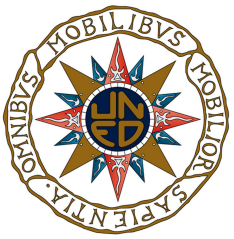


UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

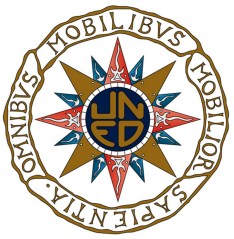
Proyecto de Fin de Grado en Ingeniería Informática

UN SISTEMA DE APOYO A LA GESTIÓN DE AMBULANCIAS EN UN CENTRO URBANO

ROI GÓMEZ LÓPEZ

Dirigido por: Dr. ALFONSO URQUÍA MORALEDA

Curso: 2018/2019 (Septiembre)



UN SISTEMA DE APOYO A LA GESTIÓN DE AMBULANCIAS EN UN CENTRO URBANO

Proyecto de Fin de Grado en Ingeniería Informática
de modalidad *genérica*

Realizado por: ROI GÓMEZ LÓPEZ

Dirigido por: Dr. ALFONSO URQUÍA MORALEDA

Fecha de lectura y defensa: . . . . . .

Resumen

Este Proyecto de Fin de Grado tiene como objetivo la construcción de un sistema de apoyo a la gestión de ambulancias en un centro urbano.

Este sistema, que se puede clasificar como una herramienta CAD (Computer Aided Dispatch), asiste durante las operaciones del personal de un centro de coordinación de emergencias y urgencias, proporcionando los mecanismos de gestión de avisos y control de ambulancias, además de servir como sistema de apoyo a la decisión durante el proceso de asignación de ambulancias.

El subsistema de apoyo a la decisión es una parte central del proyecto y se encarga de proporcionar alternativas de asignación optimizadas en función de los datos disponibles. Entre estos datos se incluyen la prioridad de los avisos, la posición geográfica de las ambulancias, el tiempo estimado de respuesta o la capacidad de actuación en el territorio.

La solución obtenida es un sistema distribuido, que se ha estructurado en varios niveles:

- Varias aplicaciones de cliente que muestran la información disponible de forma gráfica y proporcionan los mecanismos de operación necesarios. Han sido desarrolladas utilizando JavaFX y la información geográfica del proyecto OpenStreetMap.
- Una aplicación de servidor que recibe las peticiones de los clientes y procesa la información de forma centralizada. Ha sido desarrollada utilizando la tecnología Java EE.
- Varios servidores auxiliares que atienden las peticiones de la aplicación de servidor y se encargan de las tareas de almacenamiento y procesamiento externo de datos.

Para la realización del proyecto se han aplicado los conocimientos de Ingeniería del Software adquiridos durante los estudios del Grado de Ingeniería Informática.

Palabras clave

CAD, Ambulancias, Urgencias, Emergencias, Apoyo a la Decisión, Sistema distribuido, GIS, Ingeniería del Software.

Abstract

This Final Degree Project aims to build an ambulance management support system in an urban center.

This system which can be classified as a CAD tool (Computer Aided Dispatch) assists during the operations of the personnel of an urgency and emergency coordination center, providing the mechanisms of management of notices and control of ambulances, in addition to serving as a decision support system during the ambulance assignment process.

The decision support subsystem is a central part of the project, and is responsible for providing optimized allocation alternatives based on available data. These data include the priority of the notices, the geographical position of the ambulances, the estimated response time and the level of acting capacity in the territory.

The built solution is a distributed system, which has been structured on several levels:

- Client applications display the available information graphically and provide the necessary operating mechanisms. They have been developed using JavaFX and the geographic information of the OpenStreetMap project.
- The server application receives the requests from the clients and processes the information centrally. It has been developed using Java EE technology.
- Auxiliary servers handle requests from the server application and handle external storage and data processing tasks.

For the realization of the project the knowledge of Software Engineering acquired during the studies of the Degree of Computer Engineering has been applied.

Keywords

CAD, Ambulance, Urgency, Emergency, Decision Support System, Distributed System, GIS, Software Engineering.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción, objetivos y estructura

1.1. Introducción

Este proyecto puede describirse de forma simplificada como la construcción de un sistema de apoyo a la asignación computerizada, o CAD (*Computer Aided Dispatch*). Un sistema de este tipo facilita la labor de los operadores de un centro de coordinación de recursos al ofrecer posibles soluciones de asignación en función de la información disponible. No es un mecanismo completamente autónomo, siendo el personal humano el que valida y ejecuta cada operación.

Un sistema CAD suele estar compuesto de varios subsistemas, los cuales proporcionan servicios en distintos niveles. Los más importantes son:

- Un conjunto de subsistemas de procesamiento y almacenamiento de datos que permitan una gestión centralizada de la información.
- Un conjunto de subsistemas de control que permitan operar el producto.
- Un conjunto de mecanismos de comunicaciones que permitan el intercambio de la información.

Estos elementos son los típicos de un sistema distribuido sobre una red de ordenadores. Existen muchas posibles soluciones para el desarrollo de un sistema similar, como el modelo cliente-servidor o los mecanismos de invocación de métodos remotos. Las tecnologías disponibles se analizan en el capítulo de estado del arte posterior.

En el contexto del proyecto, el sistema CAD objetivo se centra en la asignación de ambulancias a emergencias médicas en un centro urbano. La principal motivación para la

construcción de un sistema de estas características es la necesidad existente en ciertos territorios con una alta densidad poblacional de gestionar de forma local los servicios de atención sanitaria. El modelo de gestión existente en la mayor parte del territorio español tiene un alcance autonómico, al ser normalmente la administración de cada comunidad quien tiene las competencias en materia de Sanidad, pero no es adecuado en grandes ciudades como Madrid o Barcelona, donde puede ser necesario un sistema propio e independiente del anterior.

Otra motivación importante es la necesidad de modernizar los sistemas de comunicaciones y gestión utilizado en los servicios de emergencias. En la actualidad existen multitud de tecnologías de intercambio de información entre máquinas que permiten el control automático de muchos procesos. El uso de este tipo de mecanismos durante las operaciones de los recursos de emergencias puede ayudar a reducir los tiempos de respuesta de dichos recursos, mejorando con ello los servicios de atención sanitaria.

La construcción de un sistema distribuido es además un reto de diseño, teniendo en cuenta que se deben integrar distintas tecnologías. El uso de sistemas comunicaciones inalámbricos o de posicionamiento global suponen un conjunto de dificultades adicionales, pero para los que es necesario encontrar soluciones factibles.

1.2. Objetivos

El principal objetivo de este proyecto es la construcción de un sistema de apoyo a la gestión y asignación de ambulancias. Este sistema debe permitir:

- La gestión de los avisos de emergencias y urgencias.
- La gestión y el control de las ambulancias.
- La gestión y la optimización de las asignaciones.

La optimización de las asignaciones es uno de los requisitos más importantes. El mecanismo de apoyo a la decisión debe proporcionar para un aviso concreto aquellas alternativas de asignación que considere más adecuadas, en función de los datos disponibles. Para ello es necesario definir la naturaleza de la información que el sistema debe conocer, además de los procesos o criterios de optimización con los que valorar dicha información.

Otro requisito importante es que el sistema debe ser útil a los operadores, no solo en las tareas de optimización, sino también en las de gestión y control. Las aplicaciones

deben mostrar la información de una forma clara y sencilla, facilitando las operaciones y automatizando aquellos procesos que el software sea capaz de gestionar eficientemente.

El producto final debe ser además, lo suficientemente genérico y ampliable como para ser utilizado en distintos territorios. Los modelos de datos y de decisión desarrollados deben ser independientes de la localización de geográfica de la información y adaptables para su despliegue en una zona concreta.

Para la consecución de este proyecto se han planificado las siguientes tareas:

1. Un estudio teórico del problema de asignación de ambulancias a emergencias, de los criterios de optimización y su aplicación como posible solución.
2. Una valoración de las tecnologías más adecuadas para la construcción y despliegue del sistema objetivo.
3. El análisis del sistema objetivo desde un punto de vista conceptual, definiendo las responsabilidades básicas de cada elemento, además de las relaciones entre ellos.
4. El análisis del sistema objetivo desde el punto de vista de los operadores del sistema, definiendo los requisitos, los actores involucrados y las operaciones más importantes.
5. La elaboración de un diseño modular que permita definir la arquitectura de una posible solución.
6. La elaboración de un modelo de pruebas que permita verificar y validar la solución obtenida.
7. La construcción del sistema basándose en los análisis y diseños obtenidos.
8. La evaluación del producto construido mediante la realización de un conjunto de pruebas sobre un escenario real.
9. La escritura de la documentación del proyecto.
10. El despliegue del producto objetivo como un prototipo funcional.

1.3. Estructura

La memoria se ha organizado en siete capítulos y tres anexos:

- El Capítulo 1, Introducción, objetivos y estructura, contiene una breve introducción del problema junto con con las motivaciones para su construcción. A continuación se definen los objetivos y tareas del proyecto. Por último, se realiza una descripción de la estructura de la documentación de entrega.

- El Capítulo 2, Estado del arte, contiene en primer lugar un estudio teórico del problema en el cual se analizan la logística de ambulancias, las posibles variantes de organización del territorio y los modelos algorítmicos utilizados para resolver el problema de asignación. En segundo lugar se valoran las herramientas y tecnologías disponibles para la realización del proyecto. En la última sección se describen varias soluciones similares en funcionamiento o disponibles.

- El Capítulo 3, Análisis, contiene el resultado del análisis del sistema software objetivo. Se divide en las secciones: modelado de negocio y requisitos del sistema, las cuales permiten establecer una especificación inicial que sirva como base para el desarrollo del producto final.

- El Capítulo 4, Diseño, contiene el resultado del diseño realizado a partir de la especificación inicial. Se divide en dos secciones: modelo de diseño y modelo de datos. El modelo de diseño contiene la representación modular de una posible solución de que satisfaga los requisitos del análisis. El modelo de datos permite definir las entidades importantes desde el punto de vista del almacenamiento persistente.

- El Capítulo 5, Implementación y pruebas, describe el trabajo de construcción del sistema. En la primera sección se enumeran las herramientas software y hardware que forman parte del entorno de desarrollo. A continuación se describen las aplicaciones implementadas, poniendo el foco en las tecnologías utilizadas y la funcionalidad obtenida. Por último, se describen los procedimientos de pruebas llevados a cabo para tratar de asegurar la validez del producto obtenido.

- El Capítulo 6, Planificación y costes del proyecto, contiene la planificación temporal del proyecto y un resumen de los costes estimados. Describe la metodología utilizada y las fases en las que se ha organizado el proyecto, mostrando para cada fase sus tareas asociadas y las fechas de realización estimadas.

- El Capítulo 7, Conclusiones y trabajos futuros, contiene las conclusiones obtenidas tras la realización del proyecto. En primer lugar se analizan los objetivos y tareas completadas y a continuación se describen posibles ampliaciones relacionadas con el trabajo realizado.

- El Anexo A, Datos del escenario de pruebas, describe el conjunto de datos utilizados para la creación del escenario de pruebas, incluyendo las fuentes consultadas y los procedimientos de pre-procesado aplicados.
- El Anexo B, Manual de despliegue, describe los procedimientos necesarios para la puesta en marcha del sistema.
- El Anexo C, Manuales de usuario, contiene los manuales para las aplicaciones de usuario desarrolladas, describiendo de forma esquematizada las operaciones más importantes.

El soporte de entrega que acompaña a la memoria se ha organizado en varios subdirectorios:

- El directorio raíz contiene la memoria en formato *pdf*.
- El directorio *datos* contiene los ficheros de datos utilizados para la construcción del escenario de pruebas.
- El directorio *despliegue* contiene los ficheros de configuración utilizados para la puesta en marcha del sistema.
- El directorio *documentación* contiene los manuales de uso y de despliegue.
- El directorio *fuente* contiene el conjunto de módulos y paquetes de código fuente con la última versión del sistema. Se incluyen los guiones de comandos necesarios para una compilación correcta.

Capítulo 2

Estado del arte

2.1. Introducción

En este capítulo se describe el problema de asignación de emergencias y urgencias a ambulancias desde varios puntos de vista.

En la primera Sección, Estudio teórico del problema, se detallan los conceptos básicos, modelos de datos y posibles soluciones desde un punto de vista teórico.

En la segunda Sección, Análisis tecnológico, se evalúan las herramientas y tecnologías disponibles para la construcción del sistema objetivo.

Por último, en la Sección Soluciones existentes se describen varios productos que ofrecen soluciones similares al problema descrito.

2.2. Estudio teórico del problema

2.2.1. Logística de ambulancias

Una ambulancia se puede definir como un recurso movilizable para la atención de una emergencia médica. Este recurso puede ser un vehículo de carretera, un helicóptero o cualquier otro medio de transporte que permita la atención sanitaria y/o el traslado de enfermos o heridos a un centro hospitalario.

La logística de ambulancias se puede describir como el proceso de planificación y ejecución del servicio de resolución de emergencias médicas mediante la movilización de

ambulancias [AP07]. La gestión de dicho servicio se lleva a cabo normalmente en un Centro de Coordinación de Urgencias y Emergencias, en adelante CCUE, y se divide en tres procesos estrechamente relacionados: la creación de un aviso, la asignación de una ambulancia y la resolución de la asignación.

La creación del aviso comienza con la recepción de una llamada al número de emergencias del territorio, la cual puede ser atendida por un operador telefónico externo o del propio CCUE. La principal tarea tras la recepción de la llamada es realizar una valoración de la incidencia, que en los casos donde se requiera atención sanitaria supone determinar las causa médicas y estimar un nivel de prioridad. Tal como se recoge en [UB11] la clasificación más habitual es en tres niveles, de mayor a menor importancia:

- Emergencia.
- Urgencias no demorables.
- Urgencias demorables.

La división de los avisos en prioridades facilita la organización de la demanda y los recursos disponibles. El primer nivel, o nivel de Emergencia, implica un peligro inmediato para la vida del paciente, por lo que las ambulancias disponibles serán primero destinadas a dichos avisos, con la menor demora posible. El segundo y tercer nivel, no presuponen un peligro inmediato a la vida, y su atención podrá ser pospuesta si las circunstancias lo requieren [dS06].

La asignación de una ambulancia comienza tras la valoración del aviso, en el momento que dicho aviso es el de mayor prioridad y más antiguo. Es un proceso realizado por el personal propio del CCUE y se divide en dos fases:

1. La elección y reserva de una ambulancia.
2. La activación de la ambulancia elegida.

La elección de la ambulancia implica determinar cuál de los recursos disponibles es el más adecuado para un atender un aviso concreto. Desde un punto de vista simplificado la mejor solución es la elección de aquella ambulancia cuyo tiempo de llegada al lugar del suceso sea menor, pero también puede ser necesario considerar otros factores, como el impacto que dicha asignación tiene en la capacidad de actuación general. Se definen entonces dos criterios de optimización principales: la minimización del tiempo de respuesta y la maximización de la capacidad de actuación.

El tiempo de respuesta se define como el transcurrido entre la recepción del aviso y la

llegada al lugar del suceso. Depende principalmente de la distancia entre la ambulancia y la localización del aviso pero puede estar influenciado por otros factores como la meteorología o el tráfico.

La capacidad de actuación se puede entender como una estimación de la calidad del servicio de emergencias ante un hipotético nuevo aviso. Esta estimación depende de múltiples factores, siendo los más importantes el número de ambulancias en rango, la densidad de población o la demanda de avisos esperada. Es además un dato dinámico y variable. Es dinámico debido a que la activación de una ambulancia hará bajar la capacidad de actuación alrededor de la base de dicha ambulancia. Es variable ya que distintas regiones del territorio tendrán densidades de población y estimaciones de demanda distintas.

El último proceso, la resolución de la asignación, es una tarea realizada por el personal de la ambulancia y contempla los siguientes estados [dS06]:

1. Activación.
2. Salida.
3. Llegada al lugar.
4. Traslado de paciente.
5. Entrega de paciente en destino.
6. Disponible.

2.2.2. Gestión del territorio

Para facilitar la gestión de recursos y el cálculo de los datos involucrados en los criterios de optimización se plantea la necesidad de dividir el territorio en zonas. En [ROB16] se analizan dos tipos de divisiones: la estructura en red y la estructura en cuadrícula.

La estructura en red consiste en un grafo en el que los nodos representan los puntos centrales de cada zona y las aristas conexiones entre zonas. Cada nodo tiene asociado un valor de peso precalculado en función de la densidad de población y estimación de demanda en la zona asociada. El valor de la arista indica el tiempo de respuesta estimado entre los nodos conectados.

La estructura en cuadrícula divide el plano en cuadrados de igual tamaño. Para cada cuadrícula se almacena un valor de peso similar al caso anterior, además de los tiempos de respuesta estimados a cada una de las demás zonas del mapa.

Esta división por zonas tiene múltiples ventajas. Cada zona tendrá un valor independiente de capacidad de actuación, simplificando los cálculos en los momentos que estos valores tengan que ser actualizados. Además, las zonas pueden mostrarse directamente sobre el mapa, junto con la estimación de capacidad en un instante dado, permitiendo al operador visualizar gráficamente el estado del territorio.

La Figuras 2.1 y 2.2 muestran dos ejemplos de división por zonas centradas en la ciudad de Madrid. A la izquierda puede verse la distribución de las bases de operaciones del SAMUR-Protección Civil y a la derecha las estructuras en red y cuadrícula. Estas divisiones son puramente ilustrativas y no representan datos reales.

De entre las dos opciones presentadas la más adecuada para un territorio urbano es la división en cuadrícula. La estructura en red se adapta mejor para la gestión de un territorio interurbano amplio, en el cual las aristas correspondan de forma aproximada a la forma y distancias de los enlaces por carretera disponibles. La estructura en cuadrícula facilita la división en zonas mucho más pequeñas mejorando la estimación de los tiempos de respuesta.

Para facilitar la definición de las reglas de cálculo y asignación, a partir de este punto se asumen como ciertos los siguientes supuestos:

- El territorio se organiza mediante una estructura en cuadrícula.
- Las zonas están precalculadas.
- Los tiempos de respuesta entre dos zonas cualquiera están precalculados.
- Es posible obtener la zona de un aviso o de una emergencia.
- Es posible obtener el tiempo de respuesta entre una zona de origen y otra zona de destino, siendo este valor dependiente del orden de los parámetros.

2.2.3. Reglas de asignación

Una regla de asignación consiste en un algoritmo que proporciona un conjunto de posibles asignaciones, entendiendo cada asignación como un par aviso-ambulancia. Este algoritmo acepta como entrada un aviso pendiente y devuelve una lista de ambulancias ordenadas según un criterio específico.

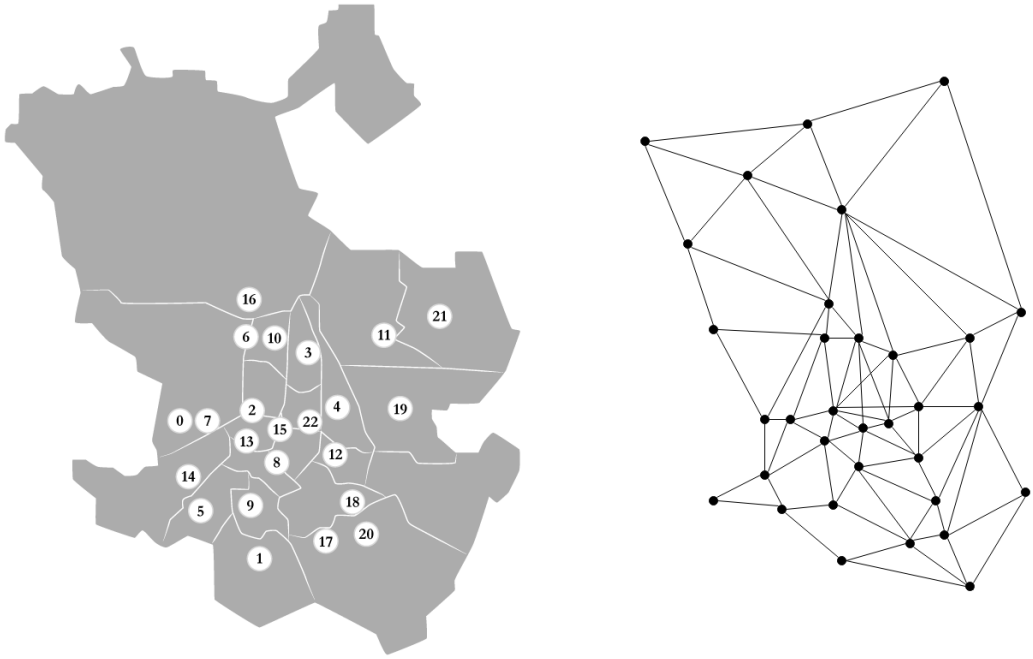


Figura 2.1: División por zonas, estructura en red [ROB16]

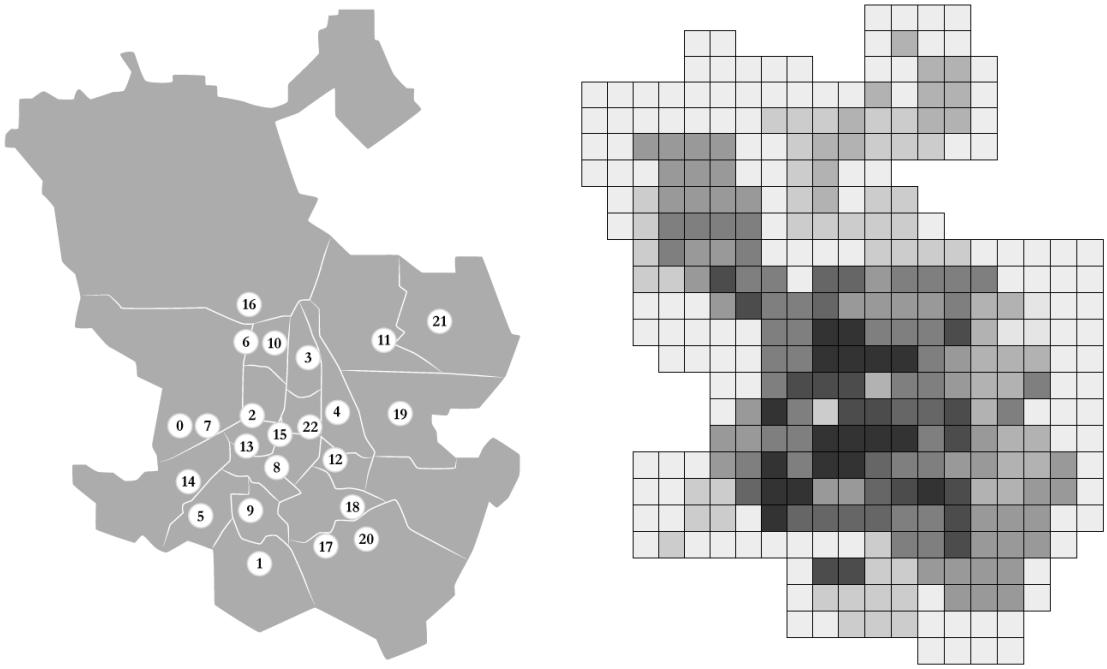


Figura 2.2: División por zonas, estructura en cuadrícula [ROB16]

Antes de la definición de este algoritmo es necesario establecer un conjunto de reglas de valoración de asignaciones, las cuales desarrollan los criterios de optimización introducidos en el Apartado 2.2.1: la minimización del tiempo de respuesta y la maximización de la capacidad de actuación.

Regla de valoración 1: minimización del tiempo de respuesta

El criterio de minimización del tiempo de respuesta valora una asignación en función del tiempo de trayecto estimado entre la ambulancia y el aviso. El Algoritmo 1 en primer lugar obtiene las zonas asociadas a la asignación y a continuación devuelve su tiempo de respuesta.

Algoritmo 1 Minimización del tiempo de respuesta

1: Sea av un aviso
2: Sea am una ambulancia
3: $zo \leftarrow$ obtenerZona(am)
4: $zd \leftarrow$ obtenerZona(av)
5: **return** obtenerTiempoRespuesta(zo, zd)

Regla de valoración 2: Maximización de la capacidad de actuación

El criterio de maximización de la capacidad de actuación valora cada asignación en función del impacto que esta tiene en la capacidad de respuesta resultante de cara a un hipotético nuevo aviso. Para ello, se calcula la capacidad de respuesta que existiría en cada una de las zonas del territorio afectadas en caso de activarse la ambulancia. La valoración para la asignación es el mínimo de entre los valores resultantes. De esta forma, las asignaciones que mantienen una estimación más elevada a lo largo de toda su zona de actuación obtienen una mayor puntuación.

Esta regla de cálculo se basa en el factor *preparedness* introducido en [AP07] y ampliado por [KL16]. Tal como se detalla en estos dos trabajos, el valor *preparedness* se refiere a la habilidad de ofrecer cuidados médicos a los habitantes de un área específica dentro de un tiempo razonable. La expresión utilizada para su cálculo es la siguiente:

$$p_n = \frac{1}{w_n} \sum_{m=1}^{L_n} \frac{\gamma^m}{t_n^m}$$

donde:

- p_n es el valor calculado para la zona n.
- L_n es el número de ambulancias que contribuyen al valor de p_n
- t_n^m es el tiempo de trayecto entre la ambulancia m y la zona n.
- γ^m es un valor contribución de la ambulancia m.
- w_n es un valor de peso que ajusta la contribución de la zona n.

El factor de contribución γ^m es un valor relativo al tiempo de respuesta de la ambulancia, reduciéndose a medida que t_n^m aumenta. En general, deben cumplirse las siguientes reglas:

$$t_n^1 \leq t_n^2 \leq \cdots \leq t_n^{L_n}$$
$$\gamma_n^1 > \gamma_n^2 > \cdots > \gamma_n^{L_n}$$

Una posible solución que cumple con las dos expresiones anteriores es la utilizada en [AP07]:

$$\gamma^m = \frac{1}{2^{m-1}}, \forall m = 1, 2, \ldots, L_n$$

De esta forma, las ambulancias contribuyen en función del orden en el que son consideradas ($\gamma^1 = 1, \gamma^2 = 0{,}5, \gamma^3 = 0{,}25, \ldots$) siendo este orden dependiente del tiempo del respuesta t_n^m.

El factor de peso w_n permite aumentar el nivel de capacidad calculado a medida que la demanda esperada en la zona baja. Para su cálculo pueden utilizarse datos referidos al historial de llamadas, el número de habitantes o la densidad poblacional.

Una posible expresión que calcula w_n en función del valor de densidad de población de la zona d_n y el máximo de todo el territorio D, es la siguiente:

$$w_n = \frac{d_n}{D}$$

La regla completa se muestra en el Algoritmo 2.

Algoritmo 2 Maximización de capacidad de actuación (*preparedness*)

1: Sea aa una ambulancia
2: Sea $z[N]$ un vector con las N zonas del territorio afectadas por al ambulancia
3: Sea $p[N]$ un vector real de dimensión N inicializado a ceros
4:
5: **for** cada zona $z[n]$ con $n = 1, 2, \ldots, N$ **do**
6: Sean $a[L_n]$ las L_n ambulancias que contribuyen a $z[n]$, excluyendo aa y ordenadas por tiempo de respuesta a $z[n]$
7: **for** cada ambulancia $a[m]$ con $m = 1, 2, \ldots, L_n$ **do**
8: $za \leftarrow$ obtenerZona$(a[m])$
9: $r_{mn} \leftarrow$ obtenerTiempoRespuesta(za, z)
10: $p[n] \leftarrow p[n] + \gamma / r_{mn}$
11: **end for**
12: $w_n \leftarrow$ pesoZona$(z[n])$
13: $p[n] \leftarrow p[n]/w_n$
14: **end for**
15: **return** $\min(p)$

Regla de asignación general

Una vez establecidos los criterios de valoración puede definirse la regla de asignación general. Tal como se ha introducido al principio de esta sección una regla de asignación devuelve un listado con las ambulancias disponibles ordenadas según un criterio. La elección del criterio depende en primer lugar del nivel de prioridad del aviso:

- Nivel de Emergencia

  Un aviso con prioridad de Emergencia supone una amenaza inmediata a la vida. El único objetivo en este caso es minimizar el tiempo que se tarda en llegar al lugar del suceso (Algoritmo 1).

- Nivel de Urgencia demorable/no demorable

  Un aviso de Urgencia no supone a priori una amenaza inmediata para la vida, por lo que se considera aceptable el asignar ambulancias que no sean las más próximas al lugar del suceso. La valoración se calcula en función del impacto que la asignación tiene en la capacidad de actuación resultante en el territorio (Algoritmo 2).

  Este mecanismo de valoración tiene un inconveniente. Si no se controla el tiempo máximo permitido para una asignación puede suceder que se sitúen en posiciones elevadas de la lista ambulancias muy alejadas del lugar del aviso. Para evitar esto se define una constante de tiempo de respuesta máxima, RT_{max}, de forma que, si

el tiempo estimado para una asignación es mayor que el dato definido, el valor de capacidad asociado a la ambulancia será fijado a cero.

En el caso hipotético en el que todas las asignaciones estén fuera del rango de tiempo máximo, el algoritmo se comporta como en el primer supuesto, devolviendo las ambulancias ordenadas según el criterio de minimización del tiempo de respuesta.

Esta solución es una variante del algoritmo desarrollado por Tobias Andersson en [AP07]. La principal diferencia radica en que la regla de asignación desarrollada en este proyecto devuelve siempre la lista completa de ambulancias, independientemente del resultado de la valoración. Esto tiene la ventaja de facilitar al operador múltiples opciones, no únicamente la primera alternativa, disponiendo de la mayor cantidad de información posible.

La regla de asignación completa se puede ver en el Algoritmo 3.

Algoritmo 3 Regla de asignación

Sea av un aviso
Sea $am[N]$ un vector con las N ambulancias disponibles
Sea $t[N]$ un vector real de tiempos inicializado a ceros
Sea $c[N]$ un vector real de capacidad de actuación inicializado a ceros

for cada ambulancia $am[n]$ **do**
    $t[n] \leftarrow$ tiempoRespuesta$(am[n], av)$
end for
Ordenar am en función de t ascendente

if prioridad$(av) =$ Emergencia **then**
    return am
else
    for cada ambulancia $am[n]$ **do**
        if $t[n] > T_{max}$ **then**
            $c[n] \leftarrow 0$
        else
            $c[n] \leftarrow$ capacidadActuación$(am[N])$
        end if
    end for
    if $\max(c) > 0$ **then**
        Ordenar am en función de c descendente primero y t ascendente después
    end if
    return am
end if

2.3. Análisis tecnológico

Desde el punto de vista tecnológico el sistema se puede entender como un entorno cliente-servidor sobre una infraestructura de red. Este modelo permite que el trabajo se organice en dos capas, una la de los clientes, realizando peticiones y otra la del servidor, atendiendo las solicitudes y generando respuestas. La comunicación entre las dos capas está gestionada por la red intermedia y facilita la conexión de múltiples elementos de cada capa.

Un diagrama típico de un entorno cliente-servidor se muestra en la Figura 2.3. La correspondencia entre los elementos de dicho modelo y los planificados para este proyecto se puede ver en la Tabla 2.1.

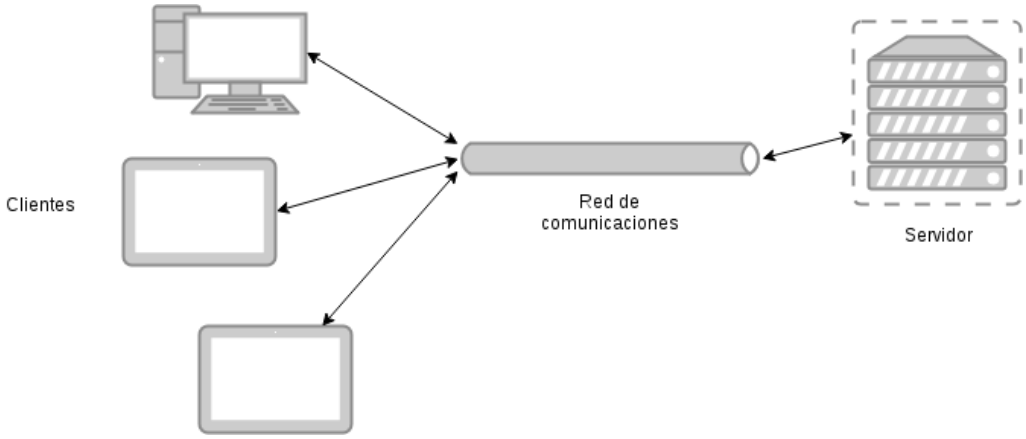


Figura 2.3: Modelo cliente-servidor

Elementos del modelo	Elementos del proyecto
Servidor	Sistema de procesamiento y almacenamiento de datos para el centro de coordinación
Clientes	Aplicaciones de usuario para el centro de coordinación y las ambulancias
Red de comunicaciones	Sistemas de comunicaciones para el centro de coordinación y las ambulancias

Tabla 2.1: Correspondencia entre los elementos del modelo cliente-servidor y los elementos del proyecto

Cada uno de estos elementos es un subsistema relativamente independiente, formado por un número variable de partes software y hardware. Sin embargo, es necesario que el sistema en general trabaje de forma conjunta, por lo que la elección de una tecnología concreta puede implicar limitar las alternativas para otras partes.

En los siguientes apartados se analiza cada subsistema por separado, prestando especial atención a los elementos software, al ser estos el principal objetivo del proyecto.

2.3.1. Sistema de procesamiento y almacenamiento de datos

El sistema de procesamiento y almacenamiento de datos, actuando con el rol de servidor, está formado por distintos componentes, tanto hardware como software, interconectados mediante una red de área local o Intranet.

Los elementos software necesarios son los siguientes:

- Sistema de gestión de bases de datos (SGBD)

  Contiene la lógica de acceso y almacenamiento de datos. Dos de las alternativas más utilizadas son:

  - RDBMS

    Sistema de gestión de bases de datos relacional. Utiliza el lenguaje funcional SQL para la formulación de consultas y almacena los datos en forma de tablas.

  - ODBMS

    Sistema de gestión de bases de datos orientada a objetos. Gestiona de forma transparente los datos como objetos, sin necesidad de capas de traducción en la aplicación.

- Aplicación de servidor

  Contiene la lógica de negocio del sistema. Se encarga del procesamiento de los datos, el acceso al SGBD y la comunicación con los clientes por medio de agentes middleware. Entre las opciones consideradas están:

  - Aplicación PHP

    Un programa informático interpretado que se ejecuta sobre un servidor web. Permite la utilización de clientes ligeros como navegadores web y ofrece un buen nivel de interactividad. Soporta de forma nativa las comunicaciones en Internet utilizando el protocolo HTTP.

  - Aplicación Java EE

    Un conjunto de elementos de software Java, habitualmente llamados Enterprise Java Beans, o EJB, que se ejecutan en un servidor de aplicaciones Java EE.

Permite el uso de distintos tipos de aplicación de cliente y está disponible en múltiples sistemas operativos y dispositivos.

- Middleware de comunicaciones

  Es un elemento software intermedio que contiene la lógica de comunicaciones del sistema. Se encarga de traducir los datos a un formato adecuado para el transporte en red, además de iniciar y recibir las transmisiones. Se utiliza tanto en las aplicaciones de servidor como de cliente y permite que estas se comuniquen de forma transparente independientemente de la arquitectura de red. No es único para todo el proyecto pudiendo utilizarse múltiples alternativas, dependiendo del tipo de comunicación.

  Entre las opciones disponibles se encuentran:

  - JAX-WS o PHP SOAP extension

    Permiten la comunicación con el servidor a través de un servicio web. Son mecanismos simples y eficientes pero solo permiten comunicaciones unidireccionales.

  - Java RMI

    Permite la comunicación entre distintas JVM en un sistema distribuido, tanto en aplicaciones Java SE como en entornos distribuidos Java EE. Soporta el intercambio de objetos, la invocación de métodos remotos y comunicaciones en ambos sentidos.

  - Java JMS

    Permite la comunicación entre distintos dispositivos utilizando un sistema de mensajes organizado en colas y tópicos. Soporta comunicaciones asíncronas, persistentes y bidireccionales. Está altamente integrado con aplicaciones Java de ámbito empresarial y facilita la comunicación con clientes ligeros en redes de Internet móvil utilizando el protocolo MQTT.

2.3.2. Aplicaciones de usuario

Las aplicaciones de usuario son las herramientas utilizadas por los operadores del sistema tanto en el centro de coordinación como en las ambulancias. Se contemplan dos alternativas:

- Cliente web

  Un cliente web es un tipo de aplicación web que se ejecuta del lado del cliente, normalmente sobre un navegador web. Tiene las ventajas de ser independiente del sistema operativo, tener una alta interactividad y no requerir dispositivos de alta capacidad de proceso. Tiene la principal desventaja de soportar únicamente servicios web para las comunicaciones.

- Aplicación Java SE

  Una aplicación Java SE es un programa informático que se ejecuta sobre la Máquina Virtual Java (JVM) en un terminal de escritorio. Tiene las ventajas de ofrecer una alta interactividad, soportar la mayoría de opciones de comunicación y requerir código menos complejo que las alternativas web.

Para la aplicación de usuario de la ambulancia se analiza una alternativa adicional, una aplicación móvil nativa.

- Aplicación móvil nativa

  Una aplicación móvil nativa es una aplicación ejecutada directamente por el sistema operativo del dispositivo móvil. Ofrece un buen rendimiento y un acceso completo a los mecanismos de interacción del dispositivo. Esto permite el uso eficiente de pantallas táctiles, la creación de vistas de tamaño óptimo o la obtención de información de sensores hardware. Tiene los inconvenientes de requerir código específico para cada tipo de dispositivo y limitar las opciones de comunicación.

2.3.3. Sistemas de comunicaciones

Para analizar los sistema de comunicaciones es necesario distinguir entre dos tipos de interacción: las internas, dentro del centro de coordinación y las externas, entre el centro de coordinación y las ambulancias.

En el primer caso la infraestructura de red es la típica de una red de área local o una Intranet. Este tipo de redes suele tener unos niveles de seguridad y fiabilidad óptimos, ademas de integrarse con la gran mayoría de dispositivos y tecnologías existentes.

En el caso de las comunicaciones entre el CCUE y las ambulancias la situación cambia significativamente, debido en gran parte al carácter inalámbrico de los enlaces. La red utilizada debe permitir comunicaciones seguras y fiables, además de proporcionar unos niveles de disponibilidad aceptables [dlR14]. Las opciones contempladas son :

- Internet móvil LTE

  LTE es un estándar de comunicación móvil, también conocido como 4G o cuarta generación de telefonía móvil. Está basado en el protocolo IP, lo cual permite una fácil integración con redes de área local e Intranets y permite una comunicación segura y fiable, en condiciones óptimas, pero puede tener problemas de disponibilidad si el nivel de cobertura desciende o la red telefónica sufre cortes. De todas formas, la cobertura 4G en centros urbanos suele ser aceptable, y se puede mejorar la disponibilidad utilizando dispositivos con conectividad dual o de reserva.

- Radio digital TETRA

  TETRA (TErrestrial Trunked RAdio) es un estándar de comunicaciones, elaborado por el ETSI (Eurpoean Telecomunications Standards Institute), que establece una norma abierta para la construcción de sistemas de comunicaciones vía radio, con alta seguridad y alta disponibilidad. Tiene el inconveniente requerir el uso de una tecnología distinta a la usada en la Intranet, además de una infraestructura de radio digital dedicada.

En todos los casos el software necesario para el uso de la red está repartido entre el sistema operativo del dispositivo y el middleware de comunicaciones.

2.4. Soluciones existentes

En esta sección se analizan tres soluciones existentes, en funcionamiento o disponibles y con ámbitos de actuación distintos: el Centro de Emergencias del 112 de la Región de Murcia, la Central de Comunicaciones 112 de SAMUR-Protección Civil dependiente del Ayuntamiento de Madrid y el sistema de control de flotas de ambulancias eCALLER de la empresa Ingenia.

ECHO: Emergencias Control Holístico Operativo

ECHO es un sistema multiplataforma que permite la gestión integral de las emergencias, desde atender la llamada hasta la gestión de recursos de Mando y Control. Está desarrollado a medida por la empresa TISSAT y es utilizado por el Centro de Emergencias 1-1-2 de la Región de Murcia [112]. Entre sus características se encuentran:

- Gestiona las llamadas y tipifica las incidencias.
- Gestiona la asignación de recursos.
- Hace un uso extensivo de las herramientas GIS, incluyendo la localización geográfica de todos sus recursos y las ubicaciones de incidencias.
- Se integra con otras plataformas operativas de la comunidad, incluida la Unidad Militar de Emergencias.

Desde el punto de vista tecnológico la plataforma ECHO se compone de varios elementos:

- Múltiples servidores y equipos ubicados en el CECARM.
- Un sistema de comunicaciones que utiliza telefonía convencional, telefonía VoIP y tecnologías de radio tanto analógica como digital (TETRA).

Central de comunicaciones 112 en el Ayuntamiento de Madrid

La Central de Comunicaciones 112 [cen] es la ubicación donde se realiza la gestión emergencias de demanda ciudadana para el Ayuntamiento de Madrid. Utiliza un sistema de gestión de recursos de emergencia que tiene como características principales:

- Uso de GIS en los puestos de radio, con acceso a base de datos del callejero.
- Uso del sistema de comunicaciones trunking digital TETRA
- Envío de activaciones desde la Central de Comunicaciones hacia las ambulancias.
- Envío de estados de operatividad desde las ambulancias hacia la Central de Comunicaciones.

Desde el punto de vista tecnológico, el sistema se compone de varios elementos:

- Un sistema digital de interacción de comunicaciones denominado VAS, que integra las líneas telefónicas y los diferentes sistemas de comunicaciones.
- Un enlace de fibra óptica para gestionar la conexión entre la Central de Comunicaciones y la Base Central de SAMUR-Protección Civil.
- Una aplicación de gestión de intervención, desarrollada en Visual Basic e integrada con la red de comunicaciones TETRA. Dispone de dos interfaces: atención telefónica y gestión de recursos.

eCALLER Ambulancias

eCALLER Ambulancias [Ing] es un sistema de control de flotas de ambulancias desarrollado por la empresa Ingenia. Supone una solución integral para los organismos y empresas privadas que presten servicios de emergencias, urgencias y traslados de pacientes críticos. Se ha implantado en los centros coordinadores del 061 de la Empresa Pública de Emergencias Sanitarias dependiente de la Junta de Andalucía.

El sistema permite disponer a los centros coordinadores de urgencias y emergencias, en tiempo real, de información sobre el estado de cada ambulancia, al tiempo que le facilita la transmisión online de los datos de cada caso a la unidad que es asignada para realizarlo. Además, el sistema registra automáticamente los tiempos de salida y llegada que cursa cada ambulancia hacia el centro coordinador. Está desarrollado siguiendo una arquitectura modular:

- Módulo de Gestión del Servicio

  Permite la gestión de los servicios de creación, asignación, seguimiento, cierre, etc. Muestra capas de información geográficas configurables y, funcionalidades derivadas de ellas (GPS, ubicación de unidades, histórico de rutas, etc).

- Módulo de terminal embarcado ligero

  Solución basada en tableta táctil Android, que ayuda a la navegación mediante voz. La comunicación con el Centro de Control de Flota (CCF) se realiza a través de GRPS/3G.

2.5. Conclusiones

El problema de la asignación de ambulancias requiere tener en cuenta múltiples factores. Los procedimientos de actuación de los centros de coordinación, la organización del territorio, la localización de cada ambulancia o la demanda esperada de cada zona son elementos cruciales a la hora tomar una decisión sobre cual de los recursos destinar a un aviso concreto.

La propuesta de solución se ha centrado en la optimización del tiempo de respuesta y la capacidad de actuación en las zonas alrededor de la ambulancia. Esto permite asegurar que los tiempos de atención médica son adecuados, sin descuidar la capacidad de atender nuevos avisos.

El análisis tecnológico ha permitido evaluar la estructura y tecnologías más apropiadas para la construcción del sistema objetivo.

La infraestructura elegida se ha basado en el modelo cliente-servidor, con sus elementos organizados en niveles:

- Varias aplicaciones de usuario actuando como los clientes del sistema.
- Una aplicación de servidor que se encarga del procesamiento y almacenamiento de datos.
- Un conjunto de sistemas de comunicaciones, que permiten el intercambio de información entre los distintos componentes.

La evaluación de las alternativas tecnológicas se ha enfocado en dar respuesta al carácter distribuido del sistema. Herramientas como Java EE, la invocación de métodos remotos o los Servicios Web proporcionan soluciones factibles para las necesidades de comunicación de las distintas aplicaciones.

Capítulo 3

Análisis

3.1. Introducción

En este capítulo se describe el análisis llevado a cabo, a partir del estudio teórico de la Sección 2.2, para obtener una especificación inicial del software objetivo. Esta especificación permite establecer qué debe hacer el sistema y cómo se va a utilizar desde el punto de vista de los actores involucrados.

Contiene las Secciones Modelado de negocio y Análisis de requisitos.

3.2. Modelado de negocio

El modelado de negocio permite identificar los elementos del dominio del problema. Tiene como resultado una lista de clases conceptuales y un diagrama de modelo de dominio, incluyendo los atributos de cada clase y sus relaciones con otros conceptos.

3.2.1. Clases conceptuales

- **CC1:** CCUE

  Descripción Representa el Centro de Coordinación de Urgencias y Emergencias. Permite gestionar la información y las comunicaciones.

- **CC2:** Aviso

  Descripción Representa cada una de las incidencias recibidas en el CCUE.

  Atributos

  - *Fecha de recepción*: contiene el instante de tiempo en que el operador crea el aviso.
  - *Prioridad*: determina la importancia del aviso.:
    - *Prioridad 1*: emergencias.
    - *Prioridad 2*: urgencias no demorables.
    - *Prioridad 3*: urgencias demorables.
  - *Estado*:
    - *Estado 0*: válido.
    - *Estado 1*: cancelado.
  - *Causa médica*: información médica sobre el aviso.
  - *Dirección*: una dirección o nombre de un lugar.
  - *Localización geográfica*: un conjunto de coordenadas geográficas.

- **CC3:** Ambulancia

  Descripción Representa cada uno de los recursos asignables a los avisos.

  Atributos

  - *Clase*: determina las características técnicas de la ambulancia [BOE]:
    - *A1* y *A2*: de transporte no urgente.
    - *B*: con soporte vital básico.
    - *C*: con soporte vital avanzado.
  - *Estado*: determina la capacidad de actuación y actividad de cada ambulancia:
    - *Estado 0*: inactiva.
    - *Estado 1*: activación.
    - *Estado 2*: salida.
    - *Estado 3*: llegada al lugar.
    - *Estado 4*: traslado de paciente.
    - *Estado 5*: entrega de paciente.
    - *Estado 6*: disponible.
  - *Localización geográfica*: un conjunto de coordenadas.

- **CC4:** Base de ambulancia

  Descripción Cada una de las localizaciones donde se pueden establecer ambulancias.

  Atributos

  - *Nombre*: un identificador conocido de la base.
  - *Dirección*: una dirección o nombre de un lugar.
  - *Localización geográfica*: un conjunto de coordenadas geográficas.

- **CC5:** Asignación

  Descripción Representa el elemento de gestión logística que relaciona avisos y ambulancias.

  Atributos

  - *Tiempo de respuesta*: estimación de tiempo entre la activación de la ambulancia y la llegada al lugar del suceso.
  - *Fecha de activación*: tiempo en que se activa la ambulancia.
  - *Fecha de finalización*: tiempo en que se termina la ejecución de la asignación
  - *Estado*: se definen tres valores:
    - *Estado 1*: pendiente.
    - *Estado 2*: registrada.
    - *Estado 3*: activa.
    - *Estado 4*: finalizada.

- **CC6:** Zona de actuación

  Descripción Cada una de las parcelas del territorio en las que se puede generar un aviso.

  Atributos

  - *Límites geográficos*: posiciones de cada uno de los límites de la zona.
  - *Capacidad de actuación*: estimación de la calidad de una actuación para un hipotético aviso en dicha zona. Depende de factores como la densidad de población, las ambulancias en rango, etc.
  - *Peso*: factor que establece la contribución de la zona a la capacidad de actuación.

- **CC7:** Operador del CCUE

  Descripción Personal de gestión del centro de coordinación. Gestiona las ambulancias y las asignaciones.

- **CC8:** Operador de ambulancia

  Descripción Personal de ambulancia. Se encarga del control de la ambulancia.

- **CC9:** Operador telefónico

  Descripción Personal encargado de atender las llamadas directamente con los afectados y transformar la información recibida a avisos utilizables por el sistema. Puede operar directamente desde el CCUE o desde un centro de coordinación externo.

3.2.2. Diagramas del modelo de dominio

Las Figuras 3.1 y 3.2 muestran los diagramas del modelo de dominio, en español e inglés respectivamente. La versión en inglés sirve de puente entre el análisis del problema, cuyos conceptos se describen en castellano y el diseño, donde las clases son nombradas en inglés.

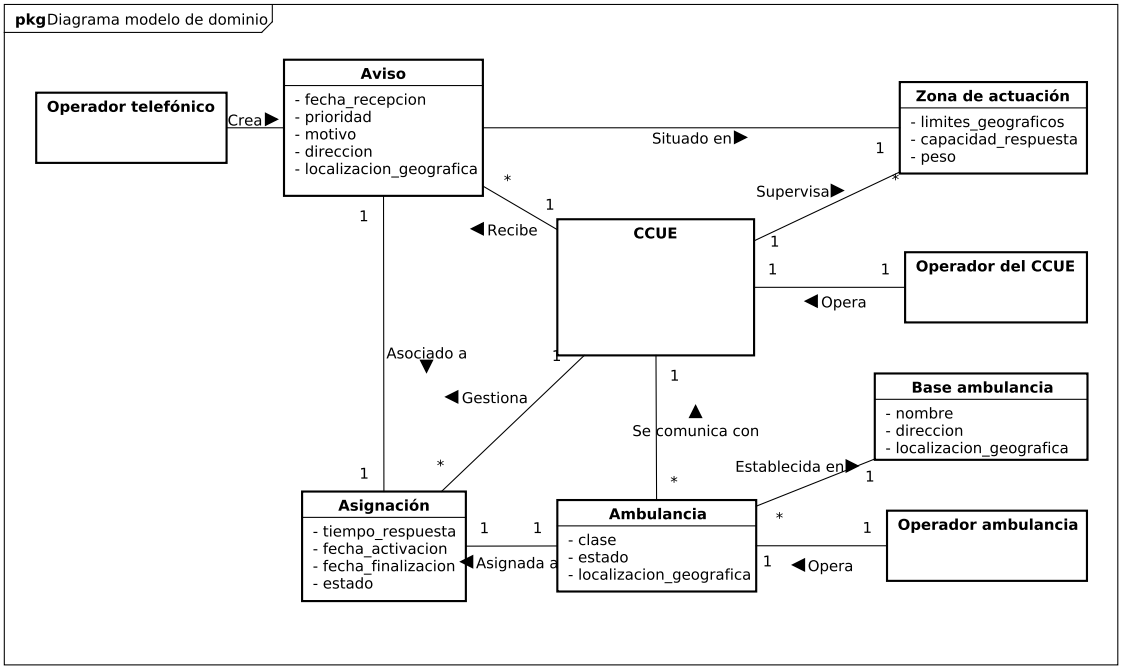


Figura 3.1: Diagrama del modelo de dominio

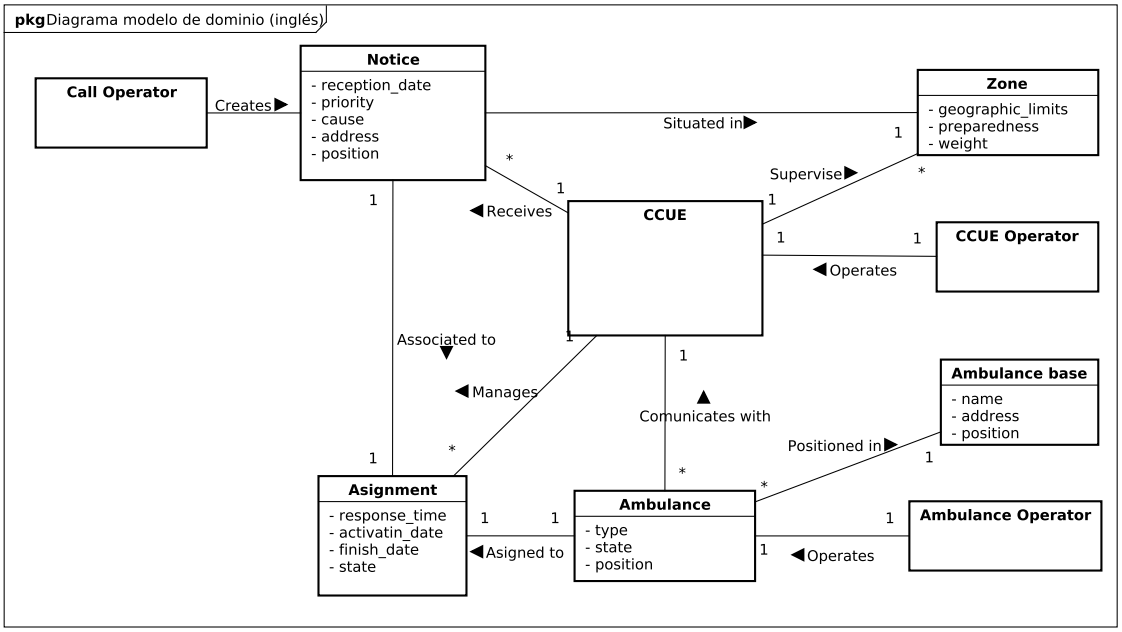


Figura 3.2: Diagrama del modelo de dominio (inglés)

3.3. Análisis de requisitos

El análisis de requisitos permite establecer de una forma sencilla las capacidades y condiciones con las cuales debe ser conforme el sistema [LV03].

En esta sección se muestra el resultado de dicho análisis, especificando en primer lugar la lista de requisitos funcionales y no funcionales. A continuación se definen los actores que interactúan con el sistema y por último, se detallan los casos de uso que desarrollan los requisitos funcionales.

3.3.1. Requisitos del sistema

Requisitos funcionales

Los requisitos funcionales permiten dar una idea inicial del comportamiento del sistema.

- **RF1:** Gestión de avisos.
    1. El sistema debe permitir el registro de avisos nuevos.
    2. El sistema debe permitir la cancelación de avisos ya registrados.
    3. Un aviso registrado debe contener una valoración de la prioridad.
    4. Un aviso registrado debe contener una localización geográfica válida.
    5. La creación de un aviso nuevo debe provocar la creación automática de una asignación pendiente.
    6. La cancelación de un aviso debe provocar la cancelación automática de cualquier asignación asociada.
- **RF2:** Gestión de ambulancias.
    1. El sistema debe permitir el registro de nuevas ambulancias.
    2. El sistema debe permitir la modificación y la eliminación de ambulancias.
    3. El sistema debe permitir el cambio de estado de la ambulancia.
    4. Las ambulancias deben estar posicionadas geográficamente.
    5. Las ambulancias deben estar autenticadas mediante una clave única.
- **RF3:** Gestión de asignaciones.
    1. El sistema debe permitir el registro de asignaciones pendientes.
    2. El sistema debe permitir la cancelación de asignaciones registradas.
    3. El sistema debe facilitar alternativas de asignación de ambulancias a avisos.

4. El sistema debe permitir la ejecución de una asignación.
5. El sistema debe comunicar las asignaciones a las ambulancias tras el registro.

Quedan fuera de los límites del sistema las operaciones de gestión de las zonas y las bases de ambulancias, al considerarse que dichas tareas forman parte del despliegue y no del funcionamiento ordinario.

Requisitos no funcionales

Los requisitos no funcionales se pueden entender como aquellos que no definen acciones específicas que el sistema deber realizar, sino condiciones relativas al rendimiento, la facilidad de uso o la seguridad.

- **RNF1:** Utilizar mecanismos de comunicaciones seguros y fiables.
- **RNF2:** Utilizar sistemas de información geográfica (GIS) con posicionamiento de ambulancias y avisos.
- **RNF3:** Utilizar estilos visuales claros para los distintos tipos de elementos: tipo de ambulancia, tipo de aviso, etc.
- **RNF4:** Utilizar interfaces de usuario para las ambulancias adaptadas a dispositivos táctiles de tamaño reducido.

3.3.2. Actores

- **ACT1:** Operador telefónico.

  Los operadores telefónicos son aquellos encargados de la gestión de los avisos. Su principal tarea es la creación y valoración de las incidencias recibidas.

- **ACT2:** Operador del CCUE.

  Los operadores del CCUE son aquellos encargados de la gestión de las asignaciones y las ambulancias desde el centro de coordinación. Su principal tarea es procesar las asignaciones pendientes, asociando ambulancias a avisos.

- **ACT3:** Operador de ambulancia.

  Los operadores de ambulancia son aquellos encargados del control de cada ambulancia. Su principal tarea es la comunicación de los cambios de estado de la ambulancia durante la ejecución de una asignación.

3.3.3. Casos de uso

La Figura 3.3 muestra el diagrama de casos de uso resultado del análisis de los requisitos funcionales. Para cada caso de uso se definen los actores involucrados, un escenario principal de éxito y uno o varios escenarios secundarios, si los hubiera.

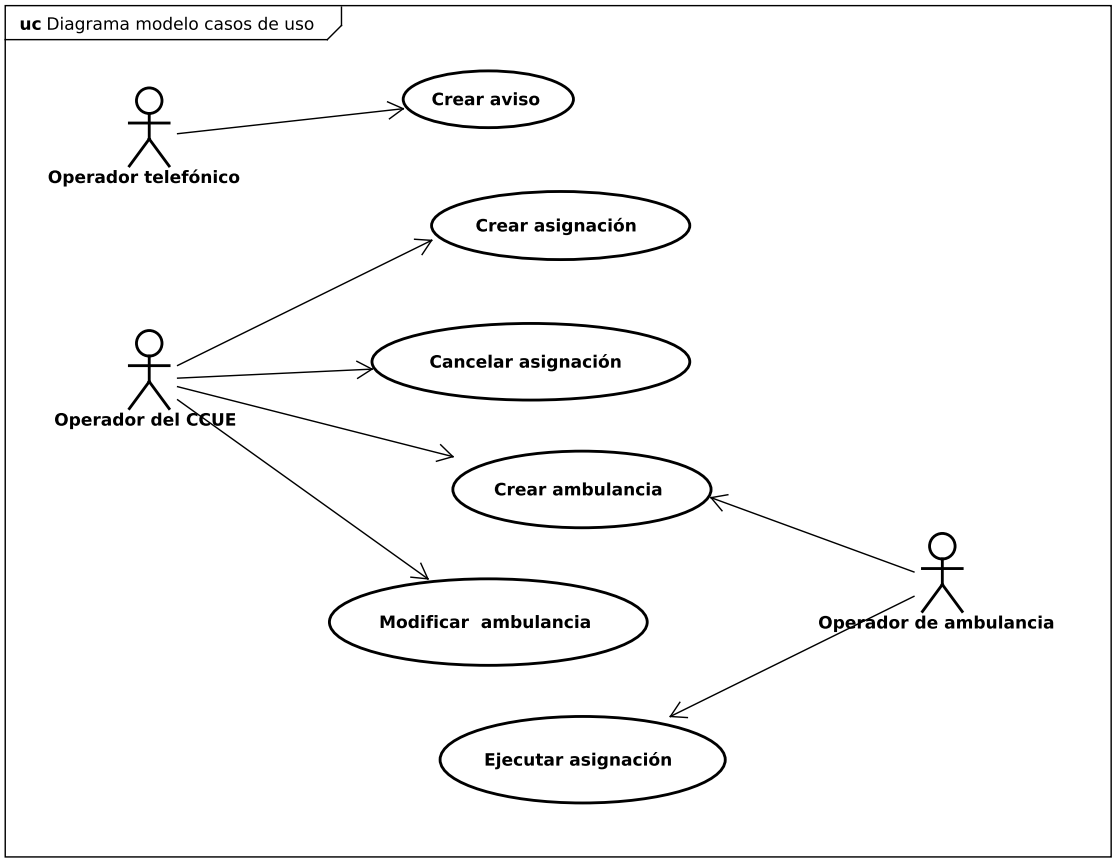


Figura 3.3: Diagrama del modelo de casos de uso

- **CU1:** Crear aviso

  Actores Operador telefónico.

  Escenario principal

  1. El caso de uso comienza cuando el operador telefónico inicia la creación de un nuevo aviso.
  2. El sistema solicita la información del aviso.
  3. El operador introduce cada uno de los datos requeridos.

4. El operador solicita obtener la posición geográfica de la dirección introducida.
5. El sistema devuelve la localización geográfica en forma de coordenadas.
6. El operador añade la localización como dato al aviso.
7. El operador solicita el registro del aviso en el sistema.
8. El sistema verifica los datos del aviso y lo registra como válido.
9. El sistema crea una asignación asociada al aviso nuevo con estado pendiente.
10. El operador cierra el aviso abierto.

Escenarios alternativos

8a. El sistema detecta que el dato de valoración de prioridad no está cubierto.
- El sistema muestra un error avisando de la ausencia del dato requerido y el fallo en el registro.

8b. El sistema detecta que el dato de posición geográfica no está cubierto.
- El sistema muestra un error avisando de la ausencia del dato requerido y el fallo en el registro.

10. El operador solicita la cancelación del aviso ya registrado.
- El sistema actualiza el aviso como cancelado.
- El sistema verifica si existe alguna asignación en curso asociada al aviso y la actualiza como cancelada.

- **CU2:** Crear asignación

Actores Operador del CCUE, Operador de ambulancia.

Escenario principal

1. El caso de uso comienza cuando el operador del CCUE selecciona el aviso pendiente de mayor prioridad y más antiguo.
2. El operador del CCUE solicita al sistema de apoyo a la decisión una valoración del aviso.
3. El sistema de apoyo proporciona una lista de posibles asignaciones ordenadas según un criterio.
4. El operador del CCUE selecciona una asignación y solicita su registro.
5. El sistema registra la asignación y envía un mensaje de asignación registrada a la ambulancia asociada.

- **CU3:** Cancelar asignación

  Actores Operador del CCUE, Operador de ambulancia.

  Escenario principal

  1. El caso de uso comienza cuando el operador del CCUE desea cancelar una asignación ya registrada.
  2. El operador del CCUE solicita la cancelación de la asignación.
  3. El sistema registra la cancelación de la asignación, envía un mensaje de cancelación a la ambulancia asignada
  4. El sistema crea una nueva asignación pendiente asociada al mismo aviso.
  5. El operador de la ambulancia recibe la cancelación y solicita el cambio de estado a disponible.
  6. El sistema registra el cambio de estado de la ambulancia.

- **CU4:** Crear ambulancia

  Actores Operador del CCUE, Operador de ambulancia.

  Escenario principal

  1. El caso de uso comienza cuando el operador del CCUE inicia la creación de una ambulancia nueva.
  2. El sistema solicita los datos de la nueva ambulancia.
  3. El operador del CCUE introduce los datos solicitados.
  4. El operador del CCUE solicita el registro de la ambulancia.
  5. El sistema registra la ambulancia y devuelve un elemento de datos de autenticación único.
  6. El operador del CCUE comunica, por un medio alternativo al sistema actual, los datos de autenticación al operador de la ambulancia.
  7. El operador de ambulancia introduce los datos de autenticación recibidos.
  8. El sistema confirma la autenticación, devuelve a la ambulancia el identificador asignado.
  9. El operador del CCUE cierra la ambulancia abierta.

  Escenarios alternativos

  8a. El sistema detecta que los datos de autenticación no son válidos.
  - El sistema solicita la reinserción de la información de autenticación.

- **CU5:** Modificar ambulancia

  Actores Operador del CCUE.

  Escenario principal

  1. El caso de uso comienza cuando el operador del CCUE inicia la modificación de una ambulancia ya creada.
  2. El operador del CCUE solicita al sistema los datos de la ambulancia.
  3. El sistema devuelve los datos de la ambulancia existente.
  4. El operador del CCUE introduce los valores nuevos de cada propiedad a actualizar.
  5. El operador del CCUE solicita al sistema la actualización de la ambulancia actualizada.
  6. El sistema actualiza los datos de la ambulancia.
  7. El operador del CCUE cierra la ambulancia abierta.

  Escenarios alternativos 4-6. El operador solicita la eliminación de la ambulancia.
  - El sistema elimina la ambulancia del sistema.

- **CU6:** Ejecutar asignación

  Actores Operador de ambulancia

  Escenario principal

  1. El caso de uso comienza cuando el operador de la ambulancia recibe un mensaje de asignación registrada.
  2. El operador de la ambulancia solicita la activación de la asignación.
  3. El sistema establece el estado de la ambulancia a *activación*, el estado de la asignación a *activa* y registra la hora de activación.
  4. El operador de la ambulancia comunica al sistema que se ha producido la puesta en marcha.
  5. El sistema establece el estado de la ambulancia a *salida*.
  6. El operador de la ambulancia comunica al sistema la llegada al lugar del suceso.
  7. El sistema establece el estado de la ambulancia a *llegada al lugar* y registra la hora de llegada.
  8. El operador de la ambulancia comunica al sistema que inicia el traslado del paciente.

9. El sistema establece el estado de la ambulancia a *traslado de paciente*.
10. El operador de la ambulancia comunica al sistema que ha realizado la entrega del paciente y finalizado la asignación.
11. El sistema establece el estado de la ambulancia a *entrega de paciente*, de la asignación a *finalizada* y registra la hora de finalización.

Escenarios alternativos

4-5. El operador de la ambulancia solicita la cancelación de la asignación y su estado a no operativa.
   - El sistema establece el estado de la asignación como *cancelada*, el estado de la ambulancia a *inactiva*.
   - El sistema crea una nueva asignación pendiente asociada al aviso de la asignación cancelada.

3.4. Conclusiones

En este capítulo se han presentado los elementos más importantes resultado del proceso de análisis del software objetivo:

- Modelo de dominio: Define las clases conceptuales más importantes del dominio del problema, junto con sus atributos y relaciones.
- Requisitos del sistema: Establecen las capacidades y condiciones con las cuales debe ser conforme el sistema.
- Actores del sistema: Representan los elementos del sistema con algún objetivo definido.
- Casos de uso: Desarrollan los requisitos funcionales relacionando los objetivos con los eventos del sistema.

Estos documentos permiten establecer una descripción aproximada del sistema y son utilizados como base para el resto de elementos desarrollados durante el proyecto.

Capítulo 4

Diseño

4.1. Introducción

En este capítulo se describe el resultado del trabajo de diseño del sistema. Para su realización se han tenido en cuenta los siguientes criterios:

1. Dar respuesta a las necesidades lógicas del sistema, sin que el resultado sea necesariamente ejecutable.
2. Realizar una división en módulos y paquetes en función de las responsabilidades encontradas.
3. Maximizar la cohesión dentro de cada módulo, buscando siempre un nivel de acoplamiento bajo.
4. Maximizar la reutilización de elementos comunes utilizando en la medida de lo posible elementos genéricos y patrones conocidos.
5. Orientar la arquitectura a una estructura distribuida, en la que los módulos de cliente y los módulos de servicio se comunican entre si.

Contiene las Secciones Modelo de diseño y Modelo de datos.

4.2. Modelo de diseño

El modelo de diseño consiste en un conjunto de diagramas y elementos documentales que permiten definir una posible solución que satisfaga los requisitos. Para su construcción, se

ha partido de los casos de uso, transformándolos en diagramas de secuencia del sistema, o DSS. A continuación, se han desarrollado las operaciones definidas en los DSS en forma de clases y métodos, organizando dichos elementos en módulos y paquetes según sus responsabilidades.

Se ha utilizado principalmente el inglés para la denominación de los objetos y los mensajes. Esto facilita la transformación del diseño en código ejecutable durante la fase de construcción. Los nombres de los casos de uso se mantienen en español.

4.2.1. Diagramas de secuencia del sistema

Las Figuras 4.1, 4.2, 4.3, 4.4, 4.5 y 4.6 muestran los DSS creados a partir de los casos de uso definidos en el Apartado 3.3.3.

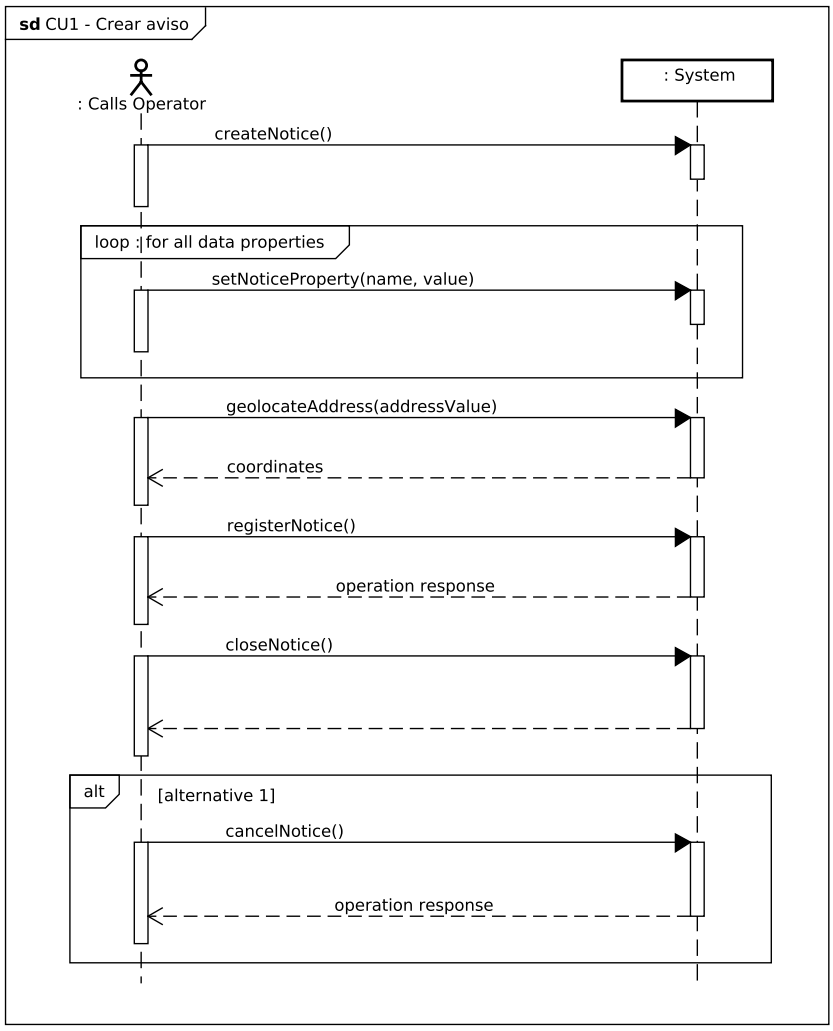


Figura 4.1: Diagrama de secuencia del sistema CU1: Crear aviso

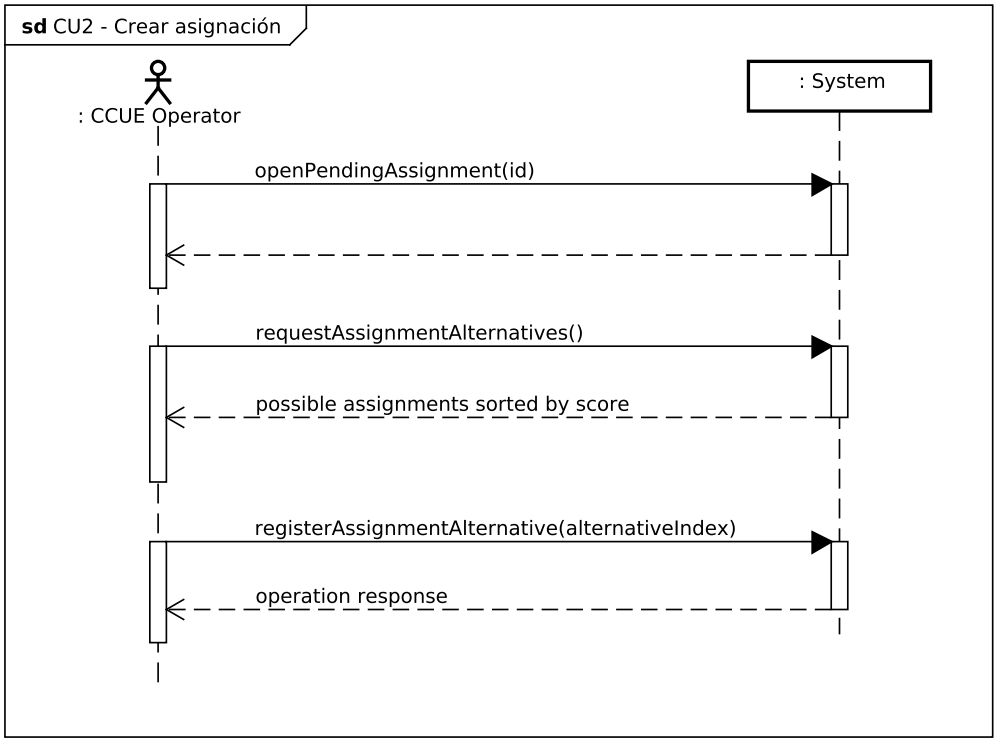


Figura 4.2: Diagrama de secuencia del sistema CU2: Crear asignación

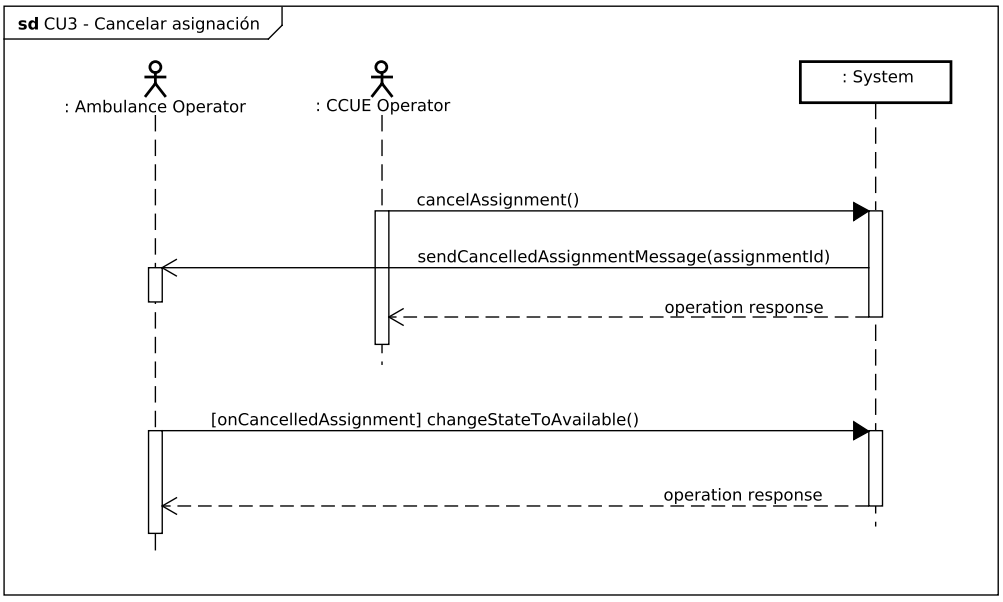


Figura 4.3: Diagrama de secuencia del sistema CU3: Cancelar asignacion

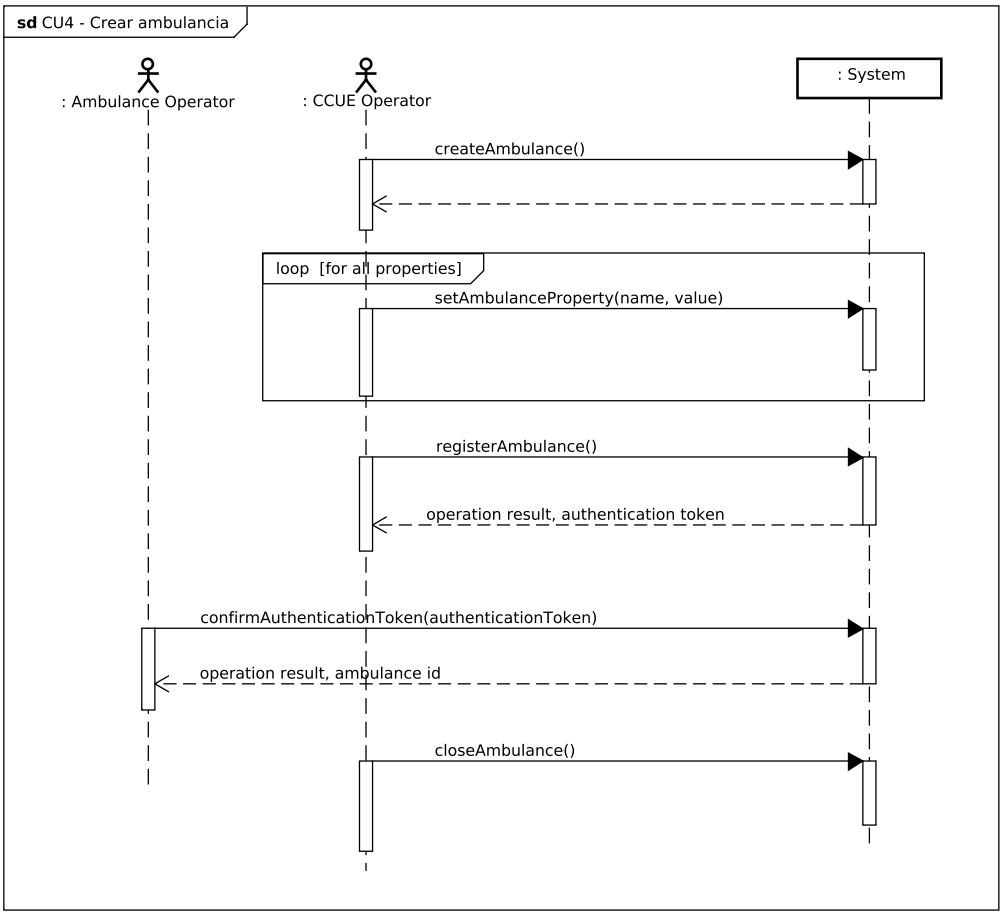


Figura 4.4: Diagrama de secuencia del sistema CU4: Crear ambulancia

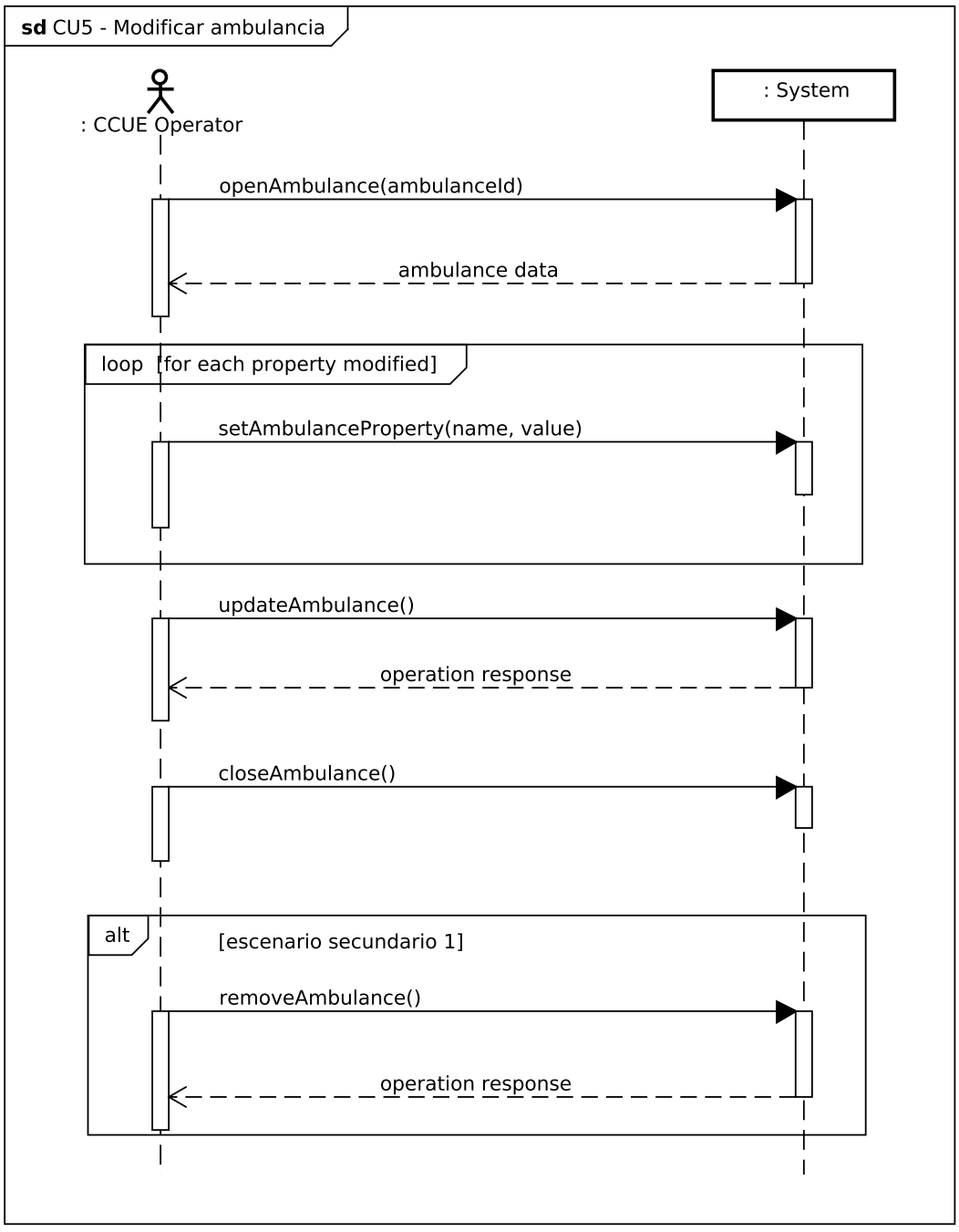


Figura 4.5: Diagrama de secuencia del sistema CU5: Modificar ambulancia

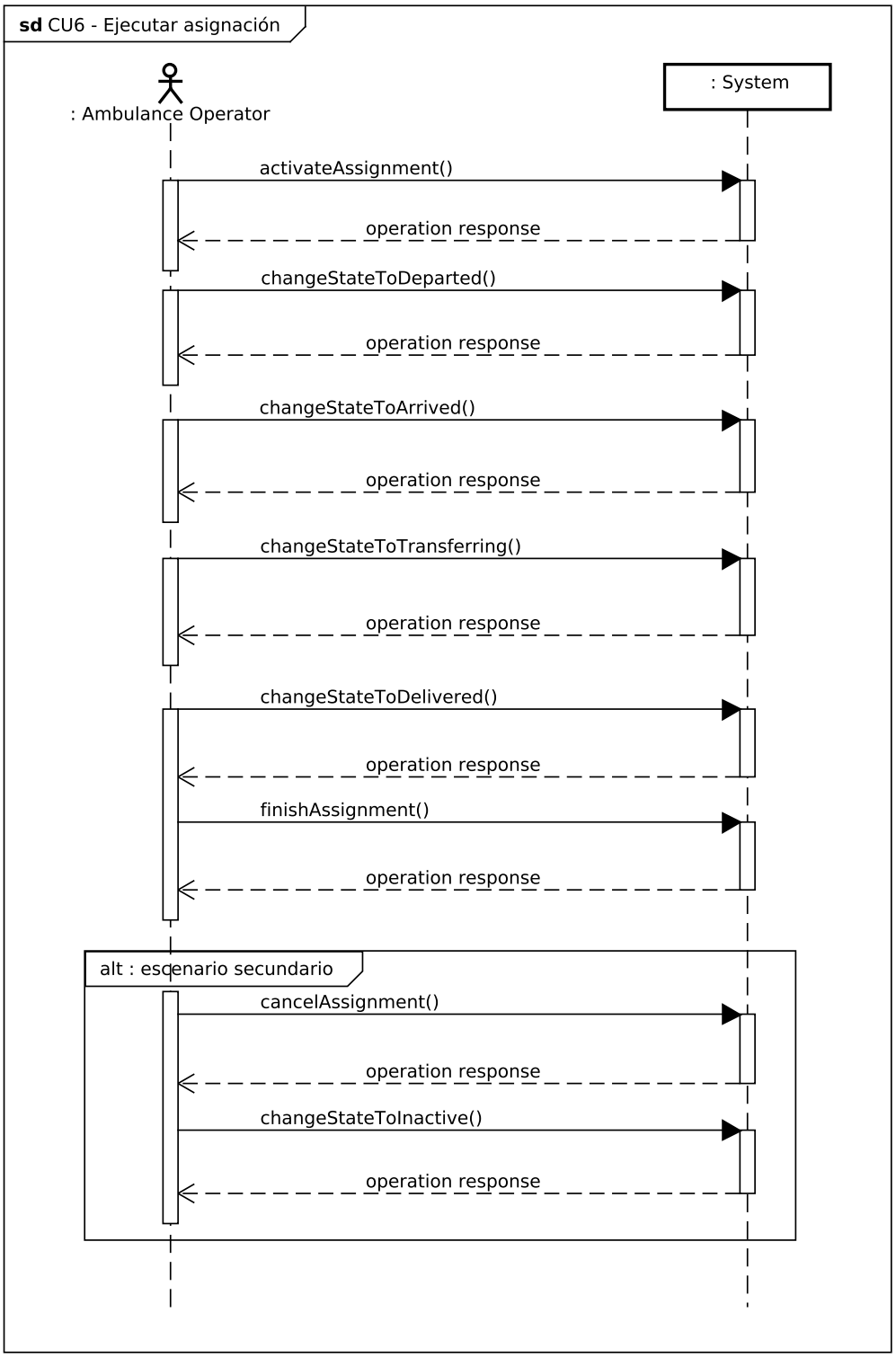


Figura 4.6: Diagrama de secuencia del sistema CU6: Ejecutar asignación

4.2.2. Arquitectura lógica

Craig Larman introduce en [LV03] el término arquitectura lógica como una de las dimensiones de la arquitectura del software: "*describe el sistema en términos de su organización conceptual en capas, paquetes, frameworks importantes, clases, interfaces y subsistemas*".

Este apartado trata de detallar la arquitectura resultante del proceso de diseño, centrándose en su dimensión lógica. El primer nivel está organizado en cuatro módulos:

- Módulo ***server***.

  Contiene la lógica de negocio del sistema y los mecanismos que permiten la comunicación entre el servidor y las aplicaciones de cliente. Está organizada en varios paquetes:

  - Paquetes internos: *persistence*, *messaging* y *remote*.
  - Paquetes de servicio: *notices*, *ambulances*, *assignments*, *alternatives*, *gis* y *zones*.

- Módulo ***client-notices***.

  Contiene la lógica de la aplicación de cliente para la gestión de avisos. Desarrolla el DSS de la Figura 4.1.

- Módulo ***client-ccue***.

  Contiene la lógica de la aplicación de cliente encargada de la gestión del centro de coordinación. Se divide a su vez dos paquetes: *assignments* y *ambulances*, con las operaciones de las asignaciones y las ambulancias respectivamente. Desarrolla los DSS de las Figuras 4.2, 4.3, 4.4 y 4.5.

- Módulo ***client-ambulance***.

  Contiene la lógica de la aplicación de cliente encargada del control de la ambulancia. Desarrolla por completo el DSS de la Figura 4.6 y parte de las operaciones del DSS de la Figura 4.4.

La Figura 4.7 muestra un diagrama de componentes simplificado, junto con las dependencias entre paquetes más importantes.

En los siguientes sub-apartados se desarrollan cada uno de los componentes de este diagrama. Para cada módulo definido se enumeran los paquetes y las clases que lo forman, explicando en cada caso sus objetivos o responsabilidades.

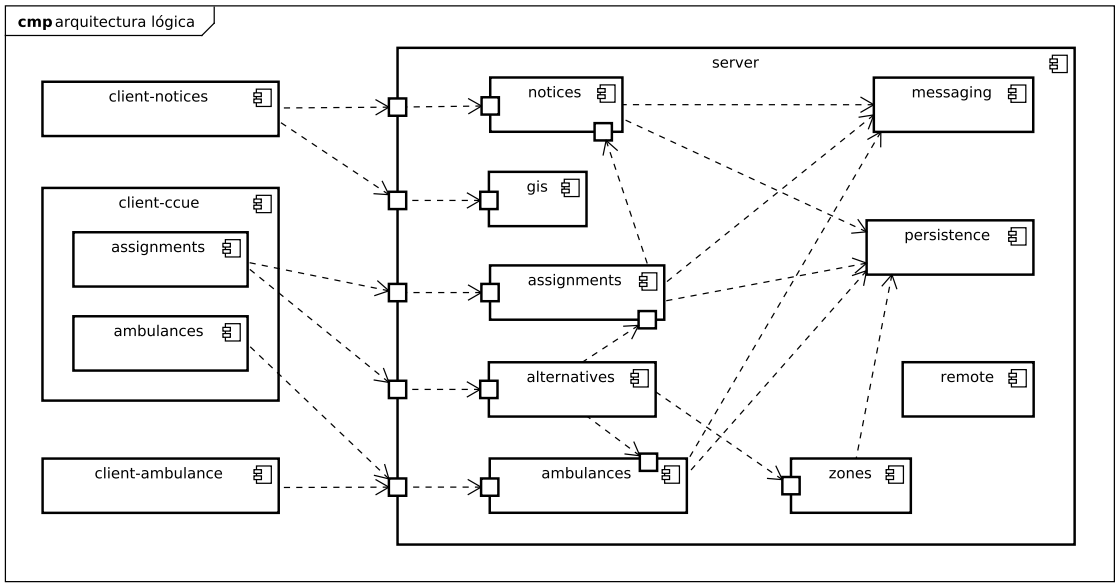


Figura 4.7: Diagrama de componentes: arquitectura lógica

Paquete *server.persistence*

El paquete *persistence* contiene las clases que permiten definir los mecanismos de acceso al sistema de base de datos.

Se ha diseñado siguiendo el patrón de acceso a objetos de datos (DAO *pattern*), el cual define una abstracción genérica base sobre la que se declaran las operaciones comunes para todas las entidades del sistema. Mediante este patrón, para cada tipo de entidad registrada en el sistema se especifica una clase derivada y sus reglas de persistencia, las cuales pueden ser dependientes del sistema que realice el almacenamiento. Como objeto fachada para el acceso al sistema real se ha utilizado una clase de gestión de entidades genérica denominada *EntityManager*.

El listado de clases del paquete se muestra en la Tabla 4.1 y el diagrama de clases en la Figura 4.10.

Nombre	Tipo	Descripción
Dao	Interfaz	Define las operaciones de persistencia genéricas.
BaseDao	Clase	Permite implementar las operaciones genéricas mediante un objeto protegido *EntityManager*.

Tabla 4.1: Listado de clases: paquete *server.persistence*

Paquete *server.messaging*

El paquete *messaging* contiene las clases que se encargan de gestionar las comunicaciones asíncronas desde el servidor hacia los clientes y entre los módulos del servidor.

Se ha definido un esquema de comunicaciones tipo productor-consumidor organizado por tópicos. Este esquema permite que las aplicaciones o los servicios se mantengan a la escucha de mensajes, usando las clases consumidoras, al mismo tiempo que otros servicios generan dichos mensajes, utilizando las clases productoras. La comunicación es llevada a cabo por un agente externo al sistema, representado en el diseño mediante la figura del *Broker*.

El listado de clases del paquete se muestra en la Tabla 4.2 y diagrama de clases en la Figura 4.11.

Nombre	Tipo	Descripción
ServerMessage	Clase	Clase abstracta que representa la base para un mensaje.
ServerMessageProducer	Clase	Clase que encapsula el mecanismo de envío de mensajes al *Broker* de comunicaciones.
ServerMessageConsumer	Clase	Clase que encapsula el mecanismo de recepción de mensajes desde el *Broker* de comunicaciones.
ServerMessageObserver	Interfaz	Interfaz que permite desarrollar un patrón observador para la recepción de los mensajes.

Tabla 4.2: Listado de clases: paquete *server.messaging*

Paquete *server.remote*

El paquete *remote* contiene una única clase que permite encapsular el mecanismo de acceso remoto a los servicios del servidor. Permite establecer una abstracción única sobre la que implementar cada tipo cliente necesario durante el desarrollo.

Nombre	Tipo	Descripción
Remote	Genérico	Elemento genérico que representa una interfaz remota cualquiera.
RemoteClient	Clase	Clase genérica y abstracta que encapsula la interfaz remota y las operaciones comunes para establecer la conexión con el servidor.

Tabla 4.3: Listado de clases: paquete *server.remote*

Paquete *server.notices*

El paquete *notices* contiene la lógica de gestión de los avisos. Tiene los objetivos:

- Proporcionar las operaciones de registro, cambio de estado y listado de avisos.
- Proporcionar las clases que desarrollan las operaciones del servicio.
- Definir las clases de gestión del sistema de persistencia para los avisos.
- Proporcionar una abstracción de acceso remoto para las aplicaciones de cliente.
- Proporcionar las clases de gestión de mensajes para el tópico de avisos.

El listado de clases del paquete se muestra en la Tabla 4.4. El diagrama de clases se puede ver en la Figura 4.12.

Nombre	Tipo	Descripción
Notice	Entidad	Define la entidad correspondiente a la clase conceptual Aviso.
NoticeDao	Clase	Define el patrón de acceso DAO sobre la entidad *Notice*.
NoticesLocal	Interfaz	Define las operaciones accesibles desde los módulos de servidor.
NoticesRemote	Interfaz	Define las operaciones accesibles desde los módulos de cliente.
NoticesRemoteClient	Clase	Clase que encapsula el mecanismo de conexión al servicio remoto.
NoticesService	Servicio	Clase que permite desarrollar el servicio de avisos sobre las interfaces *NoticesRemote* y *NoticesLocal*.
NoticeMessage	Clase	Define las propiedades de los mensajes del tópico de avisos.
NoticeMessageConsumer	Clase	Clase consumidora de mensajes del tipo *NoticeMessage*.
NoticeMessageProducer	Clase	Clase productora de mensajes del tipo *NoticeMessage*.
NoticeMessageObserver	Interfaz	Interfaz que permite desarrollar el patrón observador con mensajes de tipo *NoticeMessage*.

Tabla 4.4: Listado de clases: paquete *server.notices*

Paquete *server.ambulances*

El paquete *ambulances* contiene la lógica de gestión de las ambulancias. Tiene los siguientes objetivos:

- Proporcionar las operaciones de registro, actualización y listado de ambulancias.
- Proporcionar las clases que desarrollan las operaciones del servicio.
- Definir las clases de gestión del sistema de persistencia para las ambulancias.
- Proporcionar una abstracción de acceso remoto para las aplicaciones de cliente.
- Proporcionar las clases de gestión de mensajes para el tópico de ambulancias.

La Tabla 4.5 contiene el listado de las clases del paquete. La Figura 4.13 muestra el diagrama de clases.

Nombre	Tipo	Descripción
Ambulance	Entidad	Define la entidad correspondiente a la clase conceptual Ambulancia.
AmbulanceBase	Entidad	Define la entidad correspondiente a la clase conceptual Base de Ambulancia.
AmbulanceDao	Clase	Implementa el patrón DAO sobre la entidad *Ambulance*.
AmbulanceBaseDao	Clase	Implementa el patrón DAO sobre la entidad *AmbulanceBase*.
AmbulancesLocal	Interfaz	Define las operaciones accesibles desde los módulos de servidor.
AmbulancesRemote	Interfaz	Define las operaciones accesibles desde los módulos de cliente.
AmbulancesRemoteClient	Clase	Clase que encapsula el mecanismo de conexión al servicio remoto.
AmbulancesService	Service	Clase que desarrolla las interfaces *AmbulancesRemote* y *AmbulancesLocal*.
AmbulanceMessage	Clase	Define las propiedades de los mensajes del tópico para las ambulancias.
AmbulanceMessageConsumer	Clase	Clase consumidora de mensajes del tipo *AmbulanceMessage*.
AmbulanceMessageProducer	Clase	Clase productora de mensajes del tipo *AmbulanceMessage*.
AmbulanceMessageObserver	Interfaz	Interfaz que permite desarrollar el patrón observador con mensajes de tipo *AmbulanceMessage*.

Tabla 4.5: Listado de clases: paquete *server.ambulances*

La Figura 4.8 muestra un diagrama con los estados válidos en los que puede estar una ambulancia. Este diagrama permite establecer de una forma sencilla el conjunto de estados alcanzables a partir de un estado cualquiera, además de especificar el mensaje que provoca cada cambio de estado.

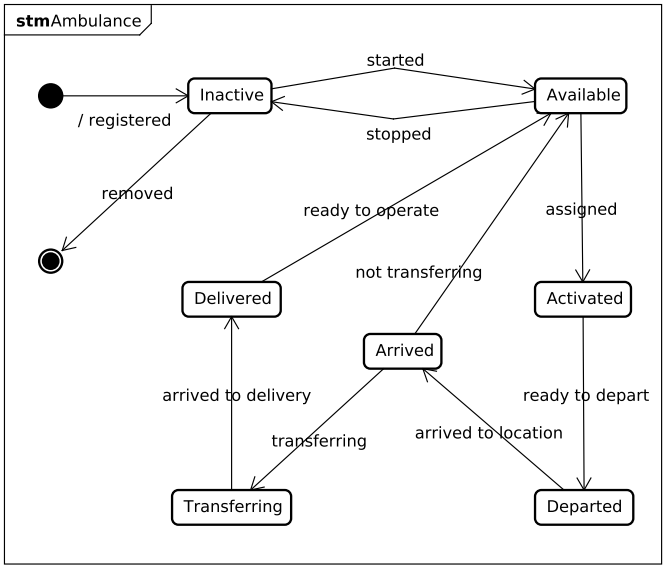


Figura 4.8: Diagrama de estados: entidad *Ambulance*

Paquete *server.assignments*

El paquete *assignments* contiene la lógica de gestión de las asignaciones. Las clases definidas cumplen los objetivos siguientes:

- Proporcionar las operaciones de registro, cambio de estado y listado de asignaciones.
- Proporcionar las clases que desarrollan las operaciones del servicio.
- Definir las clases de gestión del sistema de persistencia para las asignaciones.
- Proporcionar una abstracción de acceso remoto para las aplicaciones de cliente.
- Proporcionar las clases de gestión de mensajes para el tópico de asignaciones.
- Proporcionar un mecanismo para monitorizar las operaciones de creación o cancelación de avisos.

El listado de clases se muestra en la Tabla 4.6 y el diagrama de clases en la Figura 4.14. La Figura 4.9 contiene el diagrama de estados para la entidad *Assignment*.

Nombre	Tipo	Descripción
Assignment	Entidad	Define la entidad correspondiente a la clase conceptual Asignación.
AssignmentDao	Clase	Implementa el patrón DAO sobre la entidad *Assignment*.
AssignmentsLocal	Interfaz	Define las operaciones accesibles desde los módulos de servidor.
AssignmentsRemote	Interfaz	Define las operaciones accesibles desde los módulos de cliente.
AssignmentsRemoteClient	Clase	Clase que encapsula el mecanismo de conexión al servicio remoto.
AssignmentsService	Servicio	Permite desarrollar las interfaces *AssignmentsRemote* y *AssignmentsLocal*.
AssignmentMessage	Clase	Define las propiedades de los mensajes del tópico de asignaciones.
AssignmentMessageConsumer	Clase	Clase consumidora de mensajes del tipo *AssignmentMessage*.
AssignmentMessageProducer	Clase	Clase productora de mensajes del tipo *AssignmentMessage*.
AssignmentMessageObserver	Interfaz	Interfaz que permite desarrollar el patrón observador con mensajes de tipo *AssignmentMessage*.
AssignmentObserver	Servicio	Servicio que define un observador de mensajes del tópico de avisos y permite actualizar automáticamente las asignaciones en función de los eventos de dichos mensajes.

Tabla 4.6: Listado de clases: paquete *server.assignments*

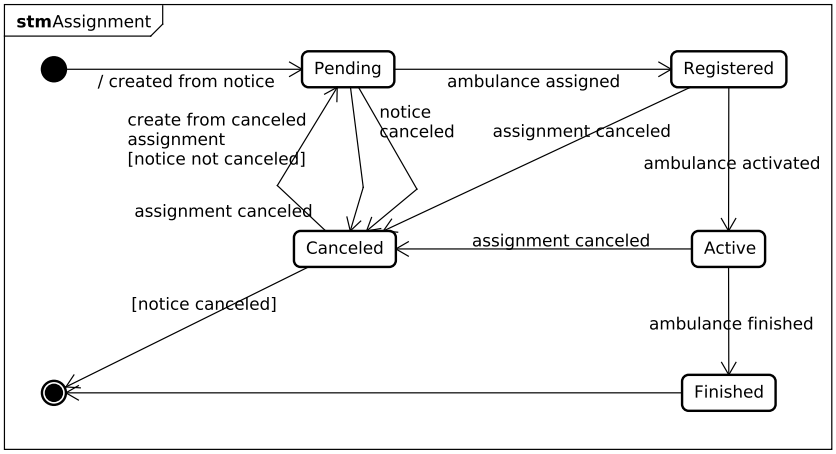


Figura 4.9: Diagrama de estados: entidad *Assignment*

Paquete *server.alternatives*

El paquete *alternatives* contiene el mecanismo de apoyo a la asignación utilizado por el operador del CCUE. Tiene los objetivos principales:

- Proporcionar las operaciones de cálculo de alternativas de asignación y valoración del impacto de una alternativa en el territorio.
- Proporcionar un mecanismo de acceso remoto para las aplicaciones de cliente.

El concepto de alternativa permite asociar una asignación pendiente con una ambulancia de una forma temporal y previa al registro definitivo. Está representada por la clase *Alternative* e incluye como atributos: la asignación, una de las ambulancias disponibles, el tiempo estimado de respuesta y una valoración de la capacidad de actuación en el territorio tras la activación.

Los tiempos de respuesta y las valoraciones se obtienen mediante la clases *ResponseTimeCalculator*, *PreparednessCalculator* y *AlternativeCalculator*, las cuales desarrollan los Algoritmos 1, 2 y 3 respectivamente.

El listado de clases del paquete se muestra en la Tabla 4.7 y los diagramas de clases en la Figura 4.17.

Nombre	Tipo	Descripción
Alternative	Clase	Representa una alternativa de asignación con una valoración calculada.
AlternativesRemote	Interfaz	Define las operaciones accesibles desde los módulos de cliente.
AlternativesRemoteClient	Clase	Clase que encapsula el mecanismo de conexión al servicio remoto.
AlternativesService	Servicio	Servicio que implementa la interfaz *AlternativesRemote*.
AlternativesCalculator	Clase	Clase que representa la regla de asignación general.
ResponseTimeCalculator	Clase	Realiza el cálculo del criterio de minimización del tiempo de respuesta.
PreparednessCalculator	Clase	Realiza el cálculo del criterio de maximización de la capacidad de respuesta.

Tabla 4.7: Listado de clases: paquete *server.alternatives*

Paquete *server.gis*

El paquete *gis* contiene la lógica de acceso los sistemas de información geográfica. Las clases definidas cumplen los objetivos siguientes:

- Proporcionar las operaciones de geolocalización y cálculo de rutas.
- Proporcionar las clases que desarrollan las operaciones del servicio.
- Proporcionar un mecanismo de acceso remoto para las aplicaciones de cliente.
- Definir varios tipos de datos y entidades para el intercambio y almacenamiento de información geográfica.
- Proporcionar las interfaces necesarias para el acceso a servicios externos de geolocalización y cálculo de rutas.

El listado de clases del paquete se muestra en la Tabla 4.8. El diagrama de clases se puede ver en la Figura 4.15.

Nombre	Tipo	Descripción
Address	Entidad	Define las propiedades para la gestión de una dirección como entidad.
Coordinates	Entidad	Define las propiedades para la gestión de un par de coordenadas geográficas como entidad.
Location	Clase	Clase que relaciona un par de coordenadas y una dirección.
Route	Clase	Representa una ruta por carretera y está formada un listado ordenado de coordenadas geográficas.
GisRemote	Interfaz	Define las operaciones accesibles desde los módulos de cliente.
GisRemoteClient	Clase	Clase que encapsula el mecanismo de conexión al servicio remoto.
GisService	Servicio	Clase que permite desarrollar la interfaz remota *GisRemote*.
GeolocateClient	Interfaz	Define una interfaz para la creación de clientes a servicios externos de geolocalización de direcciones.
RoutingClient	Interfaz	Define una interfaz para la creación de clientes a servicios externos de cálculo de rutas.

Tabla 4.8: Listado de clases: paquete *server.gis*

Paquete *server.zones*

El paquete *zones* contiene la lógica de gestión de las zonas de actuación. Tiene los objetivos:

- Proporcionar las operaciones de listado y localización de zonas.
- Proporcionar la operación de obtención de tiempo de respuesta entre zonas.
- Definir las clases de gestión del sistema de persistencia para las zonas.

El listado de clases del paquete se muestra en la Tabla 4.9 y los diagramas de clases en la Figura 4.16.

Nombre	Tipo	Descripción
Zone	Entidad	Define la entidad correspondiente a la clase conceptual Zona de Actuación.
ZoneDao	Clase	Clase que permite definir el patrón DAO sobre la entidad *Zone*.
ResponseTime	Entidad	Define las propiedades para la gestión del tiempo de respuesta entre zonas como entidad.
ResponseTimeDao	Clase	Clase que permite definir el patrón DAO sobre la entidad *ResponseTime*
ZonesLocal	Interfaz	Define las operaciones accesibles desde los módulos de servidor.
ZonesService	Servicio	Clase que permite desarrollar la interfaz *ZonesLocal*.
ZoneCalculator	Clase	Realiza los cálculos relativos a las coordenadas geográficas de las zonas.

Tabla 4.9: Listado de clases: paquete *server.zones*

Módulo *client-notices*

El módulo *client-notices* contiene la lógica de la aplicación de cliente utilizada por el operador telefónico para la creación y cancelación de avisos. Tiene los objetivos:

- Proporcionar una clase de control, que desarrolle las operaciones descritas en el DSS de la Figura 4.1 y mantenga el estado del aviso en curso.
- Proporcionar un conjunto de objetos delegados para el acceso a los servicios remotos de gestión de avisos y geolocalización. Esta abstracción permite encapsular los mecanismos de conexión remota en una clase independiente, reduciendo el acoplamiento entre el servidor y el controlador.

El listado de clases del paquete se muestra en la Tabla 4.10 y el diagrama de clases en la Figura 4.18.

Nombre	Tipo	Descripción
Controller	Clase	Representa el controlador principal del módulo. Define las operaciones principales del DSS asociado.
NoticesServiceDelegate	Clase	Clase que encapsula el cliente remoto para la conexión con el servicio de avisos.
GisServiceDelegate	Clase	Clase que encapsula el cliente remoto para la conexión con el servicio de información geográfica.

Tabla 4.10: Listado de clases: módulo *client-notices*

Paquete *client-ccue.ambulances*

El paquete *ambulances*, perteneciente al módulo *client-ccue*, contiene la lógica de aplicación de cliente utilizada por el operador del CCUE para la creación y modificación de ambulancias. Tiene los objetivos:

- Proporcionar una clase de control, que desarrolle las operaciones descritas en los DSS de las Figuras 4.4 y 4.5 y mantenga el estado de la ambulancia abierta.
- Proporcionar un objeto delegado para el acceso al servicio remoto de gestión de ambulancias.

El listado de clases del paquete se muestra en la Tabla 4.11 y el diagrama de clases en la Figura 4.19.

Nombre	Tipo	Descripción
Controller	Clase	Representa el controlador principal del paquete. Define las operaciones principales de los DSS asociados.
AmbulancesServiceDelegate	Clase	Clase que encapsula el cliente remoto para la conexión con el servicio de ambulancias.

Tabla 4.11: Listado de clases: paquete *client-ccue.ambulances*

Paquete *client-ccue.assignments*

El paquete *assignments*, perteneciente al módulo *client-ccue*, contiene la lógica de aplicación de cliente utilizada por el operador del CCUE para el registro y cancelación de asignaciones. Tiene los objetivos:

- Proporcionar una clase de control, que desarrolle las operaciones descritas en los DSS de las Figuras 4.2, 4.3 y mantenga el estado de la asignación en curso.
- Proporcionar un conjunto de objetos delegados para el acceso a los servicios remotos de gestión de asignaciones y alternativas.
- Proporcionar un objeto delegado que facilite la recepción de mensajes de servidor enviados al tópico de asignaciones.

El listado de clases del paquete se muestra en la Tabla 4.12 y el diagrama de clases en la Figura. 4.20.

Nombre	Tipo	Descripción
Controller	Clase	Representa el controlador principal del módulo. Define las operaciones principales de los DSS asociados.
AssignmentsServiceDelegate	Clase	Clase que encapsula el cliente remoto para la conexión con el servicio de asignaciones.
AlternativesServiceDelegate	Clase	Clase que encapsula el cliente remoto para la conexión con el servicio de cálculo de alternativas de asignación.
AssignmentMessageConsumerDelegate	Clase	Clase que encapsula el consumidor de mensajes para el tópico de asignaciones y permite el registro de un objeto observador de mensajes.

Tabla 4.12: Listado de clases: paquete *client-ccue.assignments*

Módulo *client-ambulance*

El módulo *client-ambulance*, contiene la lógica de aplicación de cliente utilizada por el operador de la ambulancia durante la ejecución de la asignación y el registro de la ambulancia. Tiene los objetivos:

- Proporcionar una clase de control, que desarrolle las operaciones descritas en los DSS de las Figuras 4.6, 4.4 y mantenga el estado tanto de la asignación en curso como de la ambulancia asociada al cliente.
- Proporcionar un conjunto de objetos delegados para el acceso a los servicios remotos de gestión de ambulancias y asignaciones.

- Proporcionar un objeto delegado que facilite la recepción de mensajes de servidor enviados al canal de asignaciones.

El listado de clases del paquete se muestra en la Tabla 4.13 y el diagrama de clases en la Figura 4.21.

Nombre	Tipo	Descripción
Controller	Clase	Representa el controlador principal del módulo. Define las operaciones principales de los DSS asociados.
AssignmentsServiceDelegate	Clase	Clase que encapsula el cliente remoto para la conexión con el servicio de asignaciones.
AmbulancesServiceDelegate	Clase	Clase que encapsula el cliente remoto para la conexión con el servicio de ambulancias.
AssignmentMessageConsumerDelegate	Clase	Clase que encapsula el consumidor de mensajes para el canal de asignaciones y permite el registro de un objeto observador de mensajes.

Tabla 4.13: Listado de clases: paquete *client-ambulance*

4.2.3. Diagramas de clases

Las siguientes Figuras muestran los diagramas de clases correspondientes a los componentes definidos en el diagrama de la Figura 4.7.

Diagrama de clases	Figura
Paquete *server.persistence*	Figura 4.10
Paquete *server.messaging*	Figura 4.11
Paquete *server.notices*	Figura 4.12
Paquete *server.ambulances*	Figura 4.13
Paquete *server.assignments*	Figura 4.14
Paquete *server.gis*	Figura 4.15
Paquete *server.zones*	Figura 4.16
Paquete *server.alternatives*	Figura 4.17
Módulo *client-notices*	Figura 4.18
Paquete *client-ccue.ambulances*	Figura 4.19
Paquete *client-ccue.assignments*	Figura 4.20
Módulo *client-ambulance*	Figura 4.21

Tabla 4.14: Relación de Figuras de diagramas de clases

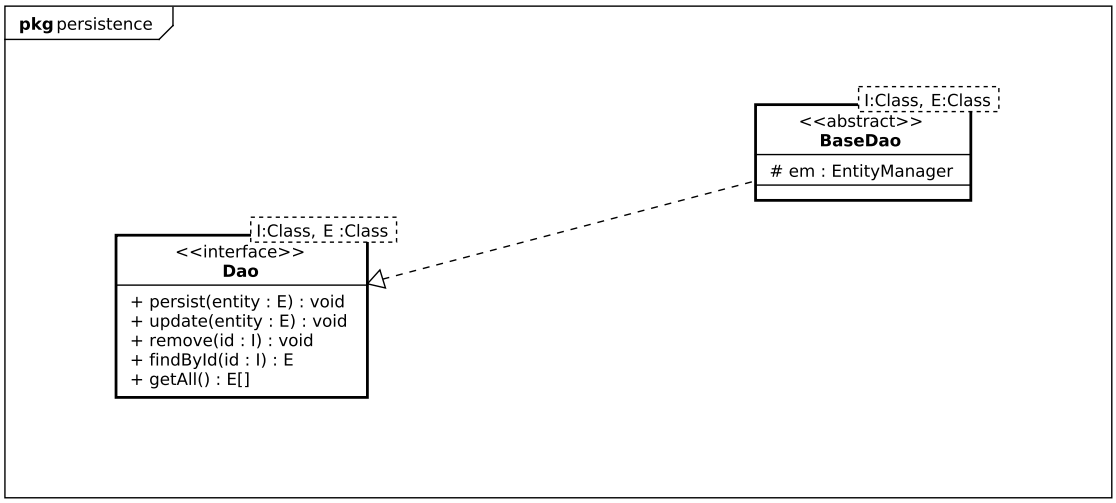


Figura 4.10: Diagrama de clases: paquete *server.persistence*

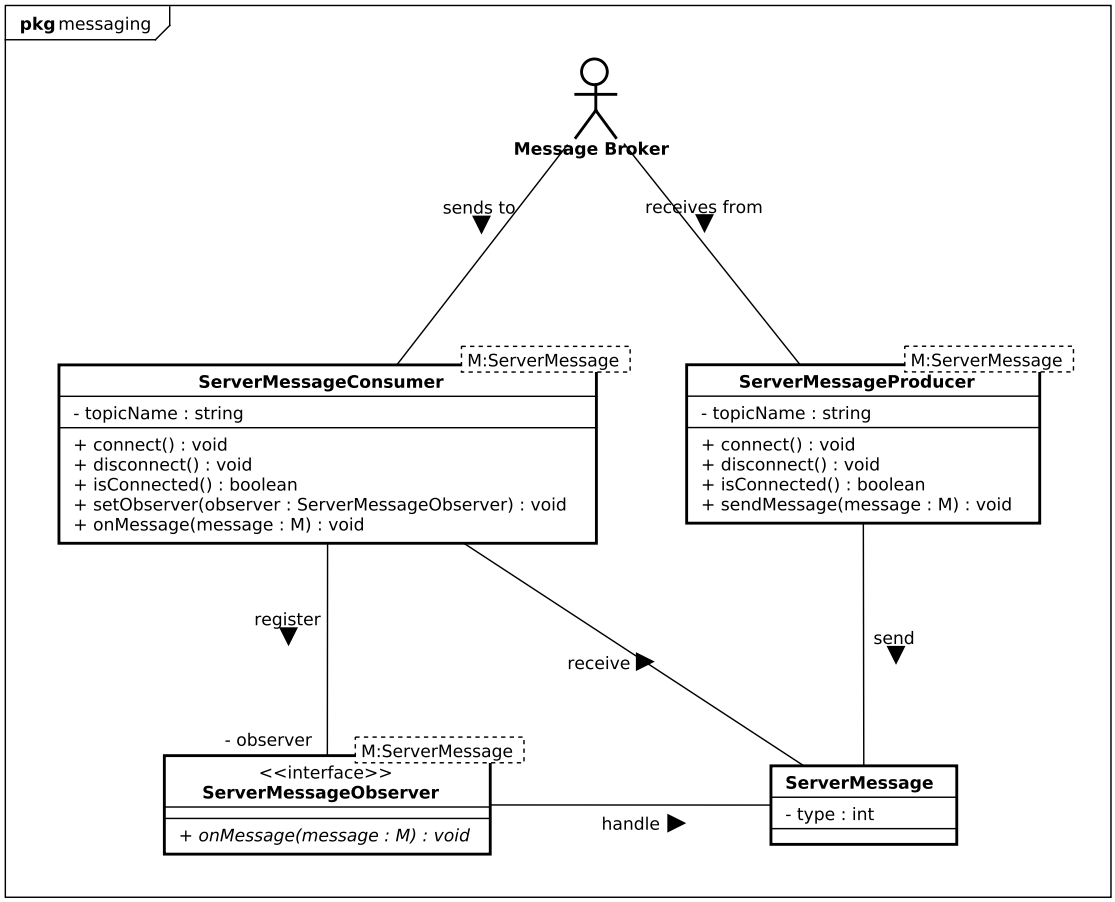


Figura 4.11: Diagrama de clases: paquete *server.messaging*

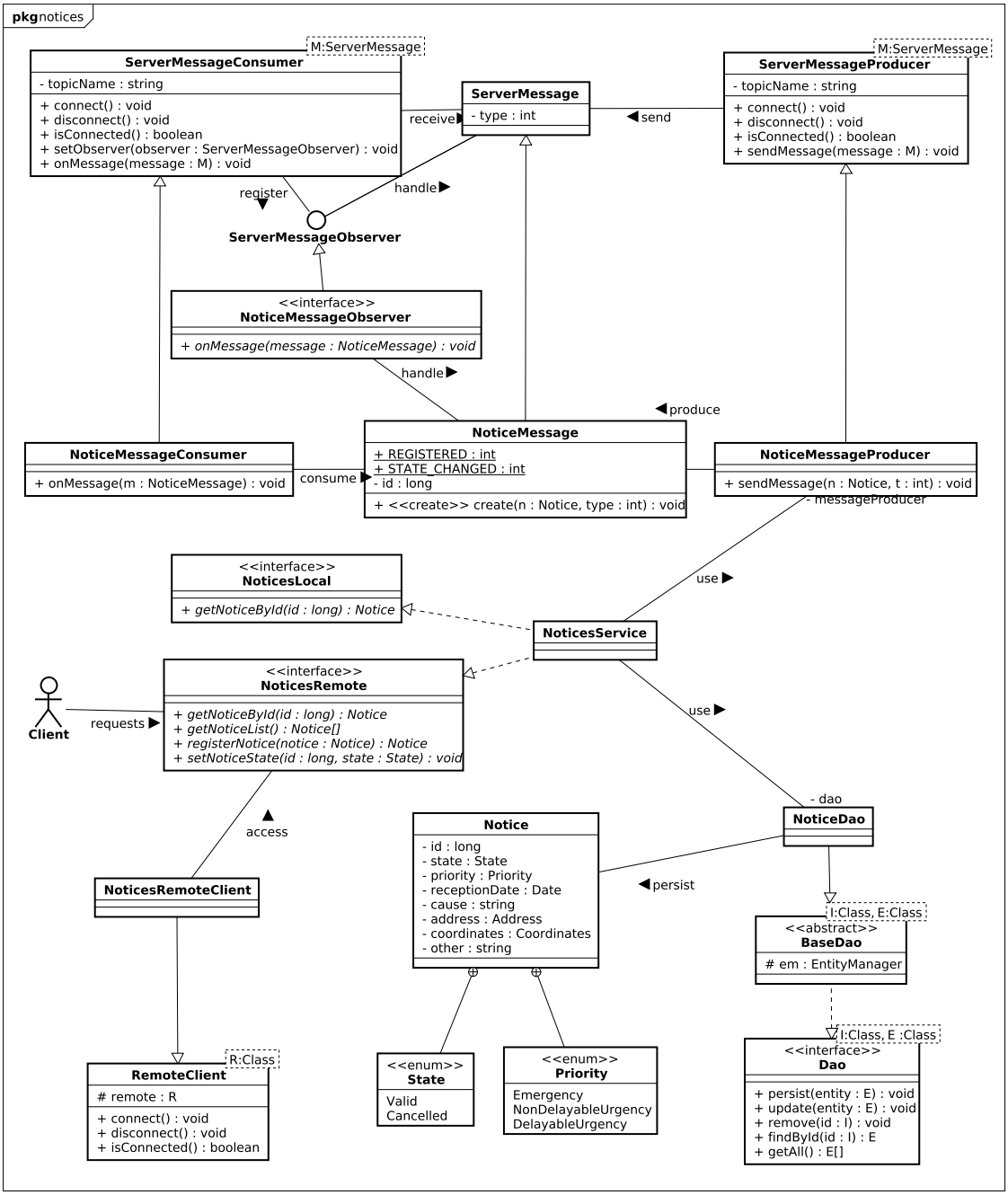


Figura 4.12: Diagrama de clases: paquete *server.notices*

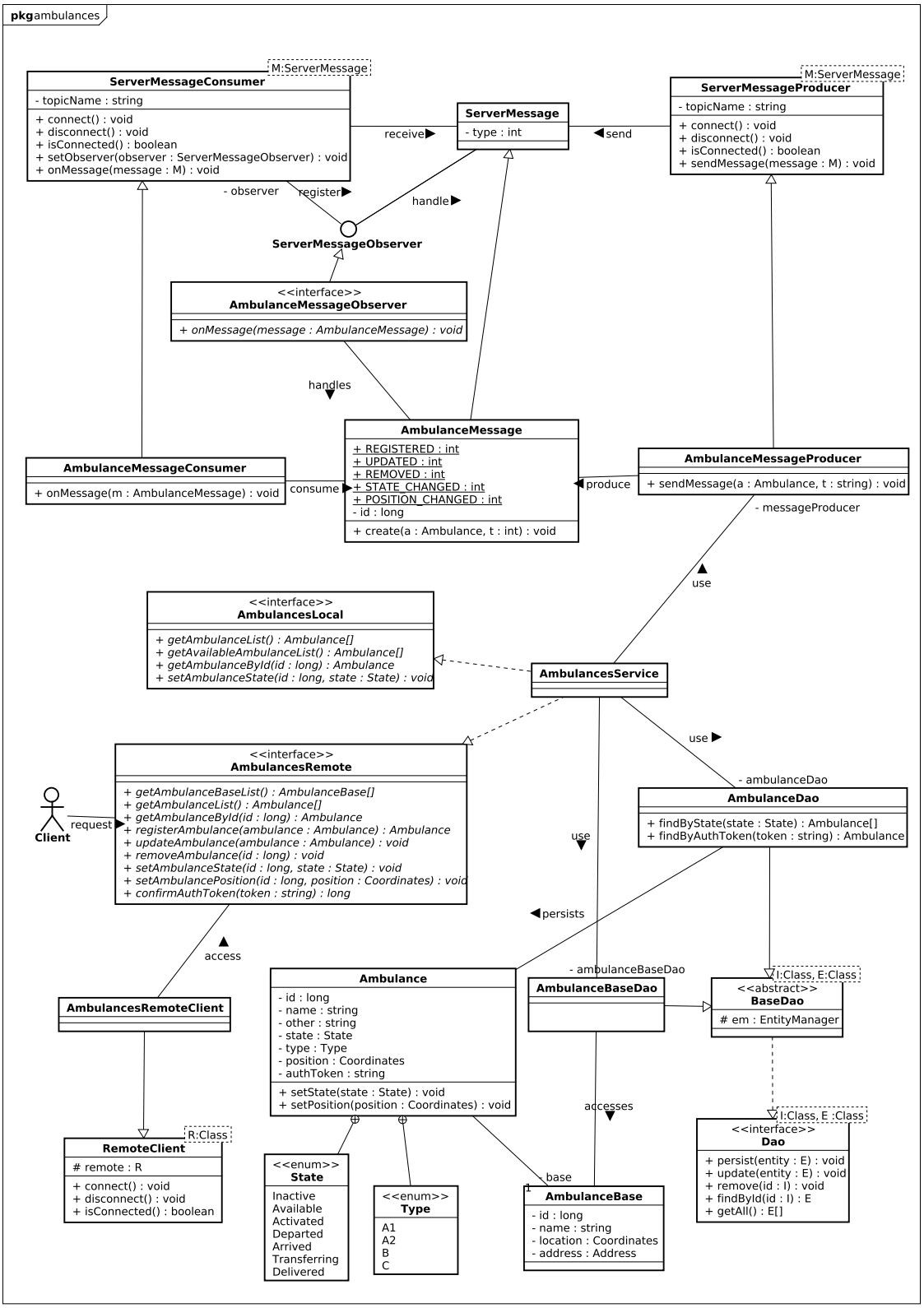


Figura 4.13: Diagrama de clases: paquete *server.ambulances*

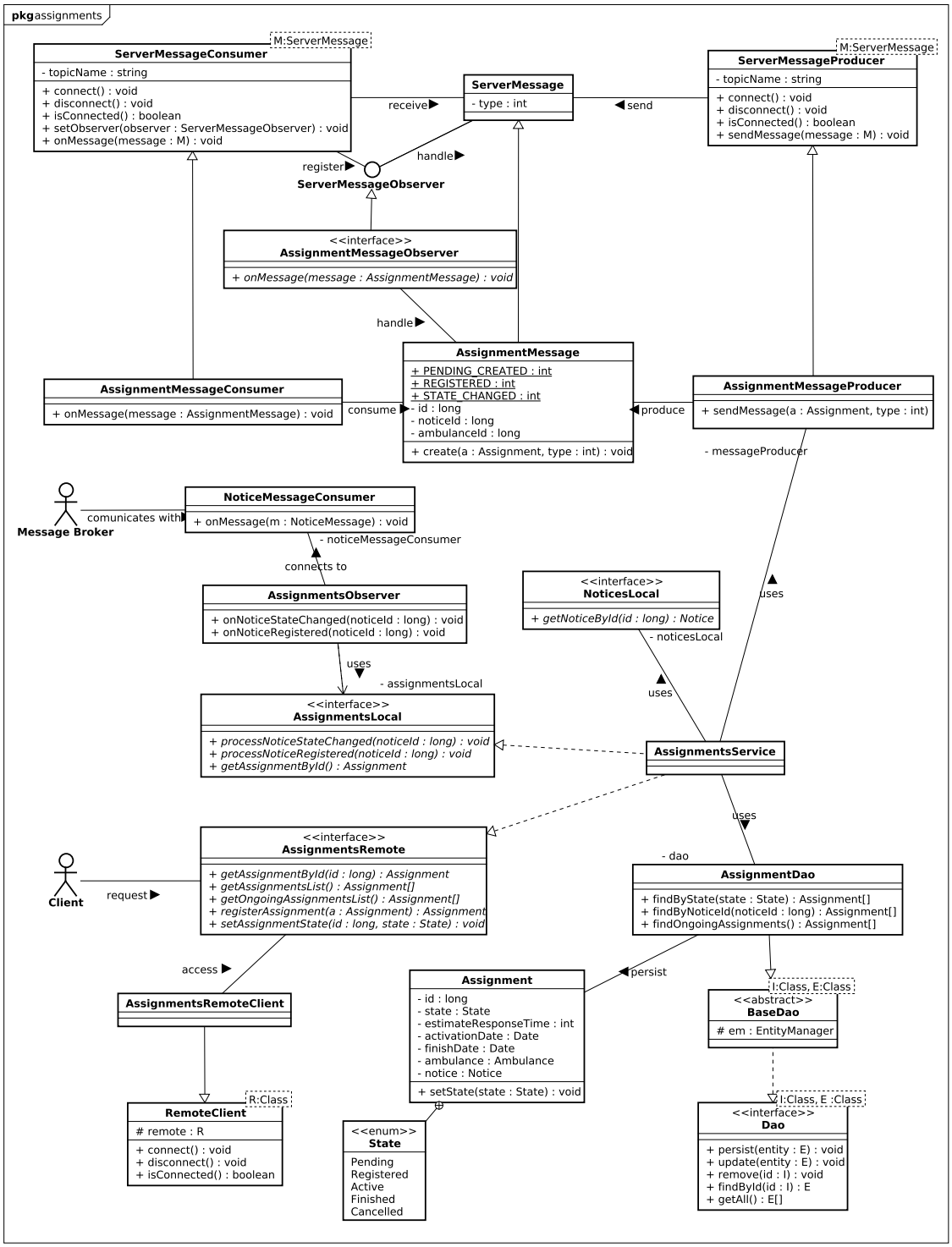


Figura 4.14: Diagrama de clases: paquete *server.assignments*

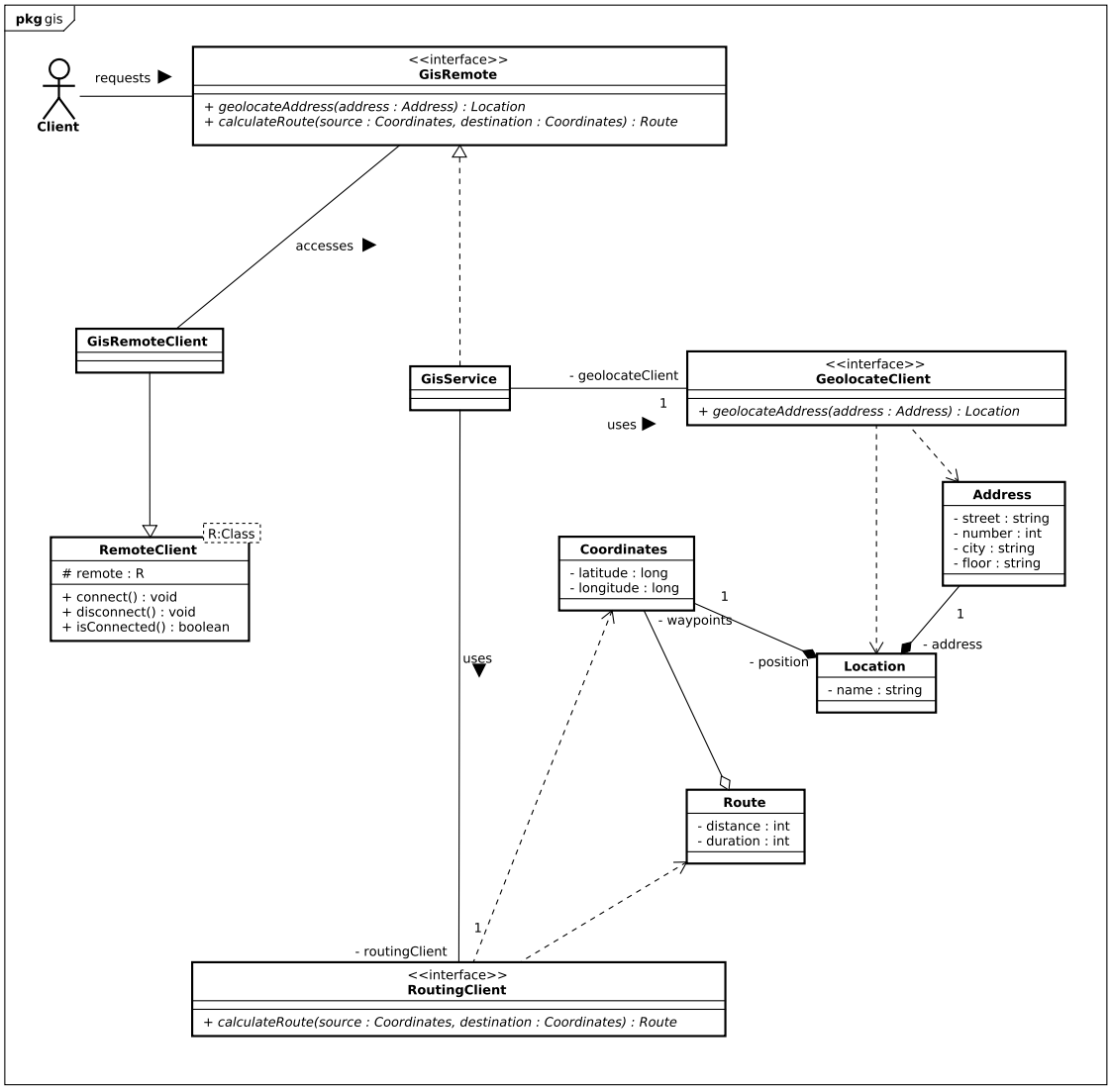


Figura 4.15: Diagrama de clases: paquete *server.gis*

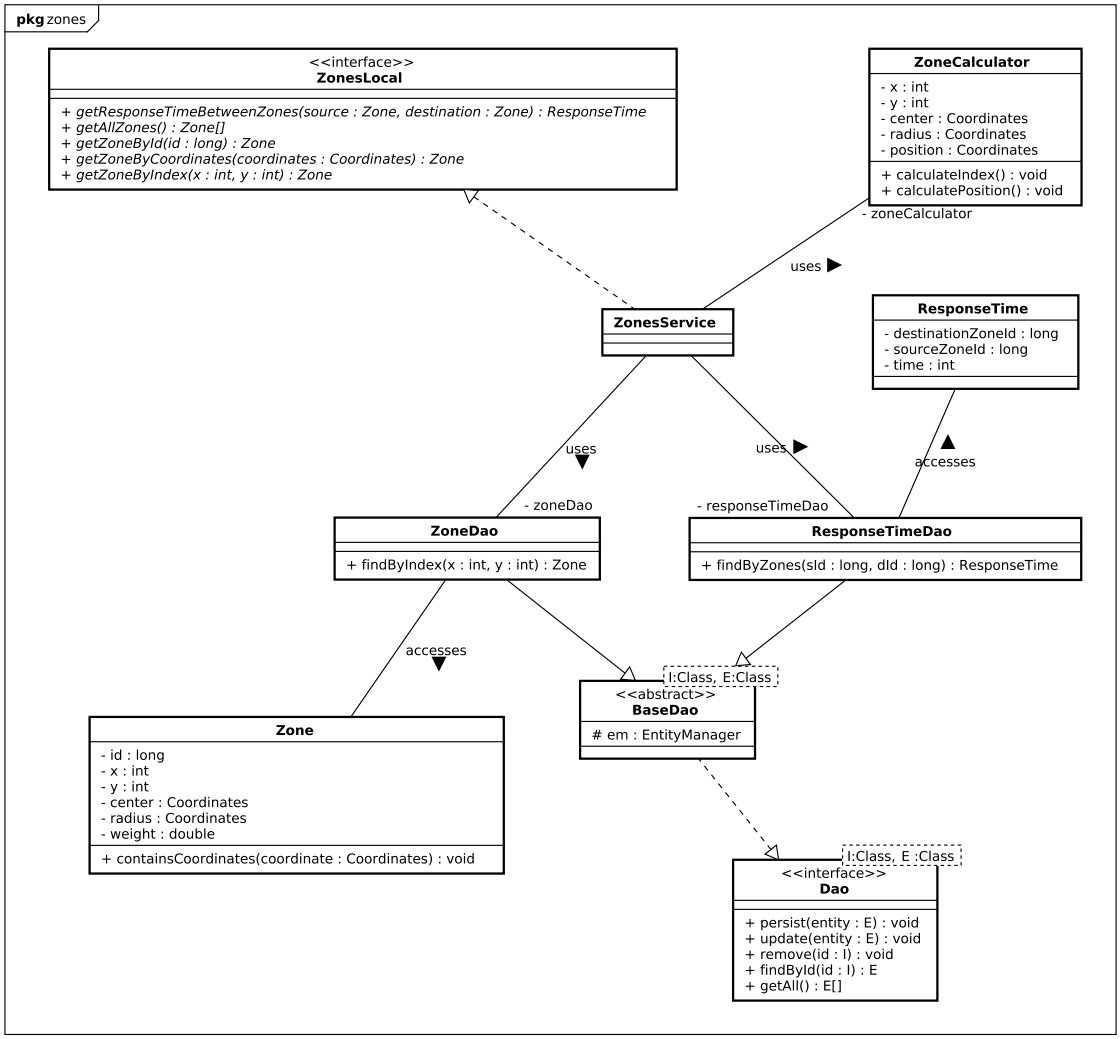


Figura 4.16: Diagrama de clases: paquete *server.zones*

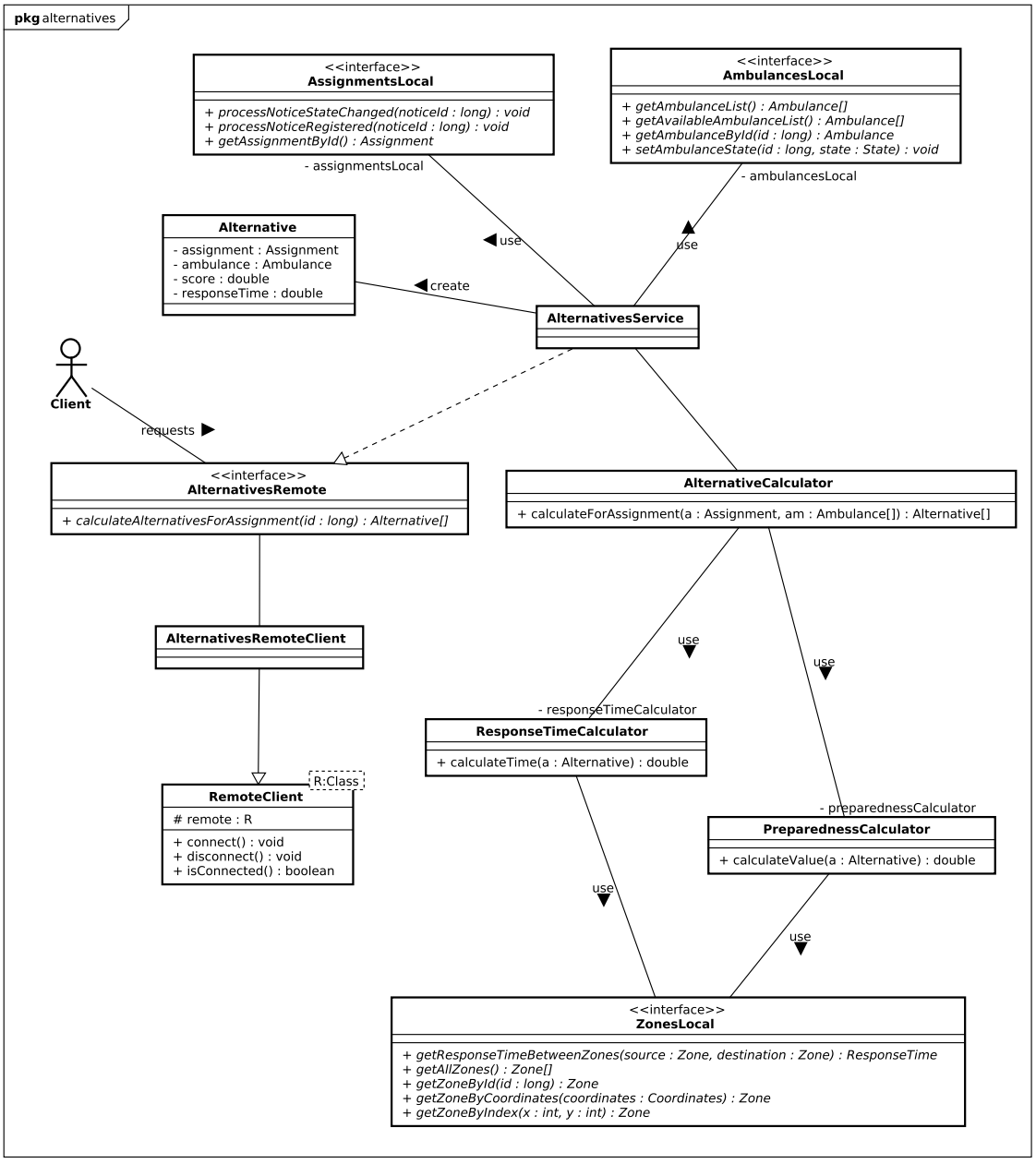


Figura 4.17: Diagrama de clases: paquete *server.alternatives*

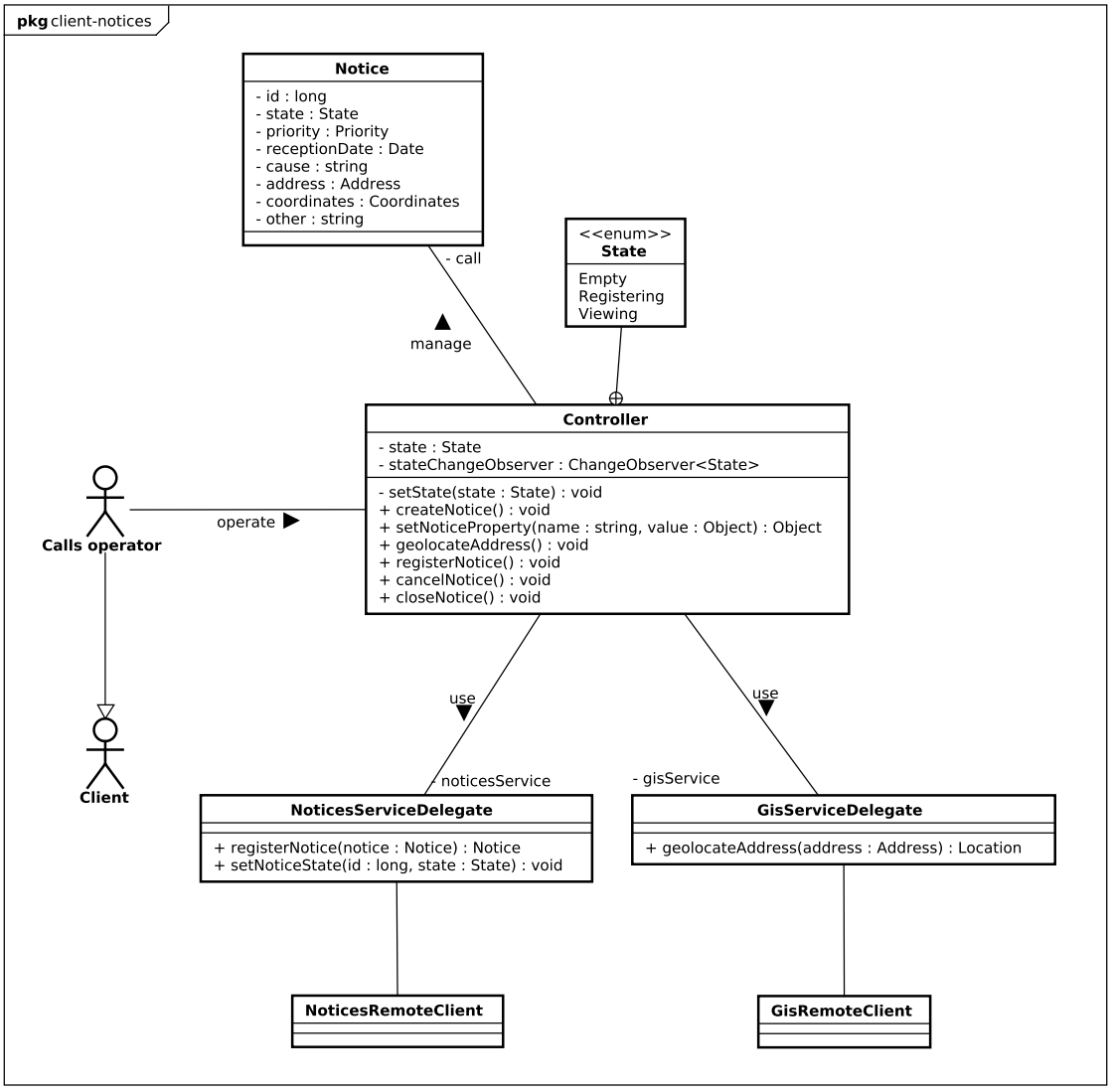


Figura 4.18: Diagrama de clases: modulo *client-notices*

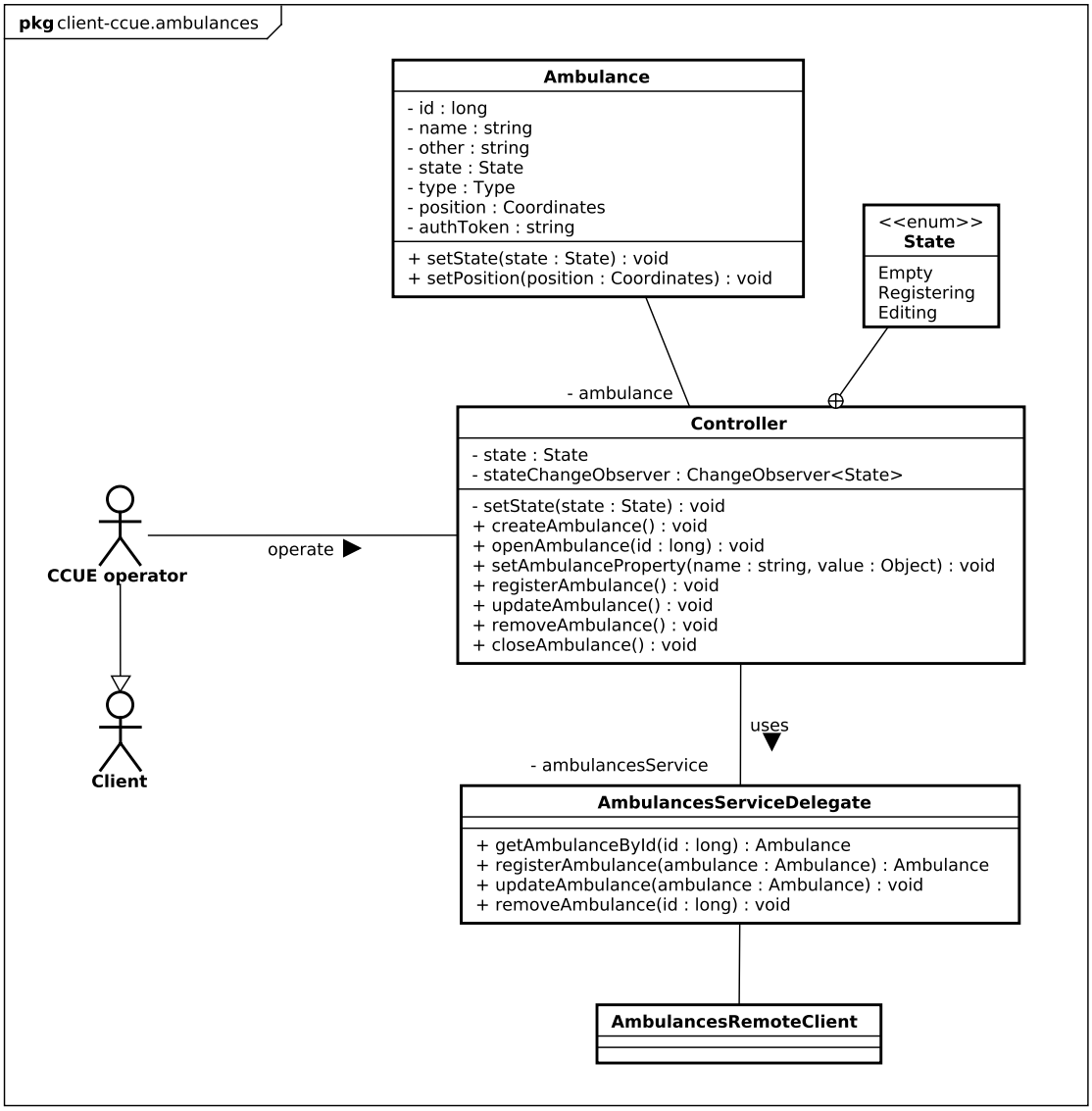


Figura 4.19: Diagrama de clases: paquete *client-ccue.ambulances*

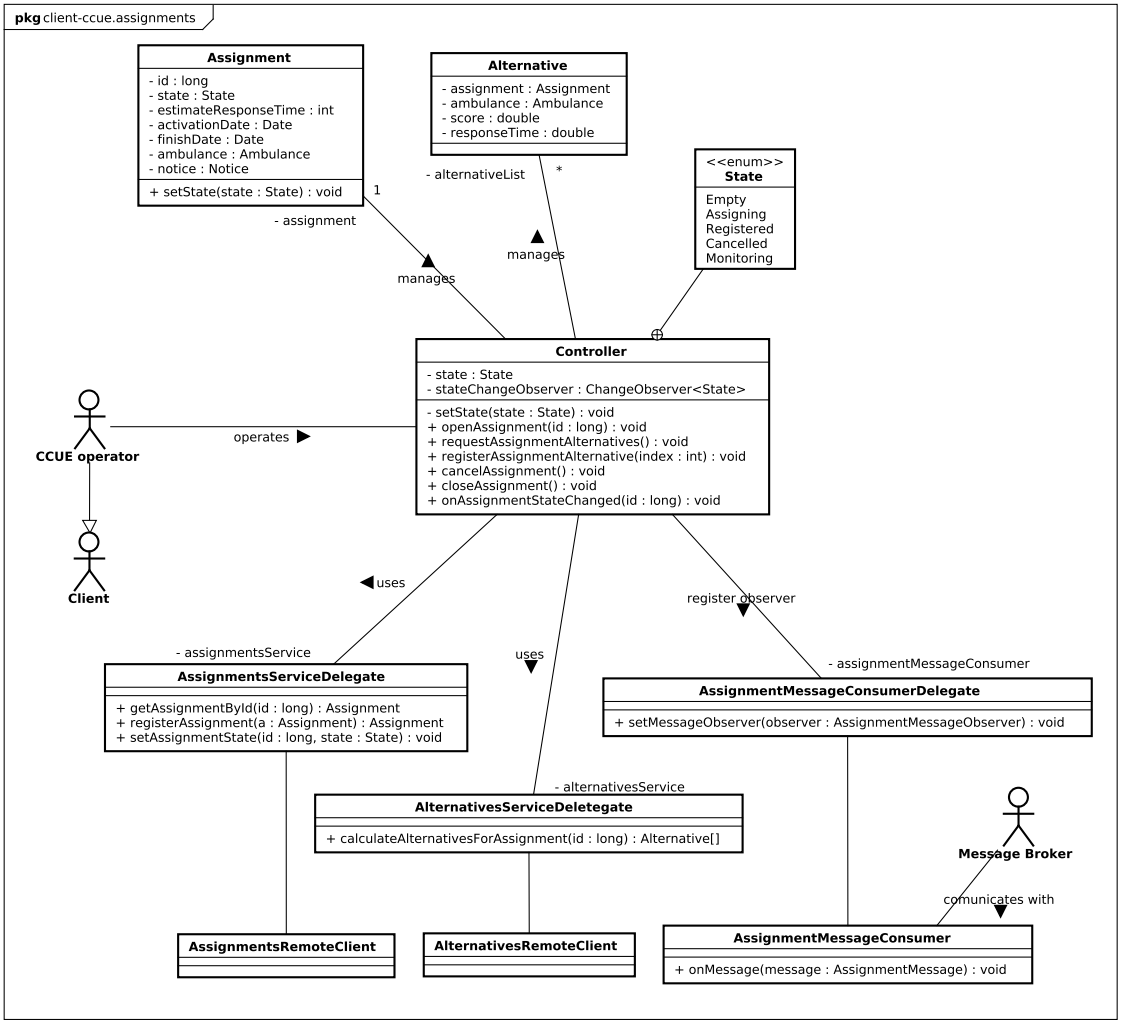


Figura 4.20: Diagrama de clases: paquete *client-ccue.assignments*

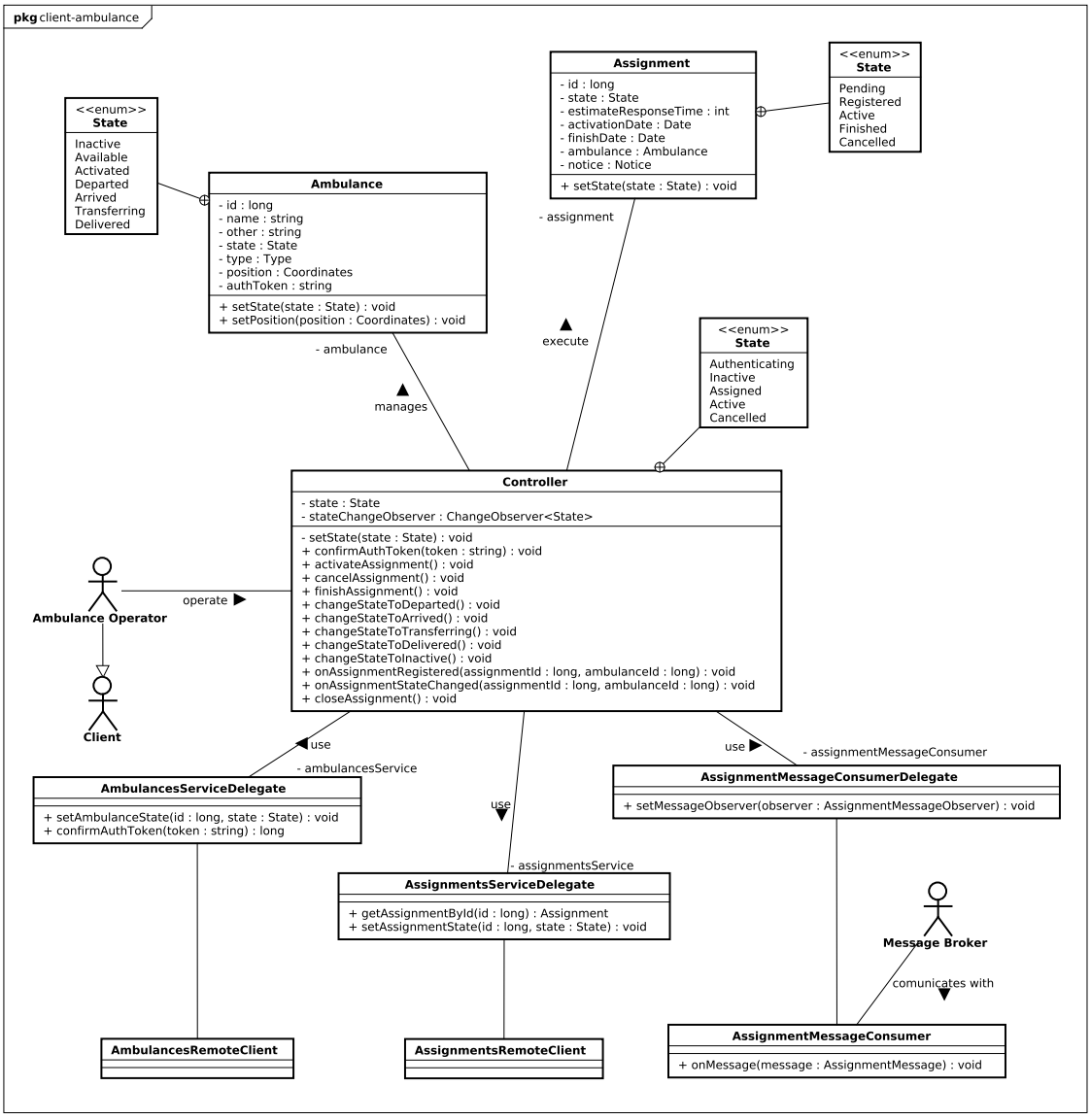


Figura 4.21: Diagrama de clases: paquete *client-ambulance*

4.3. Modelo de datos

La Figura 4.22 muestra el diagrama entidad-relación resultante del proceso de diseño junto con las propiedades de cada entidad.

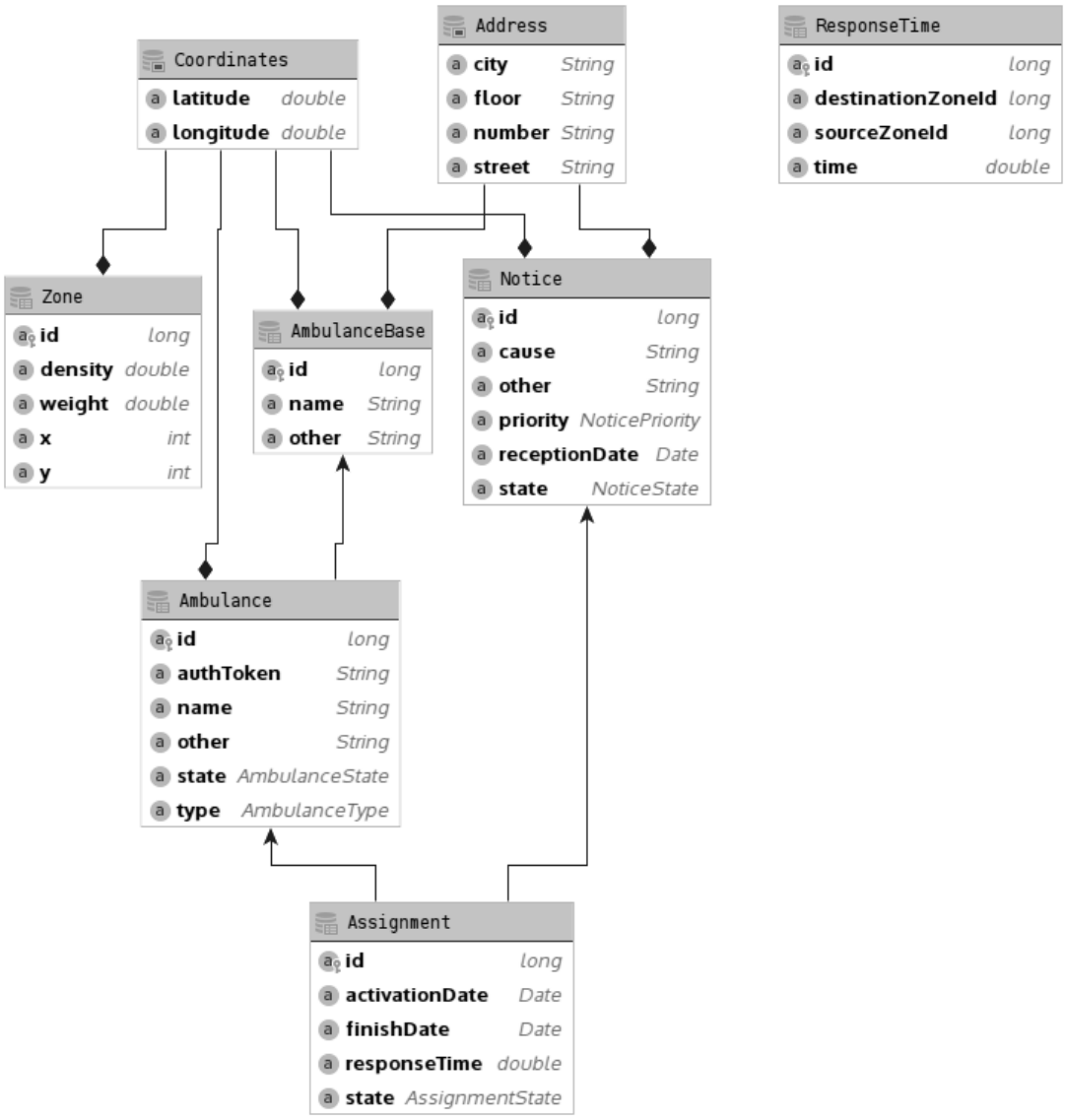


Figura 4.22: Diagrama Entidad-Relación

4.4. Conclusiones

En este capítulo se han presentado los elementos documentales más importantes resultado del proceso de diseño del sistema:

- Diagramas de secuencia del sistema: Permiten definir las operaciones entre los actores y el propio sistema.
- Diagramas de componentes: Muestran como se estructura el proyecto desde su perspectiva lógica.
- Diagramas de clases: Detallan el contenido de cada componente, junto con sus responsabilidades y dependencias.
- Diagramas Entidad-Relación: Permiten definir las entidades del sistema de persistencia.

Estos documentos son utilizados durante la fase de construcción como punto de inicio a la implementación, permitiendo en muchos casos traducir de forma directa los componentes de la arquitectura lógica en código ejecutable.

Capítulo 5

Implementación y pruebas

5.1. Introducción

En este capítulo se analiza el resultado de las tareas de implementación y pruebas desde el punto de vista de la funcionalidad obtenida y las tecnologías utilizadas.

La implementación se puede describir como la transformación de la lógica definida durante el diseño en código ejecutable. Las pruebas permiten comprobar que el sistema construido funciona adecuadamente y cumple con los requisitos especificados durante el análisis.

Para su realización se han tenido en cuenta los siguientes criterios:

1. Traducir los componentes lógicos en código de una forma lo más directa posible, evitando crear estructuras innecesarias.
2. Escribir código fuente autodocumentado y legible.
3. Crear interfaces gráficas con un adecuado nivel de interactividad y facilidad de uso.
4. Utilizar mecanismos de prueba continuos y automatizados.

Contiene las Secciones Entorno de desarrollo, Implementación y Pruebas.

5.2. Entorno de desarrollo

En esta sección se detallan los lenguajes de programación, herramientas software y hardware utilizadas durante el desarrollo.

5.2.1. Lenguajes de programación

- **Java**

  Java es un lenguaje de programación orientado a objetos, multiplataforma y con una gran cantidad de recursos disponibles. Permite el desarrollo de software escritorio, de ámbito empresarial o aplicaciones web, entre otros.

- **XML**

  XML es un lenguaje de marcas extensible derivado de SGML. Es ampliamente utilizado como elemento de transporte para datos electrónicos o como soporte para el almacenamiento de información en forma de archivos.

- **SQL**

  SQL es un lenguaje que permite la definición de estructuras de datos y operaciones sobre dichas estructuras. Es utilizado para la creación de estructuras de la base de datos y la importación inicial tras el despliegue.

5.2.2. Herramientas software

- **OpenJDK** [Oraa]

  La plataforma OpenJDK consiste en una implementación abierta del Kit de Desarrollo Java. Es compatible con los estándares existentes en torno al lenguaje y tiene como objetivos ampliar y mejorar dichos estándares.

- **Apache TomEE** [Thea]

  Apache TomEE es un servidor de aplicaciones Java EE que integra los servicios de ejecución de contenedores EJB, acceso a capas de datos con JPA o comunicaciones entre máquinas con JMS y JAX-WS. Es utilizado para la ejecución de los módulos del sistema de procesamiento del centro de coordinación.

- **OpenJFX** [Glub]

  La plataforma OpenJFX consiste en un conjunto de herramientas que facilitan la creación de aplicaciones de interfaz gráfica de usuario utilizando el lenguaje Java. Proporciona una amplia colección de controles gráficos prediseñados y facilita la creación de interfaces de aplicación completas mediante el uso de guiones *fxml* y herramientas WYSIWYG.

- **Librería JavaFXPorts** [Glua]

  JavaFXPorts es una librería que permite el desarrollo de aplicaciones JavaFX para dispositivos móviles. Facilita que un mismo código se pueda ejecutar en una amplia variedad de plataformas mejorando la capacidad de despliegue del sistema.

- **Librería MapsForge** [mapb]

  MapsForge es una librería de visualización y gestión de mapas vectoriales con licencia libre. No requiere conexión a Internet y está disponible para aplicaciones de escritorio y dispositivos Android.

- **MariaDB Server** [Theb]

  MariaDB es un servidor de bases de datos relacional con licencia libre. Ofrece una alta seguridad, disponibilidad y se integra fácilmente con los servidores de aplicaciones Apache TomEE.

- **OSRM** [osr]

  OSRM es un servidor de cálculo de rutas en redes de carreteras. Opera sobre los mapas *OpenStreetMap* y tiene licencia libre. Es usado para precalcular los tiempos de trayecto entre zonas.

- **Nominatim** [Funa]

  Nominatim es una herramienta que permite la obtención de datos de localización geográfica a partir de nombres o direcciones. Opera sobre los mapas *OpenStreetMap* [Funb] y tiene licencia libre. Es utilizado para obtener las coordenadas geográficas de los avisos a partir de una dirección o el nombre de un lugar conocido.

- **Oracle VirtualBox** [Orab]

  VirtualBox es una herramienta de virtualización que permite la ejecución de máquinas virtuales sobre el hardware de un ordenador personal. Es usado para ejecutar cada servidor en su propia máquina de forma que el entorno de desarrollo emule el sistema distribuido objetivo.

- **Astah UML** [Cha]

  Astah UML es un editor de modelado UML multiplataforma. Está disponible a través de múltiples licencias, incluida una opción académica orientada a estudiantes. Es utilizado para la creación de los diagramas y modelos presentados en este documento.

- **IntelliJ IDEA Ultimate** [Jet]

  IntelliJ IDEA Ultimate es un entorno de desarrollo integrado enfocado al desarrollo de aplicaciones Java. Facilita tareas como la escritura de código, la gestión de versiones y la integración con las soluciones de depuración. Está disponible en dos versiones, una soportada por la comunidad y otra gestionada por la empresa desarrolladora. Se ha elegido esta última, al ser la única de las dos que integra las herramientas para el desarrollo de aplicaciones Java EE.

La Tabla 5.1 muestra una relación entre estas herramientas y los números de versión utilizados durante construcción de la última revisión.

Nombre	Versión
OpenJDK	1.8.0_222
Apache TomEE Plus	7.1.0
OpenJFX	7u111
JavaFXPorts	8.60.13
MapsForge	0.11.0
MariaDB Server	10.1.37
Nominatim	3.3.0
OSRM Routed	5.22.0
Oracle VirtualBox	6.0
Astah UML	8.1
IntelliJ IDEA Ultimate	2019.2

Tabla 5.1: Relación de herramientas software y versiones utilizadas

5.2.3. Herramientas hardware

- Hardware de virtualización

  Como hardware de virtualización se considera un ordenador personal de escritorio, con tecnología de virtualización AMD-V. Es usado para ejecutar las máquinas virtuales que proporcionan los servicios del sistema durante la fase de construcción.

- Terminales de desarrollo y pruebas

  El terminal de desarrollo es un ordenador de escritorio donde se ejecutan todas las herramientas de construcción necesarias. Como dispositivos de prueba para las aplicaciones de usuario se utilizan tanto el terminal de desarrollo como un dispositivo táctil Android.

5.2.4. Diagrama de despliegue

La Figura 5.1 contiene un diagrama de despliegue que muestra el entorno de desarrollo durante la última fase del proyecto. Contiene los nodos más importantes, los componentes incluidos en cada nodo y los protocolos de comunicación utilizados.

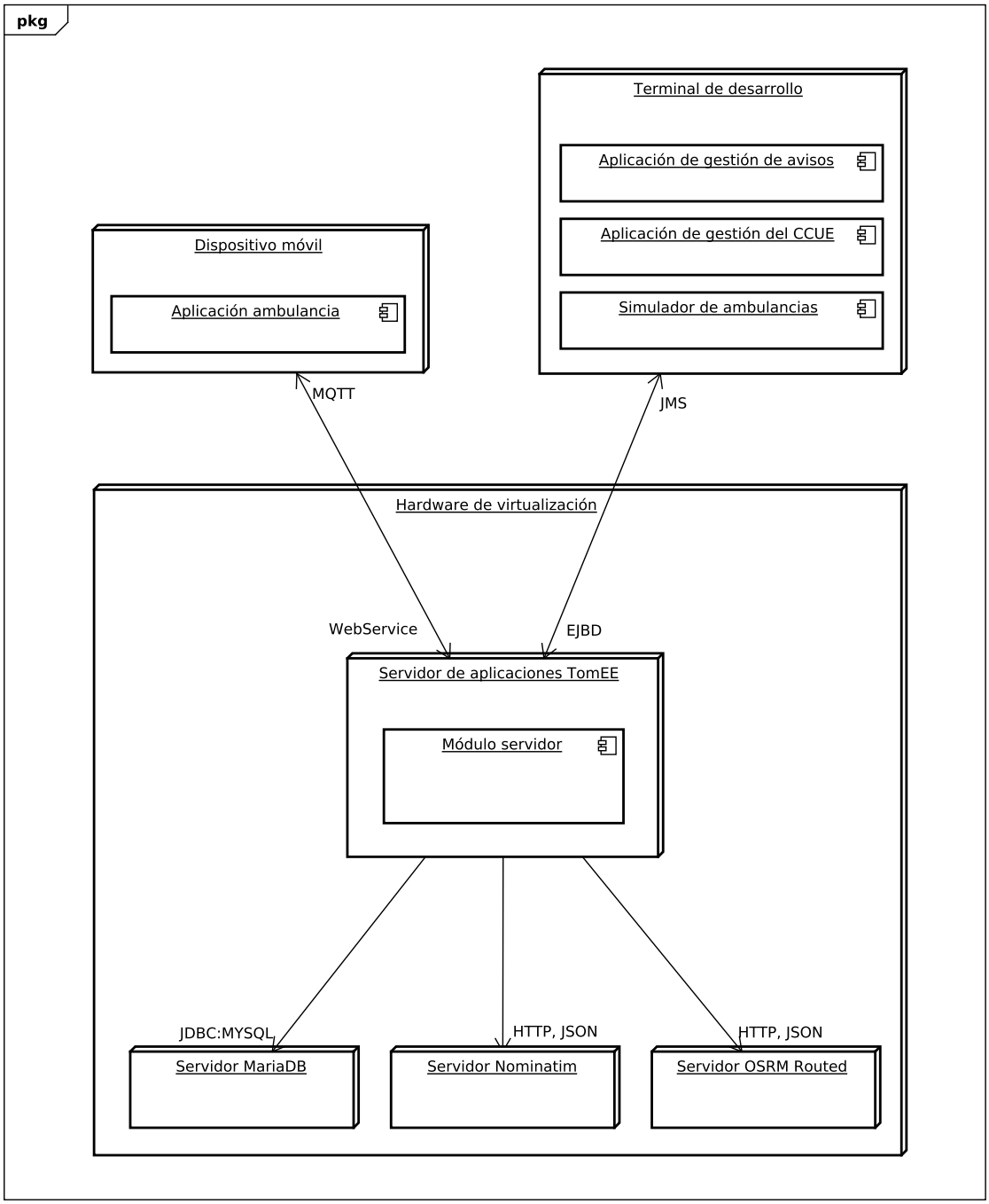


Figura 5.1: Diagrama de despliegue: entorno de desarrollo

5.3. Implementación

El conjunto de aplicaciones implementado es un reflejo de la arquitectura definida en el modelo de diseño. Para cada módulo de dicha arquitectura se ha creado una aplicación equivalente, de forma que el resultado final ofrezca una funcionalidad similar al diseño definido durante el Capítulo 4. Se han desarrollado dos aplicaciones auxiliares, utilizadas únicamente como herramientas de apoyo durante las pruebas y el despliegue.

5.3.1. Aplicación de servidor

La aplicación de servidor es una aplicación Java EE ejecutada sobre el servidor Apache TomEE. Corresponde dentro del modelo de diseño al módulo *server* e implementa cada uno los servicios definidos en forma de componentes EJB.

Los componentes EJB, o Enterprise Java Beans, son los elementos de una aplicación de tipo empresarial que ejecutan la lógica de negocio del lado de servidor. Su utilización para la creación de los servicios del sistema proporciona múltiples ventajas:

- Permite al desarrollador olvidarse de la inicialización de los servicios o de la creación de otros objetos internos.

  El servidor de aplicaciones gestiona el ciclo de vida de los componentes EJB, creándolos en el momento que se utilizan y destruyéndolos cuando ya no son necesarios. El contenedor EJB, encargado de controlar la ejecución del componente, permite a su vez la inyección de código dentro de la clase que declara el servicio, facilitando la instanciación automática de objetos.

- Gestiona de forma automática aspectos como la seguridad y la concurrencia.

  El contenedor EJB se ocupa de la identificación y sincronización de los hilos entrantes, de forma que las peticiones concurrentes sean seguras.

- Facilita las conexiones remotas tanto a través del servicio de descubrimiento JNDI como a través de servicios web.

  La clase que proporciona el servicio únicamente debe implementar una interfaz Java y anotarla como remota para hacer disponibles sus operaciones a clientes en otras máquinas. De forma similar, la definición de servicios web sólo requiere un conjunto de anotaciones Java adicionales, siendo el propio servidor de aplicaciones el que se encarga de la creación y puesta en marcha del servicio.

- Proporciona servicios de persistencia (JPA) y de mensajes (JMS).

  La implementación TomEE proporciona acceso al servicio de persistencia OpenJPA e integra el *broker* de mensajes ActiveMQ, compatible además con el protocolo MQTT, orientado a comunicaciones móviles.

La Figura 5.2 muestra una captura parcial del servidor TomEE con la aplicación desplegada.

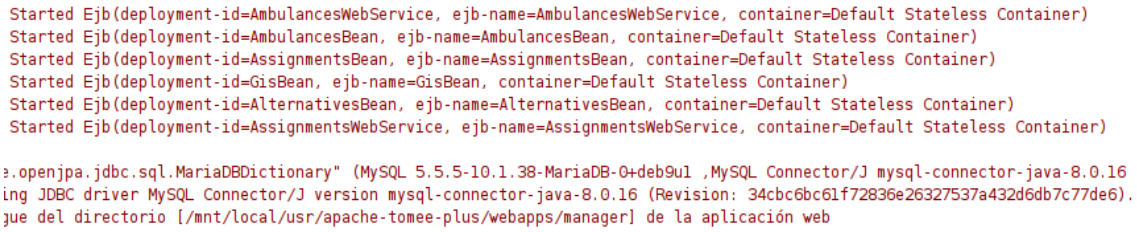


Figura 5.2: Captura de pantalla: Aplicación de servidor

En los apartados siguientes se detallan los aspectos más relevantes de la implementación para cada uno de los paquetes definidos.

Paquete *messaging*

Contiene la implementación del mecanismo de comunicación de mensajes utilizando las clases de la API JMS. Se han incluido en *ServerMessageConsumer* y *ServerMessageProducer* los objetos JMS necesarios para efectuar el intercambio de mensajes entre la aplicación y el *broker* ActiveMQ. El código referido al protocolo MQTT se ha añadido a la clase *ServerMessageConsumerMobile*, al ser la aplicación móvil el escenario principal de uso de dicho protocolo. Todo el código relacionado con el *broker* de mensajes está contenido en dichas clases y no es expuesto hacia el exterior.

Paquete *persistence*

Contiene la implementación del patrón genérico DAO sobre el objeto de gestión de entidades *EntityManager* de JPA. Las reglas de persistencia específicas para OpenJPA se incluyen en los archivos *META-INF/persistence.xml* y *META-INF/mapping.xml*. El uso de archivos *xml* facilita la definición de las entidades incluidas en el sistema de persistencia sin que las propias clases conozcan los detalles de su almacenamiento, actuando de esta forma tanto como objetos de acceso y como objetos de transferencia.

Paquete *remote*

Contiene el mecanismo básico que permite a los clientes realizar la conexión los contenedores EJB remotos. Incluye las clases *RemoteClient* y *RemoteClientContextFactory* que encapsulan el código específico de conexión al servidor de aplicaciones TomeEE usando el servicio de descubrimiento JNDI.

Paquete *notices*

Contiene la clase *NoticesBean*, correspondiente en el modelo de diseño a *NoticesService*. Esta clase define un componente EJB de tipo *Stateless*, el cual se ejecuta sobre el servidor TomEE únicamente en el momento que se recibe una petición. Esta ejecución no almacena ningún tipo de estado, permitiendo múltiples accesos concurrentes sin necesidad de establecer mecanismos de bloqueo.

Paquete *ambulances*

Contiene las clases *AmbulancesBean* y *AmbulancesWebService*.

- La clase *AmbulancesBean* es funcionalmente equivalente a *NoticesBean*, pero referida a ambulancias.
- La clase *AmbulancesWebService* define los métodos del servicio web para la conexión con la aplicación móvil de la ambulancia.

La definición de un nuevo servicio para el acceso desde la aplicación móvil es necesaria por dos razones. La primera es que las librerías Java disponibles para Android no soportan los sistemas de comunicaciones utilizados por la especificación Java EE. La segunda razón está influenciada por la naturaleza inalámbrica de las comunicaciones con la ambulancia, en las que se necesitan protocolos ligeros con una alta seguridad y privacidad.

Paquete *assignments*

Contiene las clases *AssignmentsBean*, *AssignmentsObserverBean* y *AssignmentsWebService*.

- La clase *AssigmentsBean* es similar a las otras dos clases de servicio *AmbulancesBean* y *NoticesBean*, pero en este caso referida a asignaciones.

- La clase *AssignmentsObserverBean* define un componente EJB de tipo *Singleton* que permite la ejecución constante de un objeto consumidor de mensajes sobre el tópico de avisos. De esta forma, cuando el sistema genera un mensaje de operación sobre un aviso, el observador lo recibe a través del objeto consumidor, permitiendo al servicio de gestión de asignaciones actuar automáticamente sobre la asignación asociada al aviso.
- La clase *AssignmentsWebService* define las operaciones de gestión de asignaciones que están disponibles a través del servicio web.

Paquete *gis*

Contiene la clase *GisBean*, que define un componente EJB de tipo *Stateless*. Este servicio permite a los clientes realizar peticiones de geolocalización de direcciones o cálculo de rutas. Estas peticiones son resueltas mediante solicitudes adicionales a los servidores externos OSRM y Nominatim, utilizando los clientes HTTP implementados en las clases *OsrmRoutedClient* y *NominatimClient*, respectivamente.

Paquete *zones*

Contiene la clase *ZonesBean*, que define un componente EJB de tipo *Stateless*. Es accedido desde el servicio de cálculo de alternativas para obtener los datos de zona almacenados.

5.3.2. Aplicaciones de cliente

Las aplicaciones de cliente se han implementado utilizando JavaFX. El uso de esta plataforma proporciona varias ventajas:

- Facilita la separación del código de vista y el código de controlador.

  Utiliza guiones FXML, los cuales permiten definir la distribución de los elementos de interfaz mediante archivos *xml*. Estos archivos son transformados en elementos gráficos en tiempo de ejecución y enlazados con los objetos controladores de forma automática. De esta forma el desarrollo se puede enfocar en implementar la lógica relacionada con eventos de usuario y los accesos a los servicios.
- Ofrece una alta integración con otras librerías y herramientas.

Permite el uso herramientas como SceneBuilder, un editor gráfico de guiones tipo WYSIWYG, y la integración de librerías de terceros o controles gráficos propios.

- Facilita la ejecución asíncrona de servicios remotos.

  Proporciona la clase *Service*, la cual se puede implementar para la ejecución de una petición asíncrona sobre un servidor remoto, devolviendo los resultados directamente en el hilo principal de la aplicación. Esto facilita en gran medida la gestión de accesos remotos en aplicaciones con interfaces gráficas, las cuales suelen imponer restricciones de ejecución multi-hilo. Este mecanismo actúa además como objeto delegado en la conexión con los módulos de servicio, encapsulando las clases de acceso remoto.

Para la visualización de la información geográfica se ha utilizado la librería Java MapsForge. Esta librería implementa un visualizador gráfico de mapas en formato vectorial, integrable tanto en aplicaciones de escritorio como en aplicaciones Android. Tiene las características siguientes:

- Utiliza como fuente de datos los mapas con licencia libre OpenStreetMap.
- Permite configurar los niveles de detalle renderizados.
- Permite mostrar marcadores, rutas y gráficos adicionales mediante capas superpuestas.
- Permite responder a las interacciones del operador, de forma limitada, pero suficiente para resolver localizaciones a coordenadas directamente sobre el mapa mostrado.

Aplicación de gestión de avisos

La aplicación de gestión de avisos es una aplicación JavaFX para terminales de escritorio. Corresponde dentro del modelo de diseño al módulo *client-notices* e implementa la lógica de creación y cancelación de avisos. La funcionalidad desarrollada es la siguiente:

- Mostrar un listado con los avisos recientes, a modo de histórico de la sesión.
- Registrar avisos nuevos en el sistema.
- Cancelar avisos ya registrados.
- Geolocalizar avisos mediante la resolución de direcciones a coordenadas.
- Geolocalizar avisos mediante la selección directa de la posición en el mapa.

La Figura 5.3 muestra una captura de pantalla de la aplicación.

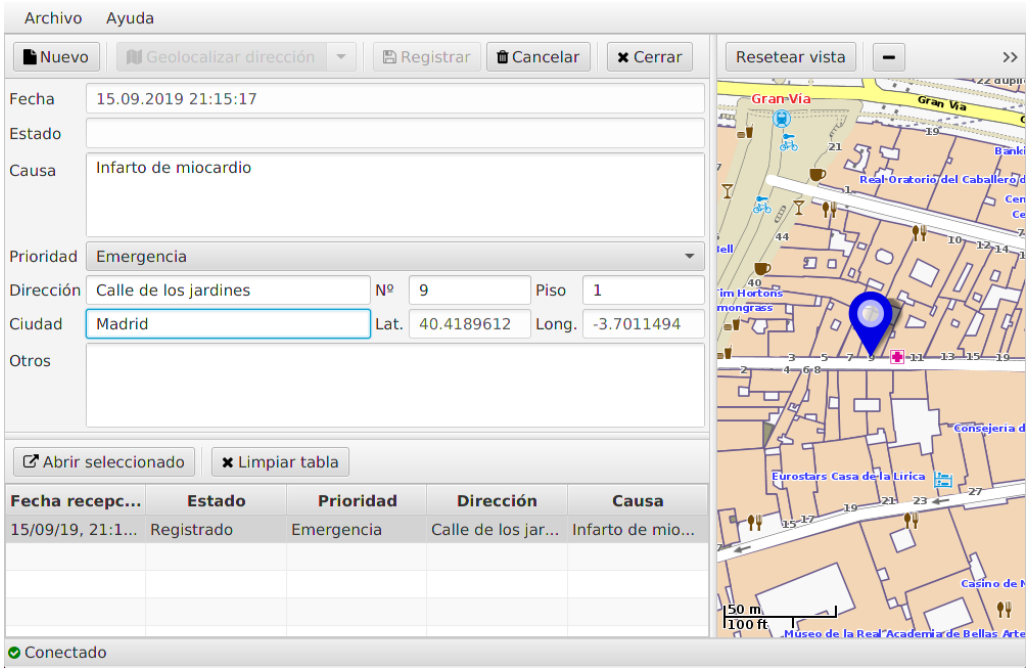


Figura 5.3: Captura de pantalla: Aplicación de gestión de avisos

Aplicación de gestión del CCUE

La aplicación de gestión del CCUE es una aplicación JavaFX destinada a ser ejecutada en un terminal de escritorio. Corresponde dentro del modelo de diseño al módulo *client-ccue* e implementa la lógica de los dos paquetes de dicho módulo, *assignments* y *ambulances*.

La funcionalidad de la parte de las asignaciones es la siguiente:

- Mostrar un listado con las asignaciones pendientes.
- Mostrar las alternativas para una asignación y permitir su registro.
- Monitorizar el estado de una asignación en curso, incluyendo la localización del aviso, la posición de la ambulancia o la ruta calculada.

La parte referida a la gestión de las ambulancias implementa las funciones:

- Registrar ambulancias nuevas.
- Modificar y eliminar las ambulancias existentes.
- Monitorizar el estado y posición de cada ambulancia.

Las Figuras 5.4 y 5.5 muestran capturas de pantalla de ambas secciones de la aplicación.

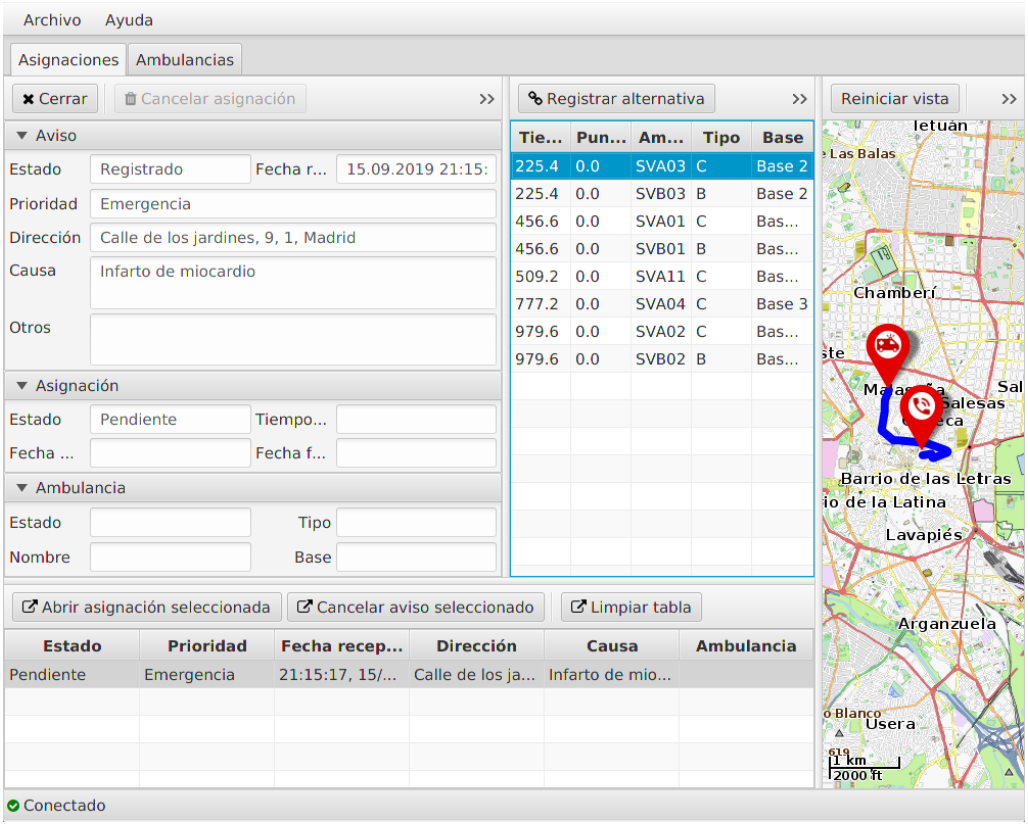


Figura 5.4: Captura de pantalla: Aplicación de gestión del CCUE (asignaciones)

Aplicación de control de la ambulancia

La aplicación de control de la ambulancia es una aplicación JavaFX destinada a ser ejecutada en un dispositivo móvil. Corresponde al módulo *client-ambulance* e implementa los mecanismos de control de la asignación en curso para las ambulancias. La funcionalidad desarrollada es la siguiente:

- Permanecer a la escucha de un mensaje de asignación registrada, y mostrar una notificación en el momento que se recibe.
- Mostrar la información de la asignación, incluida la localización geográfica del aviso, la posición de la propia ambulancia y una posible ruta.
- Gestionar el estado de ejecución de la asignación asociada.
- Gestionar el estado de la propia ambulancia.
- Gestionar la autenticación de la ambulancia en el sistema.

La Figura 5.6 muestra una captura de la pantalla principal de la aplicación.

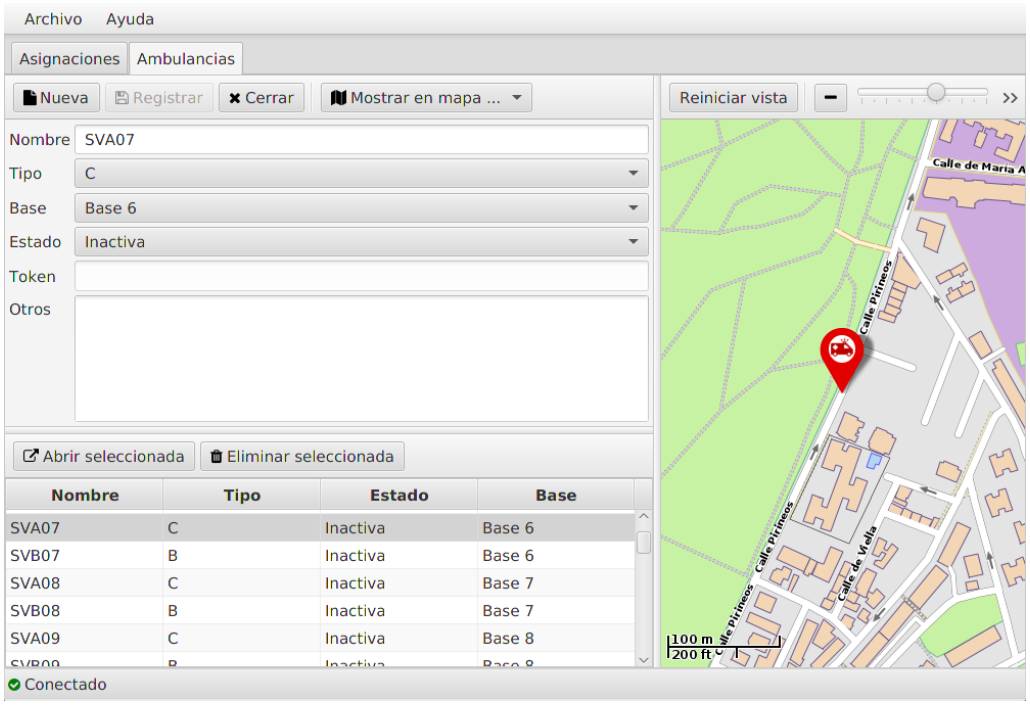


Figura 5.5: Captura de pantalla: Aplicación de gestión del CCUE (ambulancias)

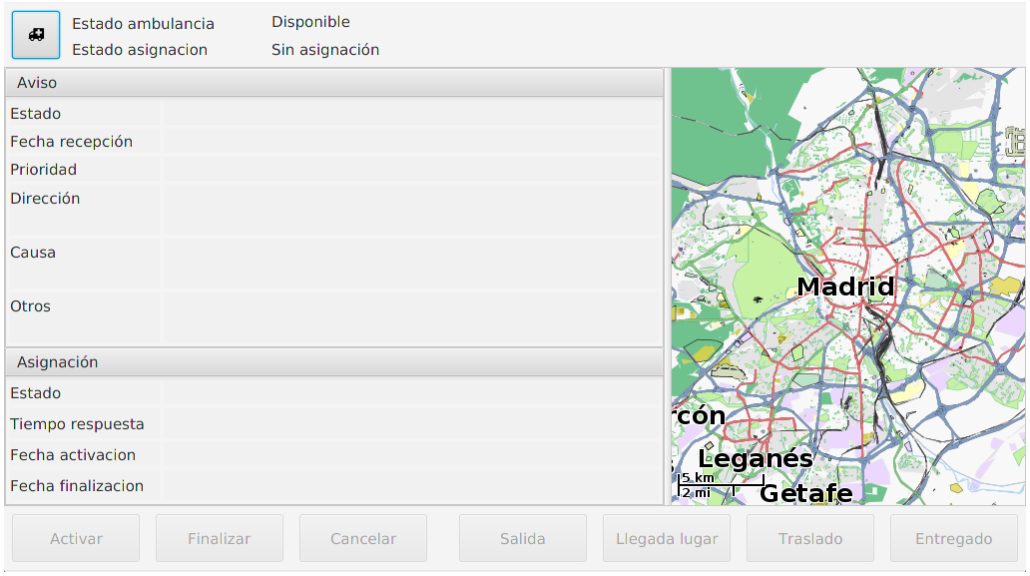


Figura 5.6: Captura de pantalla: Cliente de control de ambulancia

5.3.3. Aplicaciones auxiliares

Durante la construcción del sistema se han desarrollado dos aplicaciones adicionales: un simulador de ambulancias y un configurador de zonas.

Simulador de ambulancias

El simulador de ambulancias es una aplicación JavaFX que se encarga de gestionar de forma automática el comportamiento de una o varias ambulancias, actuando como si estuvieran operadas por personal real. Es especialmente útil durante las pruebas de validación, donde es necesario crear un escenario lo más parecido al objetivo, con múltiples ambulancias y asignaciones. Permite además visualizar en tiempo real el estado de cualquiera de las ambulancias simuladas facilitando las comprobaciones durante las tareas de prueba.

La Figura 5.7 muestra una captura de la pantalla principal de la aplicación.

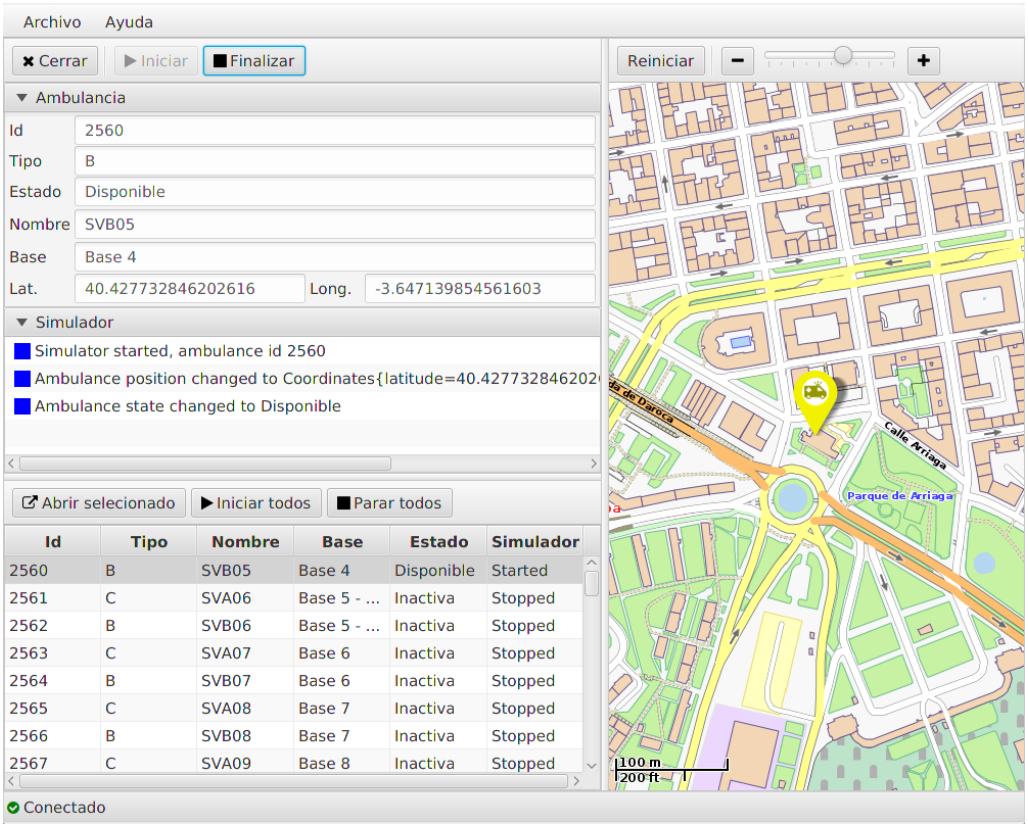


Figura 5.7: Captura de pantalla: Simulador de ambulancias

Configurador de zonas

El configurador de zonas, es una herramienta destinada al proceso de despliegue del sistema. Permite el cálculo automático de los datos relacionados con las zonas del territorio y los tiempos de respuesta entre zonas para un municipio concreto. El resultado se proporciona en forma de fichero de texto de órdenes SQL, el cual puede ser importado en una base de datos vacía.

La Figura 5.8 muestra una captura de la pantalla principal de la aplicación.

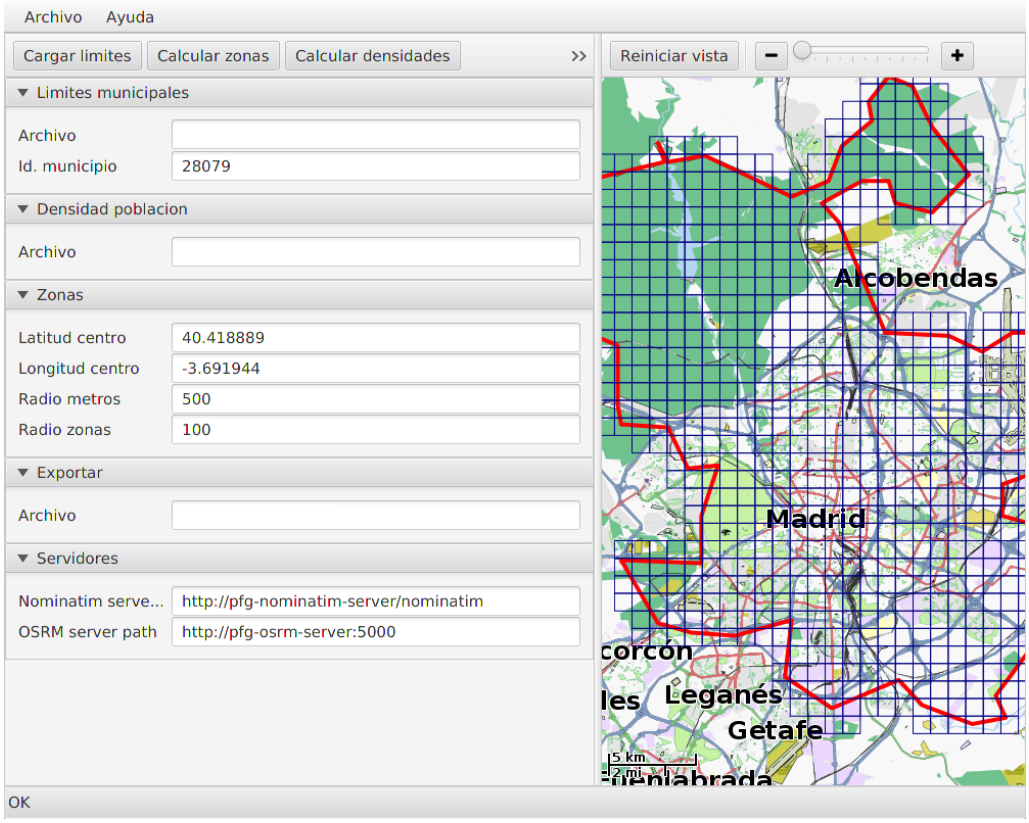


Figura 5.8: Captura de pantalla: Configurador de zonas

5.4. Pruebas

En esta sección se detallan las estrategias de prueba utilizadas durante la construcción del proyecto. En primer lugar se analizan los mecanismos de prueba unitaria y de prueba de integración. A continuación se describe el conjunto de pruebas de validación desarrolladas con el objetivo de validar el producto construido.

5.4.1. Pruebas unitarias

Una prueba unitaria es un mecanismo de prueba repetible y automatizable que permite la comprobación de unidades de código de una forma aislada. Tiene el objetivo de detectar los errores de implementación durante el desarrollo, evitando que se conviertan en defectos.

La estrategia seguida durante la implementación del proyecto ha sido construir las pruebas unitarias necesarias para cubrir la mayor parte del código referido a la lógica de negocio, donde existe un mayor riesgo de fallo. Las pruebas implementadas se pueden ver en el código fuente de cada módulo, dentro del directorio *src/test*.

La ejecución de este tipo de pruebas se realiza de forma automática antes de cada compilación completa. Si la compilación es correcta, se asume que el software construido está libre de los errores que las pruebas desarrolladas son capaces de detectar.

5.4.2. Pruebas de integración

Las pruebas de integración permiten comprobar que el sistema construido se comporta adecuadamente en su conjunto. Son realizadas tras las pruebas unitarias y permiten detectar errores durante las operaciones en las que intervienen varios elementos distintos, normalmente varios subsistemas. Suelen encontrar errores de incompatibilidad en los datos de entrada/salida o fallos en las comunicaciones.

La estrategia seguida es similar a la de las pruebas unitarias, tratando de cubrir la mayor parte del código del servidor. Sin embargo, no se ejecutan de forma automática en cada compilación del código, al requerir de un tiempo de ejecución mucho más elevado que para las pruebas unitarias. Son ejecutadas de forma regular bien manualmente o a través de un servidor de integración continua. Las pruebas implementadas están contenidas en el subdirectorio *src/integration*.

5.4.3. Pruebas de validación

Las pruebas de validación permiten probar que el sistema cumple con los requisitos especificados inicialmente. Son las últimas pruebas realizadas durante el desarrollo y son ejecutadas manualmente antes de finalizar una nueva revisión. Suelen detectar errores en comportamiento de la interfaz gráfico o fallos en la integración de los componentes de más alto nivel. Pueden ser realizados por el propio equipo de desarrollo o por personal dedicado a procesos de control de calidad.

Se han definido un conjunto de pruebas que permiten validar los requisitos descritos en la Sección 3.3, organizadas en tres grupos, una por cada tipo de aplicación de cliente.

Las pruebas se han ejecutado utilizando los datos del escenario detallado en el Anexo A, centrado en el Municipio de Madrid. Para las pruebas en las que el objeto no es la ambulancia, se ha utilizado como apoyo la aplicación auxiliar de simulación de forma que las respuestas sean lo más similares a lo esperado en un escenario real.

Pruebas de la aplicación de gestión de avisos

- **PV1:** Registro de aviso correcto.

  Acciones Registrar un aviso nuevo con los datos de prioridad y localización correctamente cubiertos.

  Resultados esperados La aplicación de gestión de avisos muestra una notificación de aviso registrado.

  La aplicación de gestión del CCUE muestra una nueva asignación pendiente asociada al aviso registrado.

- **PV2:** Registro de aviso fallido.

  Acciones Registrar un aviso nuevo sin el valor de prioridad cubierto.

  Resultados esperados La aplicación de gestión de avisos muestra una notificación de error especificando que el valor de prioridad no es válido.

- **PV3:** Registro de aviso fallido.

  Acciones Registrar un aviso nuevo sin los datos de localización cubiertos.

Resultados esperados La aplicación de gestión de avisos muestra una notificación de error especificando que los datos de localización no son válidos.

- **PV4:** Cancelación de un aviso registrado.

  Precondiciones El aviso abierto está registrado.

  Acciones Cancelar el aviso abierto y aceptar la solicitud de confirmación.

  Resultados esperados La aplicación de gestión de avisos muestra una notificación de aviso cancelado.

  La aplicación de gestión del CCUE actualiza el estado de la asignación asociada al aviso mostrándola como cancelada.

Pruebas de la aplicación de control de la ambulancia

- **PV5:** Autenticación correcta.

  Acciones Autenticar la ambulancia utilizando datos válidos.

  Resultados esperados La aplicación de control de la ambulancia muestra una notificación de autenticación realizada.

  El panel de información actualiza los datos de la ambulancia y limpia el campo de entrada de credenciales.

- **PV6:** Autenticación fallida.

  Acciones Autenticar la ambulancia utilizando datos erróneos.

  Resultados esperados La aplicación de control de la ambulancia muestra una notificación de error informando que las credenciales no son válidas.

- **PV7:** Autenticación automática tras el inicio.

  Precondiciones La autenticación manual se ha realizado con éxito al menos una vez en el dispositivo.

  Acciones Iniciar la aplicación.

  Resultados esperados La aplicación de control de la ambulancia actualiza automáticamente los datos de la ambulancia y su estado.

- **PV8:** Cambio de estado a disponible.

  Precondiciones La ambulancia está inactiva y sin asignación asociada.

  Acciones Establecer el estado de la ambulancia a disponible.

  Resultados esperados La aplicación de control de la ambulancia cambia su estado a disponible.

- **PV9:** Cambio de estado a no disponible correcto.

  Precondiciones La ambulancia está disponible y sin asignación asociada.

  Acciones Establecer el estado de la ambulancia a no disponible.

  Resultados esperados La aplicación de control de la ambulancia cambia su estado a inactiva

- **PV10:** Cambio de estado a no disponible fallido.

  Precondiciones La ambulancia está disponible y con una asignación registrada.

  Acciones Establecer el estado de la ambulancia a no disponible.

  Resultados esperados La aplicación de control de la ambulancia muestra una notificación de error informando que existe una asignación registrada.

- **PV11:** Activación de una asignación registrada.

  Precondiciones La aplicación de control de ambulancia ha recibido una asignación nueva. La asignación recibida está registrada.

  Acciones Activar la asignación.

  Resultados esperados La aplicación de control de la ambulancia cambia el estado de la asignación a activada y actualiza la hora de activación.

  El panel de información de la ambulancia cambia su estado a activa.

- **PV12:** Cancelación de una asignación registrada.

  Precondiciones La aplicación de control de ambulancia ha recibido una asignación nueva. La asignación recibida está registrada.

  Acciones Cancelar la asignación y aceptar la solicitud de confirmación.

Resultados esperados La aplicación de control de la ambulancia cambia el estado de la asignación a cancelada.

- **PV13:** Cambio de estado de la ambulancia a salida.

  Precondiciones La aplicación de control de ambulancia tiene una asignación abierta. La asignación abierta está activa.

  Acciones Establecer el estado de la ambulancia a salida.

  Resultados esperados La aplicación de control de la ambulancia cambia su estado a salida.

- **PV14:** Cambio de estado de la ambulancia a llegada al lugar.

  Precondiciones La aplicación de control de ambulancia tiene una asignación abierta. La asignación abierta está activa.

  Acciones Establecer el estado de la ambulancia a llegada al lugar.

  Resultados esperados La aplicación de control de la ambulancia cambia su estado a llegada al lugar.

- **PV15:** Cambio de estado de la ambulancia a traslado.

  Precondiciones La aplicación de control de ambulancia tiene una asignación abierta. La asignación abierta está activa.

  Acciones Establecer el estado de la ambulancia a traslado.

  Resultados esperados La aplicación de control de la ambulancia cambia su estado a traslado.

- **PV16:** Cambio de estado de la ambulancia a entregado.

  Precondiciones La aplicación de control de ambulancia tiene una asignación abierta. La asignación abierta está activa.

  Acciones Establecer el estado de la ambulancia a entregado.

  Resultados esperados La aplicación de control de la ambulancia cambia su estado a entregado.

- **PV17:** Finalización de una asignación activa.

  Precondiciones La aplicación de control de ambulancia tiene una asignación abierta. La asignación abierta está activa.

  Acciones Finalizar la asignación y aceptar la solicitud de confirmación.

  Resultados esperados La aplicación de control de la ambulancia cambia el estado de la asignación a finalizada y actualiza la hora de finalización.

- **PV18:** Cancelación de una asignación activa.

  Precondiciones La aplicación de control de ambulancia tiene una asignación abierta. La asignación abierta está activa.

  Acciones Cancelar la asignación y aceptar la solicitud de confirmación.

  Resultados esperados La aplicación de control de la ambulancia cambia el estado de la asignación a cancelada.

Pruebas de la aplicación de gestión del CCUE

- **PV19:** Apertura de una asignación pendiente.

  Precondiciones La asignación seleccionada en la lista está pendiente.

  Acciones Abrir la asignación seleccionada.

  Resultados esperados La aplicación de gestión del CCUE muestra la información del aviso, la asignación y la ambulancia en el panel de información.

  El visualizador de mapas centra su posición en las coordenadas geográficas del aviso.

- **PV20:** Obtención de las alternativas para una asignación.

  Precondiciones La asignación abierta está pendiente.

  Acciones Solicitar la obtención de las alternativas de asignación.

  Resultados esperados La aplicación de gestión del CCUE muestra una lista de alternativas para la asignación ordenadas.

- **PV21:** Registro de una alternativa.

  Precondiciones La asignación abierta está pendiente. La lista de alternativas contiene al menos un elemento.

  Acciones Seleccionar una alternativa y solicitar su registro.

  Resultados esperados La aplicación de gestión del CCUE muestra una notificación de alternativa registrada.

  La aplicación de gestión del CCUE muestra los datos de la asignación actualizados.

  La aplicación de control de la ambulancia muestra una notificación de asignación recibida.

- **PV22:** Monitorización de una asignación registrada.

  Precondiciones La asignación seleccionada en la lista está registrada.

  Acciones Abrir la asignación seleccionada.

  Resultados esperados La aplicación de gestión del CCUE actualiza automáticamente el panel de información mostrando los cambios en el estado de la ambulancia y la asignación.

  El visualizador de mapas muestra la información geográfica de los elementos asociados a la asignación.

- **PV23:** Cancelación de una asignación registrada o activa.

  Precondiciones La asignación abierta está registrada o activa.

  Acciones Cancelar la asignación abierta.

  Resultados esperados La aplicación de gestión del CCUE muestra una notificación de asignación cancelada.

  El panel de información actualiza el estado de la asignación. La lista de asignaciones muestra una nueva asignación pendiente.

  La aplicación de control de la ambulancia muestra una notificación de asignación cancelada.

- **PV24:** Cancelación de un aviso asociado a una asignación pendiente.

  Precondiciones La asignación seleccionada en la lista está pendiente y asociada a un aviso no cancelado.

  Acciones Cancelar el aviso seleccionado.

  Resultados esperados La aplicación de gestión del CCUE muestra una notificación de aviso cancelado. La entrada seleccionada actualiza su estado.

- **PV25:** Cancelación de un aviso asociado a una asignación registrada o activa.

  Precondiciones La asignación seleccionada en la lista está registrada o activa y está asociada a un aviso no cancelado.

  Acciones Cancelar el aviso seleccionado.

  Resultados esperados La aplicación de gestión del CCUE muestra una notificación de aviso cancelado. La entrada seleccionada actualiza su estado.

  La aplicación de control de la ambulancia muestra una notificación de asignación cancelada.

- **PV26:** Registro de una ambulancia.

  Acciones Iniciar el registro de una ambulancia, rellenar los datos correspondientes y registrar los cambios.

  Resultados esperados La aplicación de gestión del CCUE muestra una notificación de registro realizado.

- **PV27:** Modificación de una ambulancia registrada.

  Precondiciones La ambulancia abierta está registrada.

  Acciones Modificar los datos de la ambulancia y registrar los cambios.

  Resultados esperados La aplicación de gestión del CCUE muestra una confirmación de ambulancia actualizada.

- **PV28:** Eliminación de una ambulancia correcta.

  Precondiciones La ambulancia seleccionada está en estado no disponible.

  Acciones Eliminar la ambulancia seleccionada.

Resultados esperados La aplicación de gestión del CCUE muestra una confirmación de ambulancia eliminada.

- **PV29:** Eliminación de ambulancia fallida.

  Precondiciones La ambulancia seleccionada está en un estado distinto a no disponible.

  Acciones Eliminar la ambulancia seleccionada.

  Resultados esperados La aplicación de gestión del CCUE muestra una notificación de error especificando que solamente se puede eliminar una ambulancia en estado no disponible.

5.5. Conclusiones

En este capítulo se ha descrito el sistema objetivo tal como se encuentra en la última revisión del código. Para cada aplicación implementada se han detallado las tecnologías utilizadas y la funcionalidad desarrollada. En la última sección se han analizado las estrategias de prueba utilizadas, las cuales permiten asegurar que el sistema construido cumple con los requisitos iniciales.

Capítulo 6

Planificación y costes del proyecto

6.1. Introducción

En este capítulo se describe la planificación temporal y los costes estimados del proyecto. En primer lugar se detallan las fases en las que se ha organizado el trabajo, especificando para cada fase las tareas planificadas y los resultados esperados. A continuación se incluyen dos cronogramas con la planificación inicial y la real. Por último se incluyen los costes estimados del proyecto.

6.2. Planificación temporal

6.2.1. Fases del proyecto

Para el desarrollo de este proyecto se ha utilizado un metodología basada en las fases y disciplinas del Proceso Unificado pero con un ciclo de vida en una única iteración, más adecuado para un proyecto de tamaño reducido y con un plazo de tiempo fijo.

Las fases que contempla esta metodología son cuatro: la fase inicial, la fase de elaboración, la fase de construcción y la fase de transición [LV03].

- Fase inicial

  En esta fase se lleva a cabo un estudio introductorio del problema, se modelan los procesos de negocio y se definen los requisitos del sistema a desarrollar. El resultado es un documento que incluye:

- Una visión aproximada.
- Una especificación del sistema.
- Las especificaciones adicionales necesarias.

- Fase de elaboración

  En esta fase se realiza el diseño de la arquitectura del sistema, al mismo tiempo que se mejora y amplia el resultado de la fase inicial. Incluye los siguientes artefactos:

  - El modelo de diseño.
  - El modelo de datos.
  - Un modelo de pruebas preliminar.

- Fase de construcción

  En esta fase se llevan a cabo la implementación y las pruebas del sistema. Tiene como resultado:

  - El refinamiento del modelo de diseño.
  - Un modelo de pruebas actualizado.
  - Un conjunto de versiones con funcionalidad incremental.
  - El desarrollo de un modelo de despliegue.

- Fase de transición

  En este punto, el sistema se encuentra en una revisión finalizada y está lista para ser utilizada. Se consideran las siguientes actividades:

  - El despliegue del sistema construido.
  - La escritura de la documentación del proyecto.

6.2.2. Diagramas de Gantt

Las Figuras 6.1 y 6.2 muestran dos diagramas de Gantt con la planificación inicial y la real. Cada diagrama incluye las fechas de las fases, las tareas más importantes y los entregables programados.

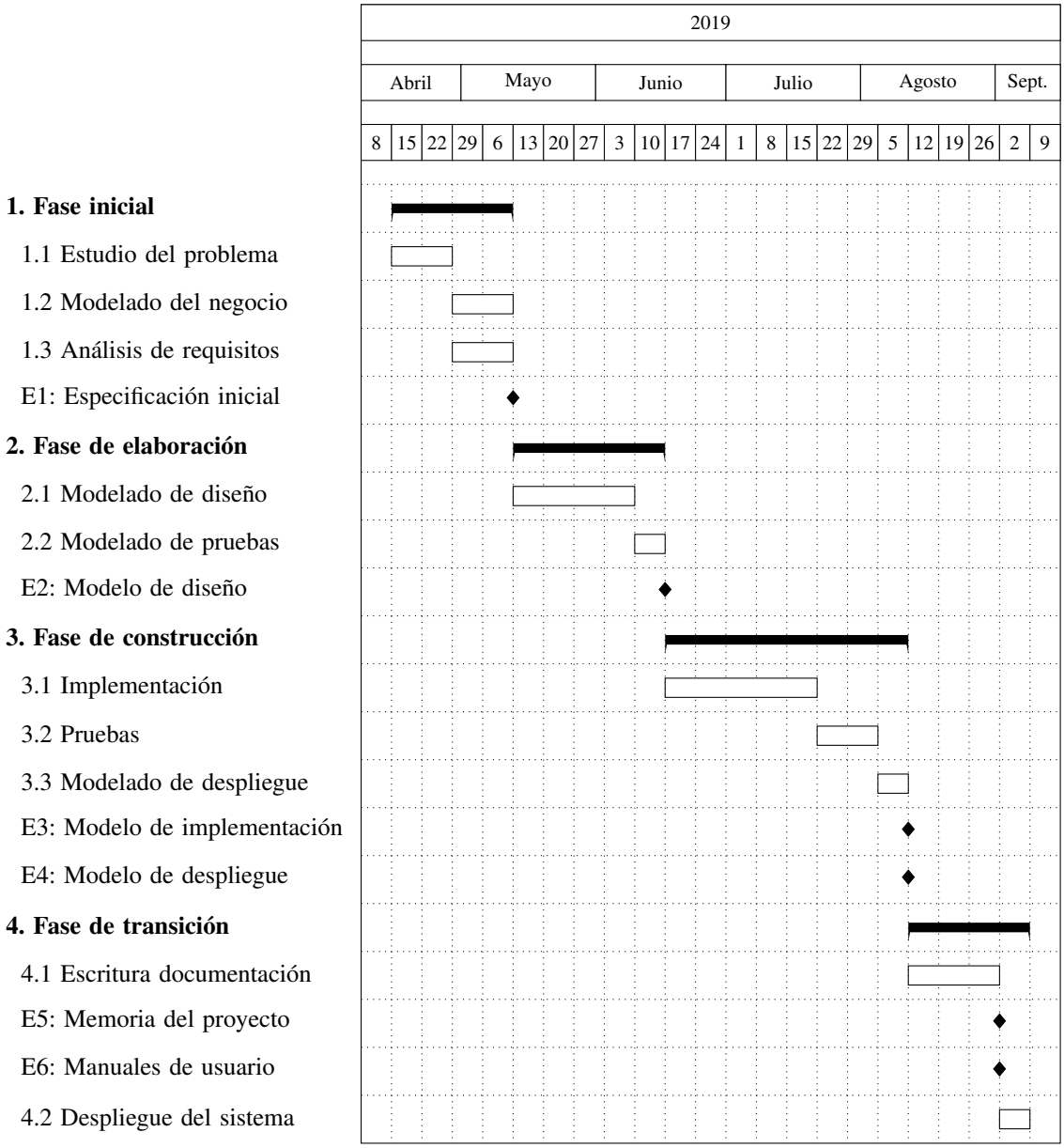


Figura 6.1: Diagrama de Gantt: planificación temporal inicial

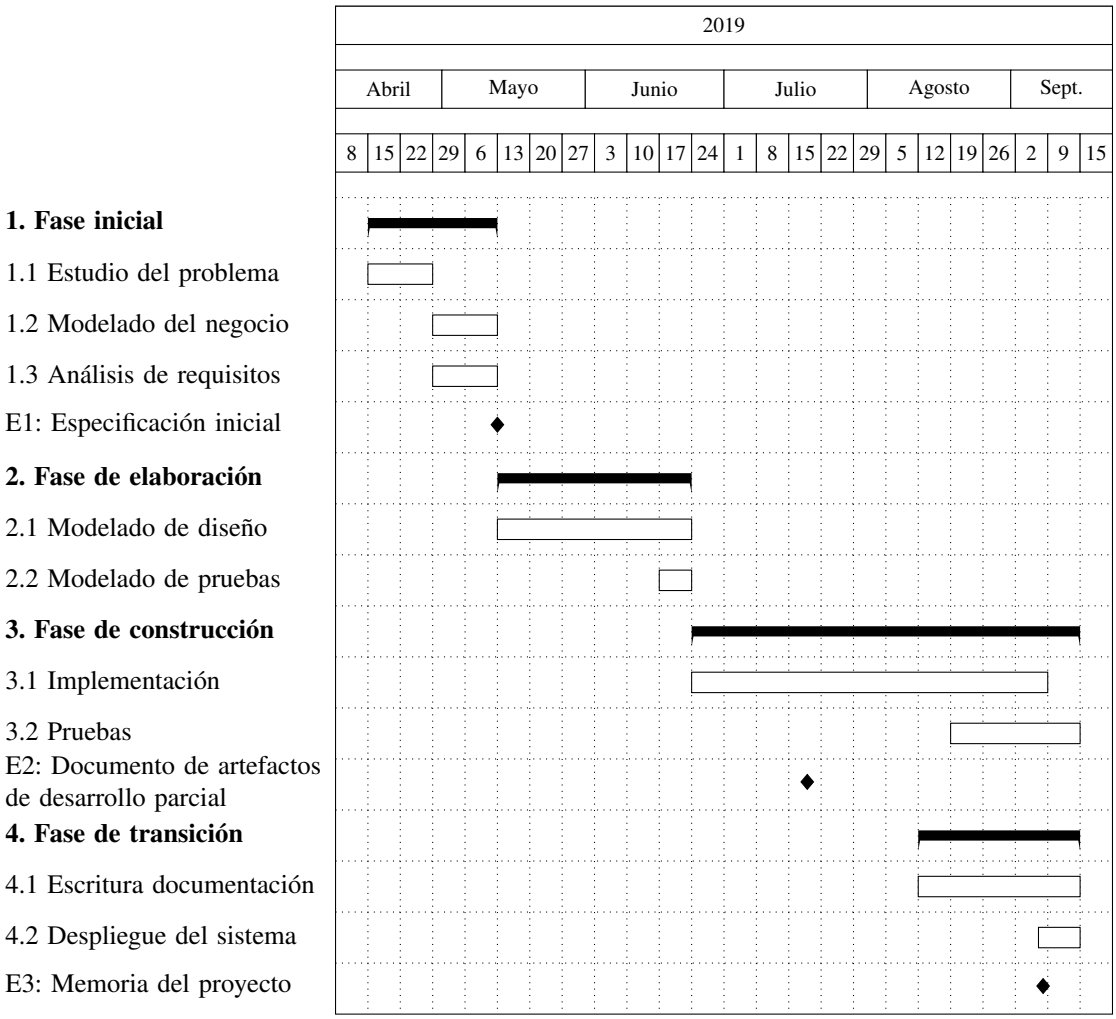


Figura 6.2: Diagrama de Gantt: planificación temporal real

6.3. Costes del proyecto

Los costes considerados son las horas de desarrollo, las licencias software y el equipamiento hardware que se ha dedicado en exclusiva al proyecto. Estos se pueden ver en la Tabla 6.1.

Concepto	Unidades	Precio/ud.	Total
Horas totales	476	60 €	28.560 €
Licencia software IntelliJ IDEA Ultimate	1	499 €	*0 €
Licencia software Astah UML	1	53 €	*0 €
Ordenador virtualización AMD	1	351 €	351 €
			28.911 €

Tabla 6.1: Tabla de costes del proyecto

Las entradas marcadas con un asterisco corresponden a licencias de estudiante. Los precios de las licencias regulares se incluyen como apunte pero no se contabilizan en el total.

Capítulo 7

Conclusiones y trabajos futuros

7.1. Introducción

En este capítulo se hace una revisión del proyecto desde el punto de vista del trabajo realizado, los resultados obtenidos y las posibles ampliaciones.

Contiene las Secciones Conclusiones y Trabajos futuros.

7.2. Conclusiones

El principal objetivo de este proyecto es la construcción de un sistema de apoyo a la gestión de ambulancias. Este sistema debe permitir:

- La gestión de los avisos de emergencias y urgencias.
- La gestión y optimización de las asignaciones.
- La gestión y el control de las ambulancias.

Tal como se ha establecido en la introducción de esta memoria, la optimización de las asignaciones es un requisito importante. El mecanismo de apoyo a la decisión debe proporcionar para un aviso concreto aquellas alternativas de asignación que considere más adecuadas, en función de los datos disponibles. El trabajo realizado para cumplir este requisito ha consistido en la definición de un modelo de datos, un modelo de organización del territorio y un conjunto de criterios de optimización.

El modelo de datos definido incluye los conceptos y propiedades necesarios para la gestión de los avisos, las ambulancias y las asignaciones. Los avisos permiten establecer las características de la incidencia médica y las asignaciones permiten relacionar ambulancias y avisos entre si. Tanto los avisos como las ambulancias están geolocalizadas.

El modelo de organización del territorio divide el terreno en un conjunto de zonas cuadriculares del mismo tamaño. El uso de múltiples zonas facilita una gestión eficiente de la información, simplificando los cálculos necesarios para la valoración de las asignaciones.

Los criterios de optimización consisten en la minimización del tiempo de respuesta entre zonas y la maximización de la capacidad de actuación en el territorio. El criterio de minimización del tiempo de respuesta permite asegurar que la atención sanitaria se realiza lo antes posible. El criterio de maximización de la capacidad de actuación tiene el objetivo de mantener unos niveles de calidad del servicio adecuados en todo el territorio.

Las valoraciones proporcionadas por el sistema de apoyo facilitan la tarea de elección de la ambulancia más adecuada para un determinado aviso. Hay que tener en cuenta que esta tarea no es automática, siendo el operador del centro de coordinación el que debe validarla. Esta es una característica habitual en los sistemas de apoyo a la asignación computerizados y evita que las decisiones críticas queden exclusivamente en las manos del software.

Otros de los trabajos llevados a cabo durante este proyecto han sido los relacionados con el proceso de ingeniería del software. Estos son: el análisis, el diseño, la implementación y las pruebas.

La tarea de análisis ha permitido establecer una descripción inicial del sistema objetivo, sin necesidad de utilizar conceptos técnicos que compliquen dicha descripción.

La tarea de diseño ha permitido definir de forma completa la arquitectura lógica de una posible solución que satisfaga la descripción inicial.

La tarea de implementación ha consistido en traducir los componentes definidos en la arquitectura en unidades de código ejecutable. Algunas de las herramientas y tecnologías utilizadas para esta tarea son:

- La plataforma Java EE y el servidor TomEE. Usados para la construcción y ejecución de la aplicación de procesamiento de datos del centro de coordinación.
- La plataforma JavaFX. Usada para el construcción y ejecución de las aplicaciones de usuario del centro de coordinación.

- La plataforma Android y la librería JavaFXPorts. Usados para la construcción y ejecución de la aplicación de usuario para la ambulancia.
- El servidor de bases de datos MariaDB. Usado para el almacenamiento persistente de información.
- La librería de visualización de mapas vectoriales MapsForge y los mapas con licencia libre OpenStreetMap. Usados para localización de los elementos de gestión directamente sobre el territorio.
- Los servidores de geolocalización Nominatim y de cálculo de rutas OSRM Routed. Usados para la resolución de localizaciones geográficas y rutas por carretera.

La tarea de pruebas ha sido paralela al trabajo de implementación y ha consistido en la evaluación continua del sistema implementado, utilizando para ello distintas estrategias y herramientas.

El resultado del proyecto es un sistema distribuido, formado por un conjunto de aplicaciones informáticas, que se comunican entre si y con otros sistemas externos, y permiten desarrollar las tareas de control y gestión descritas en los objetivos.

7.3. Trabajos futuros

Entre las posibles mejoras o ampliaciones se han considerado cuatro alternativas:

- La creación de una plataforma de acceso web para operadores de llamadas externos al centro de coordinación.
- La creación de una aplicación móvil para la comunicación directa entre los ciudadanos y el sistema de gestión de avisos.
- La mejora de la aplicación de control de la ambulancia, incluyendo un navegador de rutas de carretera con indicaciones por voz.
- La ampliación de la aplicación de gestión de avisos, integrando un manual médico completo que facilite la evaluación de las incidencias.

Como un posible trabajo derivado se plantea un sistema que realice una distribución óptima de las bases de ambulancias a lo largo de un territorio. Este problema es en realidad la especialización en torno a las ambulancias del problema de computación MCLP, o Maximal Covering Location Problem.

El problema MCLP [CR74], consiste en el cálculo de un conjunto de localizaciones que permitan maximizar la cobertura de los servicios situados en dichas localizaciones. Tiene el principal inconveniente de ser NP-completo, aunque en la actualidad existen diversas aproximaciones que permiten obtener soluciones óptimas.

En [ZDS11] se describe un algoritmo genético capaz de obtener soluciones con suficiente exactitud con hasta 2500 nodos. Este algoritmo define como parámetros de entrada la posición y población de cada nodo, las distancias entre nodos, la distancia cubierta por cada servicio y el número de servicios esperado. Obtiene como resultados las posiciones de los servicios de forma que se maximice la población cubierta por al menos un servicio.

Como se puede comprobar, los parámetros utilizados por este algoritmo son lo suficientemente genéricos como para poder aplicarlo en el contexto de las ambulancias, utilizando el modelo de datos definido en este proyecto o con una variación del mismo.

Bibliografía

[112] Memoria de actividades año 2018, 112 Región de Murcia. http://www.112rmurcia.es/dgsce/docs/memoria2018.pdf.

[AP07] Tobias Andersson and Sverker Petersson. Decision support for efficient ambulance logistics. 2007.

[ban] Banco de Datos del Ayuntamiento de Madrid. Superficie, Población y Densidad de los Distritos y Barrios de la Ciudad de Madrid. http://www-2.munimadrid.es/CSE6/control/seleccionDatos?numSerie=14010100012.

[BOE] BOE, 8 de Junio del 2012, Real Decreto 836/2012, de 25 de mayo. https://www.boe.es/boe/dias/2012/06/08/pdfs/BOE-A-2012-7655.pdf.

[cen] SAMUR - Protección Civil, Central de Comunicaciones 112. https://www.madrid.es/portales/munimadrid/es/Inicio/Aviso-Legal/SAMUR-Proteccion-Civil/?vgnextfmt=default&vgnextoid=c88fcdb1bfffa010VgnVCM100000d90ca8c0RCRD&vgnextchannel=8a0f43db40317010VgnVCM100000dc0ca8c0RCRD&idCapitulo=10270212.

[Cha] Change Vision, Inc. Astah UML is a UML editor which supports UML2.1 diagrams and Mind Map. Code Generation and reverse engineering supported. http://astah.net/editions/uml-new.

[cnd] Centro de Descargas del Centro Nacional de Información Geográfica. http://centrodedescargas.cnig.es/CentroDescargas/equipamiento.do?method=descargarEquipamiento&codEquip=3.

[CR74] Richard Church and Charles ReVelle. The maximal covering location problem. *Papers of the Regional Science Association*, 32(1):101–118, Dec 1974.

[dlR14] Rodolfo Arroyo de la Rosa. *El enlace en las emergencias*. 2014.

[dS06] Servicio Andaluz de Salud. *Protocolos de coordinación de la asistencia extrahospitalaria urgente y emergente del Sistema Sanitario Público de Andalucía.* 2006.

[Funa] Fundación OpenStreetMap. Nominatim, a tool to search OSM data by name and address and to generate synthetic addresses of OSM points. https://www.nominatim.org.

[Funb] Fundación OpenStreetMap. OpenStreetMap, Derechos de autor y licencia. https://www.openstreetmap.org/copyright.

[Glua] Gluon. JavaFX on Mobile and Embedded. Open Source, Open Community. Commercially Supported. https://gluonhq.com/products/mobile/javafxports/.

[Glub] Gluon. OpenJFX is an open source, next generation client application platform for desktop, mobile and embedded systems built on Java. https://openjfx.io.

[GV07] Tobias Andersson Granberg and Peter Värbrand. Decision support tools for ambulance dispatch and relocation. *Journal of the Operational Research Society*, 2007.

[Ing] Ingenia. eCALLER Ambulancias, sistema de control de flotas de ambulancias. https://www.ingenia.es/productos/ecaller-ambulancias.

[Jet] JetBrains. IntelliJ IDEA: The Java IDE for Professional Developers. https://www.jetbrains.com/idea/.

[KL16] Lorinde Knoops and Tilda Lundgren. Modeling ambulance dispatching rules for ems-systems. 2016.

[LV03] Craig Larman and Begoña Moros Valle. *UML y patrones: una introducción al análisis y diseño orientado a objetos y al proceso unificado*. Pearson Educación, 2 edition, 2003.

[mapa] Mapshaper is an editor for map data: Shapefile, GeoJSON, TopoJSON and CSV files . https://mapshaper.org/.

[mapb] Vector map library and writer - running on Android and Desktop. https://github.com/mapsforge/mapsforge.

[Mar] MariaDB Corporation Ab. Getting, Installing, and Upgrading MariaDB. https://mariadb.com/kb/en/library/getting-installing-and-upgrading-mariadb.

[Ope] Fundación OpenStreetMap. Basic installation for Nominatim 3.3.0. https://nominatim.org/release-docs/latest/admin/Installation.

[Oraa] Oracle. OpenJDK, an open-source implementation of the Java Platform, Standard Edition. https://openjdk.java.net.

[Orab] Oracle. VirtualBox is a general-purpose full virtualizer for x86 hardware, targeted at server, desktop and embedded use. https://www.virtualbox.org/wiki/VirtualBox.

[osr] Project OSRM, Modern C++ routing engine for shortest patsh in road networks. http://project-osrm.org.

[PG14] Sebastián Rubén Gómez Palomo and Eduardo Moraleda Gil. *Aproximación a la ingeniería del software*. Editorial Universitaria Ramón Areces, 2014.

[Pro] Proyecto Debian. Debian, el sistema operativo universal. https://www.debian.org/.

[qgi] A Free and Open Source Geographic Information System . https://www.qgis.org/en/site/.

[ROB16] Melanie Reuter-Oppermann and Christiane Bernath. German data sets for comparing ambulance location models. 02 2016.

[sam] Situación de las Bases operativas de SAMUR-Protección civil del Ayuntamiento de Madrid. https://bit.ly/2lxHlA5.

[SR] Wolfram Schneider and Slaven Rezić. OSM extracts for Madrid. https://download.bbbike.org/osm/bbbike/Madrid.

[Thea] The Apache Software Foundation. Apache TomEE, MicroProfile and Jakarta EE on Tomcat. https://tomee.apache.org.

[Theb] The MariaDB Foundation. Supporting continuity and open collaboration in the MariaDB ecosystem. https://mariadb.org.

[Tom] Tommaso Urli. How To Set Up an OSRM Server on Ubuntu 14.04. https://www.digitalocean.com/community/tutorials/how-to-set-up-an-osrm-server-on-ubuntu-14-04.

[UB11] Joseba Barroeta Urquiza and Nuria Boada Bravo. *Los servicios de emergencia y urgencias médicas extrahospitalarias en España*. 2011.

[ZDS11] M.H. Fazel Zarandi, S. Davari, and S.A. Haddad Sisakht. The large scale maximal covering location problem. *Scientia Iranica*, 18(6):1564 – 1570, 2011.

Glosario

API Application Programming Interface. 91

CAD Computer Aided Dispatch. 3, 17

CCUE Centro de Coordinación de Urgencias y Emergencias. 24, 35, 41, 42, 44, 47, 49–51, 69

DAO Data Access Object. 62, 63, 65, 68

DSS Diagrama de Secuencia del Sistema. 54, 59, 68–71

EJB Enterprise Java Bean. 33, 86, 90, 92, 93

ETSI Eurpoean Telecomunications Standards Institute. 36

GIS Geographic Information Systems. 37, 47

HTTP Hypertext Transfer Protocol. 33, 93

JAX-WS Java API for XML Web Services. 86

JMS Java Messaging System. 34, 86, 91

JNDI Java Naming and Directory Interface. 90, 92

JPA Java Persistence API. 86, 91

JVM Java Virtual Machine. 34

MQTT Message Queue Telemetry Transport. 34, 91

ODBMS Object Database Management System. 33

OSRM Open Source Routing Machine. 87

PHP PHP: Hypertext Preprocessor. 33

RDBMS Relational Database Management System. 33

RMI Remote Method Invocation. 34

SGBD Sistema de Gestión de Bases de Datos. 33

SGML Standard Generalized Markup Language. 86

SQL Structured Query Language. 33, 99

TETRA TErrestrial Trunked RAdio. 36, 37

WYSIWYG What You See Is What You Get. 86, 94

Anexo A - Datos del escenario de pruebas: Municipio de Madrid

Introducción

Este anexo contiene los datos más importantes utilizados para la construcción del escenario de pruebas, además de las fuentes consultadas para obtenerlos.

El territorio utilizado corresponde al Municipio de Madrid, con centro en la ubicación $40°25'8''N$, $3°41'31''W$. Tiene una superficie aproximada de 604 km^2 y una densidad de población media de 53,30 habitantes por hectárea.

Fuentes de datos

- Los extractos de los mapas para los servidores de cálculo de rutas y localizaciones se han obtenido utilizando la herramienta web OSM exports for Madrid [SR], la cual permite descargar regiones específicas de los mapas OpenStreetMap [Funb] en múltiples formatos.
- Los límites geográficos municipales se han obtenido en formato *shape* del Centro Nacional de Descargas [cnd]. Posteriormente se han simplificado y convertido al formato GeoJSON utilizando las herramientas QGIS [qgi] y MapShaper [mapa].
- Los datos de densidad de población detallados por distritos y barrios se han obtenido del Banco de datos online dependiente del Ayuntamiento de Madrid [ban].
- Los datos de las bases y las ambulancias se han obtenido de la página web del SAMUR-Protección civil del Ayuntamiento de Madrid [sam].

Generación de las zonas

La generación de las zonas para el desarrollo de las pruebas se ha llevado a cabo con la herramienta auxiliar *config-zones*, incluida en el directorio de código fuente del cd. Esta herramienta permite la creación automática de las zonas, el cálculo de los pesos de cada zona y la obtención de los tiempos de respuesta estimados. Toma como datos de entrada los archivos de límites territoriales y de densidad de población y genera como salida un archivo en formato SQL que puede ser importado sobre una base de datos vacía.

Archivos de datos

La Tabla 7.1 contiene una relación de los archivos de información incluidos en el directorio *datos* del proyecto.

Archivo	Formato	Descripción
osm/Madrid.map	MapsForge	Extracto de mapas OSM de la región de Madrid usado en las aplicaciones de cliente.
osm/Madrid.pbf	PBF	Extracto de mapas OSM de la región de Madrid como fuente de datos para los servidores Nominatim y OSRM Routed.
limites_municipales.json	GeoJSON	Archivo de límites municipales con los datos completos del territorio español.
densidad_pob_madrid.csv	CSV	Archivo de datos de densidad de población por distritos y barrios, utilizado durante la generación de zonas.
sql/zonas_madrid_1kms.sql	SQL	Archivo de órdenes SQL para la importación de las zonas y tiempos de respuesta del municipio de Madrid con cuadrículas de 1km^2.
sql/ambulancias_madrid.sql	SQL	Archivo de órdenes SQL para la importación de las ambulancias y bases del municipio de Madrid.

Tabla 7.1: Archivos de datos para el escenario de pruebas

Anexo B - Manual de despliegue

Introducción

En este anexo se describe el proceso de puesta en marcha del sistema hasta un estado similar al descrito en la arquitectura de la figura 5.1. Este manual contiene los pasos más importantes y se centra en la parte específica del proyecto.

Contiene las secciones: Servidores auxiliares, Servidor de aplicaciones TomEE, Aplicación de servidor y Aplicaciones de cliente.

Servidores auxiliares

Servidor de base de datos

Como servidor del sistema de gestión de bases de datos se recomienda MariaDB Server v10.1.37. El proceso de instalación se puede consultar en los manuales oficiales de la organización [Mar].

La configuración del sistema para su uso en el proyecto consiste en la creación de una tabla y su correspondiente usuario. Esto puede realizarse ejecutando los comandos siguientes sobre el terminal de acceso **mysql**:

1. Creación de la base de datos:

   ```
   CREATE DATABASE nombre_db;
   ```

2. Creación del usuario para el acceso a la base de datos:

   ```
   GRANT USAGE on *.* to usuario_db@localhost identified by 'password_db';
   GRANT ALL PRIVILEGES ON nombre_db.* TO usuario_db@localhost;
   ```

Tras la configuración ya es posible realizar la importación de los datos iniciales. Estos datos están contenidos en dos archivos *.sql* dentro del directorio *datos/sql* del cd.

```
# mysql -u usuario_db -p nombre_db < zonas_madrid_1kms.sql
# mysql -u usuario_db -p nombre_db < ambulancias_madrid.sql
```

Servidor Nominatim

El servidor Nominatim puede instalarse en una máquina independiente, en una máquina virtualizada o mediante un contenedor Docker. Para la realización de este proyecto se ha optado por la instalación en una máquina Debian virtualizada. Las instrucciones para la instalación de la versión 3.3.0 pueden verse en la página web del proyecto Nominatim [Ope].

Para la importación de datos se utiliza el archivo *Madrid.pbf* disponible en el directorio *datos/osm* del cd. Este comando debe ejecutarse desde la ruta de instalación de Nominatim.

```
# ./utils/setup.php --osm-file Madrid.pbf --all 2>&1 | tee setup.log
```

Una vez terminada la importación ya se puede arrancar el servidor web y acceder al servicio de geolocalización a través del puerto HTTP por defecto.

OSRM Routed

El servidor OSRM Routed puede instalarse de una forma similar al caso anterior utilizando una máquina virtualizada o mediante un contenedor Docker. Durante este proyecto se ha optado por utilizar una máquina virtual y para su instalación se han seguido las instrucciones referidas en [Tom].

La configuración consiste en la preparación del archivo de mapas y el cálculo de los tiempos de trayecto para el mapa. Como archivo de datos de entrada se utiliza de nuevo el extracto *Madrid.pbf*. Los comandos para la importación son los siguientes:

```
# osrm-extract Madrid.pbf
# osrm-prepare Madrid.osrm
```

Una vez terminado, ya se puede iniciar el programa, el cual incluye un servidor HTTP propio configurado por defecto en el puerto 5000.

```
# osrm-routed map.osrm
```

Servidor de aplicaciones TomEE

Para la instalación del servidor de aplicaciones Java EE se requiere la versión específica Apache TomEE Plus 7.1.0, disponible en la página web de la organización Apache. El cambio de versión es posible, pero implica la actualización de las librerías utilizadas en el código fuente de todos los módulos y una recompilación completa.

El proceso de instalación requiere únicamente desempaquetar el archivo descargado en un directorio concreto, en adelante *tomee_root*.

La configuración consiste en los siguientes pasos:

1. Especificar la información de los recursos de datos del proyecto. Para ello se edita el archivo *tomee_root/conf/tomee.xml* y se añaden los elementos siguientes:

```
<Resource id="pfg_project" type="DataSource">
  JdbcDriver          com.mysql.cj.jdbc.Driver
  JdbcUrl             jdbc:mysql://host_db:3306/nombre_db
  UserName            usuario_db
  Password            password_db
  JtaManaged          true
  DefaultAutoCommit   false
</Resource>

<Resource id="Default JMS Resource Adapter"
  type="ActiveMQResourceAdapter">
  BrokerXmlConfig =
      broker:(tcp://0.0.0.0:61616,mqtt://0.0.0.0:1883)?useJmx=false
  ServerUrl = tcp://0.0.0.0:61616
</Resource>

<Resource id="gis_config"
  class-name="rgomez881.pfg.server.config.Configuration">
  NominatimUrl        http://host_nominatim/nominatim
  OsrmRoutedUrl       http://host_osrm:5000
</Resource>
```

   Los datos *host_db*, *nombre_db*, *usuario_db*, *password_db*, *host_nominatim* y *host_osrm* deben ser cambiados por los datos reales. De forma similar si los puertos utilizados en estos servicios son distintos a los indicados estos deben ser actualizados.

2. Instalar las librerías de terceros necesarias para el despliegue de la aplicación. El documento del cd *despliegue/tomee/lib/terceros.txt* contiene una relación con los archivos necesarios. Estas librerías se instalan copiándolas al directorio *tomee_root/lib*.

3. Configurar los parámetros de ejecución del servidor. Los archivos *despliegue/tomee/bin/setenv.sh* y *despliegue/tomee/bin/setenv.bat* del cd permiten alterar el proceso de arranque de TomEE para incluir dichos parámetros. Deben ser copiados en el directorio *tomee_root/bin*.

Aplicación de servidor

El despliegue de la aplicación de servidor consiste en los siguientes pasos:

1. Compilar e instalar la librería *server-config-1.0.jar*, la cual permite la obtención de la configuración de recursos especificada en TomEE desde la aplicación de servidor. Esta librería se compila con el comando *fuente/server-config/build.sh* y el resultado, disponible en el subdirectorio *target*, debe ser copiado a la carpeta de librerías *tomee_root/lib*.

   La compilación de esta librería en un sistema Debian 9.9 requiere la instalación de los paquetes: *openjdk-8-jdk* y *maven*.

2. Compilar el módulo *server*.La compilación se realiza ejecutando el archivo *fuente/server/build.sh* disponible en el cd. El paquete resultante estará en el subdirectorio *target* con el nombre *server-1.0.jar*.

   La compilación de este módulo en un sistema Debian 9.9 requiere la instalación de los paquetes: *openjdk-8-jdk* y *maven*.

3. Desplegar la aplicación de servidor copiando el archivo *server-1.0.jar* al directorio *tomee_root/webapps* con el nombre *server.jar*.

Tras estos pasos el servidor estará listo para su ejecución.

Aplicaciones de cliente

La compilación de las aplicaciones de cliente puede realizarse tras el proceso de compilación del servidor. Los pasos son los siguientes:

1. Compilar las librerías de cliente propias *client-lib-javafx-ee* y *client-lib-javafx-gui* ejecutando los guiones de comandos *build.sh* proporcionados.

2. Compilar el módulo correspondiente a cada aplicación. Esto se realiza de manera similar a la compilación del módulo *server*, mediante el archivo *build.sh* del raíz del módulo. El resultado será un paquete *.jar* en el directorio *target* o *client-ambulance-App/build*, dependiendo del tipo de aplicación.

   La compilación de los paquetes de cliente requiere, en un sistema Debian 9.9, la instalación de los paquetes: *openjdk-8-jdk*, *openjfx* y *maven*.

3. Configurar los datos de acceso al servidor. Para ello se ha incluido un archivo *client.properties* en los directorios *despliegue/client-ambulance*, *despliegue/client-ccue* y *despliegue/client-notices* que contiene los parámetros necesarios para la ejecución de las aplicaciones. Este archivo debe copiarse al directorio desde donde se ejecute cada aplicación y define las siguientes propiedades:

   ```
   server.remote_client_path=localhost:8080
   server.message_broker_path=localhost:61616
   map.map_file_path=resources/maps/Madrid.map
   map.initial_latitude=40.4165000
   map.initial_longitude=-3.7025600
   ```

   Los dos parámetros *server.remote_client_path* y *server.message_broker_path* deben apuntar al servidor de aplicaciones TomEE instalado. El parámetro *map.map_file_path* debe apuntar al archivo *datos/osm/Madrid.map*. Las últimas dos propiedades deberán contener las coordenadas geográficas del centro del territorio.

Anexo C - Manuales de usuario

Introducción

En este anexo se incluyen los manuales de las aplicaciones de usuario desarrolladas. Contiene las Secciones Aplicación de gestión de avisos, Aplicación de gestión del CCUE y Aplicación de control de la ambulancia.

Aplicación de gestión de avisos

Vista principal

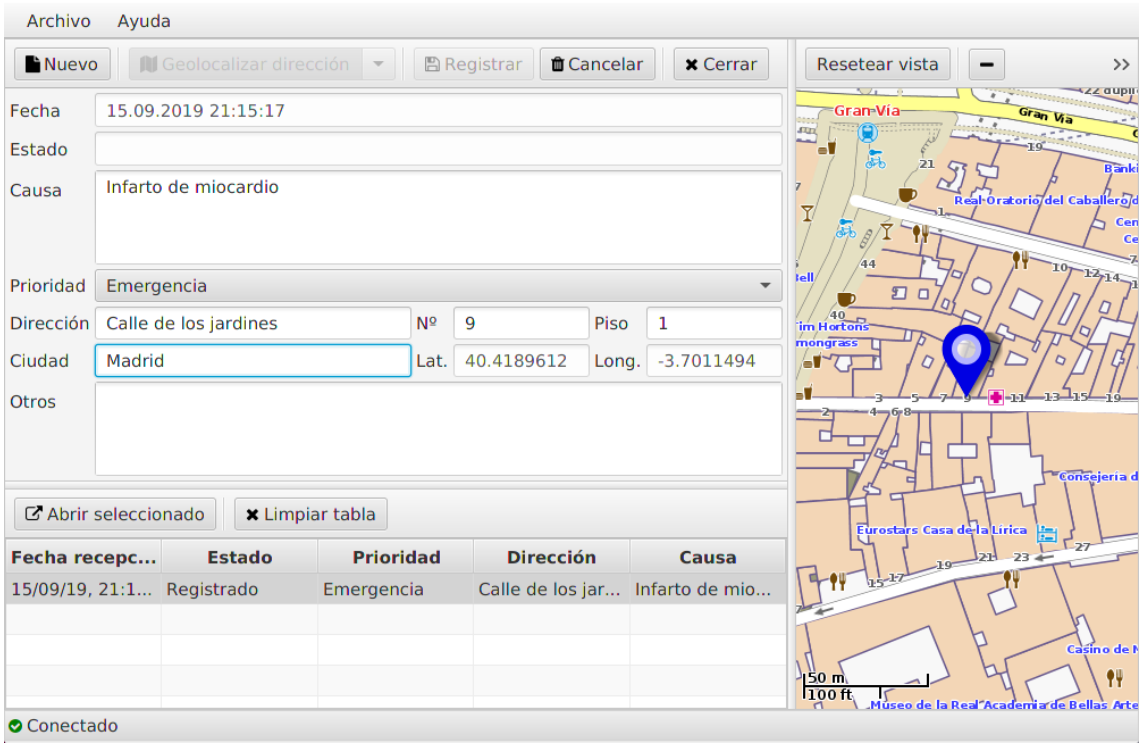


Figura 7.1: Manual de la aplicación de gestión de avisos: vista principal

Control	Acción
Nuevo	Inicia la creación de un nuevo aviso.
Geolocalizar dirección	Obtiene un listado de localizaciones geoposicionadas.
Registrar	Registra el aviso abierto.
Cancelar	Cancela el aviso abierto.
Cerrar	Finaliza la edición del aviso.
Tabla de avisos	Muestra un listado con los avisos recientes.
Abrir seleccionado	Abre el aviso seleccionado de la tabla el el formulario.
Limpiar tabla	Elimina los avisos antiguos de la tabla.
Reiniciar vista	Reestablece los niveles de zoom y centrado en el mapa.

Registrar un aviso nuevo

1. Pulsar el botón **Nuevo**. El formulario se vuelve editable. El campo de fecha es completado automáticamente con la fecha de inicio de la edición.
2. Rellenar la causa médica.
3. Seleccionar un valor de prioridad en el desplegable.
4. Establecer la dirección y la ciudad.
5. [Opcional] Establecer un número de calle y/o un número de piso.
5. Geolocalizar la dirección pulsando en la parte izquierda del botón **Geolocalizar dirección**. Este proceso lanza una petición al servidor para la resolución de la dirección y puede tardar unos segundos. Cuando termine aparecerá bajo el botón una lista con las posibles direcciones encontradas.
6. Seleccionar una de las opciones de la lista de direcciones desplegable. La selección unicamente rellena las coordenadas geográficas del formulario, no modifica los campos de texto. La flecha del botón **Geolocalizar dirección** permite acceder a los últimos resultados sin tener que realizar de nuevo la petición.
7. [Opcional] Seleccionar las coordenadas directamente sobre el mapa.Para ello únicamente hace falta hacer doble click sobre la posición elegida.
8. Pulsar el botón **Registrar**.

   Al finalizar la comunicación con el servidor el aviso aparecerá en la tabla inferior con estado *Registrado*.

Cancelar un aviso registrado

* La cancelación de un aviso abierto solo es posible en caso que este haya sido registrado previamente. En caso contrario el botón **Cancelar** estará deshabilitado.

1. Pulsar el botón **Cancelar**.
2. Aceptar el diálogo de confirmación mostrado.

   Al finalizar la comunicación con el servidor el aviso aparecerá en la tabla con estado *Cancelado*.

Aplicación de gestión del CCUE

Vista de gestión de asignaciones

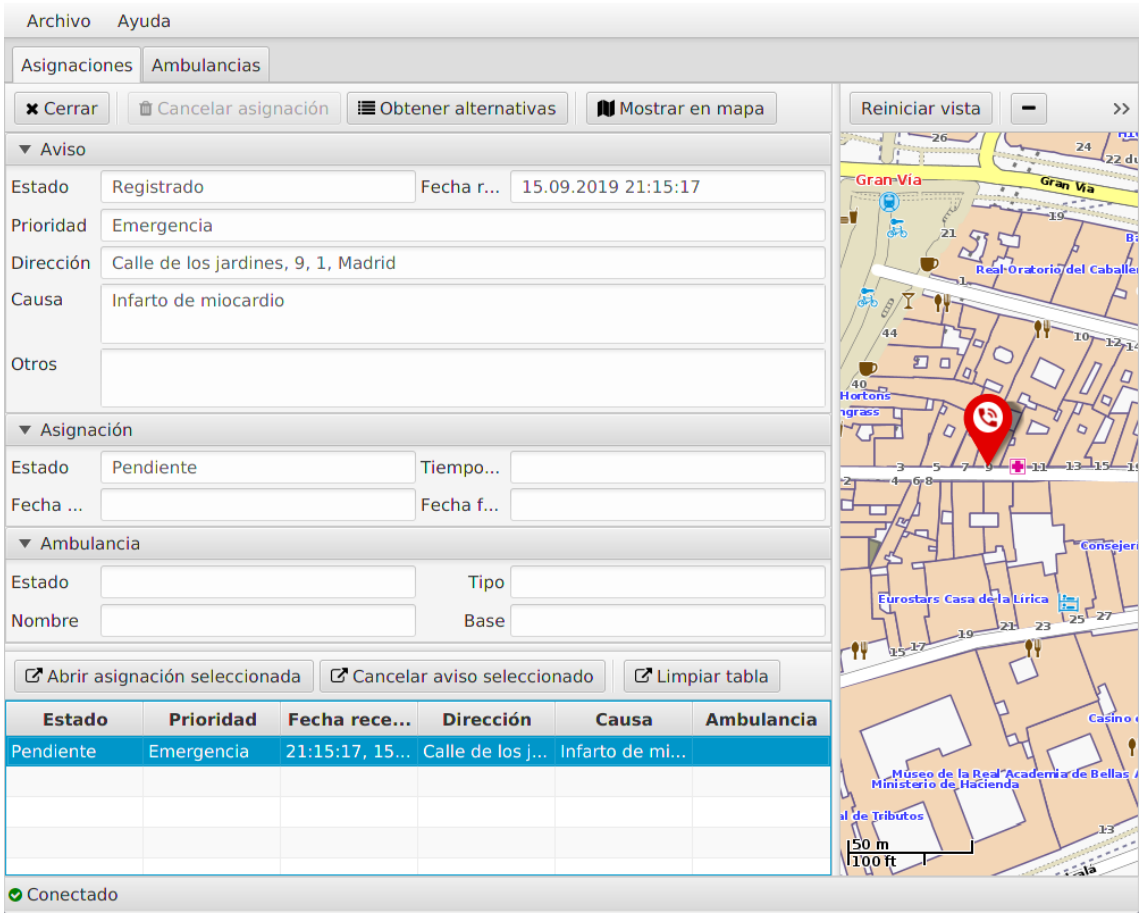


Figura 7.2: Manual de la aplicación de gestión del CCUE: vista de asignaciones

Control	Acción
Cerrar	Finaliza la visualización de la asignación abierta.
Cancelar asignación	Cancela la asignación abierta.
Obtener alternativas	Inicia el cálculo de las alternativas de asignación.
Tabla de asignaciones	Muestra un listado ordenado con las asignaciones recientes.
Abrir seleccionada	Abre la asignación seleccionada en la tabla.
Cancelar aviso seleccionado	Cancela el aviso asociado a la asignación seleccionada en la tabla.
Limpiar tabla	Elimina las asignaciones canceladas o finalizadas.
Reiniciar vista	Reestablece los niveles de zoom y centrado en el mapa.

Vista de gestión de alternativas de asignación

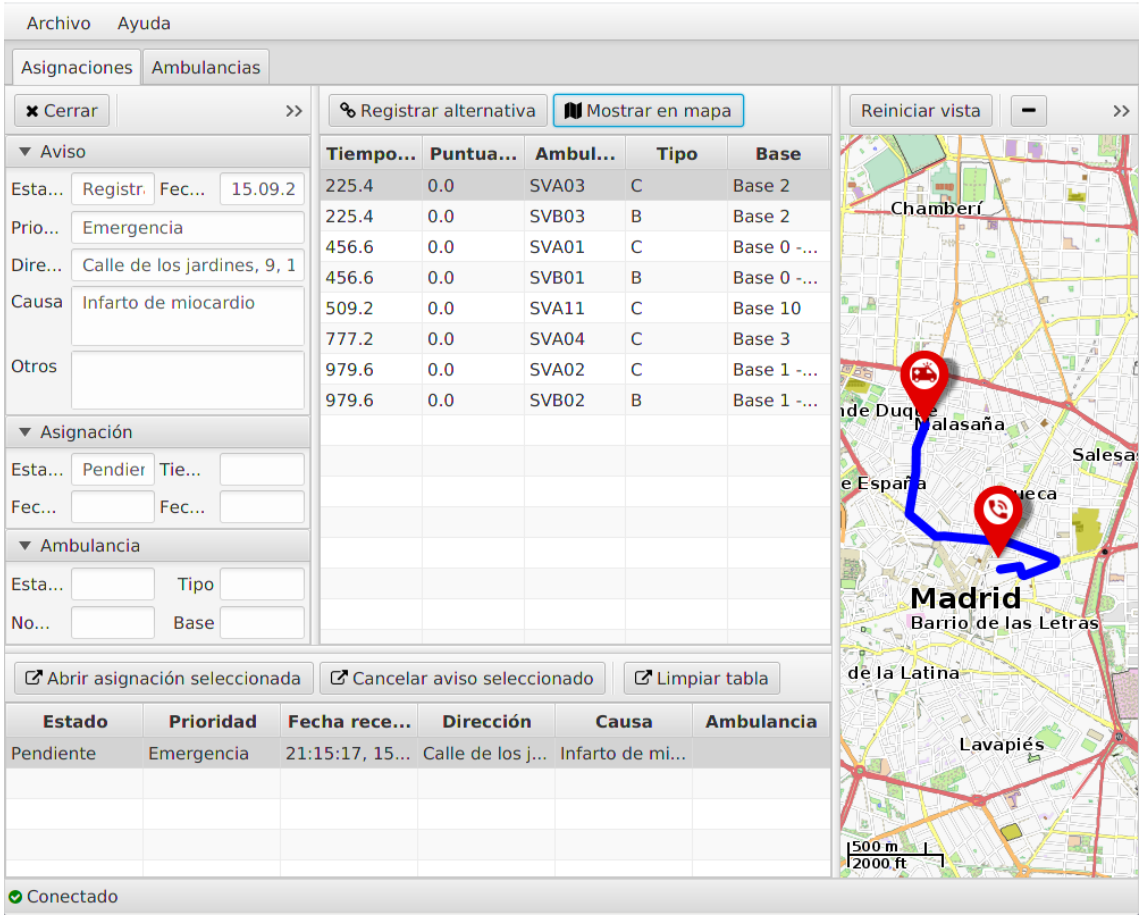


Figura 7.3: Manual de la aplicación de gestión del CCUE: vista de alternativas

Control	Acción
Registrar alternativa	Efectúa el registro de la alternativa seleccionada.
Mostrar en mapa	Muestra en el mapa la ruta y posiciones geográficas asociadas a la alternativa.
Tabla de alternativas	Muestra la información de cada alternativa calculada: el tiempo de respuesta estimado, la valoración de capacidad de actuación y los datos de la ambulancia.

Operación de registro de una asignación pendiente

1. Abrir una asignación pendiente seleccionándola en la tabla.
2. Pulsar el botón **Obtener Alternativas** para calcular las posibles alternativas de asignación. El cálculo debería ser inmediato para avisos con prioridad de Emergencia, pero puede llevar unos segundos si la prioridad es menor.
3. Seleccionar una alternativa en la lista desplegada.
4. [Opcional] Mostrar la ruta y posiciones del aviso y la ambulancia en el mapa. Esta operación puede repetirse para cada elemento de la lista.
5. Pulsar registrar alternativa. Esta operación asocia registra la asignación con la ambulancia asociada y habilita el botón de cancelar. A partir de este momento el panel de información de la asignación se autoactualiza cada vez que se recibe un cambio de estado.
6. Cerrar asignación. Esta acción unicamente finaliza la monitorización de la asignación. Esta puede volver a abrirse mientras siga activa.

Operación de cancelación de una asignación registrada

* La cancelación de una asignación solo es posible en caso que esta haya sido registrada y no esté finalizada. En caso contrario el botón **Cancelar** estará deshabilitado. Tiene como consecuencia el reinicio del proceso de asignación desde desde el estado pendiente.

1. Abrir una asignación registrada o activa.
2. Pulsar el botón **Cancelar asignación** para efectuar la cancelación.
3. Cerrar la asignación cancelada.

Operación de cancelación del aviso asociado a la asignación

* La cancelación del aviso tiene como consecuencia la finalización de la incidencia completa, cancelando además la asignación asociada.

1 Seleccionar la asignación en la tabla y pulsar **Cancelar aviso seleccionado**.

2 Aceptar la solicitud de confirmación.

Vista de gestión de ambulancias

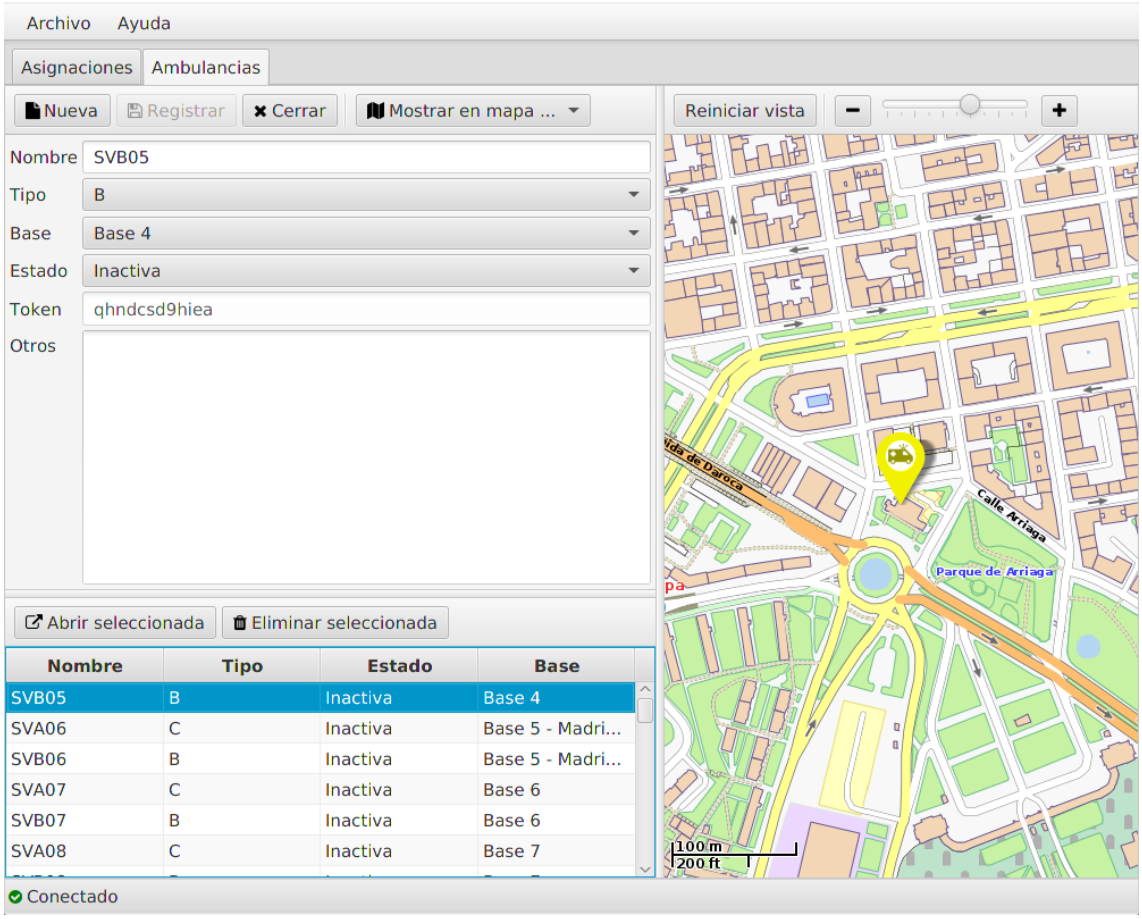


Figura 7.4: Manual de la aplicación de gestión del CCUE: vista de ambulancias

Control	Acción
Nueva	Inicia la creación de una nueva ambulancia.
Registrar	Registra los datos de la ambulancia abierta.
Cerrar	Finaliza la edición de la ambulancia abierta.
Mostrar en mapa	Muestra la posición de la ambulancia abierta o su base en el mapa.
Reiniciar vista	Reestablece los niveles de zoom y centrado en el mapa
Abrir seleccionada	Abre la ambulancia seleccionada en la tabla.
Eliminar seleccionada	Elimina la ambulancia seleccionada en la tabla.

Operación de creación de una nueva ambulancia

1. Pulsar el botón **Nuevo**. El formulario se vuelve editable y se activa el botón **Registrar**.
2. Rellenar los datos de la ambulancia.
3. [Opcional] Seleccionar la base asignada a la ambulancia.
4. Pulsar el botón **Registrar**. Esta acción tiene como consecuencia la actualización de los datos de estado y *Token*.
5. Comunicar el *Token* de autenticación al operador de la ambulancia.
6. Cerrar la edición.

Operación de eliminación de una ambulancia registrada

*. Esta operación únicamente puede realizarse si la ambulancia está en estado *Inactivo*.

1. Seleccionar la ambulancia en la tabla.
2. Pulsar **Eliminar seleccionada** y confirmar la operación.

Operación de modificación de una ambulancia registrada

1 Seleccionar la ambulancia en la tabla y pulsar **Abrir seleccionada**.

2 Editar los campos necesarios.

3 Pulsar **Registrar** para almacenar los cambios.

Aplicación de control de la ambulancia

Vista de ambulancia

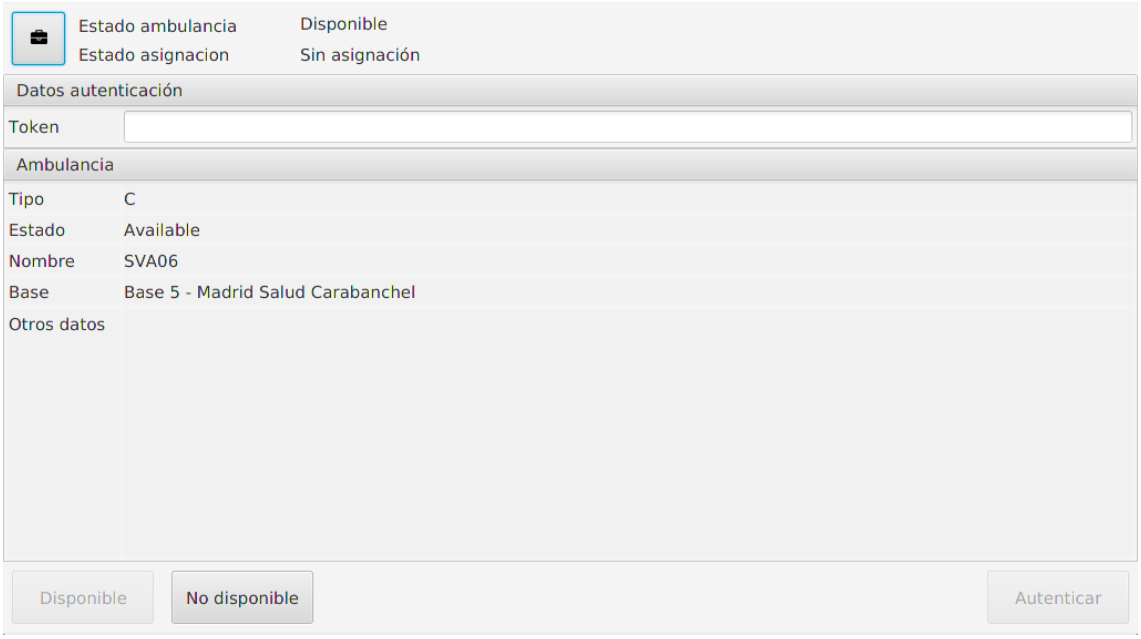


Figura 7.5: Manual de la aplicación de control de la ambulancia: vista de ambulancia

Control	Acción
Icono portafolio	Cambia a la vista de asignación.
Disponible	Establece el estado de la ambulancia a *Disponible*.
No disponible	Establece el estado de la ambulancia a *Inactiva*.
Autenticar	Autentica la aplicación de la ambulancia en el sistema.

Vista de asignación

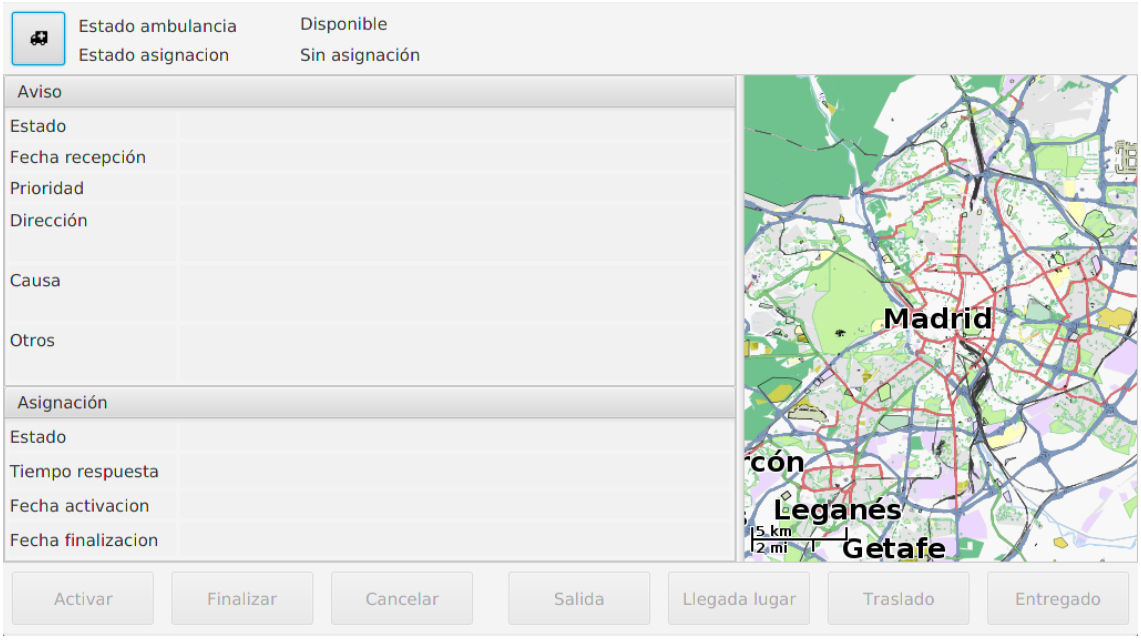


Figura 7.6: Manual de la aplicación de control de la ambulancia: vista de asignación

Control	Acción
Icono ambulancia	Cambia a la vista de ambulancia.
Activar	Establece el estado de la asignación asociada a *Activada* y el estado de la ambulancia a *Activa*.
Finalizar	Establece el estado de la asignación asociada a *Finalizada* y el estado de la ambulancia a *No disponible*.
Cancelar	Establece el estado de la asignación asociada a *Cancelada* y el estado de la ambulancia a *No disponible*.
Salida	Estable el estado de la ambulancia a *Salida*.
Llegada	Estable el estado de la ambulancia a *Llegada al lugar*.
Traslado	Estable el estado de la ambulancia a *Traslado del paciente*.
Entregado	Estable el estado de la ambulancia a *Entregado*.

Autenticación de la aplicación en el sistema

* Esta operación tiene como objetivo la asociación de la aplicación y la ambulancia. Solo es necesario realizarla la primera vez, quedando los datos almacenados en el dispositivo.

1. Introducir el *Token* recibido en el campo de edición de la vista de ambulancia.
2. Pulsar el botón **Autenticar**. Si la autenticación es correcta, los datos de la ambulancia registrada aparecen en el formulario y se habilitan los botones de cambio de estado.

Ejecución de una asignación

* Esta operación está disponible en el momento que se recibe una notificación de asignación registrada.

1. Pulsar el botón **Activar** para comunicar que se ha efectuado la activación.
2. Pulsar el botón **Salida** en el momento en que se produce la puesta en marcha.
3. [Opcional] Pulsar el botón **Cancelar** si la ambulancia pierde la capacidad de ejecutar la asignación. Esta acción termina la ejecución y establece el estado de la ambulancia a *No disponible*.
4. Pulsar el botón **Llegada al lugar** en el momento en que se llega al lugar del aviso.
5. [Opcional] Pulsar el botón **Traslado** si se va a efectuar el traslado del paciente al centro hospitalario.
6. [Opcional] Pulsar el botón **Entregado** en el momento en el que el paciente es atendido por el personal del centro hospitalario.
7. Pulsar el botón **Finalizar** para comunicar que se ha terminado la operación. La finalización provoca que la ambulancia pase a estado *No disponible* automáticamente.
8. Cambiar a la vista de ambulancia y pulsar el botón **Disponible** en el momento en el que el estado interno esté reestablecido y puedan atenderse nuevos avisos.