeza();
              else maxCarga=LeyesCortantes.getMaximaCargaRemolque();
              //Ajustamos la escala al tamaño del gráfico
              double escalaY=70/Math.abs(maxCarga);
              //Dibujamos cada punto
              //Primero usamos un booleano para que nos ayude a mostrar el valor sólo
cuando cambie la tendencia (de incremento a decremento y viceversa)
              boolean ascendente=false;
              boolean descendente=false;
              boolean mantenido=false;
              boolean tendenciaascendente=false;
              boolean tendenciadescendente=false;
              boolean tendenciamantenida=false;
              Iterator<PuntosCalculo> itp=listaPuntos.iterator();
              PuntosCalculo p1=null;
              if(itp.hasNext()) p1=itp.next();
              while(itp.hasNext()) {
                    PuntosCalculo p2=itp.next();
                    if(p2.getY()>p1.getY()) {
                          if(ascendente) {
                                tendenciaascendente=true;
                                tendenciadescendente=false;
                                tendenciamantenida=false;
                          }
                          ascendente=true;
                          mantenido=false;
                          descendente=false;
                    }
                    if(p2.getY()<p1.getY()) {
                          if(descendente) {
                                tendenciadescendente=true;
                                tendenciaascendente=false;
                                tendenciamantenida=false;
                          }
                          ascendente=false;
                          mantenido=false;
                          descendente=true;
                    }
                    if(p2.getY()==p1.getY()) {
                          if(mantenido) {
```

```
                                    tendenciadescendente=false;
                                    tendenciaascendente=false;
                                    tendenciamantenida=true;
                              }
                              ascendente=false;
                              descendente=false;
                              mantenido=true;
                        }
                        if((tendenciadescendente&&(ascendente||mantenido))||
(tendenciaascendente&&(descendente||mantenido))||(tendenciamantenida&&(ascendente||
descendente))) {
                              Integer v=(int)Math.round(p2.getY());
                              v=(int)Unidades.getCargaEnUnidadesActuales(v);
                              g.drawString(df.format(v),(int)(x+(p2.getX()/escala)),(int)
(alturaEsquema-(p2.getY()*escalaY)));
                        }
                        g.drawLine((int)(x+(p1.getX()/escala)), (int)(alturaEsquema-
(p1.getY()*escalaY)), (int)(x+(p2.getX()/escala)),(int)(alturaEsquema-
(p2.getY()*escalaY)));
                        p1=p2;
                  }

                  //Dibujamos los ejes
                  Formato.setFormatoCotas(g);
                  Formato.setColorLinea(g);
                  Formato.setFuenteEjes(g);
                  double gap=40;
                  double puntoEjeSup=gap;
                  double puntoEjeInf=2*alturaEsquema-gap;
                  g.drawLine((int)x,(int)puntoEjeInf,(int)x,(int)puntoEjeSup);
                  //Marcamos las divisiones por eje y con ese valor la longitud entre marcas
                  double marcasEje=5;
                  double longitudsup=(alturaEsquema-puntoEjeSup)/marcasEje;
                  double longitudinf=(puntoEjeInf-alturaEsquema)/marcasEje;
                  //Para que inicialmente ponga valores, si no hay cargas ponemos el valor
de 1000
                  if(maxCarga==0) maxCarga=1000;
                  double valor=(Math.round(maxCarga*100)/100)/marcasEje;
                  //Dibujamos 5 puntos en cada lado de corte
                  for(int i=1;i<=marcasEje;i++) {
                        g.drawLine((int)(x-2),(int)(alturaEsquema-longitudsup*i),(int)
(x+2),(int)(alturaEsquema-longitudsup*i));
                        g.drawLine((int)(x-2),(int)(alturaEsquema+longitudinf*i),(int)
(x+2),(int)(alturaEsquema+longitudinf*i));
                        g.drawString(df.format((int)
(Math.round(Unidades.getCargaEnUnidadesActuales(valor*i)))),(int)(x+2),(int)
(alturaEsquema-longitudsup*i));
                        g.drawString(df.format((int)
(Math.round(Unidades.getCargaEnUnidadesActuales(-valor*i)))),(int)(x+2),(int)
(alturaEsquema+longitudinf*i));
                  }

               }
    }



    @Override
    public void paint(Graphics g) {
        if (imagen != null) {
            g.drawImage(imagen, 0, 0, getWidth(), getHeight(),this);
            setOpaque(false);
            dibuja(g);
        } else {
            setOpaque(true);
        }
```

```
        super.paint(g);
    }

}
```

CLASE PanelEsquema

```
package calculosEstructura;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.Image;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Iterator;
import camiones.*;

import javax.swing.ImageIcon;
import javax.swing.JPanel;

import auxiliares.Formato;
import auxiliares.Unidades;
import cargasEstructura.Carga;
import cargasEstructura.CargaDistribuida;

public class PanelEsquema extends JPanel {

    private static final long serialVersionUID = 1L;
       private Image imagen;
    //Definimos variables para
    // distancia de frontal-del-camion a eje-delantero=dFD
       //  distancia del eje delantero a comienzo de carga=dDC
    // distancia de eje-delantero a eje-trasero=dDT
    // distancia de eje-trasero a finalcamion=dTZ
    // distancia de iniciocamion a iniciocarga=dIQ
       //     distancia eje delantero tractora a King Pin=dDK
       //  distancia de kingpin a eje de remolque=dKR
    // Longitud-camion=LC
    private double dFD,dDC,dDT,dTZ,dIQ,dDK,dKR,LC;
    private static Proyecto proyecto;
    private static int alturaEsquema=115;

    //Generamos el constructor por defecto
    public PanelEsquema() {
    }

    //Generamos el constructor con imagen de Jpanel y con el proyecto que estamos usando
    public PanelEsquema(String nombreImagen,Proyecto p) {
        if (nombreImagen != null) {
            imagen = new ImageIcon(
                           getClass().getResource(nombreImagen)
                           ).getImage();
        }
        //Guardamos el proyecto
        proyecto=p;
    }

    //Este metodo asigna el proyecto
    public void setProyecto(Proyecto p) {
       proyecto=p;
    }

    //Este metodo asigna a la variable imagen el icono
    public void setImagen(String nombreImagen) {
        if (nombreImagen != null) {
            imagen = new ImageIcon(
                   getClass().getResource(nombreImagen)
                   ).getImage();
        } else {
            imagen = null;
        }
```

```
        repaint();
    }

    //Este metodo dibuja el camion en esquema
    public void dibuja(Graphics g) {
       //Usamos el proyecto como estatico
        //Obtenemos el camion y sus caracteristicas
        Camion cam = proyecto.getCamion();
        //Asignamos los valores de las dimensiones del camion para el dibujo
        LC=cam.getLargo();
        dDC=cam.getDistanciaEjeDelanteroComienzoCaja();
        dFD=cam.getdistFrenteEjeDelantero();
        dDT=cam.getdistEjeDelanteroEjeTrasero();
        dTZ=LC-dFD-dDT;
        dIQ=LC-dFD-dDC;
        //Si es articulado cogemos tambien las medidas necesarias
        if(cam.getTipo().getTipo()=='A') {
              CamionArticulado ca = (CamionArticulado)cam;
              dDK=ca.getDistanciaEjeDelanteroKingPin();
              dKR=ca.getDistanciaKingPinEjeRemolque();
        }
        else {
              dDK=0;
              dKR=0;
        }
       //dibujamos el GRAFICO
       //Tomamos la medida del area a dibujar ancho y alto
              double largo = this.getSize().width;
              //Usamos el siguiente valor de char para saber que tipo de camion tengo que
dibujar. Articulado o Rigido
              char c = (proyecto.getCamion().getTipo()).getTipo();

              //Calculamos la escala que queremos para que la imagen sea más pequeña que
el JPanel
              //Haremos que la imagen sea el tamaño del JPanel entre 1.5
              double largoPreferido=largo/1.15;

              //Calculamos la escala
              double largoCamion=LC;
              double escala=largoCamion/largoPreferido;

              //Calculamos las nuevas medidas para dibujarlo correctamente
              double dFDesc=dFD/escala;
              double dDCesc=dDC/escala;
              double dDTesc=dDT/escala;
              double dTZesc=dTZ/escala;
              double dIQesc=dIQ/escala;
              double dDKesc=dDK/escala;
              double dKResc=dKR/escala;
              double largoesc=largoCamion/escala;


              //Seleccionamos el punto para dibujar
              double x=(largo-largoesc)/2;


              //Creamos un formateo para las cotas
        DecimalFormat df = new DecimalFormat("0.00");

              //Dividimos entre articulado o rigido
              //RIGIDO
              if (c=='R') {
              //Dibujamos la VIGA ESTRUCTURA
               Formato.setFormatoCotas(g);
                    Formato.setColorRelleno(g);
               g.fillRect((int)x,alturaEsquema,(int)(largoesc),10);
               Formato.setColorLinea(g);
```

```
                g.drawRect((int)x,alturaEsquema,(int)(largoesc),10);
                //Dibujamos los APOYOS de la viga
                     Formato.setColorRelleno(g);
                g.fillOval((int)(x+dFDesc-5), alturaEsquema+10, 10, 10);
                g.fillOval((int)(x+dFDesc+dDTesc-5), alturaEsquema+10, 10, 10);
                Formato.setColorLinea(g);
                g.drawOval((int)(x+dFDesc-5), alturaEsquema+10, 10, 10);
                g.drawOval((int)(x+dFDesc+dDTesc-5), alturaEsquema+10, 10, 10);
                //Dibujamos los textos de los apoyos
                CamionRigido cr=(CamionRigido)proyecto.getCamion();
                Formato.setFuenteCargas(g);
                g.drawString("APOYO-A ("+cr.getEjesPorTandemDelantero()+" eje/s)", (int)
(x+dFDesc-5), alturaEsquema+35);
                g.drawString("APOYO-B ("+cr.getEjesPorTandemTrasero()+" eje/s)", (int)
(x+dFDesc+dDTesc-5), alturaEsquema+35);

                //Dibujamos las cotas de la VIGA de la estructura
                Formato.setFormatoCotas(g);
                Formato.setFuenteCargas(g);
                //del frente al eje delantero
                g.fillOval((int)x, alturaEsquema+50, 5, 5);
                g.drawString(df.format(Unidades.getLongitudEnUnidadesActuales(dFD)), (int)
(x+(dFDesc/2)-10), alturaEsquema+50);
                g.fillOval((int)(x+dFDesc), alturaEsquema+50, 5, 5);
                //del eje delantero al trasero
                g.drawString(df.format(Unidades.getLongitudEnUnidadesActuales(dDT)), (int)
(x+dFDesc+(dDTesc/2)-10), alturaEsquema+50);
                g.fillOval((int)(x+dFDesc+dDTesc), alturaEsquema+50, 5, 5);
                //del eje trasero al final
                g.drawString(df.format(Unidades.getLongitudEnUnidadesActuales(dTZ)), (int)
(x+dFDesc+dDTesc+(dTZesc/2)-10), alturaEsquema+50);
                g.fillOval((int)(x+largoesc), alturaEsquema+50, 5, 5);
                g.fillOval((int)x, alturaEsquema+62, 5, 5);
                g.fillOval((int)(x+largoesc), alturaEsquema+62, 5, 5);
                //longitud total del camion
                g.drawString(df.format(Unidades.getLongitudEnUnidadesActuales(LC)), (int)
(x+((LC/escala)/2)-10), alturaEsquema+62);
                g.drawLine((int)(x+2.5),(int)(alturaEsquema+62+2.5),(int)(x+largoesc+2.5),
(int)(alturaEsquema+62+2.5));
                g.drawLine((int)(x+2.5),(int)(alturaEsquema+50+2.5),(int)(x+largoesc+2.5),
(int)(alturaEsquema+50+2.5));//Dibujamos la cota de la longitud de la caja del camion
                //Dibujamos la cota del largo de la caja
                g.fillOval((int)(x+dFDesc+dDCesc-2.5), alturaEsquema-80, 5, 5);
                g.fillOval((int)(x+largoesc-2.5), alturaEsquema-80, 5, 5);
                g.drawLine((int)(x+dFDesc+dDCesc),(int)(alturaEsquema-80+2.5),(int)
(x+largoesc),(int)(alturaEsquema-80+2.5));
                g.drawString(df.format(Unidades.getLongitudEnUnidadesActuales(LC-dFD-
dDC)), (int)(x+dFDesc+dDCesc+(largoesc-dFDesc-dDCesc)/2), alturaEsquema-80);

           //ARTICULADO
              } else if(c=='A') {
                     //Dibujamos la VIGA ESTRUCTURA
                Formato.setFormatoCotas(g);
                     Formato.setColorRelleno(g);
                g.fillRect((int)x,alturaEsquema,(int)(largoesc),10);
                //Dibujamos la divison entre parte de la cabeza tractora y remolque
                g.setColor(Color.BLACK);
                g.fillRect((int)x, alturaEsquema+5,(int)(dFDesc+dDTesc+500/escala), 5);
                g.setColor(Color.GRAY);
                g.fillRect((int)(x+dFDesc+dDKesc-300/escala), alturaEsquema,(int)
(largoesc-dFDesc-dDKesc+300/escala), 5);
                Formato.setColorLinea(g);
                g.drawRect((int)x, alturaEsquema+5,(int)(dFDesc+dDTesc+500/escala), 5);
                g.drawRect((int)(x+dFDesc+dDKesc-300/escala), alturaEsquema,(int)
(largoesc-dFDesc-dDKesc+300/escala), 5);
                Formato.setColorLinea(g);
                g.drawRect((int)x,alturaEsquema,(int)(largoesc),10);
```

```
                //Dibujamos el KingPin
                g.setColor(Color.RED);
                g.fillRect((int)(x+dFDesc+dDKesc),alturaEsquema,2,10);
                //Dibujamos los APOYOS de la viga
                     Formato.setColorRelleno(g);
                g.fillOval((int)(x+dFDesc-5), alturaEsquema+10, 10, 10);
                g.fillOval((int)(x+dFDesc+dDTesc-5), alturaEsquema+10, 10, 10);
                g.fillOval((int)(x+dFDesc+dDKesc+dKResc-5), alturaEsquema+10, 10, 10);
                Formato.setColorLinea(g);
                g.drawOval((int)(x+dFDesc-5), alturaEsquema+10, 10, 10);
                g.drawOval((int)(x+dFDesc+dDTesc-5), alturaEsquema+10, 10, 10);
                g.drawOval((int)(x+dFDesc+dDKesc+dKResc-5), alturaEsquema+10, 10, 10);
                //Dibujamos los textos de los apoyos
                CamionArticulado ca=(CamionArticulado)proyecto.getCamion();
                Formato.setFuenteCargas(g);
                g.drawString("APOYO-A ("+ca.getEjesPorTandemDelantero()+" eje/s)", (int)
(x+dFDesc-5), alturaEsquema+35);
                g.drawString("APOYO-B ("+ca.getEjesPorTandemTrasero()+" eje/s)", (int)
(x+dFDesc+dDTesc-5), alturaEsquema+35);
                g.drawString("APOYO-C ("+ca.getEjesPorTandemRemolque()+" eje/s)", (int)
(x+dFDesc+dDKesc+dKResc-5), alturaEsquema+35);

                //Dibujamos las cotas de la VIGA de la estructura
                Formato.setFormatoCotas(g);
                Formato.setFuenteCargas(g);
                //Largo de la caja del remolque
                g.fillOval((int)(x+dFDesc+dDCesc-2.5), alturaEsquema-80, 5, 5);
                g.fillOval((int)(x+largoesc-2.5), alturaEsquema-80, 5, 5);
                g.drawLine((int)(x+dFDesc+dDCesc),(int)(alturaEsquema-80+2.5),(int)
(x+largoesc),(int)(alturaEsquema-80+2.5));
                g.drawString(df.format(Unidades.getLongitudEnUnidadesActuales(LC-dFD-
dDC)), (int)(x+dFDesc+dDCesc+(largoesc-dFDesc-dDCesc)/2), alturaEsquema-80);
                //Cotas bajo el dibujo
                g.fillOval((int)x, alturaEsquema+50, 5, 5);
                //Distancia frontal delantero
                g.drawString(df.format(Unidades.getLongitudEnUnidadesActuales(dFD)), (int)
(x+(dFDesc/2)-10), alturaEsquema+50);
                g.fillOval((int)(x+dFDesc), alturaEsquema+50, 5, 5);
                //Distancia Eje Delantero a Eje Trasero Tractora
                g.drawString(df.format(Unidades.getLongitudEnUnidadesActuales(dDT)), (int)
(x+dFDesc+(dDTesc/2)-10), alturaEsquema+50);
                //Distancia Eje trasero a Eje Remolque
                g.fillOval((int)(x+dFDesc+dDTesc), alturaEsquema+50, 5, 5);
                g.drawString(df.format(Unidades.getLongitudEnUnidadesActuales(dDK+dKR-
dDT)), (int)(x+dFDesc+dDTesc+((dFDesc+dDKesc+dKResc)-(dFDesc+dDTesc))/2-10),
alturaEsquema+50);
                //Eje remolque a final
                g.fillOval((int)(x+dFDesc+dDKesc+dKResc), alturaEsquema+50, 5, 5);
                g.drawString(df.format(Unidades.getLongitudEnUnidadesActuales(LC-
(dFD+dDK+dKR))), (int)(x+largoesc-(largoesc-(dFDesc+dDKesc+dKResc))/2-10),
alturaEsquema+50);
                g.fillOval((int)(x+largoesc), alturaEsquema+50, 5, 5);
                //Ponemos la longitud total
                g.fillOval((int)x, alturaEsquema+62, 5, 5);
                g.fillOval((int)(x+largoesc), alturaEsquema+62, 5, 5);
                g.drawString(df.format(Unidades.getLongitudEnUnidadesActuales(LC)), (int)
(x+((LC/escala)/2)-10), alturaEsquema+62);
                g.drawLine((int)(x+2.5),(int)(alturaEsquema+62+2.5),(int)(x+largoesc+2.5),
(int)(alturaEsquema+62+2.5));
                g.drawLine((int)(x+2.5),(int)(alturaEsquema+50+2.5),(int)(x+largoesc+2.5),
(int)(alturaEsquema+50+2.5));
             }

             //Dibujamos la caja del camion y la cabina
        Formato.setColorLinea(g);
        g.drawRect((int)(x+dFDesc+dDCesc),alturaEsquema-60,(int)(largoesc-dFDesc-
dDCesc),60);
```

```
        Formato.setFuenteCargas(g);
        g.drawString("Zona de carga", (int)(x+dFDesc+dDCesc+10), alturaEsquema-65);
        g.drawRect((int)x,alturaEsquema-50,(int)(dFDesc+dDCesc),50);
        g.drawString("Cabina", (int)(x+10), alturaEsquema-55);


             //Dibujamos el origen de coordenadas
             g.setColor(Color.BLACK);
             g.drawLine((int)(x),(int)(alturaEsquema-70+2.5),(int)(x+15),(int)
(alturaEsquema-70+2.5));
             g.drawLine((int)(x+11),(int)(alturaEsquema-73+2.5),(int)(x+15),(int)
(alturaEsquema-70+2.5));
             g.drawLine((int)(x+11),(int)(alturaEsquema-67+2.5),(int)(x+15),(int)
(alturaEsquema-70+2.5));
        g.drawString("X",(int)(x+17),(int)(alturaEsquema-70+2.5));
             g.drawLine((int)(x),(int)(alturaEsquema-70+2.5),(int)(x),(int)
(alturaEsquema-85+2.5));
             g.drawLine((int)(x-3),(int)(alturaEsquema-81+2.5),(int)(x),(int)
(alturaEsquema-85+2.5));
             g.drawLine((int)(x+3),(int)(alturaEsquema-81+2.5),(int)(x),(int)
(alturaEsquema-85+2.5));
        g.drawString("Y",(int)(x-5),(int)(alturaEsquema-85+2.5));


        //Dibujamos las CARGAS del camion
        Formato.setFuenteCargas(g);
        ArrayList<Carga> cargas = proyecto.getCargas();
        Iterator<Carga> it = cargas.iterator();
        Carga q;
        String nombre;
        double dist;
        double distEsc;
        double valor;
        double longitud;
        while(it.hasNext()) {
             q=it.next();
             if(q.getVisible()==true) {
                    //Dibujamos si es puntual
                    if(q.getTipo().getTipoCarga()=='P') {
                       Formato.setColorCargas(g);
                           nombre=q.getNombre();
                           dist=q.getPuntoInicio();
                           valor=Unidades.getCargaEnUnidadesActuales(q.getValor());
                           distEsc=dist/escala;
                            //Puntual Conductor
                            g.drawPolygon(puntosX((int)(x+distEsc),escala), puntosY((int)
(x+distEsc),alturaEsquema,escala), 3);
                             g.fillRect((int)(x+distEsc)-1,alturaEsquema-30,(int)2,30);
                             g.fillPolygon(puntosX((int)(x+distEsc),escala), puntosY((int)
(x+distEsc),alturaEsquema,escala), 3);
                              Formato.setColorTextoCargas(g);
                              g.drawString(nombre+"="+df.format(valor)
+Unidades.getUnidadesCarga(), (int)(x+distEsc),alturaEsquema-35);
                    } else {
                             //Dibujamos la carga distribuida
                       Formato.setColorCargas(g);
                           nombre=q.getNombre();
                           dist=q.getPuntoInicio();
                           valor=Unidades.getCargaEnUnidadesActuales(q.getValor())/
Unidades.getConversionLongitud();
                           //Tomo la longitud de la carga distribuida
                           longitud=((CargaDistribuida)q).getLongitud();
                           //Vemos el punto de inicio de la carga distribuida a escala
                           dIQesc=dist/escala;
                           //Escalamos lal longitud para dibujarla
                           largoesc=longitud/escala;
                           //Calculo el número de cuadrados que deben dibujarse
```

```
                            int cuadrados = (int)((largoesc)/10);
                            //Genero una variable que controlara el dibujo de la
siguiente flechita de la carga distribuida
                            double posicion = x+dIQesc;
                            Formato.setColorTextoCargas(g);
                            //Dibujo el nombre y el valor
                            g.drawString(nombre+"="+df.format(valor)
+Unidades.getUnidadesCarga()+"/"+Unidades.getUnidadesLongitud(), (int)
(posicion),alturaEsquema-25);
                            Formato.setColorCargas(g);
                            //Dibujo cada cuadrado y cada flechita para dibujar la carga
                            for (int i=0;i<cuadrados;i++) {
                                  g.drawRect((int)(posicion),alturaEsquema-20,
(int)10,20);
                                  g.drawLine((int)(posicion),alturaEsquema,(int)
(posicion)-3,alturaEsquema-8);
                                  g.drawLine((int)(posicion),alturaEsquema,(int)
(posicion)+3,alturaEsquema-8);
                                  //Incremento la posicion en 10
                                  posicion=posicion+10;
                            }
                            //Dibujo la ultima flecha despues de pasar por todos los
cuadrados.
                            g.drawLine((int)(posicion),alturaEsquema,(int)
(posicion)-3,alturaEsquema-8);
                            g.drawLine((int)(posicion),alturaEsquema,(int)(posicion)
+3,alturaEsquema-8);
                     }
              }
        }
    }


    //GENERAMOS LOS PUNTOS DE LAS FLECHAS DE LA CARGA PUNTUAL
    //Estos dos metodos servirán para generar los puntos de la flecha
    public int[] puntosX(int x, double escala) {
       int puntos[]=new int[3];
       double x2 = (x+5);
       double x3 = (x-5);
       puntos[0]=x;
       puntos[1]=(int)x2;
       puntos[2]=(int)x3;
       return puntos;
    }
    //Estos dos metodos servirán para generar los puntos de la flecha
    public int[] puntosY(int x,int y,double escala) {
       int puntos[]=new int[3];
       double y2 = (y-10);
       double y3 = (y-10);
       puntos[0]=y;
       puntos[1]=(int)y2;
       puntos[2]=(int)y3;
       return puntos;
    }


    @Override
    public void paint(Graphics g) {
        if (imagen != null) {
            g.drawImage(imagen, 0, 0, getWidth(), getHeight(),this);
            setOpaque(false);
            dibuja(g);
        } else {
            setOpaque(true);
        }

        super.paint(g);
```

```
        }

}
```

CLASE PanelMomentos

```
package calculosEstructura;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.Image;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Iterator;
import camiones.*;

import javax.swing.ImageIcon;
import javax.swing.JPanel;

import auxiliares.DatosGenericos;
import auxiliares.Formato;
import auxiliares.Unidades;

public class PanelMomentos extends JPanel {

    private static final long serialVersionUID = 1L;
       private Image imagen;
    //Definimos variables para
    // distancia de frontal-del-camion a eje-delantero=dFD
       //  distancia del eje delantero a comienzo de carga=dDC
    // distancia de eje-delantero a eje-trasero=dDT
    // distancia de eje-trasero a finalcamion=dTZ
    // distancia de iniciocamion a iniciocarga=dIQ
       //     distancia eje delantero tractora a King Pin=dDK
       //  distancia de kingpin a eje de remolque=dKR
    // Longitud-camion=LC
    private double dFD,dDC,dDT,dTZ,dIQ,dDK,dKR,LC;
    private static Proyecto proyecto;
    private static int alturaEsquema=115;

    //Generamos el constructor por defecto
    public PanelMomentos() {
    }

    //Generamos el constructor con imagen de Jpanel y con el proyecto que estamos usando
    public PanelMomentos(String nombreImagen,Proyecto p) {
        if (nombreImagen != null) {
            imagen = new ImageIcon(
                            getClass().getResource(nombreImagen)
                            ).getImage();
        }
        //Usamos el proyecto como estatico
        proyecto=p;
    }

    //Este metodo asigna el proyecto
    public void setProyecto(Proyecto p) {
       proyecto=p;
    }

    //Este metodo asigna a la variable imagen el icono
    public void setImagen(String nombreImagen) {
        if (nombreImagen != null) {
            imagen = new ImageIcon(
                   getClass().getResource(nombreImagen)
                   ).getImage();
        } else {
            imagen = null;
        }
        repaint();
    }
```

```
    //Este metodo dibuja el camion en esquema
    public void dibuja(Graphics g) {
       //dibujamos el GRAFICO

        //Obtenemos el camion y sus datos
        Camion cam = proyecto.getCamion();
        //Asignamos los valores de las dimensiones del camion para el dibujo
        LC=cam.getLargo();
        dDC=cam.getDistanciaEjeDelanteroComienzoCaja();
        dFD=cam.getdistFrenteEjeDelantero();
        dDT=cam.getdistEjeDelanteroEjeTrasero();
        dTZ=LC-dFD-dDT;
        dIQ=LC-dFD-dDC;
        //Si es articulado cogemos tambien las medidas necesarias
        if(cam.getTipo().getTipo()=='A') {
              CamionArticulado ca = (CamionArticulado)cam;
              dDK=ca.getDistanciaEjeDelanteroKingPin();
              dKR=ca.getDistanciaKingPinEjeRemolque();
        }
        else {
              dDK=0;
              dKR=0;
        }
       //Tomamos la medida del area a dibujar ancho y alto
              double largo = this.getSize().width;
              //Usamos el siguiente valor de char para saber que tipo de camion tengo que
dibujar. Articulado o Rigido
              char c = (proyecto.getCamion().getTipo()).getTipo();

              //Calculamos la escala que queremos para que la imagen sea más pequeña que
el JPanel
              //Haremos que la imagen sea el tamaño del JPanel entre 1.5
              double largoPreferido=largo/1.15;

              //Calculamos la escala
              double largoCamion=LC;
              double escala=largoCamion/largoPreferido;

              //Calculamos las nuevas medidas para dibujarlo correctamente
              double dFDesc=dFD/escala;
              double dDCesc=dDC/escala;
              double dDTesc=dDT/escala;
              double dTZesc=dTZ/escala;
              double dIQesc=dIQ/escala;
              double dDKesc=dDK/escala;
              double dKResc=dKR/escala;
              double largoesc=largoCamion/escala;


              //Seleccionamos el punto para dibujar
              double x=(largo-largoesc)/2;

              //Generamos un formateo para los strings
              DecimalFormat df = new DecimalFormat("0.00");

              //Dividimos entre articulado o rigido
              //RIGIDO
              if (c=='R') {
              //Dibujamos la VIGA ESTRUCTURA
               Formato.setFormatoCotas(g);
                     Formato.setColorRelleno(g);
               g.fillRect((int)x,alturaEsquema,(int)(largoesc),1);
               Formato.setColorLinea(g);
               g.drawRect((int)x,alturaEsquema,(int)(largoesc),1);
               //Dibujamos los APOYOS de la viga
                     Formato.setColorRelleno(g);
```

```
                g.fillOval((int)(x+dFDesc-2), alturaEsquema, 4, 4);
                g.fillOval((int)(x+dFDesc+dDTesc-2), alturaEsquema, 4, 4);
                Formato.setColorLinea(g);
                g.drawOval((int)(x+dFDesc-2), alturaEsquema, 4, 4);
                g.drawOval((int)(x+dFDesc+dDTesc-2), alturaEsquema, 4, 4);
                //Dibujamos los textos de los apoyos
                CamionRigido cr=(CamionRigido)proyecto.getCamion();

              //Dibujamos LAS LEYES DE MOMENTOS FLECTORES de la VIGA de la estructura
                Formato.setFormatoCotas(g);
                Formato.setFuenteCargas(g);
                Formato.setFuenteEjes(g);
                //Obtenemos los puntos para dibujar
                ArrayList<PuntosCalculo>
listaPuntos=LeyesMomentos.puntosGrafico(proyecto);
                //Obtenemos la máxima carga para poder escalar convenientemente el gráfico
en eje Y
                double maxCarga=LeyesMomentos.getMaximaCarga();
                //Ajustamos la escala al tamaño del gráfico
                double escalaY=70/Math.abs(maxCarga);
                //Dibujamos cada punto
                //Primero usamos un booleano para que nos ayude a mostrar el valor sólo
cuando cambie la tendencia (de incremento a decremento y viceversa)
                boolean ascendente=false;
                boolean descendente=false;
                boolean mantenido=false;
                boolean tendenciaascendente=false;
                boolean tendenciadescendente=false;
                boolean tendenciamantenida=false;
                Iterator<PuntosCalculo> itp=listaPuntos.iterator();
                PuntosCalculo p1=null;
                if(itp.hasNext()) p1=itp.next();
                while(itp.hasNext()) {
                      PuntosCalculo p2=itp.next();
                      if(p2.getY()>p1.getY()) {
                            if(ascendente) {
                                  tendenciaascendente=true;
                                  tendenciadescendente=false;
                                  tendenciamantenida=false;
                            }
                            ascendente=true;
                            mantenido=false;
                            descendente=false;
                      }
                      if(p2.getY()<p1.getY()) {
                            if(descendente) {
                                  tendenciadescendente=true;
                                  tendenciaascendente=false;
                                  tendenciamantenida=false;
                            }
                            ascendente=false;
                            mantenido=false;
                            descendente=true;
                      }
                      if(p2.getY()==p1.getY()) {
                            if(mantenido) {
                                  tendenciadescendente=false;
                                  tendenciaascendente=false;
                                  tendenciamantenida=true;
                            }
                            ascendente=false;
                            descendente=false;
                            mantenido=true;
                      }
                      if((tendenciadescendente&&(ascendente||mantenido))||
(tendenciaascendente&&(descendente||mantenido))||(tendenciamantenida&&(ascendente||
descendente))) {
```

```
                        Integer v=(int)Math.round(p2.getY());
                        //Como son momentos tengo que transformar la longitud y la
carga cuando uso las unidades

v=(int)Unidades.getCargaEnUnidadesActuales(Unidades.getLongitudEnUnidadesActuales(v));
                        g.drawString(df.format(v),(int)(x+(p2.getX()/escala)),(int)
(alturaEsquema-(p2.getY()*escalaY)));
                  }
                  g.drawLine((int)(x+(p1.getX()/escala)), (int)(alturaEsquema-
(p1.getY()*escalaY)), (int)(x+(p2.getX()/escala)),(int)(alturaEsquema-
(p2.getY()*escalaY)));
                  p1=p2;
            }

                  //Dibujamos los ejes
            Formato.setFormatoCotas(g);
            Formato.setColorLinea(g);
            Formato.setFuenteEjes(g);
            double gap=40;
            double puntoEjeSup=gap;
            double puntoEjeInf=2*alturaEsquema-gap;
            g.drawLine((int)x,(int)puntoEjeInf,(int)x,(int)puntoEjeSup);
            //Marcamos las divisiones por eje y con ese valor la longitud entre marcas
            double marcasEje=5;
            double longitudsup=(alturaEsquema-puntoEjeSup)/marcasEje;
            double longitudinf=(puntoEjeInf-alturaEsquema)/marcasEje;
            //Para que inicialmente ponga valores, si no hay cargas ponemos el valor
de 1000
            if(maxCarga==0) maxCarga=1000;
            double valor=(Math.round(maxCarga*100)/100)/marcasEje;
            //Dibujamos 5 puntos en cada lado de corte
            for(int i=1;i<=marcasEje;i++) {
                  g.drawLine((int)(x-2),(int)(alturaEsquema-longitudsup*i),(int)
(x+2),(int)(alturaEsquema-longitudsup*i));
                  g.drawLine((int)(x-2),(int)(alturaEsquema+longitudinf*i),(int)
(x+2),(int)(alturaEsquema+longitudinf*i));
                  g.drawString(df.format((int)
(Math.round(Unidades.getCargaEnUnidadesActuales(Unidades.getLongitudEnUnidadesActuales(va
lor*i))))),(int)(x+2),(int)(alturaEsquema-longitudsup*i));
                  g.drawString(df.format((int)
(Math.round(Unidades.getCargaEnUnidadesActuales(Unidades.getLongitudEnUnidadesActuales(-
valor*i))))),(int)(x+2),(int)(alturaEsquema+longitudinf*i));
            }

         }
       //ARTICULADO
         else if(c=='A') {
                  //Vamos a ver con que parte trabajamos
                 boolean zonaCabeza=LeyesMomentos.getParteCabeza();
                 //Obtenemos la parte extra a añadir a la cabeza tractora tras el eje
trasero
                 int parteExtra=(int) DatosGenericos.getLongExtra();

                 //Dibujamos la VIGA ESTRUCTURA EN FUNCION DE LA ZONA
                 if(zonaCabeza) {
                  Formato.setFormatoCotas(g);
                        Formato.setColorRelleno(g);
                  g.setColor(Color.BLACK);
                  g.fillRect((int)x, alturaEsquema,(int)(dFDesc+dDTesc+parteExtra/
escala), 2);
                  g.setColor(Color.GRAY);
                  Formato.setColorLinea(g);
                  g.drawRect((int)x, alturaEsquema,(int)(dFDesc+dDTesc+parteExtra/
escala), 2);
                  //Dibujamos el KingPin
                  g.setColor(Color.RED);
                  g.fillRect((int)(x+dFDesc+dDKesc),alturaEsquema-2,2,8);
```

```
                    //Dibujamos los APOYOS de la viga
                          Formato.setColorRelleno(g);
                    g.fillOval((int)(x+dFDesc-2), alturaEsquema+2, 4, 4);
                    g.fillOval((int)(x+dFDesc+dDTesc-2), alturaEsquema+2, 4, 4);
                    Formato.setColorLinea(g);
                    g.drawOval((int)(x+dFDesc-2), alturaEsquema+2, 4, 4);
                    g.drawOval((int)(x+dFDesc+dDTesc-2), alturaEsquema+2, 4, 4);
                   } else {
                          //Dibujamos la viga del remolque
                    g.fillRect((int)(x+dFDesc+dDKesc-parteExtra/escala), alturaEsquema,
(int)(largoesc-dFDesc-dDKesc+parteExtra/escala), 2);
                    Formato.setColorLinea(g);
                    g.drawRect((int)(x+dFDesc+dDKesc-parteExtra/escala), alturaEsquema,
(int)(largoesc-dFDesc-dDKesc+parteExtra/escala), 2);
                    Formato.setColorLinea(g);
                    //Dibujamos el KingPin
                    g.setColor(Color.RED);
                    g.fillRect((int)(x+dFDesc+dDKesc),alturaEsquema-2,2,8);
                    //Dibujamos los APOYOS de la viga
                          Formato.setColorRelleno(g);
                    g.fillOval((int)(x+dFDesc+dDKesc+dKResc-2), alturaEsquema+2, 4, 4);
                    Formato.setColorLinea(g);
                    g.drawOval((int)(x+dFDesc+dDKesc+dKResc-2), alturaEsquema+2, 4, 4);
                   }

                   //Dibujamos LAS LEYES DE ESFUERZOS CORTANTES de la VIGA de la
estructura
              Formato.setFormatoCotas(g);
              Formato.setFuenteCargas(g);
              Formato.setFuenteEjes(g);
              //Obtenemos los puntos para dibujar
              ArrayList<PuntosCalculo>
listaPuntos=LeyesMomentos.puntosGrafico(proyecto);
              //Obtenemos la máxima carga para poder escalar convenientemente el gráfico
en eje Y
              double maxCarga;
              if(zonaCabeza) maxCarga=LeyesMomentos.getMaximaCargaCabeza();
              else maxCarga=LeyesMomentos.getMaximaCargaRemolque();
              //Ajustamos la escala al tamaño del gráfico
              double escalaY=70/Math.abs(maxCarga);
              //Dibujamos cada punto
              //Primero usamos un booleano para que nos ayude a mostrar el valor sólo
cuando cambie la tendencia (de incremento a decremento y viceversa)
              boolean ascendente=false;
              boolean descendente=false;
              boolean mantenido=false;
              boolean tendenciaascendente=false;
              boolean tendenciadescendente=false;
              boolean tendenciamantenida=false;
              Iterator<PuntosCalculo> itp=listaPuntos.iterator();
              PuntosCalculo p1=null;
              if(itp.hasNext()) p1=itp.next();
              while(itp.hasNext()) {
                    PuntosCalculo p2=itp.next();
                    if(p2.getY()>p1.getY()) {
                          if(ascendente) {
                                tendenciaascendente=true;
                                tendenciadescendente=false;
                                tendenciamantenida=false;
                          }
                          ascendente=true;
                          mantenido=false;
                          descendente=false;
                    }
                    if(p2.getY()<p1.getY()) {
                          if(descendente) {
                                tendenciadescendente=true;
```

```
                                    tendenciaascendente=false;
                                    tendenciamantenida=false;
                              }
                              ascendente=false;
                              mantenido=false;
                              descendente=true;
                        }
                        if(p2.getY()==p1.getY()) {
                              if(mantenido) {
                                    tendenciadescendente=false;
                                    tendenciaascendente=false;
                                    tendenciamantenida=true;
                              }
                              ascendente=false;
                              descendente=false;
                              mantenido=true;
                        }
                        if((tendenciadescendente&&(ascendente||mantenido))||
(tendenciaascendente&&(descendente||mantenido))||(tendenciamantenida&&(ascendente||
descendente))) {
                              Integer v=(int)Math.round(p2.getY());

v=(int)Unidades.getCargaEnUnidadesActuales(Unidades.getLongitudEnUnidadesActuales(v));
                              g.drawString(df.format(v),(int)(x+(p2.getX()/escala)),(int)
(alturaEsquema-(p2.getY()*escalaY)));
                        }
                        g.drawLine((int)(x+(p1.getX()/escala)), (int)(alturaEsquema-
(p1.getY()*escalaY)), (int)(x+(p2.getX()/escala)),(int)(alturaEsquema-
(p2.getY()*escalaY)));
                        p1=p2;
                  }

                  //Dibujamos los ejes
                  Formato.setFormatoCotas(g);
                  Formato.setColorLinea(g);
                  Formato.setFuenteEjes(g);
                  double gap=40;
                  double puntoEjeSup=gap;
                  double puntoEjeInf=2*alturaEsquema-gap;
                  g.drawLine((int)x,(int)puntoEjeInf,(int)x,(int)puntoEjeSup);
                  //Marcamos las divisiones por eje y con ese valor la longitud entre marcas
                  double marcasEje=5;
                  double longitudsup=(alturaEsquema-puntoEjeSup)/marcasEje;
                  double longitudinf=(puntoEjeInf-alturaEsquema)/marcasEje;
                  //Para que inicialmente ponga valores, si no hay cargas ponemos el valor
de 1000
                  if(maxCarga==0) maxCarga=1000;
                  double valor=(Math.round(maxCarga*100)/100)/marcasEje;
                  //Dibujamos 5 puntos en cada lado de corte
                  for(int i=1;i<=marcasEje;i++) {
                        g.drawLine((int)(x-2),(int)(alturaEsquema-longitudsup*i),(int)
(x+2),(int)(alturaEsquema-longitudsup*i));
                        g.drawLine((int)(x-2),(int)(alturaEsquema+longitudinf*i),(int)
(x+2),(int)(alturaEsquema+longitudinf*i));
                        g.drawString(df.format((int)
(Math.round(Unidades.getCargaEnUnidadesActuales(Unidades.getLongitudEnUnidadesActuales(va
lor*i))))),(int)(x+2),(int)(alturaEsquema-longitudsup*i));
                        g.drawString(df.format((int)
(Math.round(Unidades.getCargaEnUnidadesActuales(Unidades.getLongitudEnUnidadesActuales(-
valor*i))))),(int)(x+2),(int)(alturaEsquema+longitudinf*i));
                  }

               }

    }

    @Override
```

```
    public void paint(Graphics g) {
        if (imagen != null) {
            g.drawImage(imagen, 0, 0, getWidth(), getHeight(),this);
            setOpaque(false);
            dibuja(g);
        } else {
            setOpaque(true);
        }

        super.paint(g);
    }


}
```

CLASE PuntosCalculo

```
package calculosEstructura;

public class PuntosCalculo {
	private double x,y;
	public PuntosCalculo(double x, double y){
		setX(x);
		setY(y);
	}

	//Construimos los setters
	public void setX(double x) {
		this.x=x;
	}
	public void setY(double y) {
		this.y=y;
	}

	//Construimos los getters
	public double getX() {
		return x;
	}
	public double getY() {
		return y;
	}
}
```

CLASE VentanaCalculos

```
package calculosEstructura;

import java.awt.Color;
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.Point;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Iterator;
import camiones.*;
import cargasEstructura.Carga;
import cargasEstructura.CargaDistribuida;

import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.SpringLayout;
import javax.swing.WindowConstants;
import javax.swing.table.DefaultTableModel;
import javax.swing.table.JTableHeader;

import auxiliares.CheckDatos;
import auxiliares.DatosGenericos;
import auxiliares.Formato;
import auxiliares.PanelConImagen;
import auxiliares.Unidades;

public class VentanaCalculos extends JFrame{
	private static final long serialVersionUID = -7418029428404409927L;
	//Definimos los valores comunes para todos los componentes y elementos
	private static int ladopanel=200;
	private static String fondogris=DatosGenericos.getDireccion("Imagenes/fondos/
fondo-gris.png");
	private static String fondoazul=DatosGenericos.getDireccion("Imagenes/fondos/
fondo.jpg");
	private PanelConImagen panelMarca,general,cargas,reacciones,listacargas,tensiones;
//Generamos los paneles contenedores de los datos, el dibujo y la info
	private static PanelCortantes cortantes;
	private static PanelMomentos momentos;
	private PanelEsquema esquema;
	private JButton
cargaNueva,cargaElimina,cargaVisible,cargaParteCabeza,cargaParteRemolque; //Generamos los
botones para calcular
	private static Proyecto proyecto;
	private DefaultTableModel modelo,modeloReacciones,modeloTensiones;
	private JTable tablaCargas,tablaReacciones,tablaTensiones;
	private JScrollPane scrollPane,scrollPaneReacciones,scrollPaneTensiones;
	private static VentanaIntroduceCarga ventanaI;
	private static VentanaVerCarga ventanaV;
	private static VentanaEliminarCarga ventanaE;
	private static String
reaccionA,reaccionB,reaccionR,reaccionK,cortanteCabeza,momentoCabeza,cortanteRemolque,mom
entoRemolque,tensionCabeza,tensionRemolque;
	private static DecimalFormat df;
	private static VistaGeneral vg;
```

```
      public VentanaCalculos(Proyecto p,VistaGeneral vistaGeneral) {
            super("Calcular estructura del camión");
            calcularReaccionesyTensionesActuales();

            //Generamos un formateo para los strings de las tablas
            df = new DecimalFormat("0.00");

            //Guardamos las variables
            proyecto=p;
            vg=vistaGeneral;

            //Incluimos el panel de marca para el logo.
            panelMarca=new PanelConImagen(DatosGenericos.getDireccion("Imagenes/
pantallaprincipal/recurso.png")); //contiene el logo de la aplicación
            panelMarca.setPreferredSize(new Dimension(150,80));
            //Contenedor general
            general = new PanelConImagen(fondoazul);
            //dibujo contiene el panel donde se dibuja la seccion
            esquema=new PanelEsquema(fondogris,proyecto);
            //Asignamos el proyecto actual al esquema
            esquema.setProyecto(proyecto);
            //Resultado cortantes
            cortantes = new PanelCortantes(fondogris,proyecto);
            //Resultado momentos
            momentos = new PanelMomentos(fondogris,proyecto);
            //Cargas a introducir
            cargas = new PanelConImagen(fondogris);
            //Tensiones
            tensiones = new PanelConImagen(fondogris);
            //Reacciones en ejes
            reacciones = new PanelConImagen(fondogris);
            //Panel para la lista de cargas
            listacargas = new PanelConImagen(fondogris);


            //Generamos el titulo de la ventana
            JLabel tituloapp = new JLabel("EDICIÓN DE CARGAS");
        //Damos formato a los tipos de aplicacion
        Formato.setFuenteAplicacion(tituloapp);


            //Generamos el JTABLE DE LAS CARGAS
            //Creamos el modelo de la tabla
          modelo = new DefaultTableModel();
          //Le damos los nombres de las columnas
          modelo.addColumn("Número");
          modelo.addColumn("Nombre");
          modelo.addColumn("Punto Inicio");
          modelo.addColumn("Valor");
          modelo.addColumn("Tipo");
          modelo.addColumn("Longitud(distribuida)");
          //se crea la Tabla
          tablaCargas = new JTable();
          //Le asignamos el modelo
          tablaCargas.setModel(modelo);
          //Le damos las dimensiones a la tabla
        tablaCargas.setPreferredScrollableViewportSize(new Dimension((4*ladopanel-20),
(ladopanel-60)));
        //Creamos un JscrollPane y le agregamos la JTable creada
        scrollPane = new JScrollPane(tablaCargas);
        //Metemos en el panel de cargas el scrollpane que contiene la tabla-
        listacargas.add(scrollPane);
        //No permitimos la edicion directa en la tabla
        tablaCargas.setEnabled(false);
        Formato.setFormatoContenidoTabla(tablaCargas);
        rellenaTablaCargas();
```

```
            //Generamos el JTABLE DE LAS REACCIONES
            //Creamos el modelo de la tabla
          modeloReacciones = new DefaultTableModel();
          //Le damos los nombres de las columnas
          modeloReacciones.addColumn("Eje");
          modeloReacciones.addColumn("Carga tándem");
          modeloReacciones.addColumn("Carga eje");
          //se crea la Tabla
          tablaReacciones = new JTable();
          //Le asignamos el modelo
          tablaReacciones.setModel(modeloReacciones);
          //Le damos las dimensiones a la tabla
        tablaReacciones.setPreferredScrollableViewportSize(new
Dimension((2*ladopanel-20),(ladopanel-60)));
        //Creamos un JscrollPane y le agregamos la JTable creada
        scrollPaneReacciones = new JScrollPane(tablaReacciones);
        //Metemos en el panel de cargas el scrollpane que contiene la tabla-
        reacciones.add(scrollPaneReacciones);
        tablaReacciones.setEnabled(false);
        //PERSONALIZAMOS EL FORMATO DE LA TABLA
        Formato.setFormatoContenidoTabla(tablaReacciones);
        Object[] fila1={"Delantero",0,0};
        Object[] fila2={"Trasero",0,0};
        Object[] fila3={"Remolque",0,0};
        Object[] fila4={"KingPin","N/A",0};
        modeloReacciones.addRow(fila1);
        modeloReacciones.addRow(fila2);
        modeloReacciones.addRow(fila3);
        modeloReacciones.addRow(fila4);
        rellenaTablaReacciones();


            //Generamos el JTABLE DE LAS TENSIONES
            //Creamos el modelo de la tabla
          modeloTensiones = new DefaultTableModel();
          //Le damos los nombres de las columnas
          modeloTensiones.addColumn("Zona");
          modeloTensiones.addColumn("Tensión Normal");
          modeloTensiones.addColumn("Tensión Cortante");
          modeloTensiones.addColumn("Tensión Von Mises");
          //se crea la Tabla
          tablaTensiones = new JTable();
          //Le asignamos el modelo
          tablaTensiones.setModel(modeloTensiones);
          //Le damos las dimensiones a la tabla
        tablaTensiones.setPreferredScrollableViewportSize(new Dimension((4*ladopanel-20),
(ladopanel-60)));
        //Creamos un JscrollPane y le agregamos la JTable creada
        scrollPaneTensiones = new JScrollPane(tablaTensiones);
        //Metemos en el panel de cargas el scrollpane que contiene la tabla-
        tensiones.add(scrollPaneTensiones);
        tablaTensiones.setEnabled(false);
        //PERSONALIZAMOS EL FORMATO DE LA TABLA
        Formato.setFormatoContenidoTabla(tablaTensiones);
        Object[] filatens1={"Cabeza Tractora",0,0,0};
        Object[] filatens2={"Remolque",0,0,0};
        modeloTensiones.addRow(filatens1);
        modeloTensiones.addRow(filatens2);
        try {
        rellenaTablaTensiones();
        } catch (Exception e) {
            CheckDatos.okCancelaMensaje("No se han podido rellenar las tensiones");
        }
```

```
          //Generamos los botones de cargas y la etiqueta
             cargaNueva = new JButton("Nueva");
             cargaElimina = new JButton("Elimina");
             cargaVisible = new JButton("Ver");
             cargaParteCabeza= new JButton("Analizar\nCabeza");
             cargaParteRemolque= new JButton("Analizar\nRemolque");
             cargaParteCabeza.setPreferredSize(new Dimension(110,40));
             cargaParteRemolque.setPreferredSize(new Dimension(110,40));
             ImageIcon iconobtnCargaNueva = escalaIcono(new
ImageIcon(getClass().getResource(DatosGenericos.getDireccion("Imagenes/pantallaprincipal/
carga.png"))),2);
             ImageIcon iconobtnCargaVisible = escalaIcono(new
ImageIcon(getClass().getResource(DatosGenericos.getDireccion("Imagenes/pantallaprincipal/
ver.png"))),2);
             ImageIcon iconobtnCargaElimina = escalaIcono(new
ImageIcon(getClass().getResource(DatosGenericos.getDireccion("Imagenes/pantallaprincipal/
borra.png"))),2);
             //Damos Formato
             Formato.setFormatoMenor(cargaNueva);
             Formato.setFormatoMenor(cargaVisible);
             Formato.setFormatoMenor(cargaElimina);
             Formato.setFormatoMenor(cargaParteCabeza);
             Formato.setFormatoMenor(cargaParteRemolque);
             asignaIconoBoton(cargaNueva,iconobtnCargaNueva);
             asignaIconoBoton(cargaVisible,iconobtnCargaVisible);
             asignaIconoBoton(cargaElimina,iconobtnCargaElimina);


             //Inicializamos las reacciones
             if(proyecto.getCamion().getTipo().getTipo()=='R') setReacciones(0,0); //
Asignamos solo el valor del eje delantero y trasero
             else setReacciones(0,0,0,0); //Asignamos el valor del eje delantero, el
trasero y el del remolque


             //Inicializamos las tensiones
             if(proyecto.getCamion().getTipo().getTipo()=='R')
setTensionesCabeza(0,0,0); //Asignamos solo el valor de la cabeza
             else {
                    setTensionesCabeza(0,0,0); //Asignamos el valor de tension de la
cabeza
                    setTensionesRemolque(0,0,0); //Asignamos el valor de tension del
remolque
             }


             //Generamos los valores de los Gap para los SpringLayouts.
             int gapX=10;
             int gapY=10;
             int gapXVentana =
(Toolkit.getDefaultToolkit().getScreenSize().width-7*ladopanel-2*gapX)/2; //Este gap
centra todo segun la pantalla


             //Generamos los LAYOUTS A PANELES
             SpringLayout layoutgeneral = new SpringLayout();
             //Asignamos los constraints
             layoutgeneral.putConstraint(SpringLayout.WEST, panelMarca, gapXVentana,
SpringLayout.WEST, general);
             layoutgeneral.putConstraint(SpringLayout.WEST, tituloapp,
gapXVentana+54*gapX, SpringLayout.WEST, general);
             layoutgeneral.putConstraint(SpringLayout.WEST, esquema, gapXVentana,
SpringLayout.WEST, general);
             layoutgeneral.putConstraint(SpringLayout.WEST, cortantes, gapXVentana,
SpringLayout.WEST, general);
             layoutgeneral.putConstraint(SpringLayout.WEST, momentos, gapXVentana,
SpringLayout.WEST, general);
```

```
            layoutgeneral.putConstraint(SpringLayout.WEST, listacargas, gapX,
SpringLayout.EAST, esquema);
      layoutgeneral.putConstraint(SpringLayout.WEST, cargas, gapX, SpringLayout.EAST,
esquema);
      layoutgeneral.putConstraint(SpringLayout.WEST, reacciones, gapX,
SpringLayout.EAST, cargas);
      layoutgeneral.putConstraint(SpringLayout.WEST, tensiones, gapX,
SpringLayout.EAST,esquema);
      //Incluimos los elementos en los paneles las restricciones en Y
            layoutgeneral.putConstraint(SpringLayout.NORTH, panelMarca, gapY,
SpringLayout.NORTH, general);
            layoutgeneral.putConstraint(SpringLayout.NORTH, tituloapp, 3*gapX,
SpringLayout.NORTH, general);
      layoutgeneral.putConstraint(SpringLayout.NORTH, esquema, gapY, SpringLayout.SOUTH,
panelMarca);
      layoutgeneral.putConstraint(SpringLayout.NORTH, cortantes, gapY,
SpringLayout.SOUTH, esquema);
      layoutgeneral.putConstraint(SpringLayout.NORTH, momentos, gapY,
SpringLayout.SOUTH, cortantes);
      layoutgeneral.putConstraint(SpringLayout.NORTH, cargas, gapY, SpringLayout.SOUTH,
panelMarca);
      layoutgeneral.putConstraint(SpringLayout.NORTH, reacciones, gapY,
SpringLayout.SOUTH, panelMarca );
      layoutgeneral.putConstraint(SpringLayout.NORTH, listacargas, gapY,
SpringLayout.SOUTH,cargas);
      layoutgeneral.putConstraint(SpringLayout.NORTH, tensiones, gapY,
SpringLayout.SOUTH,listacargas);


      //ponemos TITULOS Y MARCOS A PANELES
      panelMarca.setVisible(true);
      //Este string aclara qué zona se está calculando
      String zonaCalculo="";
      if(proyecto.getCamion().getTipo().getTipo()=='R') zonaCalculo=" Camión Rígido";
      else if(LeyesCortantes.getParteCabeza()) zonaCalculo=" Cabeza Tractora";
      else zonaCalculo=" Zona Remolque";
      Formato.setBorder(esquema, "Esquema de Estructura");
       esquema.setLayout(new GridLayout(0,1,0,0));
            esquema.setPreferredSize(new Dimension((3*ladopanel),ladopanel));
            esquema.setVisible(true);
      Formato.setBorder(cortantes, "Diagrama de Cargas Cortantes"+zonaCalculo);
            cortantes.setPreferredSize(new Dimension((3*ladopanel),ladopanel));
            cortantes.setVisible(true);
      Formato.setBorder(momentos, "Diagrama de Momentos Flectores"+zonaCalculo);
            momentos.setPreferredSize(new Dimension((3*ladopanel),ladopanel));
            momentos.setVisible(true);
      Formato.setBorder(listacargas, "Listado Cargas");
            listacargas.setPreferredSize(new Dimension((4*ladopanel+gapX),
(ladopanel)));
            listacargas.setVisible(true);
      Formato.setBorder(tensiones, "Tensiones Estructura");
            tensiones.setPreferredSize(new Dimension((4*ladopanel+gapX),(ladopanel)));
            tensiones.setVisible(true);
      Formato.setBorder(cargas, "Edición y Visualización de Cargas y Estructura");
            cargas.setPreferredSize(new Dimension(2*ladopanel,ladopanel));
            cargas.setVisible(true);
      Formato.setBorder(reacciones, "Reacciones y Validez Ejes");
            reacciones.setPreferredSize(new Dimension(2*ladopanel,ladopanel));
            reacciones.setVisible(true);
            general.setLayout(layoutgeneral);
            general.add(panelMarca);
            general.add(tituloapp);
            general.add(esquema);
            general.add(cortantes);
            general.add(momentos);
            general.add(listacargas);
            general.add(cargas);
```

```
            general.add(reacciones);
            general.add(tensiones);


            //ActionListener boton crear carga
       ActionListener accionCreaCarga=new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                  introduceCargas();
                  //Introducimos los valores en la tabla
                  rellenaTablaCargas();
                  }
      };

      //ActionListener boton ver carga
       ActionListener accionVerCarga=new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                  verCargas();
                  //Introducimos los valores en la tabla
                  rellenaTablaCargas();
                  }
      };

      //ActionListener boton eliminar carga
       ActionListener accionEliminarCarga=new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                  eliminarCargas();
                  //Introducimos los valores en la tabla
                  rellenaTablaCargas();
                  }
      };

      //ActionListener usar parte de camion
       ActionListener accionUsarParteCabeza=new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                  usarParte(true);
                  //Introducimos los valores en la tabla
                  rellenaTablaCargas();
                  }
      };

      //ActionListener usar parte de camion
       ActionListener accionUsarParteRemolque=new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                  usarParte(false);
                  //Introducimos los valores en la tabla
                  rellenaTablaCargas();
                  }
      };

      //ASIGNAMOS ACTIONLISTENERS A ELEMENTOS
       cargaNueva.addActionListener(accionCreaCarga);
       cargaVisible.addActionListener(accionVerCarga);
       cargaElimina.addActionListener(accionEliminarCarga);
       cargaParteCabeza.addActionListener(accionUsarParteCabeza);
       cargaParteRemolque.addActionListener(accionUsarParteRemolque);


            //Mememos los botones en el panel de CARGAS
            SpringLayout layoutcargas = new SpringLayout();
            //Asignamos los constraints
            layoutcargas.putConstraint(SpringLayout.WEST, cargaNueva, gapX-5,
SpringLayout.WEST, cargas);
            layoutcargas.putConstraint(SpringLayout.WEST, cargaVisible, gapX-5,
SpringLayout.EAST, cargaNueva);
            layoutcargas.putConstraint(SpringLayout.WEST,
cargaElimina,gapX-5,SpringLayout.EAST,cargaVisible);
```

```
            layoutcargas.putConstraint(SpringLayout.WEST,
cargaParteCabeza,gapX-1,SpringLayout.EAST,cargaElimina);
            layoutcargas.putConstraint(SpringLayout.WEST,
cargaParteRemolque,gapX-1,SpringLayout.EAST,cargaElimina);
            layoutcargas.putConstraint(SpringLayout.NORTH, cargaNueva, 2*gapY,
SpringLayout.NORTH, cargas);
            layoutcargas.putConstraint(SpringLayout.NORTH, cargaVisible, 2*gapY,
SpringLayout.NORTH, cargas);
            layoutcargas.putConstraint(SpringLayout.NORTH, cargaElimina, 2*gapY,
SpringLayout.NORTH, cargas);
            layoutcargas.putConstraint(SpringLayout.NORTH, cargaParteCabeza, 4*gapY,
SpringLayout.NORTH, cargas);
            layoutcargas.putConstraint(SpringLayout.NORTH, cargaParteRemolque, gapY-10,
SpringLayout.SOUTH, cargaParteCabeza);
            cargas.setLayout(layoutcargas);
            cargas.add(cargaNueva);
            cargas.add(cargaVisible);
            cargas.add(cargaElimina);
            cargas.add(cargaParteCabeza);
            cargas.add(cargaParteRemolque);
            //Si es articulado validamos los botones de ParteCabeza y Remolque, si es
rigido los inutilizamos
            visibilidadCargaCabezaRemolque();
            //Metemos el Panel General en la ventana y la preparamos
            add(general);
            setResizable(true);
            setExtendedState(JFrame.MAXIMIZED_BOTH);
            pack();
            setVisible(true);
      }

    //Este método escala los iconos de los botones con un coeficiente de escala para
evitar problemas por tamaños
    private ImageIcon escalaIcono(ImageIcon i, int s) {
       return new ImageIcon(i.getImage().getScaledInstance(40*s, -1,
java.awt.Image.SCALE_DEFAULT));
    }

    private void actualizaEsquema() {
       esquema.repaint();
    }

    //Este método asigna los iconos a los botones disponiendo el texto y la imagen en la
posicion deseada
    private void asignaIconoBoton(JButton c,ImageIcon i) {
       c.setFocusPainted(false);
       c.setBorderPainted(false);
       c.setContentAreaFilled(false);
       c.setPreferredSize(new Dimension(80,140));
       c.setIcon(i);
        c.setIconTextGap(5);
        c.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        c.setVerticalAlignment(javax.swing.SwingConstants.CENTER);
        c.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER);
        c.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
    }

    public void borraTabla() {
        DefaultTableModel modelotabla = (DefaultTableModel) tablaCargas.getModel();
        modelotabla=new DefaultTableModel();
    }

    //Este método traerá las cargas del proyecto para mostrarlas en la tabla
    public void rellenaTablaCargas() {
       //Limpiamos la tabla para no duplicar filas
        DefaultTableModel modelotabla = (DefaultTableModel) tablaCargas.getModel();
        int filas = tablaCargas.getRowCount()-1;
```

```
        //limpiamos de la fila 0 a la que ultima
        for (int i = 0; i <=filas; i++) {
        modelotabla.removeRow(modelotabla.getRowCount()-1);
        }
       ArrayList<Carga> cargas=proyecto.getCargas();
              Iterator<Carga> it = cargas.iterator();
              int contador=1;
              Object[] datosnuevo= {0,"","","","",""};
              while(it.hasNext()){
                     Carga cargaexistente = it.next();
                     //Los valores son: 0) Contador, 1) Nombre, 2) Punto Inicio, 3) Valor,
4) Tipo, 5) Longitud Distribuida
                     datosnuevo[0]=contador;
                     datosnuevo[1]=cargaexistente.getNombre();
                     datosnuevo[4]=cargaexistente.getTipo().getDescripcion();

datosnuevo[2]=df.format(Unidades.getLongitudEnUnidadesActuales(cargaexistente.getPuntoIni
cio()))+Unidades.getUnidadesLongitud()+".";
                     if(cargaexistente.getTipo().getTipoCarga()=='D') {
                            CargaDistribuida cd=(CargaDistribuida)cargaexistente;

datosnuevo[5]=df.format(Unidades.getLongitudEnUnidadesActuales(cd.getLongitud()))
+Unidades.getUnidadesLongitud()+".";

datosnuevo[3]=df.format(Unidades.getCargaEnUnidadesActuales(cargaexistente.getValor())/
Unidades.getConversionLongitud())+Unidades.getUnidadesCarga()
+"/"+Unidades.getUnidadesLongitud()+".";
                     }
                     else {
                            datosnuevo[5]="N/A";

datosnuevo[3]=df.format(Unidades.getCargaEnUnidadesActuales(cargaexistente.getValor()))
+Unidades.getUnidadesCarga()+".";
                     }

                     modelo.addRow(datosnuevo);
                     contador++;
              };
    }

    //Este método traerá las cargas del proyecto para mostrarlas en la tabla
    public void rellenaTablaReacciones() {
       if(modeloReacciones!=null) {
              double delantero;
              double trasero;
              double remolque;
              //Para poder rellenar la tabla, primero vamos a ver los ejes por tandem
para luego dividir.
              if(proyecto.getCamion().getTipo().getTipo()=='A') {

delantero=Unidades.getCargaEnUnidadesActuales(CalculoReaccionesArticulado.getReaccionA())
/((CamionArticulado)proyecto.getCamion()).getEjesPorTandemDelantero();

trasero=Unidades.getCargaEnUnidadesActuales(CalculoReaccionesArticulado.getReaccionB())/
((CamionArticulado)proyecto.getCamion()).getEjesPorTandemTrasero();

remolque=Unidades.getCargaEnUnidadesActuales(CalculoReaccionesArticulado.getReaccionRemol
que())/((CamionArticulado)proyecto.getCamion()).getEjesPorTandemRemolque();
              } else {

delantero=Unidades.getCargaEnUnidadesActuales(CalculoReaccionesRigido.getReaccionA())/
proyecto.getCamion().getEjesPorTandemDelantero();

trasero=Unidades.getCargaEnUnidadesActuales(CalculoReaccionesRigido.getReaccionB())/
proyecto.getCamion().getEjesPorTandemTrasero();
                     remolque=0;
              }
```

```
              modeloReacciones.setValueAt("Delantero Camión", 0, 0);
              modeloReacciones.setValueAt("Trasero Camión", 1, 0);
              modeloReacciones.setValueAt("Remolque", 2, 0);
              modeloReacciones.setValueAt("KingPin", 3, 0);
              modeloReacciones.setValueAt(reaccionA, 0, 1);
              modeloReacciones.setValueAt(reaccionB, 1, 1);
              modeloReacciones.setValueAt(reaccionR, 2, 1);
              modeloReacciones.setValueAt("N/A",3,1);
              //Aqui es la columna 2 porque el KingPin nunca tiene tandem.
              modeloReacciones.setValueAt(df.format(delantero)
+Unidades.getUnidadesCarga(), 0, 2);
              modeloReacciones.setValueAt(df.format(trasero)+Unidades.getUnidadesCarga(),
1, 2);
              modeloReacciones.setValueAt(df.format(remolque)
+Unidades.getUnidadesCarga(), 2, 2);
              modeloReacciones.setValueAt(reaccionK, 3, 2);
              if(proyecto.getCamion().getTipo().getTipo()=='R') {
                     modeloReacciones.setValueAt("N/A", 2, 2);
                     modeloReacciones.setValueAt("N/A", 3, 2);
              }
       }


    }

    //Este método traerá las Tensiones del proyecto para mostrarlas en la tabla
    public void rellenaTablaTensiones() {
       if(modeloTensiones!=null) {
              modeloTensiones.setValueAt("Cabeza Tractora/Camión", 0, 0);
              modeloTensiones.setValueAt("Remolque", 1, 0);
              modeloTensiones.setValueAt(momentoCabeza, 0, 1);
              modeloTensiones.setValueAt(momentoRemolque, 1, 1);
              modeloTensiones.setValueAt(cortanteCabeza, 0, 2);
              modeloTensiones.setValueAt(cortanteRemolque, 1, 2);
              modeloTensiones.setValueAt(tensionCabeza, 0, 3);
              modeloTensiones.setValueAt(tensionRemolque, 1, 3);
       }

    }

    //Introducimos las cargas. Pasamos a la clase VentanaIntroduceCargas la instancia de
Ventana de Calculo para que
    //Actualice la información de la tabla a medida que introducimos los valores.
    public void introduceCargas() {
       if(ventanaI==null) ventanaI=new
VentanaIntroduceCarga(proyecto,esquema,cortantes,momentos,this,tensiones);
       else ventanaI.setVisible(true);
       //Ocultamos la ventana Ver cargas si existe
       if (ventanaV!=null) ventanaV.setVisible(false);
       //Ocultamos la ventana Eliminar cargas si existe
       if (ventanaE!=null) ventanaE.setVisible(false);
    }
    private void verCargas() {
       if(ventanaV==null) ventanaV=new VentanaVerCarga(proyecto,esquema,this);
       else {
              ventanaV.actualizaCombo();
              ventanaV.setVisible(true);
       }
       //Ocultamos la ventana Introducir cargas si existe
       if (ventanaI!=null) ventanaI.setVisible(false);
       //Ocultamos la ventana Eliminar cargas si existe
       if (ventanaE!=null) ventanaE.setVisible(false);
    }
    private void eliminarCargas() {
       if(ventanaE==null) ventanaE=new
VentanaEliminarCarga(proyecto,esquema,cortantes,momentos,this,tensiones);
       else {
```

```
                ventanaE.actualizaCombo();
                ventanaE.setVisible(true);
        }
        //Ocultamos la ventana Introducir cargas si existe
        if (ventanaI!=null) ventanaI.setVisible(false);
        //Ocultamos la ventana Ver cargas si existe
        if (ventanaV!=null) ventanaV.setVisible(false);
    }
        //Este método actualiza las ventanas de Ver y Eliminar cargas si existen para
reflejar los cambios de cargas introducidas
    public void actualizaVentanas() {
        //Ventana Eliminar Cargas
        if(ventanaE!=null) {
                        ventanaE = new
VentanaEliminarCarga(proyecto,esquema,cortantes,momentos,this,tensiones);
                        ventanaE.setVisible(false);
        }
        //Ventana Ver Cargas
        if(ventanaV!=null) {
                ventanaV=new VentanaVerCarga(proyecto,esquema,this);
                ventanaV.setVisible(false);
        }
    }

    //Este metodo esta sobrecargado, uno para el camion rigido y otro para el articulado
    //Asigna a las etiquetas de la ventana Reacciones el valor actual para un camion
Rigido.
    public void setReacciones(double rA, double rB) {
        if(rA>proyecto.getCamion().getCargaMaxEjeDelantero()) {
                reaccionA=df.format(Unidades.getCargaEnUnidadesActuales(rA))
+Unidades.getUnidadesCarga()+" (excedida)";
        }
        else {
                reaccionA=df.format(Unidades.getCargaEnUnidadesActuales(rA))
+Unidades.getUnidadesCarga();
        }
        if(rB>proyecto.getCamion().getCargaMaxEjeTrasero()) {
                reaccionB=df.format(Unidades.getCargaEnUnidadesActuales(rB))
+Unidades.getUnidadesCarga()+" (excedida)";
        } else {
                reaccionB=df.format(Unidades.getCargaEnUnidadesActuales(rB))
+Unidades.getUnidadesCarga();
        }
        reaccionK="N/A";
        reaccionR="N/A";
        rellenaTablaReacciones();
    }

    //Sobrecargamos el método para usarlo con el articulado
    //Asigna a las etiquetas de la ventana Reacciones el valor actual para un camion
articulado.
    public void setReacciones(double rA, double rB, double rR, double rK) {
        if(rA>proyecto.getCamion().getCargaMaxEjeDelantero()) {
                reaccionA=df.format(Unidades.getCargaEnUnidadesActuales(rA))
+Unidades.getUnidadesCarga()+" (excedida)";
        }
        else {
                reaccionA=df.format(Unidades.getCargaEnUnidadesActuales(rA))
+Unidades.getUnidadesCarga();
        }
        if(rB>proyecto.getCamion().getCargaMaxEjeTrasero()) {
                reaccionB=df.format(Unidades.getCargaEnUnidadesActuales(rB))
+Unidades.getUnidadesCarga()+" (excedida)";
        } else {
                reaccionB=df.format(Unidades.getCargaEnUnidadesActuales(rB))
+Unidades.getUnidadesCarga();
        }
```

```
       CamionArticulado ca=(CamionArticulado)proyecto.getCamion();
       if(rR>ca.getCargaMaximaEjeRemolque()) {
              reaccionR=df.format(Unidades.getCargaEnUnidadesActuales(rR))
+Unidades.getUnidadesCarga()+" (excedida)";
       }
       else {
              reaccionR=df.format(Unidades.getCargaEnUnidadesActuales(rR))
+Unidades.getUnidadesCarga();
       }
       reaccionK=df.format(Unidades.getCargaEnUnidadesActuales(rK))
+Unidades.getUnidadesCarga();
       rellenaTablaReacciones();
    }

  //Este metodo esta sobrecargado, uno para el camion rigido y otro para el articulado
    //Asigna a las etiquetas de la ventana Reacciones el valor actual para un camion
Rigido.
    public void setTensionesCabeza(double cortCabeza, double momCabeza, double
tensCabeza) {
       boolean datosOK=CheckDatos.checkViga(proyecto);
       if(datosOK) {

cortanteCabeza=df.format(Unidades.getTensionEnUnidadesActuales(cortCabeza))
+Unidades.getUnidadesTension();
              momentoCabeza=df.format(Unidades.getTensionEnUnidadesActuales(momCabeza))
+Unidades.getUnidadesTension();

tensionCabeza=df.format(Unidades.getTensionEnUnidadesActuales(tensCabeza))
+Unidades.getUnidadesTension();
                     cortanteRemolque="N/A";
              momentoRemolque="N/A";
                     tensionRemolque="N/A";
       }
       else {
              cortanteCabeza="Sin Propiedades de Estructura";
              momentoCabeza="Sin Propiedades de Estructura";
                     tensionCabeza="Sin Propiedades de Estructura";
              cortanteRemolque="N/A";
              momentoRemolque="N/A";
              tensionRemolque="N/A";
       }
       rellenaTablaTensiones();
}

    //Sobrecargamos el método para usarlo con el articulado
    //Asigna a las etiquetas de la ventana Reacciones el valor actual para un camion
articulado.
    public void setTensionesRemolque(double corRemolque, double momRemolque, double
tensRemolque) {
       boolean datosOK=CheckDatos.checkViga(proyecto);
       if(datosOK) {

cortanteRemolque=df.format(Unidades.getTensionEnUnidadesActuales(corRemolque))
+Unidades.getUnidadesTension();

momentoRemolque=df.format(Unidades.getTensionEnUnidadesActuales(momRemolque))
+Unidades.getUnidadesTension();

tensionRemolque=df.format(Unidades.getTensionEnUnidadesActuales(tensRemolque))
+Unidades.getUnidadesTension();
       } else {
              cortanteRemolque="Sin Propiedades de Estructura";
              momentoRemolque="Sin Propiedades de Estructura";
              tensionRemolque="Sin Propiedades de Estructura";
       }
       rellenaTablaTensiones();
    }
```

```
    //Este metodo actualiza la informacion de los esquemas.
    public void usarParte(boolean b) {
       String zonaCalculo;
       LeyesCortantes.setParteCabeza(b);
       LeyesMomentos.setParteCabeza(b);
       if(proyecto.getCamion().getTipo().getTipo()=='R') zonaCalculo=" Camion Rígido";
       else if(LeyesCortantes.getParteCabeza()) zonaCalculo=" Cabeza Tractora";
       else zonaCalculo=" Zona Remolque";
       Formato.setBorder(cortantes, "Diagrama de Cargas Cortantes"+zonaCalculo);
       Formato.setBorder(momentos, "Diagrama de Momentos Flectores"+zonaCalculo);
       esquema.repaint();
       cortantes.repaint();
       momentos.repaint();
    }

    //Este metodo sirve para recalcular los valores de las reacciones actuales para
poderlas pintar en la ventana ya que solo
    //se calculan al introducir cargas o eliminarlas pero no al cargar la ventana
    public void calcularReaccionesyTensionesActuales() {
       if(proyecto!=null) {
              if(proyecto.getCamion().getTipo().getTipo()=='R') {

setReacciones(CalculoReaccionesRigido.getReaccionA(),CalculoReaccionesRigido.getReaccionB
());

setTensionesCabeza(proyecto.getCamion().getTensionCortanteViga(),proyecto.getCamion().get
TensionNormalViga(),proyecto.getCamion().getTensionVonMisses());
              } else {

setReacciones(CalculoReaccionesArticulado.getReaccionA(),CalculoReaccionesArticulado.getR
eaccionB(),

CalculoReaccionesArticulado.getReaccionRemolque(),CalculoReaccionesArticulado.getReaccion
KingPin());

setTensionesCabeza(((CamionArticulado)proyecto.getCamion()).getTensionCortanteViga(),

((CamionArticulado)proyecto.getCamion()).getTensionNormalViga(),

((CamionArticulado)proyecto.getCamion()).getTensionVonMisses());

setTensionesRemolque(((CamionArticulado)proyecto.getCamion()).getTensionCortanteVigaRemol
que(),

((CamionArticulado)proyecto.getCamion()).getTensionNormalVigaRemolque(),

((CamionArticulado)proyecto.getCamion()).getTensionVonMissesVigaRemolque());
              }
              }
    }

    public void repintaTabla() {
       this.tablaCargas.repaint();
    }

    public void repaint() {
       Camion c=proyecto.getCamion();
       if(c.getTipo().getTipo()=='R') {
              CalculoReaccionesRigido.sumareacciones(proyecto);
              LeyesCortantes.puntosGrafico(proyecto);
              LeyesMomentos.puntosGrafico(proyecto);
       } else {
              CalculoReaccionesArticulado.sumareaccionesCabeza(proyecto);
                     CalculoReaccionesArticulado.sumareaccionesRemolque(proyecto);
                     LeyesCortantes.preparaParaTension();
                     LeyesMomentos.preparaParaTension();
```

```
      }

      calcularReaccionesyTensionesActuales();
      rellenaTablaReacciones();
      rellenaTablaCargas();
      reacciones.repaint();
      esquema.repaint();
      cortantes.repaint();
      momentos.repaint();
      visibilidadCargaCabezaRemolque();
   }

   public void visibilidadCargaCabezaRemolque() {
            if(proyecto.getCamion().getTipo().getTipo()=='A') {
                  cargaParteCabeza.setEnabled(true);
                  cargaParteRemolque.setEnabled(true);
            } else {
                  cargaParteCabeza.setEnabled(false);
                  cargaParteRemolque.setEnabled(false);
            }
   }

   public PanelEsquema getEsquema() {
      return esquema;
   }

   public PanelCortantes getCortantes() {
      return cortantes;
   }

   public PanelMomentos getMomentos() {
      return momentos;
   }

   public PanelConImagen getListaCargas() {
      return listacargas;
   }

   public PanelConImagen getReacciones() {
      return reacciones;
   }

   public void reseteaVentanas() {
      ventanaI=new
VentanaIntroduceCarga(proyecto,esquema,cortantes,momentos,this,tensiones);
      ventanaI.setVisible(false);

      ventanaE=new
VentanaEliminarCarga(proyecto,esquema,cortantes,momentos,this,tensiones);
      ventanaE.setVisible(false);

      ventanaV=new VentanaVerCarga(proyecto,esquema,this);
      ventanaV.setVisible(false);
   }
}
```

CLASE VentanaEliminarCarga

```
/*Esta clase sirve para introducir cargas en el proyecto*/

package calculosEstructura;

import java.awt.Color;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.Iterator;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.SpringLayout;

import auxiliares.CheckDatos;
import auxiliares.DatosGenericos;
import auxiliares.Formato;
import auxiliares.PanelConImagen;
import camiones.*;
import cargasEstructura.Carga;

public class VentanaEliminarCarga extends JFrame{
	private static final long serialVersionUID = 1L;
	private static int ladopanel=340;
	private static String fondogris=DatosGenericos.getDireccion("Imagenes/fondos/
fondo-gris.png");
	private static String fondoazul=DatosGenericos.getDireccion("Imagenes/fondos/
fondo.jpg");
	private static ArrayList<Component> c = new ArrayList<Component>();
	private static ArrayList<JTextField> t = new ArrayList<JTextField>();
	private PanelConImagen general,panelMarca,datosPanel,tensiones; //Generamos los
paneles contenedores de los datos, el dibujo y la info
	private JLabel lblCarga,lblTitulo; //Generamos las etiquetas
	private JButton btnEliminar; //Generamos el boton para eliminar
	private JComboBox<String> jcbCarga;
	private static Proyecto proyecto;
	private static PanelEsquema esquema;
	private static PanelCortantes cortantes;
	private static PanelMomentos momentos;
	private VentanaCalculos ventanaCalculo;

	public VentanaEliminarCarga(Proyecto p,PanelEsquema pe,PanelCortantes
pc,PanelMomentos pm,VentanaCalculos v,PanelConImagen t){
		super("Eliminar carga");
		this.setResizable(false);
		proyecto=p;
		esquema=pe;
		cortantes=pc;
		momentos=pm;
		ventanaCalculo=v;
		tensiones=t;


		//Preparamos los PANELES
		//datos panel contiene los elementos de entrada d2e datos
		datosPanel=new PanelConImagen(fondogris);
		//general contiene los datos de interaccion
```

```
            general=new PanelConImagen(fondoazul);
            panelMarca=new PanelConImagen(DatosGenericos.getDireccion("Imagenes/
pantallaprincipal/recurso.png")); //contiene el logo de la aplicación
            panelMarca.setPreferredSize(new Dimension(150,80));


            //Damos formato al PANEL GENERAL
            //Damos el layout para general
            SpringLayout layout = new SpringLayout();
            //Damos el layout para datosPanel
            SpringLayout layoutDatosPanel = new SpringLayout();
            //Definimos el gap entre ventanas
        int gapX=10;
        int gapY=10;
            datosPanel.setPreferredSize(new Dimension(ladopanel-4*gapX,ladopanel-130));
            Formato.setBorder(datosPanel);
        datosPanel.setVisible(true);
            general.setPreferredSize(new Dimension(ladopanel,ladopanel+110));
            Formato.setBorder(general);
        general.setVisible(true);


            // Preparamos las ETIQUETAS
        lblTitulo = new JLabel("ELIMINAR CARGAS");
        Formato.setFuenteTituloMediano(lblTitulo);
        lblCarga = new JLabel("Nombre de carga");

        //Preparamos los BOTONES
            //Generamos los botones
            btnEliminar = new JButton("Eliminar");
            btnEliminar.setPreferredSize(new Dimension(150,30));

            //Generamos los ActionListeners
        ActionListener accionbtnEliminar=new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                           // TODO Auto-generated method stub
                           ArrayList<Carga> al2 = new ArrayList<Carga>();
                     ArrayList<Carga> al = proyecto.getCargas();
                           //Si no hay cargas damos un aviso
                     boolean vacio=false;
                           if(al.size()==0) {
                                  CheckDatos.okNokMensaje("No hay cargas en el
proyecto");
                                  vacio=true;

                           }
                           if(!vacio) {
                            Iterator<Carga> it = al.iterator();
                            while (it.hasNext()) {
                                  Carga q = it.next();
                                  //Si no es el seleccionado lo metemos en el ArrayList
auxiliar
                                  if(!
q.getNombre().equals((String)jcbCarga.getSelectedItem())) {
                                         al2.add(q);
                                  }
                            }
                            //Eliminamos las cargas del proyecto
                           al = new ArrayList<Carga>();
                            //Invertimos el orden para que sigan mostrandose en el mismo
que estaban
                            Iterator<Carga> it2 = al2.iterator();
                            while (it2.hasNext()) {
                                  Carga q = it2.next();
                                         al.add(q);
                            }
```

```
                        //Actualizamos el Combo
                        jcbCarga.removeAllItems();
                        Iterator<Carga> it3 = al.iterator();
                        while (it3.hasNext()) {
                              Carga q = it3.next();
                              jcbCarga.addItem(q.getNombre());
                        }
                        jcbCarga.repaint();
                        //Añadimos las nuevas cargas al proyecto
                        proyecto.setCargas(al);
                        //actualizamos el esquema de cargas
                       actualizaEsquema();
                       }
                       muestraReacciones();
                 actualizaEsquemaCortantes();
                 actualizaEsquemaMomentos();
                 actualizaTensiones();
                 }
      };

      //Asignamos el actionListener a los botones
      btnEliminar.addActionListener(accionbtnEliminar);

            //Preparamos el comboBox
       jcbCarga=new JComboBox<String>();
       ArrayList<Carga> al = proyecto.getCargas();
       Iterator<Carga> it = al.iterator();
       while (it.hasNext()) {
            Carga q = it.next();
            jcbCarga.addItem(q.getNombre());
       }
            jcbCarga.setPreferredSize(new Dimension(120,20));


      //Marcamos las RESTRICCIONES DE COLOCACION SPRINGLAYOUT EN EL PANEL DE DATOS
      //Incluimos los elementos en los paneles las restricciones en X
      layoutDatosPanel.putConstraint(SpringLayout.EAST, jcbCarga, -2*gapX+5,
SpringLayout.EAST, datosPanel);
      layoutDatosPanel.putConstraint(SpringLayout.WEST, lblCarga, gapX+5,
SpringLayout.WEST, datosPanel);
      layoutDatosPanel.putConstraint(SpringLayout.WEST, btnEliminar, 2*gapX+5,
SpringLayout.WEST, datosPanel);

      //Incluimos los elementos en los paneles las restricciones en Y
      layoutDatosPanel.putConstraint(SpringLayout.NORTH, jcbCarga, 3*gapY,
SpringLayout.NORTH, datosPanel);
      layoutDatosPanel.putConstraint(SpringLayout.NORTH, lblCarga, 3*gapY,
SpringLayout.NORTH, datosPanel);
      layoutDatosPanel.putConstraint(SpringLayout.NORTH, btnEliminar, gapY+10,
SpringLayout.SOUTH, lblCarga);

      //Incluimos todo en la VENTANA GENERAL.
      //Incluimos los elementos en el panel las restricciones en X
      layout.putConstraint(SpringLayout.WEST, panelMarca, 7*gapX, SpringLayout.WEST,
general);
      layout.putConstraint(SpringLayout.WEST, lblTitulo, 8*gapX, SpringLayout.WEST,
general);
      layout.putConstraint(SpringLayout.WEST, datosPanel, gapX, SpringLayout.WEST,
general);
      //Incluimos los elementos en el panel las restricciones en Y
      layout.putConstraint(SpringLayout.NORTH, panelMarca, gapY, SpringLayout.NORTH,
general);
      layout.putConstraint(SpringLayout.NORTH, lblTitulo, gapY, SpringLayout.SOUTH,
panelMarca);
      layout.putConstraint(SpringLayout.NORTH, datosPanel, gapY, SpringLayout.SOUTH,
lblTitulo);
```

```
      //Metemos los componentes en los PANELES GENERAL y de RECOGIDA DE INFORMACION
      datosPanel.setLayout(layoutDatosPanel);
      datosPanel.add(jcbCarga);
      datosPanel.add(lblCarga);
      datosPanel.add(btnEliminar);
      general.setLayout(layout);
      general.add(panelMarca);
      general.add(lblTitulo);
      general.add(datosPanel);

      //Asignamos un springlayout a la ventana
      SpringLayout layoutventana = new SpringLayout();
      setLayout(layoutventana);
      //Metemos el panel general
      add(general);
      //Le damos las medidas por defecto que tenemos marcado en la variable ladopanel
            setPreferredSize(new Dimension(ladopanel,ladopanel+110));
            pack();
            this.setLocation(new Point(500,250));
            DatosGenericos.setPosicion(this, ladopanel, ladopanel+110);
            setVisible(true);


            //damos formato a todos los componentes
            c.add(lblCarga);
            c.add(jcbCarga);
            c.add(btnEliminar);
            darTipoLetra(c);
      }

    //Este metodo repinta el esquema cortantes con las cargas nuevas introducidas
    private void actualizaEsquemaCortantes() {
       cortantes.repaint();
    }

    //Este metodo repinta el esquema momentos con las cargas nuevas introducidas
    private void actualizaEsquemaMomentos() {
       momentos.repaint();
    }

    //Este metodo actualiza los valores de las tensiones
    public void actualizaTensiones() {
       LeyesCortantes.preparaParaTension();
       LeyesMomentos.preparaParaTension();
       if(proyecto.getCamion().getTipo().getTipo()=='R'){
            LeyesCortantes.preparaParaTension();
            LeyesMomentos.preparaParaTension();
            Camion c=proyecto.getCamion();

ventanaCalculo.setTensionesCabeza(c.getTensionCortanteViga(),c.getTensionNormalViga(),c.g
etTensionVonMisses());
            tensiones.repaint();
       } else {
            LeyesCortantes.preparaParaTension();
            LeyesMomentos.preparaParaTension();
            CamionArticulado c=(CamionArticulado)proyecto.getCamion();
            ventanaCalculo.setTensionesCabeza(c.getTensionCortanteViga(),
c.getTensionNormalViga(), c.getTensionVonMisses());

ventanaCalculo.setTensionesRemolque(c.getTensionCortanteVigaRemolque(),c.getTensionNormal
VigaRemolque(), c.getTensionVonMissesVigaRemolque());
            tensiones.repaint();
       }
    }

      //Este método asigna a todos los componentes de la ventana el mismo tipo de texto
      private void darTipoLetra(ArrayList<Component> c) {
```

```
            Iterator<Component> iter = c.iterator();
            while(iter.hasNext()){
               ponLetra(iter.next());
            }
       }
       //Este método da el tipo letra a cada componente igual
    private void ponLetra(Component c) {
       Formato.setFormatoEstandar(c);
    }

    //Este metodo repinta el esquema con las cargas nuevas introducidas
    private void actualizaEsquema() {
       esquema.repaint();
       ventanaCalculo.borraTabla();
       ventanaCalculo.rellenaTablaCargas();
    }

    //Este metodo actualiza el combo cuando pierde el foco al ser llamado por otras
ventanas
    public void actualizaCombo() {
       //Actualizamos el Combo
        jcbCarga.removeAllItems();
        ArrayList<Carga> al = proyecto.getCargas();
        Iterator<Carga> it = al.iterator();
        while (it.hasNext()) {
              Carga q = it.next();
              jcbCarga.addItem(q.getNombre());
        }
        jcbCarga.repaint();
    }

    //Este metodo comprueba las reacciones y avisa de errores
    public void muestraReacciones() {
       if(proyecto.getCamion().getTipo().getTipo()=='R') {
              //Actualizamos el valor
              CalculoReaccionesRigido.sumareacciones(proyecto);
              //Comprobamos las reacciones para ver si hay errores
                     CalculoReaccionesRigido.compruebaReacciones(proyecto);
                     //Actualiza en la ventana de Calculos el valor de las reacciones
                     ventanaCalculo.setReacciones(CalculoReaccionesRigido.getReaccionA(),
CalculoReaccionesRigido.getReaccionB());
       }
       if(proyecto.getCamion().getTipo().getTipo()=='A') {
              //Actualizamos el valor
              CalculoReaccionesArticulado.sumareaccionesRemolque(proyecto);
              CalculoReaccionesArticulado.sumareaccionesCabeza(proyecto);
              //Comprobamos las reacciones para ver si hay errores
                     CalculoReaccionesRigido.compruebaReacciones(proyecto);
                     //Actualiza en la ventana de Calculos el valor de las reacciones

ventanaCalculo.setReacciones(CalculoReaccionesArticulado.getReaccionA(),
CalculoReaccionesArticulado.getReaccionB(),CalculoReaccionesArticulado.getReaccionRemolqu
e(),CalculoReaccionesArticulado.getReaccionKingPin());
       }
    }

}
```

CLASE VentanaIntroduceCarga

```
/*Esta clase sirve para introducir cargas en el proyecto*/

package calculosEstructura;

import java.awt.Component;
import java.awt.Dimension;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.Iterator;

import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.SpringLayout;

import auxiliares.CheckDatos;
import auxiliares.DatosGenericos;
import auxiliares.Formato;
import auxiliares.PanelConImagen;
import auxiliares.Unidades;
import camiones.*;

public class VentanaIntroduceCarga extends JFrame{
	private static final long serialVersionUID = -7418029428404409927L;
	private static int ladopanel=340;
	private static String fondogris=DatosGenericos.getDireccion("Imagenes/fondos/
fondo-gris.png");
	private static String fondoazul=DatosGenericos.getDireccion("Imagenes/fondos/
fondo.jpg");
	private String nombre;
	private static ArrayList<Component> c = new ArrayList<Component>();
	private static ArrayList<JTextField> t = new ArrayList<JTextField>();
	private PanelConImagen general,panelMarca,datosPanel,tensiones; //Generamos los
paneles contenedores de los datos, el dibujo y la info
	private JLabel lbln,lblp,lblval,lblt,lblTitulo,lblLong; //Generamos las etiquetas
	private JTextField tfn,tfp,tfval,tflong; //Generamos los campos de texto para
recogida de los datos
	private JButton btnguardar; //Generamos el boton para calcular
	private double punto,valor,longitudQ; //Generamos las variables que contendran los
datos para el calculo
	private JComboBox<String> jcbtipo;
	private static Proyecto proyecto;
	private static PanelEsquema esquema;
	private static PanelCortantes cortantes;
	private static PanelMomentos momentos;
	private static VentanaCalculos ventanaCalculo;

	public VentanaIntroduceCarga(Proyecto p,PanelEsquema pe,PanelCortantes
pc,PanelMomentos pm,VentanaCalculos vc,PanelConImagen tensiones){
		super("Introducir carga");
		this.setResizable(false);
		proyecto=p;
		esquema=pe;
		cortantes=pc;
		momentos=pm;
		ventanaCalculo=vc;
		this.tensiones=tensiones;

		//Preparamos los PANELES
		//datos panel contiene los elementos de entrada de datos
```

```
            datosPanel=new PanelConImagen(fondogris);
            //general contiene los datos de interaccion
            general=new PanelConImagen(fondoazul);
            panelMarca=new PanelConImagen(DatosGenericos.getDireccion("Imagenes/
pantallaprincipal/recurso.png")); //contiene el logo de la aplicación
            panelMarca.setPreferredSize(new Dimension(150,80));


            //Damos formato al PANEL GENERAL
            //Damos el layout para general
            SpringLayout layout = new SpringLayout();
            //Damos el layout para datosPanel
            SpringLayout layoutDatosPanel = new SpringLayout();
            //Definimos el gap entre ventanas
        int gapX=10;
        int gapY=10;
            datosPanel.setPreferredSize(new Dimension(ladopanel-4*gapX,ladopanel-130));
            Formato.setBorder(datosPanel);
        datosPanel.setVisible(true);
            general.setPreferredSize(new Dimension(ladopanel,ladopanel+110));
            Formato.setBorder(general);
        general.setVisible(true);


            // Preparamos las ETIQUETAS
        lblTitulo = new JLabel("INTRODUCIR CARGAS ("+Unidades.getUnidadesCarga()+" y
"+Unidades.getUnidadesLongitud() +")");
        Formato.setFuenteTituloMediano(lblTitulo);
            lbln = new JLabel("Nombre de carga");
            lblp = new JLabel("Punto aplicación");
            lblp.setToolTipText("El punto desde el frontal del camión donde se sitúa la
carga en caso de ser puntual o donde empieza"
                        + "en caso de ser distribuida");
            lblval = new JLabel("Valor de carga");
            lblt = new JLabel("Tipo de carga");
            lblLong = new JLabel("Longitud distribuida");
            //Inicialmente ponemos el panel como carga puntual por lo que no se vera la
longitud
            lblLong.setVisible(false);


            //Preparamos los TEXTOS
            tfn = new JTextField("");
            tfn.setPreferredSize(new Dimension(110,20));
            Formato.setBorder(tfn);
            tfp = new JTextField("0");
            tfp.setPreferredSize(new Dimension(110,20));
            Formato.setBorder(tfp);
            tfp.setToolTipText("El punto desde el frontal del camión donde se sitúa la
carga en caso de ser puntual o donde empieza"
                        + "en caso de ser distribuida");
            tfval = new JTextField("0");
            tfval.setPreferredSize(new Dimension(110,20));
            Formato.setBorder(tfval);
            tflong = new JTextField("0");
            tflong.setPreferredSize(new Dimension(110,20));
            Formato.setBorder(tflong);
            //Inicialmente ponemos el panel como carga puntual por lo que no se vera la
longitud
            tflong.setVisible(false);


            //Preparamos los comboBox
            jcbtipo=new JComboBox<String>();
            jcbtipo.addItem("Puntual");
            jcbtipo.addItem("Distribuida");
            jcbtipo.setPreferredSize(new Dimension(120,20));
```

```
            // Accion a realizar cuando el JComboBox cambia de item seleccionado.
            jcbtipo.addActionListener(new ActionListener() {
                  @Override
                  public void actionPerformed(ActionEvent e) {
                        String seleccion=(String) jcbtipo.getSelectedItem();
                        if(seleccion=="Puntual") {
                              tflong.setText("0");
                              tflong.setVisible(false);
                              lblLong.setVisible(false);
                        }
                        else {
                              tflong.setVisible(true);
                              lblLong.setVisible(true);
                        }
                  }
            });


            //Preparamos los BOTONES DE GUARDADO
            //Generamos los botones
            btnguardar = new JButton("Guardar");
            btnguardar.setPreferredSize(new Dimension(150,30));
            //Generamos los ActionListeners
        ActionListener accionbtnguardar=new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                  boolean correccion=false;
                  boolean tieneDatos=false;
                  boolean cargaNoCero=false;
                  boolean longitudNoCero=false;
                  boolean valoresValidos=false;
                  //Obtenemos los valores introducidos
                  //Metemos todas las etiquetas en un ArrayList para comprobar y que
tengan formato adecuado
                  t.clear();
                  t.add(tfp);
                  t.add(tfval);
                  t.add(tflong);
                  //Corregimos los datos en los JTextField para ver que tienen la
informacion que deben tener
                  correccion=CheckDatos.corrigeDatos(t);
                  //Borramos los datos del ArrayList para proximas consultas
                  t.clear();
                  nombre=tfn.getText();
                  //Usamos el factor de conversion por si las unidades estan cambiadas
adecuarlas a las actuales

punto=Double.parseDouble(tfp.getText())*Unidades.getConversionLongitud();

valor=Double.parseDouble(tfval.getText())*Unidades.getConversionCarga();

longitudQ=Double.parseDouble(tflong.getText())*Unidades.getConversionLongitud();
                  //Compruebo que los datos estan bien
                  //Meto en un ArrayList los valores actuales para comprobar
                  ArrayList<String> valores = new ArrayList<String>();
                  valores.add(tfn.getText());
                  valores.add(tfp.getText());
                  valores.add(tfval.getText());
                  //Si es articulado, ademas meto la longitud de la carga
                  if(((String)jcbtipo.getSelectedItem()).equals("Distribuida"))
valores.add(tflong.getText());
                  tieneDatos=CheckDatos.compruebaDatosCompletosCargas(valores);
                  cargaNoCero=CheckDatos.compruebaNoCero(tfval);
                  longitudNoCero=CheckDatos.compruebaNoCero(tflong);

valoresValidos=CheckDatos.compruebaValoresValidosCargas(proyecto,tfn,tfp,tflong,jcbtipo);
                  if(!tieneDatos&&(correccion)&&(valoresValidos))
                        {
```

```
                                    //Uso el generador de mensajes de CheckDatos
                                    CheckDatos.okNokMensaje("Hay valores sin rellenar");
                              }
                        if((tieneDatos)&&(!cargaNoCero)&&(correccion)&&(valoresValidos))
                                    {
                                    //Uso el generador de mensajes de CheckDatos
                                    CheckDatos.okNokMensaje("La carga no puede ser 0");
                                    }
                        if((tieneDatos)&&(cargaNoCero)&&(!
longitudNoCero)&&((String)jcbtipo.getSelectedItem()).equals("Distribuida")&&(correccion)&
&(valoresValidos))
                              {
                                    //Uso el generador de mensajes de CheckDatos
                                    CheckDatos.okNokMensaje("La longitud de la carga
distribuida no puede ser 0");
                              }
                        //Creamos una carga Puntual metiendola en el proyecto
                        if(((String)jcbtipo.getSelectedItem()).equals("Puntual")) {
                                    if(correccion&&tieneDatos&&cargaNoCero&&valoresValidos)
{
                                          proyecto.addPuntual(nombre, punto, valor, true);
                                    //Reseteamos los valores para nuevas cargas
                                          reseteaValores();
                                    }
                              }
                        else
if(correccion&&tieneDatos&&cargaNoCero&&longitudNoCero&&valoresValidos) {
                                    //Si el camion es articulado y la carga es
distribuida y está en zona de cabeza y de remolque
                                    //en vez de generar solo una carga distribuidad,
generamos 2 cargas usando la introducida
                                    //dividida desde en el punto de comienzo de la
caja o el final de la cabeza (si la ca
                                    if(proyecto.getCamion().getTipo().getTipo()=='A')
{
                                          double
parteExtra=DatosGenericos.getLongExtra();
                                          double
puntoInicioCaja=((CamionArticulado)proyecto.getCamion()).getdistFrenteEjeDelantero()+
((CamionArticulado)proyecto.getCamion()).getDistanciaEjeDelanteroComienzoCaja();
                                          double
puntoFinalCabeza=((CamionArticulado)proyecto.getCamion()).getdistFrenteEjeDelantero()+
((CamionArticulado)proyecto.getCamion()).getdistEjeDelanteroEjeTrasero()+parteExtra;
                                          double
puntoDivision=Math.min(puntoInicioCaja, puntoFinalCabeza);
                                          if(punto<puntoDivision) { //Esta en cabeza
                                                if((punto+longitudQ)>puntoDivision)
{
                                                      boolean
procederDivision=false;

procederDivision=CheckDatos.okCancelaMensaje("La carga pasa por la cabeza tractora y el
remolque, se dividirá en dos subcargas para el procesado");
                                                      if(procederDivision) {

proyecto.addDistribuida(nombre+"-Cabeza", punto, valor, true,(puntoDivision-punto));

proyecto.addDistribuida(nombre+"-Remolque", puntoDivision, valor, true,
((punto+longitudQ)-puntoDivision));
                                                //Reseteamos los valores para
nuevas cargas
                                                      reseteaValores();
                                                      }

                                                } else
proyecto.addDistribuida(nombre, punto, valor, true,longitudQ);
```

```
                                                //Reseteamos los valores para nuevas
cargas
                                                      reseteaValores();
                                                } else proyecto.addDistribuida(nombre,
punto, valor, true,longitudQ);
                                          //Reseteamos los valores para nuevas cargas
                                                reseteaValores();
                                          } else {
                                                proyecto.addDistribuida(nombre, punto,
valor, true,longitudQ);
                                          //Reseteamos los valores para nuevas cargas
                                                reseteaValores();
                                          }
                     }
                           //actualizamos el esquema de cargas
                           actualizaEsquema();
                     //Rellenamos la tabla
                     ventanaCalculo.rellenaTablaCargas();
                     //Actualizamos reacciones con todas las cargas
                     muestraReacciones();
                     //Actualizamos Esquema Cortantes
                     actualizaEsquemaCortantes();
                     //Actualizamos Esquema Momentos
                     actualizaEsquemaMomentos();
                     //Actualizamos las Tensiones
                     actualizaTensiones();
                     //Actualizamos las ventanas de Ver Cargas y Eliminar Cargas si
existen para reflejar las nuevas cargas
                     ventanaCalculo.actualizaVentanas();
                     //Redibujamos la ventana de cálculo para introducir las novedades en
la tabla
                     ventanaCalculo.repintaTabla();
                     }
       };

       //Asignamos el actionListener a los botones
       btnguardar.addActionListener(accionbtnguardar);


       //Marcamos las RESTRICCIONES DE COLOCACION SPRINGLAYOUT EN EL PANEL DE DATOS
       //Incluimos los elementos en los paneles las restricciones en X
       layoutDatosPanel.putConstraint(SpringLayout.WEST, lbln, 2*gapX, SpringLayout.WEST,
datosPanel);
       layoutDatosPanel.putConstraint(SpringLayout.WEST, lblp, 2*gapX, SpringLayout.WEST,
datosPanel);
       layoutDatosPanel.putConstraint(SpringLayout.WEST, lblval, 2*gapX,
SpringLayout.WEST, datosPanel);
       layoutDatosPanel.putConstraint(SpringLayout.WEST, lblt, 2*gapX, SpringLayout.WEST,
datosPanel);
       layoutDatosPanel.putConstraint(SpringLayout.WEST, lblLong, 2*gapX,
SpringLayout.WEST, datosPanel);
       layoutDatosPanel.putConstraint(SpringLayout.EAST, tfn, -2*gapX, SpringLayout.EAST,
datosPanel);
       layoutDatosPanel.putConstraint(SpringLayout.EAST, tfp, -2*gapX, SpringLayout.EAST,
datosPanel);
       layoutDatosPanel.putConstraint(SpringLayout.EAST, tfval, -2*gapX,
SpringLayout.EAST, datosPanel);
       layoutDatosPanel.putConstraint(SpringLayout.EAST, jcbtipo, -2*gapX+5,
SpringLayout.EAST, datosPanel);
       layoutDatosPanel.putConstraint(SpringLayout.EAST, tflong, -2*gapX,
SpringLayout.EAST, datosPanel);
       //Incluimos los elementos en los paneles las restricciones en Y
       layoutDatosPanel.putConstraint(SpringLayout.NORTH, lbln, gapY, SpringLayout.NORTH,
datosPanel);
       layoutDatosPanel.putConstraint(SpringLayout.NORTH, lblp, gapY, SpringLayout.SOUTH,
lbln);
```

```
	layoutDatosPanel.putConstraint(SpringLayout.NORTH, lblval, gapY,
SpringLayout.SOUTH, lblp);
	layoutDatosPanel.putConstraint(SpringLayout.NORTH, lblt, gapY, SpringLayout.SOUTH,
lblval);
	layoutDatosPanel.putConstraint(SpringLayout.NORTH, lblLong, gapY,
SpringLayout.SOUTH, lblt);
	layoutDatosPanel.putConstraint(SpringLayout.NORTH, tfn, gapY, SpringLayout.NORTH,
datosPanel);
	layoutDatosPanel.putConstraint(SpringLayout.NORTH, tfp, gapY, SpringLayout.SOUTH,
tfn);
	layoutDatosPanel.putConstraint(SpringLayout.NORTH, tfval, gapY,
SpringLayout.SOUTH, tfp);
	layoutDatosPanel.putConstraint(SpringLayout.NORTH, jcbtipo, gapY,
SpringLayout.SOUTH, tfval);
	layoutDatosPanel.putConstraint(SpringLayout.NORTH, tflong, gapY,
SpringLayout.SOUTH, jcbtipo);

	//Incluimos todo en la VENTANA GENERAL.
	//Incluimos los elementos en el panel las restricciones en X
	layout.putConstraint(SpringLayout.WEST, panelMarca, 7*gapX, SpringLayout.WEST,
general);
	layout.putConstraint(SpringLayout.WEST, lblTitulo, 3*gapX, SpringLayout.WEST,
general);
	layout.putConstraint(SpringLayout.WEST, datosPanel, gapX, SpringLayout.WEST,
general);
	layout.putConstraint(SpringLayout.WEST, btnguardar, gapX, SpringLayout.WEST,
general);
	//Incluimos los elementos en el panel las restricciones en Y
	layout.putConstraint(SpringLayout.NORTH, panelMarca, gapY, SpringLayout.NORTH,
general);
	layout.putConstraint(SpringLayout.NORTH, lblTitulo, gapY, SpringLayout.SOUTH,
panelMarca);
	layout.putConstraint(SpringLayout.NORTH, datosPanel, gapY, SpringLayout.SOUTH,
lblTitulo);
	layout.putConstraint(SpringLayout.NORTH, btnguardar, gapY, SpringLayout.SOUTH,
datosPanel);

	//Metemos los componentes en los PANELES GENERAL y de RECOGIDA DE INFORMACION
	datosPanel.setLayout(layoutDatosPanel);
	datosPanel.add(lbln);
	datosPanel.add(lblp);
	datosPanel.add(lblval);
	datosPanel.add(lblt);
	datosPanel.add(lblLong);
	datosPanel.add(tfn);
	datosPanel.add(tfp);
	datosPanel.add(tfval);
	datosPanel.add(jcbtipo);
	datosPanel.add(tflong);
	general.setLayout(layout);
	general.add(panelMarca);
	general.add(lblTitulo);
	general.add(datosPanel);
	general.add(btnguardar);

	//Asignamos un springlayout a la ventana
	SpringLayout layoutventana = new SpringLayout();
	setLayout(layoutventana);
	//Metemos el panel general
	add(general);
	//Le damos las medidas por defecto que tenemos marcado en la variable ladopanel
		setPreferredSize(new Dimension(ladopanel,ladopanel+110));
		pack();
		this.setLocation(new Point(500,250));
		DatosGenericos.setPosicion(this, ladopanel, ladopanel+110);
		setVisible(true);
```

```
            //damos formato a todos los componentes
            c.add(lbln);
            c.add(lblp);
            c.add(lblval);
            c.add(lblt);
            c.add(lblLong);
            c.add(tfn);
            c.add(tfp);
            c.add(tflong);
            c.add(tfval);
            c.add(jcbtipo);
            c.add(btnguardar);
            darTipoLetra(c);
      }

    //Este método asigna a todos los componentes de la ventana el mismo tipo de texto
    private void darTipoLetra(ArrayList<Component> c) {
        Iterator<Component> iter = c.iterator();
        while(iter.hasNext()){
            Formato.setFormatoEstandar(iter.next());
        }
    }


    //Este metodo repinta el esquema con las cargas nuevas introducidas
    private void actualizaEsquema() {
       esquema.repaint();
    }

    //Este metodo repinta el esquema cortantes con las cargas nuevas introducidas
    private void actualizaEsquemaCortantes() {
       cortantes.repaint();
    }

    //Este metodo repinta el esquema momentos con las cargas nuevas introducidas
    private void actualizaEsquemaMomentos() {
       momentos.repaint();
    }

    //Este metodo repinta el esquema momentos con las cargas nuevas introducidas
    private void actualizaTensiones() {
       LeyesCortantes.preparaParaTension();
       LeyesMomentos.preparaParaTension();
       if(proyecto.getCamion().getTipo().getTipo()=='R'){
            LeyesCortantes.preparaParaTension();
            LeyesMomentos.preparaParaTension();
            Camion c=proyecto.getCamion();

ventanaCalculo.setTensionesCabeza(c.getTensionCortanteViga(),c.getTensionNormalViga(),c.g
etTensionVonMisses());
            tensiones.repaint();
       } else {
            LeyesCortantes.preparaParaTension();
            LeyesMomentos.preparaParaTension();
            CamionArticulado c=(CamionArticulado)proyecto.getCamion();
            ventanaCalculo.setTensionesCabeza(c.getTensionCortanteViga(),
c.getTensionNormalViga(), c.getTensionVonMisses());

ventanaCalculo.setTensionesRemolque(c.getTensionCortanteVigaRemolque(),c.getTensionNormal
VigaRemolque(), c.getTensionVonMissesVigaRemolque());
            tensiones.repaint();
       }
       }

    //Este metodo resetea los valores
    private void reseteaValores() {
       //Borramos el contenido de las celdas
```

```
       tfval.setText("0");
       tfp.setText("0");
       tflong.setText("0");
    }
    //Devuelve el proyecto para actualizar valores
    public Proyecto getProyecto() {
       return proyecto;
    }

    //Este metodo comprueba las reacciones y avisa de errores
    public void muestraReacciones() {
       if(proyecto.getCamion().getTipo().getTipo()=='R') {
              //Actualizamos el valor
              CalculoReaccionesRigido.sumareacciones(proyecto);
              //Comprobamos las reacciones para ver si hay errores
                     CalculoReaccionesRigido.compruebaReacciones(proyecto);
                     //Actualiza en la ventana de Calculos el valor de las reacciones
                     ventanaCalculo.setReacciones(CalculoReaccionesRigido.getReaccionA(),
CalculoReaccionesRigido.getReaccionB());
       }
       if(proyecto.getCamion().getTipo().getTipo()=='A') {
              //Actualizamos el valor
              CalculoReaccionesArticulado.sumareaccionesRemolque(proyecto);
              CalculoReaccionesArticulado.sumareaccionesCabeza(proyecto);
              //Comprobamos las reacciones para ver si hay errores
                     CalculoReaccionesRigido.compruebaReacciones(proyecto);
                     //Actualiza en la ventana de Calculos el valor de las reacciones

ventanaCalculo.setReacciones(CalculoReaccionesArticulado.getReaccionA(),
CalculoReaccionesArticulado.getReaccionB(),CalculoReaccionesArticulado.getReaccionRemolqu
e(),CalculoReaccionesArticulado.getReaccionKingPin());
       }
    }

    //Este metodo actualiza el valor de las unidades que muestra
    public void actualizaUnidades() {
       lblTitulo.setText("INTRODUCIR CARGAS ("+Unidades.getUnidadesCarga()+" y
"+Unidades.getUnidadesLongitud() +")");
    }
}
```

CLASE VentanaSeleccionParteArticulado

```
/*Esta clase sirve para introducir cargas en el proyecto*/

package calculosEstructura;

import java.awt.Color;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.Iterator;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.SpringLayout;
import camiones.*;

import auxiliares.CheckDatos;
import auxiliares.DatosGenericos;
import auxiliares.Formato;
import auxiliares.PanelConImagen;
import cargasEstructura.Carga;

public class VentanaSeleccionParteArticulado extends JFrame{
	private static final long serialVersionUID = 1L;
	private static int ladopanel=300;
	private static String fondogris=DatosGenericos.getDireccion("Imagenes/fondos/
fondo-gris.png");
	private static String fondoazul=DatosGenericos.getDireccion("Imagenes/fondos/
fondo.jpg");
	private static ArrayList<Component> c = new ArrayList<Component>();
	private static ArrayList<JTextField> t = new ArrayList<JTextField>();
	private PanelConImagen general,panelMarca,datosPanel; //Generamos los paneles
contenedores de los datos, el dibujo y la info
	private JLabel lblCarga,lblTitulo; //Generamos las etiquetas
	private JButton btnVer,btnOcultar,btnAislar; //Generamos el boton para calcular
	private JComboBox<String> jcbCarga;
	private static Proyecto proyecto;
	private static PanelEsquema esquema;
	private static boolean comboActualizado;

	public VentanaSeleccionParteArticulado(Proyecto p,PanelEsquema pe,VentanaCalculos
v){
		super("Visualizar Carga");
		this.setResizable(false);
		proyecto=p;
		esquema=pe;
		comboActualizado=false;

		//Preparamos los PANELES
		//datos panel contiene los elementos de entrada d2e datos
		datosPanel=new PanelConImagen(fondogris);
		//general contiene los datos de interaccion
		general=new PanelConImagen(fondoazul);
		panelMarca=new PanelConImagen(DatosGenericos.getDireccion("Imagenes/
pantallaprincipal/recurso.png")); //contiene el logo de la aplicación
		panelMarca.setPreferredSize(new Dimension(150,80));


		//Damos formato al PANEL GENERAL
```

```
            //Damos el layout para general
            SpringLayout layout = new SpringLayout();
            //Damos el layout para datosPanel
            SpringLayout layoutDatosPanel = new SpringLayout();
            //Definimos el gap entre ventanas
        int gapX=10;
        int gapY=10;
            datosPanel.setPreferredSize(new Dimension(ladopanel-2*gapX,ladopanel-130));
            Formato.setBorder(datosPanel);
        datosPanel.setVisible(true);
            general.setPreferredSize(new Dimension(ladopanel,ladopanel+110));
            Formato.setBorder(general);
        general.setVisible(true);


            // Preparamos las ETIQUETAS
        lblTitulo = new JLabel("VISUALIZAR CARGAS");
        Formato.setFuenteTituloMediano(lblTitulo);
        lblCarga = new JLabel("Nombre de carga");

        //Preparamos los BOTONES
            //Generamos los botones
            btnVer = new JButton("Ver");
            btnOcultar = new JButton("Ocultar");
            btnAislar=new JButton("Aislar");
            btnVer.setPreferredSize(new Dimension(150,30));
            btnOcultar.setPreferredSize(new Dimension(150,30));
            btnAislar.setPreferredSize(new Dimension(150,30));

            //Generamos los ActionListeners
        ActionListener accionbtnVer=new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                            // TODO Auto-generated method stub
                     ArrayList<Carga> al = proyecto.getCargas();
                     //Vemos si las cargas están vacias y enviamos un mensaje si no hay
cargas
                     if(al.size()==0) CheckDatos.okNokMensaje("No hay cargas en el
proyecto");
                     Iterator<Carga> it = al.iterator();
                     while (it.hasNext()) {
                            Carga q = it.next();
                            if(q.getNombre().equals((String)jcbCarga.getSelectedItem())) {
                                    q.setVisible(true);
                            }
                     }
                     //actualizamos el esquema de cargas
                    actualizaEsquema();
                    }
       };

        ActionListener accionbtnOcultar=new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                            // TODO Auto-generated method stub
                     ArrayList<Carga> al = proyecto.getCargas();
                     //Vemos si las cargas están vacias y enviamos un mensaje si no hay
cargas
                     if(al.size()==0) CheckDatos.okNokMensaje("No hay cargas en el
proyecto");
                     Iterator<Carga> it = al.iterator();
                     while (it.hasNext()) {
                            Carga q = it.next();
                            if(q.getNombre().equals((String)jcbCarga.getSelectedItem())) {
                                    q.setVisible(false);
                            }
                     }
```

```
                    //actualizamos el esquema de cargas
                   actualizaEsquema();
                   }
      };

       ActionListener accionbtnAislar=new ActionListener() {
                   @Override
                   public void actionPerformed(ActionEvent e) {
                         // TODO Auto-generated method stub
                    ArrayList<Carga> al = proyecto.getCargas();
                    //Vemos si las cargas están vacias y enviamos un mensaje si no hay
cargas
                    if(al.size()==0) CheckDatos.okNokMensaje("No hay cargas en el
proyecto");
                    Iterator<Carga> it = al.iterator();
                    while (it.hasNext()) {
                         Carga q = it.next();
                         if(q.getNombre().equals((String)jcbCarga.getSelectedItem())) {
                               q.setVisible(true);
                         }
                         else q.setVisible(false);
                    }
                    //actualizamos el esquema de cargas
                   actualizaEsquema();
                   }
      };

      //Asignamos el actionListener a los botones
      btnVer.addActionListener(accionbtnVer);
      btnOcultar.addActionListener(accionbtnOcultar);
      btnAislar.addActionListener(accionbtnAislar);


      //Preparamos el comboBox
       jcbCarga=new JComboBox<String>();
       ArrayList<Carga> al = proyecto.getCargas();
       Iterator<Carga> it = al.iterator();
       while (it.hasNext()) {
            Carga q = it.next();
            jcbCarga.addItem(q.getNombre());
       }
            jcbCarga.setPreferredSize(new Dimension(120,20));


      //Marcamos las RESTRICCIONES DE COLOCACION SPRINGLAYOUT EN EL PANEL DE DATOS
      //Incluimos los elementos en los paneles las restricciones en X
      layoutDatosPanel.putConstraint(SpringLayout.EAST, jcbCarga, -2*gapX+5,
SpringLayout.EAST, datosPanel);
      layoutDatosPanel.putConstraint(SpringLayout.WEST, lblCarga, gapX+5,
SpringLayout.WEST, datosPanel);
      layoutDatosPanel.putConstraint(SpringLayout.WEST, btnVer, 2*gapX+5,
SpringLayout.WEST, datosPanel);
      layoutDatosPanel.putConstraint(SpringLayout.WEST, btnOcultar, 2*gapX+5,
SpringLayout.WEST, datosPanel);
      layoutDatosPanel.putConstraint(SpringLayout.WEST, btnAislar, 2*gapX+5,
SpringLayout.WEST, datosPanel);

      //Incluimos los elementos en los paneles las restricciones en Y
      layoutDatosPanel.putConstraint(SpringLayout.NORTH, jcbCarga, 3*gapY,
SpringLayout.NORTH, datosPanel);
      layoutDatosPanel.putConstraint(SpringLayout.NORTH, lblCarga, 3*gapY,
SpringLayout.NORTH, datosPanel);
      layoutDatosPanel.putConstraint(SpringLayout.NORTH, btnVer, gapY+10,
SpringLayout.SOUTH, lblCarga);
      layoutDatosPanel.putConstraint(SpringLayout.NORTH, btnOcultar, gapY-10,
SpringLayout.SOUTH, btnVer);
```

```
      layoutDatosPanel.putConstraint(SpringLayout.NORTH, btnAislar, gapY-10,
SpringLayout.SOUTH, btnOcultar);

      //Incluimos todo en la VENTANA GENERAL.
      //Incluimos los elementos en el panel las restricciones en X
      layout.putConstraint(SpringLayout.WEST, panelMarca, 7*gapX, SpringLayout.WEST,
general);
      layout.putConstraint(SpringLayout.WEST, lblTitulo, 7*gapX, SpringLayout.WEST,
general);
      layout.putConstraint(SpringLayout.WEST, datosPanel, gapX, SpringLayout.WEST,
general);
      //Incluimos los elementos en el panel las restricciones en Y
      layout.putConstraint(SpringLayout.NORTH, panelMarca, gapY, SpringLayout.NORTH,
general);
      layout.putConstraint(SpringLayout.NORTH, lblTitulo, gapY, SpringLayout.SOUTH,
panelMarca);
      layout.putConstraint(SpringLayout.NORTH, datosPanel, gapY, SpringLayout.SOUTH,
lblTitulo);

      //Metemos los componentes en los PANELES GENERAL y de RECOGIDA DE INFORMACION
      datosPanel.setLayout(layoutDatosPanel);
      datosPanel.add(jcbCarga);
      datosPanel.add(lblCarga);
      datosPanel.add(btnVer);
      datosPanel.add(btnOcultar);
      datosPanel.add(btnAislar);
      general.setLayout(layout);
      general.add(panelMarca);
      general.add(lblTitulo);
      general.add(datosPanel);

      //Asignamos un springlayout a la ventana
      SpringLayout layoutventana = new SpringLayout();
      setLayout(layoutventana);
      //Metemos el panel general
      add(general);
      //Le damos las medidas por defecto que tenemos marcado en la variable ladopanel
            setPreferredSize(new Dimension(ladopanel,ladopanel+110));
            pack();
            this.setLocation(new Point(500,250));
            this.setAlwaysOnTop(true);
            setVisible(true);

            //damos formato a todos los componentes
            c.add(lblCarga);
            c.add(jcbCarga);
            c.add(btnVer);
            c.add(btnOcultar);
            c.add(btnAislar);
            darTipoLetra(c);
      }

      //Este método asigna a todos los componentes de la ventana el mismo tipo de texto
      private void darTipoLetra(ArrayList<Component> c) {
          Iterator<Component> iter = c.iterator();
          while(iter.hasNext()){
             ponLetra(iter.next());
          }
      }
      //Este método da el tipo letra a cada componente igual
    private void ponLetra(Component c) {
       Formato.setFormatoEstandar(c);
    }

    //Este metodo repinta el esquema con las cargas nuevas introducidas
    private void actualizaEsquema() {
       esquema.repaint();
```

```
        }
}
```

CLASE VentanaVerCarga

```
/*Esta clase sirve para introducir cargas en el proyecto*/

package calculosEstructura;

import java.awt.Color;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.Iterator;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.SpringLayout;

import auxiliares.CheckDatos;
import auxiliares.DatosGenericos;
import auxiliares.Formato;
import auxiliares.PanelConImagen;
import camiones.*;
import cargasEstructura.Carga;

public class VentanaVerCarga extends JFrame{
	private static final long serialVersionUID = 1L;
	private static int ladopanel=340;
	private static String fondogris=DatosGenericos.getDireccion("Imagenes/fondos/
fondo-gris.png");
	private static String fondoazul=DatosGenericos.getDireccion("Imagenes/fondos/
fondo.jpg");
	private static ArrayList<Component> c = new ArrayList<Component>();
	private static ArrayList<JTextField> t = new ArrayList<JTextField>();
	private PanelConImagen general,panelMarca,datosPanel; //Generamos los paneles
contenedores de los datos, el dibujo y la info
	private JLabel lblCarga,lblTitulo; //Generamos las etiquetas
	private JButton btnVer,btnOcultar,btnAislar; //Generamos el boton para calcular
	private JComboBox<String> jcbCarga;
	private static Proyecto proyecto;
	private static PanelEsquema esquema;
	private static boolean comboActualizado;

	public VentanaVerCarga(Proyecto p,PanelEsquema pe,VentanaCalculos v){
		super("Visualizar carga");
		this.setResizable(false);
		proyecto=p;
		esquema=pe;
		comboActualizado=false;

		//Preparamos los PANELES
		//datos panel contiene los elementos de entrada d2e datos
		datosPanel=new PanelConImagen(fondogris);
		//general contiene los datos de interaccion
		general=new PanelConImagen(fondoazul);
		panelMarca=new PanelConImagen(DatosGenericos.getDireccion("Imagenes/
pantallaprincipal/recurso.png")); //contiene el logo de la aplicación
		panelMarca.setPreferredSize(new Dimension(150,80));


		//Damos formato al PANEL GENERAL
```

```
            //Damos el layout para general
            SpringLayout layout = new SpringLayout();
            //Damos el layout para datosPanel
            SpringLayout layoutDatosPanel = new SpringLayout();
            //Definimos el gap entre ventanas
        int gapX=10;
        int gapY=10;
            datosPanel.setPreferredSize(new Dimension(ladopanel-4*gapX,ladopanel-130));
            Formato.setBorder(datosPanel);
        datosPanel.setVisible(true);
            general.setPreferredSize(new Dimension(ladopanel,ladopanel+110));
            Formato.setBorder(general);
        general.setVisible(true);


            // Preparamos las ETIQUETAS
        lblTitulo = new JLabel("VISUALIZAR CARGAS");
        Formato.setFuenteTituloMediano(lblTitulo);
        lblCarga = new JLabel("Nombre de carga");

        //Preparamos los BOTONES
            //Generamos los botones
            btnVer = new JButton("Ver");
            btnOcultar = new JButton("Ocultar");
            btnAislar=new JButton("Aislar");
            btnVer.setPreferredSize(new Dimension(150,30));
            btnOcultar.setPreferredSize(new Dimension(150,30));
            btnAislar.setPreferredSize(new Dimension(150,30));

            //Generamos los ActionListeners
        ActionListener accionbtnVer=new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                            // TODO Auto-generated method stub
                     ArrayList<Carga> al = proyecto.getCargas();
                     //Vemos si las cargas están vacias y enviamos un mensaje si no hay
cargas
                     if(al.size()==0) CheckDatos.okNokMensaje("No hay cargas en el
proyecto");
                     Iterator<Carga> it = al.iterator();
                     while (it.hasNext()) {
                            Carga q = it.next();
                            if(q.getNombre().equals((String)jcbCarga.getSelectedItem())) {
                                    q.setVisible(true);
                            }
                     }
                     //actualizamos el esquema de cargas
                    actualizaEsquema();
                    }
       };

        ActionListener accionbtnOcultar=new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
                            // TODO Auto-generated method stub
                     ArrayList<Carga> al = proyecto.getCargas();
                     //Vemos si las cargas están vacias y enviamos un mensaje si no hay
cargas
                     if(al.size()==0) CheckDatos.okNokMensaje("No hay cargas en el
proyecto");
                     Iterator<Carga> it = al.iterator();
                     while (it.hasNext()) {
                            Carga q = it.next();
                            if(q.getNombre().equals((String)jcbCarga.getSelectedItem())) {
                                    q.setVisible(false);
                            }
                     }
```

```
                    //actualizamos el esquema de cargas
                   actualizaEsquema();
                   }
      };

       ActionListener accionbtnAislar=new ActionListener() {
                   @Override
                   public void actionPerformed(ActionEvent e) {
                         // TODO Auto-generated method stub
                    ArrayList<Carga> al = proyecto.getCargas();
                    //Vemos si las cargas están vacias y enviamos un mensaje si no hay
cargas
                    if(al.size()==0) CheckDatos.okNokMensaje("No hay cargas en el
proyecto");
                    Iterator<Carga> it = al.iterator();
                    while (it.hasNext()) {
                         Carga q = it.next();
                         if(q.getNombre().equals((String)jcbCarga.getSelectedItem())) {
                               q.setVisible(true);
                         }
                         else q.setVisible(false);
                    }
                    //actualizamos el esquema de cargas
                   actualizaEsquema();
                   }
      };

      //Asignamos el actionListener a los botones
      btnVer.addActionListener(accionbtnVer);
      btnOcultar.addActionListener(accionbtnOcultar);
      btnAislar.addActionListener(accionbtnAislar);


      //Preparamos el comboBox
       jcbCarga=new JComboBox<String>();
       ArrayList<Carga> al = proyecto.getCargas();
       Iterator<Carga> it = al.iterator();
       while (it.hasNext()) {
            Carga q = it.next();
            jcbCarga.addItem(q.getNombre());
       }
            jcbCarga.setPreferredSize(new Dimension(120,20));


      //Marcamos las RESTRICCIONES DE COLOCACION SPRINGLAYOUT EN EL PANEL DE DATOS
      //Incluimos los elementos en los paneles las restricciones en X
      layoutDatosPanel.putConstraint(SpringLayout.EAST, jcbCarga, -2*gapX+5,
SpringLayout.EAST, datosPanel);
      layoutDatosPanel.putConstraint(SpringLayout.WEST, lblCarga, gapX+5,
SpringLayout.WEST, datosPanel);
      layoutDatosPanel.putConstraint(SpringLayout.WEST, btnVer, 2*gapX+5,
SpringLayout.WEST, datosPanel);
      layoutDatosPanel.putConstraint(SpringLayout.WEST, btnOcultar, 2*gapX+5,
SpringLayout.WEST, datosPanel);
      layoutDatosPanel.putConstraint(SpringLayout.WEST, btnAislar, 2*gapX+5,
SpringLayout.WEST, datosPanel);

      //Incluimos los elementos en los paneles las restricciones en Y
      layoutDatosPanel.putConstraint(SpringLayout.NORTH, jcbCarga, 3*gapY,
SpringLayout.NORTH, datosPanel);
      layoutDatosPanel.putConstraint(SpringLayout.NORTH, lblCarga, 3*gapY,
SpringLayout.NORTH, datosPanel);
      layoutDatosPanel.putConstraint(SpringLayout.NORTH, btnVer, gapY+10,
SpringLayout.SOUTH, lblCarga);
      layoutDatosPanel.putConstraint(SpringLayout.NORTH, btnOcultar, gapY-10,
SpringLayout.SOUTH, btnVer);
```

```
      layoutDatosPanel.putConstraint(SpringLayout.NORTH, btnAislar, gapY-10,
SpringLayout.SOUTH, btnOcultar);

      //Incluimos todo en la VENTANA GENERAL.
      //Incluimos los elementos en el panel las restricciones en X
      layout.putConstraint(SpringLayout.WEST, panelMarca, 7*gapX, SpringLayout.WEST,
general);
      layout.putConstraint(SpringLayout.WEST, lblTitulo, 7*gapX, SpringLayout.WEST,
general);
      layout.putConstraint(SpringLayout.WEST, datosPanel, gapX, SpringLayout.WEST,
general);
      //Incluimos los elementos en el panel las restricciones en Y
      layout.putConstraint(SpringLayout.NORTH, panelMarca, gapY, SpringLayout.NORTH,
general);
      layout.putConstraint(SpringLayout.NORTH, lblTitulo, gapY, SpringLayout.SOUTH,
panelMarca);
      layout.putConstraint(SpringLayout.NORTH, datosPanel, gapY, SpringLayout.SOUTH,
lblTitulo);

      //Metemos los componentes en los PANELES GENERAL y de RECOGIDA DE INFORMACION
      datosPanel.setLayout(layoutDatosPanel);
      datosPanel.add(jcbCarga);
      datosPanel.add(lblCarga);
      datosPanel.add(btnVer);
      datosPanel.add(btnOcultar);
      datosPanel.add(btnAislar);
      general.setLayout(layout);
      general.add(panelMarca);
      general.add(lblTitulo);
      general.add(datosPanel);

      //Asignamos un springlayout a la ventana
      SpringLayout layoutventana = new SpringLayout();
      setLayout(layoutventana);
      //Metemos el panel general
      add(general);
      //Le damos las medidas por defecto que tenemos marcado en la variable ladopanel
            setPreferredSize(new Dimension(ladopanel,ladopanel+110));
            pack();
            this.setLocation(new Point(500,250));
            DatosGenericos.setPosicion(this, ladopanel, ladopanel+110);
            setVisible(true);

            //damos formato a todos los componentes
            c.add(lblCarga);
            c.add(jcbCarga);
            c.add(btnVer);
            c.add(btnOcultar);
            c.add(btnAislar);
            darTipoLetra(c);
      }

      //Este método asigna a todos los componentes de la ventana el mismo tipo de texto
      private void darTipoLetra(ArrayList<Component> c) {
          Iterator<Component> iter = c.iterator();
          while(iter.hasNext()){
             ponLetra(iter.next());
          }
      }
      //Este método da el tipo letra a cada componente igual
    private void ponLetra(Component c) {
      Formato.setFormatoEstandar(c);
    }

    //Este metodo repinta el esquema con las cargas nuevas introducidas
    private void actualizaEsquema() {
      esquema.repaint();
```

```
    }

    //Este metodo actualiza las cargas que quedan
    public void actualizaCombo() {
       jcbCarga.removeAllItems();
       ArrayList<Carga> al = proyecto.getCargas();
        Iterator<Carga> it = al.iterator();
        while (it.hasNext()) {
             Carga q = it.next();
             jcbCarga.addItem(q.getNombre());
        }
        jcbCarga.repaint();
    }

    //Este metodo repinta la ventana
    public void repaint() {
       super.repaint();
       actualizaCombo();
    }

}
```

C.5. Paquete basesDatos

Se expone a continuación el contenido del paquete basesDatos que recoge el código que se encarga de la creación y manipulación de bases de datos de secciones y camiones.

CLASE BaseDatosCamiones

```
package basesDatos;

import java.io.File;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.Statement;
import java.util.ArrayList;

import auxiliares.CheckDatos;
import auxiliares.DatosGenericos;
import camiones.*;


public class BaseDatosCamiones{
	public ArrayList<ArrayList<Object>> camiones = new ArrayList<ArrayList<Object>>();
	public ArrayList<Object> al = new ArrayList<Object>();

	public void CrearDB() {
		//Preparamos la conexion
		Connection con;
		//Usamos el separador de archivos
		String proyecto = DatosGenericos.getDireccionDB("Vehiculos");
		File url = new File(proyecto);

		//Si no existe la creamos
		if(!url.exists()) {
			try {
				//Usamos el driver embebido de Apache Derby
				Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
				//Creamos la base de datos si no existe en la ubicacion
				String db = "jdbc:derby:"+proyecto+";create=true";
				//Generamos la conexion
				con=DriverManager.getConnection(db);

				//Preparamos el string que contendrá el statement
				String tabla = "create table Camiones( Id INT PRIMARY KEY,
MARCA VARCHAR(50), MODELO VARCHAR(50), PMA DOUBLE,"
					+ "MMA DOUBLE, TIPO VARCHAR(1), PESODELANTERO
DOUBLE, EJESDELANTERO INT, PESOTRASERO DOUBLE, EJESTRASERO INT,"
					+ "PESOREMOLQUE DOUBLE, EJESREMOLQUE INT,
FRENTEDELANTERO DOUBLE, DELANTEROCAJA DOUBLE,"
					+ "DELANTEROTRASERO DOUBLE, DELANTKINGPIN DOUBLE,
KINGPINREM DOUBLE, LONGITUD DOUBLE)";
				//Generamos el statement
				PreparedStatement ps = con.prepareStatement(tabla);
				//Ejecutamos y cerramos
				ps.execute();
				ps.close();
				CheckDatos.okCancelaMensaje("Base de Datos Creada");
			}catch (Exception ex) {
				CheckDatos.okCancelaMensaje("Error. "+ex.toString());

			}
		}
	}

	public void IntroduceDatos(int num,String marca, String modelo, double pma, double
mma, char tipo, double pesodelantero,
			int ejesdelantero, double pesotrasero, int ejestrasero, double
pesoremolque, int ejesremolque, double frentedelantero,
			double delanterocaja, double delanterotrasero, double
delanteroKingPin, double kingPinRemolque, double longitud) {
		//Preparamos la conexion
		Connection con;
```

```
		//Usamos el separador de archivos
		String proyecto = DatosGenericos.getDireccionDB("Vehiculos");
		File url = new File(proyecto);

		if(url.exists()) {
			try {
				//Usamos el driver embebido
				Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
				String db = "jdbc:derby:"+proyecto;
				con=DriverManager.getConnection(db);

				//Generamos el string que contendra el statement para SQL
				String tabla = "INSERT INTO Camiones VALUES
("+num+",'"+marca+"','"+modelo+"',"+

pma+","+mma+",'"+tipo+"',"+pesodelantero+","+ejesdelantero+","+pesotrasero+","+ejestraser
o+","+

pesoremolque+","+ejesremolque+","+frentedelantero+","+delanterocaja+","+delanterotrasero+
","+

delanteroKingPin+","+kingPinRemolque+","+longitud+")";
				//Generamos el statement
				PreparedStatement ps = con.prepareStatement(tabla);
				//Ejecutamos y cerramos el statement
				ps.execute();
				ps.close();
			}catch (Exception ex) {
				CheckDatos.okCancelaMensaje("No se pudieron introducir los
datos. "+ex.toString());
			}
		}
	}

	public void EliminaDatos(int num) {
		//Preparamos la conexion
		Connection con;
		//Usamos el separador de archivos
		String proyecto = DatosGenericos.getDireccionDB("Vehiculos");
		File url = new File(proyecto);

		if(url.exists()) {
			try {
				//Usamos el driver embebido
				Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
				//Generamos la direccion del proyecto
				String db = "jdbc:derby:"+proyecto;
				//generamos la conexion
				con=DriverManager.getConnection(db);

				//Preparamos el string que contendrá el statement
				String tabla = "DELETE FROM Camiones WHERE ID ="+num;
				//Generamos el statement
				PreparedStatement ps = con.prepareStatement(tabla);
				//Ejecutamos el statement y cerramos
				ps.execute();
				ps.close();
			}catch (Exception ex) {
				CheckDatos.okCancelaMensaje("No se pudieron borrar los datos.
"+ex.toString());
			}
		}
	}

	public void reemplazaDatosDouble(String campo, double valor, int ID) {

			//Creamos la conexion
```

```
                    Connection con;
                    //Usamos el separador de archivos
                    String proyecto = DatosGenericos.getDireccionDB("Vehiculos");
                    File url = new File(proyecto);


                    if(url.exists()) {
                          try {
                                //usamos el driver de la base de datos embebida
                                Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
                                //Generamos la direccion de la base de datos
                                String db = "jdbc:derby:"+proyecto;
                                //Generamos la conexion
                                con=DriverManager.getConnection(db);

                                //Generamos el string para el statement
                                String tabla = "UPDATE Camiones SET "+campo+"="+valor+"
WHERE Id="+ID;

                                //Generamos el statement
                                PreparedStatement ps = con.prepareStatement(tabla);
                                //Lo ejecutamos y cerramos
                                ps.execute();
                                ps.close();

                          }catch (Exception ex) {
                                CheckDatos.okCancelaMensaje("No se pudieron modificar
los datos. "+ex.toString());
                          }
                    }
      }

      public void reemplazaDatosInteger(String campo, int valor, int ID) {

            //Creamos la conexion
            Connection con;
            //Usamos el separador de archivos
            String proyecto = DatosGenericos.getDireccionDB("Vehiculos");
            File url = new File(proyecto);


            if(url.exists()) {
                  try {
                        //usamos el driver de la base de datos embebida
                        Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
                        //Generamos la direccion de la base de datos
                        String db = "jdbc:derby:"+proyecto;
                        //Generamos la conexion
                        con=DriverManager.getConnection(db);

                        //Generamos el string para el statement
                        String tabla = "UPDATE Camiones SET "+campo+"="+valor+" WHERE
Id="+ID;

                        //Generamos el statement
                        PreparedStatement ps = con.prepareStatement(tabla);
                        //Lo ejecutamos y cerramos
                        ps.execute();
                        ps.close();

                  }catch (Exception ex) {
                        CheckDatos.okCancelaMensaje("No se pudieron modificar los
datos. "+ex.toString());
                  }
            }
      }
```

```
	public void reemplazaDatosString(String campo, String valor, int ID) {

		//Creamos la conexion
		Connection con;
		//Usamos el separador de archivos
		String proyecto = DatosGenericos.getDireccionDB("Vehiculos");
		File url = new File(proyecto);


		if(url.exists()) {
			try {
				//usamos el driver de la base de datos embebida
				Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
				//Generamos la direccion de la base de datos
				String db = "jdbc:derby:"+proyecto;
				//Generamos la conexion
				con=DriverManager.getConnection(db);

				//Generamos el string para el statement
				String tabla = "UPDATE Camiones SET "+campo+"='"+valor+"'
WHERE Id="+ID;

				//Generamos el statement
				PreparedStatement ps = con.prepareStatement(tabla);
				//Lo ejecutamos y cerramos
				ps.execute();
				ps.close();

			}catch (Exception ex) {
				CheckDatos.okCancelaMensaje("No se pudieron modificar los
datos. "+ex.toString());
			}
		}
	}

	public void RecogeDatos() {
		//Preparamos la conexion
		Connection con;
		//Usamos el separador de archivos
		String proyecto = DatosGenericos.getDireccionDB("Vehiculos");
		File url = new File(proyecto);

		if(url.exists()) {
			try {
				//Usamos el driver de apache derby embebido
				Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
				//Creamos la direccion del proyecto
				String db = "jdbc:derby:"+proyecto;
				//Generamos la conexion
				con=DriverManager.getConnection(db);
				//Generamos el statement para obtener los datos en un ResulSet
				Statement ps=con.createStatement();
				java.sql.ResultSet lista = ps.executeQuery("SELECT * FROM
Camiones ORDER BY Id");
				//Recogemos los datos y los metemos en un ArrayList para
manejarlos con el resto de clases.
				while(lista.next()) {
					al=new ArrayList<Object>();
					for(int i=1;i<=18;i++) {
						al.add(lista.getObject(i));
					}
					camiones.add(al);
				}
				ps.close();
			}catch (Exception ex) {
				System.out.println("Error. "+ex);
			}
```

```
                }

    }

        public ArrayList<ArrayList<Object>> datos(){
                return camiones;
        }

}
```

CLASE BaseDatosSecciones

```
package basesDatos;

import java.io.File;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.Statement;
import java.util.ArrayList;

import auxiliares.CheckDatos;
import auxiliares.DatosGenericos;
import camiones.*;


public class BaseDatosSecciones{
	public ArrayList<ArrayList<Object>> secciones = new
ArrayList<ArrayList<Object>>();
	public ArrayList<Object> al = new ArrayList<Object>();

	public void CrearDB() {

		//Creamos la conexion
		Connection con;
		String proyecto = DatosGenericos.getDireccionDB("Secciones");
		File url = new File(proyecto);

		//Si la base de datos no existe la creamos
		if(!url.exists()) {
			try {
				Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
				String db = "jdbc:derby:"+proyecto+";create=true";
				con=DriverManager.getConnection(db);

				String tabla = "create table Vigas( Id INT PRIMARY KEY, ESPINF
DOUBLE,"
					+ "LONGINF DOUBLE, ESPMED DOUBLE, LONGMED DOUBLE,
ESPSUP DOUBLE, LONGSUP DOUBLE,"
					+ "AREA DOUBLE, INERTIA DOUBLE, MODULUS DOUBLE)";
				PreparedStatement ps = con.prepareStatement(tabla);
				ps.execute();
				ps.close();

				CheckDatos.okCancelaMensaje("Base de datos creada");

			}catch (Exception ex) {
				System.out.println("Error: "+ex);
			}
		}
	}

	public void IntroduceDatos(int num,double espinf, double longinf, double espmed,
double longmed,
			double espsup, double longsup, double area, double inertia, double
modulus) {

		//Creamos la conexion
		Connection con;
		//Usamos el separador de archicos
		String proyecto = DatosGenericos.getDireccionDB("Secciones");
		File url = new File(proyecto);

		if(url.exists()) {
			try {
				//Usamos el Driver de Derby Embebido
				Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
```

```
                        //La base de datos es la del proyecto
                        String db = "jdbc:derby:"+proyecto;
                        //Conectamos
                        con=DriverManager.getConnection(db);

                        //Insertamos el valor generando un string que contendrá el
statement preparado
                        String tabla = "INSERT INTO Vigas VALUES
("+num+","+espinf+","+longinf+","+

espmed+","+longmed+","+espsup+","+longsup+","+area+","+inertia+","+modulus+")";
                        //Generamos el Statement preparado
                        PreparedStatement ps = con.prepareStatement(tabla);
                        //Lo ejecutamos y lo cerramos
                        ps.execute();
                        ps.close();

                }catch (Exception ex) {
                        CheckDatos.okCancelaMensaje("No se pudieron introducir los
datos. "+ex.toString());
                }
        }
   }

      public void EliminaDatos(int num) {

            //Creamos la conexion
            Connection con;
            //Usamos el separador de archivos
            String proyecto = DatosGenericos.getDireccionDB("Secciones");
            File url = new File(proyecto);


            if(url.exists()) {
                  try {
                        //usamos el driver de la base de datos embebida
                        Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
                        //Generamos la direccion de la base de datos
                        String db = "jdbc:derby:"+proyecto;
                        //Generamos la conexion
                        con=DriverManager.getConnection(db);

                        //Generamos el string para el statement
                        String tabla = "DELETE FROM Vigas WHERE ID ="+num;

                        //Generamos el statement
                        PreparedStatement ps = con.prepareStatement(tabla);
                        //Lo ejecutamos y cerramos
                        ps.execute();
                        ps.close();

                  }catch (Exception ex) {
                        CheckDatos.okCancelaMensaje("No se pudieron borrar los datos.
"+ex.toString());
                  }
            }
      }

      public void reemplazaDatos(String campo, double valor, int ID) {

            //Creamos la conexion
            Connection con;
            //Usamos el separador de archivos
            String proyecto = DatosGenericos.getDireccionDB("Secciones");
            File url = new File(proyecto);
```

```
		if(url.exists()) {
			try {
				//usamos el driver de la base de datos embebida
				Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
				//Generamos la direccion de la base de datos
				String db = "jdbc:derby:"+proyecto;
				//Generamos la conexion
				con=DriverManager.getConnection(db);

				//Generamos el string para el statement
				String tabla = "UPDATE Vigas SET "+campo+"="+valor+" WHERE
Id="+ID;

				//Generamos el statement
				PreparedStatement ps = con.prepareStatement(tabla);
				//Lo ejecutamos y cerramos
				ps.execute();
				ps.close();

			}catch (Exception ex) {
				CheckDatos.okCancelaMensaje("No se pudieron modificar los
datos. "+ex.toString());
			}
		}
	}

	public void RecogeDatos() {

		//Preparamos la conexion
		Connection con;
		//Usamos el separador de archivos
		String proyecto = DatosGenericos.getDireccionDB("Secciones");
		File url = new File(proyecto);

		if(url.exists()) {
			try {
				//Usamos el driver embebido
				Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
				//Generamos la direccion de la base de datos
				String db = "jdbc:derby:"+proyecto;
				//generamos la conexion
				con=DriverManager.getConnection(db);
				//Generamos un statement
				Statement ps=con.createStatement();
				//Obtenemos un ResultSet con los datos
				java.sql.ResultSet lista = ps.executeQuery("SELECT * FROM
Vigas ORDER BY Id");
				//Metemos todos los datos del ResultSet en un ArrayList
				while(lista.next()) {
					al=new ArrayList<Object>();
					for(int i=1;i<=10;i++) {
						al.add(lista.getObject(i));
					}
					secciones.add(al);
				}
				//Cerramos la conexion
				ps.close();
				con.close();
			}catch (Exception ex) {
				System.out.println("Error: "+ex);
			}
		}
   }

	public ArrayList<ArrayList<Object>> datos(){
		return secciones;
	}
```

```
}
```

CLASE VentanaBaseDatosCamion

```
package basesDatos;

import java.awt.*;
import javax.swing.*;
import javax.swing.border.LineBorder;
import javax.swing.table.DefaultTableModel;

import auxiliares.CheckDatos;
import auxiliares.DatosGenericos;
import auxiliares.Formato;
import auxiliares.PanelConImagen;
import camiones.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Iterator;

import static java.awt.GridBagConstraints.*;


@SuppressWarnings("unused")
public class VentanaBaseDatosCamion extends JFrame{

	private static final long serialVersionUID = -7418029428404409927L;
	//Definimos los valores comunes para todos los componentes y elementos
	private static int ladopanel=300, filaSeleccionada, columnaSeleccionada;
	private static String fondogris=DatosGenericos.getDireccion("Imagenes/fondos/
fondo-gris.png");
	private static String fondoazul=DatosGenericos.getDireccion("Imagenes/fondos/
fondo.jpg");
	private Image fondo=new ImageIcon(getClass().getResource(fondogris)).getImage();
	private static ArrayList<Component> c = new ArrayList<Component>();
	private PanelConImagen general,panelMarca,datosPanel; //Generamos los paneles
contenedores de los datos, el dibujo y la info
	private static int dimension=800; //Generamos una dimension de ventana para
cambiarla si nos interesa
	private GridBagConstraints constraints; //Generamos las restriccioones del
GridBagLayout
	private static Proyecto proyecto;
	private static JTable tablaCamion;
	private static DefaultTableModel modeloCamion;
	private static JScrollPane scrollPaneCamion;
	private static JTextField camionText;
	private static JButton importaCamion, borraCamion, modificaCamion;
	private static VistaGeneral ventGeneral;
	private static BaseDatosCamiones b;


	public VentanaBaseDatosCamion(Proyecto p,VistaGeneral vg) {
		super("Base de datos de camiones");
		this.setResizable(false);
		proyecto=p;
		ventGeneral=vg;


		//Preparamos los PANELES
		//datos panel contiene los elementos de entrada de datos
		datosPanel=new PanelConImagen(fondogris);
		//general contiene los datos de interaccion
		general=new PanelConImagen(fondoazul);
```

```
            panelMarca=new PanelConImagen(DatosGenericos.getDireccion("Imagenes/
pantallaprincipal/recurso.png")); //contiene el logo de la aplicación
            panelMarca.setPreferredSize(new Dimension(150,80));


            //Preparamos los BOTONES
            //Generamos los botones
            importaCamion = new JButton("Importa camión");
            borraCamion = new JButton("Borra camión");
            modificaCamion = new JButton("Modifica camión");
            importaCamion.setPreferredSize(new Dimension(200,30));
            borraCamion.setPreferredSize(new Dimension(200,30));
            modificaCamion.setPreferredSize(new Dimension(200,30));
            Formato.setFormatoEstandar(importaCamion);
            Formato.setFormatoEstandar(borraCamion);
            Formato.setFormatoEstandar(modificaCamion);


            //Preparamos los JTEXTFIELD
            camionText = new JTextField();
            camionText.setPreferredSize(new Dimension(200,30));
            Formato.setFormatoEstandar(camionText);

            //Generamos el titulo de la ventana
            JLabel tituloBBDDCamiones = new JLabel("BBDD de camiones");
            //Damos formato a los tipos de aplicacion
            Formato.setFuenteAplicacion(tituloBBDDCamiones);


            //Generamos el JTABLE DE LAS REACCIONES
            //Creamos el modelo de la tabla
         modeloCamion = new DefaultTableModel();
         //Le damos los nombres de las columnas
         modeloCamion.addColumn("Ref. Num.");
         modeloCamion.addColumn("Marca.");
         modeloCamion.addColumn("Modelo.");
         modeloCamion.addColumn("MMA.");
         modeloCamion.addColumn("TARA.");
         modeloCamion.addColumn("Tipo.");
         modeloCamion.addColumn("Peso Delantero.");
         modeloCamion.addColumn("Ejes Delantero.");
         modeloCamion.addColumn("Peso Trasero.");
         modeloCamion.addColumn("Ejes Trasero.");
         modeloCamion.addColumn("Peso Remolque.");
         modeloCamion.addColumn("Ejes Remolque.");
         modeloCamion.addColumn("Frente-Delantero.");
         modeloCamion.addColumn("Delantero-Caja.");
         modeloCamion.addColumn("Delantero-Trasero.");
         modeloCamion.addColumn("Delantero-KingPin.");
         modeloCamion.addColumn("KingPin-Remolque.");
         modeloCamion.addColumn("Longitud.");
         //se crea la Tabla
         tablaCamion = new JTable() {
            private static final long serialVersionUID = 1L;
                  @Override
            public boolean isCellEditable(int row, int column) {
                  return false;
            }
         };
         //Le asignamos el modelo
         tablaCamion.setModel(modeloCamion);
         //Le damos las dimensiones a la tabla
        tablaCamion.setPreferredScrollableViewportSize(new Dimension((3*ladopanel-20),
(ladopanel-60)));
        tablaCamion.setEnabled(true);
        tablaCamion.setAutoResizeMode(JTable.AUTO_RESIZE_OFF);
        tablaCamion.setAutoscrolls(true);
```

```
        //Creamos un Jscrollane y le agregamos la JTable creada
        scrollPaneCamion = new JScrollPane(tablaCamion);

scrollPaneCamion.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
        //Metemos en el panel de cargas el scrollpane que contiene la tabla
        add(scrollPaneCamion);
        //PERSONALIZAMOS EL FORMATO DE LA TABLA
        Formato.setFormatoContenidoTabla(tablaCamion);


        //LAS SIGUIENTES LINEAS SOLO SIRVEN PARA METER VIGAS Y HACER PRUEBAS
        try {
        b = new BaseDatosCamiones();
        b.CrearDB();

        /*

b.IntroduceDatos(1,"Iveco","DEX",20000,19000,'R',7500,1,19500,2,20000,2,400,600,5000,0,0,
6000);

b.IntroduceDatos(2,"Iveco","DEX2",20000,19000,'R',7500,1,19500,2,20000,2,400,600,5000,0,0
,6000);

b.IntroduceDatos(3,"Iveco","DEX",20000,19000,'R',7500,1,19500,2,20000,2,400,600,5000,0,0,
6000);

b.IntroduceDatos(4,"Scania","Stratos",20000,19000,'A',7500,1,19500,2,20000,2,400,600,2500
,900,8000,12000);
        */


        b.RecogeDatos();
        ArrayList<ArrayList<Object>> camiones = b.datos();
              Iterator<ArrayList<Object>> iteradorCamiones = camiones.iterator();
              while (iteradorCamiones.hasNext()) {
                     ArrayList<Object> camion = iteradorCamiones.next();
                     Iterator<Object> itcamion = camion.iterator();
                     //18 datos por camion asi que creo un vector de 18 valores
                     Object[] fila={0,"","",0,0,'R',0,0,0,0,0,0,0,0,0,0,0,0};
                     int contador=0;
                     while(itcamion.hasNext()) {
                            Object o = itcamion.next();
                            fila[contador] = o;
                            contador++;
                     }
                     modeloCamion.addRow(fila);
                     System.out.println(fila);
              }
        } catch (Exception e) {
              CheckDatos.okCancelaMensaje(e.toString());
        }


        //Metemos la tabla en el panel de Datos
        datosPanel.add(scrollPaneCamion);
              datosPanel.setPreferredSize(new Dimension(3*ladopanel-10,(ladopanel-10)));
              Formato.setBorder(datosPanel);


              //Damos formato al PANEL GENERAL
              SpringLayout layout = new SpringLayout();
              //Damos formato
              Formato.setBorder(general);
              general.setPreferredSize(new Dimension(3*ladopanel,ladopanel*2));
       general.setLayout(layout);
        int gapX=20;
        int gapY=20;
```

```
        //Marcamos las RESTRICCIONES DE COLOCACION SPRINGLAYOUT EN EL PANEL
       //Incluimos los elementos en los paneles las restricciones en X
       layout.putConstraint(SpringLayout.WEST, panelMarca, (gapX*2), SpringLayout.WEST,
general);
       layout.putConstraint(SpringLayout.WEST, tituloBBDDCamiones,
(gapX*16),SpringLayout.WEST,general);
       layout.putConstraint(SpringLayout.WEST,
datosPanel,gapX,SpringLayout.WEST,general);
       layout.putConstraint(SpringLayout.WEST,
importaCamion,gapX,SpringLayout.WEST,general);
       layout.putConstraint(SpringLayout.WEST, borraCamion,15,SpringLayout.WEST,general);

       layout.putConstraint(SpringLayout.WEST,
modificaCamion,15,SpringLayout.WEST,general);
       layout.putConstraint(SpringLayout.WEST,
camionText,15,SpringLayout.EAST,modificaCamion);
       //Incluimos los elementos en los paneles las restricciones en Y
       layout.putConstraint(SpringLayout.NORTH, panelMarca, gapY, SpringLayout.NORTH,
general);
       layout.putConstraint(SpringLayout.NORTH,
tituloBBDDCamiones,gapY*2,SpringLayout.NORTH,general);
       layout.putConstraint(SpringLayout.NORTH, datosPanel, gapY, SpringLayout.SOUTH,
panelMarca);
       layout.putConstraint(SpringLayout.NORTH, importaCamion, gapY, SpringLayout.SOUTH,
datosPanel);
       layout.putConstraint(SpringLayout.NORTH, borraCamion, gapX, SpringLayout.SOUTH,
importaCamion);
       layout.putConstraint(SpringLayout.NORTH, modificaCamion, gapX, SpringLayout.SOUTH,
borraCamion);
       layout.putConstraint(SpringLayout.NORTH, camionText, gapX, SpringLayout.SOUTH,
borraCamion);


       //Generamos los ActionListeners
        ActionListener accionbtnInmporta=new ActionListener() {
             public void actionPerformed(ActionEvent e) {
                    importaDatos();
                    }
       };
       //Generamos los ActionListeners
        ActionListener accionbtnBorra=new ActionListener() {
             public void actionPerformed(ActionEvent e) {
                    borraDatos();
                    }
       };
       //Generamos los ActionListeners
        ActionListener accionbtnModifica=new ActionListener() {
             public void actionPerformed(ActionEvent e) {
                    modificaDatos();
                    }
       };

       //Para la tabla
       MouseListener tablaListener=new MouseAdapter() {
                public void mouseClicked(MouseEvent e) {
                     //Guardamos para otros metodos cual es la celda seleccionada.
                    filaSeleccionada = tablaCamion.getSelectedRow();
                    columnaSeleccionada = tablaCamion.getSelectedColumn();
                    if(columnaSeleccionada!=0) {
                           String valor = modeloCamion.getValueAt(filaSeleccionada,
columnaSeleccionada).toString();
                           camionText.setText(valor);
                    } else {
                      CheckDatos.okCancelaMensaje("El ID no puede cambiarse o
modificarse");
                    }
```

```
                    //Marcamos la fila como seleccionada
                    Formato.setFormatoContenidoTablaSeleccionado(tablaCamion,
filaSeleccionada);
                    actualiza();
                }
        };


        importaCamion.addActionListener(accionbtnInmporta);
        borraCamion.addActionListener(accionbtnBorra);
        modificaCamion.addActionListener(accionbtnModifica);
        tablaCamion.addMouseListener(tablaListener);

        //Metemos el Logo y la tabla de las vigas en la lista
        general.add(panelMarca);
        general.add(tituloBBDDCamiones);
        general.add(datosPanel);
        general.add(importaCamion);
        general.add(borraCamion);
        general.add(modificaCamion);
        general.add(camionText);
        add(general);

        //Damos las medidas y localizacion de la ventana
               setPreferredSize(new Dimension(ladopanel*3+50,ladopanel*2+50));
               pack();
               this.setLocation(new Point(100,100));
               DatosGenericos.setPosicion(this, ladopanel*3+50, ladopanel*2+50);
               setVisible(true);
        }

    //Este método asigna a todos los componentes de la ventana el mismo tipo de texto
    private void darTipoLetra(ArrayList<Component> c) {
         Iterator<Component> iter = c.iterator();
         while(iter.hasNext()){
               ponLetra(iter.next());
         }
    }


    //Este método da el tipo letra a cada componente igual
    private void ponLetra(Component c) {
       Formato.setFormatoEstandar(c);
    }

    //Este metodo actualizará la tabla cuando se metan nuevos valores
    public void actualiza() {
       tablaCamion.repaint();
       repaint();
    }

    //Este metodo importa los datos de la fila elegida de la tabla de base de datos a la
ventana VentanaSeccion
    private void importaDatos() {
       //Tiene que ver qué fila está seleccionada y meterla en la ventanaSeccion.
       int fila = (int)tablaCamion.getSelectedRow();
       String[] valores=
{"0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0","0"};
        if(fila!=-1) {
               for(int columna=0;columna<(tablaCamion.getColumnCount());columna++) {
                      //Ajustamos los anchos para poder ver mejor la información

tablaCamion.getColumnModel().getColumn(columna).setPreferredWidth(100);
                      String valor = modeloCamion.getValueAt(fila,columna).toString();
                      valores[columna]=valor;
               }
               //Metemos los valores en VistaGeneral
```

```
ventGeneral.extraeDeTabla(valores[1],valores[2],valores[3],valores[4],valores[5],valores[
6],valores[7],valores[8],valores[9],

valores[10],valores[11],valores[12],valores[13],valores[14],valores[15],valores[16],valor
es[17]);
       }
       ventGeneral.asignaDatosAProyecto();
              this.setVisible(false);
    }

    //Este metodo exporta los datos que llegan de VentanaSeccion a la base de datos
    public void incluyeNuevosDatos(String marca, String modelo, double PMA, double MMA,
String Tipo, double PesoD, double EjesD,
              double PesoT, double ejesT, double pesoR, double ejesR, double distFD,
double distDC, double distDT,
              double distDK, double distKR, double longitud) {
       //Los datos son:
       /*
        * PMA=Peso Maximo Autorizado
        * MMA=Masa Maxima Autorizada
        * Tipo=Articulado o Rigido
        * PesoD=Peso máximo tandem/eje delantero
        * EjesD=Ejes en el tandem delantero
        * PesoT=Peso máximo tandem/eje trasero
        * EjesT=Ejes en el tandem trasero
        * PesoR=Peso máximo tandem/eje remolque
        * EjesR=Ejes en el tandemo remolque
        * DistFD=Distancia del frente al eje delantero
        * DistDC=Distancia del eje delantero a la caja
        * DistDT=Distancia del eje delantero al trasero
        * DistDK=Distancia del eje delantero al KingPin
        * DistKR=Distancia del KingPin al Remolque
        * Longitud=Longitud del camion
        */

       //Cogemos el modelo de tabla
       DecimalFormat df = new DecimalFormat("0.00");
       DecimalFormat df2 = new DecimalFormat("0");
        DefaultTableModel modelotabla = (DefaultTableModel) tablaCamion.getModel();

        //Miramos el numero de filas que tenemos
        int filas = tablaCamion.getRowCount();

        //Para exportar asignamos inicialmente un ID que sera el máximo de las filas
        int identificadorParaFila = filas+1;

        //miramos si queda algun numero de ID que se ha ido borrando y se puede asignar
entre el 1 y
        //el numero de filas para darselo y no tener numeros descontrolados
        int i=1;
        //El booleano asignado me dira si ya ha sido asignado y parara el bucle
        boolean asignado=false;

        //Paso por todos los posibles valores de ID
        while((i<=filas)&&(!asignado)) {

              //El booleano existeElNumero me dira si se ha encontrado el numero y dejara
de seguir buscandolo.
              boolean existeElNumero=false;
              //Con cada valor de ID compruebo para i compruebo si lo tiene la fila j en
su ID desde la 0 a la última
              int j=0;
              while((j<=filas-1)&&(!existeElNumero)) {

              //Cogemos el numero en la casilla de ID de la fila
              int numero = (int)modeloCamion.getValueAt(j, 0);
```

```
            if (numero==i) {
                  existeElNumero=existeElNumero||true;
            }
            else {
                  existeElNumero=existeElNumero||false;
            }

            j++;

            }
            if(!existeElNumero) {
                  //En caso de que no este ese ID lo asignamos al identificador para
fila y paramos la buscqueda
                  asignado=true;
                  //Le asignamos i+1 porque el ID tiene un valor de una unidad más que
la del indice usado
                  identificadorParaFila=i;
            }
            else i++;
        }
        CheckDatos.okCancelaMensaje("Identificador a introducir:"+identificadorParaFila);
            Object[] datosnuevo=
{identificadorParaFila,marca,modelo,df.format(PMA),df.format(MMA),Tipo,df.format(PesoD),

df2.format(EjesD),df.format(PesoT),df2.format(ejesT),df.format(pesoR),df2.format(ejesR),d
f.format(distFD),df.format(distDC),df.format(distDT),

df.format(distDK),df.format(distKR),df.format(longitud)};

b.IntroduceDatos(identificadorParaFila,marca,modelo,PMA,MMA,Tipo.charAt(0),PesoD,
                        (int)EjesD,PesoT,(int)ejesT,pesoR,
(int)ejesR,distFD,distDC,distDT,
                                          distDK,distKR,longitud);
            modeloCamion.addRow(datosnuevo);
            actualiza();
    }


    //Este método borra datos de la tabla y la base de datos
    public void borraDatos() {
       try {
            int fila = tablaCamion.getSelectedRow();
            int columna=tablaCamion.getSelectedColumn();
            int numero=(int)modeloCamion.getValueAt(fila, 0);
            modeloCamion.removeRow(fila);
            b.EliminaDatos(numero);
            actualiza();
       } catch (Exception e) {
            CheckDatos.okCancelaMensaje("No se pudo eliminar. "+e.toString());
       }
    }

  //Este método modifica datos especificos de la tabla y la base de datos
    public void modificaDatos() {
       try {
            if(camionText.getText().isEmpty()) {
                  CheckDatos.okCancelaMensaje("Es necesario hacer click en el valor a
cambiar");
            } else {
                  //Cogemos el identificador de la fila a cambiar
                  int IdACambiar = (int)modeloCamion.getValueAt(filaSeleccionada, 0);

                  //Cogemos el valor de la casilla
                  String valorCambio = camionText.getText();
                  double valorCambioDouble=0;
                  int valorCambioInteger=0;
```

```
                    String valor="";

                    try {
                    //Cambiamos el valor en la tabla

                    //Elegimos la columna para ponerlo en el Statement del SQL en UPDATE
y
                    //pasamos el valor al tipo necesario Double para guardarlo en la base
de datos
                           switch(columnaSeleccionada){
                                  case 1:valor = "MARCA";
                                  b.reemplazaDatosString(valor, valorCambio, IdACambiar);
                                  modeloCamion.setValueAt(camionText.getText(),
filaSeleccionada, columnaSeleccionada);
                                  break;
                                  case 2:valor = "MODELO";
                                  b.reemplazaDatosString(valor, valorCambio, IdACambiar);
                                  modeloCamion.setValueAt(camionText.getText(),
filaSeleccionada, columnaSeleccionada);
                                  break;
                                  case 3:valor = "PMA";
                                  valorCambioDouble =
(double)Double.valueOf(valorCambio);
                                  b.reemplazaDatosDouble(valor, valorCambioDouble,
IdACambiar);
                                  modeloCamion.setValueAt(camionText.getText(),
filaSeleccionada, columnaSeleccionada);
                                  break;
                                  case 4:valor = "MMA";
                                  valorCambioDouble = Double.valueOf(valorCambio);
                                  b.reemplazaDatosDouble(valor, valorCambioDouble,
IdACambiar);
                                  modeloCamion.setValueAt(camionText.getText(),
filaSeleccionada, columnaSeleccionada);
                                  break;
                                  case 5:valor = "TIPO";
                                  if((valorCambio!="R")&&(valorCambio!="A"))
CheckDatos.okCancelaMensaje("El valor para esa variable debe ser 'R' o 'A'");
                                  else {
                                         b.reemplazaDatosString(valor, valorCambio,
IdACambiar);
                                         modeloCamion.setValueAt(camionText.getText(),
filaSeleccionada, columnaSeleccionada);
                                  }
                                  break;
                                  case 6:valor = "PESODELANTERO";
                                  valorCambioDouble = Double.valueOf(valorCambio);
                                  b.reemplazaDatosDouble(valor, valorCambioDouble,
IdACambiar);
                                  modeloCamion.setValueAt(camionText.getText(),
filaSeleccionada, columnaSeleccionada);
                                  break;
                                  case 7:valor = "EJESDELANTERO";
                                  valorCambioInteger = Integer.valueOf(valorCambio);
                                  b.reemplazaDatosDouble(valor, valorCambioInteger,
IdACambiar);
                                  modeloCamion.setValueAt(camionText.getText(),
filaSeleccionada, columnaSeleccionada);
                                  break;
                                  case 8:valor = "PESOTRASERO";
                                  valorCambioDouble = Double.valueOf(valorCambio);
                                  b.reemplazaDatosDouble(valor, valorCambioDouble,
IdACambiar);
                                  modeloCamion.setValueAt(camionText.getText(),
filaSeleccionada, columnaSeleccionada);
                                  break;
                                  case 9:valor = "EJESTRASERO";
```

```
                                    valorCambioInteger = Integer.valueOf(valorCambio);
                                    b.reemplazaDatosDouble(valor, valorCambioInteger,
IdACambiar);
                                    modeloCamion.setValueAt(camionText.getText(),
filaSeleccionada, columnaSeleccionada);
                                    break;
                                    case 10:valor = "PESOREMOLQUE";
                                    valorCambioDouble = Double.valueOf(valorCambio);
                                    b.reemplazaDatosDouble(valor, valorCambioDouble,
IdACambiar);
                                    modeloCamion.setValueAt(camionText.getText(),
filaSeleccionada, columnaSeleccionada);
                                    break;
                                    case 11:valor = "EJESREMOLQUE";
                                    valorCambioInteger = Integer.valueOf(valorCambio);
                                    b.reemplazaDatosDouble(valor, valorCambioInteger,
IdACambiar);
                                    modeloCamion.setValueAt(camionText.getText(),
filaSeleccionada, columnaSeleccionada);
                                    break;
                                    case 12:valor = "FRENTEDELANTERO";
                                    valorCambioDouble = Double.valueOf(valorCambio);
                                    b.reemplazaDatosDouble(valor, valorCambioDouble,
IdACambiar);
                                    modeloCamion.setValueAt(camionText.getText(),
filaSeleccionada, columnaSeleccionada);
                                    break;
                                    case 13:valor = "DELANTEROCAJA";
                                    valorCambioDouble = Double.valueOf(valorCambio);
                                    b.reemplazaDatosDouble(valor, valorCambioDouble,
IdACambiar);
                                    modeloCamion.setValueAt(camionText.getText(),
filaSeleccionada, columnaSeleccionada);
                                    break;
                                    case 14:valor = "DELANTEROTRASERO";
                                    valorCambioDouble = Double.valueOf(valorCambio);
                                    b.reemplazaDatosDouble(valor, valorCambioDouble,
IdACambiar);
                                    modeloCamion.setValueAt(camionText.getText(),
filaSeleccionada, columnaSeleccionada);
                                    break;
                                    case 15:valor = "DELANTKINGPIN";
                                    valorCambioDouble = Double.valueOf(valorCambio);
                                    b.reemplazaDatosDouble(valor, valorCambioDouble,
IdACambiar);
                                    modeloCamion.setValueAt(camionText.getText(),
filaSeleccionada, columnaSeleccionada);
                                    break;
                                    case 16:valor = "KINGPINREM";
                                    valorCambioDouble = Double.valueOf(valorCambio);
                                    b.reemplazaDatosDouble(valor, valorCambioDouble,
IdACambiar);
                                    modeloCamion.setValueAt(camionText.getText(),
filaSeleccionada, columnaSeleccionada);
                                    break;
                                    case 17:valor = "LONGITUD";
                                    valorCambioDouble = Double.valueOf(valorCambio);
                                    b.reemplazaDatosDouble(valor, valorCambioDouble,
IdACambiar);
                                    modeloCamion.setValueAt(camionText.getText(),
filaSeleccionada, columnaSeleccionada);
                                    break;
                                    }
                    } catch(Exception e) {
                            CheckDatos.okCancelaMensaje("No se ha podido cambiar el valor
propuesto.\nRevise las entradas");
                    }
```

```
                }
        } catch (Exception e) {
                CheckDatos.okCancelaMensaje("No se pudo modificar. "+e.toString());
        }
        actualiza();
    }

}
```

CLASE VentanaBaseDatosSeccion

```
package basesDatos;

import java.awt.*;
import javax.swing.*;
import javax.swing.border.LineBorder;
import javax.swing.table.DefaultTableModel;

import auxiliares.CheckDatos;
import auxiliares.DatosGenericos;
import auxiliares.Formato;
import auxiliares.PanelConImagen;
import camiones.*;
import secciones.VentanaSeccion;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.text.DecimalFormat;
import java.util.ArrayList;
import java.util.Iterator;

import static java.awt.GridBagConstraints.*;


@SuppressWarnings("unused")
public class VentanaBaseDatosSeccion extends JFrame{

	private static final long serialVersionUID = -7418029428404409927L;
	//Definimos los valores comunes para todos los componentes y elementos
	private static int ladopanel=300,filaSeleccionada, columnaSeleccionada;
	private static String fondogris=DatosGenericos.getDireccion("Imagenes/fondos/
fondo-gris.png");
	private static String fondoazul=DatosGenericos.getDireccion("Imagenes/fondos/
fondo.jpg");
	private Image fondo=new ImageIcon(getClass().getResource(fondogris)).getImage();
	private static ArrayList<Component> c = new ArrayList<Component>();
	private PanelConImagen general,panelMarca,datosPanel; //Generamos los paneles
contenedores de los datos, el dibujo y la info
	private static int dimension=800; //Generamos una dimension de ventana para
cambiarla si nos interesa
	private GridBagConstraints constraints; //Generamos las restriccioones del
GridBagLayout
	private static Proyecto proyecto;
	private static JTable tablaVigas;
	private static JTextField vigaText;
	private static DefaultTableModel modeloVigas;
	private static JScrollPane scrollPaneVigas;
	private static JButton importaViga,borraViga,modificaViga;
	private static VentanaSeccion ventSeccion;
	private static BaseDatosSecciones b;

	public VentanaBaseDatosSeccion(Proyecto p,VentanaSeccion vs) {
		super("Base de datos de secciones");
		this.setResizable(false);
		proyecto=p;
		ventSeccion=vs;


		//Preparamos los PANELES
		//datos panel contiene los elementos de entrada de datos
		datosPanel=new PanelConImagen(fondogris);
		//general contiene los datos de interaccion
		general=new PanelConImagen(fondoazul);
```

```
            panelMarca=new PanelConImagen(DatosGenericos.getDireccion("Imagenes/
pantallaprincipal/recurso.png")); //contiene el logo de la aplicación
            panelMarca.setPreferredSize(new Dimension(150,80));


            //Generamos el titulo de la ventana
            JLabel tituloBBDDSecciones = new JLabel("BBDD de secciones");
            //Damos formato a los tipos de aplicacion
            Formato.setFuenteAplicacion(tituloBBDDSecciones);


            //Preparamos los BOTONES
            //Generamos los botones
            importaViga = new JButton("Importa viga");
            importaViga.setPreferredSize(new Dimension(200,30));
            Formato.setFormatoEstandar(importaViga);
            borraViga = new JButton("Borra viga");
            borraViga.setPreferredSize(new Dimension(200,30));
            Formato.setFormatoEstandar(borraViga);
            modificaViga = new JButton("Modifica valor viga");
            modificaViga.setPreferredSize(new Dimension(200,30));
            Formato.setFormatoEstandar(modificaViga);

            //Preparamos el JTextField
            vigaText = new JTextField();
            Formato.setFormatoEstandar(vigaText);
            vigaText.setPreferredSize(new Dimension(200,30));

            //Generamos el JTABLE DE LAS REACCIONES
            //Creamos el modelo de la tabla
          modeloVigas = new DefaultTableModel();
          //Le damos los nombres de las columnas
          modeloVigas.addColumn("Ref. Num.");
          modeloVigas.addColumn("Espesor ala inferior(mm)");
          modeloVigas.addColumn("Longitud ala inferior(mm)");
          modeloVigas.addColumn("Espesor intermedio(mm)");
          modeloVigas.addColumn("Longitud alma(mm)");
          modeloVigas.addColumn("Espesor ala superior(mm)");
          modeloVigas.addColumn("Longitud ala superior(mm)");
          modeloVigas.addColumn("Area(mm2)");
          modeloVigas.addColumn("Iz(mm4)");
          modeloVigas.addColumn("Wz(mm3)");
          //se crea la Tabla

          tablaVigas = new JTable() {
          private static final long serialVersionUID = 1L;
            @Override
          public boolean isCellEditable(int row, int column) {
            return false;
          }
          };
          //Le asignamos el modelo
          tablaVigas.setModel(modeloVigas);
          //Le damos las dimensiones a la tabla
        tablaVigas.setPreferredScrollableViewportSize(new Dimension((3*ladopanel-20),
(ladopanel-50)));
        tablaVigas.setAutoResizeMode(JTable.AUTO_RESIZE_OFF);
        tablaVigas.setAutoscrolls(true);
        //Creamos un JscrollPane y le agregamos la JTable creada
        scrollPaneVigas = new JScrollPane(tablaVigas);

scrollPaneVigas.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
        //Metemos en el panel de cargas el scrollpane que contiene la tabla
        add(scrollPaneVigas);
        //tablaVigas.setEnabled(true);
        //PERSONALIZAMOS EL FORMATO DE LA TABLA
        Formato.setFormatoContenidoTabla(tablaVigas);
```

```
        //LAS SIGUIENTES LINEAS SOLO SIRVEN PARA METER VIGAS Y HACER PRUEBAS
        try {
        b = new BaseDatosSecciones();
        b.CrearDB();

        b.RecogeDatos();
        ArrayList<ArrayList<Object>> secciones = b.datos();
              Iterator<ArrayList<Object>> iteradorsecciones = secciones.iterator();
              while (iteradorsecciones.hasNext()) {
                     ArrayList<Object> seccion = iteradorsecciones.next();
                     Iterator<Object> itseccion = seccion.iterator();
                     Object[] fila={0,0,0,0,0,0,0,0,0,0};
                     int contador=0;
                     while(itseccion.hasNext()) {
                            Object o = itseccion.next();
                            fila[contador] = o;
                            contador++;
                     }
                     modeloVigas.addRow(fila);
                     System.out.println(fila);
              }
        } catch (Exception e) {
              CheckDatos.okCancelaMensaje(e.toString());
        }


        //Metemos la tabla en el panel de Datos
        datosPanel.add(scrollPaneVigas);
              datosPanel.setPreferredSize(new Dimension(3*ladopanel-10,(ladopanel-10)));
              Formato.setBorder(datosPanel);


              //Damos formato al PANEL GENERAL
              SpringLayout layout = new SpringLayout();
              //Damos formato
              Formato.setBorder(general);
              general.setPreferredSize(new Dimension(3*ladopanel,ladopanel*2));
       general.setLayout(layout);
        int gapX=20;
        int gapY=20;

        //Marcamos las RESTRICCIONES DE COLOCACION SPRINGLAYOUT EN EL PANEL
       //Incluimos los elementos en los paneles las restricciones en X
       layout.putConstraint(SpringLayout.WEST, panelMarca, (gapX*2), SpringLayout.WEST,
general);
       layout.putConstraint(SpringLayout.WEST, tituloBBDDSecciones, (gapY*16),
SpringLayout.WEST, general);
       layout.putConstraint(SpringLayout.WEST, datosPanel, gapY, SpringLayout.WEST,
general);
       layout.putConstraint(SpringLayout.WEST,
importaViga,gapX,SpringLayout.WEST,general);
       layout.putConstraint(SpringLayout.WEST, borraViga,gapX,SpringLayout.WEST,general);
       layout.putConstraint(SpringLayout.WEST,
modificaViga,gapX,SpringLayout.WEST,general);
       layout.putConstraint(SpringLayout.WEST,
vigaText,gapX,SpringLayout.EAST,borraViga);
       //Incluimos los elementos en los paneles las restricciones en Y
       layout.putConstraint(SpringLayout.NORTH, panelMarca, gapY, SpringLayout.NORTH,
general);
       layout.putConstraint(SpringLayout.NORTH, tituloBBDDSecciones, gapY*2,
SpringLayout.NORTH, general);
       layout.putConstraint(SpringLayout.NORTH, datosPanel, gapY, SpringLayout.SOUTH,
panelMarca);
       layout.putConstraint(SpringLayout.NORTH, importaViga, gapY, SpringLayout.SOUTH,
datosPanel);
```

```
      layout.putConstraint(SpringLayout.NORTH, borraViga, 10, SpringLayout.SOUTH,
importaViga);
      layout.putConstraint(SpringLayout.NORTH, modificaViga, 10, SpringLayout.SOUTH,
borraViga);
      layout.putConstraint(SpringLayout.NORTH, vigaText, 10, SpringLayout.SOUTH,
borraViga);


      //Generamos los ActionListeners
       ActionListener accionbtnImporta=new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                  importaDatos();
                  }
      };
      ActionListener accionbtnBorra=new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                  borraDatos();
                  }
      };
      ActionListener accionbtnModifica=new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                  modificaDatos();
                  }
      };

      //Para la tabla
      MouseListener tablaListener=new MouseAdapter() {
               public void mouseClicked(MouseEvent e) {
                   //Guardamos para otros metodos cual es la celda seleccionada.
                   filaSeleccionada = tablaVigas.getSelectedRow();
                   columnaSeleccionada = tablaVigas.getSelectedColumn();
                   if(columnaSeleccionada!=0) {
                         String valor = modeloVigas.getValueAt(filaSeleccionada,
columnaSeleccionada).toString();
                         vigaText.setText(valor);
                   } else {
                      CheckDatos.okCancelaMensaje("El ID no puede cambiarse o
modificarse");
                   }
                   //Marcamos la fila como seleccionada
                   Formato.setFormatoContenidoTablaSeleccionado(tablaVigas,
filaSeleccionada);
                   actualiza();
               }
      };


      importaViga.addActionListener(accionbtnImporta);
      borraViga.addActionListener(accionbtnBorra);
      modificaViga.addActionListener(accionbtnModifica);
      tablaVigas.addMouseListener(tablaListener);

      //Metemos el Logo y la tabla de las vigas en la lista
      general.add(panelMarca);
      general.add(tituloBBDDSecciones);
      general.add(datosPanel);
      general.add(importaViga);
      general.add(borraViga);
      general.add(modificaViga);
      general.add(vigaText);
      add(general);

      //Damos las medidas y localizacion de la ventana
            setPreferredSize(new Dimension(ladopanel*3+50,ladopanel*2+50));
            pack();
            this.setLocation(new Point(100,100));
            DatosGenericos.setPosicion(this, ladopanel*3+50, ladopanel*2+50);
```

```
            setVisible(true);

       }

    //Este método asigna a todos los componentes de la ventana el mismo tipo de texto
    private void darTipoLetra(ArrayList<Component> c) {
        Iterator<Component> iter = c.iterator();
        while(iter.hasNext()){
             ponLetra(iter.next());
        }
    }


    //Este método da el tipo letra a cada componente igual
    private void ponLetra(Component c) {
       Formato.setFormatoEstandar(c);
    }

    //Este metodo actualizará la tabla cuando se metan nuevos valores
    public void actualiza() {
       tablaVigas.repaint();
       repaint();
    }

    //Este metodo importa los datos de la fila elegida de la tabla de base de datos a la
ventana VentanaSeccion
    private void importaDatos() {
       //Tiene que ver qué fila está seleccionada y meterla en la ventanaSeccion.
       int fila = (int)tablaVigas.getSelectedRow();
       String[] valores= {"0","0","0","0","0","0","0","0","0","0"};
       if(fila!=-1) {
              for(int columna=0;columna<(tablaVigas.getColumnCount());columna++) {
                     String valor = modeloVigas.getValueAt(fila,columna).toString();
                     valores[columna]=valor;
              }

ventSeccion.extraeDeTabla(valores[1],valores[2],valores[3],valores[4],valores[5],valores[
6]);
       }
              this.setVisible(false);
    }

    //Este metodo exporta los datos que llegan de VentanaSeccion a la base de datos
    public void incluyeNuevosDatos(double e1, double l1, double e2, double l2, double e3,
double l3, double area, double inercia, double modulo) {

        //Miramos el numero de filas que tenemos
        int filas = tablaVigas.getRowCount();

        //Para exportar asignamos inicialmente un ID que sera el máximo de las filas
        int identificadorParaFila = filas+1;
        //miramos si queda algun numero de ID que se ha ido borrando y se puede asignar
entre el 1 y
        //el numero de filas para darselo y no tener numeros descontrolados
        int i=1;
        //El booleano asignado me dira si ya ha sido asignado y parara el bucle
        boolean asignado=false;

        //Paso por todos los posibles valores de ID
        while((i<=filas)&&(!asignado)) {

              //El booleano existeElNumero me dira si se ha encontrado el numero y dejara
de seguir buscandolo.
              boolean existeElNumero=false;
              //Con cada valor de ID compruebo para i compruebo si lo tiene la fila j en
su ID desde la 0 a la última
              int j=0;
```

```
            while((j<=filas-1)&&(!existeElNumero)) {

            //Cogemos el numero en la casilla de ID de la fila
            int numero = (int)modeloVigas.getValueAt(j, 0);
            if (numero==i) {
                  existeElNumero=existeElNumero||true;
            }
            else {
                  existeElNumero=existeElNumero||false;
            }

            j++;

            }
            if(!existeElNumero) {
                  //En caso de que no este ese ID lo asignamos al identificador para
fila y paramos la buscqueda
                  asignado=true;
                  //Le asignamos i+1 porque el ID tiene un valor de una unidad más que
la del indice usado
                  identificadorParaFila=i;
            }
            else i++;
        }

            //Cogemos el modelo de la tabla
        DefaultTableModel modelotabla = (DefaultTableModel) tablaVigas.getModel();
        //Generamos un objeto nuevo para introducir
            Object[] datosnuevo=
{identificadorParaFila,e1,l1,e2,l2,e3,l3,area,inercia,modulo};
            //Metemos los datos como nuevo registro de la base de datos

b.IntroduceDatos(identificadorParaFila,e1,l1,e2,l2,e3,l3,area,inercia,modulo);
            //Metemos los datos en la tabla mostrada.
            modeloVigas.addRow(datosnuevo);

            repaint();
    }


    //Este método borra datos de la tabla y la base de datos
    public void borraDatos() {
       try {
            int fila = tablaVigas.getSelectedRow();
            int columna=tablaVigas.getSelectedColumn();
            int numero=(int)modeloVigas.getValueAt(fila, 0);
            modeloVigas.removeRow(fila);
            b.EliminaDatos(numero);
            actualiza();
       } catch (Exception e) {
            CheckDatos.okCancelaMensaje("No se pudo eliminar. "+e.toString());
       }
    }


  //Este método borra datos de la tabla y la base de datos
    public void modificaDatos() {
       try {
            if(vigaText.getText().isEmpty()) {
                  CheckDatos.okCancelaMensaje("Es necesario hacer click en el valor a
cambiar");
            }
            String valor="";
            //Elegimos la columna para ponerlo en el Statement del SQL en UPDATE
            switch(columnaSeleccionada){
                  case 1:valor = "ESPINF";
                  break;
```

```
                        case 2:valor = "LONGINF";
                        break;
                        case 3:valor = "ESPMED";
                        break;
                        case 4:valor = "LONGMED";
                        break;
                        case 5:valor = "ESPSUP";
                        break;
                        case 6:valor = "LONGSUP";
                        break;
                        case 7:valor = "AREA";
                        break;
                        case 8:valor = "INERTIA";
                        break;
                        case 9:valor = "MODULUS";
                        break;
                }

                //Pasamos el valor a Double para guardarlo en la base de datos
                try {
                //Intentamos convertir el valor en double
                Double valorMod = Double.valueOf(vigaText.getText());
                //Lo guardamos en la tabla de la ventana
                modeloVigas.setValueAt(vigaText.getText(), filaSeleccionada,
columnaSeleccionada);

                //Generamos el UPDATE
                int IdACambiar = (int)modeloVigas.getValueAt(filaSeleccionada, 0);
                b.reemplazaDatos(valor, valorMod, IdACambiar);

                //OJO FALTA ACTUALIZAR EL AREA Y LA INERCIA SI LO CONSIDERAMOS NECESARIO

                } catch(Exception e) {
                        CheckDatos.okCancelaMensaje("No se ha podido cambiar el valor
propuesto.\nRevise las entradas");
                }

        } catch (Exception e) {
                CheckDatos.okCancelaMensaje("No se pudo modificar. "+e.toString());
        }
    }

}
```

C.6. Paquete auxiliares

Se expone a continuación el contenido del paquete auxiliares que contiene el código de las clases usadas para propósitos auxiliares.

CLASE CheckDatos

```
package auxiliares;

import java.awt.Color;
import java.util.ArrayList;
import java.util.Iterator;

import javax.swing.ImageIcon;
import javax.swing.JComboBox;
import javax.swing.JOptionPane;
import javax.swing.JTextField;
import camiones.*;

import cargasEstructura.Carga;

//Esta clase abstracta se usará para comprobar los datos que se van suministrando en los
JTextField de la aplicacion
public abstract class CheckDatos {

       private static double largoCabinaMin=800;
       private static double largoCabinaMax=1500;

       //Este metodo corrige los datos introducidos y avisa si estan mal
    public static boolean corrigeDatos(ArrayList<JTextField> t) {
       JTextField jtf;
       //Variable que almacena si cualquier resultado esta mal
       boolean resultadoOK = true;
       //Iteramos sobre los textos pasados
        Iterator<JTextField> iter = t.iterator();
        while(iter.hasNext()){
              jtf=(JTextField)iter.next();
              //Si se ha introducido una coma se reemplaza por un punto
              if(jtf.getText().contains(","))
                     jtf.setText(jtf.getText().replace(",","."));
              //Se define la variable numero que dirá si el siguiente valor es o no un
número
            boolean numero = true;
            //Comprobamos la longitud de la cadena y vemos si cada caracter es un número
o un punto
            //contamos el numero de puntos
            int contadorPuntos=0;
            for (int i = 0; i < jtf.getText().length(); i++) {
             //Si no es un digito el caracter o no es el punto decimal no es un numero
             //Con esto tambien evitamos numeros negativos
                if ((!Character.isDigit(jtf.getText().charAt(i)))&&(!
(jtf.getText().charAt(i)=='.'))) numero = false;
                //Si empieza con un punto no es un numero
                if((i==0)&&(jtf.getText().charAt(i)=='.')) numero=false;
                //Si el numero de puntos es mayor de 1 no es un numero
                if(jtf.getText().charAt(i)=='.') contadorPuntos++;
                if(contadorPuntos>1) numero=false;
            }
            //Una vez comprobado todo marcamos el numero como rojo si no es correcto para
identificarlo
            if (!numero) {
             jtf.setForeground(Color.RED);
             jtf.setText("0");
             resultadoOK=false;
            } else {
             Formato.setColorEstandar(jtf);
            }
            numero=true;
        }
        //Si en alguno de los digitos de la comprobacion vemos algun error se muestra un
mensaje
```

```
      if(!resultadoOK) okNokMensaje("Alguno de los datos no es un número o es negativo.
\nSe ha sustituido por un 0 y marcado en rojo.\nPor favor revise las entradas si desea
modificar el valor");
      //Vemos si se cumplen todas las restricciones
      return resultadoOK;
   }

   public static boolean corrigeNumerosEnString(ArrayList<String> t) {
      String s;
      //Variable que almacena si cualquier resultado esta mal
      boolean resultadoOK = true;
      //Iteramos sobre los textos pasados
       Iterator<String> iter = t.iterator();
       while(iter.hasNext()){
            s=(String)iter.next();
            //Si se ha introducido una coma se reemplaza por un punto
            if(s.contains(","))
                  s.replace(",",".");
            //Se define la variable numero que dirá si el siguiente valor es o no un
número
           boolean numero = true;
           //Comprobamos la longitud de la cadena y vemos si cada caracter es un número
o un punto
           //contamos el numero de puntos
           int contadorPuntos=0;
           for (int i = 0; i < s.length(); i++) {
            //Si no es un digito el caracter o no es el punto decimal no es un numero
            //Con esto tambien evitamos numeros negativos
               if ((!Character.isDigit(s.charAt(i)))&&(!(s.charAt(i)=='.'))) numero =
false;
               //Si empieza con un punto no es un numero
               if((i==0)&&(s.charAt(i)=='.')) numero=false;
               //Si el numero de puntos es mayor de 1 no es un numero
               if(s.charAt(i)=='.') contadorPuntos++;
               if(contadorPuntos>1) numero=false;
           }
           //Una vez comprobado todo marcamos el numero como rojo si no es correcto para
identificarlo
           if (!numero) {
            resultadoOK=false;
           }
           numero=true;
       }
       //Si en alguno de los digitos de la comprobacion vemos algun error se muestra un
mensaje
      if(!resultadoOK) okNokMensaje("Alguno de los datos no es un número o es
negativo.");
      //Vemos si se cumplen todas las restricciones
      return resultadoOK;
   }

   //Este método comprueba que están todos los datos y no hay ninguno vacio
   public static boolean compruebaDatosCompletos(ArrayList<JTextField> t) {
      //Usamos una variable booleana para asignar la respuesta
      boolean datosCompletos=true;
      //Utilizamos un iterador sobre el arraylist de datos
      Iterator<JTextField> it = t.iterator();
      JTextField jtf;
      //Iteramos sobre los valores para analizar uno por uno
      while(it.hasNext()) {
            jtf=it.next();
            if (jtf.getText().equals("")||(jtf.getText().equals("0"))||
(jtf.getText().equals("0,00"))) {
                  //Vamos viendo el resultado acumulado junto con el que estamos viendo
                  datosCompletos=datosCompletos&&false;
            }else {
                  datosCompletos=datosCompletos&&true;
```

```
            }
       }
       if(!datosCompletos) okNokMensaje("Los datos no se han completado.\nPor favor
revise las entradas para introducir todos los datos");
       //Retornamos el resultado final tras iterar en todos
       return datosCompletos;
    }

    //Este metodo devuelve el largo de cabina Minimo recomendado
    public static double largoCabinaMinRecomendado() {
       return largoCabinaMin;
    }
    //Este metodo devuelve el largo de cabina Maximo recomendado
    public static double largoCabinaMaxRecomendado() {
       return largoCabinaMax;
    }

    //Este metodo comprueba los datos de entrada del camion y da sugerencias sobre ellos
    public static String recomendacionesEntradas(Camion c,double dFD,double dDC,double
dDT,double LC,double dDK,double dKR) {
             String aviso="RECOMENDACIONES: ";
                    double distFrenteEjeDelantero=dFD*Unidades.getConversionLongitud();
                    double
distEjeDelanteroFrenteCaja=dDC*Unidades.getConversionLongitud();
                    double largoCabina=distFrenteEjeDelantero+distEjeDelanteroFrenteCaja;
                    double
distEjeDelanteroEjeTrasero=dDT*Unidades.getConversionLongitud();
                    double largo=LC*Unidades.getConversionLongitud();
                    double distprimerEjeKingPin=dDK*Unidades.getConversionLongitud();
                    double distKingPinEjeRemolque=dKR*Unidades.getConversionLongitud();
                    double
distEjeTraseroEjeRemolque=distprimerEjeKingPin+distKingPinEjeRemolque-
distEjeDelanteroEjeTrasero;
                    double voladizoTrasero=largo-distFrenteEjeDelantero-
distEjeDelanteroEjeTrasero;
                    if(distFrenteEjeDelantero>=largo) aviso=aviso+"\nError: La distancia
del Frente al Eje Delantero es excesiva.";
                    if(distEjeDelanteroEjeTrasero>=largo) aviso=aviso+"\nError: La
distancia del Eje Delantero al Eje Trasero es excesiva.";
                    if(distEjeDelanteroFrenteCaja>=largo) aviso=aviso+"\nError: La
distancia del Frente al comienzo de la caja/remolque del camion es excesiva.";
                    if(largoCabina>=largo) aviso=aviso+"\nError: La longitud de la cabina
es excesiva.";
                    if(voladizoTrasero>=largo) aviso=aviso+"\nError: La distancia en
voladizo es excesiva.";
                    if(voladizoTrasero<0) aviso=aviso+"\nError: Los datos introducidos no
permiten un diseño correcto";
                    if (c.getTipo().getTipo()=='R'){
                           if (largoCabina<CheckDatos.largoCabinaMinRecomendado())
aviso=aviso+"\nLa cabina:\n\t\t-(Distancia frente a eje delantero + Distancia eje
delantero a comienzo de caja)\npodría ser demasiado corta.";
                           if (largoCabina>CheckDatos.largoCabinaMaxRecomendado())
aviso=aviso+"\nLa cabina:\n\t\t-(Distancia frente a eje delantero + Distancia eje
delantero a comienzo de caja)\npodría ser demasiado larga.";

if(voladizoTrasero>=(distFrenteEjeDelantero+distEjeDelanteroEjeTrasero))
aviso=aviso+"\nEl voladizo podría ser excesivo:\n\t\t-Sugerencia: Aumente distancia entre
ejes.";
                           if((voladizoTrasero>=distEjeDelanteroEjeTrasero)&&(!
aviso.contains("El voladizo podría ser excesivo"))) aviso=aviso+"\nEl voladizo podría ser
excesivo:\n\tSugerencia: Aumente distancia entre ejes.";
                           if((distFrenteEjeDelantero+distEjeDelanteroEjeTrasero)>largo)
aviso=aviso+"\nError: Las medidas sobrepasan el largo del camion.";
                    }
                    if (c.getTipo().getTipo()=='A'){
                           if((distprimerEjeKingPin-
distEjeDelanteroFrenteCaja)>(DatosGenericos.getLongExtra()*Unidades.getConversionLongitud
```

```
())) aviso=aviso+"\nError: La distancia de eje delantero a KingPin no puede superar en
más de 300 mm la distancia de eje delantero a Comienzo de Remolque";
                            if(distprimerEjeKingPin>=largo) aviso=aviso+"\nError: La
distancia del primer eje al KingPin es excesiva.";
                            if(distKingPinEjeRemolque>=largo) aviso=aviso+"\nError: La
distancia del KingPin al eje del remolque es excesiva.";
                            if (largoCabina<CheckDatos.largoCabinaMinRecomendado())
aviso=aviso+"\nLa cabina(Distancia frente a eje delantero + Distancia eje delantero a
comienzo remolque)\npodría ser demasiado corta.";
                            if (largoCabina>CheckDatos.largoCabinaMaxRecomendado())
aviso=aviso+"\nLa cabina(Distancia frente a eje delantero + Distancia eje delantero a
comienzo remolque)\npodría ser demasiado larga.";
                            voladizoTrasero=largo-distFrenteEjeDelantero-
distEjeDelanteroEjeTrasero-distEjeTraseroEjeRemolque;
                            if(distprimerEjeKingPin>distEjeDelanteroEjeTrasero)
aviso=aviso+"\nEl KingPin queda por detras del eje trasero de la cabeza tractora. El
cálculo puede generar errores.";
                            if(voladizoTrasero>=distKingPinEjeRemolque) aviso=aviso+"\nEl
voladizo trasero podría ser excesivo.\n\t\t-Sugerencia: Aumente distancia entre ejes.";
                            if(distEjeTraseroEjeRemolque<=distEjeDelanteroEjeTrasero)
aviso=aviso+"\nLa distancia del eje trasero al eje de remolque es menor que la distancia
entre ejes de cabeza tractora.";
                            if(distKingPinEjeRemolque<distEjeDelanteroEjeTrasero)
aviso=aviso+"\nLa distancia del KingPin al eje del remolque es muy pequeña.";

if((distFrenteEjeDelantero+distEjeDelanteroEjeTrasero+distEjeTraseroEjeRemolque+voladizoT
rasero)!=largo) aviso=aviso+"\nLas medidas de ejes no coinciden con el largo del
camión.";
                            if(voladizoTrasero<0) aviso=aviso+"\nEl eje del remolque queda
fuera de la longitud del camion. \nSe corrige la distancia del eje de remolque para que
quede dentro de los límites del camión.";

if((distFrenteEjeDelantero+distprimerEjeKingPin+distKingPinEjeRemolque+voladizoTrasero)!
=largo) aviso=aviso+"Las medidas sobre la posición del KingPin no coinciden con el largo
del camión.";

if((distFrenteEjeDelantero+distEjeDelanteroEjeTrasero+distEjeTraseroEjeRemolque)>largo)
aviso=aviso+"\nError: Las medidas sobrepasan el largo del camión.";
                    }
                    if (aviso.contains("Error")) return aviso+"\n\nNo se puede seguir con
estos datos. Corrija los datos necesarios";
                    else return aviso+"\n\n¿Proseguir?";
    }

    //Este metodo muestra un mensaje de sistema con una advertencia con el string pasado
    public static void okNokMensaje(String s) {
       /*
       //Preparamos un icono para el mensaje en pantalla
       ImageIcon iconoAjustable = new ImageIcon(CheckDatos.class.getResource("Imagenes/
pantallaprincipal/recurso.png"));
       //Llamamos al metodo ajusta escala con el ancho que queremos
       ImageIcon icon=ajustaEscala(iconoAjustable,120);
       //Savamos un panel de advertencia
        *
        */
       //JOptionPane.showMessageDialog(null, s);

       JOptionPane.showMessageDialog(null, s, "Advertencia", 2, escalaIcono(new
ImageIcon(DatosGenericos.getDireccionDB("Imagenes/pantallaprincipal/recurso.png")),2));
       //JOptionPane.showInternalMessageDialog(null,s,"Advertencia",0,icon);
    }

    //Este metodo muestra un mensaje de sistema con un tipo de mensaje determinado
    public static void okNokMensaje(String s, String titulo) {
       /*
       //Preparamos un icono para el mensaje en pantalla
```

```
       ImageIcon iconoAjustable = new ImageIcon(CheckDatos.class.getResource("Imagenes/
pantallaprincipal/recurso.png"));
       //Llamamos al metodo ajusta escala con el ancho que queremos
       ImageIcon icon=ajustaEscala(iconoAjustable,120);
       //Savamos un panel de advertencia
       JOptionPane.showInternalMessageDialog(null,s,titulo,0,icon);
       */

       //JOptionPane.showMessageDialog(null, s);
       JOptionPane.showMessageDialog(null, s, "Advertencia", 2, escalaIcono(new
ImageIcon(DatosGenericos.getDireccionDB("Imagenes/pantallaprincipal/recurso.png")),2));
    }

    //Este metodo muestra un mensaje de sistema con una advertencia con el string pasado
con posibilidad de cancelar
    public static boolean okCancelaMensaje(String s) {
       boolean ok=true;
       int respuesta=0;
       /*
       //Preparamos un icono para el mensaje en pantalla
       ImageIcon iconoAjustable = new ImageIcon(CheckDatos.class.getResource("Imagenes/
pantallaprincipal/recurso.png"));
       //Llamamos al metodo ajusta escala con el ancho que queremos
       ImageIcon icon=ajustaEscala(iconoAjustable,120);
       //Savamos un panel de advertencia
        *
        *
        */
       respuesta=JOptionPane.showConfirmDialog(null, s, "Advertencia",
JOptionPane.OK_CANCEL_OPTION, JOptionPane.QUESTION_MESSAGE ,escalaIcono(new
ImageIcon(DatosGenericos.getDireccionDB("Imagenes/pantallaprincipal/recurso.png")),2));

       if (respuesta==0) ok=true;
       else ok=false;
       return ok;
    }

    public static boolean compruebaNoCero(JTextField jtf) {
       boolean cargaNoCero=true;
       cargaNoCero=cargaNoCero&&(!jtf.getText().equals("0"));
       return cargaNoCero;
    }

    public static boolean compruebaValoresValidosCargas(Proyecto proyecto, JTextField
nombre,JTextField posicion, JTextField longitud,JComboBox<String> tipo) {
       Camion c = proyecto.getCamion();
       boolean valoresValidos=true;
       //Revisamos primero que no estemos introduciendo un nombre duplicado
       ArrayList<Carga> al=proyecto.getCargas();
       Iterator<Carga> it = al.iterator();
       while(it.hasNext()) {
              Carga q=(Carga)it.next();
              valoresValidos=valoresValidos&&(!q.getNombre().equals(nombre.getText()));
       }
       if(!valoresValidos) CheckDatos.okNokMensaje("El nombre de la carga ya existe.
Modifíquelo");
       else {
              double
posicionCarga=Double.parseDouble(posicion.getText())*Unidades.getConversionLongitud();
              double longitudCamion=c.getLargo();
              double longitudCarga=0;
              String cargaTipo="Puntual";
              //Comprobamos que la posicion de la carga no esté fuera de los límites del
camion
              valoresValidos=valoresValidos&&(!(posicionCarga>longitudCamion));
              if(!valoresValidos) CheckDatos.okNokMensaje("La posición es mayor que el
largo del camión");
```

```
              //Si es distribuida comprobamos tambien que la longitud
              //de la carga distribuida desde el punto de aplicación no sobrepase la
longitud del camion

longitudCarga=Double.parseDouble(longitud.getText())*Unidades.getConversionLongitud();
              cargaTipo=(String)tipo.getSelectedItem();
              //Comprobamos
              if (cargaTipo.equals("Distribuida")) valoresValidos=valoresValidos&&(!
((posicionCarga+longitudCarga)>longitudCamion));
              //Avisamos
              if(!valoresValidos&&(cargaTipo.equals("Distribuida")))
CheckDatos.okNokMensaje("La longitud de la carga distribuida excede la longitud del
camión");
           }
       return valoresValidos;
    }

    //Este método comprueba que están todos los datos y no hay ninguno vacio en las
cargas
    public static boolean compruebaDatosCompletosCargas(ArrayList<String> al) {
       boolean datosCompletos=true;
       Iterator<String> it = al.iterator();
       String dato="";
       while(it.hasNext()) {
              dato=(String)it.next();
              datosCompletos=datosCompletos&&(!dato.equals(""));
       }
       return datosCompletos;
    }

    //Este metodo devuelve true si las propiedades de la viga del proyecto para la cabeza
tractora o camion han sido asignadas,false en caso contrario
    private static boolean checkPropiedadesVigaCamionOK(Proyecto p) {
       double area=p.getCamion().getAreaViga();
       double inercia=p.getCamion().getInerciaViga();
       if((area!=0)&&(inercia!=0)) return true;
       else return false;
    }

    //Este metodo devuelve true si las propiedades de la viga del proyecto para el
remolque han sido asignadas, false en caso contrario
    private static boolean checkPropiedadesVigaRemolqueOK(Proyecto p) {
       if(p.getCamion().getTipo().getTipo()=='A') {
              double area=((CamionArticulado)p.getCamion()).getAreaVigaRemolque();
              double inercia=((CamionArticulado)p.getCamion()).getInerciaVigaRemolque();
              if((area!=0)&&(inercia!=0)) return true;
              else return false;
       } else return true;
    }

    //Este metodo comprueba tanto remolque como Cabeza o Camion
    public static boolean checkViga(Proyecto p) {
       return (checkPropiedadesVigaCamionOK(p)&&checkPropiedadesVigaRemolqueOK(p));
    }

    //Este metodo ajusta la escala del icono al tamaño que deseamos para la advertencia
    private static ImageIcon ajustaEscala(ImageIcon i, int ancho) {
       // alto (para que conserve la proporcion pasamos -1)
       int alto = -1;
       //Escalamos el icono
       ImageIcon iconoOK = new ImageIcon(i.getImage().getScaledInstance(ancho, alto,
java.awt.Image.SCALE_DEFAULT));
       //Retornamos el icono escalado
       return iconoOK;
    }
```

```
    //Este método escala los iconos de los botones con un coeficiente de escala para
evitar problemas por tamaños
    private static ImageIcon escalaIcono(ImageIcon i, int s) {
       return new ImageIcon(i.getImage().getScaledInstance(80*s, -1,
java.awt.Image.SCALE_DEFAULT));
    }
}
```

CLASE CogeImagen

```
package auxiliares;
import java.awt.Image;

import javax.swing.ImageIcon;

public class CogeImagen {
       public CogeImagen() {}
       public Image devuelveImagen(String nombre,int s) {
                     ImageIcon logo=escalaIcono(new
ImageIcon(getClass().getResource(nombre)),s);
                     return logo.getImage();
       }

    //Este método escala los iconos de los botones con un coeficiente de escala para
evitar problemas por tamaños
    public ImageIcon escalaIcono(ImageIcon i, int s) {
       return new ImageIcon(i.getImage().getScaledInstance(40*s, -1,
java.awt.Image.SCALE_DEFAULT));
    }

}
```

CLASE CreaPDF

```
package auxiliares;

import java.awt.Color;
import java.awt.image.BufferedImage;
import java.io.FileNotFoundException;
import java.util.ArrayList;
import java.util.Iterator;
import camiones.*;

import javax.swing.ImageIcon;

import com.itextpdf.io.image.ImageData;
import com.itextpdf.io.image.ImageDataFactory;
import com.itextpdf.kernel.pdf.PdfDocument;
import com.itextpdf.kernel.pdf.PdfWriter;
import com.itextpdf.layout.Document;
import com.itextpdf.layout.element.AreaBreak;
import com.itextpdf.layout.element.Image;
import com.itextpdf.layout.element.Paragraph;
import com.itextpdf.layout.element.Text;

public abstract class CreaPDF {
	public static String nombre;
	private static Proyecto proyecto;
	private static ArrayList<BufferedImage> listaImagenes,listaImagenes2;

	public static void creaResultadosPdf() {
		String path=nombre+".pdf";
		try {
			PdfWriter pdfw = new PdfWriter(path);
			PdfDocument pdfdoc=new PdfDocument(pdfw);
			pdfdoc.addNewPage();
			Document document = new Document(pdfdoc);

			try {
				CogeImagen ci=new CogeImagen();
				ImageData imageninic =
ImageDataFactory.create(ci.devuelveImagen(DatosGenericos.getDireccion("Imagenes/
pantallaprincipal/recurso.png"),10), Color.BLACK);
				Image im1 = new Image(imageninic);
				document.add(im1);
			} catch (Exception e) {
				CheckDatos.okCancelaMensaje("Fallo al cargar el logo");
			};


			Paragraph p = new Paragraph("");
			document.add(p);

			//Añadimos informacion del proyecto
			//TITULO
			Text texto = new Text("ANEXO TÉCNICO PROYECTO CARROZADO");
			texto.setBold();
			texto.setUnderline();
			texto.setFontSize(20);
			p = new Paragraph(texto);
			document.add(p);

			//DATOS TECNICOS
			texto = new Text("Datos del proyecto");
			texto.setBold();
			texto.setUnderline();
			texto.setFontSize(16);
			p = new Paragraph(texto);
			document.add(p);
```

```
                    p = new Paragraph("Nombre: "+proyecto.getCamion().getCliente());
                    document.add(p);
                    p = new Paragraph("Matrícula: "+proyecto.getCamion().getMatricula());
                    document.add(p);
                    p = new Paragraph("Bastidor: "+proyecto.getCamion().getBastidor());
                    document.add(p);
                    p = new Paragraph("Marca: "+proyecto.getCamion().getMarca());
                    document.add(p);
                    p = new Paragraph("Modelo: "+proyecto.getCamion().getModelo());
                    document.add(p);
                    p = new Paragraph("Tipo de Camión:
"+proyecto.getCamion().getTipo().getDescripcion());
                    document.add(p);
                    p = new Paragraph("");
                    document.add(p);

                    //DATOS GENERALES DE DIMENSIONES DEL CAMION
                    texto = new Text("Datos generales de dimensiones del camión");
                    texto.setBold();
                    texto.setUnderline();
                    texto.setFontSize(16);
                    p = new Paragraph(texto);
                    document.add(p);
                    p = new Paragraph("Distancia Frente - Eje delantero:
"+proyecto.getCamion().getdistFrenteEjeDelantero()+"mm.");
                    document.add(p);
                    p = new Paragraph("Distancia Eje Delantero - Comienzo de Caja:
"+proyecto.getCamion().getDistanciaEjeDelanteroComienzoCaja()+"mm.");
                    document.add(p);
                    p = new Paragraph("Distancia Eje Delantero - Eje Trasero:
"+proyecto.getCamion().getdistEjeDelanteroEjeTrasero()+"mm.");
                    document.add(p);
                    p = new Paragraph("Longitud Total del Camión:
"+proyecto.getCamion().getLargo()+"mm.");
                    document.add(p);
                    if(proyecto.getCamion().getTipo().getTipo()=='A') {
                           CamionArticulado ca = (CamionArticulado)proyecto.getCamion();
                           p = new Paragraph("Distancia Eje Delantero - KingPin:
"+ca.getDistanciaEjeDelanteroKingPin()+"mm.");
                           document.add(p);
                           p = new Paragraph("Distancia KingPin - Eje Remolque:
"+ca.getDistanciaKingPinEjeRemolque()+"mm.");
                           document.add(p);
                    }
                    p = new Paragraph("");
                    document.add(p);

                    //DATOS GENERALES DE CARGAS DEL CAMION
                    texto = new Text("Datos generales de cargas en ejes del camión");
                    texto.setBold();
                    texto.setUnderline();
                    texto.setFontSize(16);
                    p = new Paragraph(texto);
                    document.add(p);
                    p = new Paragraph("Carga máxima tándem delantero:
"+proyecto.getCamion().getCargaMaxEjeDelantero()+"Kg.");
                    document.add(p);
                    p = new Paragraph("Ejes en tándem delantero:
"+proyecto.getCamion().getEjesPorTandemDelantero());
                    document.add(p);
                    p = new Paragraph("Carga máxima tándem trasero:
"+proyecto.getCamion().getCargaMaxEjeTrasero()+"Kg.");
                    document.add(p);
                    p = new Paragraph("Ejes en tándem trasero:
"+proyecto.getCamion().getEjesPorTandemTrasero());
                    document.add(p);
                    //PASAMOS PAGINA
```

```
                    document.add(new AreaBreak());
                    if(proyecto.getCamion().getTipo().getTipo()=='A') {
                          CamionArticulado ca = (CamionArticulado)proyecto.getCamion();
                          p = new Paragraph("Carga máxima tándem remolque:
"+ca.getCargaMaximaEjeRemolque()+"Kg.");
                          document.add(p);
                          p = new Paragraph("Ejes en tándem remolque:
"+ca.getEjesPorTandemRemolque());
                          document.add(p);
                    }
                    p = new Paragraph("");
                    document.add(p);


                    //DATOS GENERALES DE TARA MAXIMA
                    texto = new Text("Datos generales de cargas máximas del camión");
                    texto.setBold();
                    texto.setUnderline();
                    texto.setFontSize(16);
                    p = new Paragraph(texto);
                    document.add(p);
                    p = new Paragraph("Masa Maxima Autorizada:
"+proyecto.getCamion().getMMA()+"Kg.");
                    document.add(p);
                    p = new Paragraph("Peso Maximo Autorizado:
"+proyecto.getCamion().getPMA()+"Kg.");
                    document.add(p);
                    p = new Paragraph("");
                    document.add(p);


                    //METEMOS LAS CARGAS
                    texto = new Text("Lista de cargas del proyecto");
                    texto.setBold();
                    texto.setUnderline();
                    texto.setFontSize(16);
                    p = new Paragraph(texto);
                    document.add(p);
                    try {
                          BufferedImage bi=listaImagenes2.get(0);
                          ImageData imageCargas = ImageDataFactory.create(bi,
Color.BLACK);
                          Image i2 = new Image(imageCargas);
                          document.add(i2);
                    } catch (Exception e) {
                          CheckDatos.okCancelaMensaje("Se produjo un problema intentando
insertar la tabla de cargas");
                    }
                    p = new Paragraph("");
                    document.add(p);
                    p = new Paragraph("");
                    document.add(p);


                    //RESULTADOS DEL PROYECTO
                    texto = new Text("Resultados del proyecto");
                    texto.setBold();
                    texto.setUnderline();
                    texto.setFontSize(20);
                    p = new Paragraph(texto);
                    document.add(p);
                    p = new Paragraph("");
                    document.add(p);


                    //REACCIONES
                    texto = new Text("Reacciones del proyecto");
```

```
                    texto.setBold();
                    texto.setUnderline();
                    texto.setFontSize(16);
                    p = new Paragraph(texto);
                    document.add(p);
                    p = new Paragraph("");
                    document.add(p);

                    try {
                           BufferedImage bi=listaImagenes2.get(1);
                           ImageData imageReacciones = ImageDataFactory.create(bi,
Color.BLACK);
                           Image i2 = new Image(imageReacciones);
                           document.add(i2);
                    } catch (Exception e) {
                           CheckDatos.okCancelaMensaje("Se produjo un problema intentando
insertar las reacciones");
                    }
                    p = new Paragraph("");
                    document.add(p);


                    //PASAMOS PAGINA
                    document.add(new AreaBreak());

                    //DIAGRAMAS
                    texto = new Text("ESQUEMAS Y DIAGRAMAS");
                    texto.setBold();
                    texto.setUnderline();
                    texto.setFontSize(20);
                    p = new Paragraph(texto);
                    document.add(p);
                    texto = new Text("Del camión");
                    texto.setBold();
                    texto.setUnderline();
                    texto.setFontSize(16);
                    p = new Paragraph(texto);
                    document.add(p);
                    p = new Paragraph("");
                    document.add(p);
                    try {
                           //Damos el nombre al pdf
                           CreaPDF.setNombrePDF(path);

                           //Sacamos las imagenes en el orden Esquema, Cortante y Momento
y las metemos en el documento
                           Iterator<BufferedImage> ibi =listaImagenes.iterator();
                           //Con el contador estableceremos si es el Esquema, cortante o
momento
                           int contador = 0;
                           while(ibi.hasNext()) {
                                  switch(contador){
                                         case 0:
                                                p = new Paragraph("Esquema del camión");
                                                document.add(p);
                                                break;
                                         case 1:
                                                p = new Paragraph("Diagrama de cortantes
del camión");
                                                document.add(p);
                                                break;
                                         case 2:
                                                p = new Paragraph("Diagrama de momentos
del camión");
                                                document.add(p);
                                                break;
                                         case 3:
```

```
                                                texto = new Text("Del remolque");
                                                texto.setBold();
                                                texto.setUnderline();
                                                texto.setFontSize(16);
                                                p = new Paragraph(texto);
                                                document.add(p);
                                                p = new Paragraph("Esquema del remolque");
                                                document.add(p);
                                                break;
                                          case 4:
                                                p = new Paragraph("Diagrama de cortantes
del remolque");
                                                document.add(p);
                                                break;
                                          case 5:
                                                p = new Paragraph("Diagrama de momentos
del remolque");
                                                document.add(p);
                                                break;
                                          default:
                                                break;
                                    }
                                    BufferedImage imagen = ibi.next();
                                    ImageData image = ImageDataFactory.create(imagen,
Color.BLACK);
                                    Image i = new Image(image);
                                    document.add(i);
                                    //Si es el final de la cabeza introducimos otra pagina
                                    if((contador==2)||(contador==5)) document.add(new
AreaBreak());
                                    p = new Paragraph("");
                                    document.add(p);
                                    p = new Paragraph("");
                                    document.add(p);
                                    contador++;
                              }
                        } catch(Exception e3) {
                              CheckDatos.okCancelaMensaje("Error al crear la imagen. "+e3);
                        }

                        //Cerramos el documento
                        document.close();

                  } catch (FileNotFoundException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                  }

      }

      //Este metodo asigna el nombre con el que se creará el PDF
      public static void setNombrePDF(String n) {
            nombre=n;
      }

      //Este metodo asigna el proyecto del que se creara el PDF
      public static void setProyecto(Proyecto p) {
            proyecto=p;
      }

      //Este metodo asigna la lista de imagenes que se deben pegar en el PDF
      public static void setImages(ArrayList<BufferedImage> imagenes) {
            listaImagenes=imagenes;
      }

      //Este metodo asigna las imagenes de las tablas de cargas y de reacciones
      public static void setImages2(ArrayList<BufferedImage> imagenes2) {
```

```
                listaImagenes2=imagenes2;
        }

}
```

CLASE DatosGenericos

```
package auxiliares;

import java.awt.Dimension;
import java.awt.Toolkit;
import java.io.File;

import javax.swing.JFrame;

public abstract class DatosGenericos {
	private static final double longitudExtraDeTractora=300;

	public static double getLongExtra() {
		return longitudExtraDeTractora;
	}

	public static int getAnchoPantalla() {
		Dimension pantalla = Toolkit.getDefaultToolkit().getScreenSize();
	    return pantalla.width;
	}

	public static int getAltoPantalla() {
		Dimension pantalla = Toolkit.getDefaultToolkit().getScreenSize();
	    return pantalla.height;
	}

	public static void setPosicion(JFrame jf, int anchura, int altura) {
		int alto = getAltoPantalla();
	    int ancho = getAnchoPantalla();
	    //Establecemos las dimensiones y la posicion;
	    jf.setBounds( ancho/2-anchura/2,alto/2-altura/2,anchura , altura );
	}

	public static String getDireccion(String dir) {
		String separador = File.separator;
		String direccion=dir.replace("/", separador);
		String ruta = separador+direccion;
		return ruta;
	}

	public static String getDireccionDB(String dir) {
		String separador = File.separator;
		String direccion=dir.replace("/", separador);
		String ruta = "."+separador+direccion;
		return ruta;
	}
}
```

CLASE FicheroUsado

```
package auxiliares;


import java.awt.Component;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.Locale;
import camiones.*;

import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.UIManager;
import javax.swing.filechooser.FileNameExtensionFilter;

public class FicheroUsado extends JPanel{
	private static final long serialVersionUID = 1L;
	public String nombre;
	public FicheroUsado() {
		Formato.setFormatoEstandar(this);
	}
	public void guarda(Object dato) {
		File ficheroGuardar=null;
		ficheroGuardar=abreVentanaFicheroGuarda();
		String ruta = ficheroGuardar.getPath();
		if(!ruta.endsWith(".txt"))
		{
		    ficheroGuardar = new File(ruta + ".txt");
		}
		try {
			if(ficheroGuardar!=null) {
				ObjectOutputStream oos = new ObjectOutputStream(new
FileOutputStream(ficheroGuardar));
				oos.writeObject(dato);
				oos.close();
			}
		} catch (Exception e) {
			CheckDatos.okNokMensaje("Se detectó un problema al escribir el
fichero");
		}
	}
	public Object abreFichero() {
		File ficheroAbrir=null;
		ficheroAbrir=abreVentanaFicheroCarga();
		String ruta = ficheroAbrir.getPath();
		if(!ruta.endsWith(".txt"))
		{
		    ficheroAbrir = new File(ruta + ".txt");
		}
		Object objeto=null;
		try {
			if(ficheroAbrir!=null) {
				ObjectInputStream ois = new ObjectInputStream(new
FileInputStream(ficheroAbrir));
				objeto=ois.readObject();
				ois.close();
			}
		} catch (Exception e) {
```

```
                    CheckDatos.okNokMensaje("Se detectó un problema al leer del
fichero");
             }
             return objeto;
      }

      public File abreVentanaFicheroCarga() {
             File archivoElegido=null;
             JFileChooser fichero=new JFileChooser();
             fichero.setFileFilter(new FileNameExtensionFilter("Camión (txt)","txt"));
             int respuesta=fichero.showOpenDialog(null);
             if(respuesta==JFileChooser.APPROVE_OPTION) {
                    archivoElegido=fichero.getSelectedFile();
             }
             return archivoElegido;
      }
      public File abreVentanaFicheroGuarda() {
             File archivoElegido=null;
             JFileChooser fichero2=new JFileChooser();
             fichero2.setFileFilter(new FileNameExtensionFilter("Camión (txt)","txt"));
             fichero2.setApproveButtonText("Aceptar");
             int respuesta=fichero2.showSaveDialog(null);
             if(respuesta==JFileChooser.APPROVE_OPTION) {
                    archivoElegido=fichero2.getSelectedFile();
             }
             return archivoElegido;
      }

      //Como no existe opcion para cambiar el texto para Cancelar del JFileChooser he
buscado sus componentes para elegirlo y cambiarlo
      public void cambiarBotonCancelarTextoAbrir(JFileChooser fichero) {
             //Asignamos el valor Formato en vez de Type
             JPanel menuElegir=(JPanel)fichero.getComponent(4);
             JPanel menuSeleccion=(JPanel)menuElegir.getComponent(2);
             JPanel menuSeleccionP2=(JPanel)menuElegir.getComponent(0);
             JLabel labelSeleccion=(JLabel)menuSeleccionP2.getComponent(0);
             labelSeleccion.setText("Formato:");
             //Asignamos el valor Cancelar en vez de Cancel
             JPanel menuSeleccion2=(JPanel)menuSeleccion.getComponent(1);
             JPanel menuSeleccion3=(JPanel)menuSeleccion2.getComponent(1);
             JButton btnCancelar=(JButton)menuSeleccion3.getComponent(2);
             btnCancelar.setText("Cancelar");
             //Eliminamos lo que no queremos mostrar
             menuSeleccion2.remove(0);
      }

      //Como no existe opcion para cambiar el texto para Cancelar del JFileChooser he
buscado sus componentes para elegirlo y cambiarlo
      public void cambiarBotonCancelarTextoGuardar(JFileChooser fichero) {
             //Cambiamos el texto para pedir el formato
             JPanel menuSeleccion=(JPanel)fichero.getComponent(4);
             JPanel menuSeleccion2=(JPanel)menuSeleccion.getComponent(0);
             JLabel etiqueta=(JLabel)menuSeleccion2.getComponent(0);
             etiqueta.setText("Formato:");
             //Eliminamos la posibilidad de crear otra carpeta para simplificar
             JPanel menuSeleccion3=(JPanel)menuSeleccion.getComponent(2);
             JPanel menuSeleccion4=(JPanel)menuSeleccion3.getComponent(1);
             JPanel menuSeleccion5=(JPanel)menuSeleccion4.getComponent(0);
             menuSeleccion5.remove(1);
             //Cambiamos el texto de Cancel por Cancelar
             JPanel menuSeleccion6=(JPanel)menuSeleccion4.getComponent(1);
             JButton btn1=(JButton)menuSeleccion6.getComponent(2);
             btn1.setText("Cancelar");
      }
}
```

CLASE Formato

```
package auxiliares;

import java.awt.Color;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;

import javax.swing.BorderFactory;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JMenu;
import javax.swing.JMenuItem;
import javax.swing.JPanel;
import javax.swing.JTable;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.SwingConstants;
import javax.swing.border.LineBorder;
import javax.swing.border.TitledBorder;
import javax.swing.table.JTableHeader;
import javax.swing.table.TableColumn;

public abstract class Formato {
	//Definimos los valores comunes para todos los componentes y elementos
	private static Color colorDefecto=Color.BLACK; //Color de fuente por defecto
	private static Color colorDefectoBorde=Color.GRAY; //Color de bordes por defecto
	private static Color colorSeleccionado=Color.BLUE; //Color del borde de un boton
al seleccionarlo entre opciones
	private static Color colorCotas=Color.RED;//Color de cotas
	private static Color colorDibujoCargas=Color.RED;//Color  al dibujar las cargas
	private static Color colorTextoCargas=Color.DARK_GRAY;//Color del texto de las
cotas
	private static Color colorFondoBlanco=Color.WHITE; //Color del fondo
	private static Color colorGraficas=Color.LIGHT_GRAY; //Color de graficas
	private static Color colorRellenoFormas=Color.LIGHT_GRAY; //Color de rellenos al
dibujar formas
	private static Color colorLineasFormas=Color.BLUE; //Color de lineas al dibujar
formas
    private static Font fuenteDefecto = new Font("Avenir", Font.PLAIN, 14); //Tipo letra
por defecto
    private static Font fuenteMenor = new Font("Avenir", Font.PLAIN, 12); //Tipo letra
mas pequeño
    private static Font fuenteAplicacion = new Font("Avenir",Font.BOLD, 30);//Tipo letra
titulo aplicacion
    private static Font fuenteTipoCamion = new Font("Avenir",Font.BOLD, 20);//Tipo letra
titulo aplicacion
    private static Font fuenteTitulo = new Font("Avenir",Font.BOLD, 20);//Tipo letra
titulo aplicacion
    private static Font fuenteTituloMediano = new Font("Avenir",Font.BOLD, 16);//Tipo
letra titulo aplicacion
	private static Font fuenteCotas = new Font("Avenir", Font.ITALIC, 12); //Tipo
letra para cotas
	private static Font fuenteCargas = new Font("Avenir", Font.ITALIC, 10); //Tipo
letra para cotas
	private static Font fuenteEjes = new Font("Avenir", Font.ITALIC, 8); //Tipo letra
para cotas
	private static Font fuenteTabla = new Font("Avenir", Font.ITALIC, 12); //Tipo
letra para cotas
    private static LineBorder bordeDefecto=new LineBorder(colorDefectoBorde); //Tipo de
bordes
    private static LineBorder bordeSeleccionado=new LineBorder(colorSeleccionado); //Tipo
de borde para seleccion
```

```
//Metodos para formato de TEXTOS
//normal
public static void setFormatoEstandar(Component c) {
   c.setFont(fuenteDefecto);
   c.setForeground(colorDefecto);
}
//mas pequeño
public static void setFormatoMenor(Component c) {
   c.setFont(fuenteMenor);
   c.setForeground(colorDefecto);
}
//Cotas
public static void setFormatoCotas(Component c) {
   c.setFont(fuenteCotas);
   c.setForeground(colorCotas);
}
//Cotas en Graficas
public static void setFormatoCotas(Graphics2D g) {
         g.setFont(fuenteCotas);
         g.setColor(Color.RED);
}
//Cotas en Graficas
public static void setFormatoCotas(Graphics g) {
         g.setColor(Color.RED);
}
//Aplicacion
public static void setFormatoAplicacion(JLabel jl) {
   jl.setFont(fuenteAplicacion);
    jl.setHorizontalAlignment(SwingConstants.CENTER);
    jl.setAlignmentX(SwingConstants.CENTER);
   jl.setForeground(colorDefecto);
}
//Titulo Imagenes Camiones
public static void setFormatoTipoCamion(JLabel jl) {
    jl.setFont(fuenteTipoCamion);
    jl.setHorizontalAlignment(SwingConstants.LEFT);
    jl.setAlignmentX(SwingConstants.LEFT);
    jl.setForeground(Color.GRAY);
    jl.setPreferredSize(new Dimension(500,50));
}
//Fuente para titulos
public static void setFuenteTitulo(JLabel jl) {
    jl.setFont(fuenteTitulo);
    jl.setForeground(Color.WHITE);
}
//Fuente para menuItem
public static void setFuenteMenuItem(JMenuItem jmi) {
    jmi.setFont(fuenteDefecto);
}
//Fuente para menu
public static void setFuenteMenu(JMenu jm) {
    jm.setFont(fuenteDefecto);
}
//Fuente para aplicacion
public static void setFuenteAplicacion(JLabel jl) {
    jl.setFont(fuenteAplicacion);
    jl.setForeground(Color.WHITE);
}

public static void setFuenteTituloMediano(JLabel jl) {
    jl.setFont(fuenteTituloMediano);
    jl.setForeground(Color.WHITE);
}

public static void setFuenteTabla(JLabel jl) {
    jl.setFont(fuenteTabla);
    jl.setHorizontalAlignment(SwingConstants.CENTER);
```

```
    }

    public static void setFuenteTablaSeleccionada(JLabel jl) {
        jl.setFont(fuenteTabla);
        jl.setForeground(Color.RED);
        jl.setBackground(Color.LIGHT_GRAY);
        jl.setHorizontalAlignment(SwingConstants.CENTER);
    }

    //Damos formato a las tablas
    public static void setFormatoContenidoTabla(JTable jt) {
       int columnas = jt.getColumnCount();
        JTableHeader jtableHeader = jt.getTableHeader();
        jtableHeader.setDefaultRenderer(new FormatoEncabezadoTabla());
        jt.setTableHeader(jtableHeader);
        //personalizamos el formato de las celdas
        for (int i = 0; i < columnas; i++) {
                jt.getColumnModel().getColumn(i).setCellRenderer(new
FormatoCeldaTabla());
        }
    }

    //Damos formato a las celdas seleccionadas
    public static void setFormatoContenidoTablaSeleccionado(JTable jt, int fila) {
       int columnas = jt.getColumnCount();
        //personalizamos el formato de las celdas
        for (int i = 0; i < columnas; i++) {
             jt.getColumnModel().getColumn(i).setCellRenderer(new
FormatoCeldaTablaSeleccionada(fila));
        }
    }

    //Fuente para gráficos
    public static void setFuenteGraficos(Graphics g) {
        g.setFont(fuenteDefecto);
    }
    //Fuente para cargas
    public static void setFuenteCargas(Graphics g) {
        g.setFont(fuenteCargas);
    }
    //Fuente para ejes
    public static void setFuenteEjes(Graphics g) {
        g.setFont(fuenteEjes);
    }
    //Fuente para ejes
    public static void setFuenteTitulo(Graphics g) {
        g.setFont(fuenteAplicacion);
    }

    //Metodos para el COLOR
    //Por defecto
    public static void setColorEstandar(Component c) {
       c.setForeground(colorDefecto);
    }
    //Color seleccionado
    public static void setColorSeleccionado(Component c) {
       c.setForeground(colorSeleccionado);
    }
    //Color seleccionado
    public static void setColorCotas(Component c) {
       c.setForeground(colorCotas);
    }
    public static void setColorGraficas(Graphics2D g) {
       g.setColor(colorDefecto);
    }
    public static void setColorGraficas(Graphics g) {
       g.setColor(colorGraficas);
```

```
    }
    public static void setColorCargas(Graphics g) {
       g.setColor(colorDibujoCargas);
    }
    public static void setColorTextoCargas(Graphics g) {
       g.setColor(colorTextoCargas);
    }
    public static void setColorLinea(Graphics2D g) {
       g.setColor(colorLineasFormas);
    }
    public static void setColorRelleno(Graphics2D g) {
       g.setColor(colorRellenoFormas);
    }
    public static void setColorLinea(Graphics g) {
       g.setColor(colorLineasFormas);
    }
    public static void setColorRelleno(Graphics g) {
       g.setColor(colorRellenoFormas);
    }


    //Metodos para los BORDES
    //Boton Por defecto
    public static void setBordeEstandar(JButton jbtn) {
       jbtn.setBorder(bordeDefecto);
       jbtn.setBorderPainted(true);
    }
    //Tabla Por defecto
    public static void setFormatoTable(JTable jt) {
       jt.setBorder(bordeDefecto);
       jt.setAutoCreateColumnsFromModel(true);

    }
    //Boton Seleccionado
    public static void setBordeSeleccionado(JButton jbtn) {
       jbtn.setBorder(bordeSeleccionado);
       jbtn.setBorderPainted(true);
    }
    //JTEXTAREAS
    //Por defecto
    public static void setBordeEstandar(JTextArea jta) {
       jta.setBorder(bordeDefecto);
       jta.setForeground(colorSeleccionado);
        jta.setBackground(colorFondoBlanco);
    }
    //JPANELS
    //Por defecto con Titulo
    public static void setBorder(JPanel jp, String s) {
             TitledBorder tituloesquema;
        tituloesquema = BorderFactory.createTitledBorder(s);
        jp.setBorder(tituloesquema);
    }
    //Por defecto sin Titulo
    public static void setBorder(JPanel jp) {
       jp.setBorder(new LineBorder(colorDefectoBorde));
    }
    //JTEXT
    //Por defecto en Jtext
    public static void setBorder(JTextField texto) {
       texto.setForeground(colorDefecto);
       texto.setBorder(bordeDefecto);
    }

}
```

CLASE FormatoCeldaTabla

```
package auxiliares;

import java.awt.Color;
import java.awt.Component;
import java.awt.Dimension;
import camiones.*;

import javax.swing.JComponent;
import javax.swing.JLabel;
import javax.swing.JTable;
import javax.swing.border.LineBorder;
import javax.swing.table.TableCellRenderer;

public class FormatoCeldaTabla implements TableCellRenderer {

       @Override
       public Component getTableCellRendererComponent(JTable table, Object value, boolean
isSelected, boolean hasFocus,int row, int column) {
              JComponent jcomponent = null;
                     if( value instanceof String ) {
                    jcomponent = new JLabel((String) value);
               } else if (value instanceof Integer) {
                     jcomponent = new JLabel(Integer.toString((int)value));
               } else if (value instanceof Double) {
                     jcomponent = new JLabel(Double.toString((double)value));
              }
        Formato.setFuenteTabla((JLabel)jcomponent);
        jcomponent.setBorder(new LineBorder(Color.GRAY));
        jcomponent.setBackground(Color.LIGHT_GRAY);
        table.getColumnModel().getColumn(column).setPreferredWidth(100);
        return jcomponent;
    }

}
```

CLASE FormatoCeldaTablaSeleccionada

```
package auxiliares;

import java.awt.Color;
import java.awt.Component;
import camiones.*;

import javax.swing.JComponent;
import javax.swing.JLabel;
import javax.swing.JTable;
import javax.swing.border.LineBorder;
import javax.swing.table.TableCellRenderer;

public class FormatoCeldaTablaSeleccionada implements TableCellRenderer {
       int fila;
       public FormatoCeldaTablaSeleccionada() {

       }
       //Creamos un constructor en el que podamos pasarle la fila que queremos
       public FormatoCeldaTablaSeleccionada(int fila) {
             this.fila=fila;
       }
       @Override
       public Component getTableCellRendererComponent(JTable table, Object value, boolean
isSelected, boolean hasFocus,int row, int column) {
             JComponent jcomponent = null;
                   if( value instanceof String ) {
                  jcomponent = new JLabel((String) value);
              } else if (value instanceof Integer) {
                   jcomponent = new JLabel(Integer.toString((int)value));
              } else if (value instanceof Double) {
                   jcomponent = new JLabel(Double.toString((double)value));
             }
        jcomponent.setBorder(new LineBorder(Color.GRAY));
        if(row==fila) {
             Formato.setFuenteTablaSeleccionada((JLabel)jcomponent);
        } else {
            Formato.setFuenteTabla((JLabel)jcomponent);
        }
        return jcomponent;
    }
}
```

CLASE FormatoEncabezadoTabla

```
package auxiliares;

import java.awt.Color;
import java.awt.Component;
import javax.swing.JComponent;
import javax.swing.JLabel;
import javax.swing.JTable;
import javax.swing.border.LineBorder;
import javax.swing.table.TableCellRenderer;
import camiones.*;

public class FormatoEncabezadoTabla  implements TableCellRenderer {

    @Override
    public Component getTableCellRendererComponent(JTable table, Object value, boolean
isSelected, boolean hasFocus, int row, int column) {
        JComponent jcomponent = new JLabel((String) value);
        Formato.setFuenteTabla((JLabel)jcomponent);
        jcomponent.setBorder(new LineBorder(Color.GRAY));
        jcomponent.setBackground(new Color(46, 134, 193));
        jcomponent.setForeground(Color.WHITE);
        jcomponent.setOpaque(true);
        return jcomponent;
    }

}
```

CLASE IconoAjustado

```
package auxiliares;

//Esta clase se crea porque da problemas el acceso al recurso desde la clase estática
CheckDatos
import javax.swing.ImageIcon;
import camiones.*;

public class IconoAjustado extends ImageIcon{
	String path;
	ImageIcon i;

	public IconoAjustado(String p) {
		path=p;
		i = new
ImageIcon(getClass().getResource(DatosGenericos.getDireccion(path)));
	}
	public ImageIcon getIcono() {
		return i;
	}


}
```

CLASE PanelConImagen

```
package auxiliares;

import java.awt.Graphics;
import java.awt.Image;
import javax.swing.ImageIcon;
import javax.swing.JPanel;

public class PanelConImagen extends JPanel {

    private Image imagen;

    public PanelConImagen() {
    }

    public PanelConImagen(String nombreImagen) {
        if (nombreImagen != null) {
            imagen = new ImageIcon(getClass().getResource(nombreImagen)
                           ).getImage();
        }
    }

    public void setImagen(String nombreImagen) {
        if (nombreImagen != null) {
            imagen = new ImageIcon(
                   getClass().getResource(nombreImagen)
                   ).getImage();
        } else {
            imagen = null;
        }

        repaint();
    }

    @Override
    public void paint(Graphics g) {
        if (imagen != null) {
            g.drawImage(imagen, 0, 0, getWidth(), getHeight(),
                              this);

            setOpaque(false);
        } else {
            setOpaque(true);
        }

        super.paint(g);
    }


}
```

CLASE Unidades

```
package auxiliares;

//Esta clase se encarga de hacer la transformacion de las unidades a la hora de
presentarlas en pantalla o de recogerlas en la entrada
//Consideraremos el milimetro y el kilogramo como unidades por defecto
public abstract class Unidades {
	private static String longitud="mm";
	private static String carga="kg";
	private static double conversionCarga=1;
	private static double conversionLongitud=1;

	//Establece Newtons como la unidad por defecto por lo que el factor de
transformacion es 9,8
	public static void Nw() {
		carga="N";
		conversionCarga=1/9.8;
	}

	//Establece Kilos como unidad por defecto por lo que el factor de transformacion
es 1
	public static void Kg() {
		carga="kg";
		conversionCarga=1;
	}

	//Establece centimetros como unidad de longitud, el factor de transformacion desde
el milimetro es 0,1
	public static void cm() {
		longitud="cm";
		conversionLongitud=10;
	}

	//Establece decimetros como unidad de longitud, el factor de transformacion desde
el milimetro es 0,01
	public static void dm() {
		longitud="dm";
		conversionLongitud=100;
	}

	//Establece milimetros como unidad de longitud, el factor de transformacion es
desde el milimetro es 1
	public static void mm() {
		longitud="mm";
		conversionLongitud=1;
	}

	//Establece metros como unidad de longitud, el factor de transformacion es desde
el milimetro es 0,001
	public static void m() {
		longitud="m";
		conversionLongitud=1000;
	}

	//Devuelve las unidades de Longitud actuales
	public static String getUnidadesLongitud() {
		return longitud;
	}

	//Devuelve las unidades de Carga actuales
	public static String getUnidadesCarga() {
		return carga;
	}

	//Devuelve las unidades de Area actuales
	public static String getUnidadesArea() {
```

```
		return longitud+"^2";
	}

	//Devuelve las unidades de Inercia actuales
	public static String getUnidadesInercia() {
		return longitud+"^4";
	}

	//Devuelve las unidades de Tension actuales
	public static String getUnidadesTension() {
		return carga+"/"+longitud+"^2";
	}

	//Devuelve las unidades de Modulo actuales
	public static String getUnidadesModuloResistente() {
		return carga+"/"+longitud+"^3";
	}
	//Devuelve el valor en unidades de momento actuales
	public static double getTensionEnUnidadesActuales(double tension) {
		return (tension/conversionCarga)*Math.pow(conversionLongitud,2);
	}

	//Devuelve el valor en unidades de Area actuales
	public static double getAreaEnUnidadesActuales(double area) {
		return (area/Math.pow(conversionLongitud,2));
	}

	//Devuelve el valor en unidades de Inercia actuales
	public static double getInerciaEnUnidadesActuales(double inercia) {
		return (inercia/Math.pow(conversionLongitud,4));
	}

	//Devuelve el valor en unidades de Modulo Resistente actuales
	public static double getModuloResistenteEnUnidadesActuales(double inercia) {
		return (inercia/Math.pow(conversionLongitud,3));
	}

	//Devuelve el valor en unidades de carga actuales
	public static double getCargaEnUnidadesActuales(double carga) {
		return carga/conversionCarga;
	}

	//Devuelve el valor en unidades de longitud actuales
	public static double getLongitudEnUnidadesActuales(double longitud) {
		return longitud/conversionLongitud;
	}

	public static double getConversionLongitud() {
		return conversionLongitud;
	}

	public static double getConversionCarga() {
		return conversionCarga;
	}

}
```

CLASE VentanaIntroduceNombrePDF

```
/*Esta clase sirve para introducir cargas en el proyecto*/

package auxiliares;

import java.awt.Component;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import java.io.File;
import java.util.ArrayList;
import java.util.Iterator;
import camiones.*;

import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.SpringLayout;

import calculosEstructura.VentanaCalculos;

public class VentanaIntroduceNombrePDF extends JFrame{
	private static final long serialVersionUID = -7418029428404409927L;
	private static int ladopanel=300;
	private static String fondogris=DatosGenericos.getDireccion("Imagenes/fondos/
fondo-gris.png");
	private static String fondoazul=DatosGenericos.getDireccion("Imagenes/fondos/
fondo.jpg");
	private String nombre;
	private static ArrayList<Component> c = new ArrayList<Component>();
	private static ArrayList<JTextField> t = new ArrayList<JTextField>();
	private PanelConImagen general,panelMarca,datosPanel; //Generamos los paneles
contenedores de los datos, el dibujo y la info
	private JLabel lbln,lblTitulo; //Generamos las etiquetas
	private JTextField nombreText; //Generamos los campos de texto para recogida de
los datos
	private JButton btnguardar; //Generamos el boton para calcular
	private static Proyecto proyecto;
	private static VentanaCalculos ventCalculo;
	private static VistaGeneral vistaGeneral;

	public VentanaIntroduceNombrePDF(VistaGeneral vg){
		super("Crear fichero de resultados");
		this.setResizable(false);
		vistaGeneral=vg;

		//Preparamos los PANELES
		//datos panel contiene los elementos de entrada de datos
		datosPanel=new PanelConImagen(fondogris);
		//general contiene los datos de interaccion
		general=new PanelConImagen(fondoazul);
		panelMarca=new PanelConImagen(DatosGenericos.getDireccion("Imagenes/
pantallaprincipal/recurso.png")); //contiene el logo de la aplicación
		panelMarca.setPreferredSize(new Dimension(150,80));


		//Damos formato al PANEL GENERAL
		//Damos el layout para general
```

```
            SpringLayout layout = new SpringLayout();
            //Damos el layout para datosPanel
            SpringLayout layoutDatosPanel = new SpringLayout();
            //Definimos el gap entre ventanas
        int gapX=10;
        int gapY=10;
            datosPanel.setPreferredSize(new Dimension(ladopanel-2*gapX,ladopanel-230));
            Formato.setBorder(datosPanel);
        datosPanel.setVisible(true);
            general.setPreferredSize(new Dimension(ladopanel,ladopanel));
            Formato.setBorder(general);
        general.setVisible(true);


            // Preparamos las ETIQUETAS
        lblTitulo = new JLabel("NOMBRE DEL FICHERO:");
        Formato.setFuenteTituloMediano(lblTitulo);
            lbln = new JLabel("Nombre para el fichero:");


            //Preparamos los TEXTOS
            nombreText = new JTextField("");
            nombreText.setPreferredSize(new Dimension(ladopanel-4*gapX,20));

            //Preparamos los BOTONES DE GUARDADO
            //Generamos los botones
            btnguardar = new JButton("Guardar");
            btnguardar.setPreferredSize(new Dimension(150,30));
            //Generamos los ActionListeners
        ActionListener accionbtnguardar=new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                  guardaImagen();
                  }
      };

      //Asignamos el actionListener a los botones
      btnguardar.addActionListener(accionbtnguardar);


      //Marcamos las RESTRICCIONES DE COLOCACION SPRINGLAYOUT EN EL PANEL DE DATOS
      //Incluimos los elementos en los paneles las restricciones en X
      layoutDatosPanel.putConstraint(SpringLayout.WEST, lbln, gapX, SpringLayout.WEST,
datosPanel);
      //Incluimos los elementos en los paneles las restricciones en Y
      layoutDatosPanel.putConstraint(SpringLayout.NORTH, lbln, gapY, SpringLayout.NORTH,
datosPanel);

      //Incluimos todo en la VENTANA GENERAL.
      //Incluimos los elementos en el panel las restricciones en X
      layout.putConstraint(SpringLayout.WEST, panelMarca, 7*gapX, SpringLayout.WEST,
general);
      layout.putConstraint(SpringLayout.WEST, lblTitulo, 2*gapX, SpringLayout.WEST,
general);
      layout.putConstraint(SpringLayout.WEST, datosPanel, gapX, SpringLayout.WEST,
general);
      layout.putConstraint(SpringLayout.WEST, btnguardar, gapX, SpringLayout.WEST,
general);
      //Incluimos los elementos en el panel las restricciones en Y
      layout.putConstraint(SpringLayout.NORTH, panelMarca, gapY, SpringLayout.NORTH,
general);
      layout.putConstraint(SpringLayout.NORTH, lblTitulo, gapY, SpringLayout.SOUTH,
panelMarca);
      layout.putConstraint(SpringLayout.NORTH, datosPanel, gapY, SpringLayout.SOUTH,
lblTitulo);
      layout.putConstraint(SpringLayout.NORTH, btnguardar, gapY, SpringLayout.SOUTH,
datosPanel);
```

```
       //Marcamos las RESTRICCIONES DE COLOCACION SPRINGLAYOUT EN EL PANEL DE DATOS
       //Incluimos los elementos en los paneles las restricciones en X
       layoutDatosPanel.putConstraint(SpringLayout.WEST, lbln, gapX+10,
SpringLayout.WEST, datosPanel);
       layoutDatosPanel.putConstraint(SpringLayout.WEST, nombreText, gapX,
SpringLayout.WEST, datosPanel);
       //Incluimos los elementos en los paneles las restricciones en Y
       layoutDatosPanel.putConstraint(SpringLayout.NORTH, lbln, gapY, SpringLayout.NORTH,
datosPanel);
       layoutDatosPanel.putConstraint(SpringLayout.NORTH, nombreText, gapY,
SpringLayout.SOUTH, lbln);

       //Metemos los componentes en los PANELES GENERAL y de RECOGIDA DE INFORMACION
       datosPanel.setLayout(layoutDatosPanel);
       datosPanel.add(lbln);
       datosPanel.add(nombreText);
       general.setLayout(layout);
       general.add(panelMarca);
       general.add(lblTitulo);
       general.add(datosPanel);
       general.add(btnguardar);

       //Asignamos un springlayout a la ventana
       SpringLayout layoutventana = new SpringLayout();
       setLayout(layoutventana);
       //Metemos el panel general
       add(general);
       //Le damos las medidas por defecto que tenemos marcado en la variable ladopanel
              setPreferredSize(new Dimension(ladopanel,ladopanel));
              pack();
              this.setLocation(new Point(500,250));
              DatosGenericos.setPosicion(this, ladopanel, ladopanel);
        this.setIconImage((new ImageIcon(DatosGenericos.getDireccion("Imagenes/
pantallaprincipal/recurso.png"))).getImage());
              setVisible(true);

              //damos formato a todos los componentes
              c.add(lbln);
              c.add(btnguardar);
              c.add(nombreText);
              darTipoLetra(c);
       }

    //Este método asigna a todos los componentes de la ventana el mismo tipo de texto
    private void darTipoLetra(ArrayList<Component> c) {
        Iterator<Component> iter = c.iterator();
        while(iter.hasNext()){
              Formato.setFormatoEstandar(iter.next());
        }
    }


    //Devuelve el proyecto para actualizar valores
    public Proyecto getProyecto() {
       return proyecto;
    }

    public void guardaImagen()
    {
       //Cogemos el nombre que hemos introducido en la ventana para el pdf
        nombre=nombreText.getText();
        vistaGeneral.llamaAPdf(nombre);
        this.setVisible(false);
    }

    protected void paintComponent(Graphics g)
    {
```

```
            g.drawRect(50,50,50,50);
        }

}
```

CLASE VentanaProceso

```
package auxiliares;

import java.awt.*;
import javax.swing.*;

public class VentanaProceso extends JFrame{
       private static final long serialVersionUID = 8376298301193580407L;

       //Generamos el constructor de la ventana de ayuda de la aplicacion
       public VentanaProceso(String s) {
              JTextArea jta = new JTextArea();
        jta.setBorder(BorderFactory.createLineBorder(Color.gray));
        jta.setText(s); //Le damos al area de mensajes algo más de espacio para que no
sea solo una linea.
        jta.setEditable(false);
        jta.setForeground(Color.blue);
        jta.setBackground(Color.WHITE);
        //Generamos un scroll para el area e incluimos en el el area de texto
        JScrollPane jtaScroll = new
JScrollPane(jta,JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,JScrollPane.HORIZONTAL_SCROLLBAR_NE
VER);
        this.add(jtaScroll);
        this.setSize(400,200);
        this.setTitle("Contenido Proceso");
        this.setVisible(true);
       }
}
```