uoc.edu

J-Tracking

Plataforma de control de flotas Open Source

Rafael Fernández Lorenzo

Grado en Ingeniería Informática

Java EE

Vicenç Font Sagrista

Santi Caballe Llobet

10/01/2018

FICHA DEL TRABAJO FINAL

Título del trabajo:	*J-Tracking. Plataforma de control de flotas*
Nombre del autor:	*Rafael Fernández Lorenzo*
Nombre del consultor/a:	*Vicenç Font Sagrista*
Nombre del PRA:	*Santi Caballe Llobet*
Fecha de entrega:	*01/2018*
Titulación::	*Grado en Ingeniería Informática*
Área del Trabajo Final:	*Java EE*
Idioma del trabajo:	*Español/Inglés*
Palabras clave	*JEE, fleet management, car tracking, GPS*

Resumen del Trabajo

El producto J-tracking, es una solución Open Source para el control de flotas que aspira a proporcionar un sistema fiable, sencillo, escalable y asequible a aquellas organizaciones con necesidad de controlar sus flotas de vehículos.

La utilización de las tecnologías basadas en JEE ha permitido implementar una solución software que permite, mediante la definición de un interfaz común, integrar dispositivos hardware de control de flotas heterogéneos, permitiendo a los usuarios disponer de información centralizada referente al posicionamiento de sus vehículos, con independencia de las capacidades de los dispositivos empleados.

Desde un punto de vista didáctico, el desarrollo del Trabajo de Fin de Grado ha permitido llevar a la práctica los conocimientos adquiridos a lo largo de itinerario docente del Grado en Ingeniería Informática.

Ha permitido profundizar significativamente en las asignaturas relacionadas con el itinerario especifico entorno a la Ingeniería del Software, desde la Gestión de Requisitos, el desarrollo de Bases de Datos o el Diseño de Componentes de Software Distribuido.

Desde el punto del desarrollo de la solución, el proyecto ha permitido profundizar en el conocimiento de las tecnologías JEE. El desarrollo de una aplicación web multicapa ha supuesto la utilización de framework de presentación Primefaces, Inyección de

Dependencias CDI, Enterprise Java Beans, framework de persistencia Java Persistence API, JPA, así como la utilización de los servicios transversales proporcionados por los contenedores del servidor Wildfly, seguridad, autorización, autenticación y transaccionalidad.

Abstract

The J-tracking software is an Open Source solution that aims to provide a reliable, simple, scalable and inexpensive system for fleet management.

The use of JEE-based technologies has allowed the implementation of a software solution that allows the use of heterogeneous tracking devices. Through a common interface, hardware devices equipped with different technologies can be easily integrated into the system. This approach allows users to have centralized information regarding the positioning of their vehicles, regardless of the capabilities of the devices used.

From a didactic point of view, the development has allowed to put into practice the knowledge acquired throughout the teaching itinerary of the Degree in Computer Engineering.

It has allowed to deepen significantly in the subjects related to the specific itinerary of the Software Engineering, from the Management of Requirements, the development of Databases or the Design of Components of Distributed Software.

From the point of view of the development, the project has allowed to deepen the knowledge of JEE technologies. The development of a multilayer web application has taken advantage of the Primefaces presentation framework, CDI Dependency Injection, Enterprise Java Beans, Java Persistence API persistence framework, JPA, as well as the use of the transversal services provided by the Wildfly server containers, security , authorization, authentication and transactionality.

Índice de contenido

1 Introducción

1.1 Contexto y justificación del trabajo

En la actualidad, Organizaciones no Gubernamentales (ONG) de diferentes países, se despliegan a lo largo de las zonas de conflicto y catástrofe en misiones de ayuda y cooperación. Los desplazamientos dentro de sus áreas de actuación originan unas necesidades de información acerca de la ubicación de sus medios vehiculares y humanos.

Las organizaciones tradicionales: Cuerpos de Protección civil, Servicios de Urgencias cuentan con dispositivos de comunicación avanzados que incluyen sistemas de control de flotas. Estos sistemas permiten representar los vehículos sobre sistemas de cartografía[1]. Podríamos citar como ejemplo las ambulancias del Servicio de Urgencias del Ayuntamiento de Madrid, equipadas con un sistema de comunicaciones, que permite conocer la ubicación de las diferentes unidades de soporte vital básico o avanzado.

Los sistemas utilizados son mayoritariamente propietarios, están desarrollados para cubrir unas necesidades concretas y tienen unos costes muy elevados.

Se dispone también, en el mercado, de empresas dedicadas a proporcionar servicios de control de flotas, cobrando diferentes tarifas según el servicio. Estos proveedores de servicio instalan sus dispositivos y proporcionan al cliente un aplicativo para el acceso a la información de sus vehículos. La utilización de estas empresas supondría tener que identificar a un proveedor que sea adecuado para la zona de utilización[2], con los costes que puede llegar a suponer decisiones de contratación de servicios

1 Los vehículos utilizan un sistema de comunicaciones TETRA para el envío de mensajes de estado, confirmación de avisos y posicionamiento de sus unidades.

2 Pensemos en organizaciones que se desplazan a zonas carentes de infraestructuras de comunicaciones básicas como GSM, o más avanzadas como UMTS o LTE. Lo que obligaría a utilizar productos que trabajen vía satélite sobre redes de cobertura global como INMARSAT o IRIDIUM SBD.

erróneas.

El proyecto J-Tracking aspira a ser una solución de control de flotas, Open Source, que permita proporcionar un servicio flexible y de bajo coste a las organizaciones interesadas.

La plataforma J-Tracking es un producto de software que permite la representación de las posiciones de los vehículos de una flota, su seguimiento y localización sobre un sistema de cartografía público, está orientado a organizaciones de tamaño pequeño y medio.

El sistema esta dotado de un software de control y explotación que permite, junto con el posicionamiento, el control de los dispositivos embarcados y su configuración remota, así como el almacenamiento de la información enviada para su posterior explotación.

El proyecto está concebido como un sistema escalable que partiendo de una funcionalidades básicas centradas en la representación de posiciones, permita alcanzar un conjunto de funcionalidades avanzadas como la configuración y control remoto de dispositivos de control de flotas. Proporciona una interfaz única que deberá ser implementadas por los diferentes dispositivos que se desarrollen o integren en el sistema.

1.2 Objetivos del trabajo

1.2.1 Implementación

El objetivo principal a alcanzar ha sido el desarrollo de una aplicación que satisfaga los requisitos de alto nivel. Dado que la variedad de dispositivos de control del flotas es muy alta y sus funcionalidades y capacidades muy diversas, se ha considerado que la mejor solución es la definición de un interfaz único de entrada y salida desde el software de control hacia los dispositivos y viceversa. Esta interfaz deberá ser

implementada por los dispositivos que deseen integrarse en el sistema. Al mismo tiempo el sistema deberá proporcionar mecanismos para el control de dispositivos, representación de sus posiciones, configuración y control remoto así como explotación de la información proporcionada sobre un sistema de cartografía público.

1.2.2 Objetivos didácticos

Desde el punto de vista didáctico el desarrollo de este proyecto deberá permitir consolidar los conocimientos adquiridos en asignaturas de corte transversal como la Gestión o de Proyectos o la Gestión de Requisitos, profundizar en disciplinas relacionadas directamente con el itinerario elegido como Bases de Datos o Ingeniería del software de Componentes y Sistemas Distribuidos e introducir nuevos contenidos como el uso de frameworks de presentación como Primefaces, CDI o Java Persistence API.

1.3 Enfoque y método seguido

El proyecto J-Tracking es un Trabajo de Fin de Grado, y como tal, se encuentra sujeto a la disciplina de entregas marcadas por el Plan Docente de la Asignatura. Si bien la metodología es básicamente un desarrollo en cascada, se han realizado algunas modificaciones que han permitido minimizar los riesgos para el proyecto.

Se ha utilizado una metodología que sigue las prácticas recomendadas por el Proceso Unificado de Rational para equipos de desarrollo.[3]

- Proceso iterativo e incremental
- Gestión de requisitos
- Modelo arquitectónico basado en componentes
- Utilización exhaustiva de UML

3 Rational Unified Process Best Practices for Software Development Teams, Páginas 1 y 2

- Verificación de la calidad.
- Control del cambio

Para la documentación de la arquitectura se ha utilizado el modelo de *4+1 vistas*. En este punto cabe destacar que se ha descartado la utilización de *RM-ODP* por considerar que utiliza un proceso de documentación mucho más denso y complejo. *4+1 vistas de Kruchten* proporciona igualmente un modelo basado en vistas, no tan elaborado ni denso pero que cubre suficientemente la necesidades del proyecto.

1.4 Planificación del trabajo

Se ha elaborado una planificación siguiendo la propuesta del calendario reflejada en el Plan Docente.

Se han previsto entregas parciales a la finalización de la etapa de requisitos por considerar que es un hito clave en la toma de decisiones respecto al proyecto. Igualmente para dar seguimiento a la fase de construcción se ha previsto otra entrega parcial durante esta fase.

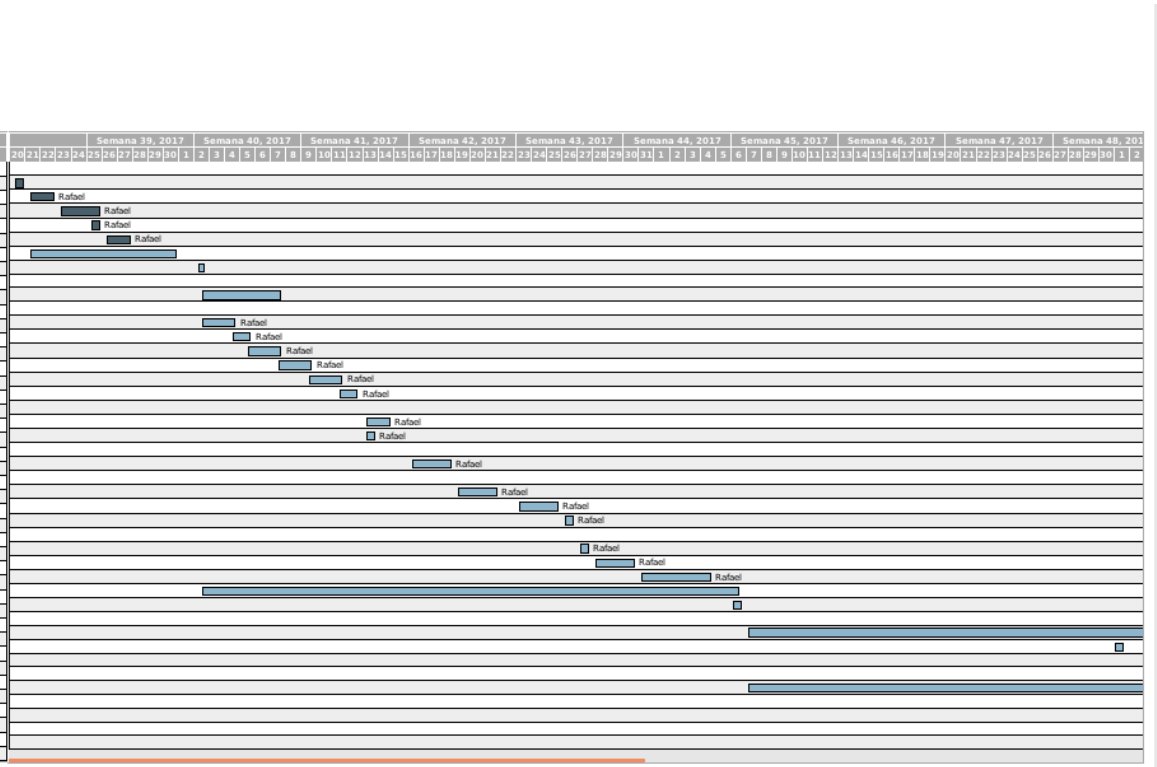
Siguiendo la metodología RUP, donde se recomienda realizar a la mayor brevedad las tareas susceptibles de introducir riesgos en el proyecto, se ha realizado el despliegue de la base de datos en el servidor, tarea 2.5.3, una vez finalizado el mapeo ORM. Se estima que un error en la definición del modelo de datos de la aplicación tendría un impacto muy alto, susceptible de provocar el fracaso de proyecto o costes de reconstrucción no asumibles en el tiempo disponible.

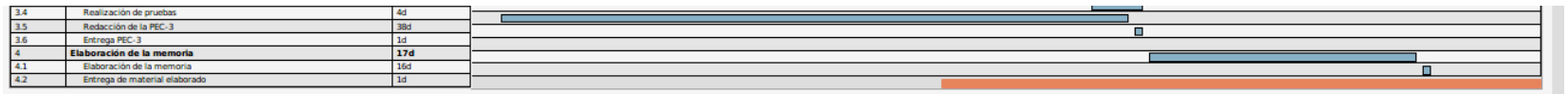
Trabajo fin de Grado

Empresa: UOC
Encargado: Rafael Fernández Lorenzo
Inicio: 20 septiembre, 2017
Fin: 10 enero, 2018
Fase: Planificación

Gráfica Gantt

WBS	Nombre	Trabajo
1	**Planificación temporal proyecto**	**17d 6h**
1.1	Plan docente y orientaciones iniciales	1d
1.2	Captura de requisitos de alto nivel	2d
1.3	Especificación de requisitos de alto niel	2d
1.4	Elección herramientas de desarrollo	1d
1.5	Elaboración de la planificación temporal	2d
1.6	Redacción de la PEC-1	9d
1.7	Entrega PEC-1	6h
2	**Requisitos, análisis y diseño**	**66d**
2.1	Instalación entorno de desarrollo	5d
2.2	**Especificación de requisitos**	**9d**
2.2.1	Captura de requisitos candidatos	2d
2.2.2	Priorizacion de requisitos candidatos	1d
2.2.3	Especificación de casos de uso	2d
2.2.4	Trazabilidad y modelos	1d
2.2.5	Descripción casos de uso	2d
2.2.6	Requisitos no funcionales	1d
2.3	**Prototipado del GUI**	**3d**
2.3.1	Elaboración de patallas	2d
2.3.2	Entrega parcial	1d
2.4	**Modelado del dominio**	**3d**
2.4.1	Diagrama de clases	3d
2.5	**Diseño de la BB.DD**	**7d**
2.5.1	Mapeo ORM	3d
2.5.2	Descripción de tablas	3d
2.5.3	Despliegue en el SGDB	1d
2.6	**Arquitectura de la aplicación**	**8d**
2.6.1	Arquitectura de alto nivel	1d
2.6.2	Arquitectura Java EE de alto nivel	2d
2.6.3	Arquitectua Java EE detallada	5d
2.7	Redacción de la PEC-2	30d
2.8	Entrega PEC-2	1d
3	**Construcción y pruebas**	**88d**
3.1	Codificación de la aplicación	39d
3.2	Entrega parcial	1d
3.3	Elaboración del plan de pruebas	5d
3.4	Realización de pruebas	4d
3.5	Redacción de la PEC-3	38d
3.6	Entrega PEC-3	1d
4	**Elaboración de la memoria**	**17d**
4.1	Elaboración de la memoria	16d
4.2	Entrega de material elaborado	1d





1.5 Sumario de productos obtenidos

- Plan de trabajo: descripción de objetivos, requisitos de alto nivel, metodología, planificación y estimación de costes
- Especificación de requisitos, análisis y diseño: especificación de requisitos y arquitectura del sistema.
- Producto J-tracking: J-tracking.ear, fichero desplegable en el servidor de aplicaciones Wildfly 10.1
- Cliente simulador: Jtrackingclient.jar cliente para simular dispositivos de control de flotas, ejecutable desde consola.
- Plan de pruebas: plan de pruebas y resultados de su ejecución.
- Manual de usuario J-tracking: manual de usuario de la aplicación.
- Manual de usuario J-trackingClient: manual de usuario del simulador.
- Proyectos Eclipse: J-tracking.zip
- Manual de despliegue: fichero Instrucciones.pdf con información acerca de la compilación y despliegue del aplicativo.

1.6 Descripción de otros capítulos de la memoria

Se detalla, mediante un modelo de vistas y puntos de vista, la especificación de requisitos, la arquitectura y el diseño del producto.

La representación del modelo de arquitectura del producto J-Tracking incluirá, junto con el modelado y descripción de los elementos de la arquitectura, un apartado de caracterización de usuarios, dispositivos y contextos de utilización.

La motivación del modelo arquitectónico y las decisiones de diseño e implementación más importantes se han detallado dentro de los apartados de análisis de las diferentes

vistas.

- Vista de escenarios: enumera y detalla los casos de uso y actores identificados dentro del sistema. Se ha realizado una agrupación lógica y funcional de los casos de uso dentro de los componentes de alto nivel del sistema.

  En el análisis de requisitos se han considerado, los perfiles de usuarios, sus contextos de utilización así como las posibles características de los dispositivos del sistema.

- Vista lógica: muestra mediante diagramas invariantes el modelo del dominio, identificando las entidades persistentes del sistema y su mapeo objeto-relacional.

- Vista de desarrollo: ilustra, por medio de diagrama de componentes, el proceso de refinado de la arquitectura desde un muy alto nivel a un arquitectura detallada para la tecnología Java EE.

- Vista de procesos: por medio de diagramas de actividades y de secuencia, documenta el comportamiento de los elementos más complejos y significativos del sistema, las decisiones de diseño y su implementación.

- Vista física: describe, por medio de diagramas de despliegue, una posible distribución del sistema sobre componentes hardware.

2 Vista de escenarios

2.1 Usuarios, dispositivos y contextos de utilización

Para satisfacer los requisitos de alto nivel detallados en la especificación se deben considerar un una serie de elementos que determinan la arquitectura del producto.

- Características de los dispositivos de control de flotas.
- Necesidades de los usuarios y características de los mismos.

- Contextos de utilización.
- Necesidades y capacidades de las organizaciones usuarias del sistema.

2.1.1 Características de los dispositivos

Tal y como se muestra en el diagrama de integración J-Tracking, los dispositivos de control de flotas estarán dotados de capacidades de comunicación diversas. Muchos dispondrán de capacidad de comunicación sobre redes IP, por medio de conexiones de banda ancha móviles GPRS, UMTS o LTE. Para estos dispositivos se supondrá que disponen de capacidad de implementación de la interfaz única del producto J-Tracking.

Otros productos, dotados de capacidades inferiores o desplegados en zonas donde la disponibilidad de redes avanzadas de comunicación móvil sea escasa o directamente no exista, podrán implementar la interfaz común mediante pasarelas Ej. SMS/TCP que permitirán el intercambio de información entre el sistema y los dispositivos.

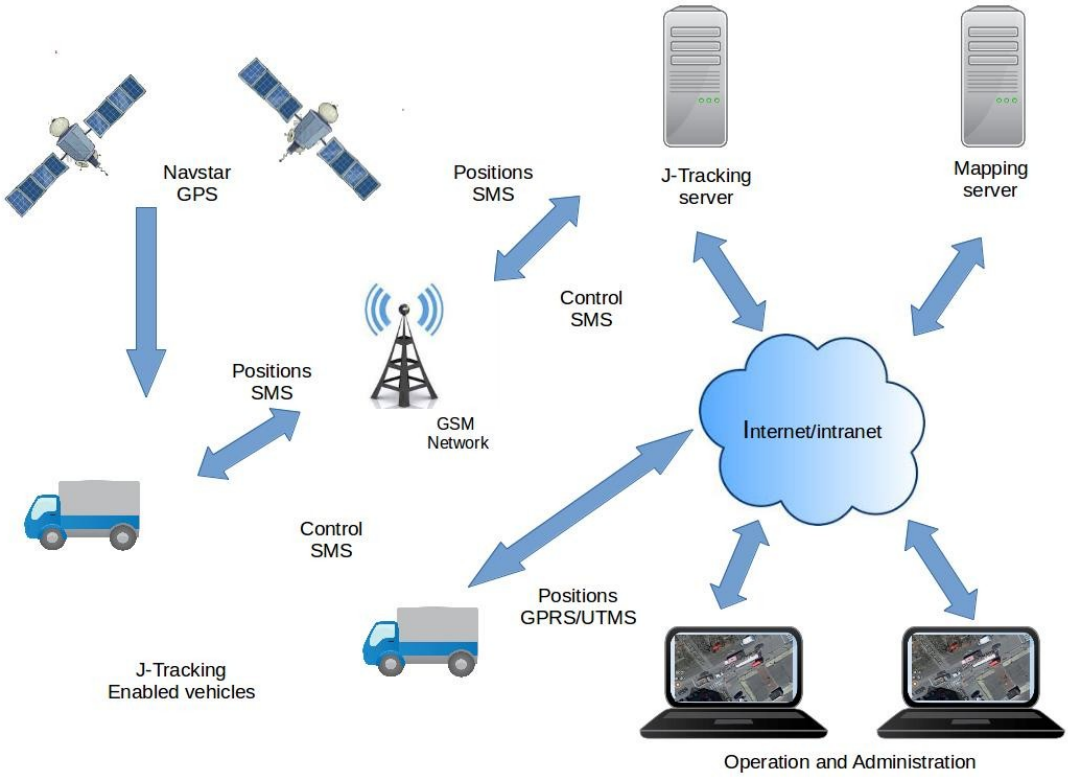


Ilustración 1: Diagrama de integración sistema J-Tracking

2.1.2 Usuarios

El software J-Tracking es un producto orientado a su utilización por personal familiarizado con las Tecnologías de la Información y la Comunicaciones, TIC, pertenecientes a organizaciones de tamaño pequeño y medio, típicamente ONG.

No requerirá de un alto grado de capacitación técnica para el desempeño de ninguno de los roles previstos en el aplicativo.

No será necesaria realizar una fase de formación específica para los usuarios o administradores. El manual suministrado debería ser suficiente para el uso del aplicativo por los diferentes roles.

2.1.3 Contextos de utilización

Se establecen dos contextos de utilización:

- Usuarios no desplazados a zonas de conflicto.

Los usuarios no desplazados dispondrán de capacidades de comunicación completas, estando en disposición de asumir las labores de administración y explotación del sistema cuando los usuarios de contextos más limitados lo requieran.

- Usuarios operando en zonas de conflicto.

Los usuarios desplazados operaran en unos escenarios complejos, muchas veces carentes de infraestructuras modernas de comunicación, como redes móviles de banda ancha, no desplegadas en todo el territorio o temporalmente no disponibles.

Dispondrán de equipos portátiles tipo laptop o notebook, de diferentes marcas o modelos. No se dispondrá de medios técnicos de reparación y configuración de los mismos.

2.2 Captura inicial de requisitos funcionales candidatos

Se ha efectuado una captura inicial de requisitos funcionales siguiendo el juicio experto del desarrollador.

En el Anexo I Especificacion_requisitos_J-Tracking.pdf se detallan los requisitos candidatos, una matriz de trazabilidad entre requisitos candidatos y casos de uso del sistema y la descripción textual de los mismos.

2.3 Casos de uso

Los siguientes diagramas muestran la relación de casos de uso identificados en el sistema. Se han agrupado en unos componentes de alto nivel que actuarán como fachada de todos los casos de uso que contienen.

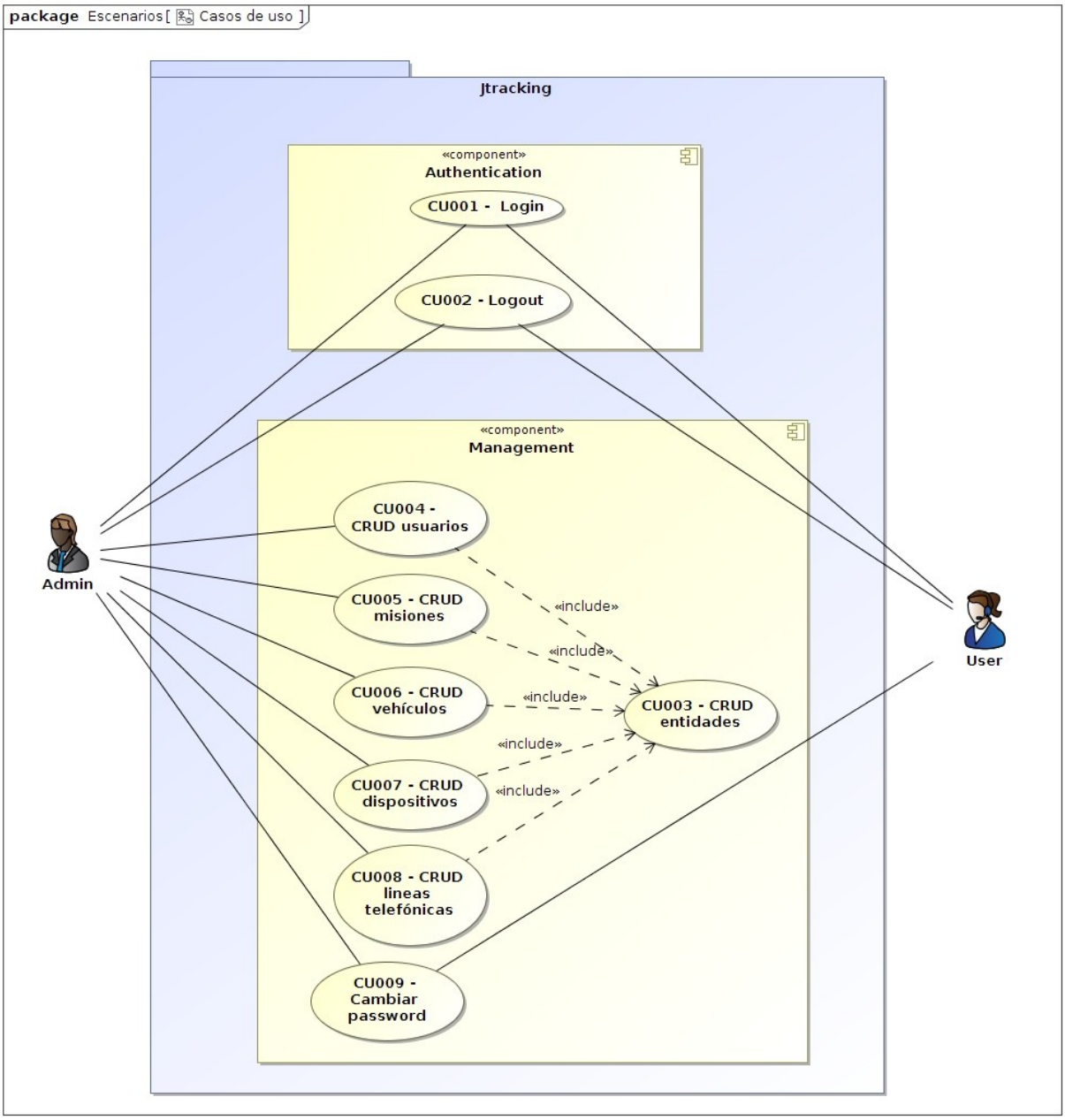


Figura 1: Casos de uso

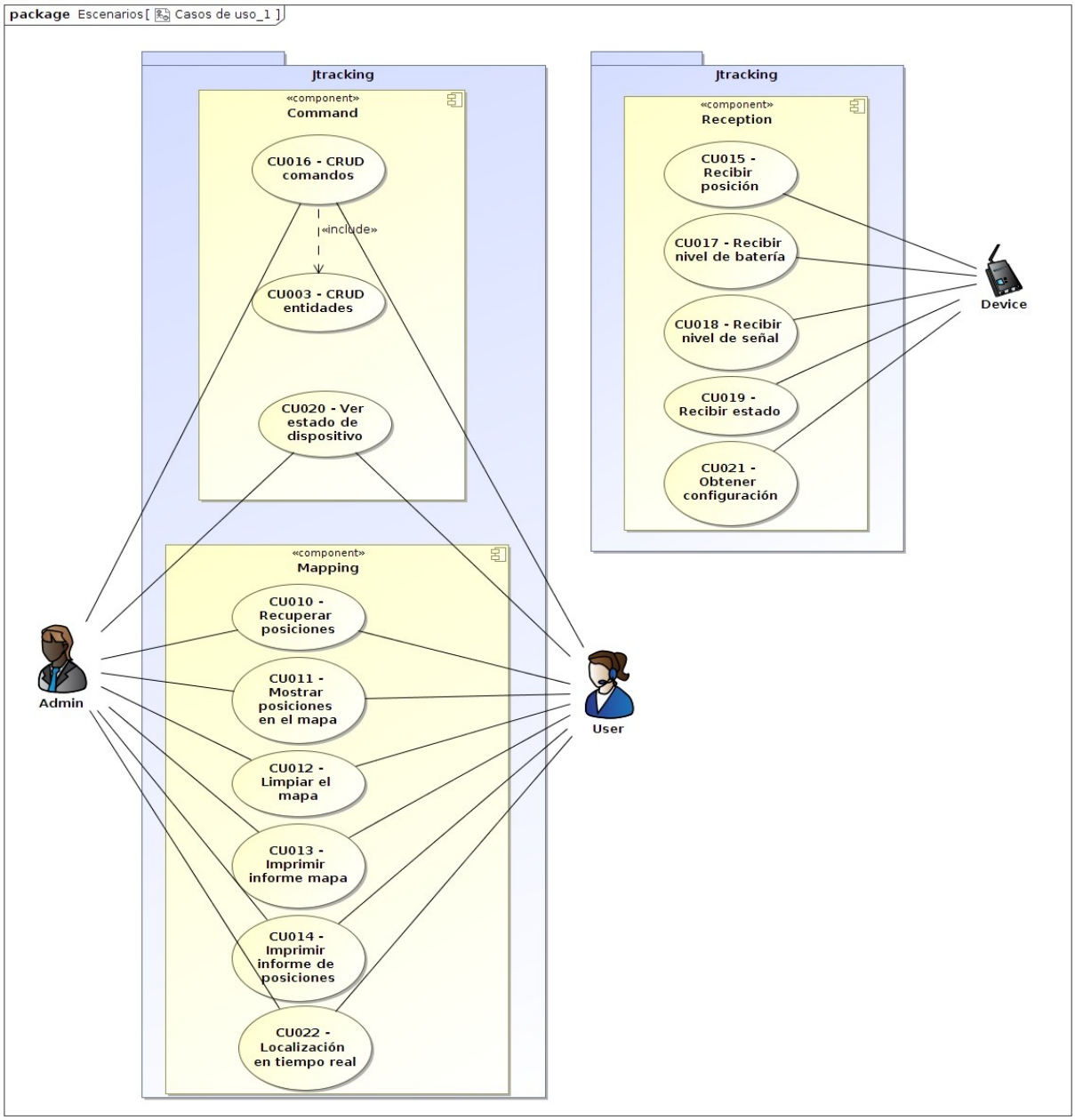


Figura 2: Casos de uso

2.3.1 Descripción textual de los casos de uso del sistema J-Tracking

Se relacionan como ejemplo los casos de uso más significativos.

Nombre	CU001 Login
Descripción	Acceder al sistema
Actor principal	Administrador o usuario
Ámbito	Sistema J-Tracking
Nivel	Tarea
Intereses	Un usuario quiere utilizar alguna función del sistema
Precondición	El usuario no se encontrará logeado
Postcondición	El usuario accederá a la función solicitada
Garantías mínimas	Recibirá una pantalla de error
Garantías en caso de éxito	Accederá al área solicitada
Escenario principal de éxito	
1	El usuario solicita la pantalla de bienvenida de la aplicación
2	El sistema muestra la pantalla de bienvenida
3	El usuario selecciona área de administración o explotación
4	El sistema muestra pantalla de login
5	El usuario introduce usuario y password y pulsa login
6	El sistema valida al usuario y muestra la pantalla del área solicitada
Escenarios alternativos	
6a1	El sistema no valida al usuario y muestra la pantalla de error.
6a2	El usuario vuelve a la página de bienvenida
6b1	El sistema no autoriza acceso al área solicitada al rol del usuario
6b2	El sistema devuelve el error forbidden

Nombre	CU004 Administración de usuarios
Descripción	Crear, actualizar, eliminar y listar usuarios
Actor principal	Administrador
Ámbito	Sistema J-Tracking
Nivel	General
Intereses	Un administrador quiere gestionar usuarios
Incluye	CU003. Administración de entidades
Escenario principal de éxito	
1	El administrador hace uso del caso de uso incluido
2	La entidad es usuario
3	Los datos de creación son: login, nombre, apellido, email, número de teléfono,

	role y password
4	Los datos de actualización son: nombre, apellido, email, número de teléfono, role y password.
Escenarios alternativos	

Nombre	CU006 Administración de vehículos
Descripción	Crear, actualizar, eliminar y listar vehículos
Actor principal	Administrador
Ámbito	Sistema J-Tracking
Nivel	General
Intereses	Un administrador quiere gestionar vehículos
Incluye	CU003. Administración de entidades
Escenario principal de éxito	
1	El administrador hace uso del caso de uso incluido
2	La entidad es un vehículo
3	Los datos de creación son: matrícula, marca, modelo, color
4	Los datos de actualización son: marca, modelo, color, misión y dispositivo
Escenarios alternativos	

Nombre	CU011 Mostrar posiciones en el mapa
Descripción	Mostrar posiciones en el mapa
Actor principal	Administrador o usuario
Ámbito	Sistema J-Tracking
Nivel	Tarea
Intereses	Un actor quiere ver posiciones en el mapa
Precondición	El actor se encontrará logeado
Postcondición	
Garantías mínimas	Recibirá mensaje de error
Garantías en caso de éxito	Se mostrará un mapa con el track
Escenario principal de éxito	
1	El actor solicita representar posiciones
2	El sistema muestra la pantalla de búsqueda
3	El actor introduce vehículo y fechas
4	El sistema muestra las posiciones disponibles junto con un track en el mapa

Escenarios alternativos	

Nombre	CU020 Ver estado de un dispositivo
Descripción	Ver el estado de un dispositivo
Actor principal	Administrador o usuario
Ámbito	Sistema J-Tracking
Nivel	General
Intereses	Ver el estado de un dispositivo
Precondición	El dispositivo estará registrado en el sistema
Postcondición	
Garantías mínimas	Se recibirá un mensaje de error
Garantías en caso de éxito	Se obtendrá el estado de un dispositivo
Escenario principal de éxito	
1	Se solicita ver el estado de un dispositivo
2	El sistema muestra los datos de nivel de batería, señal GSM, y estado
	Escenarios alternativos
2a	No hay datos disponibles
2b	Los campos no disponibles se muestran en blanco

Nombre	CU021 Obtener configuración
Descripción	Un dispositivo quiere actualizar su configuración
Actor principal	Dispositivo
Ámbito	Sistema J-Tracking
Nivel	General
Intereses	Obtener una nueva configuración de funcionamiento
Precondición	El dispositivo estará registrado en el sistema
Postcondición	
Garantías mínimas	Los comandos pasaran a estado enviado
Garantías en caso de éxito	El dispositivo modificará su configuración
Escenario principal de éxito	
1	Se solicitan los cambios de configuración
2	El sistema envía los cambios pendientes para el dispositivo
	Escenarios alternativos
2a	No hay cambios disponibles
2b	Finaliza el caso de uso

Nombre	CU021 Control en tiempo real
Descripción	Un usuario quiere controlar un vehículo en tiempo real
Actor principal	Administrador o usuario
Ámbito	Sistema J-Tracking
Nivel	General
Intereses	Control de un vehículo en tiempo real
Precondición	El usuario se encontrará logeado
Postcondición	
Garantías mínimas	Se recibirá la ultima posición disponible
Garantías en caso de éxito	Se recibirá la ultima posición actualizada
Escenario principal de éxito	
1	Se solicita tiempo real sobre un vehículo
2	El muestra el mapa con las últimas posiciones del vehículo y las actualiza periódicamente
	Escenarios alternativos
2a	No hay posiciones disponibles
2b	Se muestra el mapa por defecto

2.4 Requisitos no funcionales

2.4.1 Requisitos de seguridad

El acceso a la aplicación se realizará mediante alguna de las modalidades de autenticación declarativa disponibles, Basic, Form o Certificados.

La capa de transporte se protegerá mediante SSL.

Las passwords se encontrará encriptadas dentro de la base de datos.

Las sesiones tendrán caducidad limitada, expirado el tiempo será necesaria una nueva autenticación.

2.4.2 Requisitos del sistema de cartografía

El sistema permitirá la representación de posiciones de los dispositivos filtrados entre

fechas.

El sistema será público, no se destinarán recursos económicos para el funcionamiento del software otros que el pago de licencias si el proveedor de servicio decidiera establecer una cuota por uso.

A ser posible dispondrá de cartografía vectorial e imagen aérea.

El sistema debe permitir la utilización de clientes ligeros tipo navegadores web.

2.4.1 Interfaz común del sistema J-Tracking

El sistema J-Tracking implementa una interfaz de salida hacia los dispositivos de control de flotas consistente en lo siguientes mensajes:

1. Jradia #start sending positions#
2. Jradia #stop sending positions#
3. Jradia #get battery level#
4. Jradia #get signal strength#
5. Jradia #get status#

El sistema J-Tracking implementa una interfaz de entrada que aceptará los siguientes mensajes procedentes de los dispositivos de control de flotas:

1. Jradia-01 -nnn position, fecha, hora, latitud, indicador N-S, longitud, indicador E-O,altura, velocidad, rumbo, estado GPS. Donde el estado GPS puede ser 0 para indicar sin posición, 2 para posiciones 2D y 3 para posiciones 3D.
2. Jradia-01 -nnn battery: nnnn mV.
3. Jradia-01 -nnn GSM signal strength: -nnn dbm.
4. Jradia-01 -nnn status: estado de GSM, estado de GPS

2.5 Interfaces externas

2.5.1 Interfaz gráfica de usuario

Para la consecución de los requisitos funcionales, el sistema dispondrá de una interfaz gráfica de usuario basada en navegador web. Se dispondrá de tantas vistas como sean necesarias para la consecución de los casos de uso de la especificación.

Se ha considerado un diagrama de navegabilidad para el conjunto de la aplicación, donde se diferencian las capacidades de los dos roles identificados en el sistema, administrador (ADMIN) y usuario no privilegiado (USER). De esta forma todas las tareas asociadas al área de administración solamente podrán ser accedidas por usuarios con rol administrador.

El área de explotación podrá ser accedido por cualquiera de los roles disponibles en el sistema. Se obtiene de esta forma una estructura jerárquica que permite a los administradores un acceso total al sistema y a los usuarios no privilegiados un acceso restringido a sus labores de explotación y control.

2.5.2 Diagramas de navegabilidad

En el documento Anexo II Prototipos _GUI_J-Tracking.pdf se muestra el diagrama de navegabilidad completo del GUI, así como los prototipos de las ventanas de la interfaz gráfica de usuario.

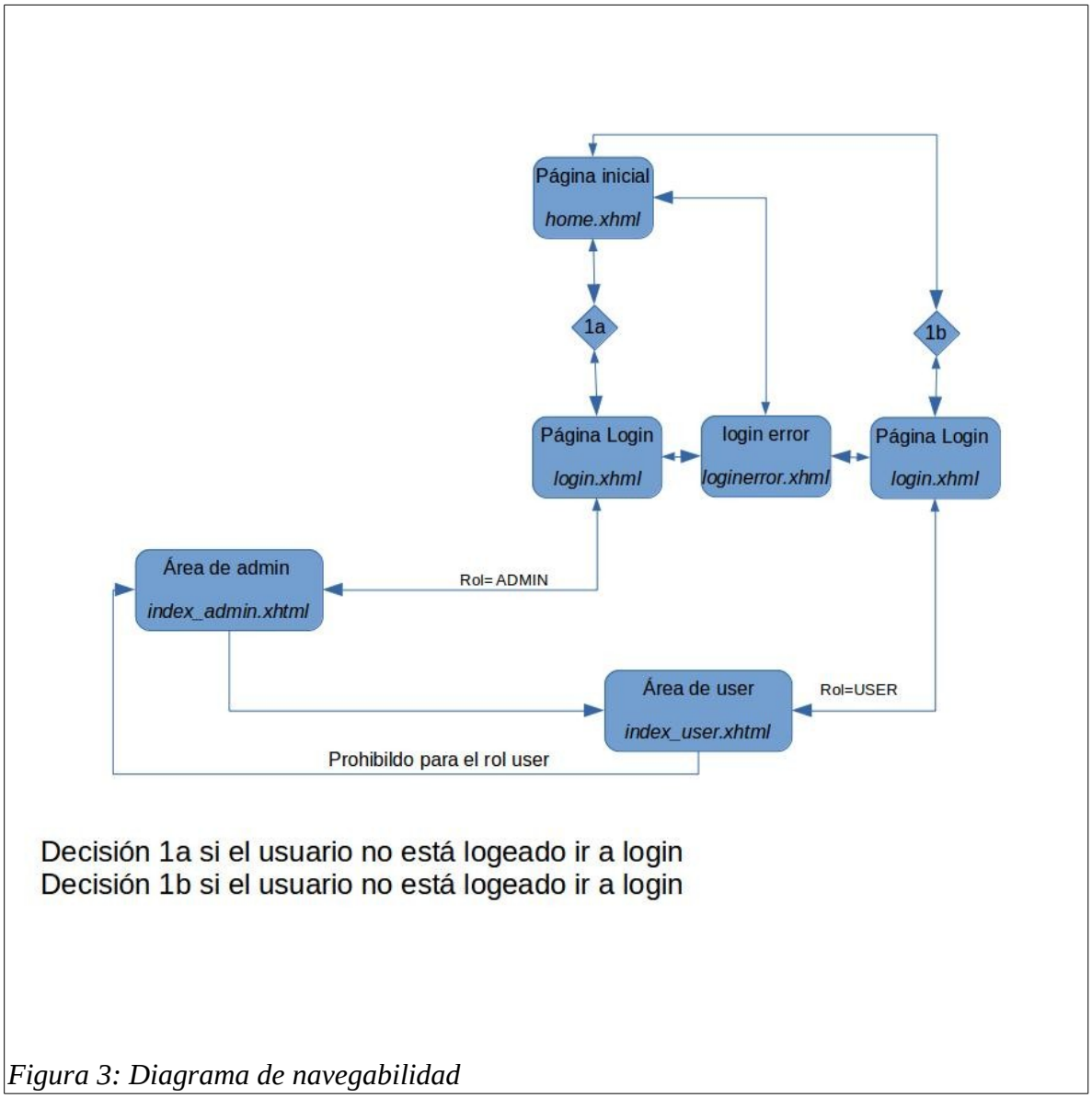


Figura 3: Diagrama de navegabilidad

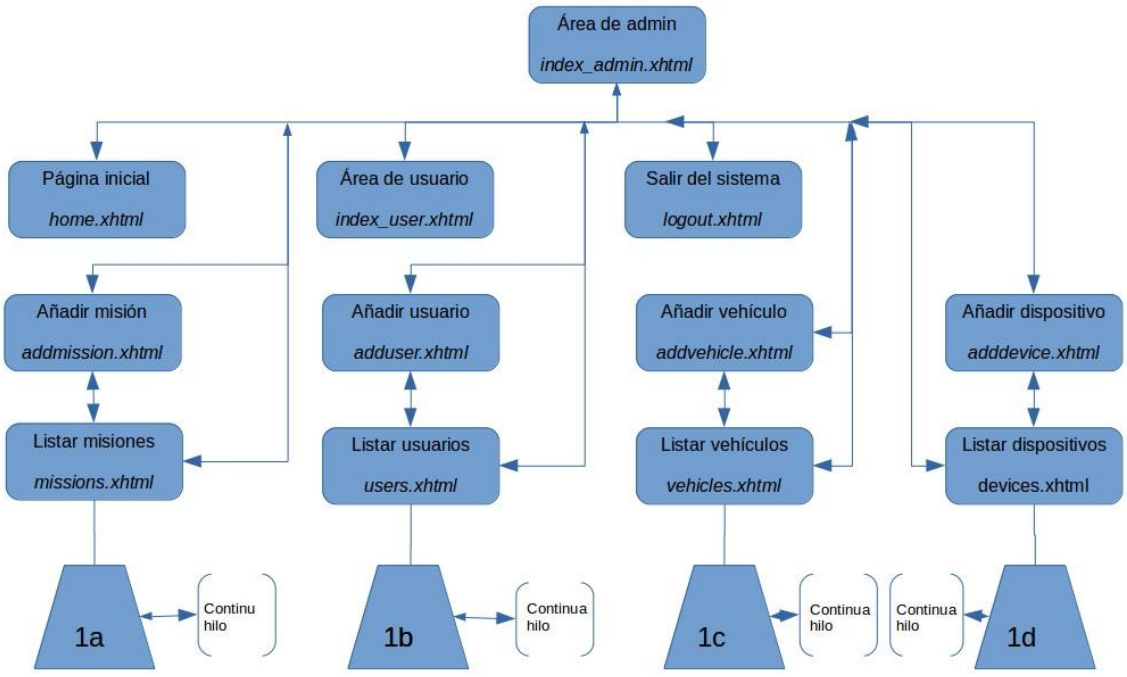


Figura 4: Navegabilidad 2

2.5.3 Vistas

Para la consecución de los casos de uso en los que se requiere la participación de actores con roles ADMIN o USER se necesita de una serie de vistas que permitan la interacción con el sistema. Las siguientes pantallas presentan una aproximación de las vistas que deben satisfacer el diagrama de navegabilidad descrito anteriormente.

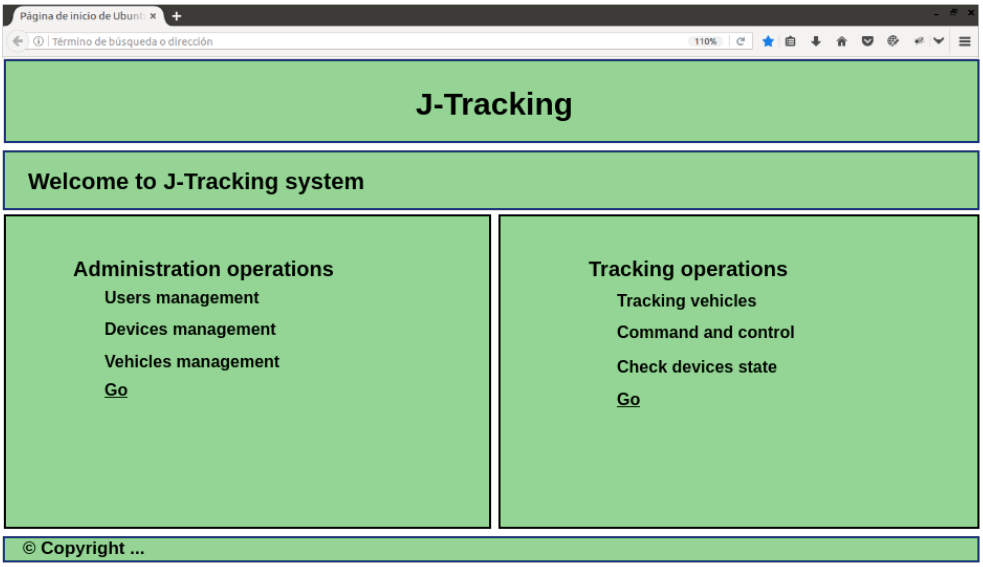


Figura 5: home.xhtml

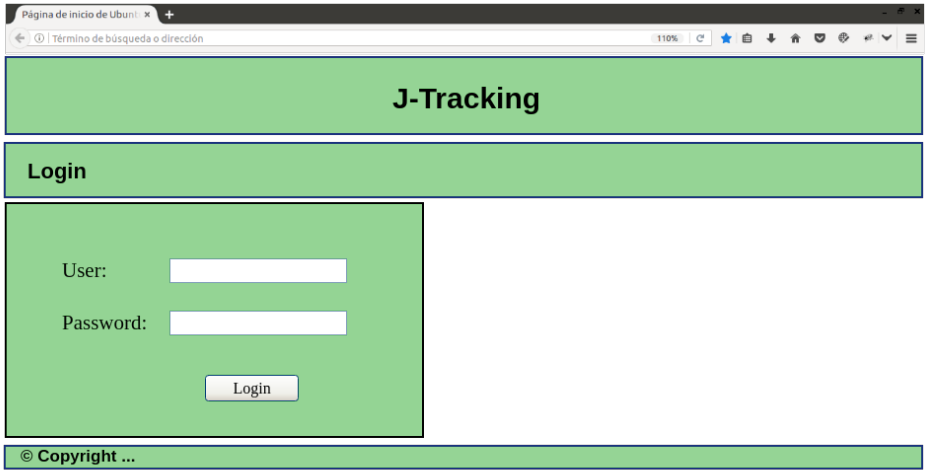


Figura 6: login.xhtml

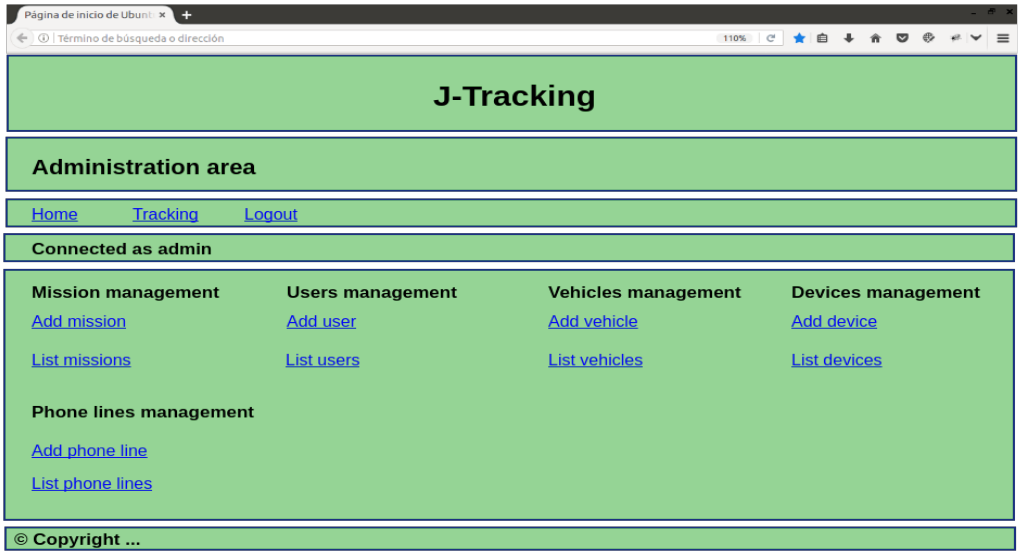


Figura 7: index_admin.xhtml

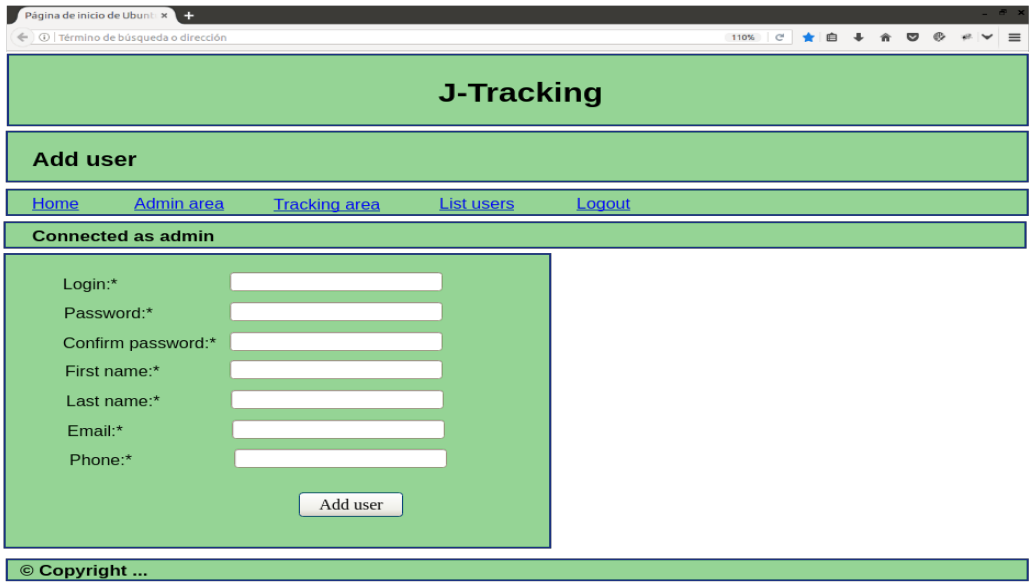


Figura 8: adduser.xhtml

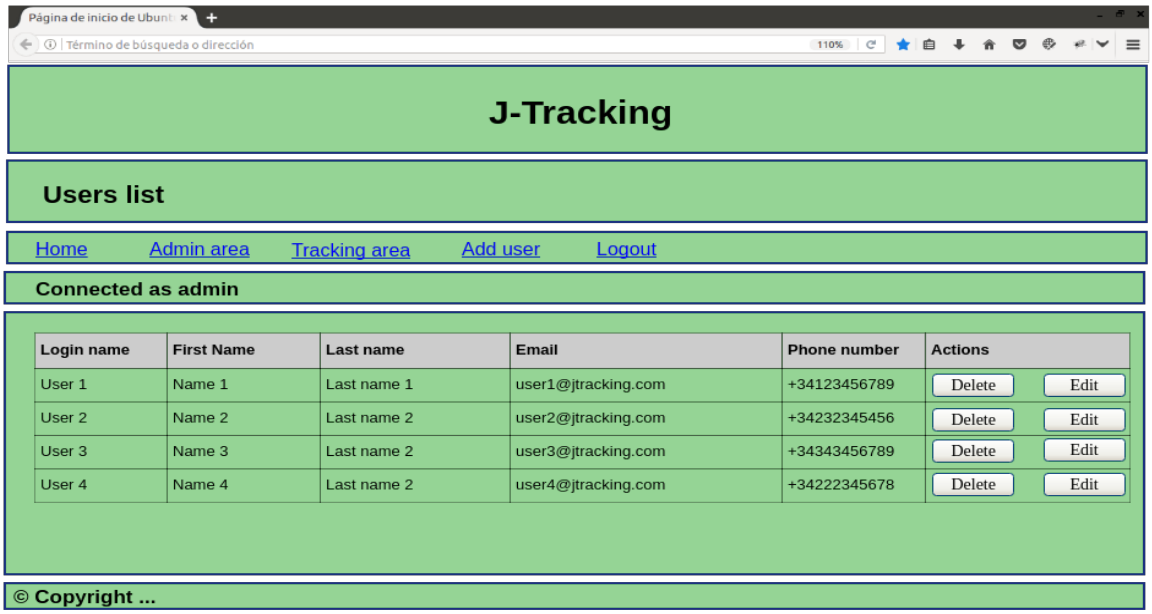


Figura 9: users.xhtml

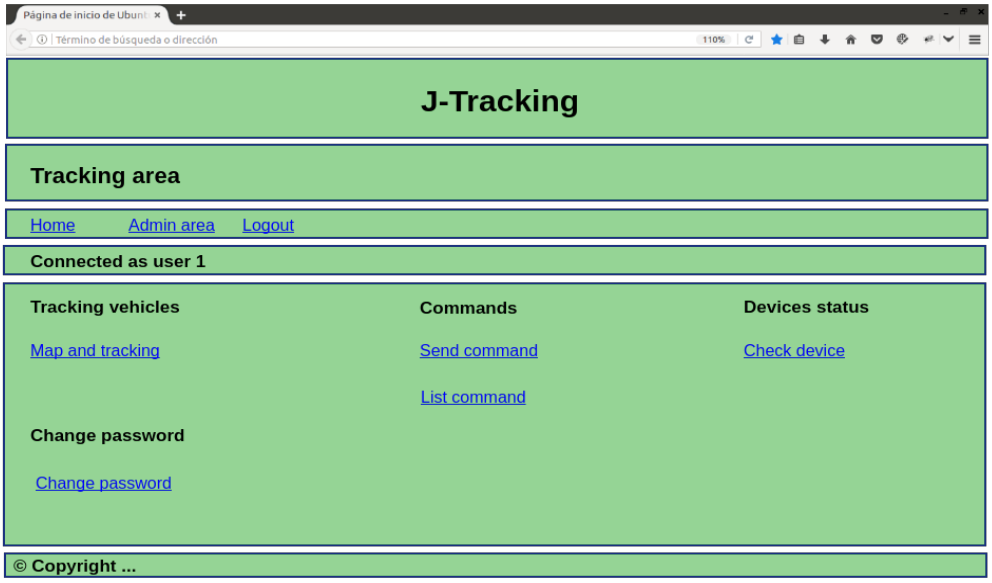


Figura 10: index_user.xhtml

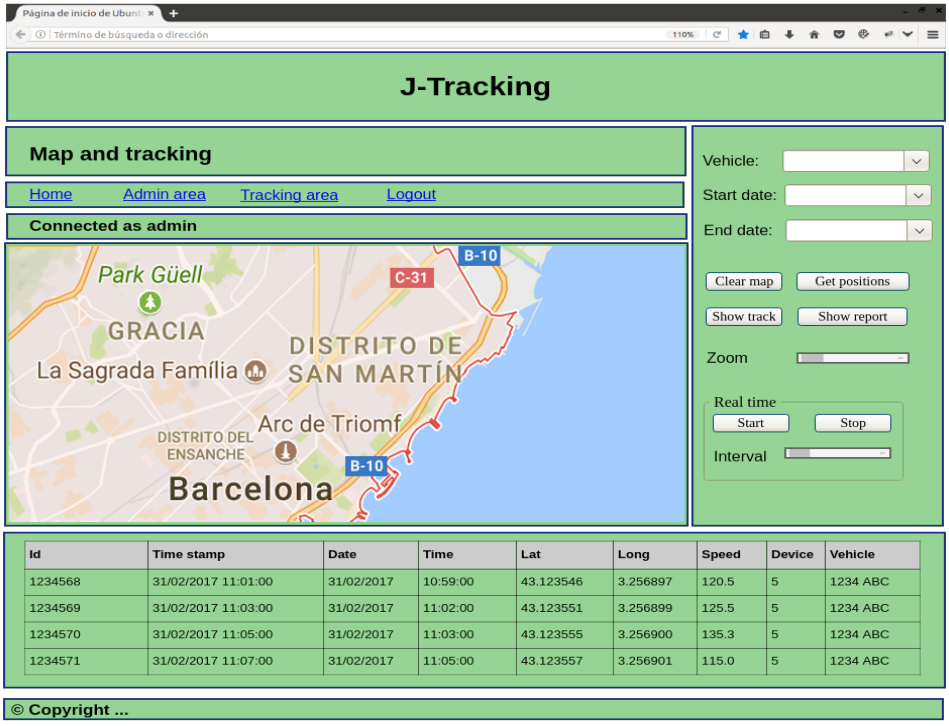


Figura 11: mapping.xhtml

3 Vista lógica

3.1 Análisis

Como consecuencia del análisis de los requisitos y del dominio del problema, se han identificado las entidades reflejadas en el modelo de dominio. Como se observa, todas ellas serán persistentes.

El mapeo entre el modelo del dominio y el esquema de la base de datos se ha realizado por medio de framework de persistencia JPA. Se ha respetando la navegabilidad, multiplicidad y relaciones de herencia reflejadas en el mismo. En consecuencia se ha obtenido una base de datos normalizada y que representa un fiel reflejo de la semántica capturada en el diagrama de clases.

En este punto destaca la importancia de establecer relaciones correctas entre las

entidades que forman el modelo del dominio, de lo contrario el mapeo ORM generado incurrirá en las trampas de diseño típicas del modelado de base de datos, abanico, perdida de afiliación, corte, etc.

Se debe tener especial cuidado de no incluir por error claves foráneas en los atributos de las entidades, lo que originaría falta de normalización de la base de datos generada.

Se han tomado entre otras las siguiente decisiones de diseño:

- Al objeto de dar cabida a diferentes dispositivos del sistema se ha modelado un clase abstracta DeviceJPA, e introducido una subclase que correspondería a a un dispositivo concreto; el dispositivo se ha denominado Jradia01JPA y observa un comportamiento polimórfico de la interfaz única del sistema. Esta estrategia, al definir comportamientos específicos en las subclases, permite cierta flexibilidad a la hora incorporar dispositivos al sistema.
- Los vehículos vinculados a una misión dependen de esta. Si se borra la misión lo harán los vehículos vinculados mediante un borrado en cascada.
- Las posiciones se encuentran vinculadas con el vehículo al que pertenecen, la eliminación de un vehículo provoca el borrado en cascada de las posiciones del mismo.
- Los comandos se encuentran vinculados a los dispositivos, un borrado de un dispositivo elimina los comandos relacionados.
- La eliminación de entidades vinculadas no es posible en los siguientes casos:
    - Eliminar un dispositivo que está asignado a un vehículo, notificando el error.
    - La eliminación de un dispositivo que tiene una linea telefónica asignada no es posible se notifica el error.
    - El usuario por defecto *admin* con password *master* no puede ser borrado ni

actualizado, tan solo se permite cambiar su password.

- Los comandos no son modificables tan solo se crean, se consultan y eliminan.
- La enumeración menos evidente GpsState identifica el tipo de posición suministrada por un dispositivo GPS siendo posible posiciones en dos dimensiones o en tres, TWO_D, THREE_D.
- La política de generación de claves primarias decidida ha consistido en asignar claves simples de un solo campo. En el caso en que sean necesarios varios campos para generación de las claves primarias se utilizarán claves auto-numéricas generadas por el mismo SGDB. Al tratarse de una base de datos de nueva creación ha descartado la utilización de claves primarias compuestas.[4]
- Para el mapeo de la herencia en la base de datos se ha utilizado la estrategia Single Table, de forma que todas las entidades que hereden de DevideJPA quedarán mapeadas a esta tabla e identificadas por medio del discriminador por defecto. Se ha considerado este diseño al disponer de un modelo del dominio donde no está prevista la aparición de valores nulos en los campos.
- Se ha creado un índice para la entidad PositionJPA, sobre los campos Vehicle y PositionTimeStamp de forma que se mejore el rendimiento en las consultas para la explotación de información.
- Los accesos a la tabla de posiciones se realizará mediante sentencias JPQL. Se considera que resulta mucho más eficiente esta modalidad de acceso que la recuperación de una entidad DeviceJPA y acceder de forma *lazy* al atributo tipo Collection para recuperar las posiciones.
- Para el resto de accesos se utilizarán las operaciones de la interfaz Entity Manager sin crear consultas JPQL.

4 JSR-338 Java Persistence Specification, pg 30.

3.2 Modelos de la vista lógica

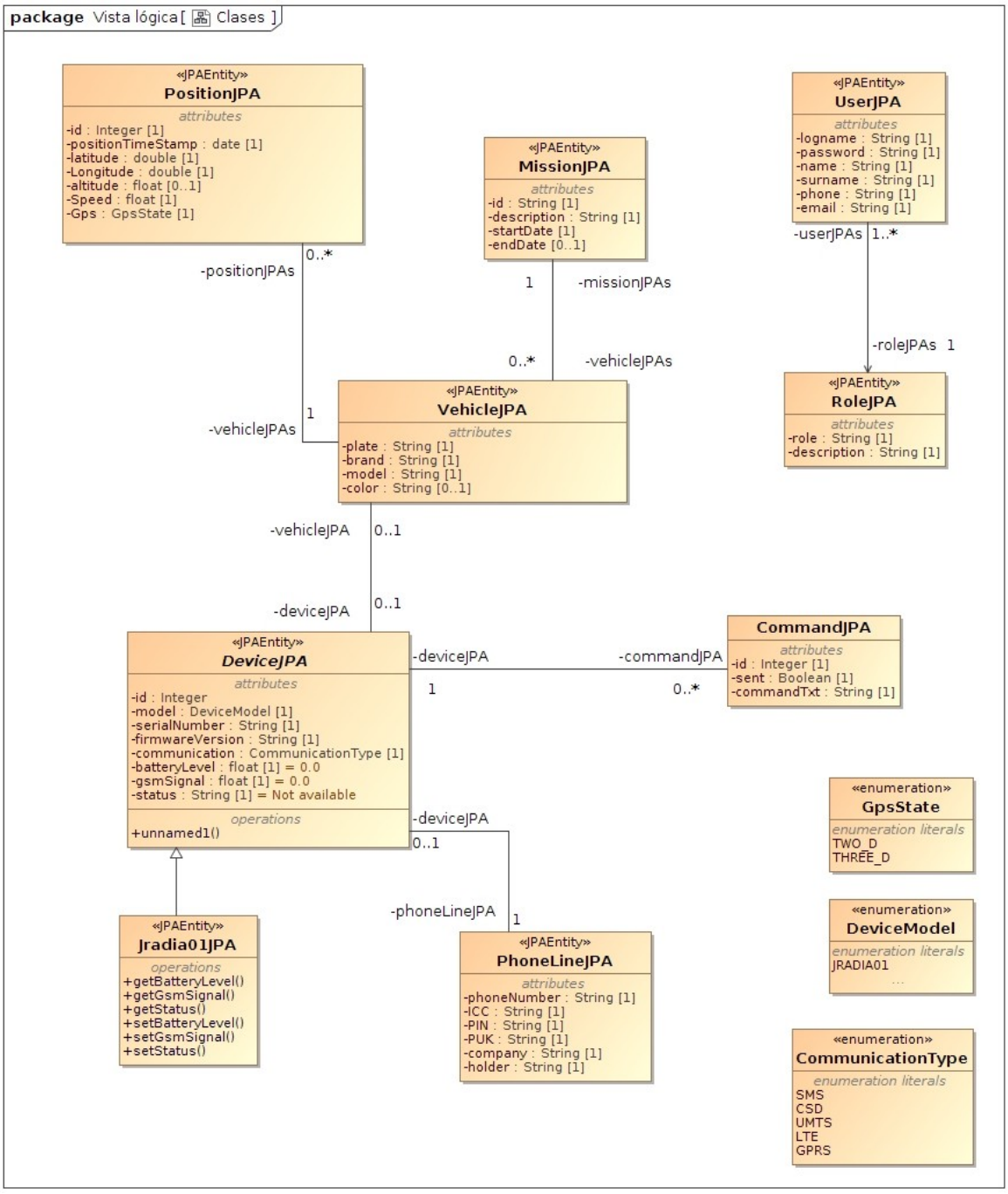


Figura 12: Modelo del dominio

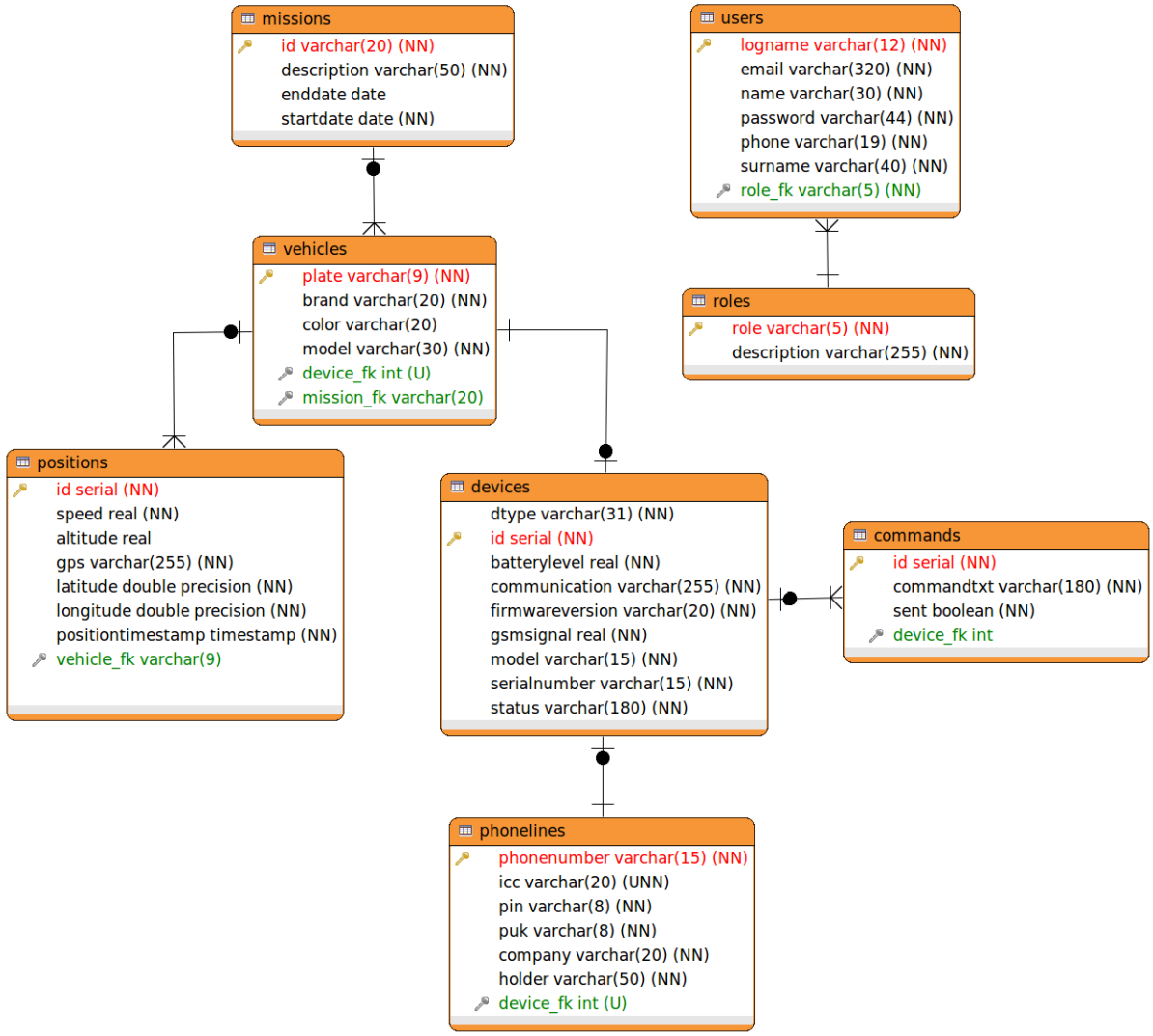


Figura 13: Diagrama ORM

3.2.1 Descripción de la base de datos J-Tracking

Se describen las tablas generadas como consecuencia del mapeo objeto-relacional.

Base de Datos: postgres

- **Tablas del esquema jtracking**

- commands
- devices
- missions
- phonelines
- positions
- roles
- users
- vehicles

Tabla: commands

F-Key	Name	Type	Description
	id	serial	*PRIMARY KEY*
	commandtxt	character varying(180)	*NOT NULL*
	sent	boolean	*NOT NULL*
devices.id	device_fk	integer	

Table: devices

F-Key	Name	Type	Description
	dtype	character varying(31)	*NOT NULL*
	id	serial	*PRIMARY KEY*
	batterylevel	real	*NOT NULL*
	communication	character varying(255)	*NOT NULL*
	firmwareversion	character varying(20)	*NOT NULL*
	gsmsignal	real	*NOT NULL*
	model	character varying(15)	*UNIQUE#1 NOT NULL*
	serialnumber	character varying(15)	*UNIQUE#1 NOT NULL*
	status	character varying(180)	*NOT NULL*

Referenciada vía clave foránea por:

- commands
- phonelines
- vehicles

Table: missions

F-Key	Name	Type	Description
	id	character varying(20)	*PRIMARY KEY*
	description	character varying(50)	*NOT NULL*
	enddate	date	
	startdate	date	*NOT NULL*

Referenciada vía clave foránea por:

- vehicles

Table: phonelines

F-Key	Name	Type	Description
	phonenumber	character varying(15)	*PRIMARY KEY*
	icc	character varying(20)	*UNIQUE NOT NULL*
	pin	character varying(8)	*NOT NULL*
	puk	character varying(8)	*NOT NULL*
	company	character varying(20)	*NOT NULL*
	holder	character varying(50)	*NOT NULL*
devices.id	device_fk	integer	*UNIQUE*

Table: positions

F-Key	Name	Type	Description
	id	serial	*PRIMARY KEY*
	speed	real	*NOT NULL*
	altitude	real	
	gps	character varying(255)	*NOT NULL*
	latitude	double precision	*NOT NULL*
	longitude	double precision	*NOT NULL*
	positiontimestamp	timestamp without time zone	*NOT NULL*
vehicles.plate	vehicle_fk	character varying(9)	

Indice: positionindex vehicle_fk, positiontimestamp

Table: roles

F-Key	Name	Type	Description
	role	character varying(5)	*PRIMARY KEY*
	description	character varying(255)	*NOT NULL*

Referenciada vía clave foránea por:

- users

Table: users

F-Key	Name	Type	Description
	logname	character varying(12)	*PRIMARY KEY*
	email	character varying(320)	*NOT NULL*
	name	character varying(30)	*NOT NULL*
	password	character varying(44)	*NOT NULL*

	phone	character varying(19)	*NOT NULL*
	surname	character varying(40)	*NOT NULL*
roles.role	role_fk	character varying(5)	*NOT NULL*

Table: vehicles

F-Key	Name	Type	Description
	plate	character varying(9)	*PRIMARY KEY*
	brand	character varying(20)	*NOT NULL*
	color	character varying(20)	
	model	character varying(30)	*NOT NULL*
devices.id	device_fk	integer	*UNIQUE*
missions.id	mission_fk	character varying(20)	

Referenciada vía clave foránea por:

- positions

4 Vista de desarrollo

4.1 Análisis

La información obtenida acerca de dispositivos, usuarios, contextos de utilización y organizaciones permite definir el modelo arquitectónico de alto nivel que ha guiado el desarrollo de la aplicación.

Considerando las características de los usuarios y los contextos de utilización, se ha concluido que el modelo arquitectónico que mejor se adapta a las necesidades y características es un modelo heterogéneo, en capas que haga uso de la arquitectura servidor cliente.

Dentro del modelo arquitectónico anterior se deberán considerar las diferentes

opciones de implementación. Atendiendo a las necesidades de los usuarios el aplicativo podría implementarse en dos o tres capas, pudiendo utilizar cliente ligero tipo navegador o pesado tipo aplicación de escritorio.

Atendiendo a las restricciones impuestas por los escenarios y las características de los usuarios y organizaciones, se han descartado las arquitecturas basadas en clientes pesados. De esta forma, se evita la problemática de la instalación de paquetes de software y la resolución de dependencias para el funcionamiento del aplicativo. Al mismo tiempo se evita tener que contar con personal especializado que solvente las incidencias, los desplazamientos a zonas de conflictivas o el envío de equipos pre-configurados.

La elección de cliente ligero impone una solución web, pudiendo implementarse en diferentes capas, se ha optado por utilizar tres capas. Se diferenciará en el lado del servidor una capa de presentación, otra de negocio y otra de persistencia. Este modelo arquitectónico permite la re-ubicación de capas completas en servidores diferentes, como pudiera ser la migración de la capa de persistencia.

El modelo arquitectónico anterior no resuelve de forma completa todas las necesidades. Los dispositivos de localización embarcados en los vehículos de la flota necesitan acceder al sistema para enviar sus posiciones, mensajes con información de control y descargar los cambios de configuración.

Determinados dispositivos no contarán con capacidades IP para acceder al sistema o las zonas de operación no ofrecerán redes de comunicaciones IP inalámbricas por lo que será las pasarelas las encargadas de realizar el intercambio de información entre el dispositivo y el sistema.

Para tratar con la heterogeneidad de dispositivos y medios de comunicación se ha decidido implementar la parte de la capa de negocio dedicada al intercambio de información con los dispositivos con una arquitectura orientada a servicios.

4.2 Análisis y arquitectura de alto nivel

4.2.1 Modelo de la arquitectura

Basado en el análisis realizado se documenta un diagrama de componentes que modela las capas de la arquitectura elegida.

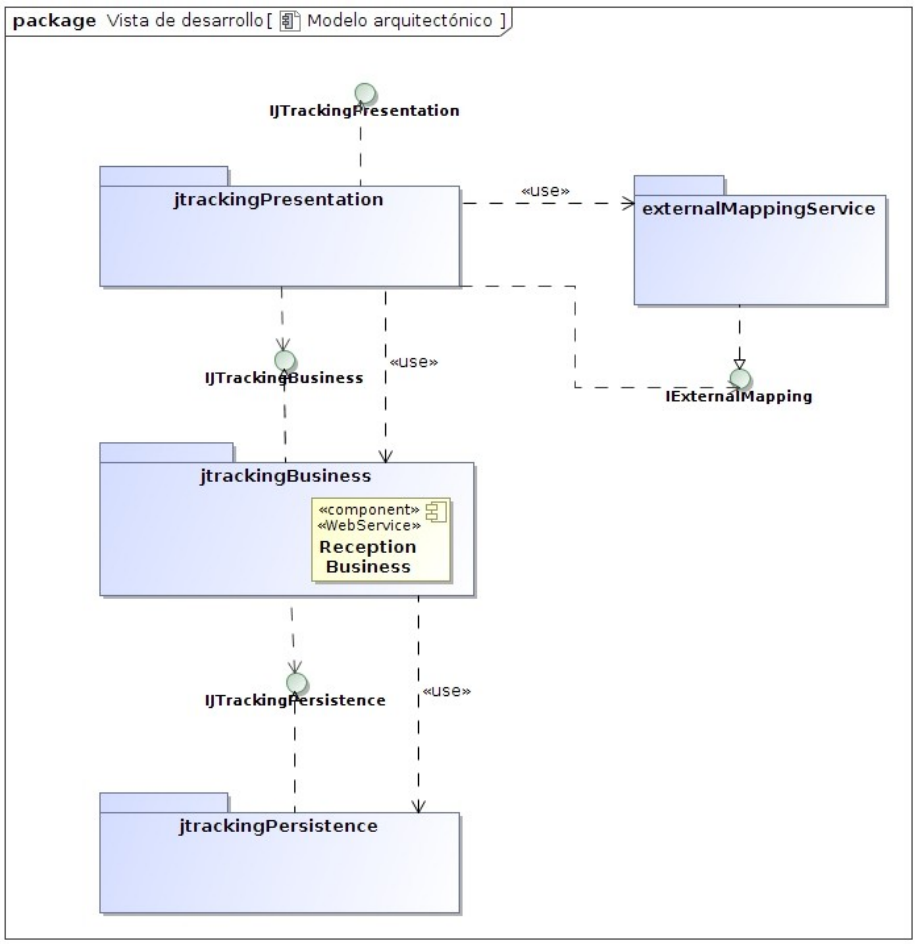


Figura 14: Modelo arquitectónico

4.3 Análisis y modelado de la arquitectura independiente de la tecnología

4.3.1 Análisis

Una vez definido el modelo arquitectónico de alto nivel, se ha realizado un proceso de refinado, partiendo de los casos de uso y de su agrupación entono a componentes dedicados a su implementación.

Se realiza una segregación de interfaces, especializando los componentes de alto nivel y utilizando un patrón fachada para concentrar las operaciones de una interfaz en un solo componente. Se ha desestimado la utilización de una facha común para toda la capa de negocio dado que el acceso entre capa de presentación y de negocio será local.

4.3.2 Modelo de la arquitectura

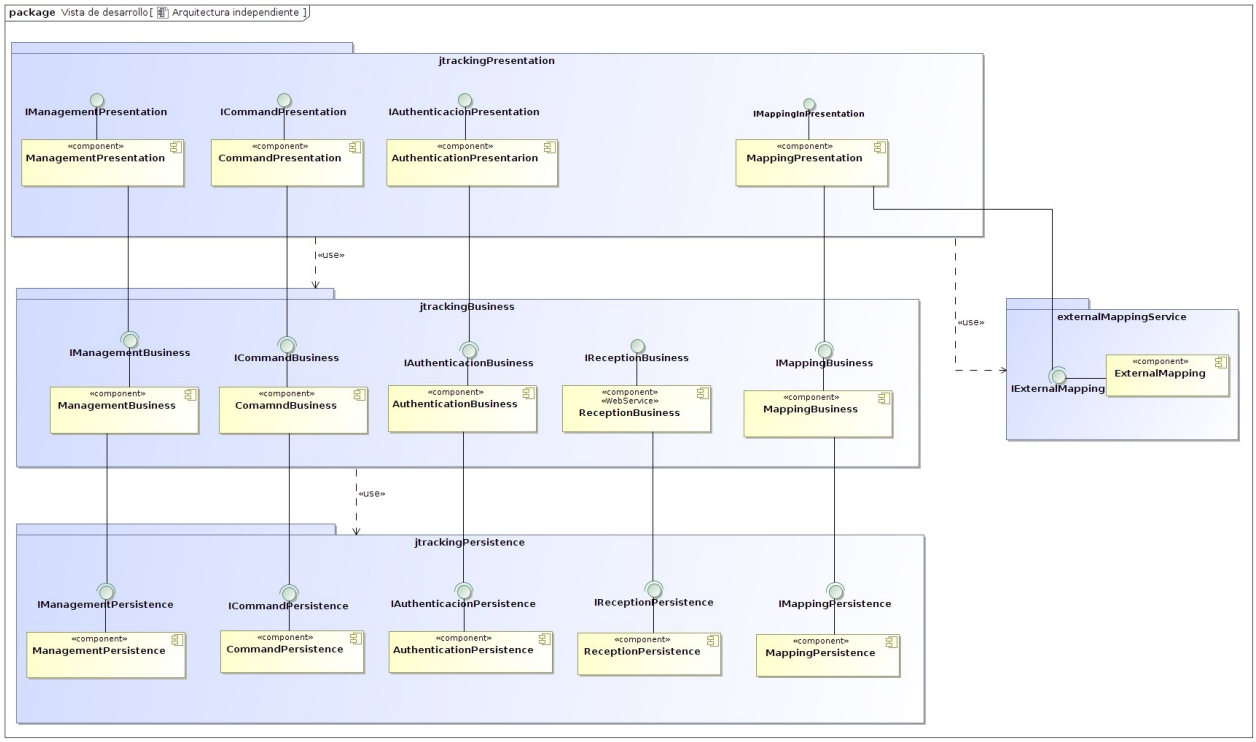


Figura 15: Arquitectura independiente de la tecnología

4.4 Análisis y modelado de la arquitectura Java EE7

4.4.1 Análisis

Una vez determinada la arquitectura independiente se plantea la elección de una tecnología que permita la implementación de una aplicación multicapa, que pueda ser utilizada por medio de clientes ligeros, sin necesidad de la instalación de software en los mismos. Al mismo tiempo deberá satisfacer la necesidad de poder interactuar con dispositivos y pasarelas heterogéneas a través de un servicio web.

Se ha optado por la utilización de la plataforma Java Enterpise Edition Versión 7 de Oracle. Se considera que es una plataforma madura y que puede satisfacer todos los requisitos no funcionales en cuanto a escalabilidad, fiabilidad, rendimiento y portabilidad y las restricciones establecidas en cuanto a la utilización de software libre.

4.4.2 Capa de presentación

La capa de presentación se ha implementado por medio de Java Server Faces, utilizando Facelets y el framework de presentación Primefaces 6 en su versión libre, Comunity Edition. La elección de este framework asegura la disponibilidad de componentes gráficos de calidad y con funcionalidades avanzadas.

La elección de Primefaces ha permitido mejorar la a usabilidad de la aplicación y la experiencia de usuario, al tiempo que facilita el desarrollo al disponer de componentes con capacidades difíciles de implementar directamente con las implementaciones de referencia de JSF.

La elección de Primefaces como framework de presentación ha estado motiva por la decisión de no continuar con el desarrollo de RichFaces por parte de su equipo de desarrollo. De no haber producido esta situación casi con toda probabilidad se hubiera

utilizado este framework incluido por defecto en Wildfly.[5]

4.4.3 Capa de negocio

Para la capa de negocio se ha utilizado Enterprise Java Beans 3.2 lo que garantiza una implementación sencilla, escalable y robusta.

En cuanto al servicio web necesario para atender las necesidades de envío y recepción de información desde y hacia los dispositivos, se considera que la solución más adecuada es la utilización de servicios SOAP. Admite la utilización de diferentes protocolos, no solamente HTTP/HTTPS, lo que permite consumir el servicio desde plataformas diversas y se adapta perfectamente a los requisitos no funcionales del proyecto, a la heterogeneidad de dispositivos y a las posibles plataformas que fuera necesario implementar.

4.4.4 Capa de persistencia

La capa de persistencia se ha implementado utilizando una estrategia top-down, por medio de Java Persistence API 2.1 lo que ha facilitado notablemente todas las tareas de persistencia, al tiempo que aligera parcialmente las labores de diseño conceptual y lógico de la base de datos, delegando en el framework de persistencia las labores de mapeo objeto-relacional.

La utilización de JPA desacopla la capa de negocio de la de persistencia y al mismo tiempo facilita la sustitución completa del SGDB.

Como gestor de base de datos se utilizará PostgreSQL si bien podría utilizarse otro SGDB con adaptaciones mínimas en el código y metadatos de las entidades a persistir.

La elección de de JPA como modelo de persistencia supone la utilización de facto de un patrón DAO genérico. Si observamos las operaciones proporcionadas por la

5 https://developer.jboss.org/people/michpetrov/blog/2016/02/12/the-future-of-richfaces

interfaz EntityManager[6], veremos que implementa las operaciones típicas de creación, eliminación, actualización y consulta de entidades en una base de datos relacional.

Se ha desestimado la creación de un objeto u objetos DAO genéricos o específicos para la aplicación. Se considera que no realizarían una gran aportación al diseño, otro que agrupar las llamadas de persistencia en determinados objetos, con el coste añadido que supondría introducir elementos de indirección.

4.4.5 Capa de servicios transversales

Algunos de los servicios transversales facilitados por los servidores Java EE pueden ser personalizados a través de la configuración de los contenedores. En el proyecto J-Tracking se ha recurrido a la configuración de algunos parámetros básicos de los contenedores para la utilización específica de algunos de estos servicios.

Los servicios de seguridad se han configurado de forma declarativa. De esta forma el proceso de autenticación y autorización de usuarios lo realiza el contenedor web siguiendo la especificación JSR-340 Java Servlet 3.1. Igualmente la capa de trasporte se ha asegurado de forma declarativa siguiendo la especificación anterior. De esta forma la interacción con el usuario se realiza a través de protocolo HTTPS.

Las labores de persistencia de la aplicación se deben llevar a cabo en un contexto transaccional. En el proyecto J-Tracking el contexto transaccional lo proporcionará el contenedor y se gestionará a través de la interfaz EntityManager, la configuración se realiza de forma declarativa por medio de la *Persistence Unit.*

4.4.6 Servidor de aplicaciones

Para la ejecución de la aplicación J-Tracking se seleccionado el servidor de aplicaciones Wildfly 10.1. La motivación principal es el conocimiento que se ha adquirido del producto a lo largo de las asignaturas cursadas. Se ha considerado el hecho de ser una implementación Java EE 7 completa y certificada por Oracle.

6 http://docs.oracle.com/javaee/7/api/javax/persistence/EntityManager.html

Merece la pena destacar que Red Hat ha realizado cambios en el middleware Wildfly. La versión 10.1 utiliza como contenedor de Servlets Undertow en lugar de Tomcat. Algunos análisis muestran un mejor rendimiento y una menor ocupación de memoria en comparación con otros productos competidores. En cuanto a la configuración de los servicios transversales proporcionados por este contenedor, se sigue utilizando los mismos descriptores de despliegue estandarizados de la especificación JEE 7.

4.4.7 Modelo de la arquitectura

El siguiente diagrama de componentes muestra una descripción y modelado de la arquitectura de la aplicación de alto nivel para la tecnología Java EE.

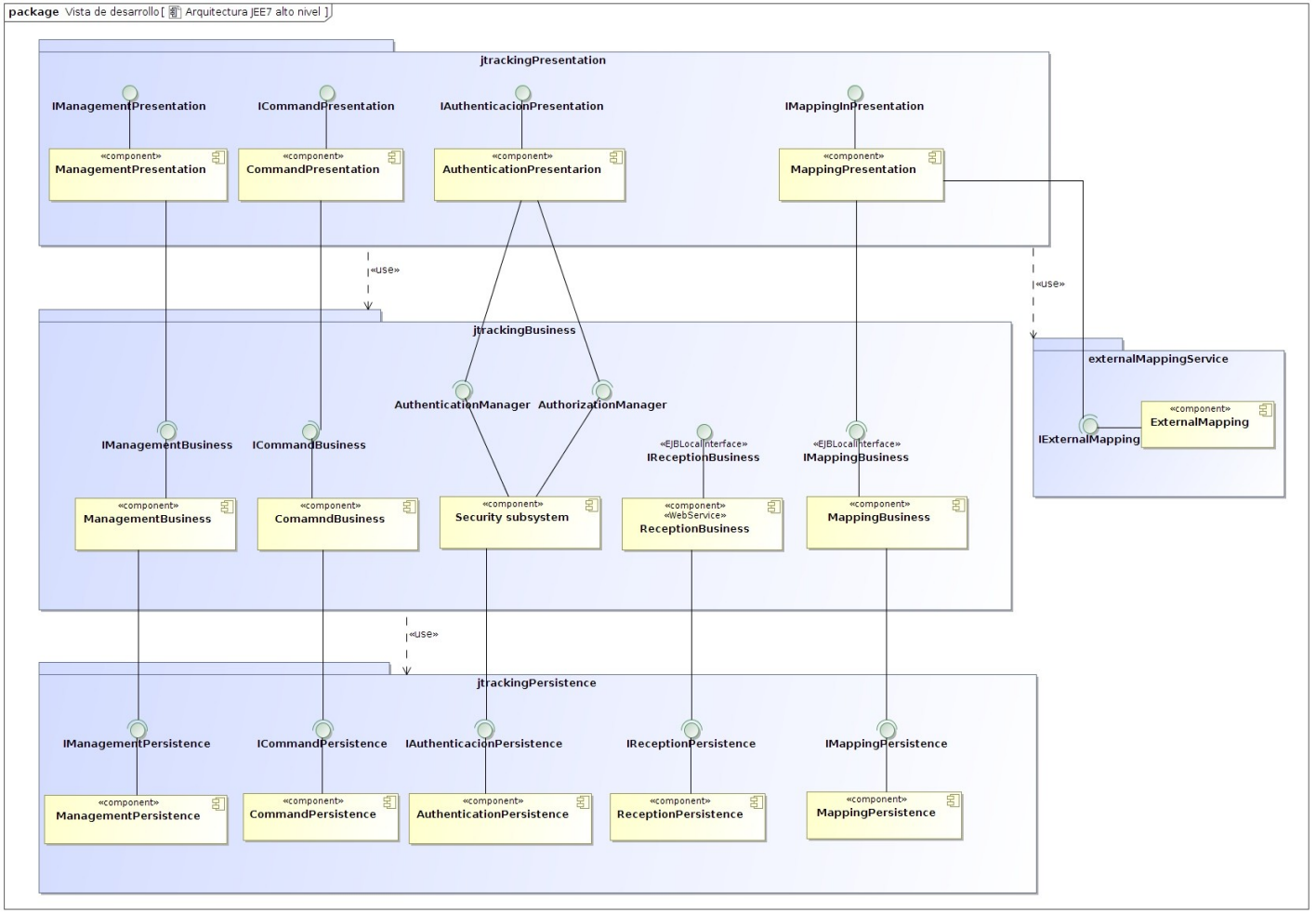


Figura 16: Arquitectura Jee 7 de alto nivel

En esta primera aproximación se evidencia la utilización de los servicios de autenticación y autorización. Si bien se han modelado en la capa de negocio, en la arquitectura detallada se mostrará la implementación de estos mecanismos con máximo detalle en la capa donde tengan lugar.

4.5 Análisis y arquitectura detallada Java EE 7

4.5.1 Análisis

La elección de Java EE 7 permite la selección de diferentes tecnologías para cada capa del modelo. En los detalles de cada capa se abunda en el análisis de las elecciones efectuadas.

4.5.2 Capa de presentación

Tal y como se especificó en la arquitectura de alto nivel para la tecnología Java Enterprise Edition 7, la capa de presentación se implementara por medio de JSF con apoyo del framework de presentación Primefaces.

Se seguirá un patrón modelo-vista-controlador (MVC) tipo 3, utilizando CDI Beans para enlazar con la capa de negocio, elemento de respaldo e implementar las decisiones de navegación a través de las vistas.

La elección de CDI beans como backing beans en lugar de Managed Beans[7] se debe al hecho de ofrecer mejores prestaciones desde el punto de vista de la inyección de dependencias e implementación de patrones.

Se utilizará el Servlet implícito FacesServlet para gestionar las llamadas al contenedor web por parte de los clientes ligeros.

Para implementar las vistas se han utilizado Facelets, utilizando plantillas para dar mayor cohesión a las vistas, y componentes Primefaces para generar el GUI. No se

7 Léase Managed Beans de JSF

han utilizado de forma directa hojas de estilo en cascada, delegando los aspectos visuales de las vistas a los *Themes* implementados por Primefaces.

Se muestran los modelos correspondientes a la capa de presentación de los diferentes componentes. El componente encargado del envío y recepción de información con los dispositivos no dispone de interfaz gráfica y se ejecutará en la capa de negocio como un servicio web.

El componente Authentication no tiene elementos definidos en la capa de negocio. Si bien en la arquitectura de alto nivel para la tecnología Java Enterprise Edition se mostraba un subsistema de seguridad. En la implementación del servidor de aplicaciones elegida, Wildfly 10.1, la autenticación de usuarios mediante formularios se realiza en la capa de presentación[8].

El contenedor de servelet, Undertow, recibe las llamadas por medio de la acción *j_security_check* con parámetros *j_username* y *j_password* tal y como se describe en la especificación[9]. La acción anterior delega las labores de autenticación mediante formulario en la clase FormAuthenticationMechanism quien recibe la petición inicial y tras almacenarla, devuelve la página de login. Una vez validados usuario y password, y comprobada la autorización al recurso solicitado, se recupera la petición original y se devuelve para efectuar la redirección.

8 Aunque la división es puramente lógica podríamos considerar al contenedor web como un componente de alto nivel perteneciente a la capa de presentación.

9 JSR-340 Java Servlet 3.1, 13-140

4.5.3 Interfaces capa de presentación

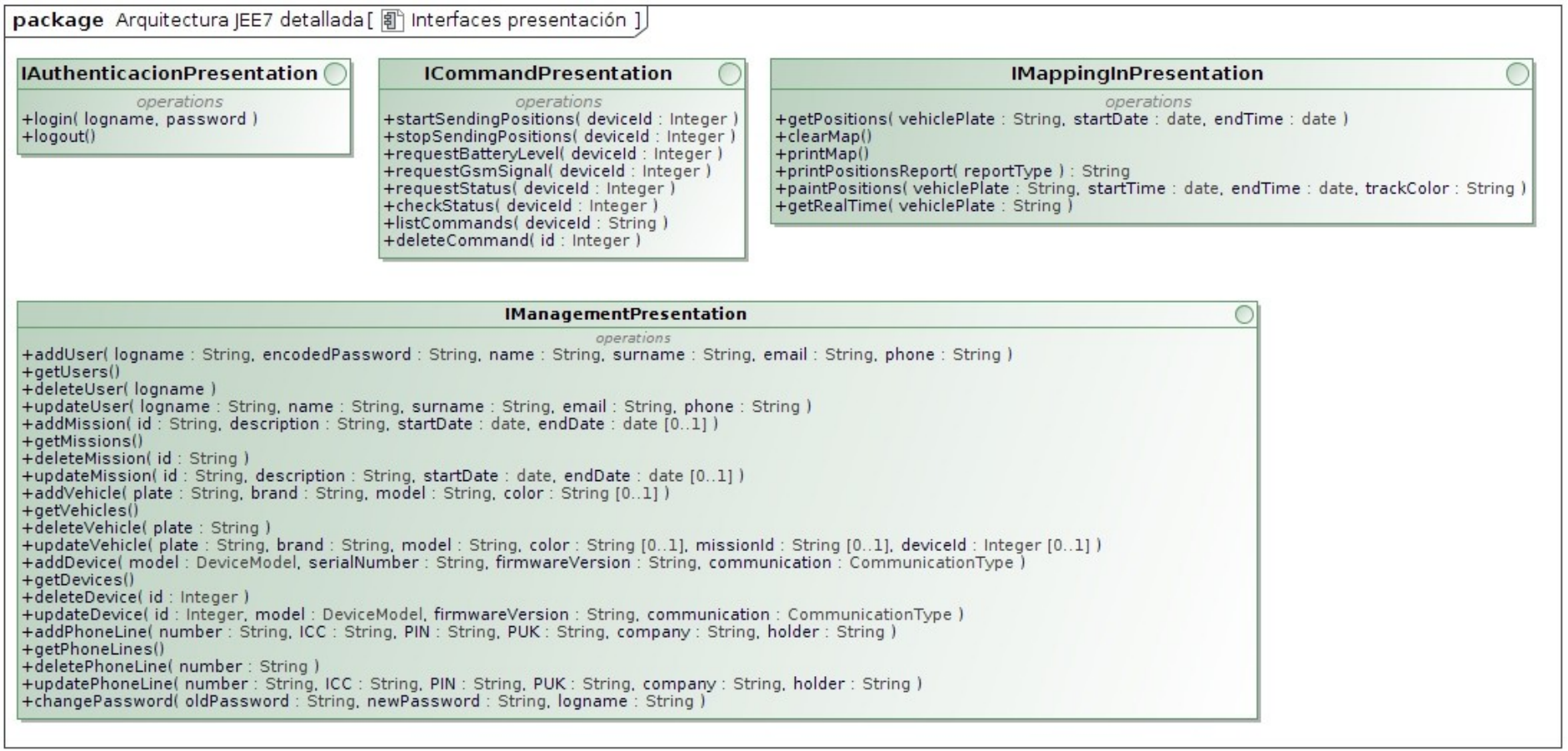


Figura 17: Interfaces de la capa de presentación

4.5.4 Modelos de la capa de presentación

Se muestran los diagramas UML más significativos de la capa de Presentación.

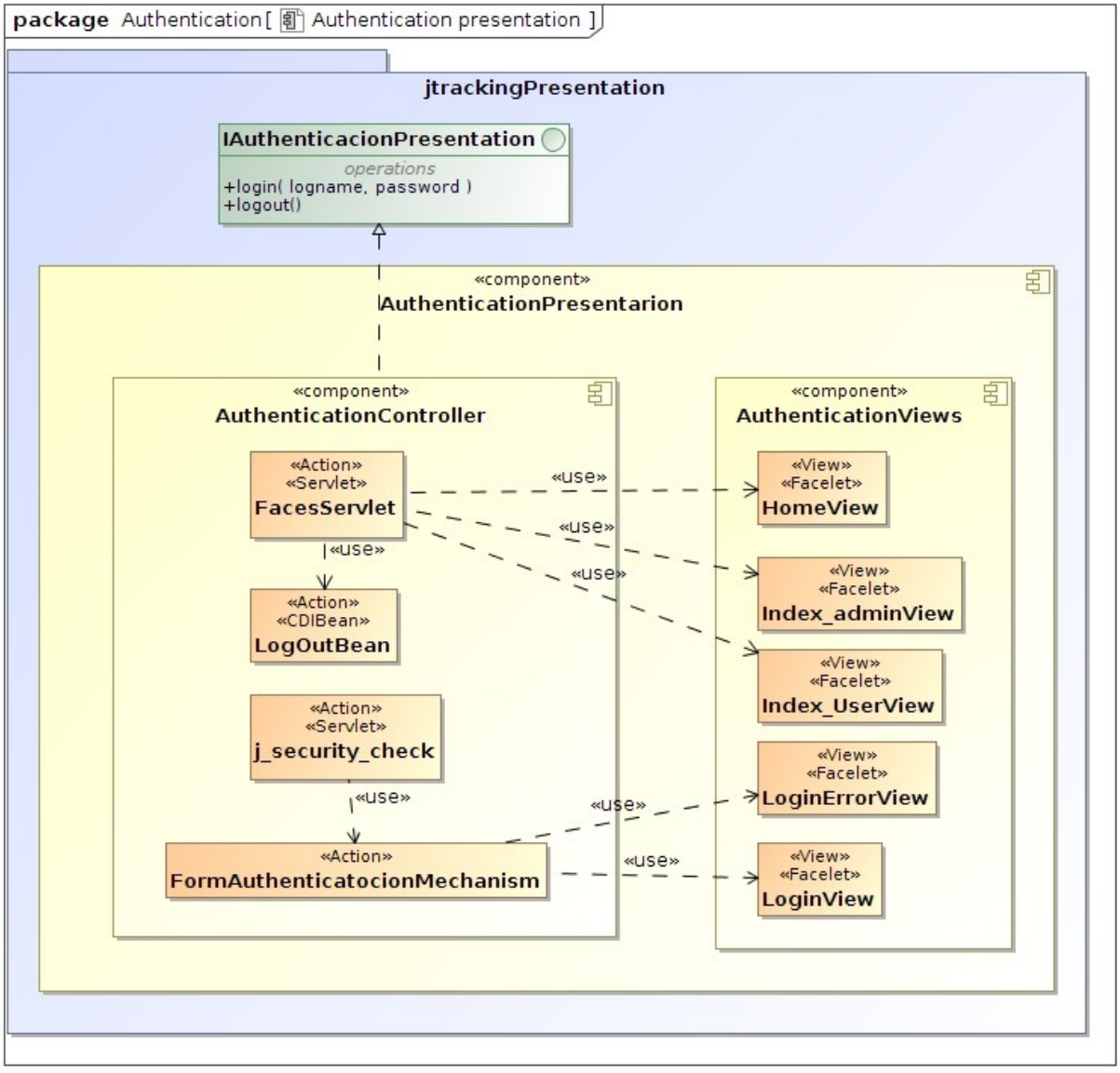


Figura 18: Autenticación y autorización

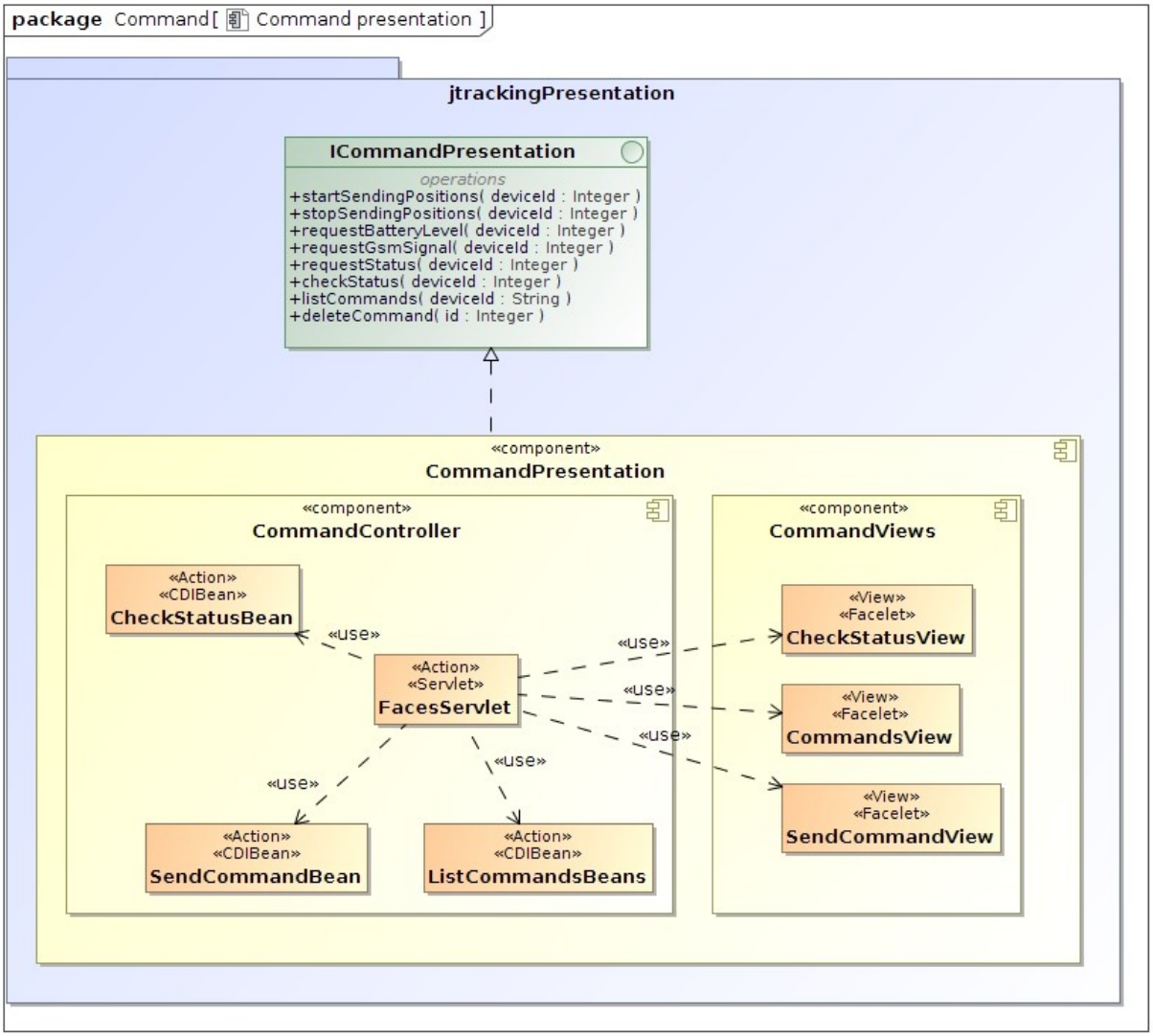


Figura 19: Comando

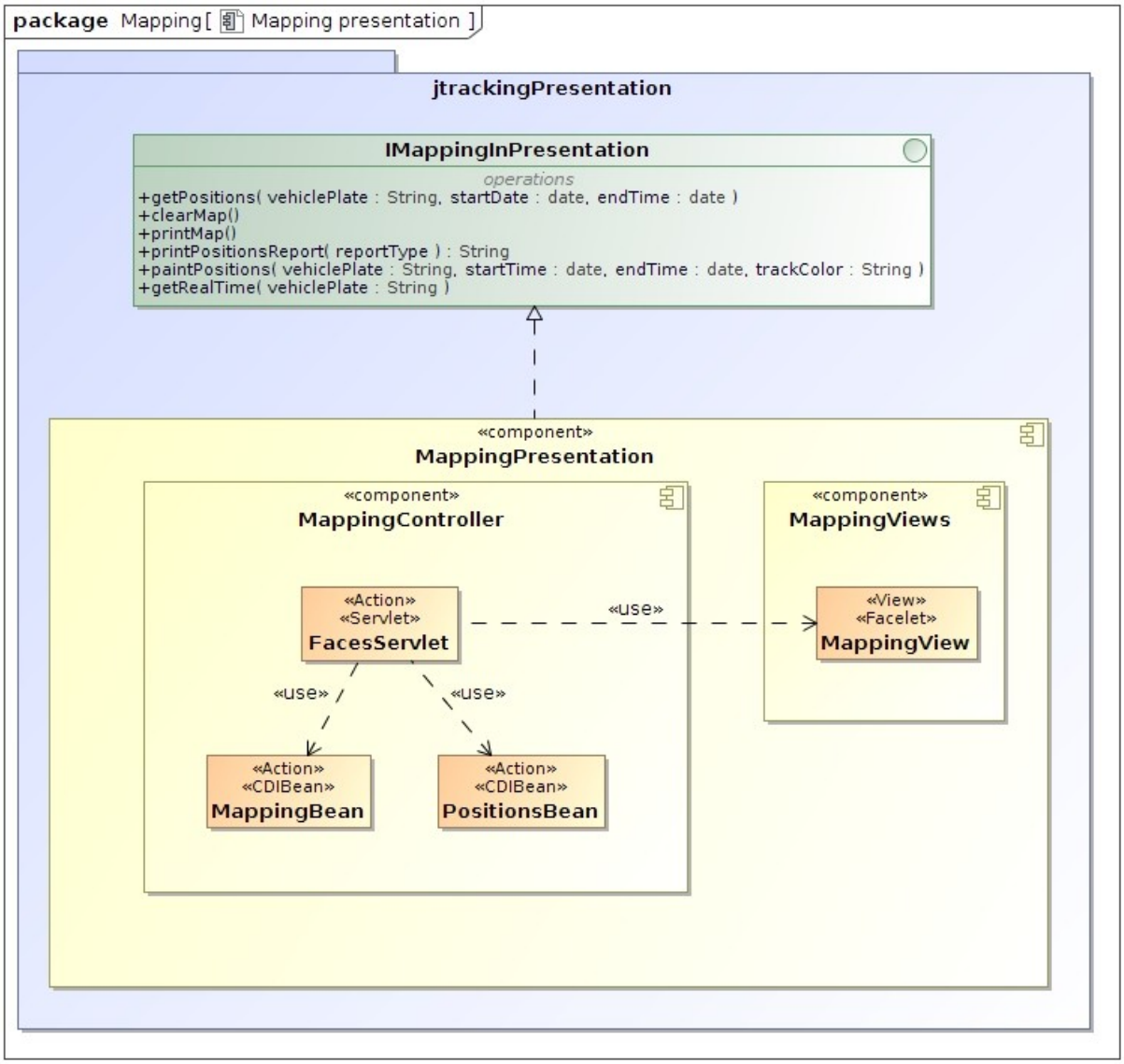


Figura 20: Mapping

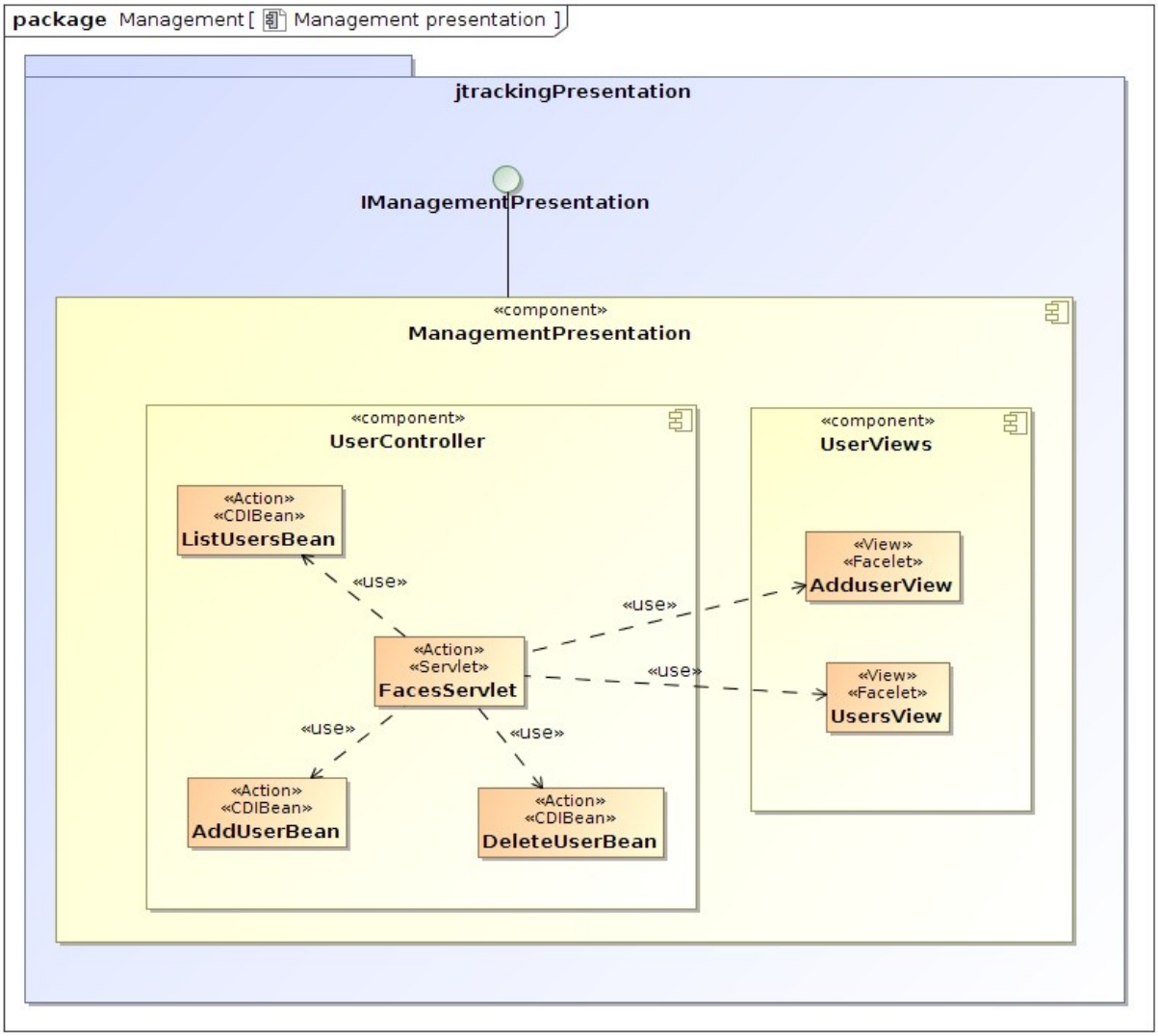


Figura 21: Usuarios

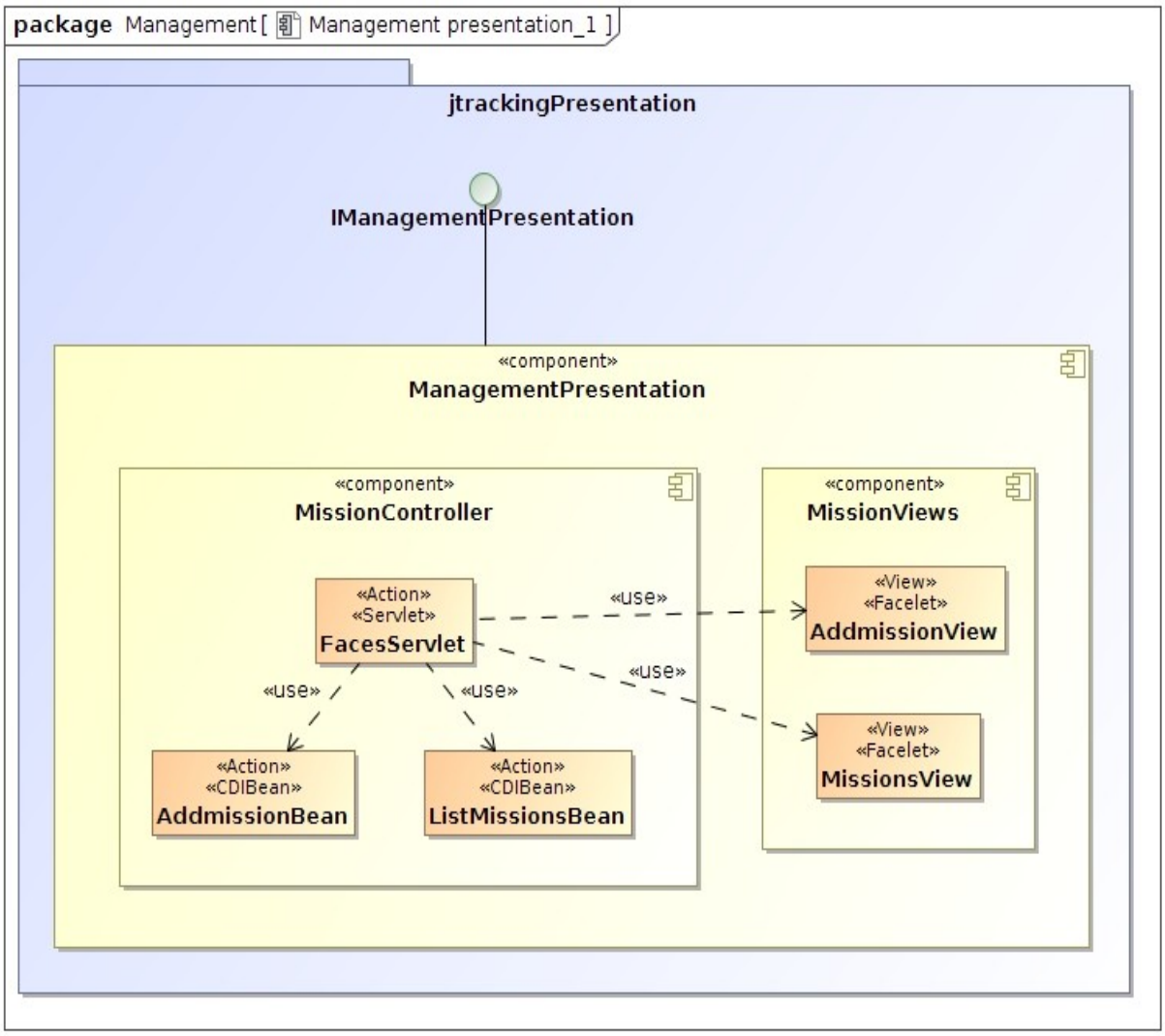


Figura 22: Misiones

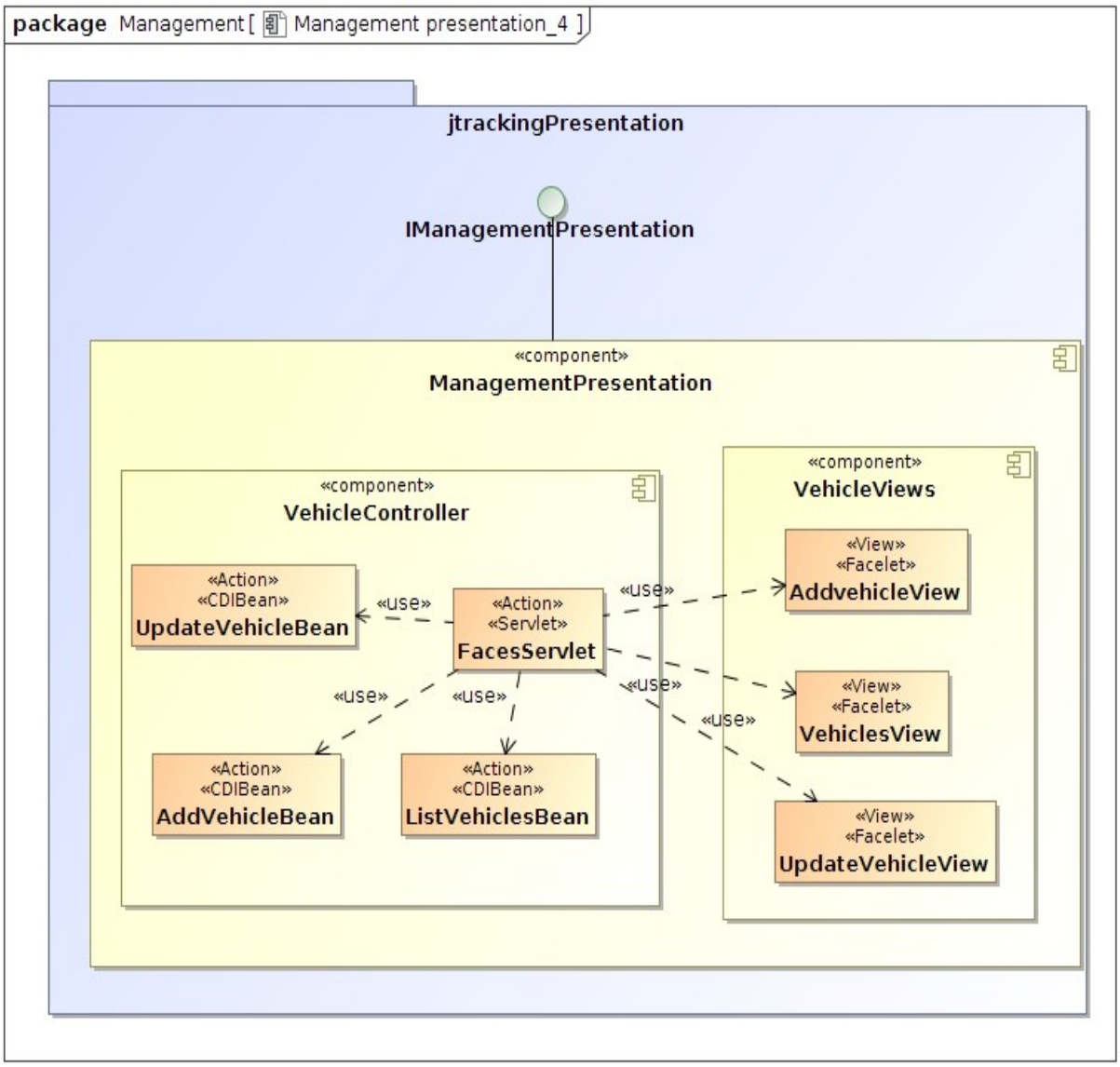


Figura 23: Vehículos

4.5.5 Modelos de la capa de negocio

La capa de negocio se ha implementado mediante Session Beans sin estado, todos los componentes implementan una interfaz local dado que todas las llamadas a sus métodos de negocio tendrán ámbito local.

Todos los métodos del componente Reception se han implementado mediante un WebService SOAP que se apoya en un Session Bean sin estado.

Figura 24: Interfaces de la capa de negocio

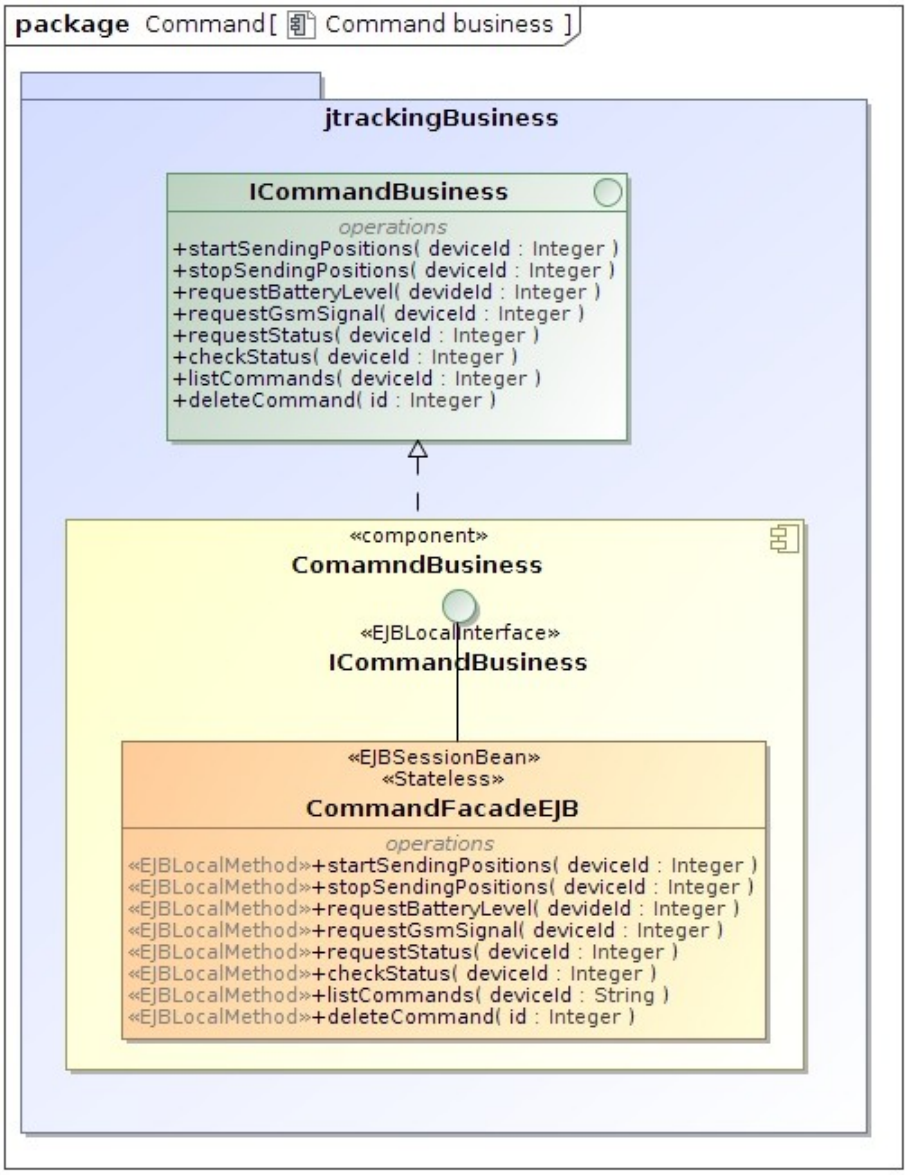


Figura 25: Comandos

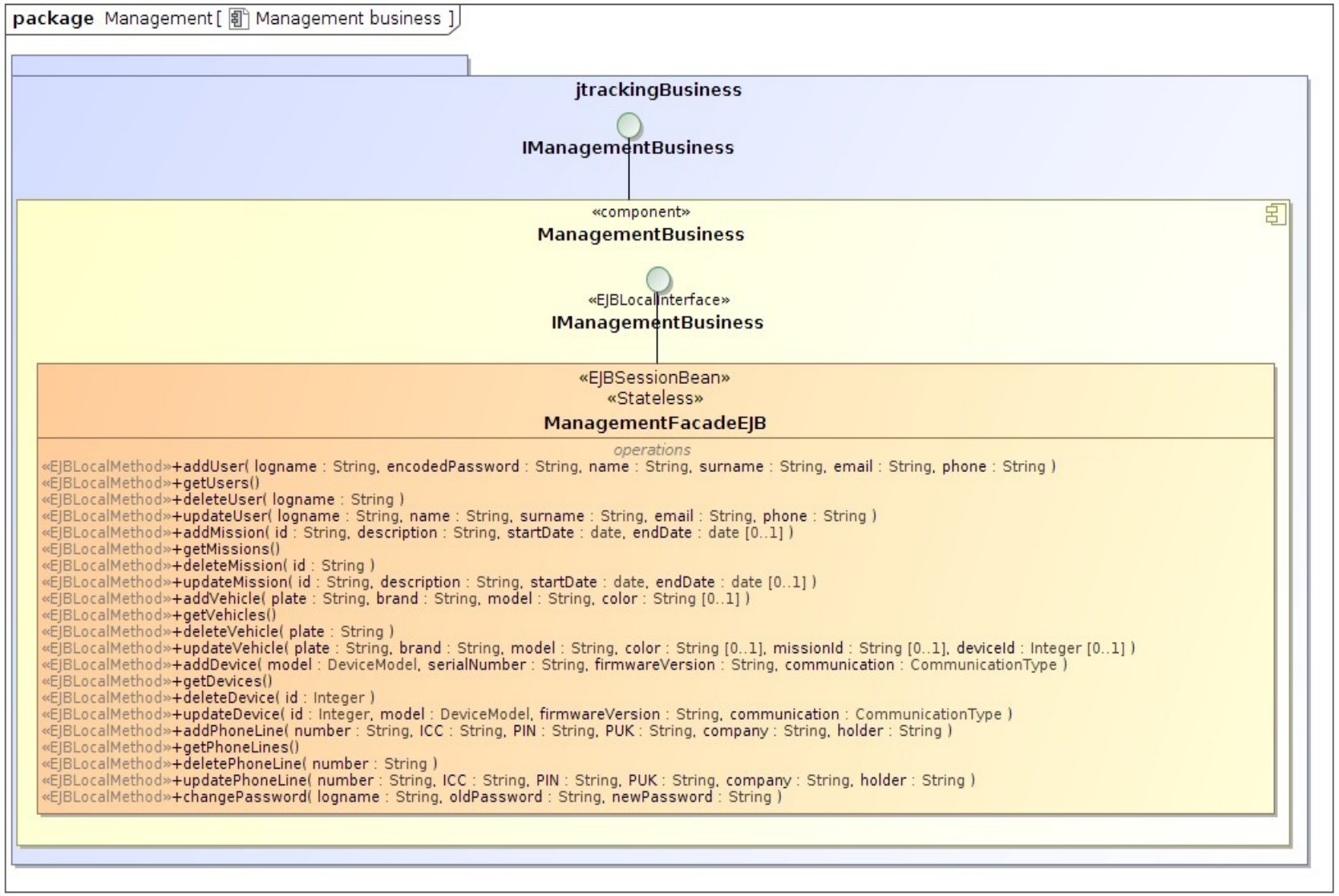


Figura 26: Administración

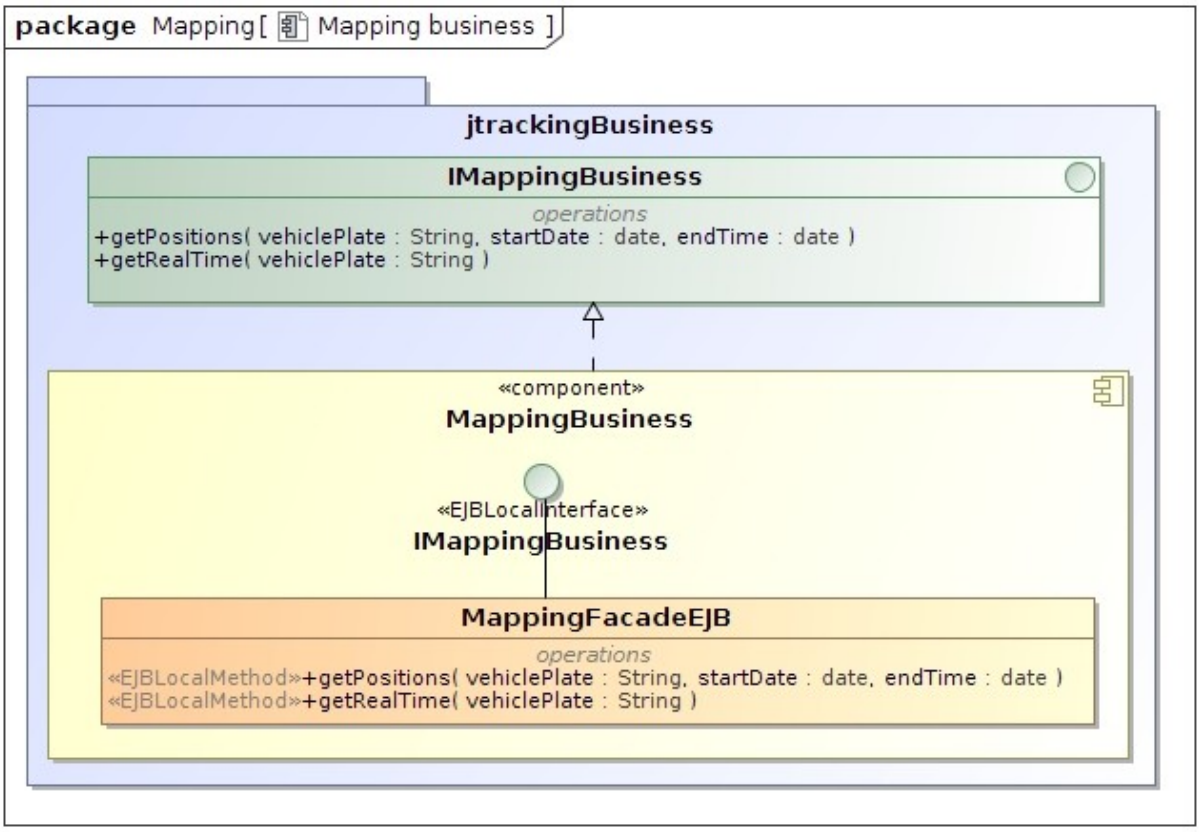


Figura 27: Mapping

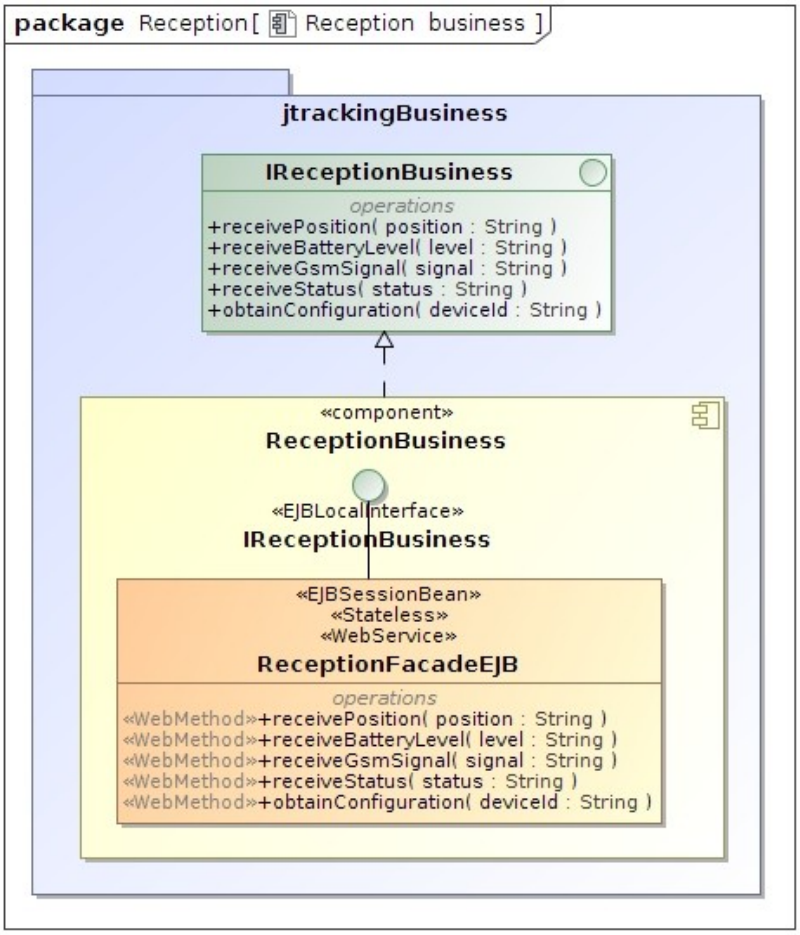


Figura 28: Recepción

4.5.6 Modelos de la capa de persistencia

La capa de persistencia da apoyo, entre otros, a los casos de uso relacionados con los usuarios del sistema, concretamente a las operaciones CRUD y de acceso.

El modelo de seguridad utilizado *Form based authentication* supone el uso de una base de datos relacional para la autenticación y autorización de usuarios. El servidor de aplicaciones Wildfly sigue esta propuesta y utiliza dos tablas, una para usuarios y otra para roles.

La propuesta original de Wildfly ofrece una tabla de roles sin clave primaria. Java Persistence API impone que toda entidad declarada *@Entity* disponga de una clave primaria *@Id*[10]. Por consiguiente y habida cuenta que se ha decidido la utilización de JPA a lo largo de todo el desarrollo, se ha efectuado una modificación de la base de datos propuesta para añadir información y adaptarla al modelo objeto-relacional de JPA. La tabla de roles se ha mapeado desde la entidad RolesJPA dotándola de clave primaria.

Siguiendo el modelo propuesto por Wildfly se ha utilizado una función resumen para proteger las passwords almacenadas en la base de datos. Se utilizará el algoritmo SHA-256 para el cálculo de la función resumen, y BASE64 para la codificación del hash resultante.

La configuración declarativa se realiza en el fichero standalone.xml junto con web.xml y jboss-web.xml. La utilización de este último descriptor de despliegue hace que esta parte de la implementación no se totalmente portable al ser un fichero específico de Wildfly.

El contenedor web, durante el proceso de autenticación y autorización, no utilizará la interfaz EntiyManager para el acceso a la base de datos; se utilizara el *DataSource* definido en el servidor y las consultas establecidas en el fichero standalone.xml según la propuesta de Wildfly, convenientemente modificadas para adaptarlas al esquema y

10 JSR-338 Java Persistence Specification, pg 30

tablas implementadas.

```
−<security-domain name="secureDomain" cache-type="default">
  −<authentication>
    −<login-module code="Database" flag="required">
        <module-option name="dsJndiName" value="java:/PostgresDS"/>
        <module-option name="principalsQuery" value="select password from jtracking.Users where logname=?"/>
        <module-option name="rolesQuery" value="select role_fk, 'Roles' from jtracking.Users where logname=?"/>
        <module-option name="hashAlgorithm" value="SHA-256"/>
        <module-option name="hashEncoding" value="base64"/>
      </login-module>
    </authentication>
  </security-domain>
```

El componente Management, en la capa de persistencia, hará uso de la interfaz EntityManager para el acceso a las entidades UserJPA y RoleJPA. Esta implementación permite que la totalidad de los accesos a datos realizados desde la capa de negocio utilicen un mecanismo uniforme de acceso, con independencia de que las mismas entidades sean accedidas por medio de otros mecanismos durante el proceso de autenticación y autorización de usuarios.

Las llamadas a la capa de persistencia se concentran en los componente de negocio que actúan como fachada de las operaciones de su interfaz. Se reutilizarán las operaciones de los componentes de negocio para cualquier llamada desde la capa de presentación, evitando de esta forma dispersar y repetir código para realizar llamadas a la capa de presentación.

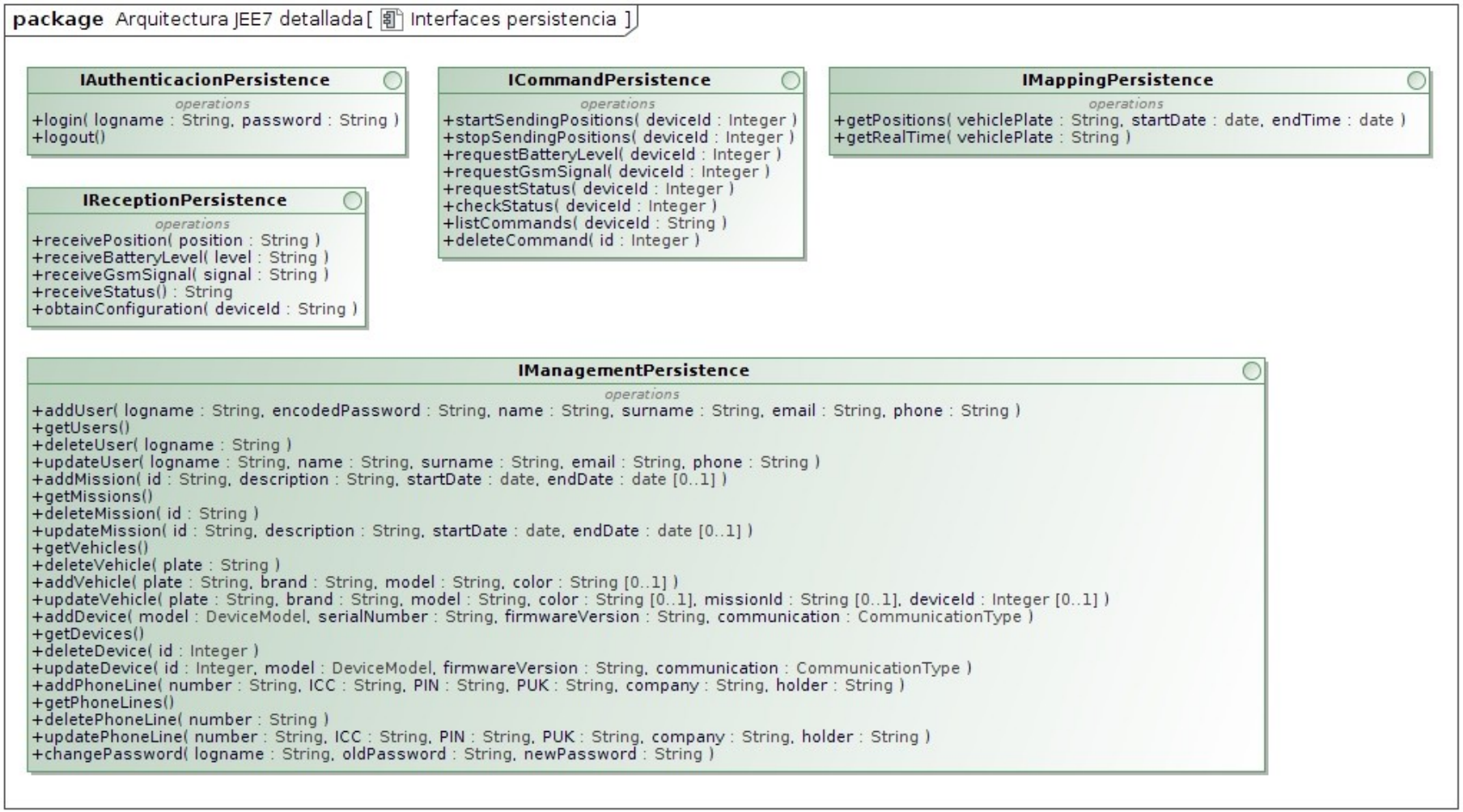


Figura 29: Interfaces de la capa de persistencia

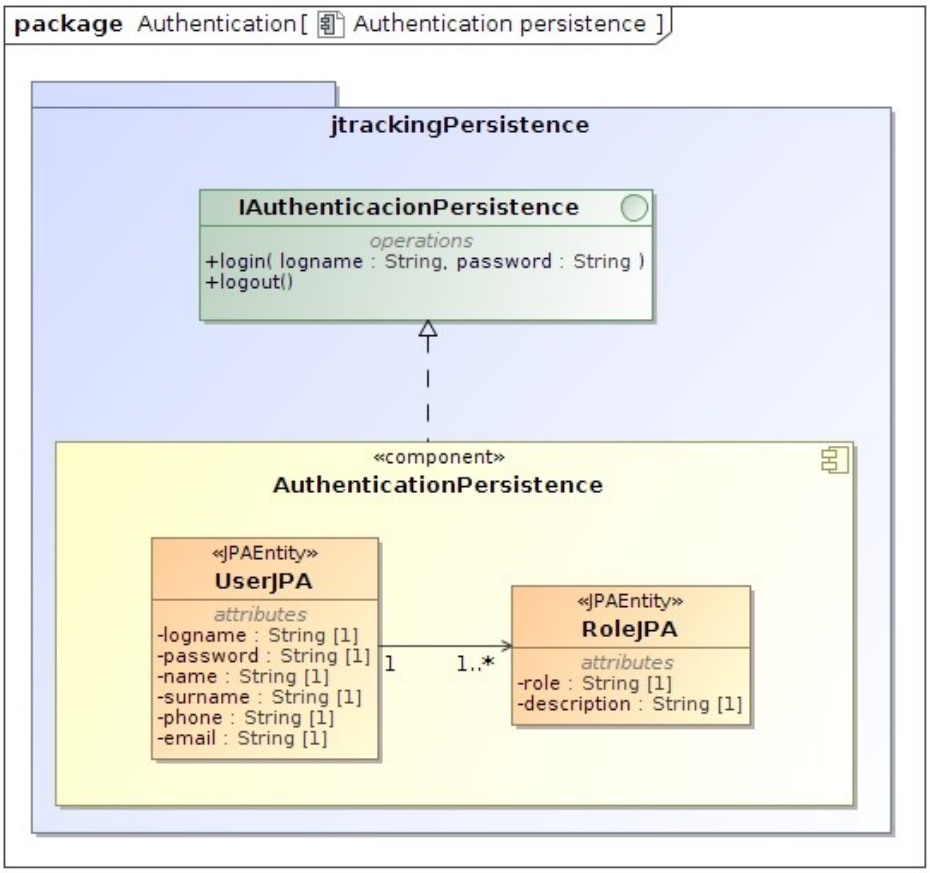


Figura 30: Autenticación y autorización

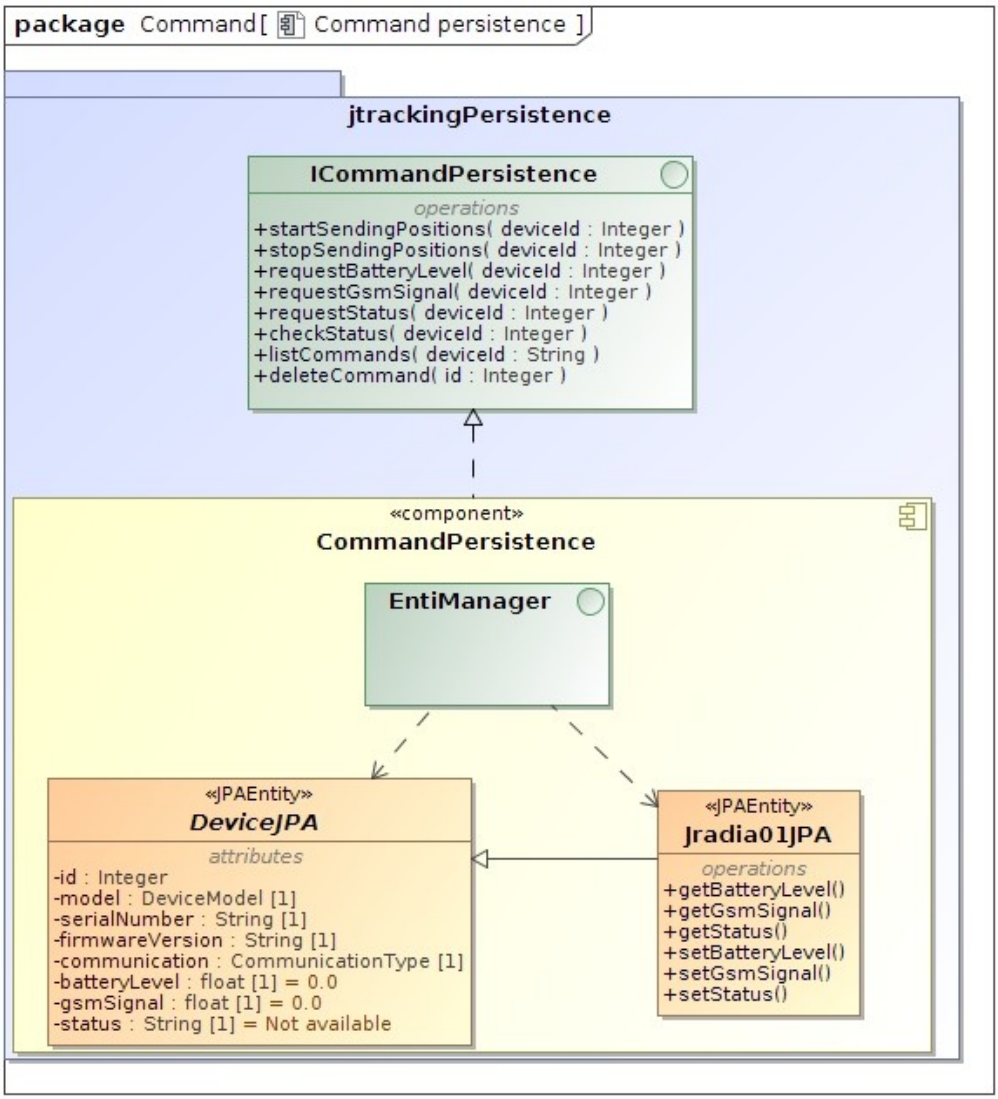


Figura 31: Comando

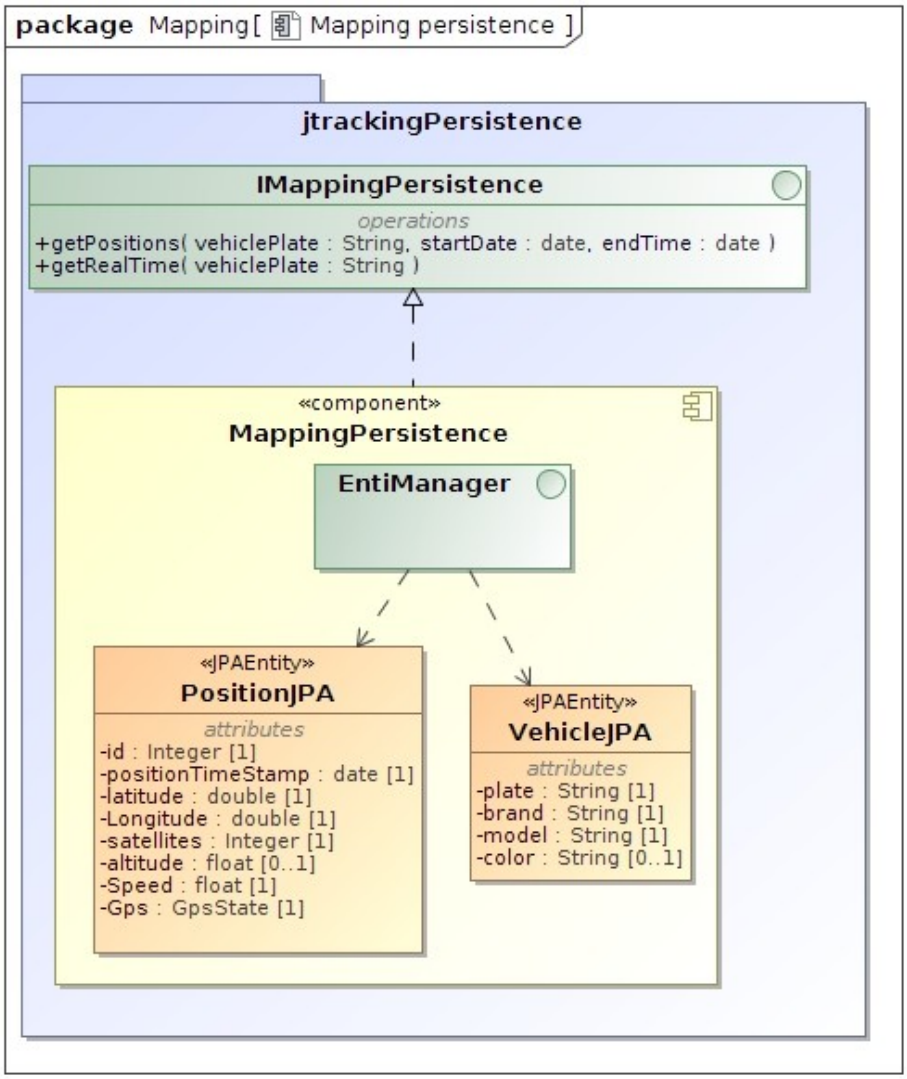


Figura 32: Mapping

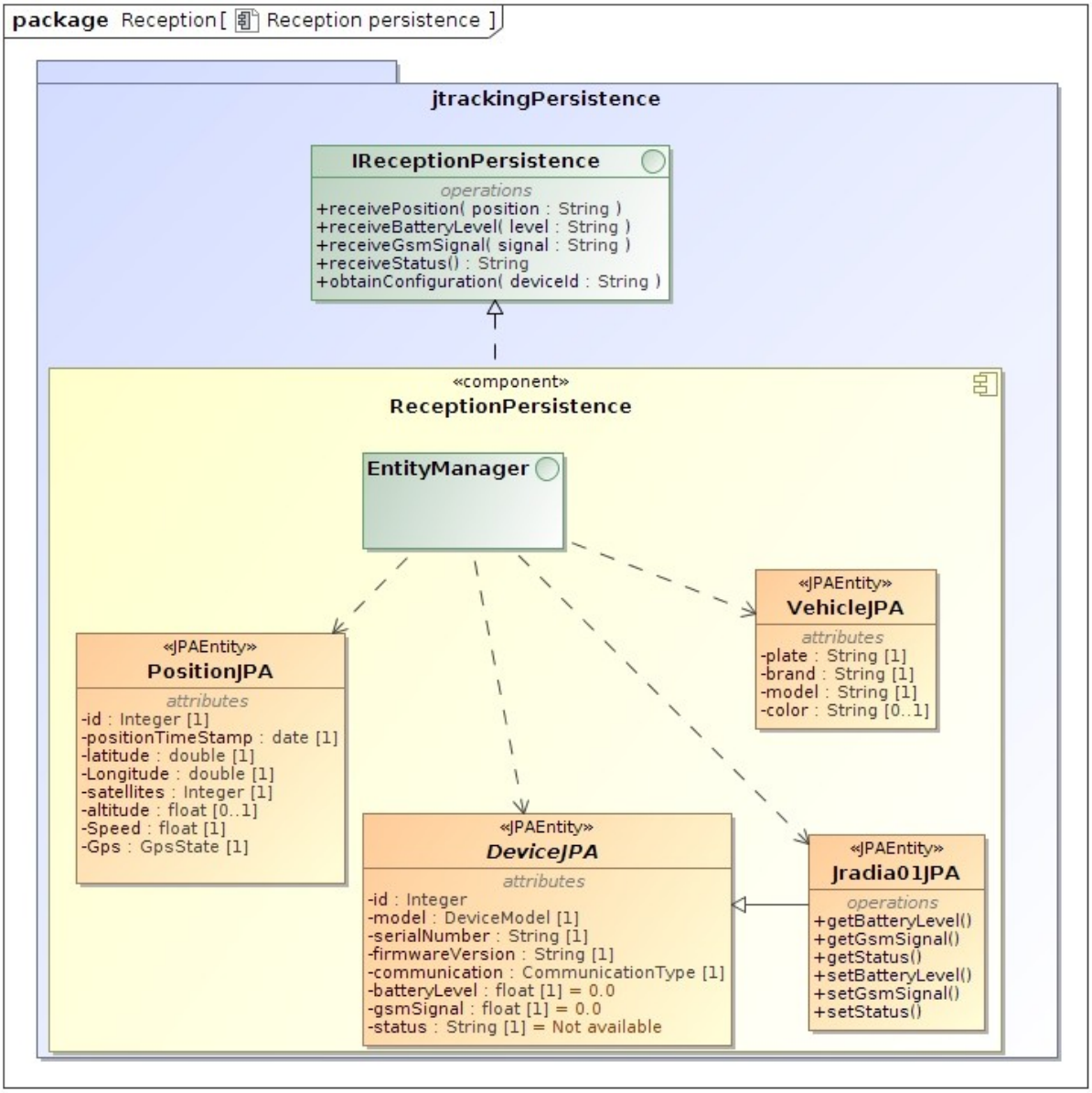


Figura 33: Recepción

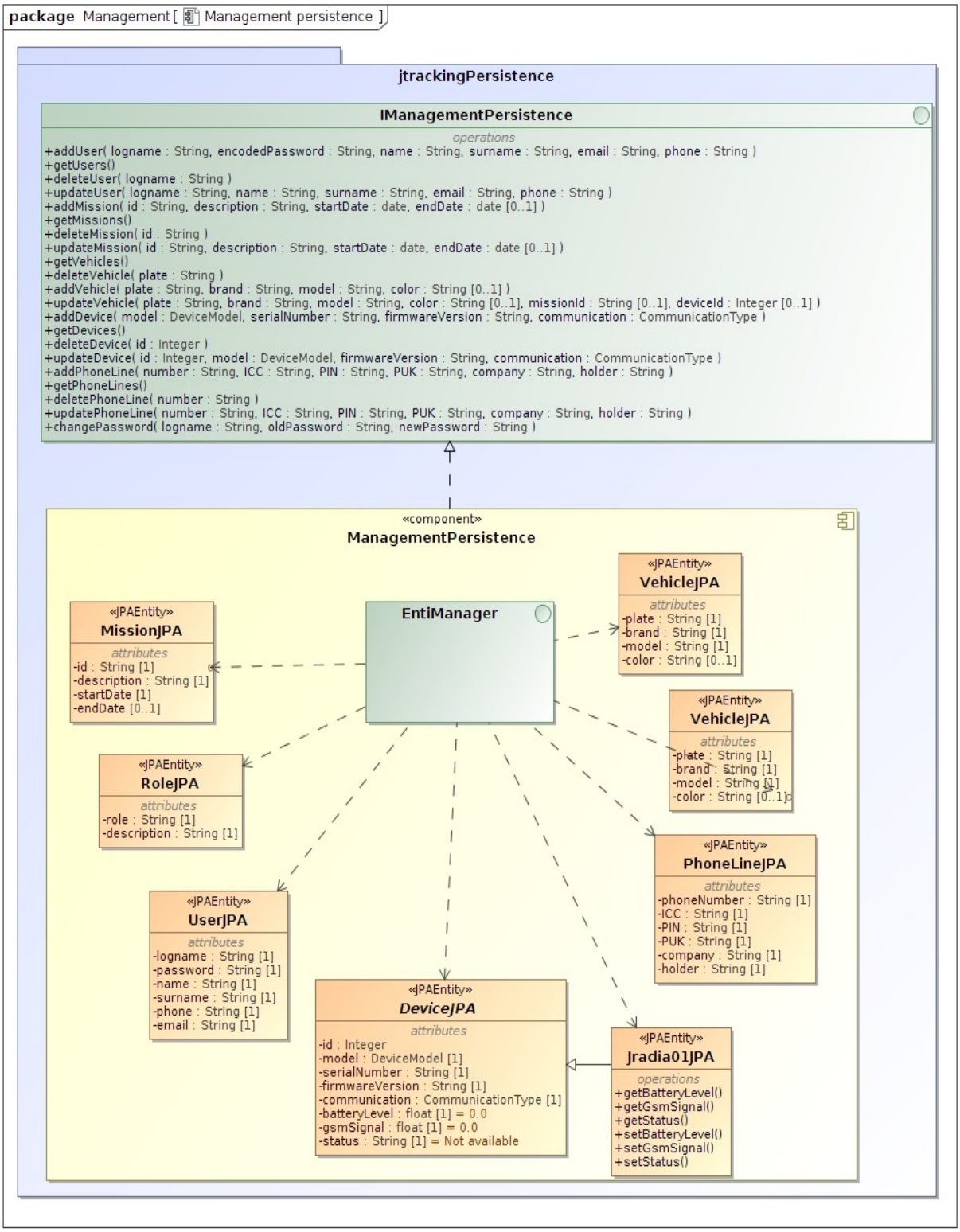


Figura 34: Administración

5 Vista de procesos

En esta vista se documentan los procesos de negocio más significativos de la aplicación. Se parte de los casos de uso descritos en los escenarios y se utilizan diagramas de secuencia o de actividades según la complejidad y detalle necesarios.

5.1 Autenticación y autorización

El caso de uso CU001 – login implementa el autorizado a los recursos del sistema.

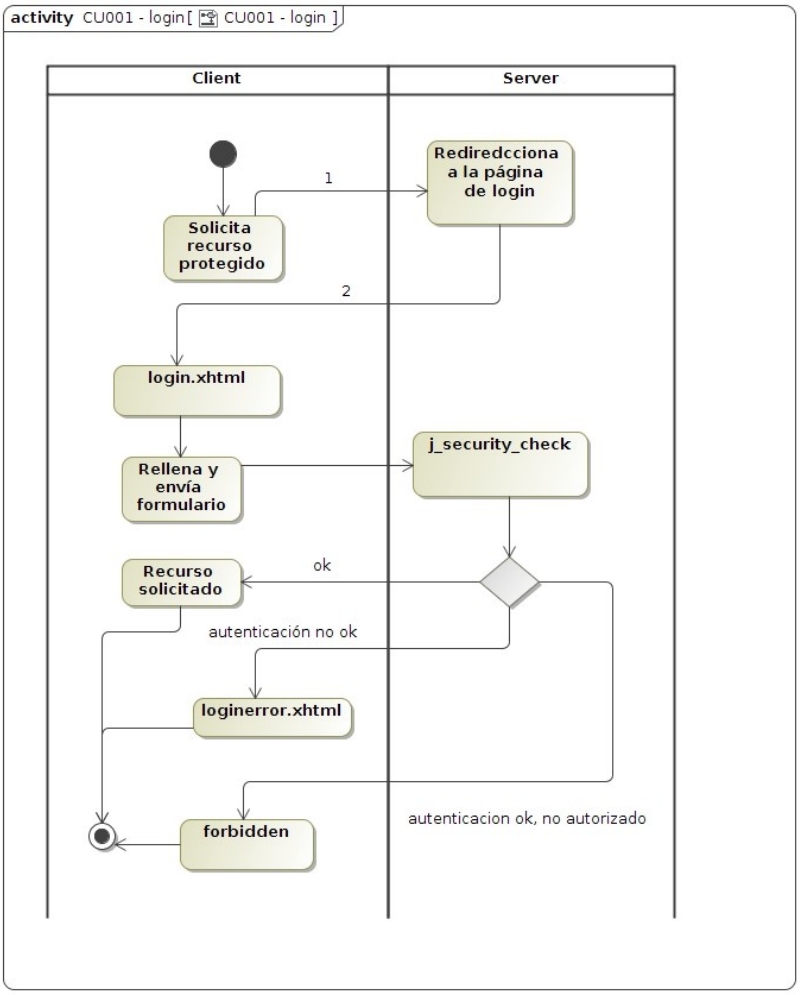
Las especificaciones[11] describen los mecanismos de autenticación y autorización de usuarios mediante *Form based authentication*. Dichos mecanismos se implementan en los contenedores web de los servidores de aplicaciones.

El intercambio de información entre servidor y cliente se realiza mediante protocolo HTTP lo que supone que tanto usuario como password se enviarán en claro, codificados en Base64. Se impone proteger la capa de transporte mediante SSL.

La configuración de seguridad de la capa de transporte se realiza generando un certificado auto-firmado o mediante la adquisición de uno en una autoridad certificadora e instalando dicho certificado en el servidor. Todas las operaciones se encuentran descritas en la guia de administración online: https://docs.jboss.org/author/display/WFLY10/Admin+Guide

En el caso de Wildfly 10.1 se sigue la especificación y se proporcionan los mecanismos para realizar dichas tareas.

11 JSR-342 Java Platform, Enterprise Edition 7 y JSR-340 Java Servlet 3.1



5.2 Gestión de entidades

En la gestión de entidades se ha dado un tratamiento diferenciado a las actualizaciones en las que no es necesario vincular ninguna otra entidad de aquellas en que si hay que vincular entidades. Ej vehículo – misión o linea telefónica – dispositivo.

En el caso más sencillo la actualización sin vinculación con otras entidades el proceso se realiza en linea dentro de la misma vista de listado de la entidad.

En el caso de que sea necesario establecer una relación se ha credo una vista de

actualización específica. Ej. updatevehicle.xhtml.

Como ejemplo de los mantenimientos se documenta las operaciones CRUD sobre un vehículo y la asignación de un vehículo a una misión.

La creación de un vehículo se realiza con los parámetros estrictamente necesarios, no se contempla la asignación del vehículo a una misión concreta en el momento de su creación.

El listado de vehículos se realiza de forma independiente de su creación de forma que la creación de un vehículo no supone la visualización del listado.

La actualización de vehículos de usuarios puede suponer el cambio de valores propios de la entidad, la asignación a una misión, la asignación de un dispositivo o una combinación de todos ellos se realza en una vista específica.

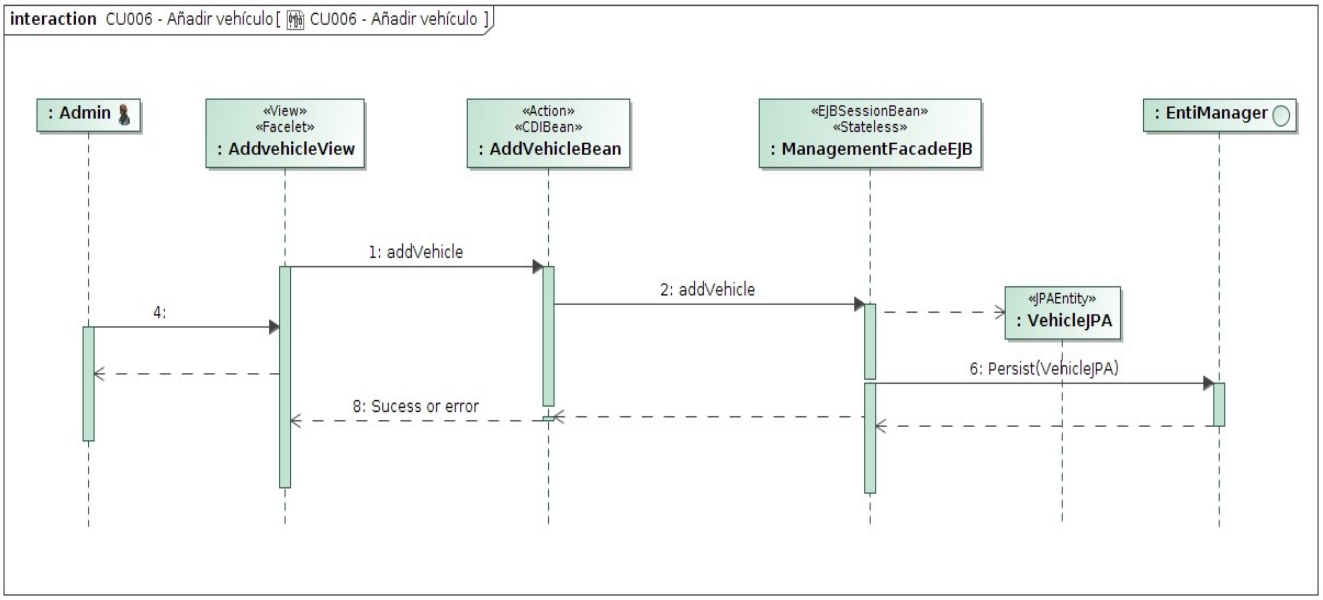


Figura 35: Añadir vehículo

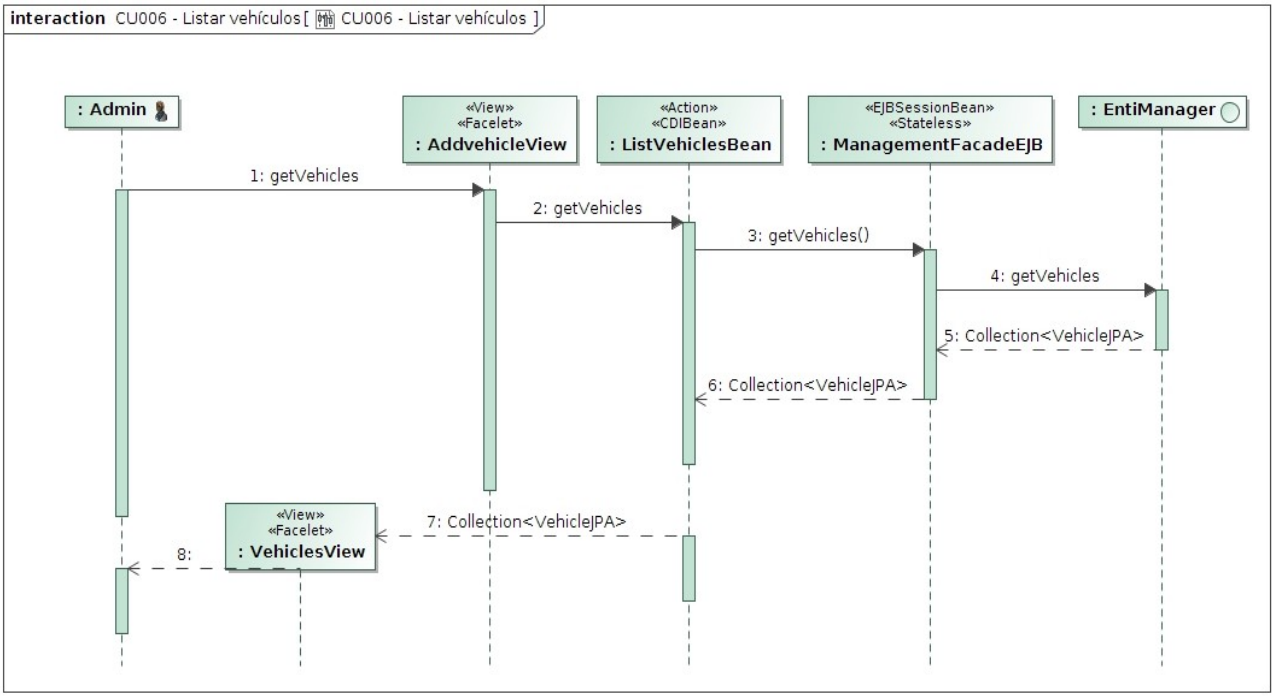


Figura 36: CU006 - Listar vehículos

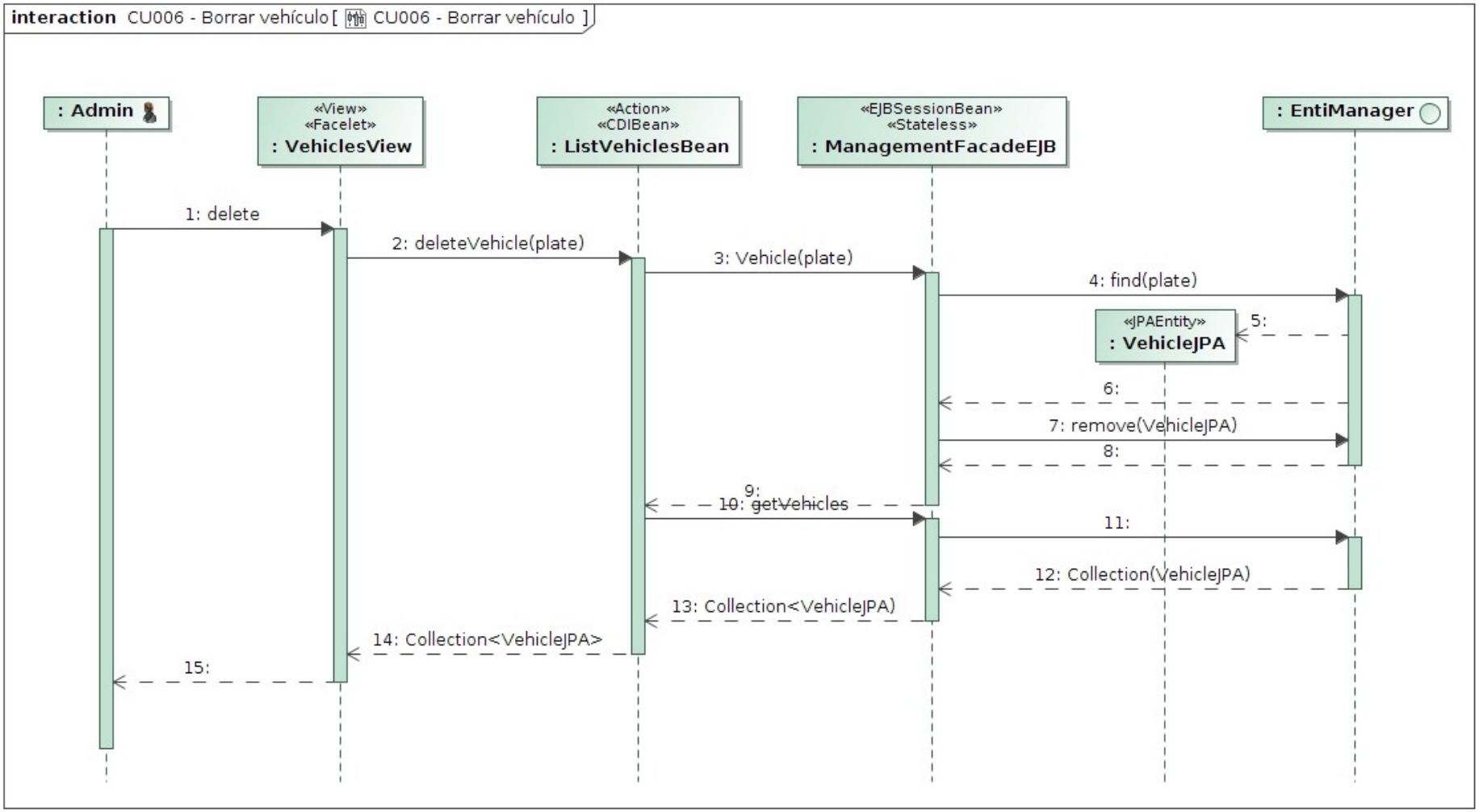


Figura 37: CU006 - Borrar vehículo

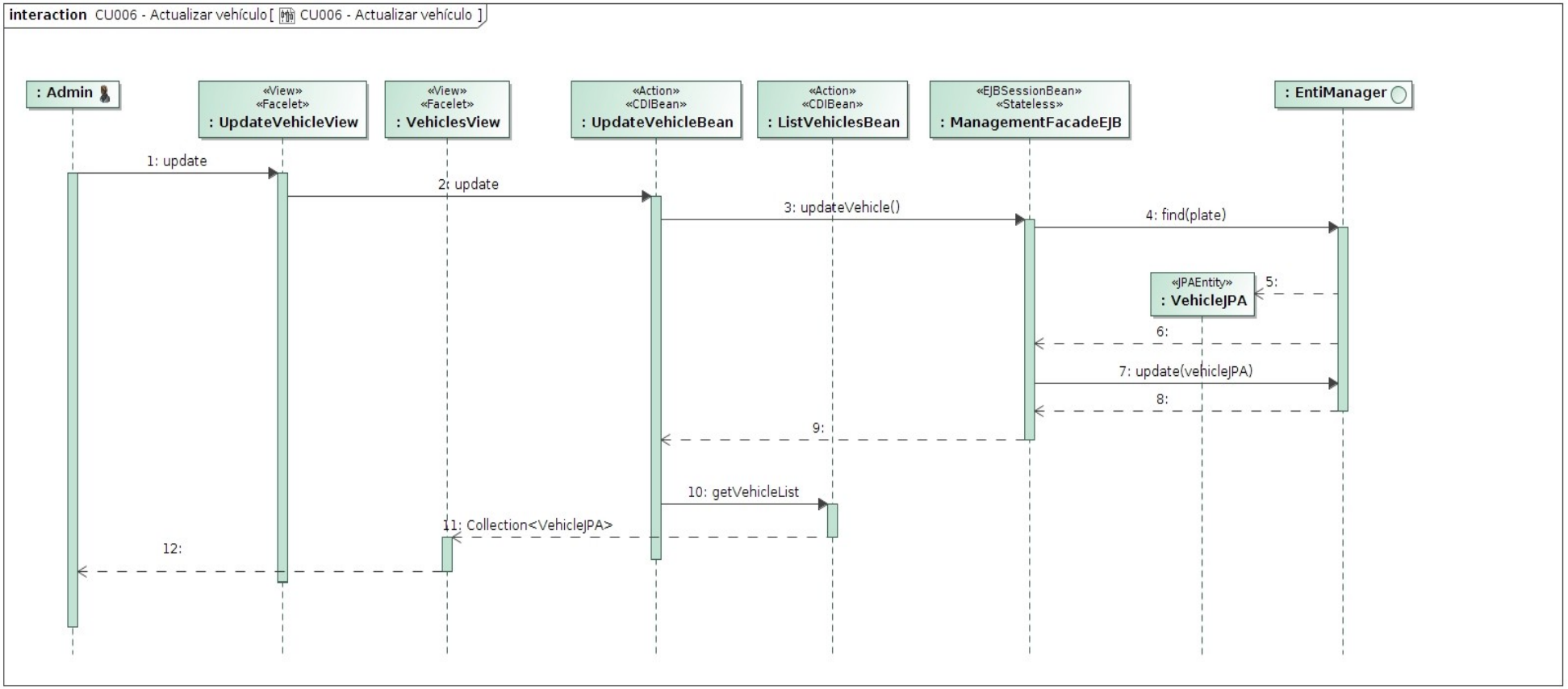


Ilustración 2: CU006 - Actualizar vehículo

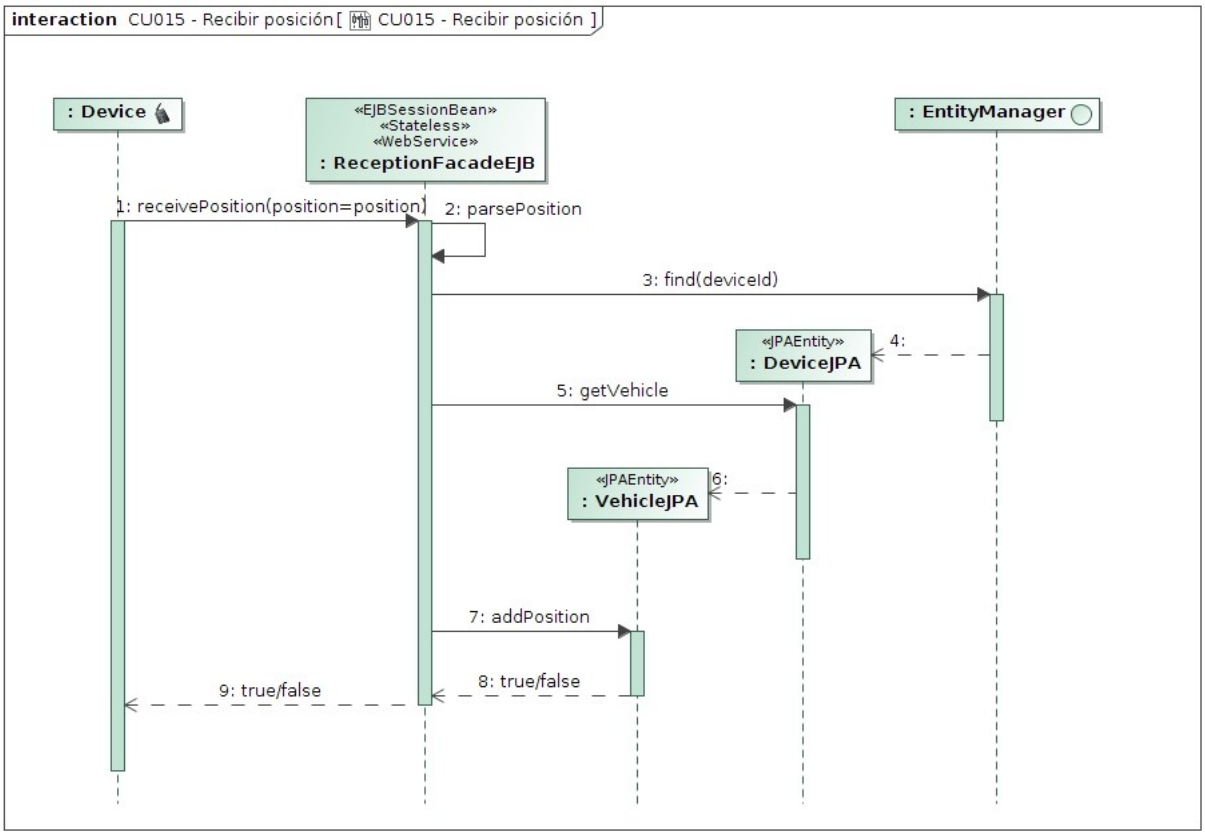


Figura 38: CU015 - Recibir posición

5.3 Tracking

La explotación de información se agrupa entorno a seis casos de uso que permiten recuperar un listado de posiciones, mostrar el recorrido de un vehículo en el mapa, obtener la información en tiempo real, etc.

Algunas de estas operaciones tienen lugar exclusivamente en la capa de presentación como pudiera ser la de limpiar el mapa, otras abarcan todas las capas de la aplicación.

Se documenta los casos de uso CU011 – Mostrar posiciones en el mapa, CU022 – Localización en tiempo real y el CU012 - Limpiar el mapa.

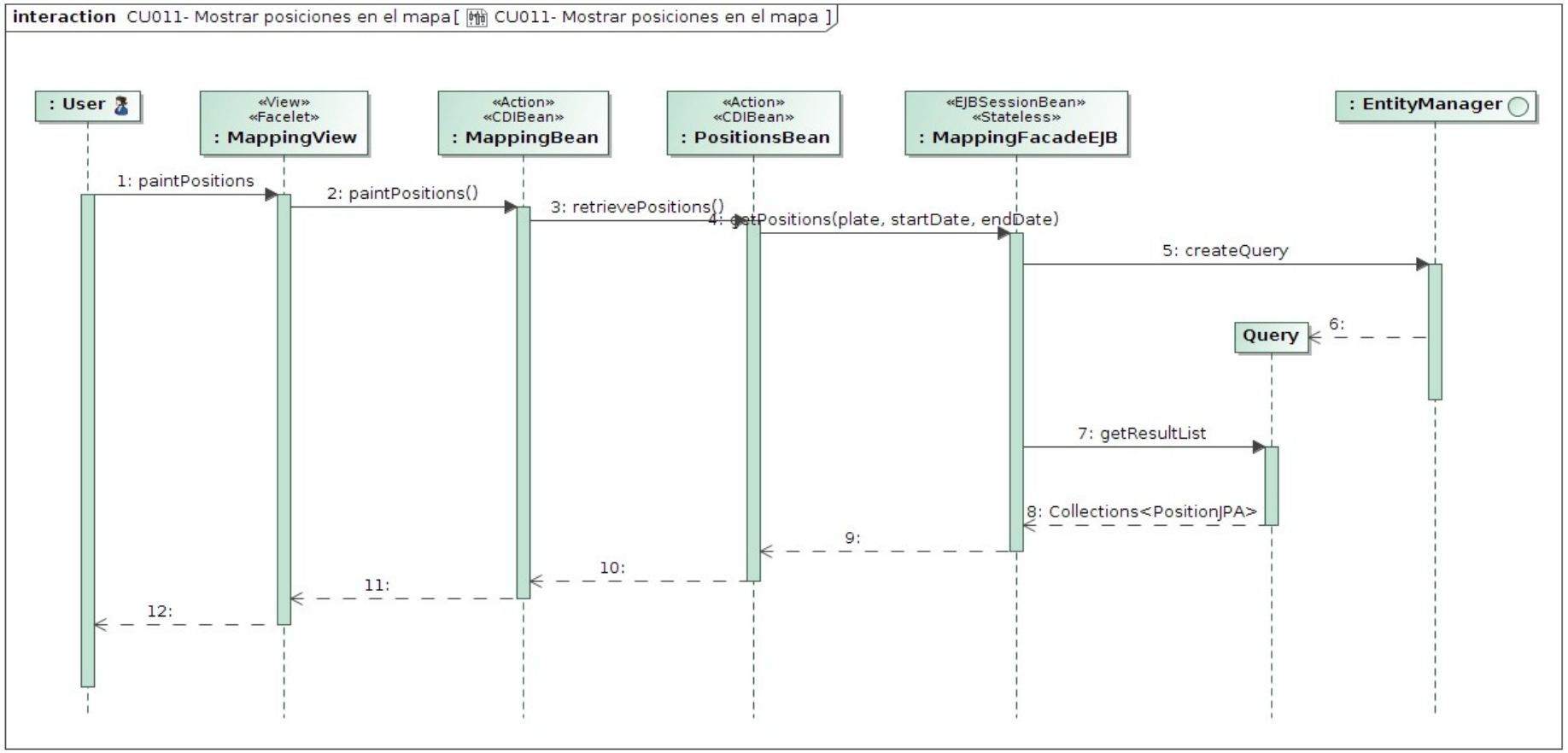


Figura 39: CU011 - Mostrar posiciones en el mapa

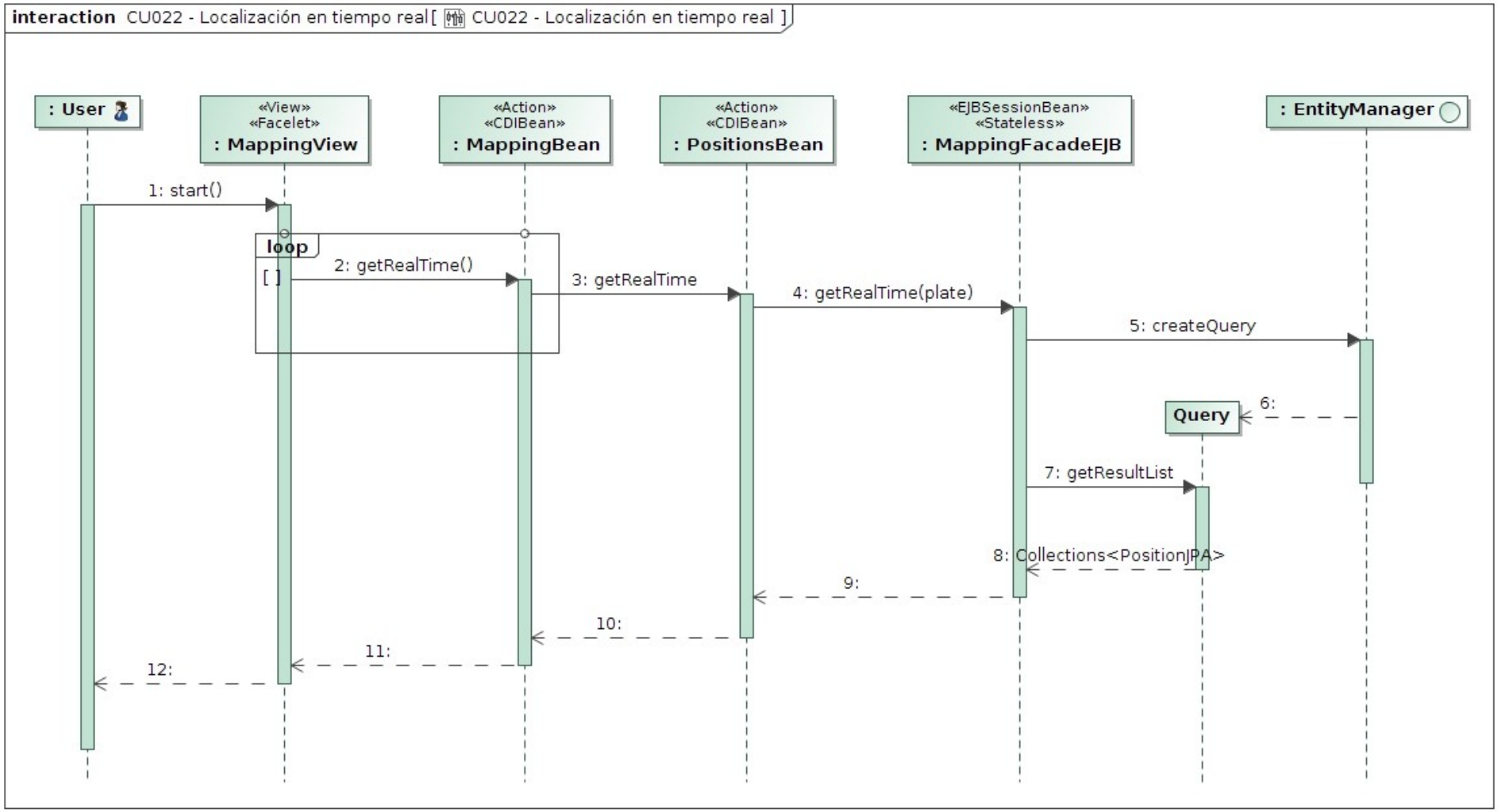


Figura 40: CU022 - Localización en tiempo real

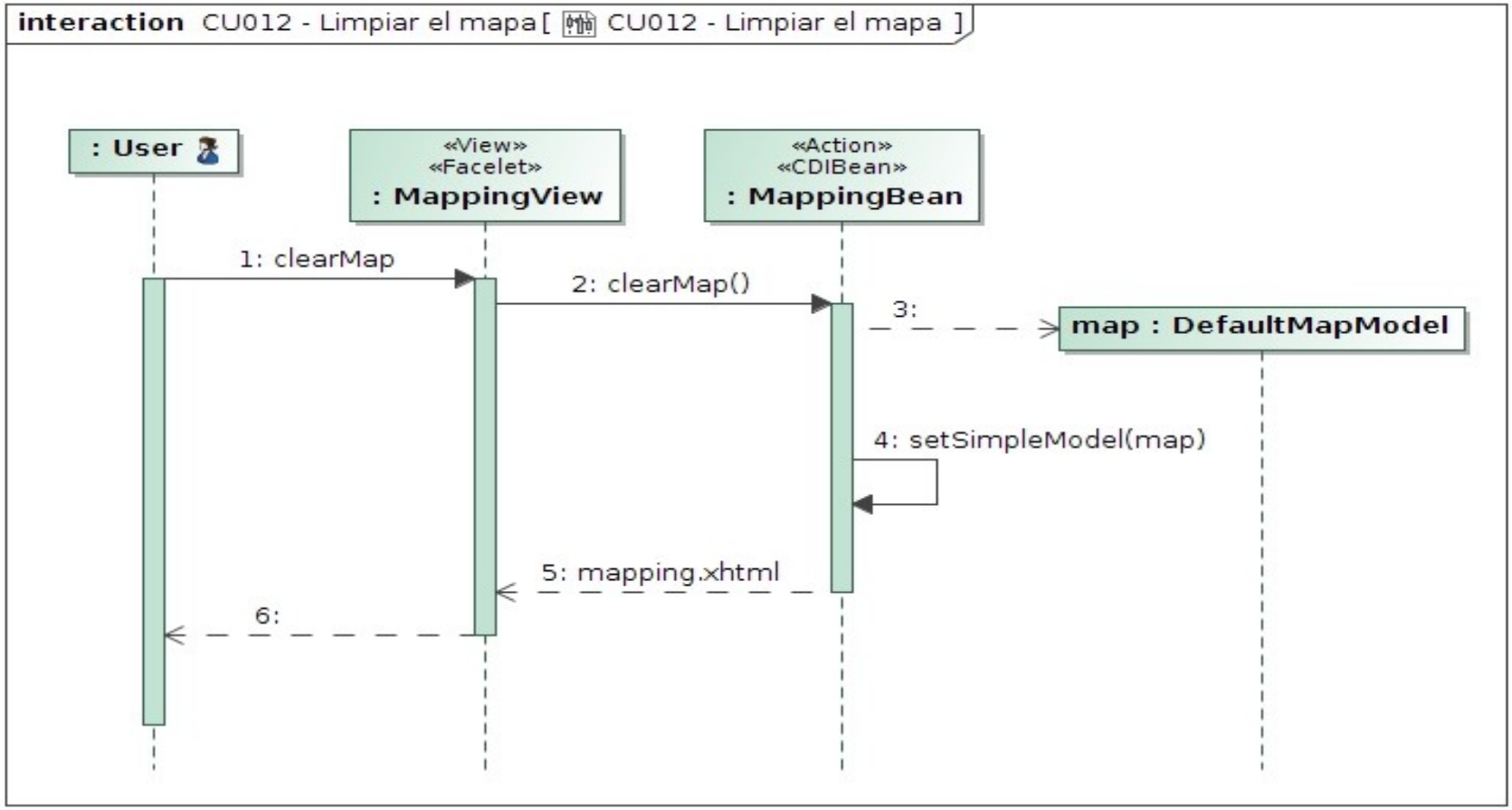


Ilustración 3: Limpiar el mapa

6 Vista física

El despliegue de la aplicación, tanto si se realiza en un hosting, en un servidor propio o en un hosting Docker supone disponer de los elementos necesarios para la ejecución de la aplicación. En este caso del servidor de aplicaciones Wildfly 10.1 y el gestor de base de datos PostgreSQL 9.5. Se deberá contar igualmente con una máquina virtual Java y el driver JDBC para la conexión con el SGDB.

Una tarea previa al despliegue es la de empaquetado de la aplicación. En este caso se ha partido de un proyecto de aplicación empresarial de Eclipse (Enterprise Application Project). Este tipo de proyecto permite referenciar otros proyectos que serán empaquetados de forma conjunta en un archivo ear. En este caso se ha optado por cuatro proyectos uno por cada capa y un cliente para la realización de simulaciones y pruebas. Como puede observarse los elementos compartidos entre capas se empaquetan en un directorio lib. En este caso, este directorio contiene el paquete de la capa de persistencia, por lo que las entidades JPA quedan disponibles para toda la aplicación.

Los artefactos generados tras el empaquetado de la aplicación se muestran en el siguiente diagrama:

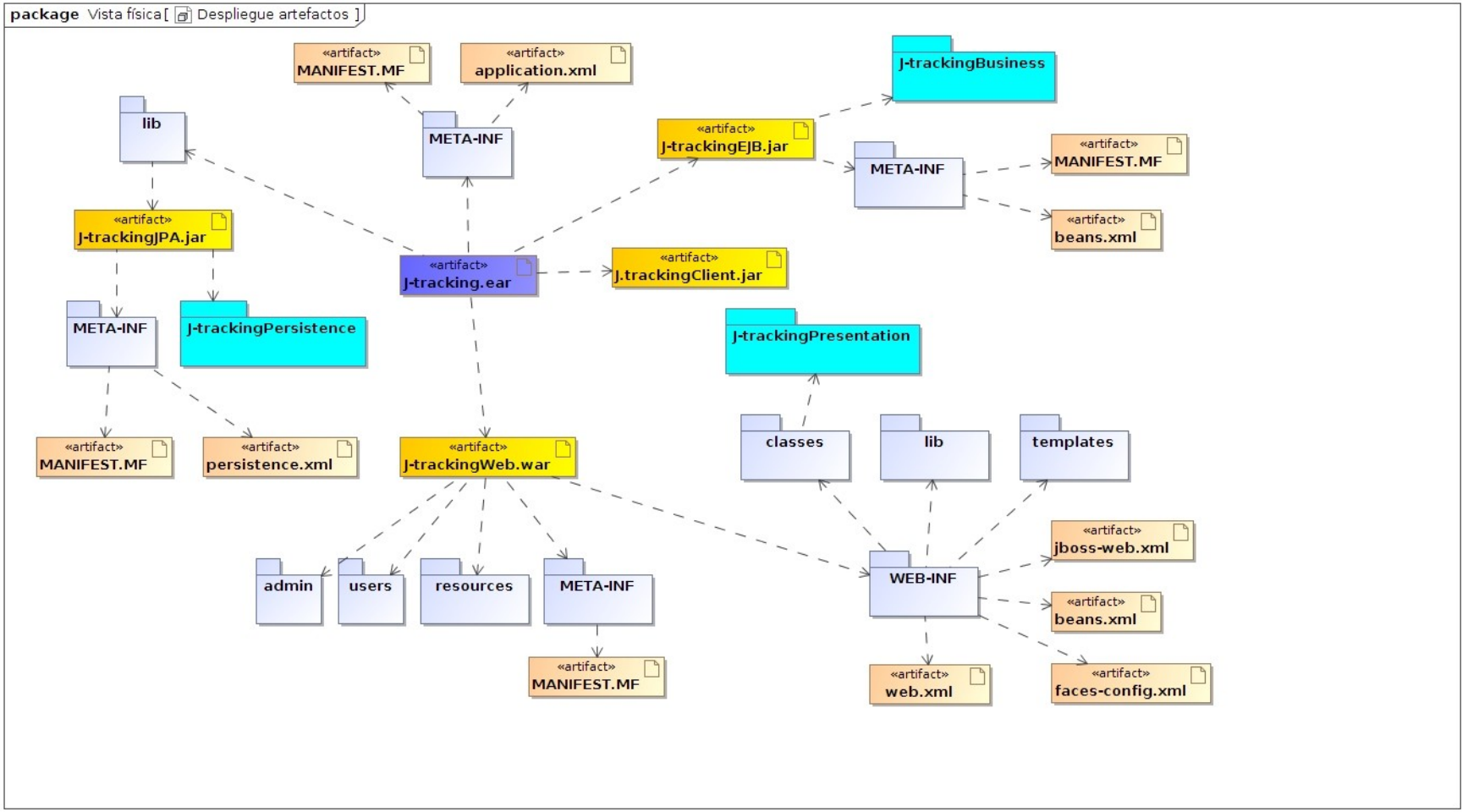


Figura 41: Empaquetado y artefactos

Una vez obtenido el fichero J-Tracking.ear su despliegue se realiza sobre el servidor de aplicaciones Wildfly 10.1. En el siguiente diagrama se muestra la distribución de la aplicación una vez desplegada.

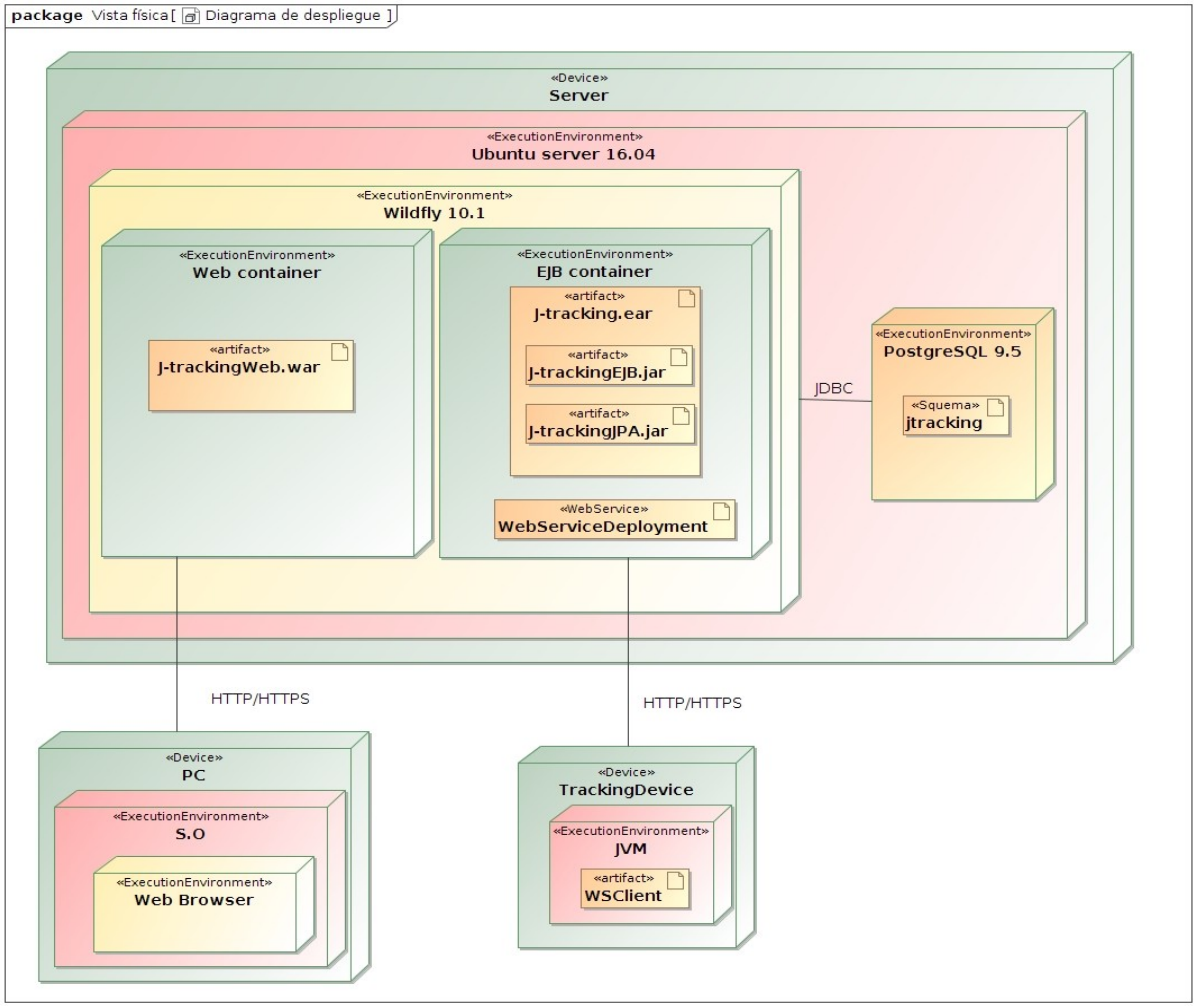


Figura 42: Diagrama de despliegue

El mapeo objeto relacional genera de forma automática las entidades JPA, las secuencia para las claves primarias auto-numéricas y los índices creados en el código de las entidades.

7 Evolución y mejoras del producto

Finalizada la implementación del proyecto se considera que se deberían incluir nuevas funcionalidades y mejoras que permitieran disponer de un producto más versátil y completo.

7.1 Interfaz único

Inclusión de nuevas operaciones y comandos para los dispositivos como la regulación del intervalo de envío de posiciones o la solicitud del número de satélites visibles y utilizados para el cálculo de las posiciones.

7.2 Aplicación

En cuanto al aplicativo se considera que podrían incluirse requisitos como:

- Visualización de varios recorridos del mismo o diferentes vehículos
- Obtención de la última posición conocida de todos los vehículos de una misión o todos los vehículos del aplicativo.
- Borrado del listado de posiciones.
- Selección del tema de Primefaces preferido para cada usuario.
- Se deberá explorar la utilización de tecnologías como Java Message Service para la implementación de algunas funcionalidades de la capa de negocio.
- Envío de información desde los dispositivos de forma asíncrona.
- Inclusión del manual de usuario en linea.

7.3 Pruebas

- Incorporación de un framework de pruebas automatizadas que permita la

realización de pruebas de integración de forma ágil.

- Realización de pruebas de carga.
- Elaboración de métricas del código del aplicativo

7.4 Seguridad

- Se deberán incorporar mecanismos para asegurar la protección de la información que envían los dispositivos de control de flotas. Se deberá decidir que tipo de seguridad se implementa en función de las capacidades de los dispositivos.
- Se deberá sustituir el certificado autofirmado de Wildfly para asegurar la capa de transporte mediante SSL por uno expedido por una autoridad certificadora.

8 Conclusiones

La realización del Trabajo de Fin de Grado supone la culminación de los estudios del Grado en Ingeniería Informática. En consecuencia tanto los resultados del mismo, como la metodología y enfoque empleado o las tecnologías seleccionadas para la obtención del producto deben avalar el grado de madurez obtenido.

La realización del proyecto supone llevar a la práctica los conocimientos adquiridos a lo largo de los años, permitiendo que el alumno seleccione la temática, metodología y producto final con total libertad, dentro de un abanico amplio.

En este caso, la temática elegida, Java Enterprise Edition, ha permitido abordar un problema real, *el control de flotas de vehículos con dispositivos heterogéneos por medio de un software distribuido e interfaz único*, *que permite independizar el software de control y explotación de la tecnología de comunicaciones y dispositivos utilizados*. El desarrollo del proyecto ha permitido llevar a la práctica la práctica totalidad de las asignaturas del itinerario de Ingeniería del Software. Se ha

profundizado especialmente en las disciplinas relacionadas:

- Diseño de base de datos: alzando conocimiento suficientes para abordar el diseño de una base de datos objeto-relacional utilizando un framework de persistencia.
- Ingeniería de Software de Componentes y Sistemas Distribuidos: abordando la elección de una arquitectura y los componentes adecuados para la construcción de un software multicapa.
- Ingeniería del Software, Programación Orientada a Objetos y Análisis y Diseño con Patrones: asignaturas que aportan los fundamentos necesarios para el desarrollo de software de calidad, reutilizable, de fácil mantenimiento y correctamente documentado.
- Asignaturas de corte transversal como Gestión de Proyectos o Ingeniería de Requisitos: que han permitido construir una solución de forma estructurada, siguiendo procesos y metodología para la captura, gestión y documentación de requisitos. Estableciendo una metodología de desarrollo que ha permitido la entrega de un producto que satisface los requisitos demandados, dentro del plazo establecido, con una calidad adecuada, tanto en el producto como en el proceso.

Podemos concluir que la realización del Trabajo de Fin de Grado ha permitido llevar a la práctica de forma satisfactoria los conocimientos adquiridos durante el estudio del Grado en Ingeniería Informática. Ha posibilitado la construcción de un software que da solución a un problema real, con un cumplimiento muy alto de los requisitos previstos en su especificación. Por lo que se considera que se han cumplido los objetivos tanto desde el punto de vista didáctico como desde el de la implementación.

9 Glosario

- **GPRS: G**eneral Packet Radio Service o servicio general de paquetes vía radio
- **GPS:** Global Positioning System o sistema de posicionamiento global
- **GSM:** Global System for Mobile communications o sistema global para las comunicaciones móviles.
- **IRIDIUM SBD:** Short Burst Data, sistema de comunicaciones satelital basado en la transmisión de mensajes cortos.
- **LTE:** Long Term Evolution, designa a la tecnología 4G.
- **NAVSTAR**: Red de satélites del sistema de posicionamiento global ,GPS, de Estados Unidos de Norte América
- **SMS:** Short Message Service o servicio de mensajes cortos.
- **UMTS:** Universal mobile telecommunications system. Sistema universal de telecomunicaciones móviles.

10 Bibliografía

Gonsalves, Antonio. Beginning Java EE 7. 1ª. ed. Apress. 2013. 577 p. ISBN: 978-1-4302-4626-8.

Çivici, Çağatay. Primefaces User Guide 6.1. 1ª. ed. [en linea]. PrimeTek Informatics. [fecha de consulta: 14 de octubre de 2017]. Disponible en: https://www.primefaces.org/docs/guide/primefaces_user_guide_6_1.pdf

Documentation. Red Hat, Inc. [en linea]. [fec ha de consulta: 30 de diciembre de 2017]. Disponible en: https://docs.jboss.org/author/display/WFLY10/Documentation

Jendrock Eric [et al.]. Java Platform, Enterprise Edition The Java EE Tutorial, Release 7. O. [en linea]. Oracle. Septiembre 2014. [fecha de consulta: 8 de octubre de 2017]. Disponible en: https://docs.oracle.com/javaee/7/tutorial/

Oracle America, Inc. JSR 338: Java TM Persistence API, Version 2.1. [en linea]. 2 Abril de 2013. [fecha de consulta: 5 de octubre de 2017]. Disponible en: http://download.oracle.com/otndocs/jcp/persistence-2_1-fr-eval-spec/index.html

Oracle America, Inc. JSR-340 Java Servlet 3.1 Specification. [en linea]. Abril de 2013. [fecha de consulta: 25 de octubre de 2017]. Disponible en: https://jcp.org/aboutJava/communityprocess/final/jsr340/index.html

Oracle America, Inc. JSR-342 Java Platform, Enterprise Edition 7 Specification. [en linea]. 5 de Abril de 2013. [fecha de consulta: 20 de octubre de 2017]. Disponible en: https://jcp.org/aboutJava/communityprocess/final/jsr342/index.html

Oracle America, Inc. JSR-344 JavaServer Faces 2.2 [en linea]. 22 de Abril de 2013. [fecha de consulta: 3 de noviembre de 2017]. Disponible en: https://jcp.org/aboutJava/communityprocess/final/jsr344/index.html

Oracle America, Inc. JSR-109 Implementing Enterprise Web Services. [en linea]. 15 de marzo de 2013. [fecha de consulta: 10 de octubre de 2017]. Disponible en: https://jcp.org/aboutJava/communityprocess/final/jsr109/index.html

Rational Software. Rational Unified Process. Best Practices for Software Development Teams. [en linea]. 1998. [fecha de consulta: 25 de diciembre de 2017]. Disponible en: https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

Red Hat, Inc. JSR-346 Contexts and Dependency Injection for Java EE 1.1. [en linea]. 2013. [fecha de consulta: 7 de octubre de 2017]. Disponible en:

http://download.oracle.com/otndocs/jcp/cdi-1_1-fr-eval-spec/

Wetherbee, Jonathan [et al.]. Beginning EJB 3: Java EE 7 Edition. 1ª. ed. Apress. 2013. 426 p. ISBN: 978-1-4302-4629-3

11 Anexos

Anexo I: Especificación_requisitos_J-Tracking.pdf

Anexo II: Prototipos _GUI_J-Tracking.pdf

Anexo III: Manual_de_usuario_J-Tracking.pdf