UNIVERSIDAD DE OVIEDO

ESCUELA DE INGENIERÍA INFORMÁTICA

PROYECTO FIN DE MÁSTER

"SISTEMA DE GESTIÓN INTEGRAL PARA UNA COMPAÑÍA DE TAXIS REAL"

Vº Bº del Director del Proyecto

DIRECTOR: José Manuel Redondo López

AUTOR: Vanesa Rodríguez Rodríguez

Agradecimientos

Parecía imposible llegar hasta aquí y finalmente he llegado, esto no podría haberlo hecho sin mi director de proyecto, José Manuel Redondo, quien aparte de ser uno de los mejores docentes que tenemos en esta Universidad, ya que no sólo consigue transmitirnos conocimientos, trabajo ya de por sí complicado, sino que su vocación va más allá y nos transmite hambre de aprender, motivación para ser buenos profesionales y aprecio por el trabajo bien hecho. Hace divertido este trabajo a veces de por sí no muy agradecido, no abandonándose al tedio y el conformismo., transmitiéndonos su entusiasmo, su buen hacer y cómo no, la gran persona que es, siempre dispuesto ayudar con una sonrisa, a pesar del día a día, el estrés, los plazos apretados y el trabajo sin fin. Enseñándonos también con su ejemplo que el compañerismo y el trabajo en equipo pueden llevarnos mucho más lejos. Gracias por tu optimismo, por tu confianza en mí, por tus conocimientos y por tu tiempo, me siento muy afortunada de que hayas sido mi director de proyecto.

También he de agradecer a Santiago Castro, por su impulso, por cuando decía, “No llego, no llego”, él siempre me respondía “Claro que sí llegas, hombre”. Gracias por en los momentos más oscuros ponerme ese punto de luz para no rendirme, sin ti y tu ayuda jamás habría llegado a ponerme delante de este tribunal.

Y cómo no Víctor Manuel Fernández Agüero, quien quizá ha pagado mis peores momentos de estrés y a pesar de todo, ha estado ahí apoyándome día a día.

A Laura Carús Palacio y a Taxi Piloña S. COOP por haber pensado en mí para este proyecto. Sé que es algo que les hace mucha ilusión.

Y por último y no menos importante, mis amigos y familia, por impulsarme, creer en mí, animarme y perdonar mi ausencia durante estos dos largos años.

Resumen

En la época en la que vivimos las aplicaciones móviles y web responsivas van ganando terreno sobre las aplicaciones tradicionales y páginas pensadas sólo para ordenadores.

Hoy en día el teléfono móvil se ha convertido en un imprescindible en la vida de la gente y las aplicaciones que nos facilitan el acceso de forma cómoda a todo tipo de servicios, restaurantes, cosmética, viajes, etc. han ido creciendo de manera exponencial.

Los taxistas de "Taxi Piloña S. COOP.", han pensado que ha llegado el momento de subirse al carro de las nuevas tecnologías de una manera más activa y eficiente, facilitando el acceso a sus servicios a clientes y la gestión de estos a ellos mismos. Por esto, desean una aplicación móvil en las que los clientes puedan registrarse y en su perfil de cliente solicitar un servicio. Por otra parte, un servidor se encargará de geolocalizar a los taxistas que estén disponibles y enviarles una petición de servicio empezando por los más cercanos al cliente.

La aplicación móvil para el taxista, aparte de poder rechazar o aceptar la solicitud, podrá ver en un mapa cuál es su posición respecto al cliente y podrá ver la ruta y ser guiado. También dispondrá de un botón de socorro que en caso de peligro enviará un aviso a sus compañeros.

En cuanto al cliente, podrá escoger en un mapa en que ubicación quiere que el taxista le recoja y podrá saber cuánto tiempo aproximado le espera para ser recogido ya que cada cierto tiempo el taxista enviará su posición a un servidor que mostrará esta ubicación en la aplicación del cliente.

Palabras Clave

Mapas, geolocalización, reserva, Android, iOS, taxi, cliente, distancias, servicios, SOS.

Abstract

At the time we live mobile applications and web responsive gaining ground on traditional applications and pages designed only for computers.

Nowadays the mobile phone has become an essential in the life of people and applications that facilitate us access in a comfortable way to all kinds of services, restaurants, cosmetics, travel, etc. They have been growing exponentially.

The taxi drivers of "Taxi Piloña S. COOP.", They have thought that the time has come to get on the bandwagon of new technologies in a more active and efficient way, facilitating access to their services and clients and the management of these to them. They themselves can use a mobile application in which customers can register and in their customer, profile request a service. On the other hand, a server will be responsible for geolocating the taxi drivers that are available and will send a service request starting with those closest to the client.

The mobile application for the taxi driver, besides being able to reject or accept the request, can see on a map his opinion about the clientele and can see the route and be guided. It also has a distress button that in case of danger will send a warning to his companions.

As for the client, you can choose from a map where the location wants the taxi driver to pick it up and can know how long the waiting time for the waiting time is the same as that of its location in the client's application.

Keywords

Maps, geolocation, booking, Android, iOS, cabby, client, distances, services, SOS.

Índice General

Índice de Figuras

Capítulo 1. Introducción

1.1 Justificación del Proyecto

La agilidad que nos da el disponer de los datos que necesitamos para nuestro trabajo en tiempo real en nuestro móvil, ese dispositivo que se ha vuelto imprescindible en nuestras vidas y que, en casi todos los trabajos, ha cobrado una importancia tan grande que podemos catalogarlo como imprescindible.

Compañías como Cabify o Uber, han apostado por aplicaciones de este estilo para ofrecer sus servicios de transporte, lo que las hace mucho más cercanas y competitivas.

El poder estar a un clic de reservar un taxi y que los taxistas gracias a esta herramienta sean convocados a aceptar ese servicio teniendo en cuenta su distancia geolocalizada del cliente y su disponibilidad como prioridad para aceptarlo.

Que el cliente sepa que su taxi está en camino y cuánto falta aproximado para su llegada.

Y, por último, la posibilidad de que si corre peligro la integridad del taxista este con un solo clic pueda alertar a sus compañeros.

Todas estas mejoras y motivos han llevado a los taxistas de “Taxi Piloña S. COOP.” a demandar estas aplicaciones móviles.

1.2 Objetivos del Proyecto

El objetivo principal de este proyecto es la creación de aplicaciones que hagan más accesibles la cooperativa de taxis de Piloña a sus clientes, así como facilitar su trabajo y mejorar su seguridad.

Con este proyecto se busca:

- Mejorar la eficacia de los servicios
- Facilitar el acceso a los clientes
- Facilitar la gestión de servicios
- Mejorar la seguridad de los taxistas

1.3 Estudio de la Situación Actual

En el mercado actual hay algunas aplicaciones diseñadas para taxistas, una parte de ellas permite registrar en una comunidad global de taxistas y aparecer en su red, otras proporcionan soluciones "a medida", añadiendo los logos de la empresa y permitiendo aparecer en una red en la que sólo aparezca la compañía del interesado.

Las de carácter global tienen el problema de que los taxistas de tu compañía deben estar cerca del cliente para tener la opción al viaje, los algoritmos de reparto de los viajes son basados en distancia y la que he encontrado que te permite una solución "a medida" hace la distribución de los servicios por tiempo de inactividad del taxista.

1.3.1.1 TaxiCaller

https://www.taxicaller.com/es/caracteristicas/app-de-pasajero

1.3.1.1.1 Descripción

TaxiCaller es una solución móvil para reservas de taxis. Consta de una aplicación para clientes gratuita y otra para taxistas que dependiendo de si son flotas de más de 20 taxistas o menos te ofrece una tarifa que denominan "Profesional" y otra que denominan "Enterprise".

Los servicios que ofrecen tanto en una como en la otra son prácticamente los mismos, salvo que la "Entreprise" se puede personalizar. La tarifa de la primera son 20€ mensuales más IVA por taxi y la segunda te hacen ellos un precio a medida.

La aplicación para taxista tiene la opción de un taxímetro virtual en caso de que las tarifas estén basadas en él, permite configurar turnos y horarios, le indica si el servicio fue demandado en la calle o desde la central de taxis, aceptar y rechazar servicios, así como indicar que el taxista está ocupado, libre o ausente.

Le da la opción de navegar para conocer el cómo llegar desde el destino al origen, así como estimar el tiempo que tardará.

En la parte del cliente, tú puedes escoger en un mapa dónde quieres que te recoja el taxi y dónde quieres que te deje, indicas el número de ocupantes y si llevas silla de ruedas o no y las compañías dadas de alta en esa zona en la aplicación te aparecen para que selecciones en la que deseas viajar. También te aparece el precio estimado y la opción de escoger si quieres el taxi para ahora o para otra hora.

1.3.1.1.2 Opiniones y pruebas

1.3.1.1.2.1 Aplicación cliente

Las opiniones que tienen en el store son negativas, parece que hay pocas compañías adscritas a ella, aparte de que informa de precios y tarifas erróneas o que luego disienten con las cobradas por la empresa de taxis seleccionada.

Por los comentarios también parece ser que, al aproximar la tarifa a los clientes, se escoge una ruta predeterminada para el taxista, lo que hace que si toma otra ruta y el precio no es similar al de la aplicación, los clientes estén descontentos.

Me la he descargado para probarla y en el caso de Oviedo si me aparecen tres compañías de taxis, pero tengo el mismo problema que muchos de los usuarios, la aplicación no me deja escoger tipo de vehículo, ni hacer la reserva porque me pone que no hay vehículos disponibles con lo que cuando escojo una de las compañías se activa la llamada, no dejándote hacer la reserva por la aplicación, con lo que finalmente no tiene mucho sentido su utilización en mi ciudad.

Su soporte para clientes, en cuanto las quejas responden siempre lo mismo que contactes con ellos y les digas que compañía de taxis quieres contratar que ellos se limitan a realizar software para estas compañías.

1.3.1.1.2.2 Aplicación taxista

En cuanto a los conductores, los comentarios son positivos, excepto algunas quejas sobre el mapa, que dicen que no es exacto y por los que usan taxímetro.

1.3.1.2 myTaxi

https://es.mytaxi.com/index.html

1.3.1.2.1 Descripción

Es una solución del estilo a las conocidas Cabify o Uber. Es gratuita tanto para clientes como para taxistas, pero en el caso de los taxistas deberán pagar una comisión por carrera completada. Y sólo está presente en cuatro ciudades españolas; Barcelona, Madrid, Sevilla y Valencia.

La aplicación del cliente te permite registrarte o bien iniciar sesión con Facebook o con tu cuenta de Google. Aunque escojas una de estas últimas opciones, debes rellenar campos de registro como tú contraseña y tú número de móvil, no accedes directamente a los servicios con la cuenta.

Tiene una opción que te permite guardar tus taxistas favoritos y también te permite pagar a través de la aplicación. También te estiman el precio del trayecto. Escogerías en un mapa tu origen y el destino al que quieres ir, también te permite introducir las direcciones a mano y si hay taxistas que operen en tu zona, el taxista más cercano pasará a recogerte.

En cuanto a la aplicación del taxista, permite cobrar a través de la aplicación, si el cliente no se presenta dónde se le debería recoger siguiendo unas pautas que te indican, al taxista le reembolsan 5€ por las molestias.

La aplicación le indica al taxista los servicios cercanos para que pueda recoger a los clientes y no hay más información de cómo funciona.

1.3.1.2.2 Opiniones y pruebas

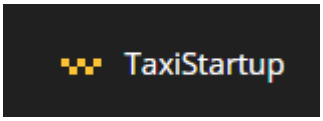
1.3.1.2.2.1 Aplicación cliente

La mayor parte de las opiniones de los clientes son positivas, exceptuando algunas que comentan que la estimación que la aplicación da del coste del servicio puede variar mucho.

Me he descargado la aplicación, pero no he podido hacer más pruebas que el registro, ya que me dice que lamentablemente en mi zona (Oviedo) no operan, cosa que ya sabíamos porque sólo operan en las ciudades anteriormente mencionadas.

1.3.1.2.2.2 Aplicación taxista

Tiene algunas quejas por la priorización de la selección de taxis y que a veces es difícil que te permita coger un servicio si no es muy cercano a la ubicación del taxista en ese momento.



1.3.1.3 TaxiStartup

https://taxistartup.com/es/product/

1.3.1.3.1 Descripción

Tienen dos tarifas, una denominada “Inicio”, en la que se pagaría un 5% + 20C por viaje exitoso y otra denominada escala en la que se pagarían 20$ por conductor y mes.

Las tarifas también dependen de si quieres una solución de marca blanca o la estándar genérica, es decir, que te hagan específicas para tu flota de taxis las aplicaciones que aparte del as cuotas mensuales hay que incluir el plus de desarrollo que son 99$ al mes hasta que este adaptada y 9c por conductor cuando este esté asignado con éxito.

La aplicación del taxista ofrece, datos del cliente que solicita el servicio y le permite navega para conocer las rutas. Tiene un botón específico para reservas en hoteles, restaurantes, bares y recepciones. También permite establecer costes fijos por zonas y tiene un algoritmo de distribución de los servicios que se basa en la cantidad de tiempo que lleva un taxista sin recibir servicios.

La aplicación del cliente te ofrece la posibilidad de escoger el tipo de taxi que quieres que te recoja entre los que hay disponibles, donde quieres que te recojan y te lleven en un mapa, te dice si hay taxistas disponibles y, sino que lo intentes más tarde. El cliente también puede indicar un medio de pago y pagar a través de la aplicación si lo desea. También permite hacer una valoración del taxista.

1.3.1.3.2 Opciones y pruebas

1.3.1.3.2.1 Aplicación del cliente

Hay una aplicación de prueba en el store, la he probado y me deja registrarme, en Asturias no existe así que la he probado en Madrid, me deja escoger el tipo de taxi que tiene y luego me salta el mensaje de que no hay taxistas disponibles.

Las opiniones que tiene la aplicación del a demo son positivas, pero sin poner motivos o motivos como “el taxista ha sido puntual”, por lo que no tengo ninguna referencia sobre la aplicación.

1.3.1.3.2.2 Aplicación del taxista

En el store también hay una aplicación para el taxista, es la solución no a medida. Te piden introducir tú teléfono y luego te envían un código, cuando lo introduces sale un mensaje de que no perteneces a ninguna compañía de taxis que debes ponerte en contacto con ella.

En las opiniones de los clientes, son positivas la mayoría sin comentarios y luego hay algunas de que les pasa el mismo problema que a mí, entiendo que no pueden contratar los servicios a menos que sea poniéndose en contacto con el servicio técnico de la compañía antes de descargarse la aplicación. Por otro lado, algún taxista se queja de que, si el cliente no paga, no se hacen cargo.

Capítulo 2. Aspectos Teóricos

Con el fin de aportar claridad se va a subdividir la explicación de estos entre las tecnologías, patrones de diseño y demás aspectos a tener en cuenta entre los empleados en el servicio web y en las aplicaciones cliente, así como un apartado específico para los patrones de diseño ya que alguno no se usa exclusivamente en ninguno de los dos.

2.1 Servicio web

En el servidor se decidió utilizar **Spring Boot,** debido a que es una tecnología que me interesaba aprender y facilita mucho este tipo de implementaciones. Obviamente al utilizar Spring Boot se está utilizando **Spring Framework**, para enlazar la explicación de estos dos términos que van unidos vamos a empezar por explicar este último, que es la base de todo.

Un framework explicado de manera sencilla es un conjunto de clases que se utilizan para crear objetos que la aplicación que estamos desarrollando necesitan y nos facilita esta creación. Es habitual la necesidad de utilizar varios frameworks, lo que suele conllevar problemas ya que cada uno gestiona el ciclo de vida de sus objetos. Y aquí es donde entra Spring Framework, solventa este problema encargándose de generar los objetos de cada framework basándose en ficheros XML y anotaciones.

Spring Boot nos ayuda aún a simplificar más el uso del Spring Framework, normalmente cuando construimos una aplicación debemos seguir los siguientes pasos:

1- Seleccionar los jars con Maven para descargarse las dependencias necesarias
2- Crear la aplicación
3- Desplegar en el servidor

Spring Boot simplifica los pasos 1 y 3. Algunas de las características que nos permiten hacer esto son:

- Servidores de aplicaciones embebidos
- POMs con dependencias y plug-ins para Maven
- Uso extensivo de anotaciones que realizan funciones de configuración, inyección, etc.

Para la implementación de los controladores, se han utilizado también características de **Spring MVC,** que está basado en patrón de diseño modelo-vista-controlador, del que hablaremos en el apartado correspondiente.

Un controlador, normalmente prepara el modelo y selecciona que vista es la encargada de mostrar los datos, pero, los controladores también son capaces de generar respuestas sin necesidad de una vista y esto es útil al generar los servicios que generan respuestas en formato **JSON**, que es el utilizado en este proyecto para el intercambio de servicios. JSON es una notación de objetos basada en JavaScript, que permite de forma rápida y simple crear objetos. Un ejemplo sería el siguiente:

```
var objetoJSON = {"usuario":"user","password":"passw"};
```

En cuanto a nuestro modelo del que hemos hablado anteriormente, aprovechando que estamos utilizando Spring, se ha utilizado **Spring Data** para facilitar el acceso a la tecnología de persistencia, proporciona soporte para **JPA,** que es la utilizada, simplificando de esta forma la implementación de la capa de acceso de datos. Permite automatizar la construcción de repositorios para recuperar los datos de la base de datos. Se construye el modelo de la aplicación en JPA y se utiliza el **entityManager** de **Hibernate** que Spring Data ya te da hecho.

Con cada tipo de tecnología de persistencia los DAOs ofrecen las funcionalidades típicas de un CRUD para objetos de dominios propios, métodos de búsqueda, ordenación y paginación. Spring Data proporciona interfaces genéricas para estos aspectos (CrudRepository, PagingAndSortingRespository) e implementaciones específicas para cada tipo de tecnología de persistencia.

En **java** (lenguaje de programación en el que se ha implementado el servidor), los datos del modelo se representan como objetos, pero las bases de datos, en este caso se ha optado por **MySQL** debido a que es ampliamente utilizada para implementaciones web y que tiene una documentación extensa y de calidad, guardan sus datos en forma relacional. Por lo que es necesaria una herramienta que adapte estos dos "mundos". Los frameworks que se encargan de esto se conocen como ORM. JPA es un framework que forma parte de java y ofrece un conjunto de interfaces y APIs para resolver el problema de los objetos en una base de datos relacional. JPA no es una implementación, sólo proporciona las interfaces para ser implementadas por los distintos proveedores. **Hibernate** es un ORM que implementa JPA. Es de los más utilizados y tiene una documentación al igual que MySQL amplia.

2.1.1 Referencias a la documentación

- Spring Boot: https://spring.io/projects/spring-boot.
- Spring Data JPA: https://spring.io/projects/spring-data-jpa.
- Spring Framework: https://spring.io/projects/spring-framework.
- MySql: https://spring.io/projects/spring-framework.
- JPA:http://www.oracle.com/technetwork/java/javaee/tech/persistence-jsp-140049.html
- Hibernate: http://hibernate.org/orm/

2.2 Clientes

Para las aplicaciones cliente se ha decidido utilizar **Ionic,** que es un SDK de código abierto para el desarrollo de aplicaciones móviles híbridas, utilizando tecnologías web como **CSS, HTML5 y Sass.** Está basado en **Angular.** Las aplicaciones pueden compilarse en estas tecnologías web y luego distribuirse a través de las tiendas de aplicaciones nativas para que se instalen en los dispositivos aprovechando **Cordova**, sobre el que está construido Ionic.

Otras de las características que han llevado a decantarse por este framework ha sido su alto rendimiento, es más rápido porque está construido para la mínima manipulación del DOM, y no usa JQuery, además de aceleración de transiciones por hardware.

Otra de sus grandes ventajas, como ya se ha mencionado, es que te permite con un desarrollo, exceptuando cosas muy específicas, realizar un desarrollo y compilar para iOS y para Android.

Como hemos mencionado Ionic está basado en Angular, que es un framework para aplicaciones web desarrollado en **TypeScript** y basado en MVC. Porque Ionic utiliza TypeScript para el desarrollo del código.

TypeScript es lo que se conoce como un **superset** de JavaScript, es decir es un lenguaje escrito sobre otro o, mejor dicho, que compila a otro lenguaje. TypeScript compila a JavaScript.

Los lenguajes como JavaScript basados en un estándar evolucionan muchas veces más lentos de las necesidades de los desarrolladores, por esto, empresas o comunidades, deciden expandir el lenguaje aportando herramientas necesarias para poder desarrollar en unas mejores condiciones.

Los superset compilan en el lenguaje estándar, por lo que el desarrollador programa en el lenguaje expandido, pero luego el código es “transpilado” para ser transformado en el lenguaje estándar.

Y Apache Cordova es una plataforma de código abierto que contiene un set de librerías en JavaScript que permite al desarrollador acceder a funciones nativas del dispositivo móvil, tales como el GPS. Es multiplataforma y permite el desarrollo de aplicaciones multiplataforma en HTML5, CSS y JavaScript.

2.2.1 Referencias a la documentación

- Ionic: https://ionicframework.com/docs/
- Angular: https://angular.io/tutorial/toh-pt1
- TypeScript: https://www.typescriptlang.org/docs/handbook/basic-types.html
- Cordova: https://cordova.apache.org/

2.3 Patrones de diseño

Un patrón de diseño son unas técnicas para resolver problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño e interacción de interfaces. Según la escala o nivel de abstracción podemos catalogarlos en:

- Patrones de arquitectura: expresan un esquema organizativo estructural fundamental para sistemas de software.
- Patrones de diseño: expresan esquemas para definir estructuras de diseño o sus relaciones, con las que construir sistemas de software.
- Dialectos: Patrones de bajo nivel específicos para un lenguaje de programación o entorno concreto.

En este caso tanto en el servidor con Spring MVC como en el cliente con Angular, se utiliza el **patrón de diseño MVC**.

Este es un patrón de diseño del primer tipo, de arquitectura, que consiste en una separación de los datos (capa de modelo), del módulo encargado de gestionar los eventos y las comunicaciones (controlador) y por último la interacción con el usuario(vista).

Se basa en la idea de reutilización de código y la separación de los conceptos, con esto, se busca facilitar las tareas de desarrollo y su mantenimiento.

En el servidor, también se utiliza el **patrón de arquitectura n-capas**, cuyo objetivo es el desacoplamiento de las partes que componen el sistema, que es necesario en una arquitectura cliente-servidor: lógica de negocios, capa de presentación y capa de datos.

Y con Angular se ha utilizado el **patrón observer**. Angular tiene la capacidad de proporcionar automáticamente una actualización de las fuentes de información, lo que nos ahorra mucho trabajo como desarrolladores, pero, tiene un coste de rendimiento. La solución a esto fue la utilización del patrón observer, que evita tener que hacer consultas repetitivas de acceso a la fuente de información.

Al hablar de observables hemos de hablar de un concepto muy relacionado con estos que es la programación reactiva, esta tiene que ver con el flujo de ejecución de las instrucciones.

En la programación tradicional las instrucciones se ejecutan secuencialmente, por lo tanto, los resultados que vamos obteniendo no varían, aunque cambien los valores de las variables, por ejemplo:

```
let a = 1;

let b = 2;

let resultado = a + b;

a = 7;
```

En este momento, aunque el valor de a haya cambiado, nuestro resultado sigue siendo 3.

En el caso de la programación reactiva, la variable "resultado" habría actualizado su valor al modificarse las variables con las que se ha realizado el cálculo.

La programación reactiva, es la programación con flujos asíncronos. En este tipo de programación, se pueden crear flujos o streams a partir de casi cualquier cosa, como puede ser el valor de una variable que va cambiando a lo largo del tiempo. Casi todo puede ser un flujo de datos, clics sobre botones, cambios en una estructura de datos, una consulta para traer un JSON del servidor, etc.

Estos flujos de datos se tienen en cuenta y se crean sistemas que son capaces de consumirlos de distintos modos, fijándose en lo que realmente nos interesa de esos streams y desechando el resto.

Como objetivo final, reactive programming se ocupa de lanzar distintos eventos sobre los flujos:

- La aparición de algo "interesante" dentro de ese flujo
- La aparición de un error en el stream
- La finalización del stream

El patrón observer, es un modo de implementación de la programación reactiva. Lo que hace es poner en funcionamiento diferentes actores produciendo dichos eventos y consumiéndolos de diversos modos. Los componentes principales de este patrón son:

- Observable: Lo que deseamos observar. Mediante él, nos podemos suscribir a eventos que nos permiten actuar cuando cambia lo que estamos observando.
- Observer: Es el que se dedica a observar. Se implementa mediante una colección de funciones callback, que nos permiten escuchar los eventos o valores emitidos por el observable.
- Subject: Es el emisor de eventos, el que crea los eventos cuando el observable sufre cambios, que serán los eventos consumidos por el observer.

Existen varias librerías para implementar programación reactiva que hacen uso de este patrón, una de ellas y la que se ha utilizado en este proyecto, ya que es la que usa Angular, es RxJs. Esta librería nos ofrece una base de código JavaScript para producir, consumir streams y para manipularlos.

2.3.1 Referencias a la documentación

- RxJs:http://reactivex.io/rxjs/class/es6/Observable.js~Observable.html#instance-method-subscribe

Capítulo 3. Planificación del Proyecto y Resumen de Presupuestos

3.1 Planificación

Para realizar la planificación se han tenido en cuenta las horas libres, ya que tengo un trabajo a jornada completa y el tiempo que me permite este es el que voy a dedicar al proyecto. Se ha hecho una estimación según la experiencia laboral, no es sobre aplicaciones móviles, pero si trabajo en proyectos informáticos. Y se han tenido en cuenta las fechas de entrega final del proyecto.

Este proyecto va a ser realizado por una única persona, por lo que no se han podido realizar tareas en paralelo, lo que si se ha hecho el desarrollo un poco conjunto ya que muchas de ellas están vinculadas.

3.1.1 Diagrama de Gantt

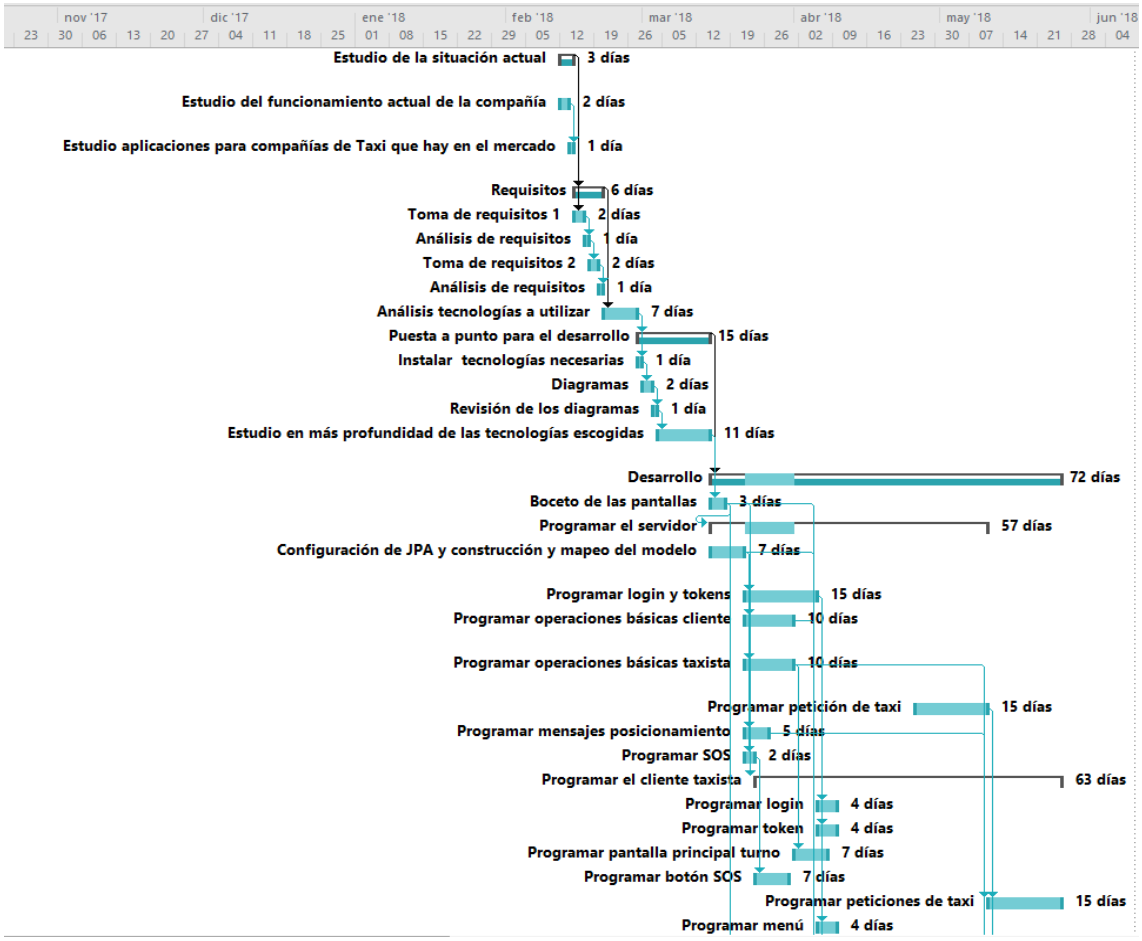


Ilustración 1 – Diagrama de Gantt(detalle I)

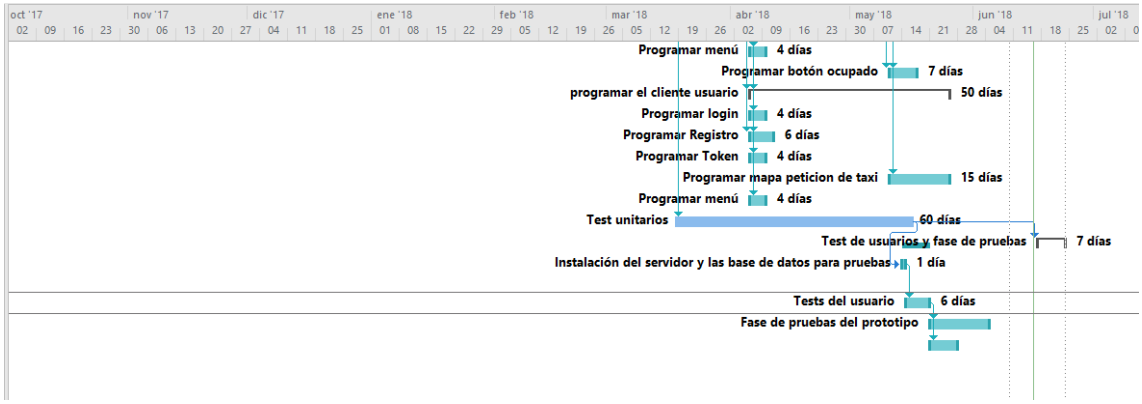


Ilustración 2 – Diagrama de Gantt (Detalle II)

3.1.2 Detalle de las principales tareas

Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
	Estudio de la situación actual	**3 días**	**lun 12/02/18**	**mié 14/02/18**	
	Estudio del funcionamiento actual de la compañía	2 días	lun 12/02/18	mar 13/02/18	
	Estudio aplicaciones para compañías de Taxi que hay en el mercado	1 día	mié 14/02/18	mié 14/02/18	2
	Requisitos	**6 días**	**jue 15/02/18**	**mar 20/02/18**	**1**
	Toma de requisitos 1	2 días	jue 15/02/18	vie 16/02/18	1
	Análisis de requisitos	1 día	sáb 17/02/18	sáb 17/02/18	5
	Toma de requisitos 2	2 días	dom 18/02/18	lun 19/02/18	6
	Análisis de requisitos	1 día	mar 20/02/18	mar 20/02/18	7
	Análisis tecnologías a utilizar	7 días	mié 21/02/18	mar 27/02/18	4
	Puesta a punto para el desarrollo	**15 días**	**mié 28/02/18**	**mié 14/03/18**	**9**
	Instalar tecnologías necesarias	1 día	mié 28/02/18	mié 28/02/18	9
	Diagramas	2 días	jue 01/03/18	vie 02/03/18	11
	Revisión de los diagramas	1 día	sáb 03/03/18	sáb 03/03/18	12
	Estudio en más profundidad de las tecnologías escogidas	11 días	dom 04/03/18	mié 14/03/18	13
	Desarrollo	**72 días**	**jue 15/03/18**	**vie 25/05/18**	**10**
	Boceto de las pantallas	3 días	jue 15/03/18	sáb 17/03/18	14
	Programar el servidor	**57 días**	**jue 15/03/18**	**jue 10/05/18**	**16**
	Configuración de JPA y construcción y mapeo del modelo	7 días	jue 15/03/18	mié 21/03/18	
	Programar login y tokens	15 días	jue 22/03/18	jue 05/04/18	18
	Programar operaciones básicas cliente	10 días	jue 22/03/18	sáb 31/03/18	18
	Programar operaciones básicas taxista	10 días	jue 22/03/18	sáb 31/03/18	18
	Programar petición de taxi	15 días	jue 26/04/18	jue 10/05/18	
	Programar mensajes posicionamiento	5 días	jue 22/03/18	lun 26/03/18	18
	Programar SOS	2 días	jue 22/03/18	vie 23/03/18	18
	Programar el cliente taxista	**63 días**	**sáb 24/03/18**	**vie 25/05/18**	**16**
	Programar login	4 días	vie 06/04/18	lun 09/04/18	19
	Programar token	4 días	vie 06/04/18	lun 09/04/18	19
	Programar pantalla principal turno	7 días	dom 01/04/18	sáb 07/04/18	21
	Programar botón SOS	7 días	sáb 24/03/18	vie 30/03/18	24
	Programar peticiones de taxi	15 días	vie 11/05/18	vie 25/05/18	23;22
	Programar menú	4 días	vie 06/04/18	lun 09/04/18	19

Ilustración 3 – Detalle principales tareas (I)

Modo de	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
	Programar mapa peticion de taxi	15 días	vie 11/05/18	vie 25/05/18	22
	Programar menú	4 días	vie 06/04/18	lun 09/04/18	19
	Test unitarios	60 días	dom 18/03/18	mié 16/05/18	16
	Test de usuarios y fase de pruebas	**7 días**	**dom 17/06/18**	**sáb 23/06/18**	**39**
	Instalación del servidor y las base de datos para pruebas	1 día	lun 14/05/18	lun 14/05/18	39
	Tests del usuario	6 días	mar 15/05/18	dom 20/05/18	41
	Fase de pruebas del prototipo	15 días	lun 21/05/18	lun 04/06/18	42
	Cierre del proyecto para presentación	7 días	lun 21/05/18	dom 27/05/18	42

Ilustración 4 -Detalle tareas principales (II)

3.2 Resumen del Presupuesto

El siguiente presupuesto, sería el que se le entregaría al cliente. Para ver en detalle el mismo se puede recurrir al presupuesto detallado.

DETALLE	COSTE (€)
FASES DEL PROYECTO	*7144,62*
FORMACIÓN	799,26
ANÁLISIS	379,70
DISEÑO	379,70
IMPLEMENTACIÓN	4634,32
DOCUMENTACIÓN	951,64
IVA (21%)	1500,37
PRESUPUESTO TOTAL	***8644,99***

Capítulo 4. Análisis

En este apartado se expondrá la especificación de requisitos, así como toda la documentación de análisis que posteriormente se utilizará para la elaboración del diseño.

4.1 Definición del Sistema

4.1.1 Determinación del Alcance del Sistema

Este proyecto tiene la finalidad de proveer a los taxistas de Piloña de herramientas tecnológicas para desarrollar su trabajo de forma más sencilla, eficiente y competitiva.

Tras realizar un estudio del tiempo disponible y los recursos, se ha decidido que el proyecto abarque la implantación de un prototipo de aplicación móvil tanto para taxista como para cliente que, a la finalización de este, estará en fase de pruebas por los clientes para sus posibles retoques e implantación final.

Tras el estudio de la situación actual se ha decidido:

- La aplicación móvil va a desarrollarse en Ionic porque permite el desarrollo de aplicaciones tanto para Android, como para Iphone.
- El prototipo inicial, si tuviese partes específicas, es decir nativas se desarrollará en un primer momento para Android.
- Los usuarios podrán conectarse desde varios dispositivos, siendo posible ver estás sesiones en un menú y desactivarlas.
- El taxista dispondrá de un botón de socorro que en caso de peligro alertará a sus compañeros.
- El taxista podrá aceptar o rechazar una petición de servicio.
- El taxista podrá ver en un mapa dónde se encuentra el cliente que solicita el servicio.
- El taxista dispondrá de la posibilidad de navegación.
- El cliente podrá solicitar un taxi en un mapa.
- El taxista que esté más cerca y libre será el primero en recibir la petición de taxi, si este la rechaza o no responde en un tiempo determinado dicha petición se le enviará al siguiente y así consecutivamente.
- El taxista dispondrá de un botón (ocupado/libre)
- No se contempla un chat entre usuarios, ya que se va a utilizar en conducción y puede ser peligroso, y en el caso de los clientes, no estamos ante una red social.

4.2 Requisitos del Sistema

4.2.1 Obtención de los Requisitos del Sistema

Código	Nombre Requisito	Descripción del Requisito
R1.1	Registro de usuario	Se deben registrar los usuarios cliente, ya que se debe disponer de sesión para pedir un taxi, para así evitar servicios fallidos. A los taxistas los registrará el administrador del sistema.
R1.2	Inicio de sesión aplicaciones móvil	Se permitirá a los usuarios iniciar sesión en distintos dispositivos.
R1.3	Gestión de sesiones abiertas	Los usuarios podrán consultar y cancelar las sesiones que tengan abiertas.
R1.4	Deslogueo	Los usuarios podrán desloguearse de la sesión activa cuando lo deseen.
R2.1	Petición de taxi	Un cliente puede pedir un taxi a un lugar determinado. Dispondrá de un mapa para dicha solicitud, localizándose en su posición actual.
R3.1	Recepción de la petición de taxi	Se enviará la petición al taxista más cercano libre. En caso de no poder atender este la petición se le enviará al resto de taxistas libres por orden de proximidad al cliente.
R3.2	Rechazar una petición	El taxista podrá rechazar una petición
R3.3	Aceptación de petición	El taxista podrá aceptar una petición
R3.4	Ubicación de la petición	El taxista podrá comprobar en un mapa dónde es el lugar de recogida solicitado.
R4.1	El cliente podrá ver la posición del taxista	El cliente, cuando el taxista ya esté cubriendo el servicio, podrá ver cada cierto tiempo en el mapa que ha hecho la petición como el taxista asignado se aproxima a su localización en tiempo real.
R4.2	El taxista dispondrá del a posibilidad de navegabilidad	Cuando el taxista este cubriendo el servicio, dispondrá de la opción de ver en un mapa la posición del cliente y si fuese necesario o lo desease activar el navegador para que le indique la ruta.
R.5.1	Taxi ocupado en servicio	Cuando un taxista haya aceptado una petición aparecerá como ocupado.
R5.2	Taxi ocupado sin servicio	El taxista podrá indicar al sistema que está ocupado en el momento que lo desee.
R5.3	Pantalla inicio taxista	En la pantalla de inicio (una vez logueado) del taxista aparecerá el turno en curso, inicio, final y cuánto tiempo falta para su finalización.
R5.4	Pantalla inicio cliente	En la pantalla de inicio del cliente, aparecerá el mapa para solicitar un servicio.
R6.1	Botón de socorro	El sistema dispondrá de un botón de socorro que el taxista pulsará en caso de peligro y enviará un mensaje al resto de compañeros con su ubicación.

Además de lo descrito anteriormente el sistema debe cumplir los siguientes requisitos no funcionales:

- Requisitos de usuario: Las aplicaciones deben ser sencillas e intuitivas, para personas de todos los rangos de edad que quieran solicitar un taxi.
- Requisitos tecnológicos: Para poder instalar la aplicación móvil el dispositivo con Android debe tener la versión del sistema operativo igual o superior a 4.4 (Kit Kat), el nivel de API será el 19.
- Requisitos de usabilidad: La aplicación del taxista debe contener elementos grandes que permitan su uso de una forma fácil desde el taxi.
- Requisitos de seguridad: Los inicios de sesión serán gestionados por tokens en el móvil, que tendrán una longitud suficiente y aleatoriedad para que sean difíciles de generar para usurpar una identidad. Se tendrá control de los accesos, no permitiendo accesos al sistema no permitidos. Los usuarios tendrán perfiles bien definidos para que un usuario común no pueda hacerse pasar por taxista.

4.2.2 Identificación de Actores del Sistema

En este sistema se identifican los siguientes actores:

- Usuario taxista
- Usuario cliente
- Usuario administrador: Será el encargado de dar de alta a los taxistas y sus turnos en las bases de datos.

4.2.3 Especificación de Casos de Uso

A continuación, se muestran los diferentes casos de uso del sistema, dependiendo del actor implicado en el mismo.

4.2.3.1 Casos de uso cliente

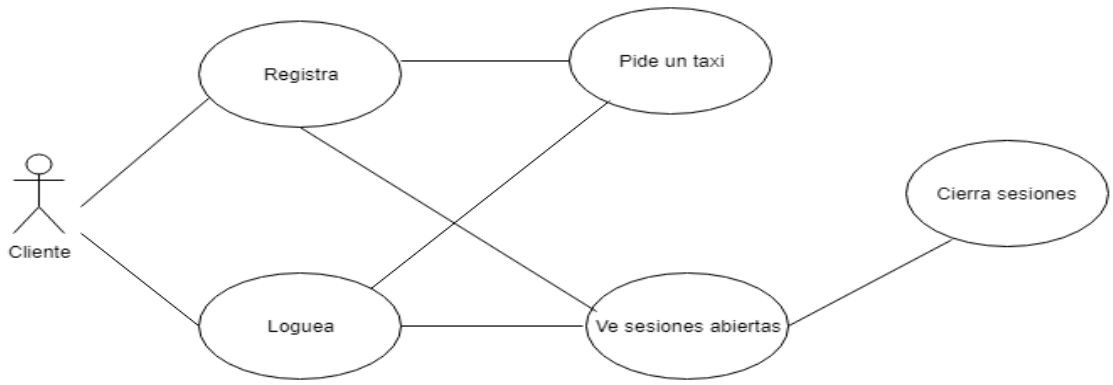


Ilustración 5 – Casos de uso cliente

4.2.3.2 Casos de uso taxista

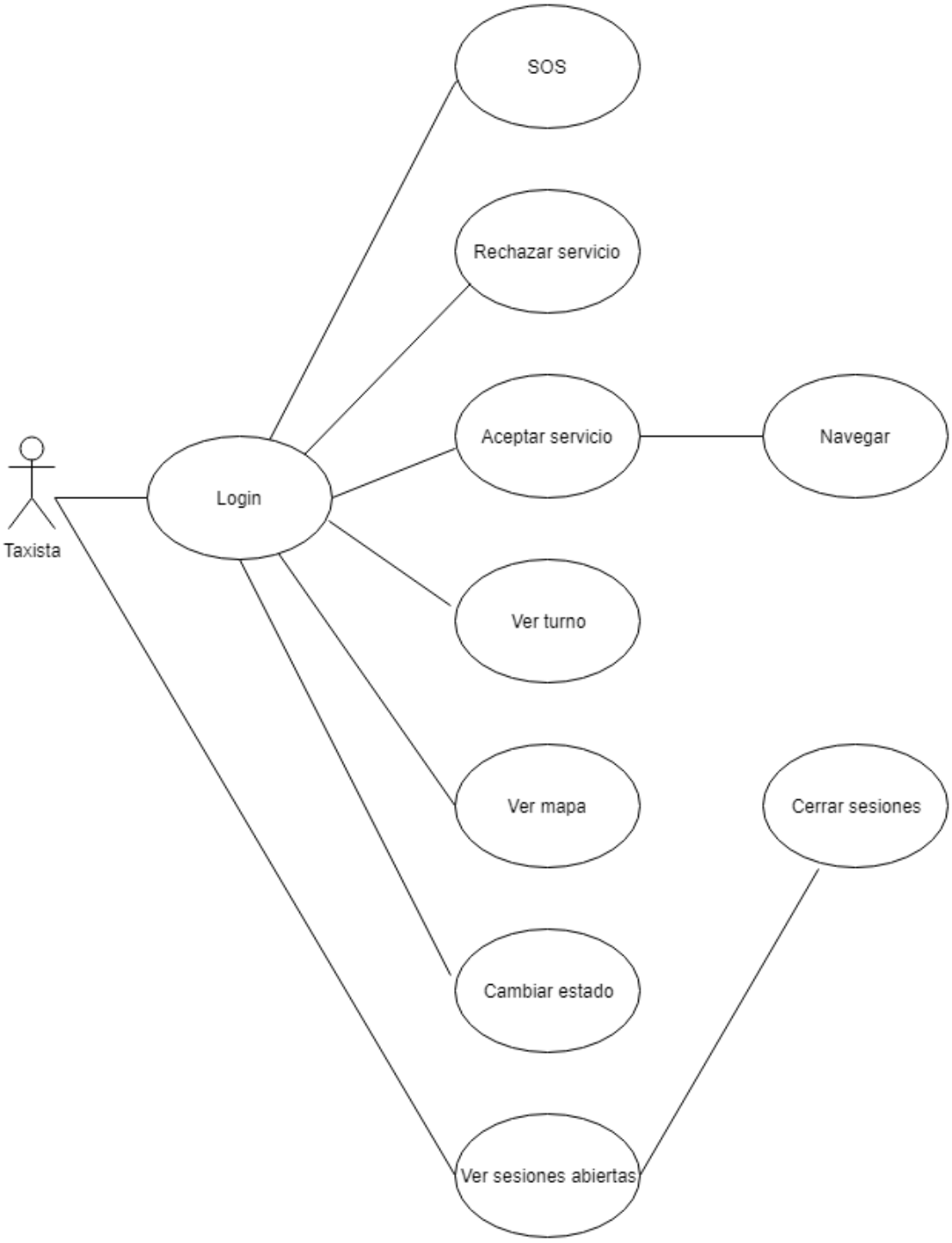


Ilustración 6 – Casos de uso taxista

4.2.3.3 Casos de uso administrador

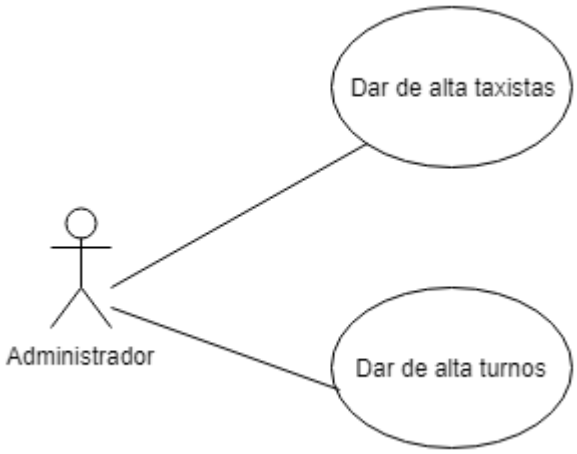


Ilustración 7 – Casos de uso administrador

Nombre del Caso de Uso
Registro de usuario
Descripción
A través de una pantalla que contendrá un formulario se podrá registrar un cliente nuevo en el sistema. En el formulario se pedirán: • Nombre • Teléfono • DNI • Dirección • Contraseña • Confirmación de la contraseña

Nombre del Caso de Uso
Logueo
Descripción
Los usuarios tanto taxista como cliente, dispondrán de una pantalla para loguearse en el sistema.

Nombre del Caso de Uso
Pedir un taxi
Descripción
El programa cargará y mostrará el examen al alumno y dará las indicaciones oportunas para su realización. Anteriormente, el alumno podrá elegir uno de los exámenes disponibles que la aplicación le ofrecerá.

Nombre del Caso de Uso
Visualizar sesiones abiertas
Descripción
Tanto el cliente como el taxista podrán mantener sesiones abiertas en distintos dispositivos y dispondrán de una opción en un menú que les permita visualizar dichas sesiones.

Nombre del Caso de Uso
Cerrar sesiones
Descripción
En el menú en el que se visualicen las sesiones abiertas, que será el mismo que contenta la opción de deslogueo, permitirá que los usuarios, tanto taxistas como clientes cierren las sesiones que consideren pertinentes.

Nombre del Caso de Uso
SOS
Descripción
El taxista dispondrá de un botón de socorro, de color rojo, que al pulsarlo enviará su ubicación al resto de taxistas informándoles de que esta en peligro.

Nombre del Caso de Uso
Rechazar servicio
Descripción
El taxista dispondrá de una opción para rechazar un servicio. Si el taxista no lo acepta en un tiempo predefinido, será rechazado por el sistema.

Nombre del Caso de Uso
Aceptar un servicio
Descripción
El taxista podrá aceptar un servicio desde que le entra la petición en un tiempo determinado, sino esta caducará.

Nombre del Caso de Uso
Ver turno
Descripción
El taxista dispondrá de una pantalla en la que verá en que turno se encuentra, cuando empezó, a que hora finaliza y cuánto tiempo le queda para acabar la jornada laboral.

Nombre del Caso de Uso
Navegar
Descripción
El taxista dispondrá de la posibilidad de navegación en caso de necesitar indicaciones para llegar al destino solicitado.

Nombre del Caso de Uso
Ver mapa
Descripción
Cuando una petición de un servicio de taxi aparezca en el móvil, el taxista tendrá la opción de ver un mapa con la posición del cliente antes de aceptar o rechazar el servicio.

Nombre del Caso de Uso
Cambiar estados
Descripción
El taxista podrá cambiar su estado de libre a ocupado cuando lo desee. Cuando acepte un servicio el estado se pondrá como ocupado automáticamente. Cuando finalice un servicio. el taxista cambiará su estado y este desaparecerá de la pantalla.

Nombre del Caso de Uso
Dar de alta taxista
Descripción
El administrador del servidor será el encargado de registrar los taxistas en la base de datos.

Nombre del Caso de Uso
Dar de alta turnos
Descripción
El administrador del servidor será el encargado de insertar en la base de datos los turnos de cada taxista.

4.3 Identificación de los Subsistemas en la Fase de Análisis

El sistema se puede dividir en cuatro partes principales:

- Subsistema de gestión de la base de datos.
- La aplicación móvil para el cliente.
- La aplicación móvil para el taxista.
- El servicio Web.

4.3.1 Descripción de los Interfaces entre Subsistemas

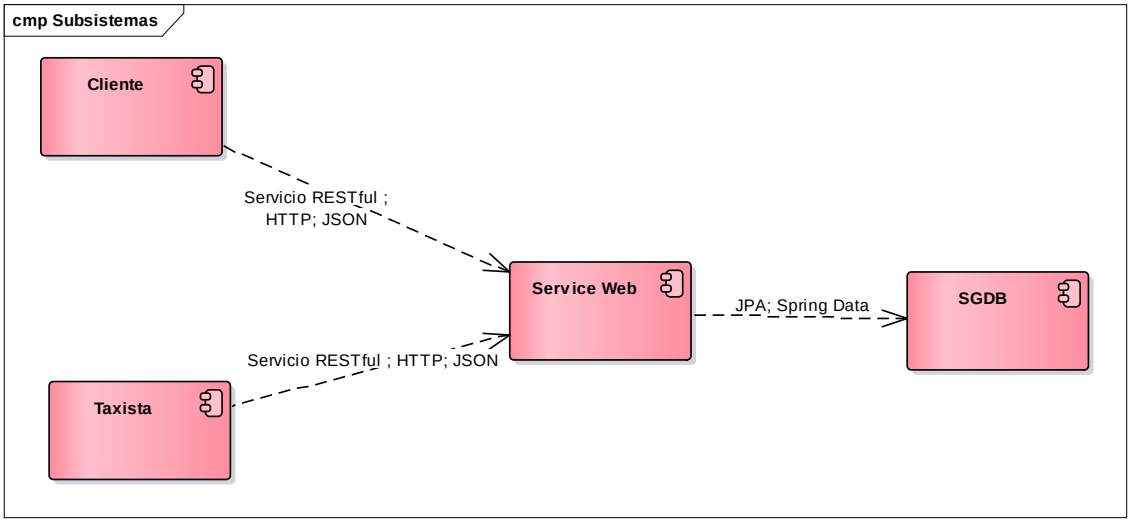


Ilustración 8 - Subsistemas

El subsistema de gestión de bases de datos es el encargado de comunicar la base de datos con el servidor para ello utiliza JPA y Spring Data (sobre los que se puede ampliar información en Aspectos Teóricos.

Mientras que el de taxista y cliente a su vez se comunican con el servidor utilizando servicios RESTful y como lenguaje de comunicación JSON.

4.4 Diagrama de Clases Preliminar del Análisis

4.4.1 Diagrama de Clases

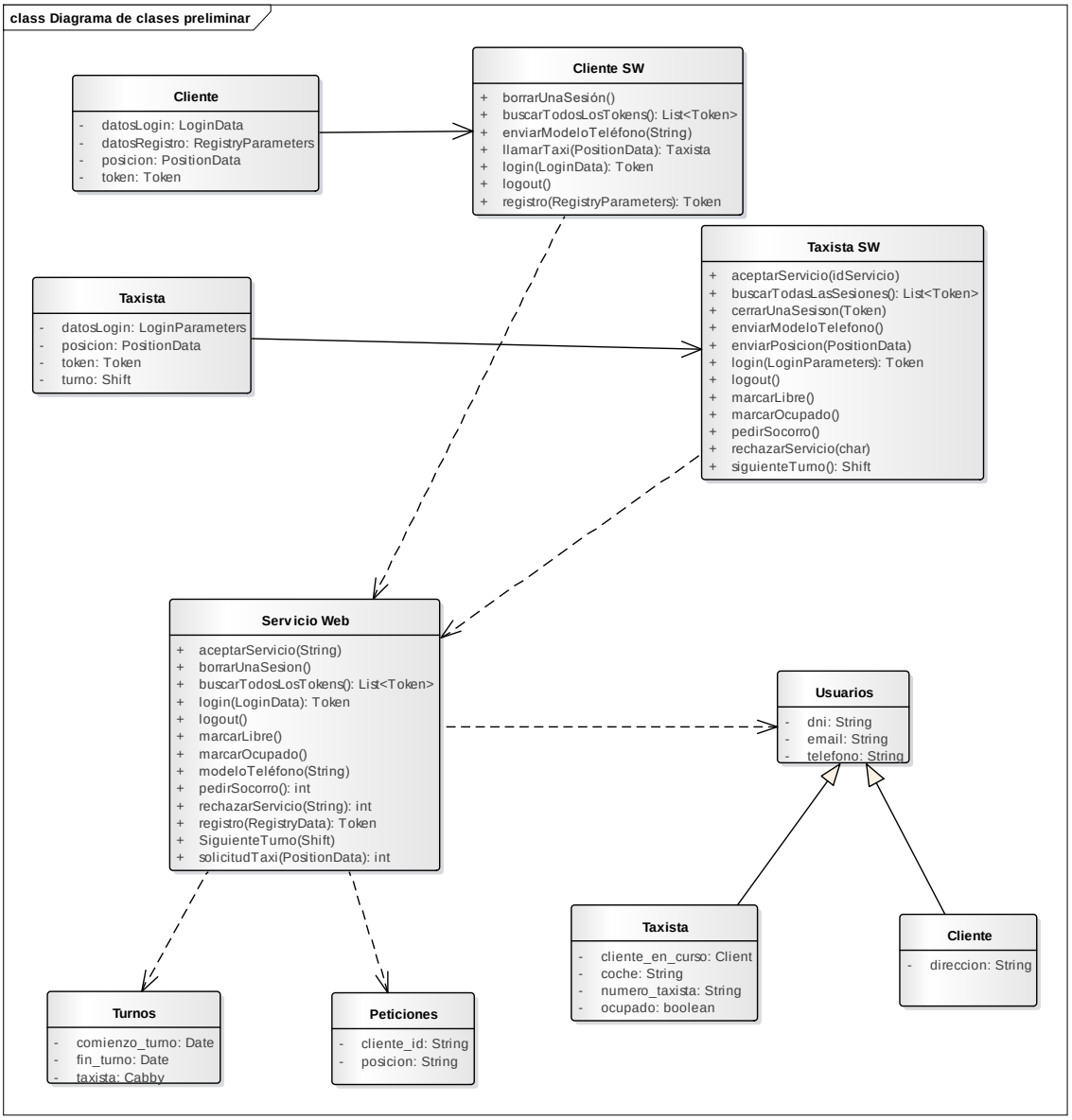


Ilustración 9 – Diagrama de clases preliminar

4.5 Análisis de Casos de Uso y Escenarios

Los casos de uso de "rechazar petición", "aceptar petición", están contemplados en el caso de uso "pedir taxi", ya que en conjunto es más claro.

En el caso de uso de "Cerrar sesiones abiertas" se contempla también el de verlas, hacer un esquema específico para esto sería redundante.

4.5.1 Caso de Uso: Login

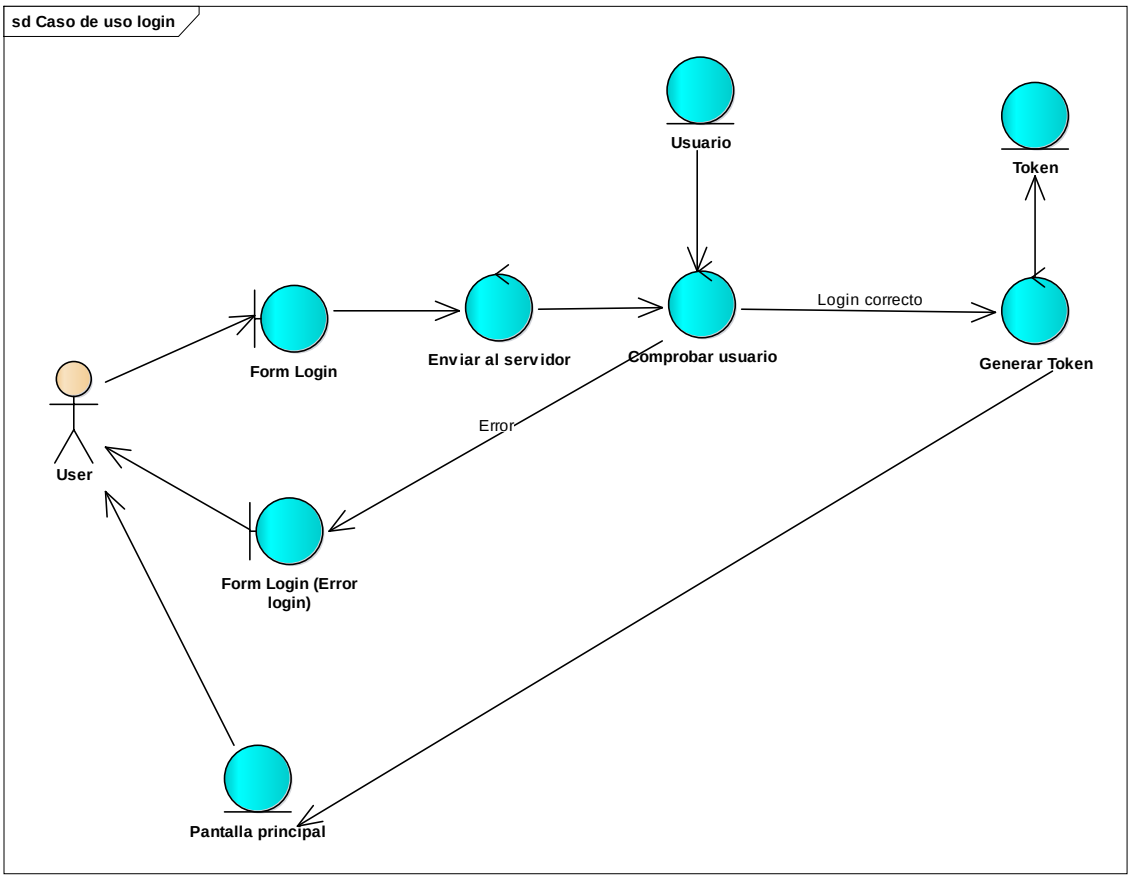


Ilustración 10 – Caso de uso Login

Login	
Precondiciones	No
Poscondiciones	El usuario si es correcto, iniciará sesión en el sistema.
Actores	Taxistas y clientes
Descripción	1. El usuario accederá a la pantalla de Login e introducirá en los campos correspondientes su nombre de usuario y contraseña. 2. El sistema enviará los datos al servidor. 3. Se comprobará que realmente ese usuario existe en la base de datos y si su rol se corresponde. 4. Si el usuario existe se le crea un token para ese dispositivo. 5. Se le muestra la pantalla principal del sistema.
Variaciones (escenarios secundarios)	• **El usuario no existe en el sistema**: Si el usuario no está dado de alta en el sistema, bien por el administrador en caso del taxista o a través del formulario de registro en el de usuario: o Se devolverá un error de usuario no valido y se mostrará la página de Login informando del problema.
Excepciones	o **Si el usuario no existe**: no puede acceder al sistema. o **Si el rol del usuario no se corresponde con la aplicación**: Si un usuario intenta loguearse en la aplicación de taxista o un taxista en la aplicación de cliente con su rol de taxista, el sistema devolverá el mismo error que si el usuario no existe. o **La base de datos no está disponible**: No se puede acceder a los datos del cliente.

4.5.2 Caso de Uso: Registro

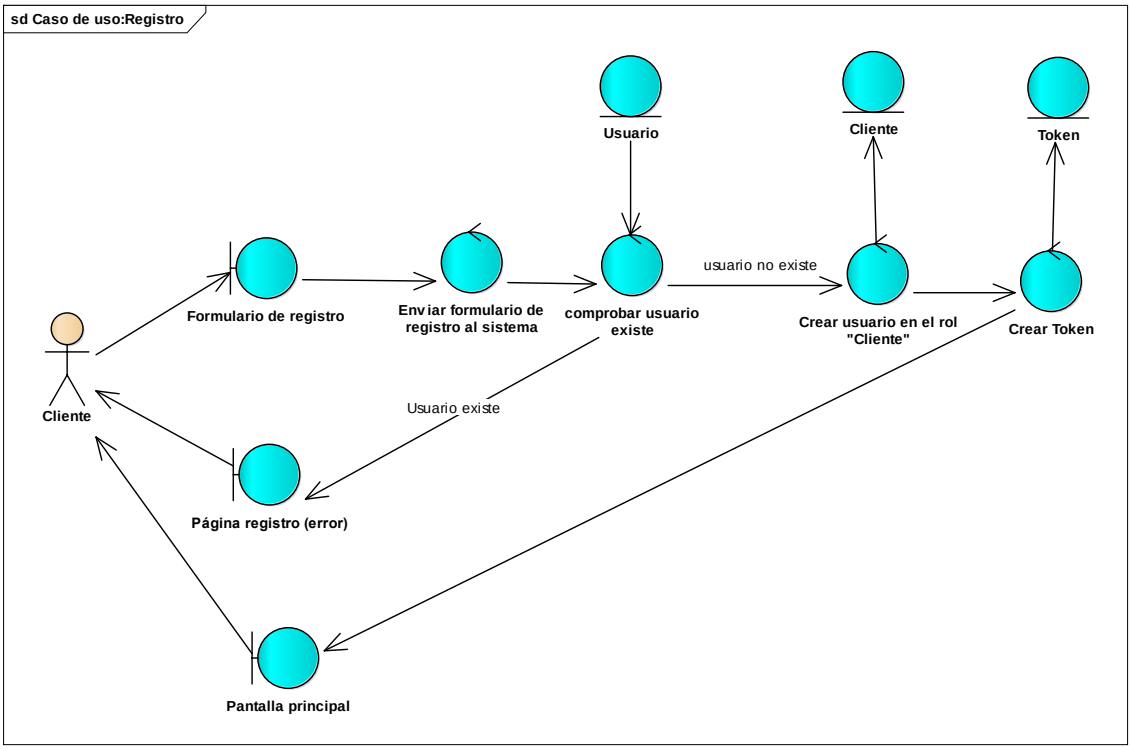


Ilustración 11 – Caso de uso regitro.

Registro en el Sistema	
Precondiciones	No
Poscondiciones	El usuario debe estar validado y con un rol asignado
Actores	Clientes
Descripción	1. El sistema muestra la pantalla de registro. 2. El usuario introducirá al menos los datos obligatorios del retistro. 3. El sistema valida que no exista ya otro usuario registrado con el mismo Login. 4. El sistema registra al usuario y genera su Token correspondiente al dispositivo que está utilizando. 5. Se le muestra la pantalla principal al cliente.
Variaciones (escenarios secundarios)	• **Exista un usuario con ese login**: Se le indica al usuario en el formulario de registro que no es válido el Login.
Excepciones	• **Ya existe el Login:** se muestra error al cliente. • **La base de datos no está disponible**: No se pueden obtener nombres ni contraseñas de usuario ○ Notificar un error asociado al problema encontrado

4.5.3 Caso de uso: Pedir taxi

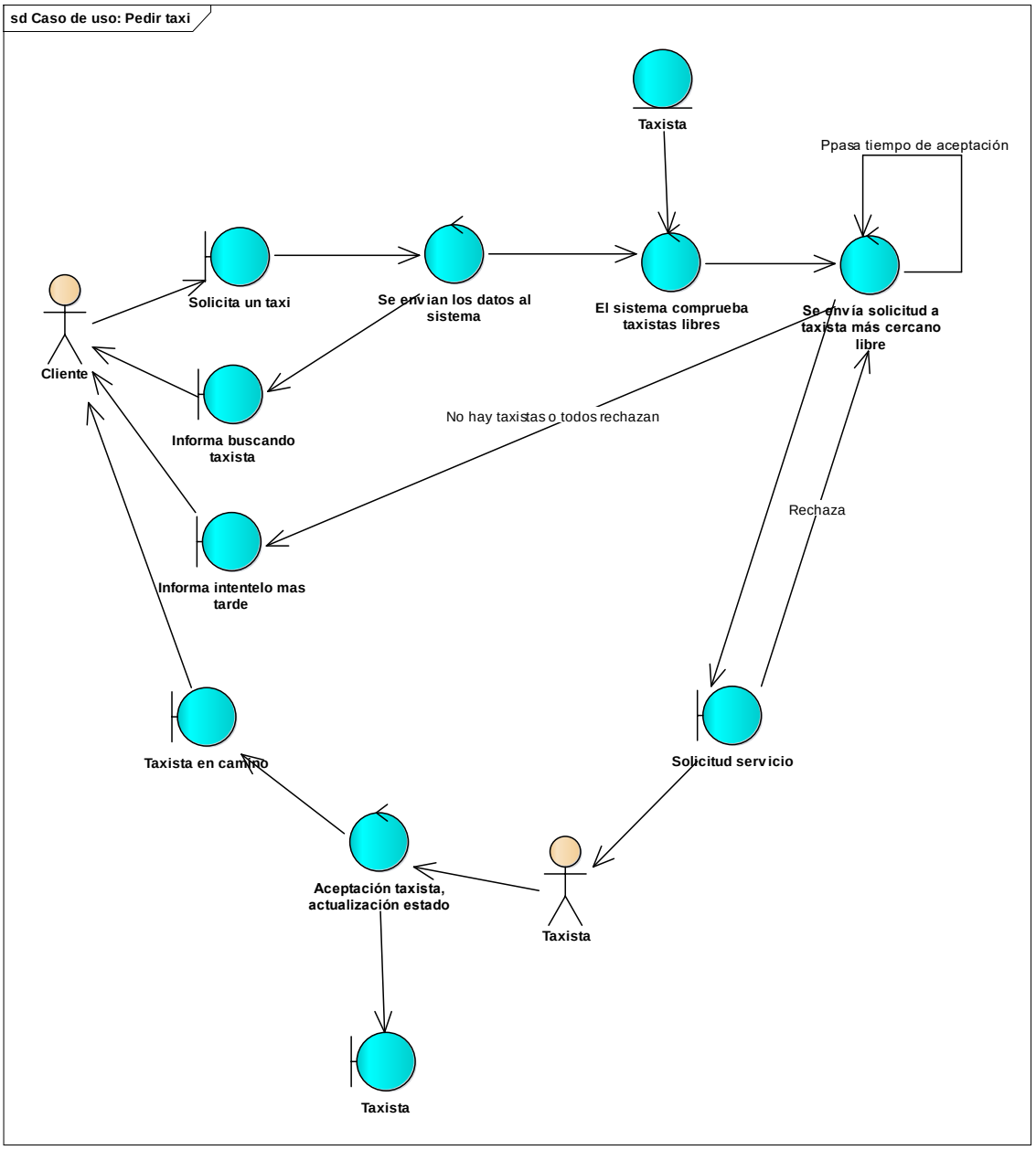


Ilustración 12 – Caso de uso: pedir taxi

Pedir taxi	
Precondiciones	El cliente debe estar logueado en el sistema.
Poscondiciones	Se le asignará un taxi al cliente.
Actores	Taxistas y clientes
Descripción	1. El usuario seleccionará en la pantalla dónde quiere que lo recoja el taxi y solicitará el mismo. 2. Se envían los datos al sistema y se informa al cliente de que su petición está en curso. 3. Se comprueban los taxistas que están libres y se ordenan por cercanía al cliente. 4. Se les envían peticiones a los taxistas por orden de cercanía. 5. Un taxista acepta 6. Se le asigna el viaje al taxista y se pone en estado ocupado. 7. Se le informa al cliente que taxi está en camino y dónde
Variaciones (escenarios secundarios)	• **No hay taxistas libres:** Se informa al cliente de que es imposible su petición en estos momentos que lo intente más tarde. • **Ningún taxista acepta el viaje:** Se informa al cliente de que es imposible su petición en estos momentos que lo intente más tarde.
Excepciones	○ **La base de datos no está disponible**: No se puede acceder a los datos.

4.5.4 Caso de uso: Pedir SOS

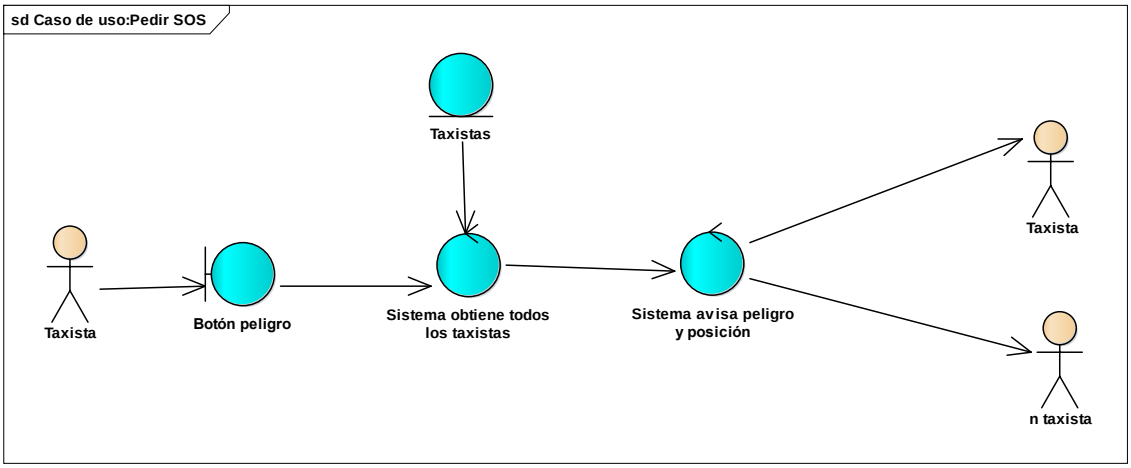


Ilustración 13 – Caso de uso: pedir SOS

Pedir SOS	
Precondiciones	Taxista logueado en el sistema.
Poscondiciones	Todos los taxistas recibirán la posición del taxista en peligro.
Actores	Taxistas
Descripción	1. Taxista pulsa botón de SOS. 2. El sistema busca en la base de datos todos los taxistas. 3. Se envía un mensaje de socorro con la posición al resto de taxistas.
Variaciones (escenarios secundarios)	
Excepciones	○ **La base de datos no está disponible**: No se puede acceder a los datos. ○ **El sistema está caído**: No se pueden acceder a los datos ni enviar las peticiones.

4.5.5 Caso de uso: Cerrar sesiones abiertas

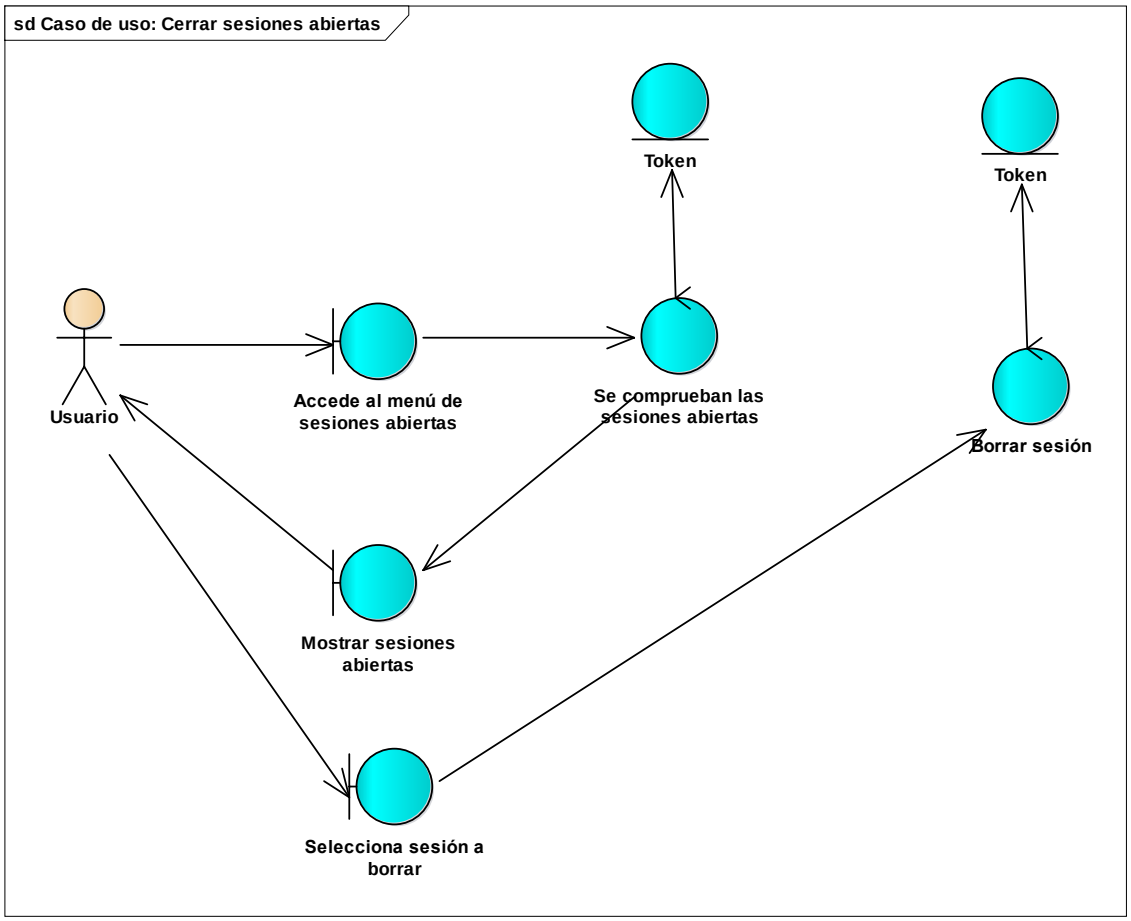


Ilustración 14 – Caso de uso: Cerrar sesiones abiertas

Cerrar sesiones abiertas	
Precondiciones	Usuario logueado en el sistema.
Poscondiciones	Se borrará la sesión del cliente para ese dispositivo en el sistema.
Actores	Usuarios: taxistas y clientes
Descripción	1. Usuario accede al menú de sesiones abiertas. 2. El sistema comprueba las sesiones abiertas 3. Se muestran las sesiones abiertas al usuario. 4. En la tabla de sesiones abiertas el usuario selecciona una sesión y la cierra. 5. El sistema elimina la sesión.
Variaciones (escenarios secundarios)	• **Sólo existe la sesión abierta**: Se muestra la sesión actual.
Excepciones	• **La base de datos no está disponible**: No se puede acceder a los datos ni borrarlos.

4.5.6 Caso de uso: Logout

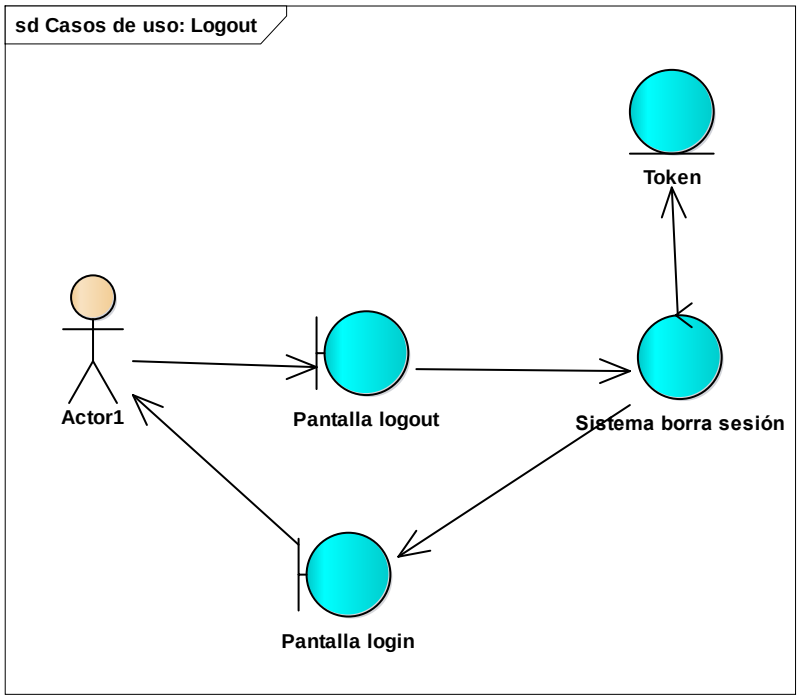


Ilustración 15 – Caso de uso: Logout

Logout	
Precondiciones	Usuario logueado en el sistema.
Poscondiciones	El usuario no estará logueado en el sistema con ese dispositivo.
Actores	Usuarios: taxistas y clientes
Descripción	1. Usuario pulsa el logout. 2. El sistema busca la sesión y la borra. 3. Se le muestra la pantalla de Login al usuario.
Variaciones (escenarios secundarios)	
Excepciones	• **La base de datos no está disponible**: No se puede acceder a los datos ni borrarlos.

4.6 Análisis de Interfaces de Usuario

4.6.1 Descripción de la Interfaz

En el caso de este proyecto va a haber dos aplicaciones móviles diferenciadas, una para el taxista y otra para el cliente, por lo que se va a explicar cada aplicación por separado, ya que tienen pantallas y funcionalidades distintas en muchos casos.

4.6.1.1 Aplicación móvil taxista

En los formularios tanto el de registro como el de Login pueden producirse errores porque falta rellenar el campo, o ya existe el usuario, en tales casos la aplicación indicará donde está el error al usuario, marcando el campo en rojo y con un texto explicativo.

La pantalla de Login, contendrá un formulario para loguearse:

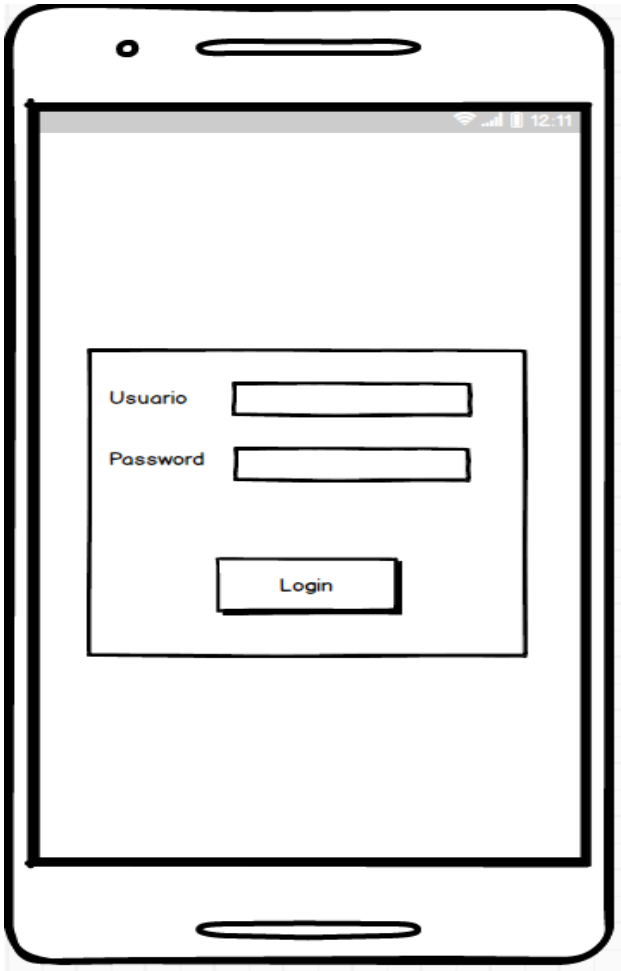


Ilustración 16- Pantalla de login

Una vez logueados en la pantalla principal se encontrarán los botones de ocupado, libre y la información sobre el turno.

Ilustración 17 – Pantalla principal taxista

Las peticiones de taxi irán apareciendo en esta pantalla según se produzcan, desplazando un poco hacia abajo le información horaria y poniendo una barra de scroll en la lista cuando son muchas.

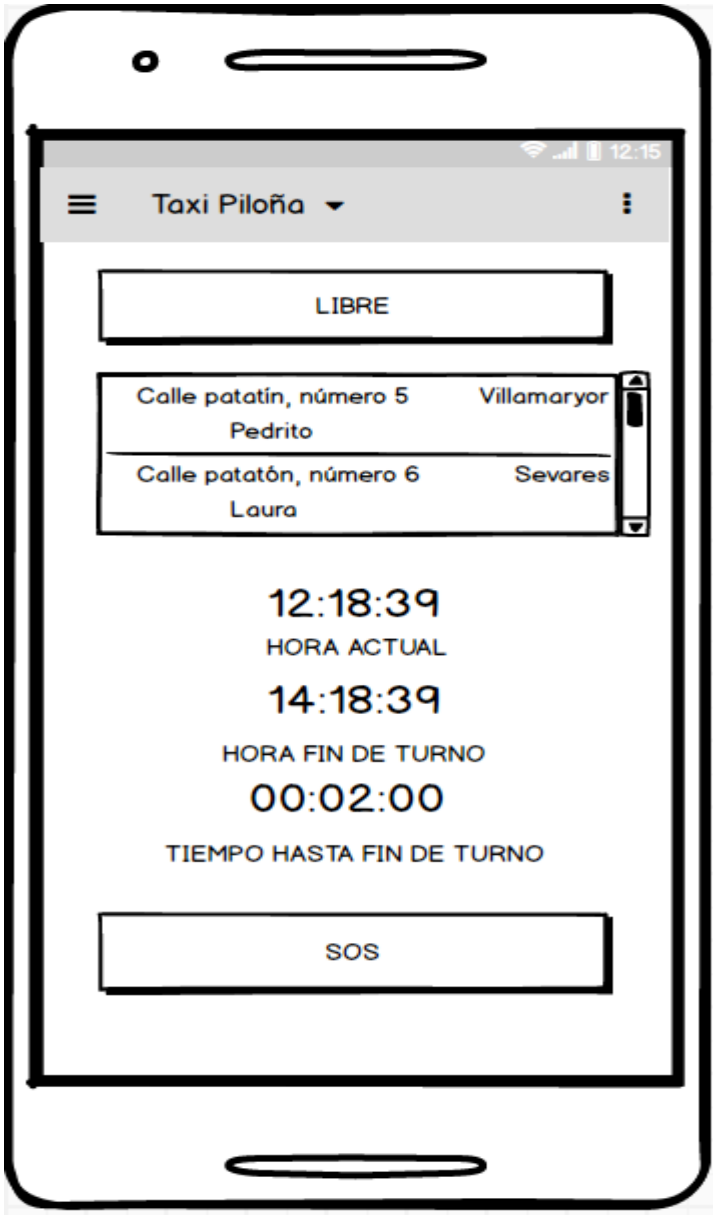


Ilustración 18 – Pantalla principal con peticiones de taxi

En cuanto a las peticiones de taxi, deslizando hacia la izquierda la petición nos aparecerá un botón para acceder a un mapa con la posición del cliente y otro para aceptar la petición, mientras que si se desliza a la derecha aparecerá un botón para rechazarla.

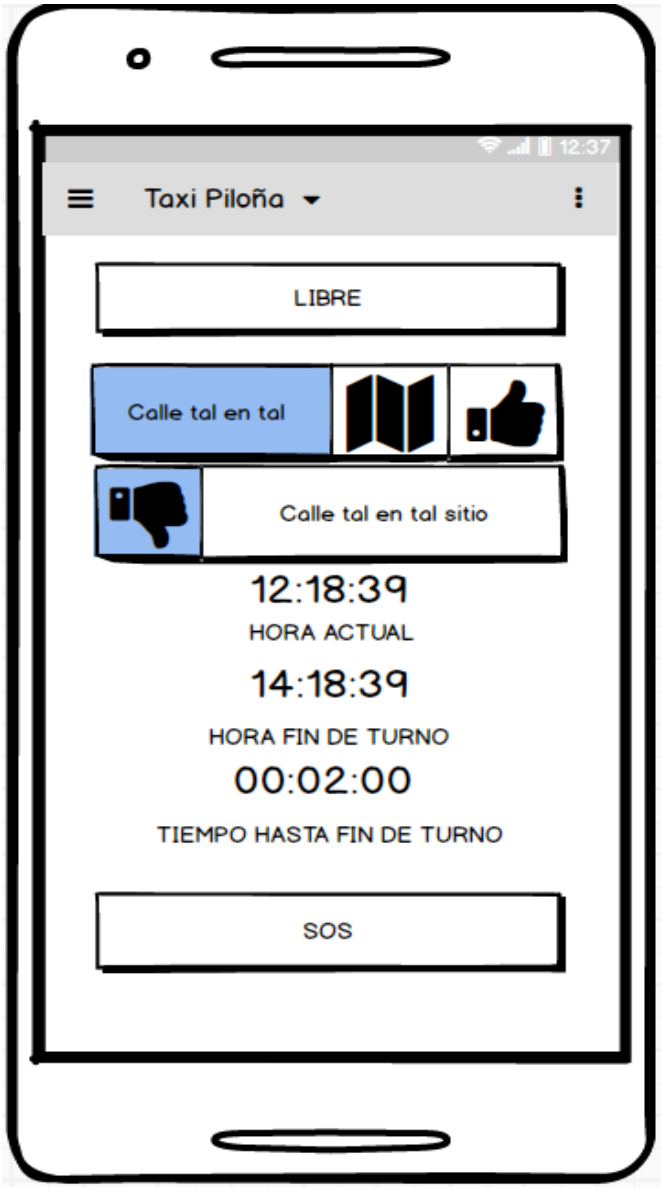


Ilustración 19 – Opciones de las peticiones de taxi

Cuando se acepta una petición el botón de libre pasa a ocupado automáticamente, quedando en pantalla sólo la petición en curso con la opción únicamente de ir al mapa.

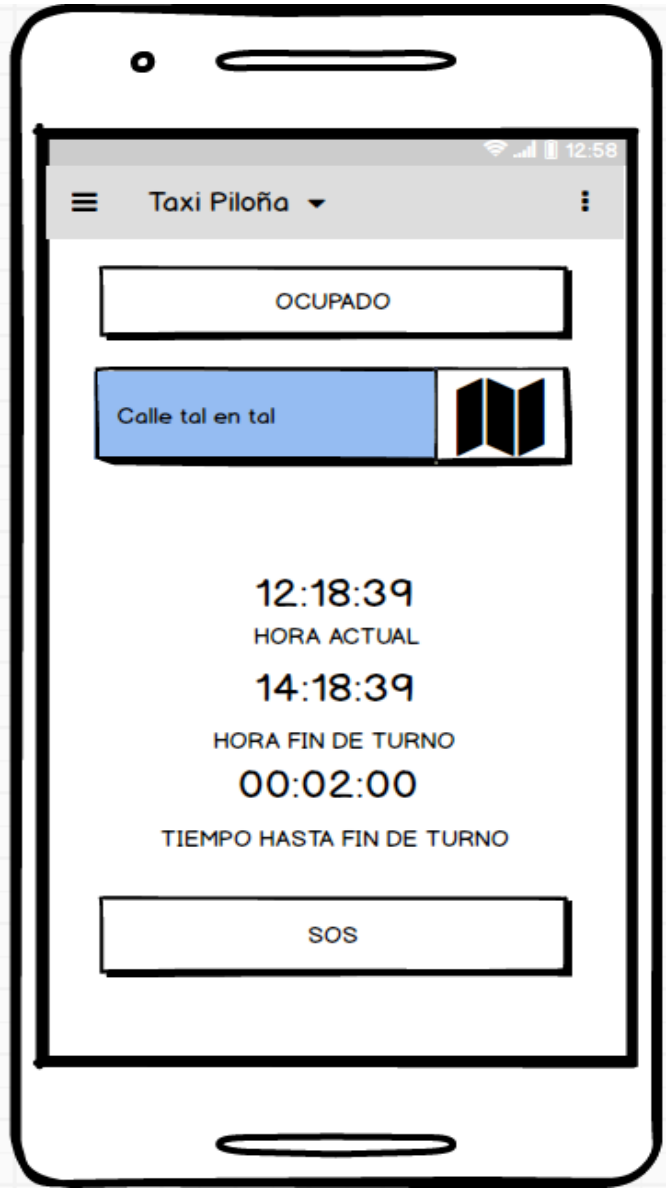


Ilustración 20 – Petición de taxi aceptada

En cuanto al mapa, cuando se pulsa en el icono, va a una pantalla con un mapa centrado en la ubicación del cliente y la propia del taxista. Dando la opción de utilizar el navegador de Google si no deseamos llegar.

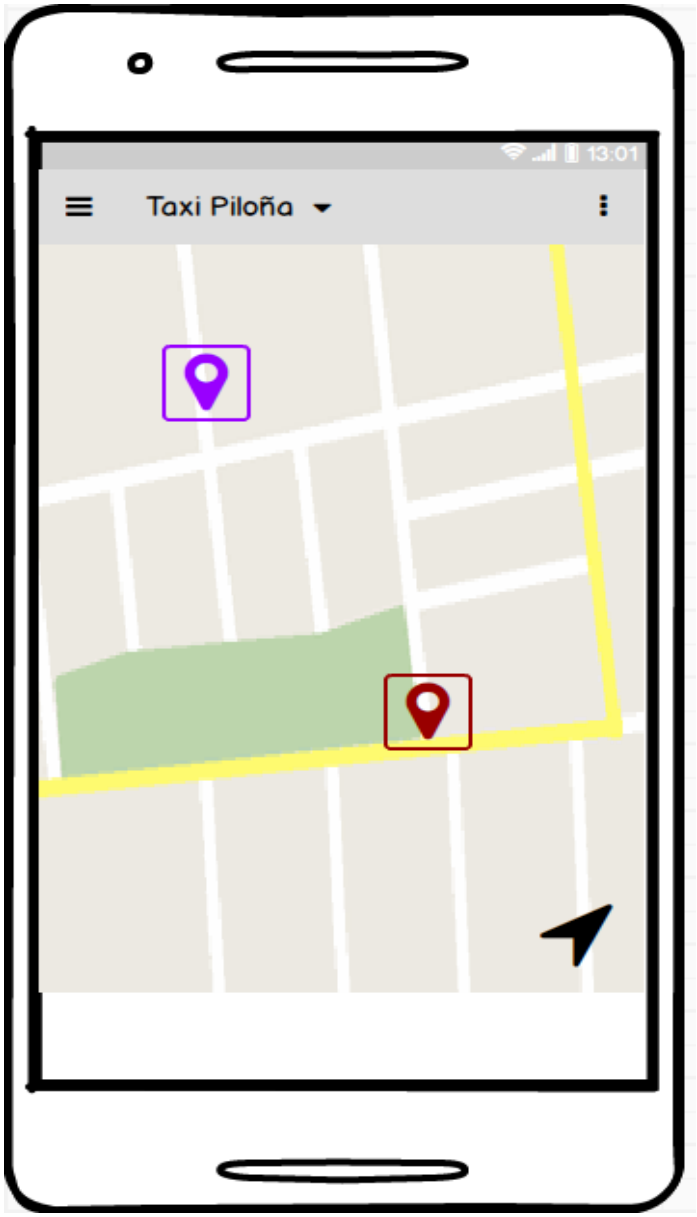


Ilustración 21 - Pantalla mapa

En cuanto al menú desplegable aparecerán las opciones de desloguearse y la de ver las sesiones abiertas.

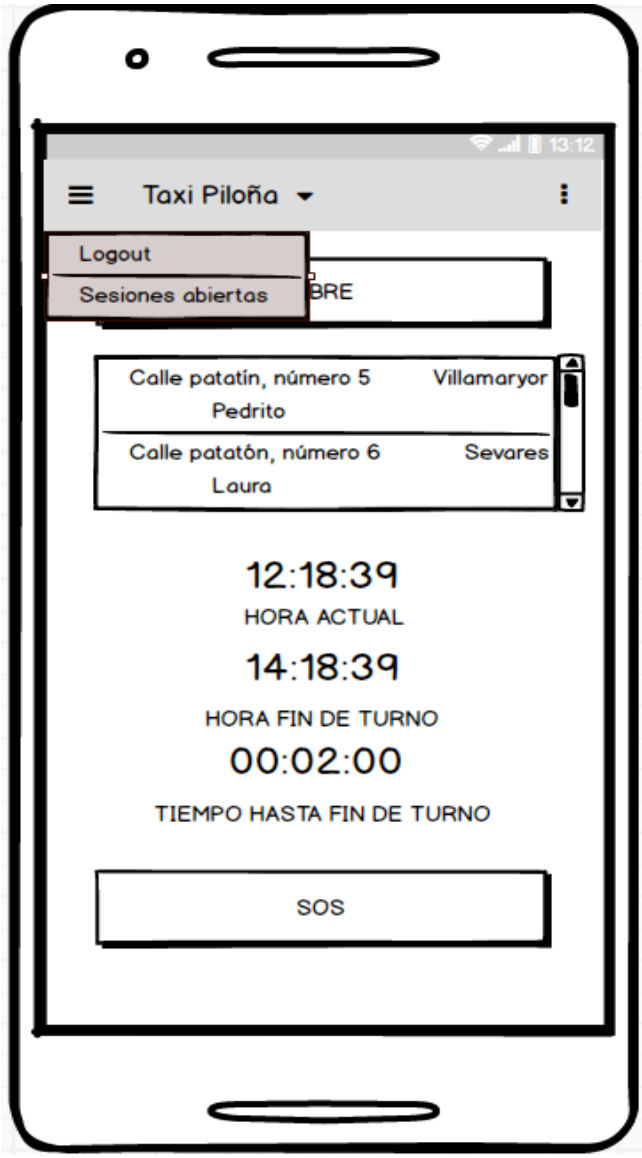


Ilustración 22 – Menú

Y las sesiones abiertas abrirán otra pantalla que nos permitirán con el mismo tipo de menú que el de las peticiones, desplazando a la derecha ver un botón de eliminar que cerrará la sesión en ese dispositivo.

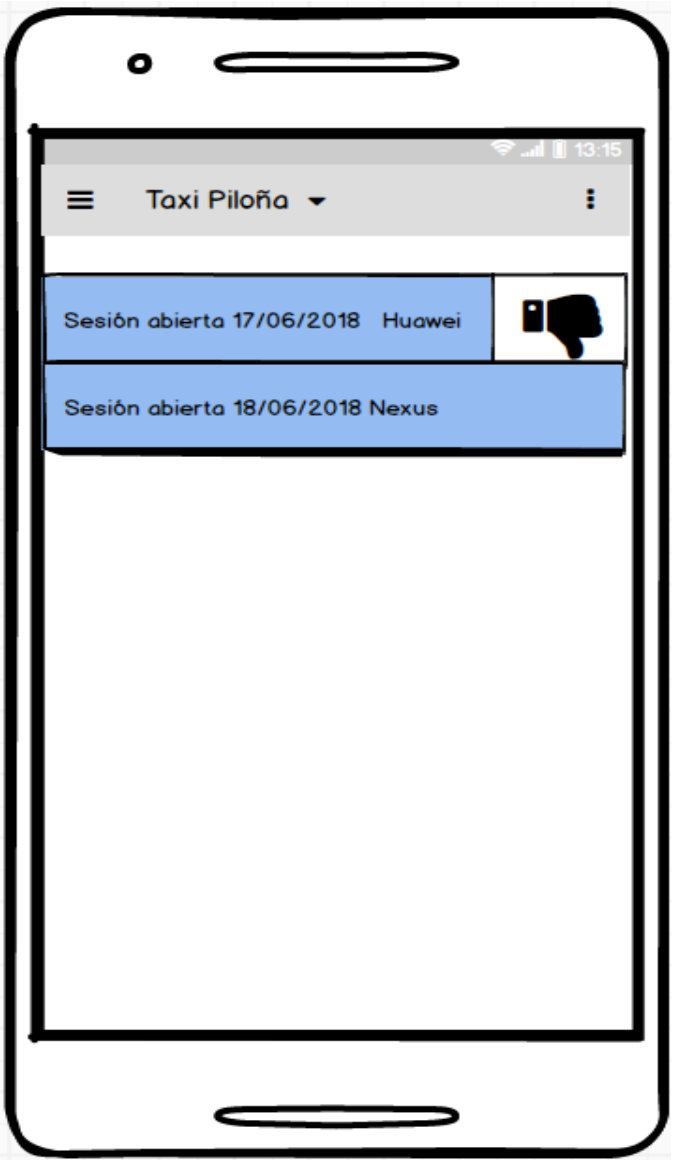


Ilustración 23 – Pantalla sesiones abiertas

4.6.1.2 Aplicación móvil cliente

La aplicación de móvil del cliente será muy similar a la del taxista, sólo que tendrá un link que permitirá a este ir a la pantalla de registro en caso de no estar registrado.

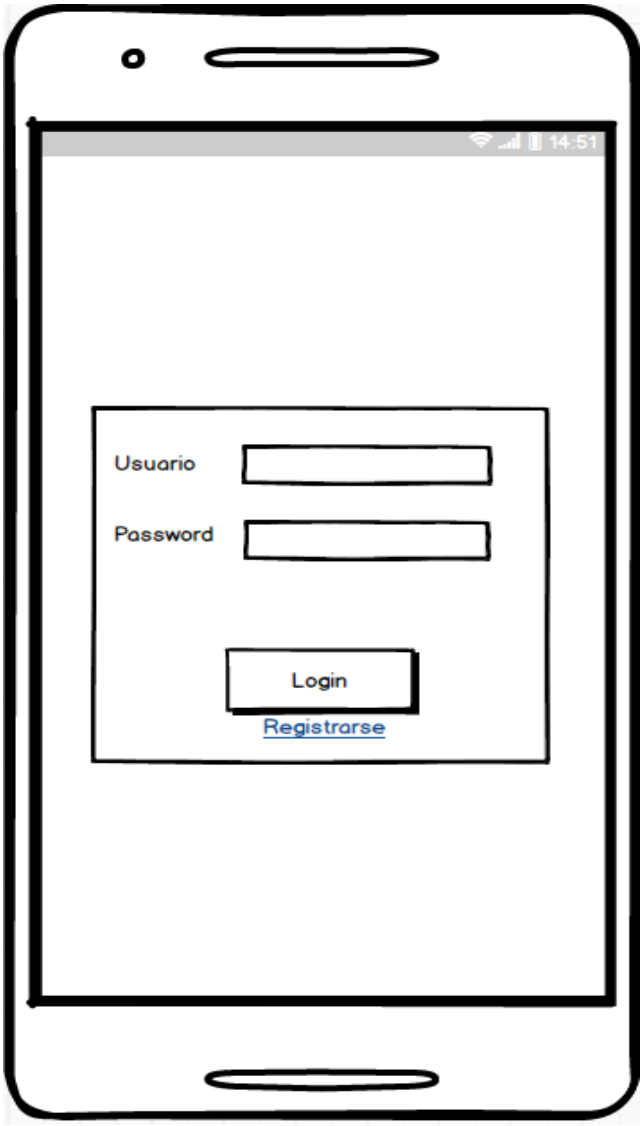


Ilustración 24 -Login cliente

La página principal del cliente será ya en la que pueda pedir el taxi.

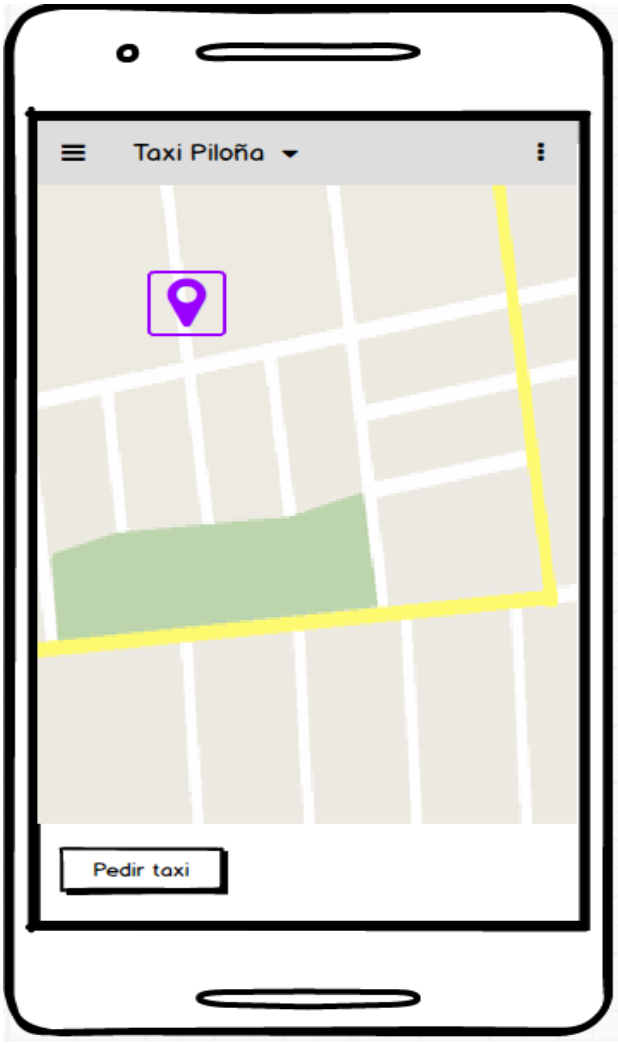


Ilustración 25 – Ventana pedir taxi

Una vez que se solicita el taxi aparecerá abajo en lugar del botón indicando que se está realizando su petición.

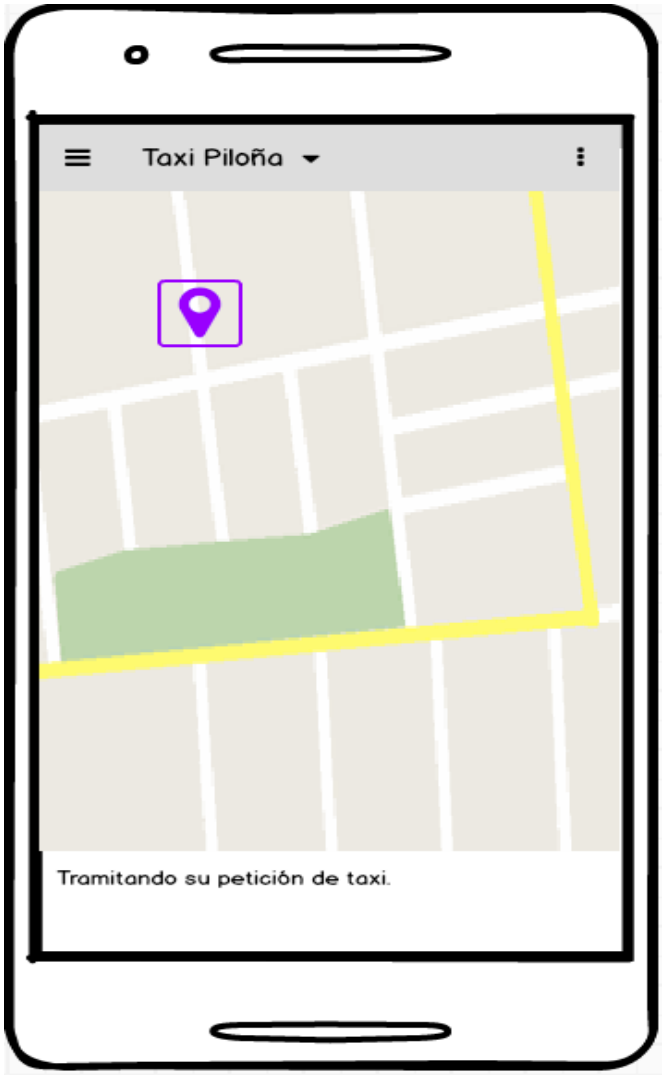


Ilustración 26 – Petición de taxi

Y cuando al final un taxi sea asignado aparecerá que taxi ha sido asignado y su ubicación en el mapa.

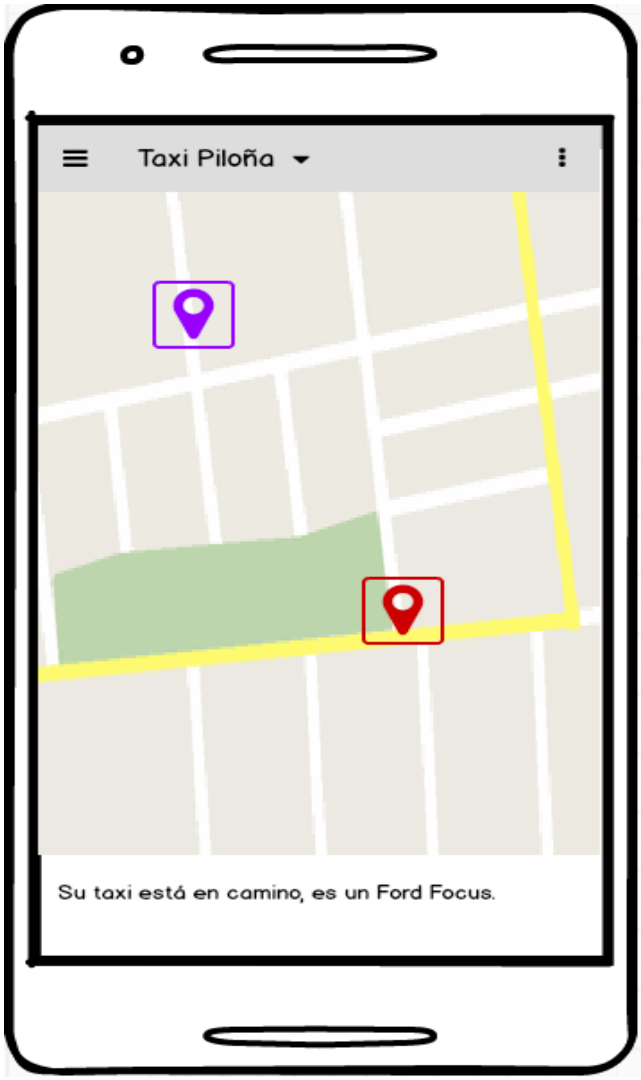


Ilustración 27 – Taxi en camino

En cuanto al menú y la pantalla de sesiones abiertas será igual que la del taxista.

El registro de usuario será un formulario.

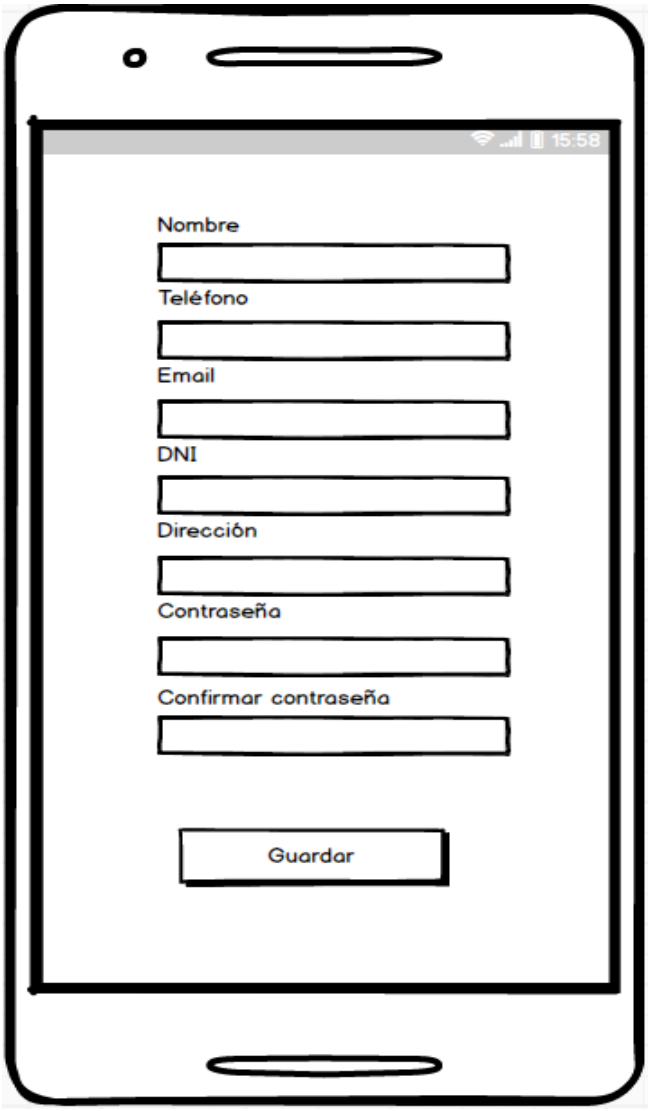


Ilustración 28 - Registro

4.6.2 Diagrama de Navegabilidad

4.6.2.1 Aplicación móvil taxista

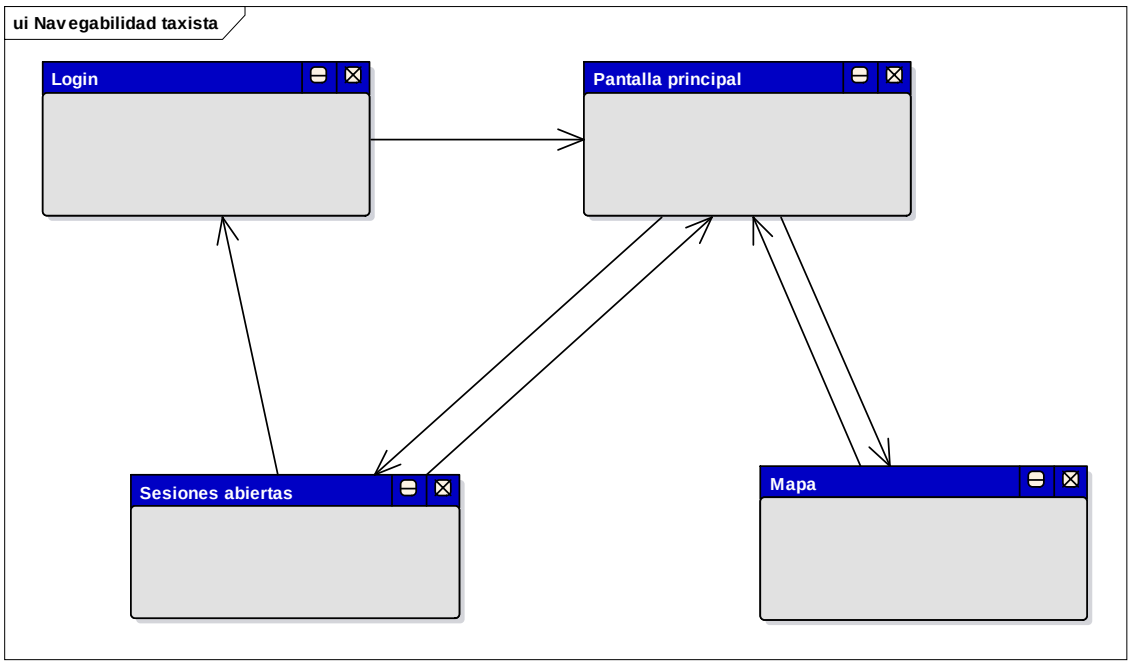


Ilustración 29 – Diagrama de navegabilidad de taxista

4.6.2.2 Aplicación móvil cliente

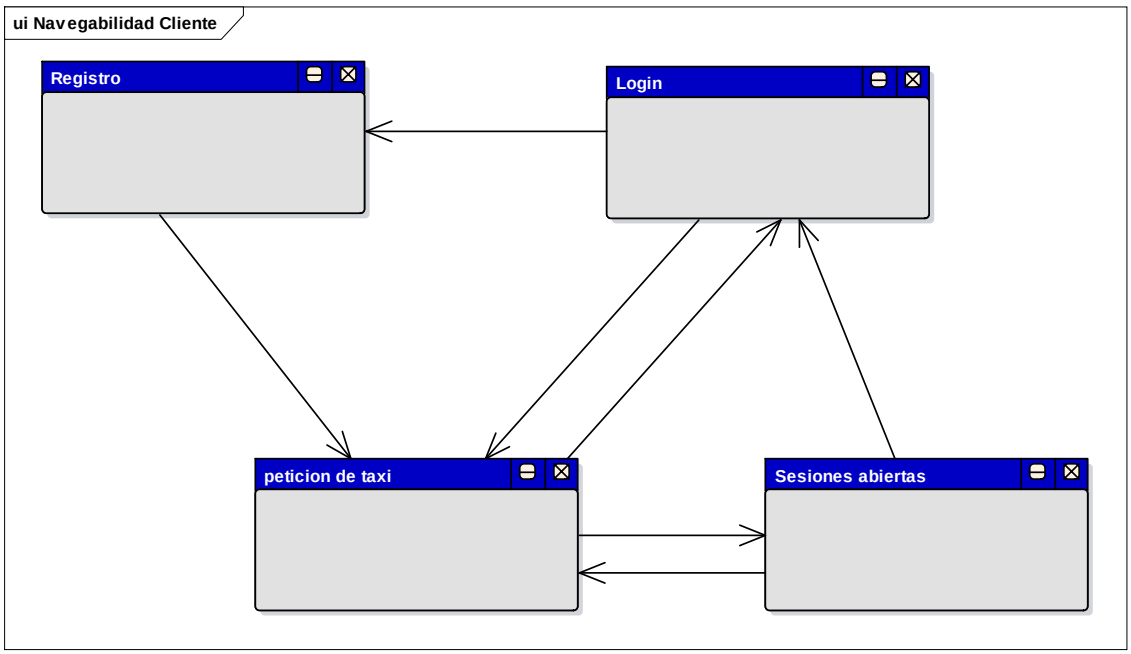


Ilustración 30 – Diagrama de navegabilidad de cliente

4.7 Especificación del Plan de Pruebas

- **Pruebas Unitarias**: Para realizar las pruebas unitarias del servidor se ha utilizado Junit y Mockito, lo que ha permitido automatizarlas.

  Las de las aplicaciones móviles:

Caso de Uso 1: Registrase	
Prueba	**Resultado Esperado**
Añadir un usuario no existente	El sistema posee un usuario más
Prueba	**Resultado Esperado**
Añadir un usuario que ya existe	El sistema no posee un usuario más y se muestra un dialogo notificándolo
Prueba	**Resultado Esperado**
Cancelar la Operación	El sistema permanece sin cambios.

Caso de Uso 1: Loguearse	
Prueba	**Resultado Esperado**
Loguearse en la aplicación de taxista como cliente	No permite el logueo y emite un mensaje de error.
Prueba	**Resultado Esperado**
Loguearse en la aplicación de cliente como taxista	No permite el logueo y emite un mensaje de error.
Prueba	**Resultado Esperado**
Loguearse en la aplicación de cliente con un cliente existente	El sistema genera el Token para ese usuario y ese dispositivo y aparece la pantalla de inicio.
Prueba	**Resultado Esperado**
Loguearse en la aplicación de taxista con un taxista existente	El sistema genera el Token para ese usuario y ese dispositivo y aparece la pantalla de inicio.
Prueba	**Resultado Esperado**
Loguearse en la aplicación de cliente con un usuario que no existe	No permite el logueo y emite un mensaje de error.
Prueba	**Resultado Esperado**
Loguearse en la aplicación de taxista con un usuario que no existe	No permite el logueo y emite un mensaje de error.

Caso de Uso 1: Pedir un taxi	
Prueba	**Resultado Esperado**
Taxistas libres, uno con coordenadas muy cercanas al cliente y otras lejanas	Le llega primero la petición al taxista que las tiene cercanas
Prueba	**Resultado Esperado**

Petición de taxi con taxistas libres, coordenadas cercanas ocupado y el resto libres.	Se salta la petición al que está ocupado.
Prueba	**Resultado Esperado**
Petición taxi todos los taxistas ocupados.	Le indica al usuario que lo intente más tarde.
Prueba	**Resultado Esperado**
Comprobar en mapa de taxista ubicación de cliente en petición.	Se corresponde con las coordenadas del cliente.
Prueba	**Resultado Esperado**
Rechazar petición	Si todos lo rechazan salta mensaje al cliente.
Prueba	**Resultado Esperado**
Aceptar petición	Aparece correctamente en el mapa del cliente el taxista y los datos de quién le recoge.
Prueba	**Resultado Esperado**
Petición escogiendo en el mapa un punto que no es en el que estamos posicionados	Llega la petición al taxista para las coordenadas escogidas.
Prueba	**Resultado Esperado**
Petición aceptada	Comprobamos como se mueve el taxista en el mapa.

Caso de Uso 1: Ver sesiones abiertas	
Prueba	**Resultado Esperado**
Loguearse en varios dispositivos	Aparecen en el listado todas las sesiones abiertas.
Prueba	**Resultado Esperado**
Desloguearse en un dispositivo.	Ya no aparece en la lista.
Prueba	**Resultado Esperado**
Borrar un dispositivo de la lista.	El dispositivo no tiene la sesión abierta y ya no aparece en el listado de sesiones abiertas.

- **Pruebas de Integración**: Si aún no disponen de un servidor propio, se desplegará el servidor en el servidor que tengo asignado en el máster y se comprobará que funcionan correctamente las aplicaciones, las comunicaciones con las bases de datos, los test se pasan correctamente.
- **Pruebas del sistema**: Estas pruebas consistirán en una mezcla de las dos antecesoras a este punto.
- **Pruebas de Usabilidad**: Las pruebas de usabilidad las realizaran los propios taxistas y clientes habituales, para hacer que las aplicaciones sean lo más intuitivas y fáciles de utilizar posibles. Para ello se realizará un listado de acciones a realizar en ambas aplicaciones y se comprobará como se desenvuelven. También en la fase de test, en la que van a tener los prototipos se estudiara la facilidad de uso.

Capítulo 5. Diseño del Sistema

5.1 Arquitectura del Sistema

5.1.1 Diagramas de Paquetes

5.1.1.1 Diagrama de paquetes del servidor

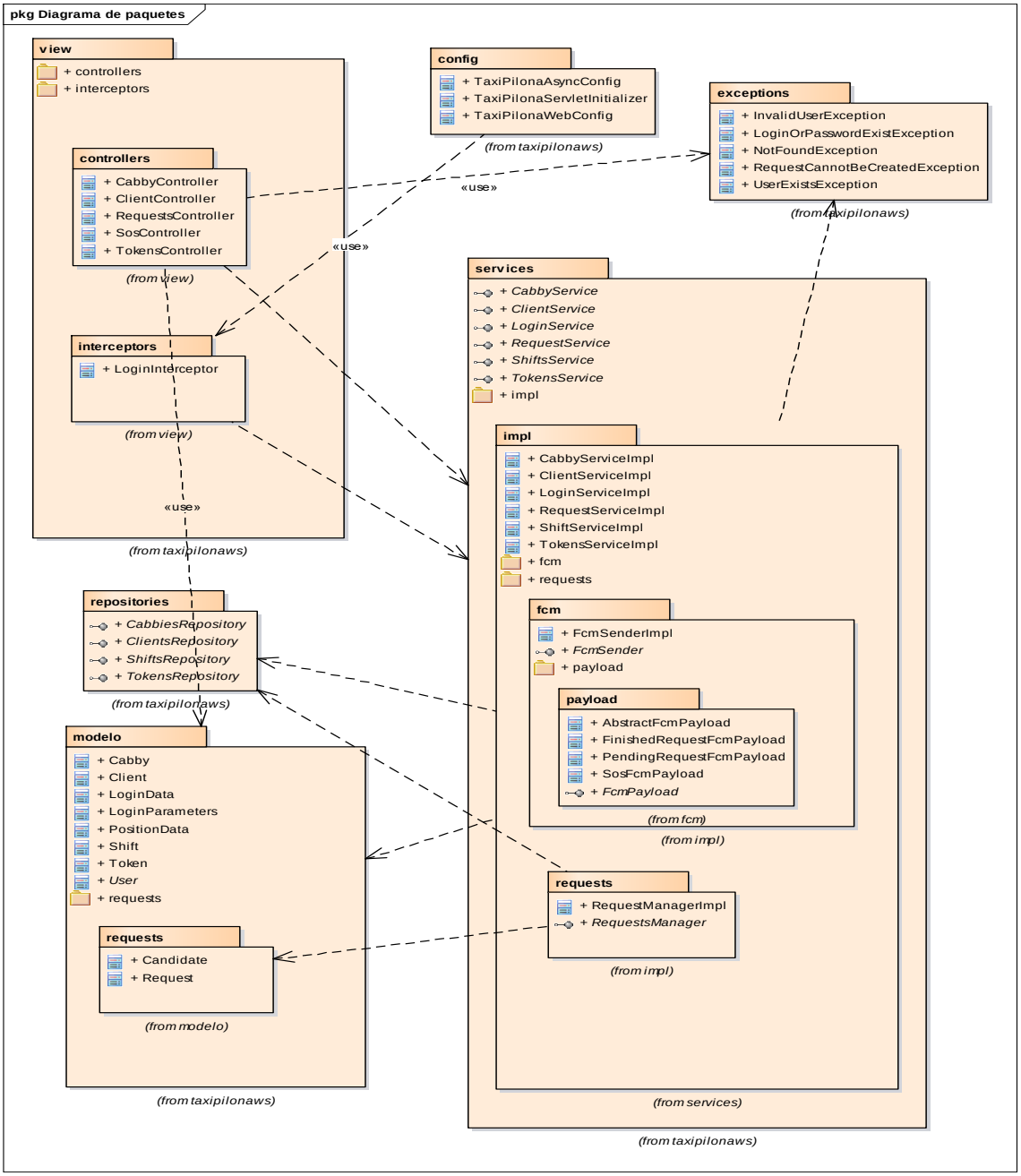


Ilustración 31 – Diagrama de paquetes del servidor

5.1.1.1.1 Paquete View

En este paquete tenemos todos los elementos de la aplicación responsables de implementar la capa de presentación. Estos elementos serán los responsables de interactuar con los clientes a través del servicio web.

Estos componentes se dividen en dos bloques, implementados en los paquetes hijos. Controladores e Interceptores.

5.1.1.1.1.1 Paquete Controladores

Los controladores son los responsables de generar las URLs del servicio web a implementar, y definir el código que se ejecutará cada vez que el usuario acceda a una de estas URLs.

Constituirán por tanto el punto de entrada a la aplicación a través de los métodos del servicio web.

5.1.1.1.1.2 Paquete Interceptores

Los interceptores permiten la ejecución de código antes de la invocación al método principal para cada URL que estará definido en el controlador correspondiente.

En nuestro caso se usa un interceptor para la validación del Login de usuario, realizada en cada petición al servicio a través de un token que se envía en las cabeceras HTTP.

5.1.1.1.2 Paquete config

Clases de configuración de Spring. Las clases de configuración de Spring permiten definir la configuración básica del framework, como por ejemplo la definición de los interceptores que deben asociarse a cada URL.

No toda la configuración del servicio se realiza desde clases en este paquete. Parte se encuentra externalizada en un fichero properties para facilitar la edición sin necesidad de recompilar la aplicación de aquellos parámetros de configuración más susceptibles de requerir ser cambiados con el tiempo, o al implantar la aplicación.

5.1.1.1.3 Paquete exceptions

Excepciones con todos los errores de negocio que se pueden disparar durante el uso del servicio.

5.1.1.1.4 Paquete services

En este paquete tenemos todos los componentes de la capa de negocio de la aplicación. En el paquete raíz tenemos las interfaces, que serán los únicos componentes conocidos de esta capa por las capas superiores (Presentación) para reducir el acoplamiento.

Las asociaciones entre las clases de la capa superior (Presentación) y esta se realizarán mediante inyección de dependencias gracias al framework de Spring.

5.1.1.1.4.1 Paquete impl

En este paquete tenemos las implementaciones de todas las interfaces definidas en el paquete raíz services. Estas implementaciones permanecen ocultas para las capas superiores.

Dentro de este paquete tenemos subpaquetes donde definimos clases para la implementación de determinadas acciones de negocio excepcionalmente complejas, como solicitudes de taxi o envio de mensajes a través de Firebase Cloud Messaging.

5.1.1.1.4.1.1 Paquete fcm

Clases encargadas del envío de peticiones a través de Firebase Cloud Messaging. Se dividen entre la clase encargada de realizar el envío y las que definen los distintos tipos de paquetes susceptibles de ser enviados a través de esta plataforma, incluidos en un paquete hijo.

5.1.1.1.4.1.1.1 Paquete payload

Contiene las clases que implementan los paquetes a enviar a través de Firebase Cloud Messaging.

5.1.1.1.4.2 Paquete requests

Componentes relacionados con la petición de taxi por parte de los clientes. En estas clases se implementa el algoritmo de selección de taxista, empleando el servicio Google Distance Matrix para evaluar la cercanía de cada taxista libre.

5.1.1.1.5 Paquete repositories

Los elementos de este paquete permiten implementar el acceso a la base de datos. Este paquete estará formado exclusivamente por interfaces, ya que las interfaces a estos son proporcionadas automáticamente por Spring Data JPA.

5.1.1.1.6 Paquete del modelo

En el modelo se definen las clases que representan las entidades de la aplicación, y contienen la lógica de negocio relacionada con dichas entidades. Estas clases están mapeadas a las tablas de la base de datos usando en framework JPA.

5.1.1.2 Diagrama de paquetes de la aplicación de cliente

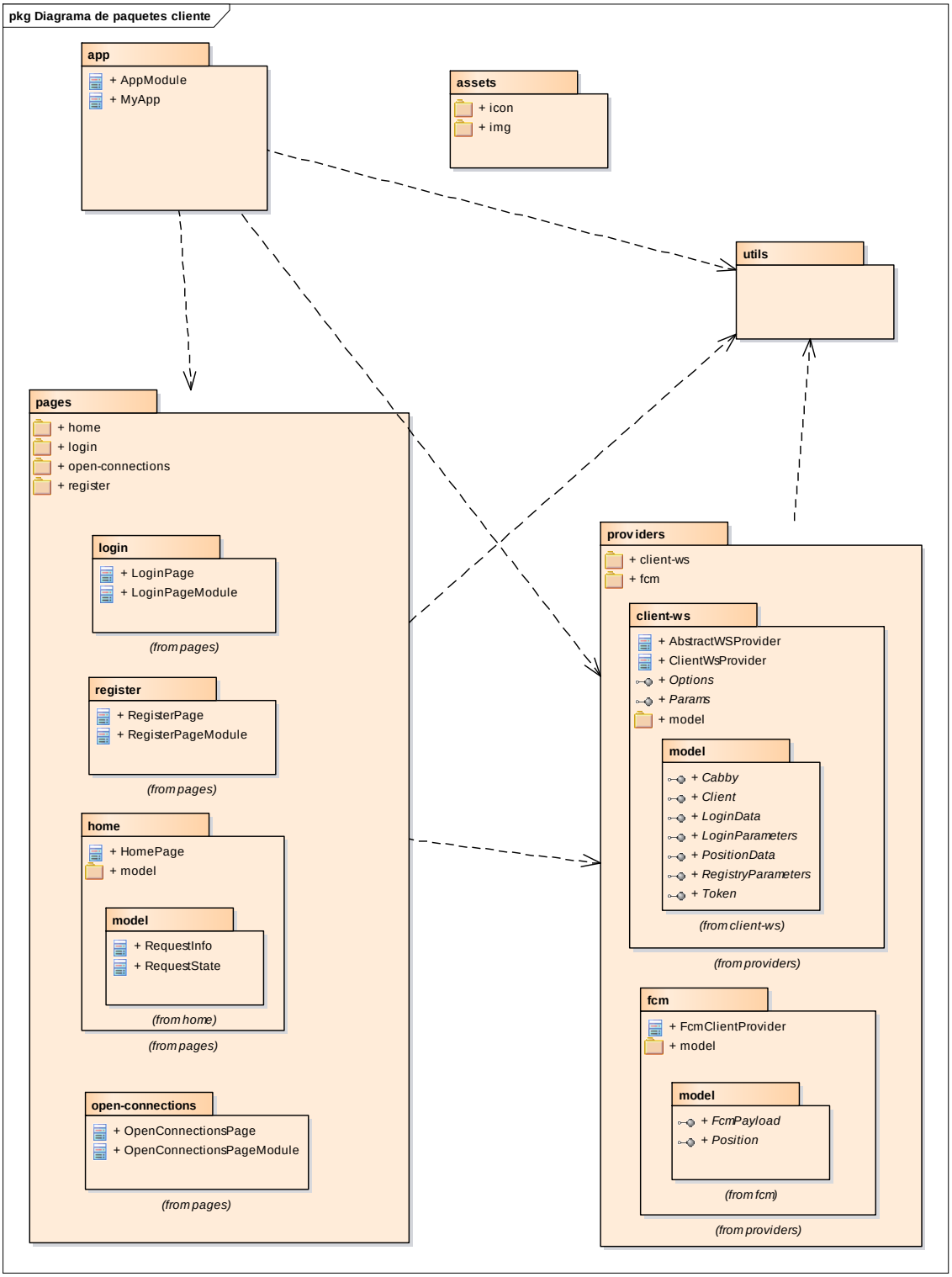


Ilustración 32 – Diagrama de paquetes del cliente

5.1.1.2.1 Paquete app

Este es el paquete raíz del proyecto. En el se incluyen los elementos que constituyen el punto de entrada a la aplicación. En Ionic este punto de entrada estará formado por el módulo raíz (app) y la primera vista de la aplicación (app).

5.1.1.2.2 Paquete page

En este paquete se encuentra la definición de todas las páginas de la aplicación. Los distintos recursos que forman cada página se encuentran a su vez incluidos en el paquete respectivo de cada una. Estos son home, Login, open-connections y register.

Todos estos paquetes incluirán el controlador de la página, la plantilla que representa su interfaz y el estilo de la misma. En algunos casos también se incluye un paquete modelo con las clases de modelo propias de esa pantalla.

5.1.1.2.3 Paquete providers

Los providers constituyen la fuente de información común de todos los componentes visuales (paginas) de la aplicación. Cada provider se encuentra en un subpaquete de este.

5.1.1.2.3.1 Paquete client-ws

Este paquete contiene el código que permite interactuar con el servicio web rest. Esta formado por la clase que define los métodos de acceso y las interfaces que definen los objetos que se envían y reciben desde el servicio web.

5.1.1.2.3.2 Paquete fcm

Es el paquete que contiene los componentes responsables de recibir las notificaciones push enviadas por el servicio rest a través de Firebase Cloud Messaging, y las hace disponibles para el resto de la aplicación.

Contiene la clase responsable de interactuar con Firebase Cloud Messaging y las interfaces donde se definen los objetos que llegarán a la aplicación a través de estas notificaciones.

5.1.1.2.4 Paquete utils

Este paquete contiene clases que proporcionan herramientas cuyo uso es transversal a todo el proyecto, como son las constantes de configuración.

5.1.1.2.5 Paquete assets

Recursos empleados en la interfaz de la aplicación que no están formados por código fuente.

5.1.1.3 Diagrama de paquetes de la aplicación de taxista

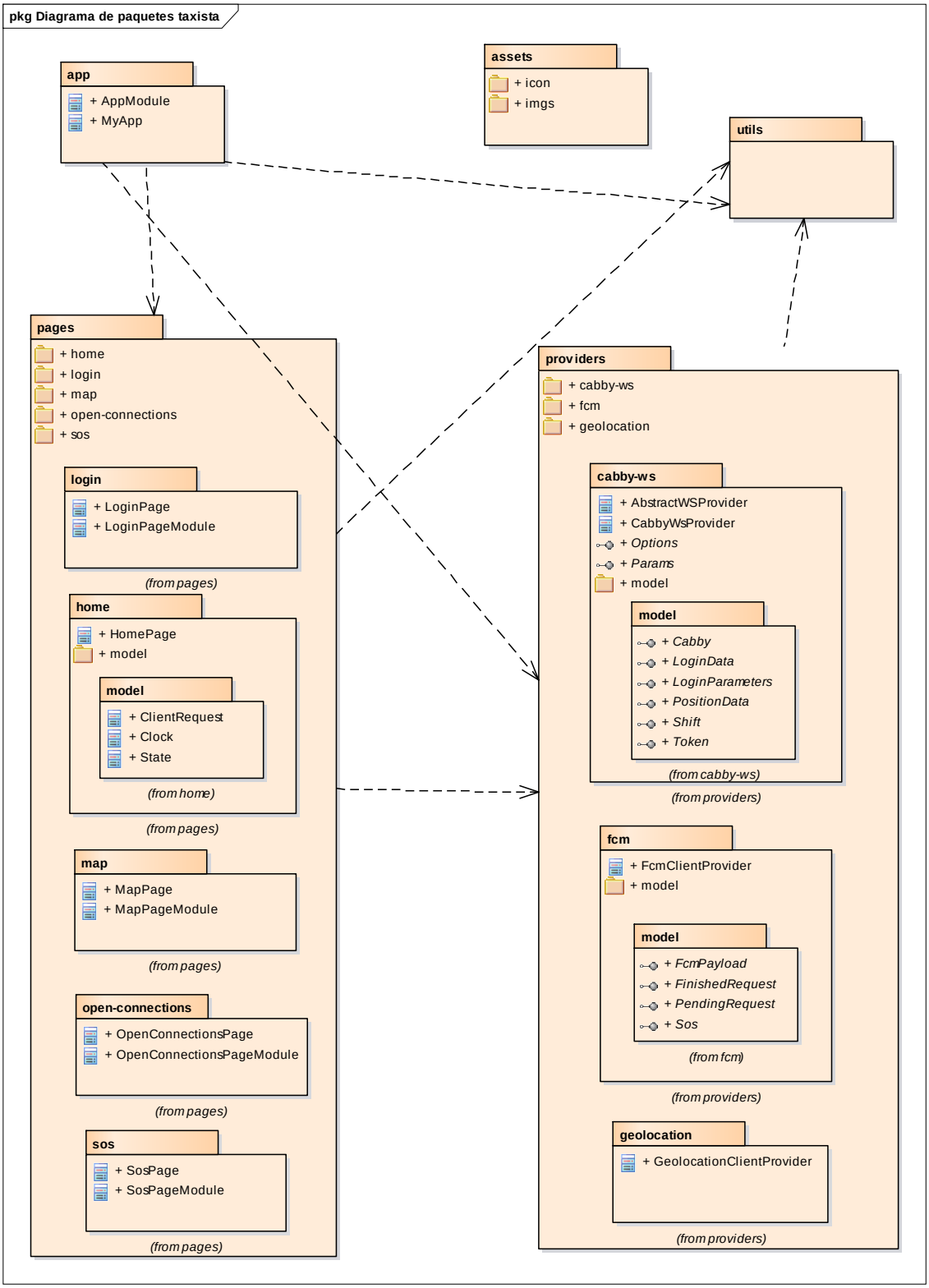


Ilustración 33 – Diagrama de paquetes taxista

Los paquetes de esta aplicación tienen el mismo cometido que los de la anterior.

5.1.2 Diagramas de Componentes

Los diagramas de componentes para una mejor comprensión de este, vamos a dividirlo en servidor, cliente y taxista.

5.1.2.1 Diagramas de componentes del servidor

Este apartado vamos a subdividirlo por controlador ya que serán diagramas más claros.

5.1.2.1.1 Diagrama de componentes: taxista

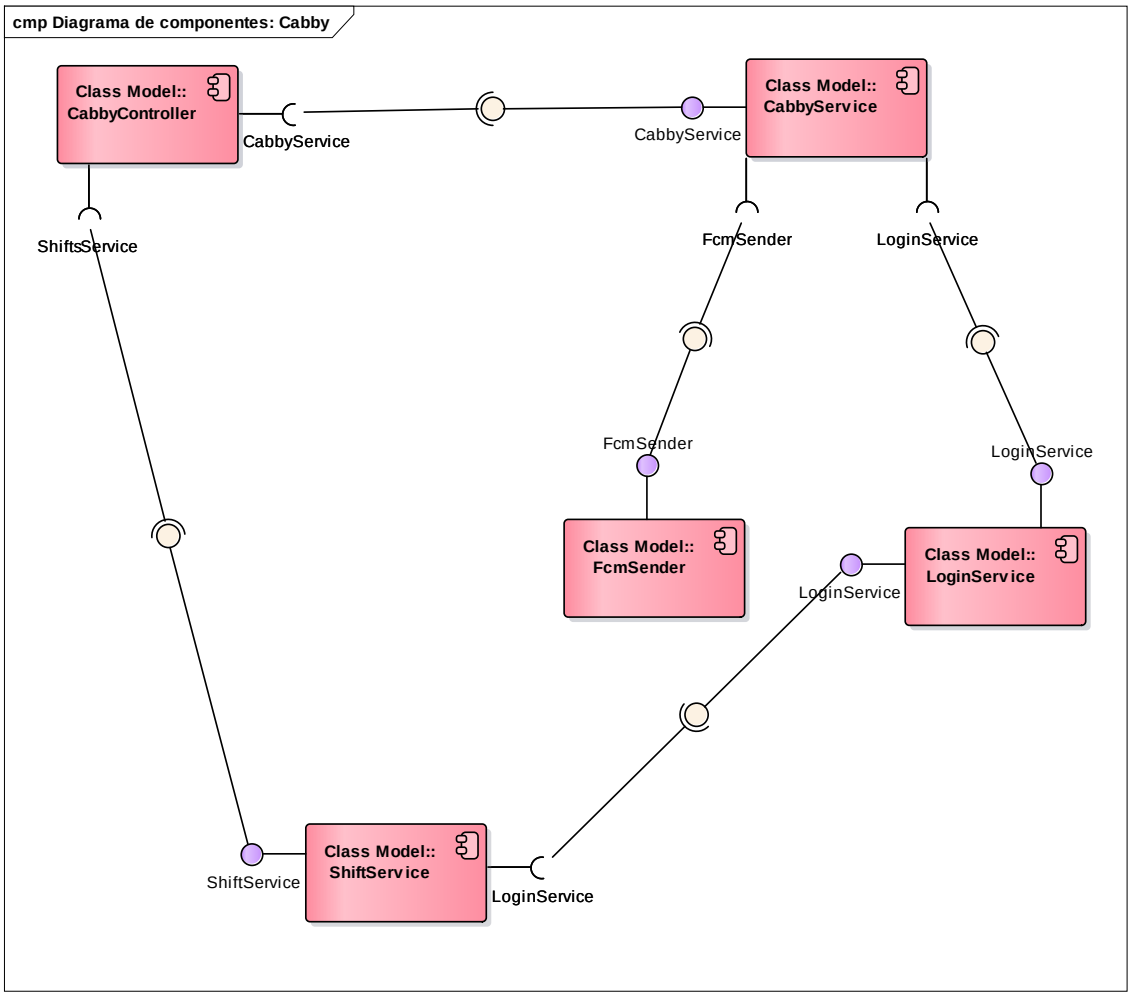


Ilustración 34 – Diagrama de componentes de taxista

5.1.2.1.2 Diagrama de componentes: Cliente

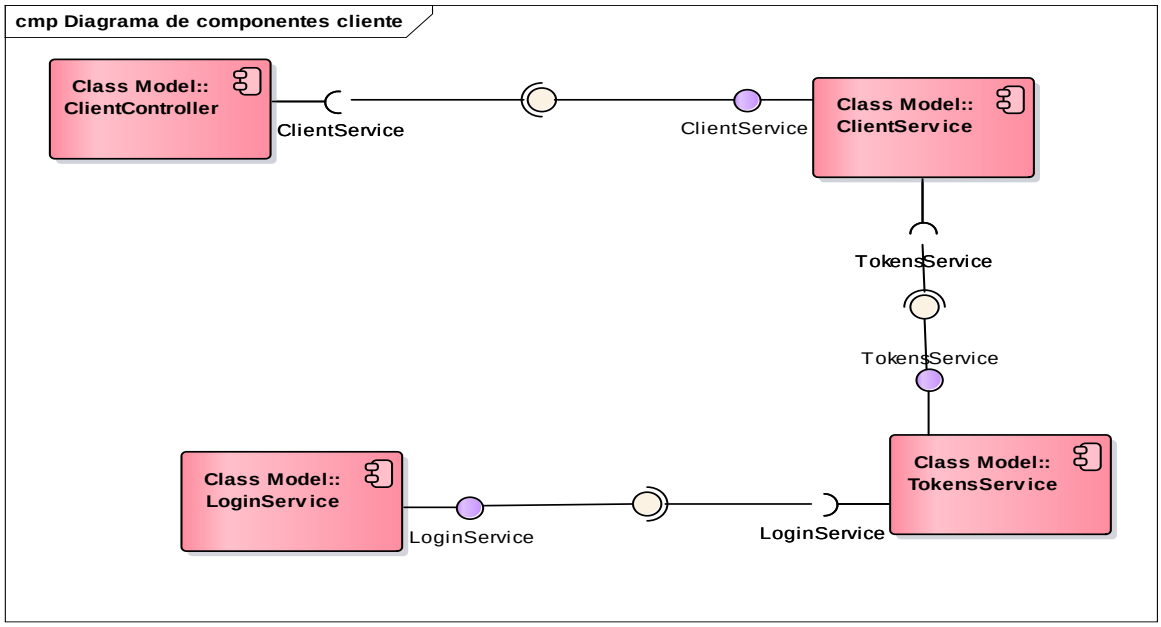


Ilustración 35 – Diagrama de componentes cliente

5.1.2.1.3 Diagrama de componentes: Request

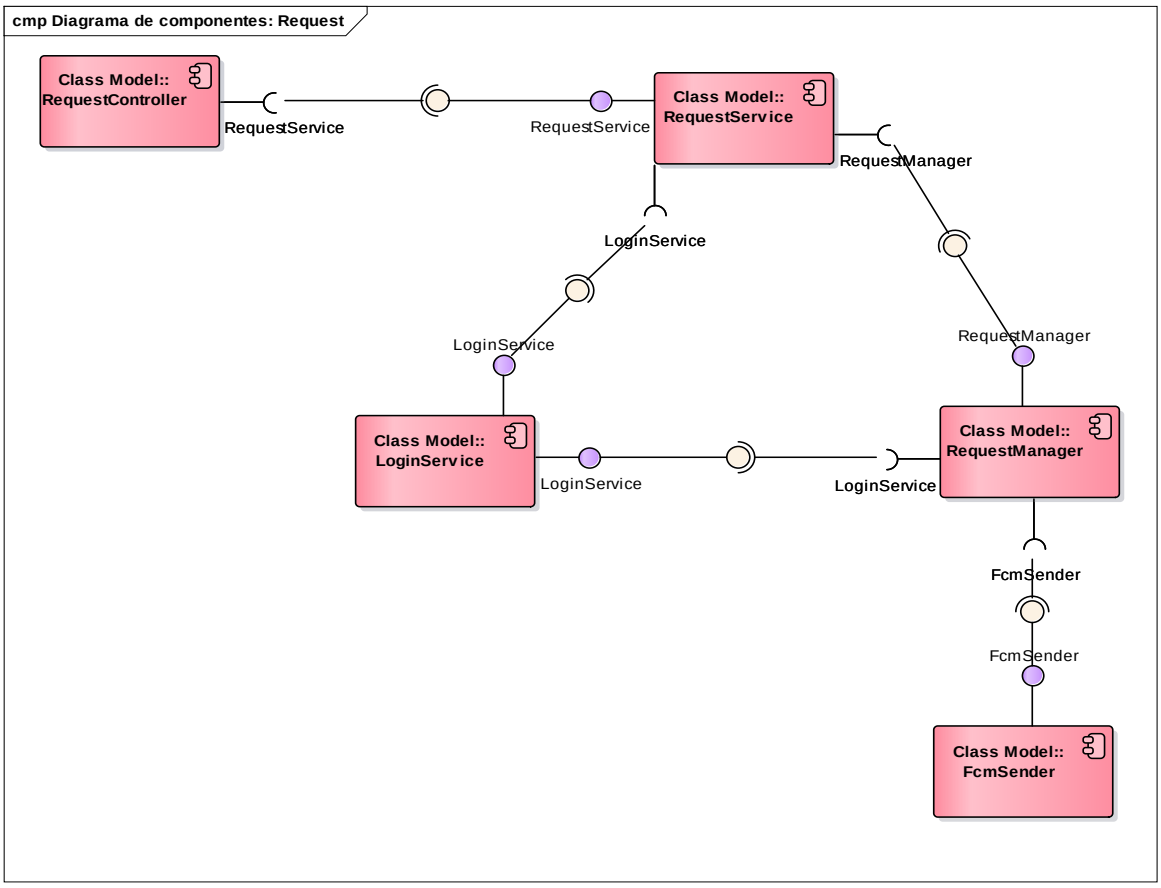


Ilustración 36- Diagrama de componentes de peticiones

5.1.2.1.4 Diagrama de componentes: SOS

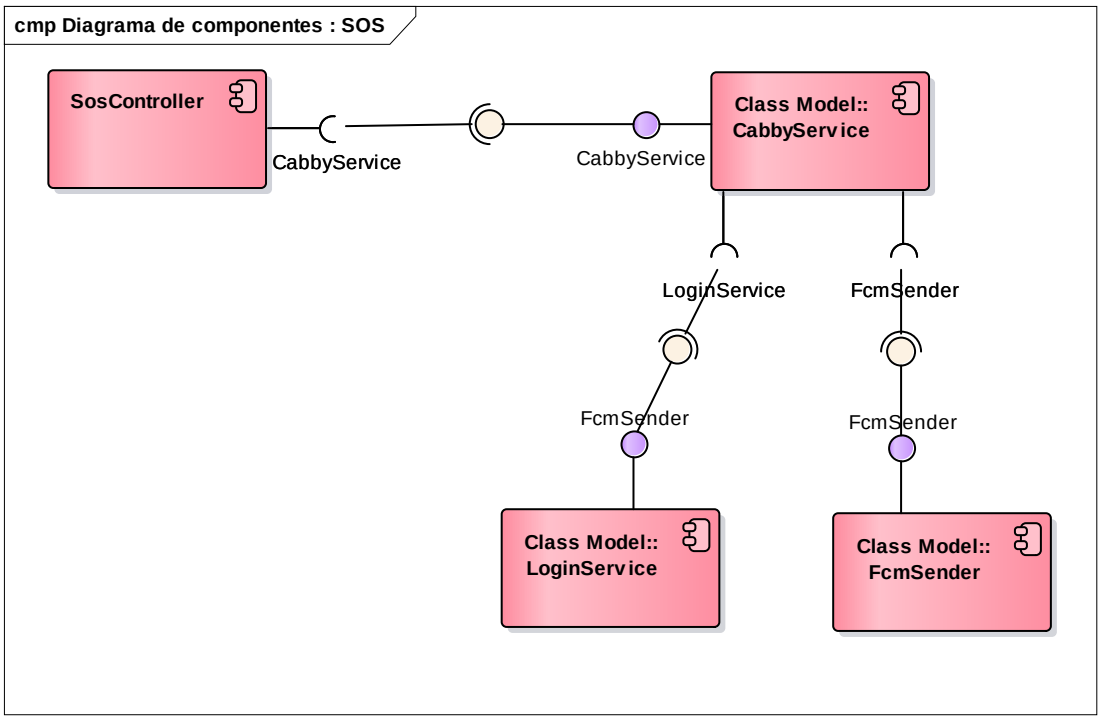


Ilustración 37 – Diagrama de componentes: SOS

5.1.2.1.5 Diagrama de componentes de Token

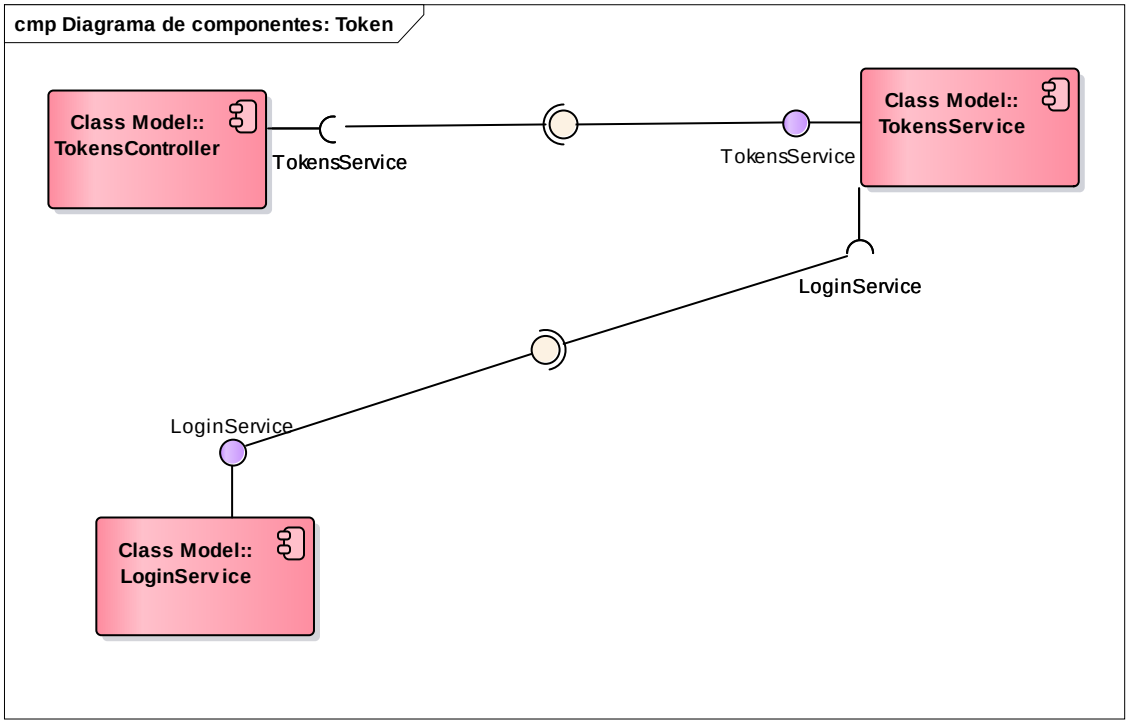


Ilustración 38 – Diagrama de componentes: Token

5.1.2.2 Diagramas de componentes de la aplicación cliente

En el caso de las aplicaciones vamos a dividir los diagramas por página, para así facilitar su visualización.

5.1.2.2.1 Diagrama de componentes: Home

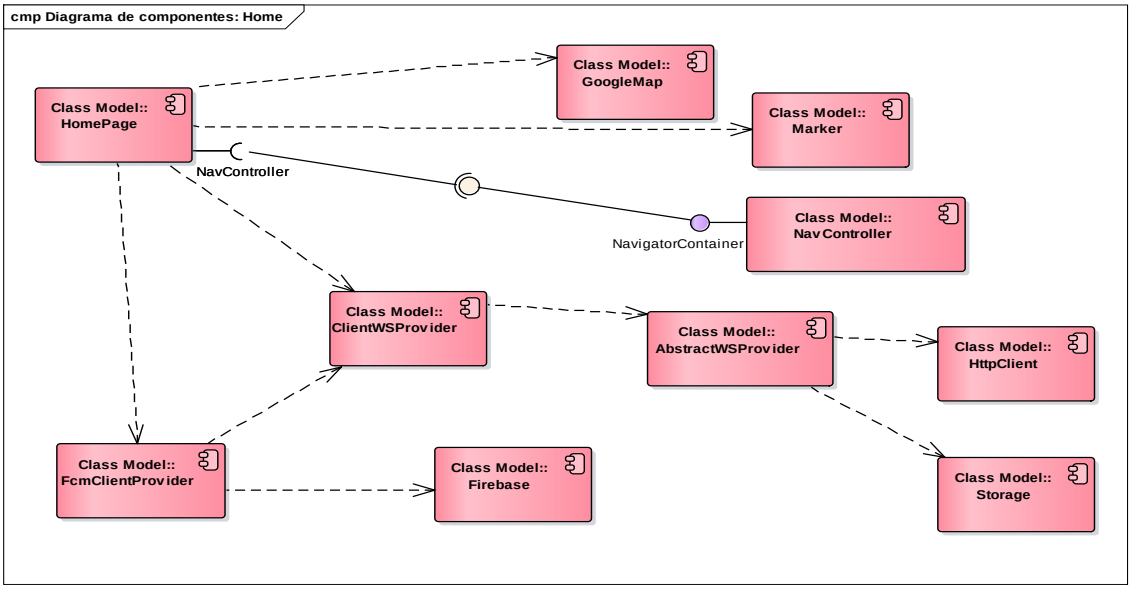


Ilustración 39 -Diagrama de componentes: Home

5.1.2.2.2 Diagrama de componentes: Login

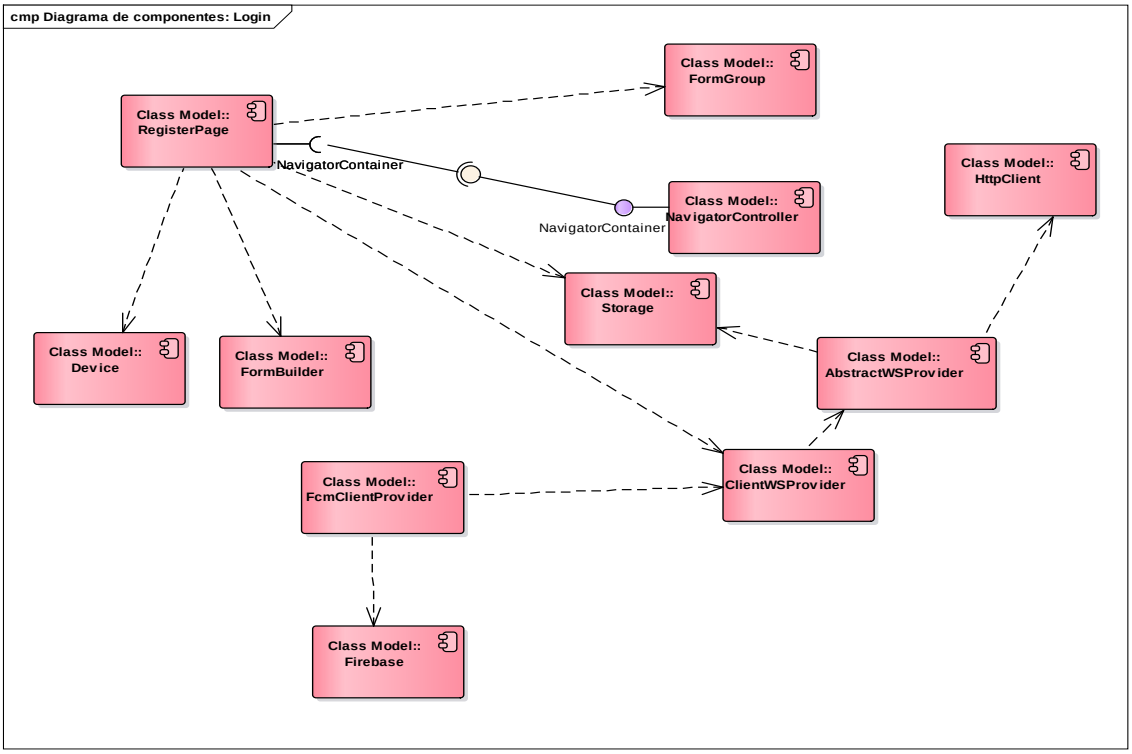


Ilustración 40 –Diagrama de componentes: LoginPage

5.1.2.2.3 Diagrama de componentes: Open-connections

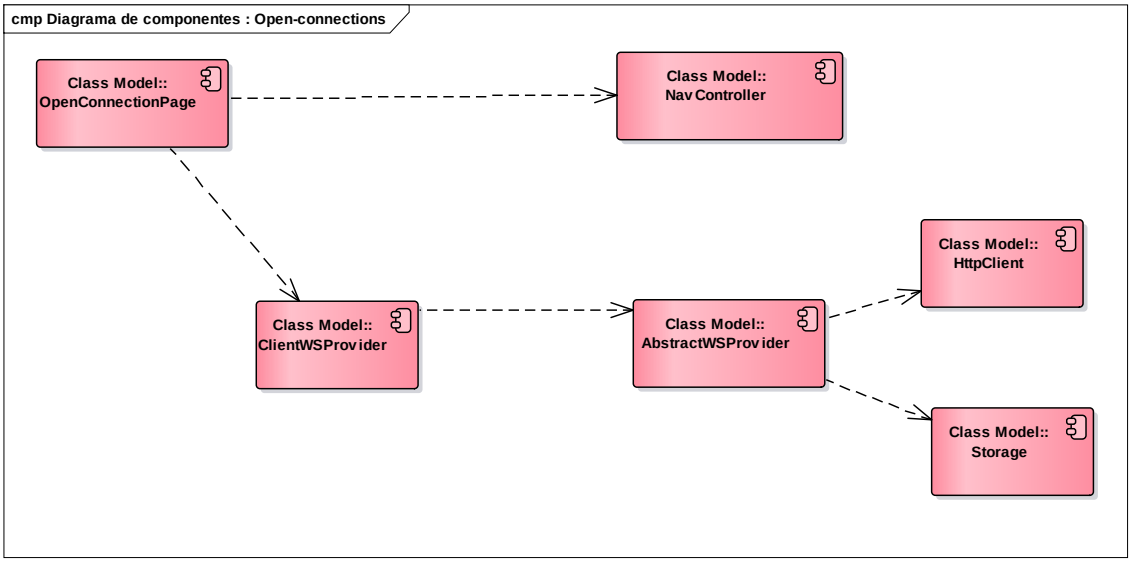


Ilustración 41 – Diagrama de componentes: Open-connections

5.1.2.2.4 Diagrama de componentes: Register

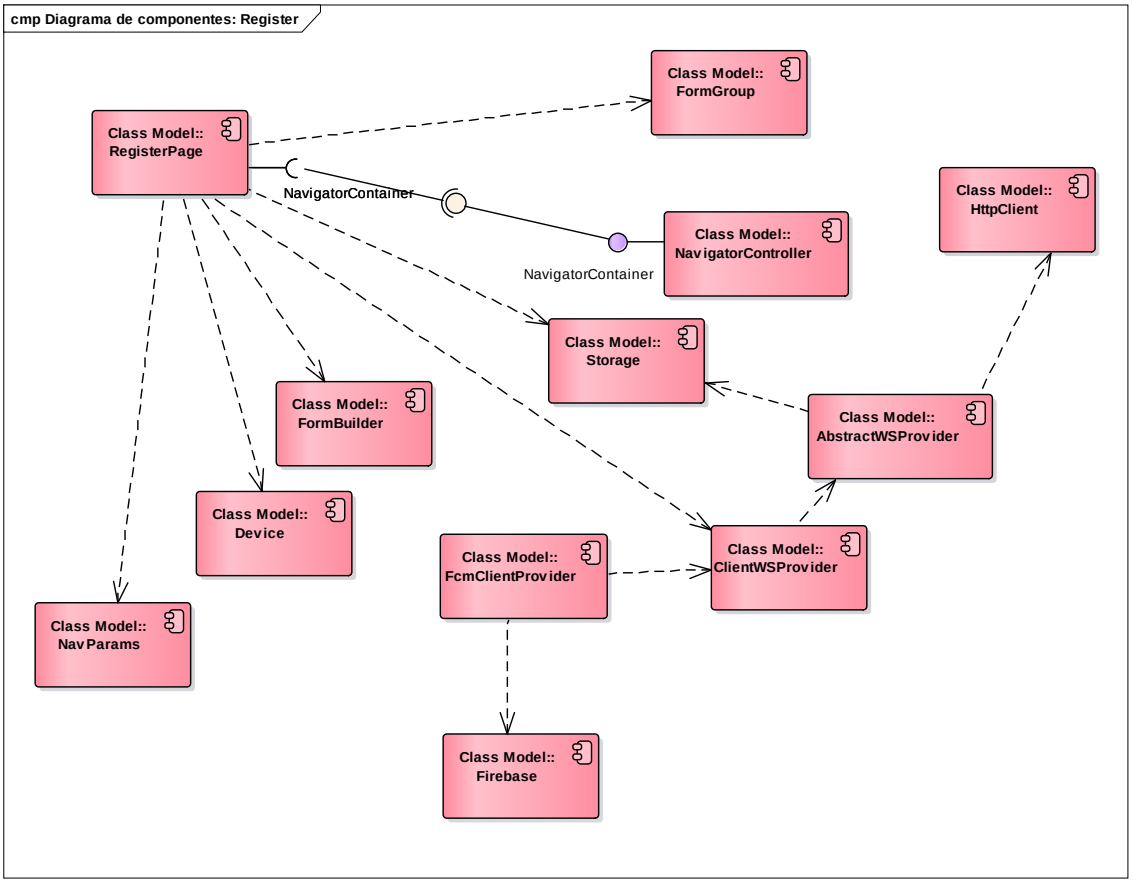


Ilustración 42- Diagrama de componentes: Register

5.1.2.3 Diagramas de componentes de la aplicación taxista

En este caso como el anterior, se hará un diagrama por pantalla. El diagrama de "Login" y el de "open-connections" se omite, ya que es prácticamente igual que el del cliente con la salvedad de que utiliza "CabbyWSProvider" en vez de "ClientWSProvider" y este componente forma parte también de los otros diagramas.

5.1.2.3.1 Diagrama de componentes: Home

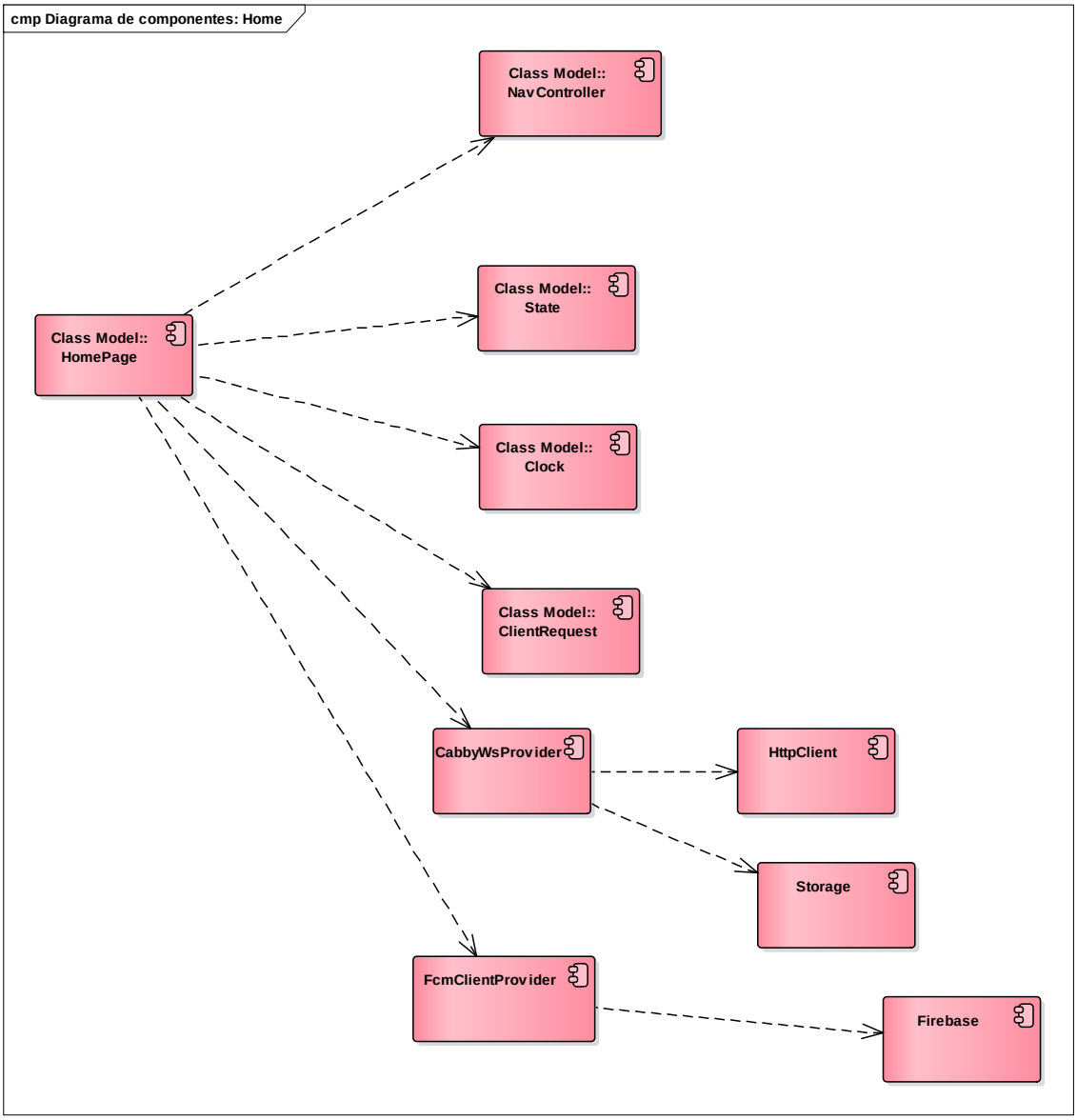


Ilustración 43 – Diagrama de componentes :Home

5.1.2.3.2 Diagrama de componentes: SOS

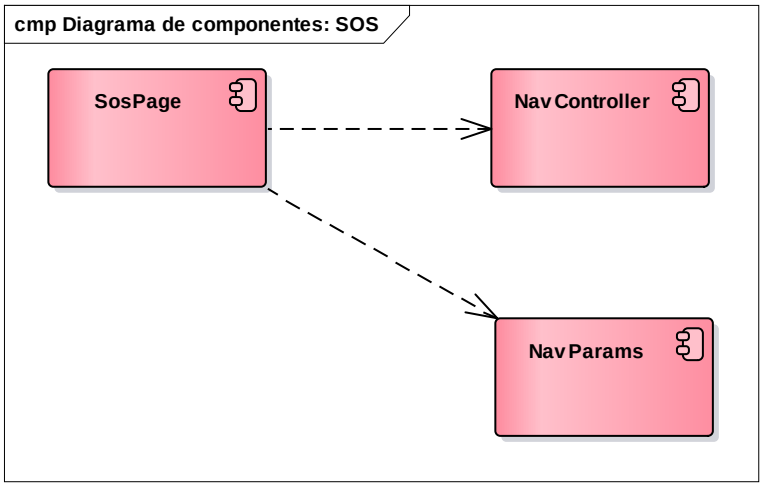


Ilustración 44 – Diagrama de componentes: SOS

5.1.2.3.3 Diagrama de componentes: map

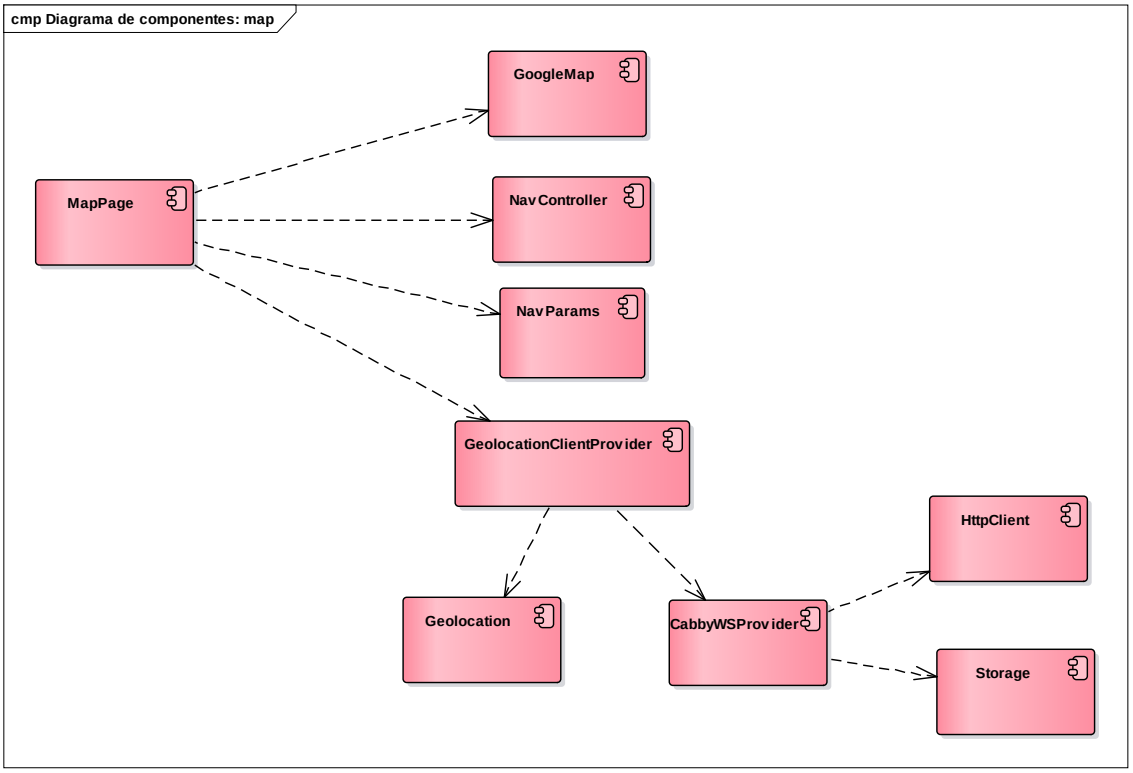


Ilustración 45 – Diagrama de componentes: map

5.1.3 Diagramas de Despliegue

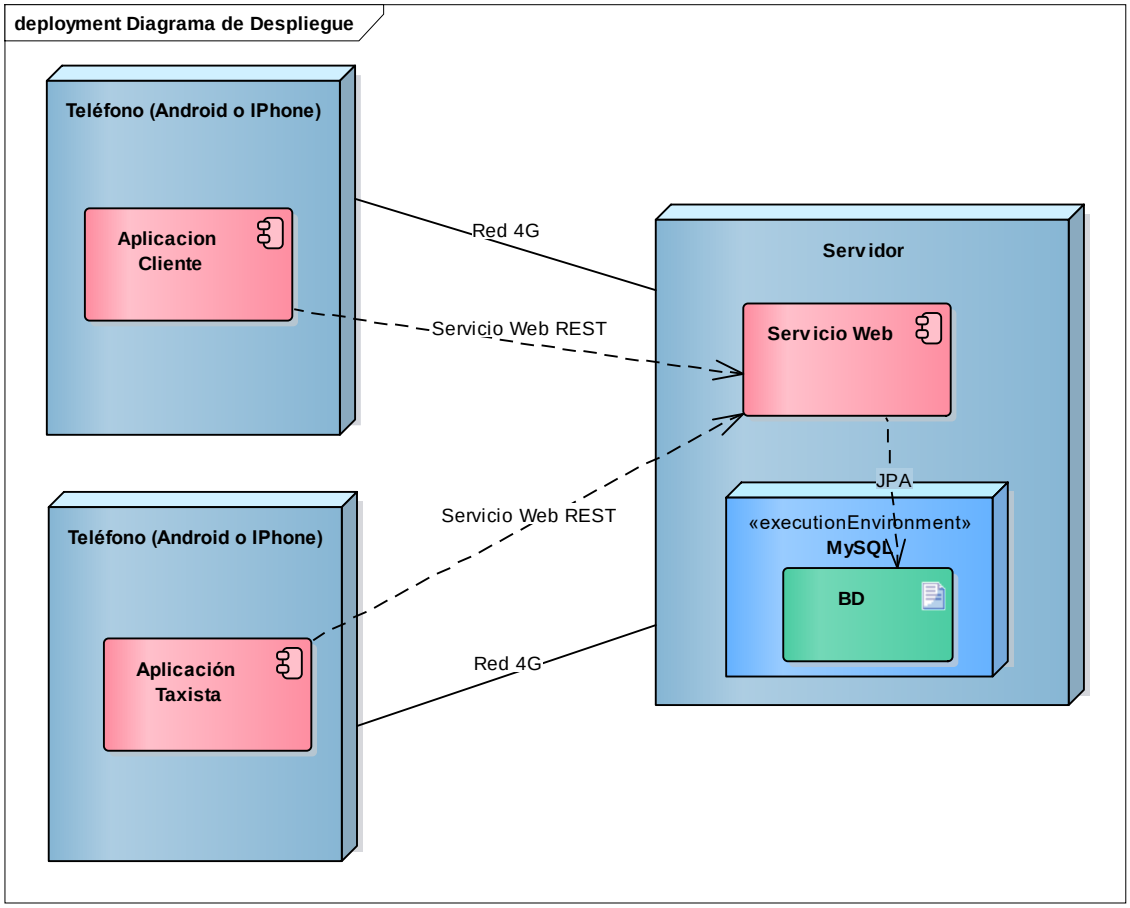


Ilustración 46 – Diagrama de despliegue

Los móviles deben tener al menos la versión de Android 4.4 (Kit Kat) y de Apple debería ser la iOS 8.

En cuanto al servidor debe tener instalado Java 9, ASÍ COMO MySql 8.0.

5.2 Diseño de Clases

5.2.1 Diagrama de Clases

A continuación, se muestran los diferentes diagramas de clases del sistema. Se ha dividido en varias partes para hacer más legible.

5.2.1.1 Diagramas de clases del servidor

Este diagrama en su vez se ha dividido porque debido a su tamaño en este formato sino sería ilegible, se ha optado para una mejor comprensión del sistema dividirlo en los diferentes controladores que lo componen.

5.2.1.1.1 Tokens

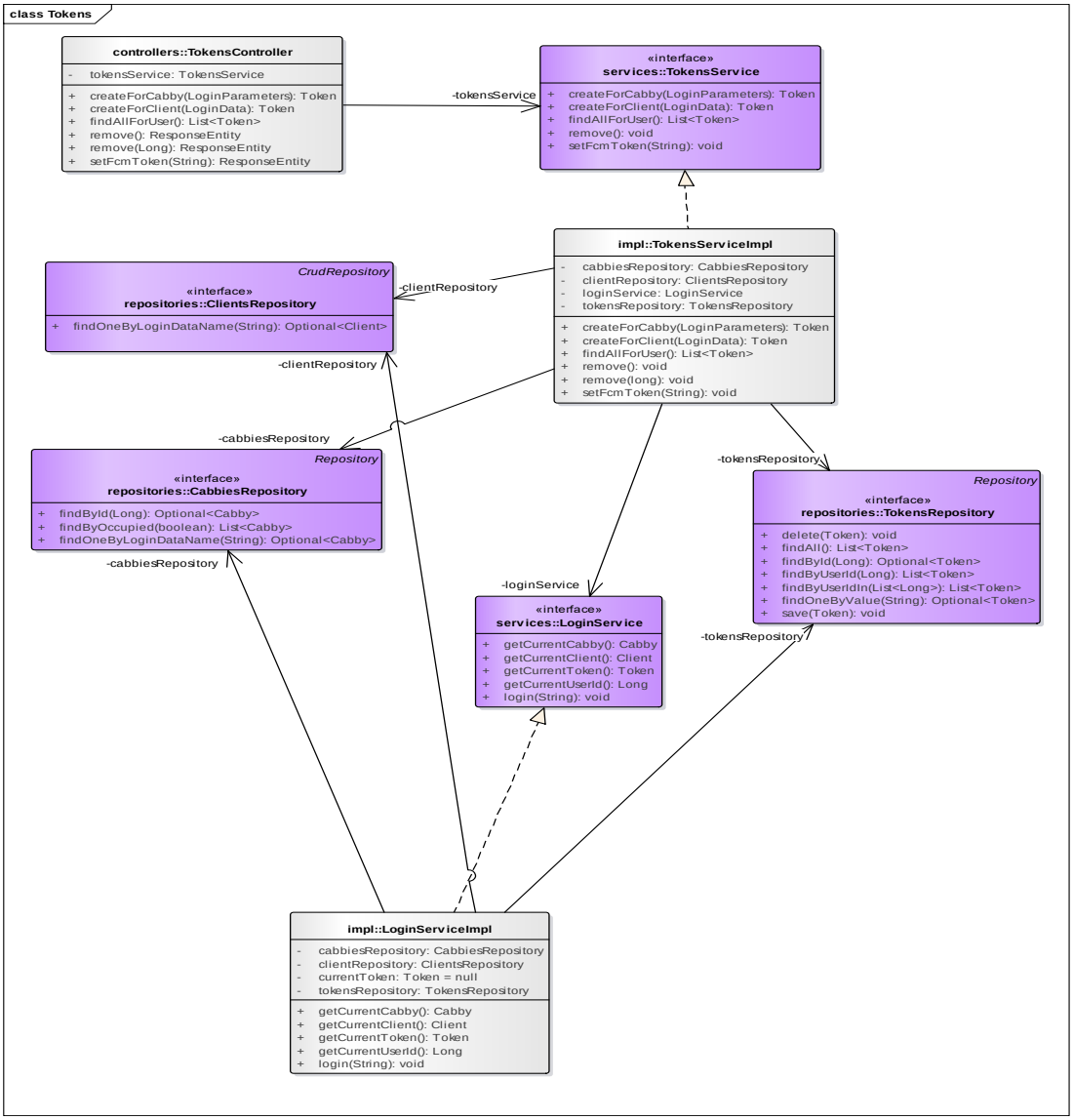


Ilustración 47 – Diagrama de clases: Tokens

5.2.1.1.2 Clientes

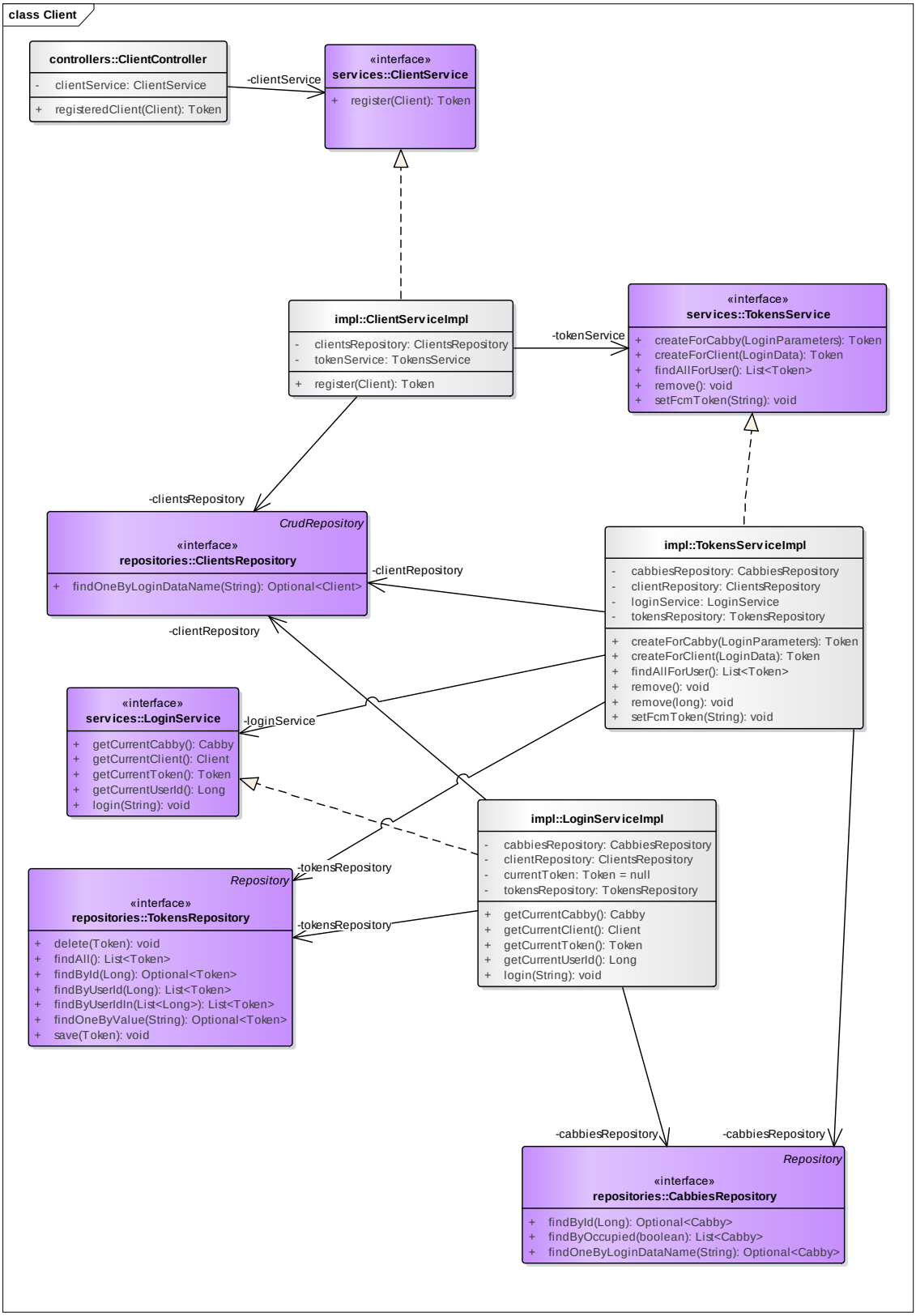


Ilustración 48- Diagrama de clases: Clientes

5.2.1.1.3 Taxistas

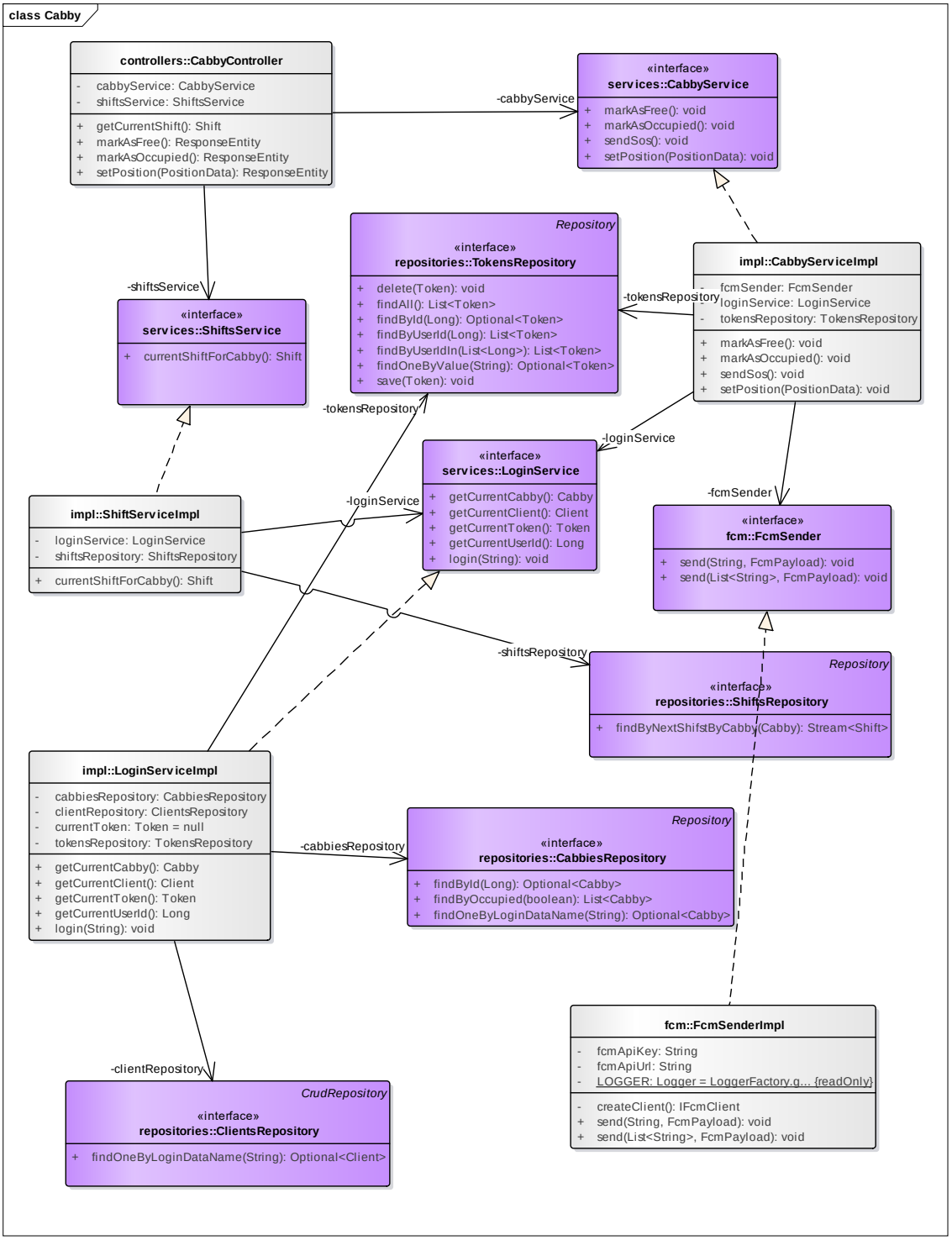


Ilustración 49 – Diagrama de clases: Taxistas

5.2.1.1.4 Peticiones

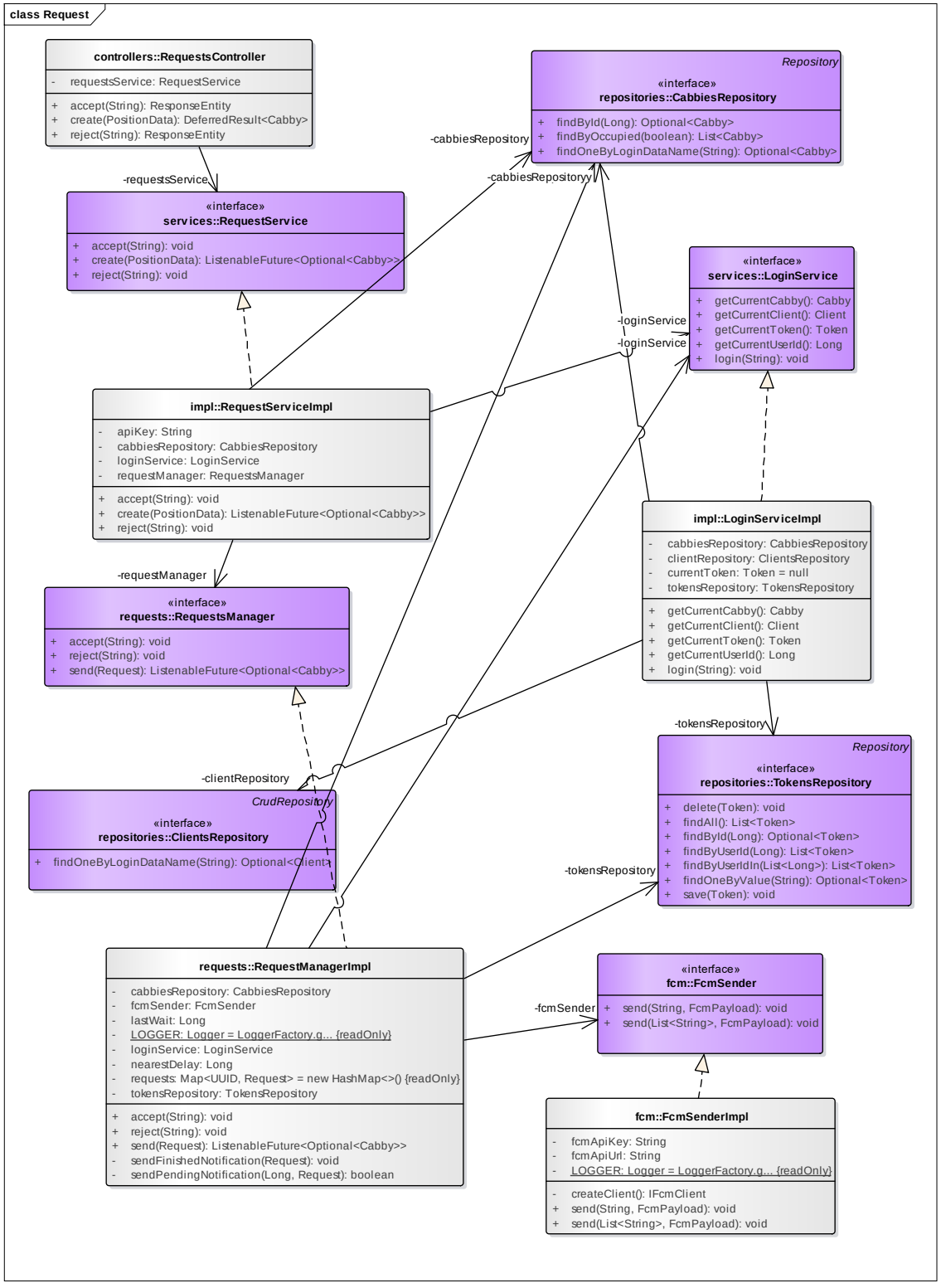


Ilustración 50 – Diagrama de clases Peticiones

5.2.1.1.5 SOS

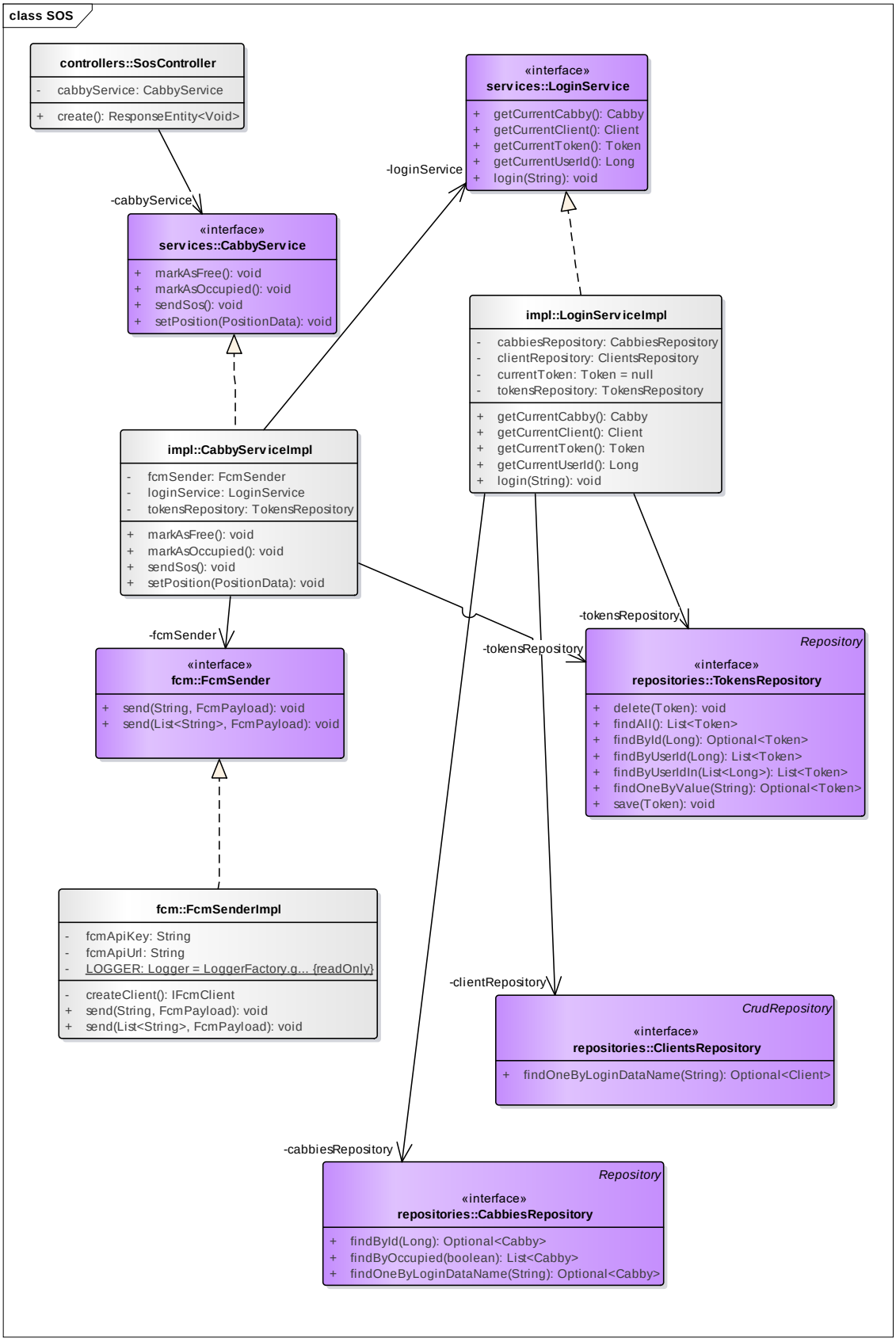


Ilustración 51 – Diagrama de clases: Sos

5.2.1.2 Diagrama de clases de la aplicación del cliente

Como estamos haciendo hasta ahora los diagramas de las aplicaciones móviles los vamos a dividir por pantalla, para que así sean más claros y menos sobrecargados.

5.2.1.2.1 Diagrama de clases: Home

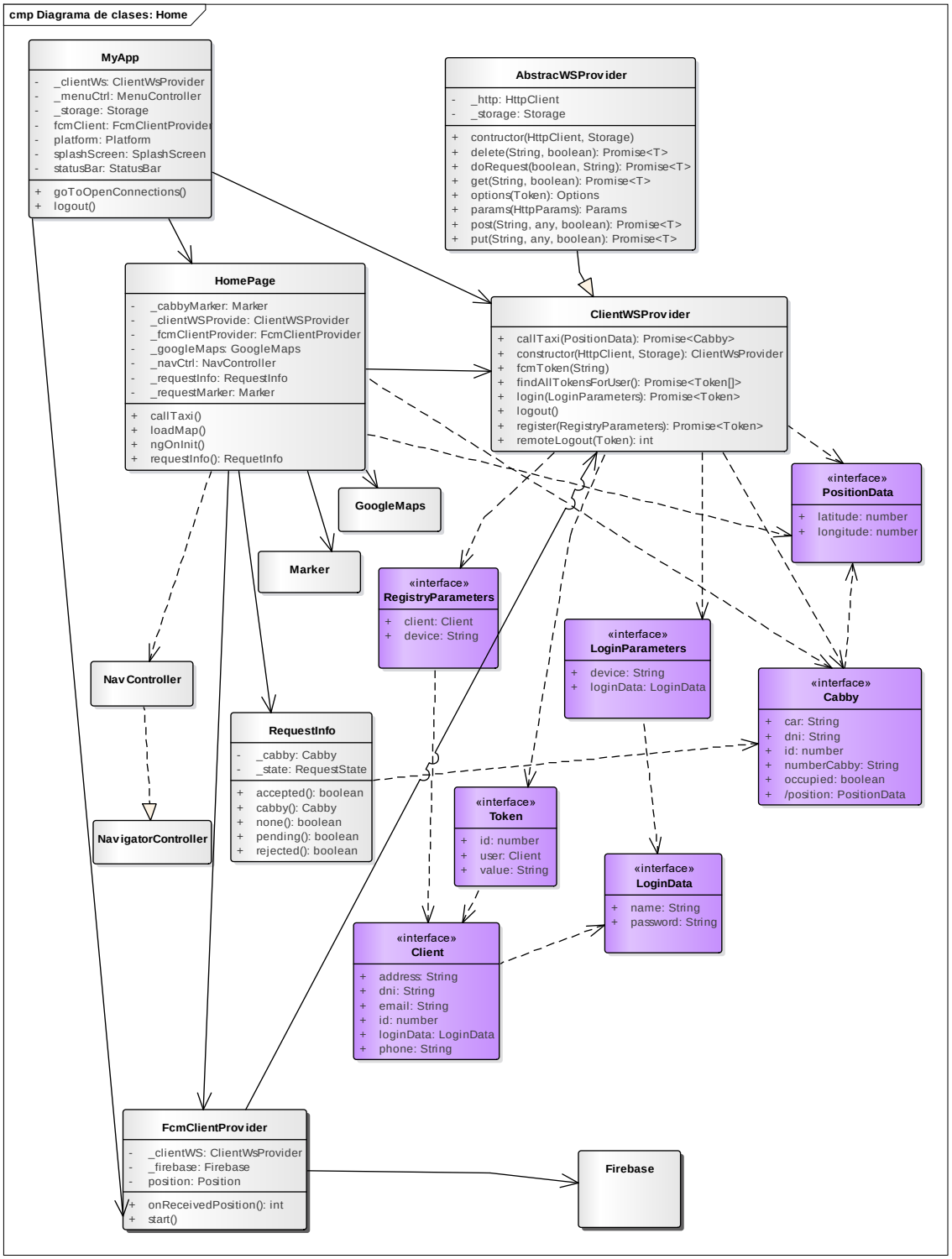


Ilustración 52 – Diagrama de clases: Home

5.2.1.2.2 Diagrama de clases: Login

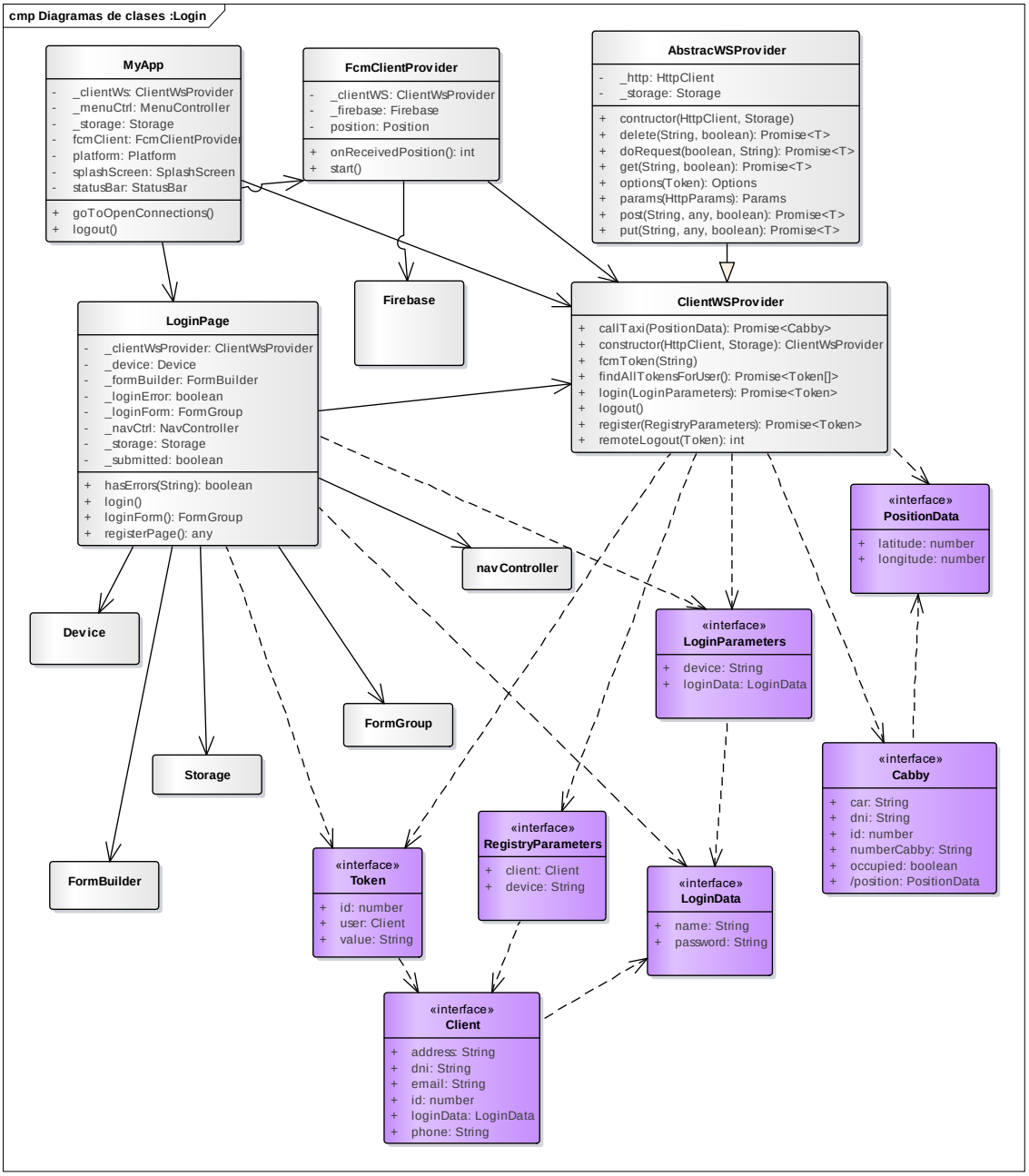


Ilustración 53 – Diagrama de clases: Login

5.2.1.2.3 Diagrama de clases: Register

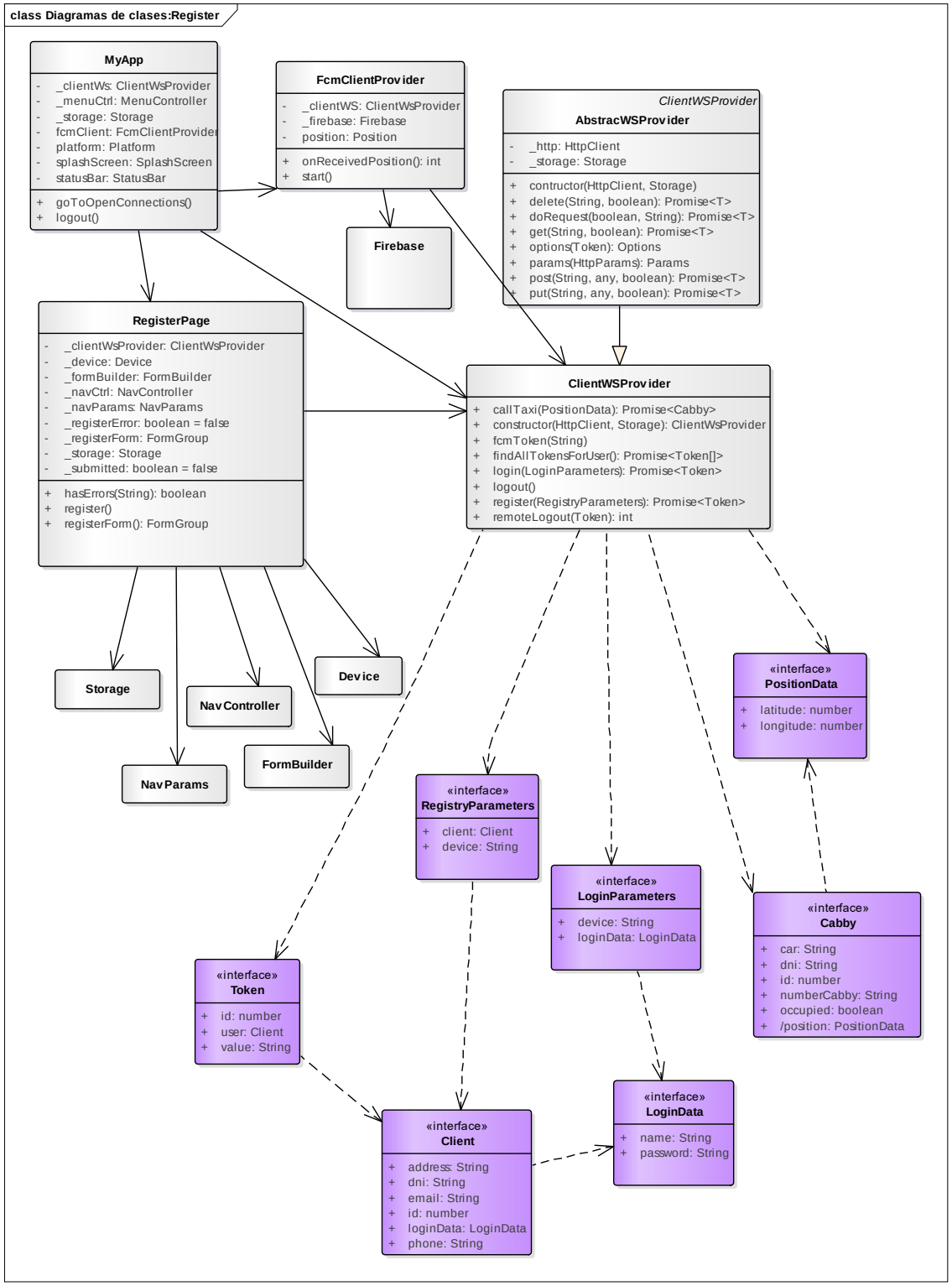


Ilustración 54 – Diagrama de clases Register

5.2.1.2.4 Diagrama de clases: Open. Connections

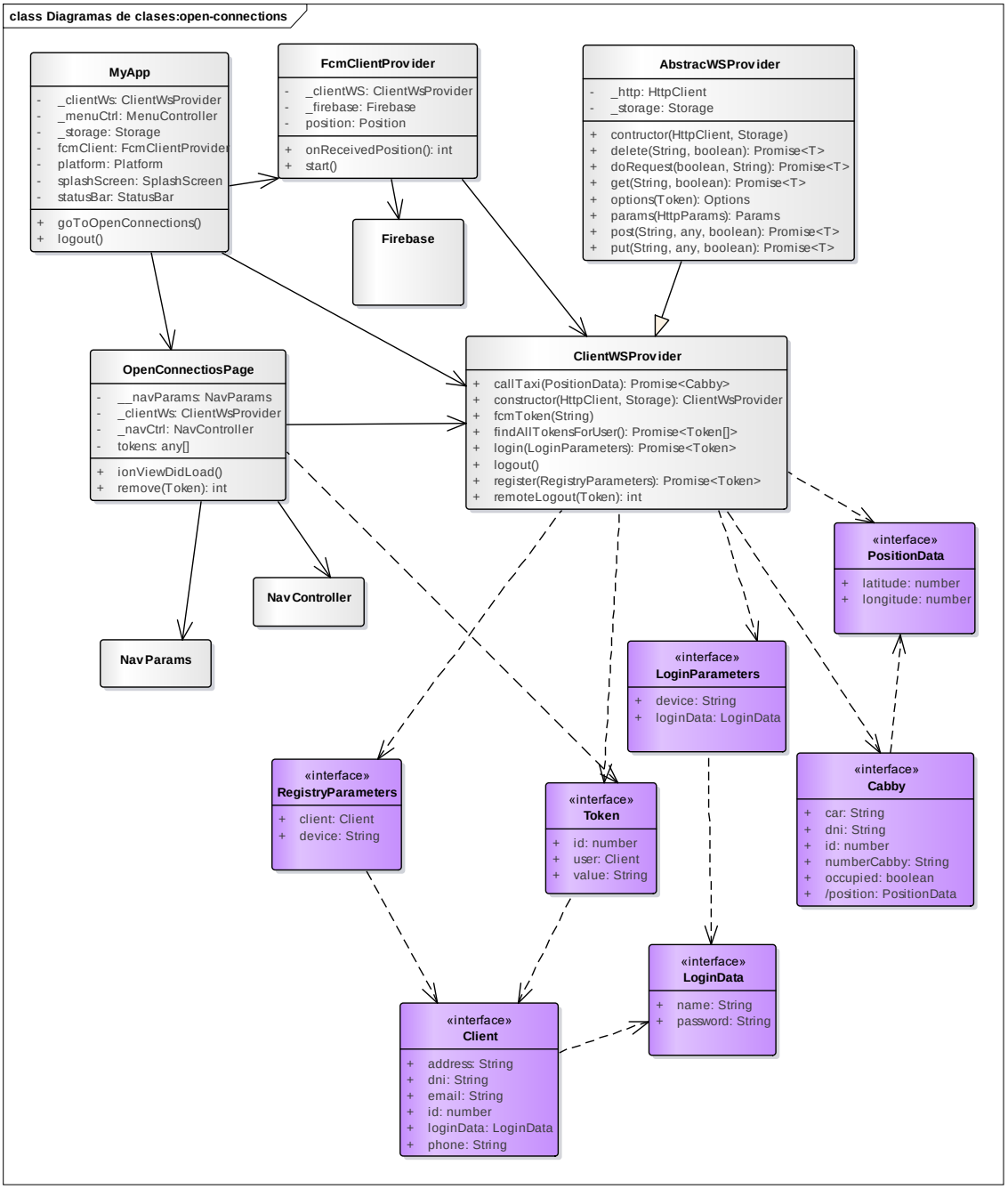


Ilustración 55 – Diagrama de clases: Openn-connections

5.2.1.3 Diagrama de clases de la aplicación taxista

Por una mejor claridad y comprensión se hace la misma subdivisión de los diagramas de clases que en la de clientes.

Debido al tamaño de los diagramas, en algunos casos clases que ya se han definido en uno no aparecen definidas en otro. El diagrama de clases open-connections es casi exactamente igual que el de la aplicación de cliente , por eso es omitido en este apartado.

5.2.1.3.1 Diagrama de clases: Home

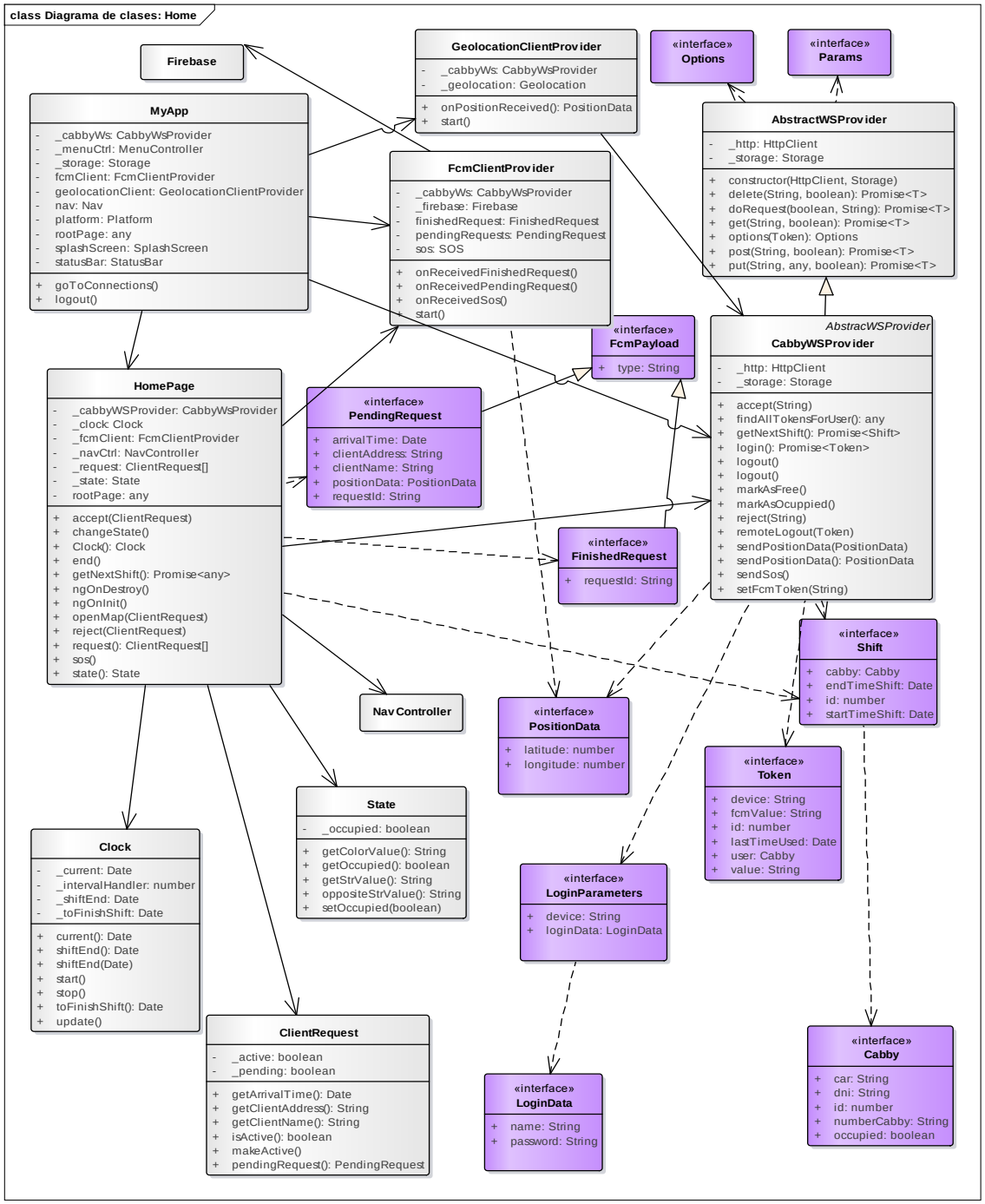


Ilustración 56 – Diagrama de clases: Home

5.2.1.3.2 Diagrama de clases: Login

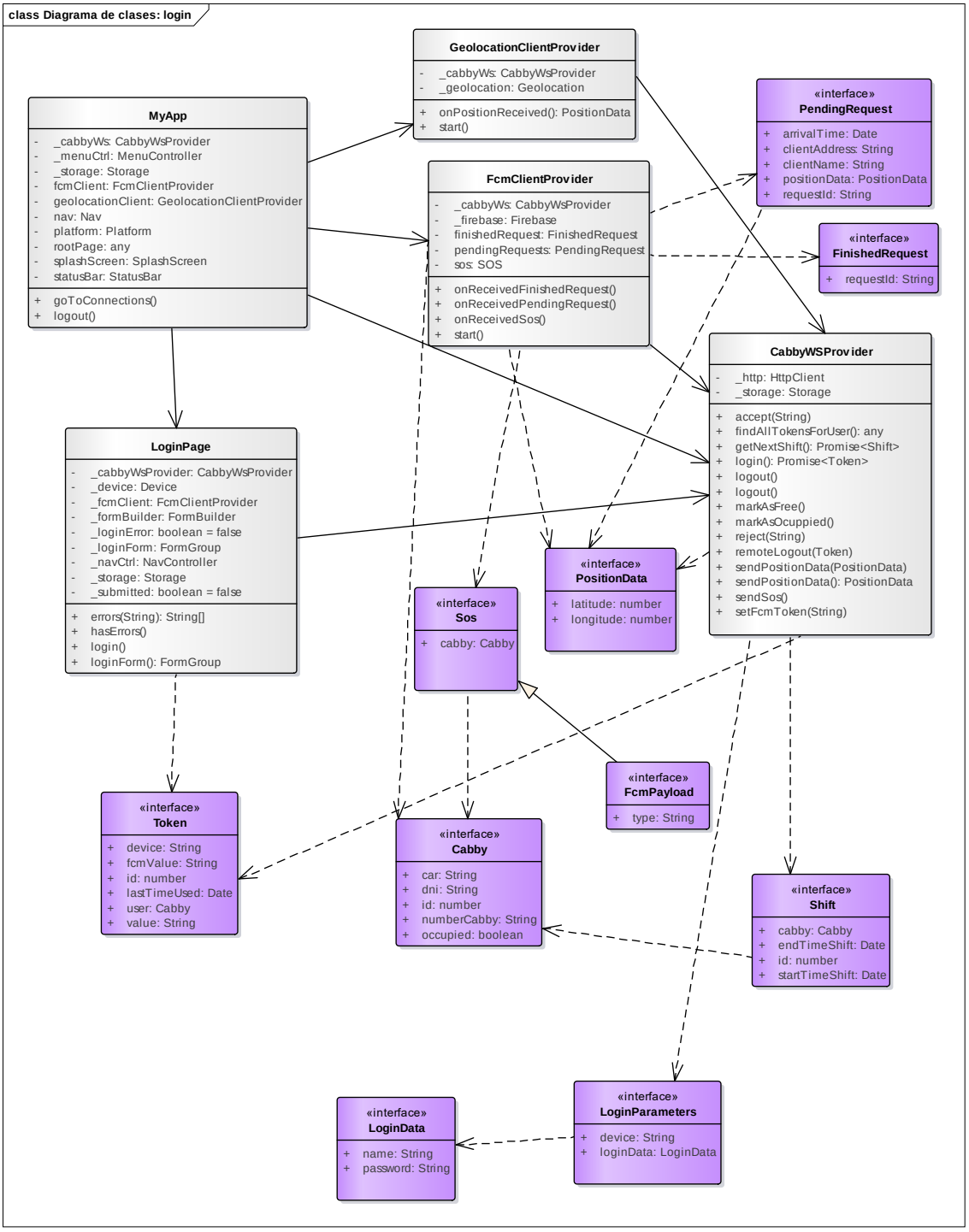


Ilustración 57 – Diagrama de clases: Login

5.2.1.3.3 Diagrama de clases: map

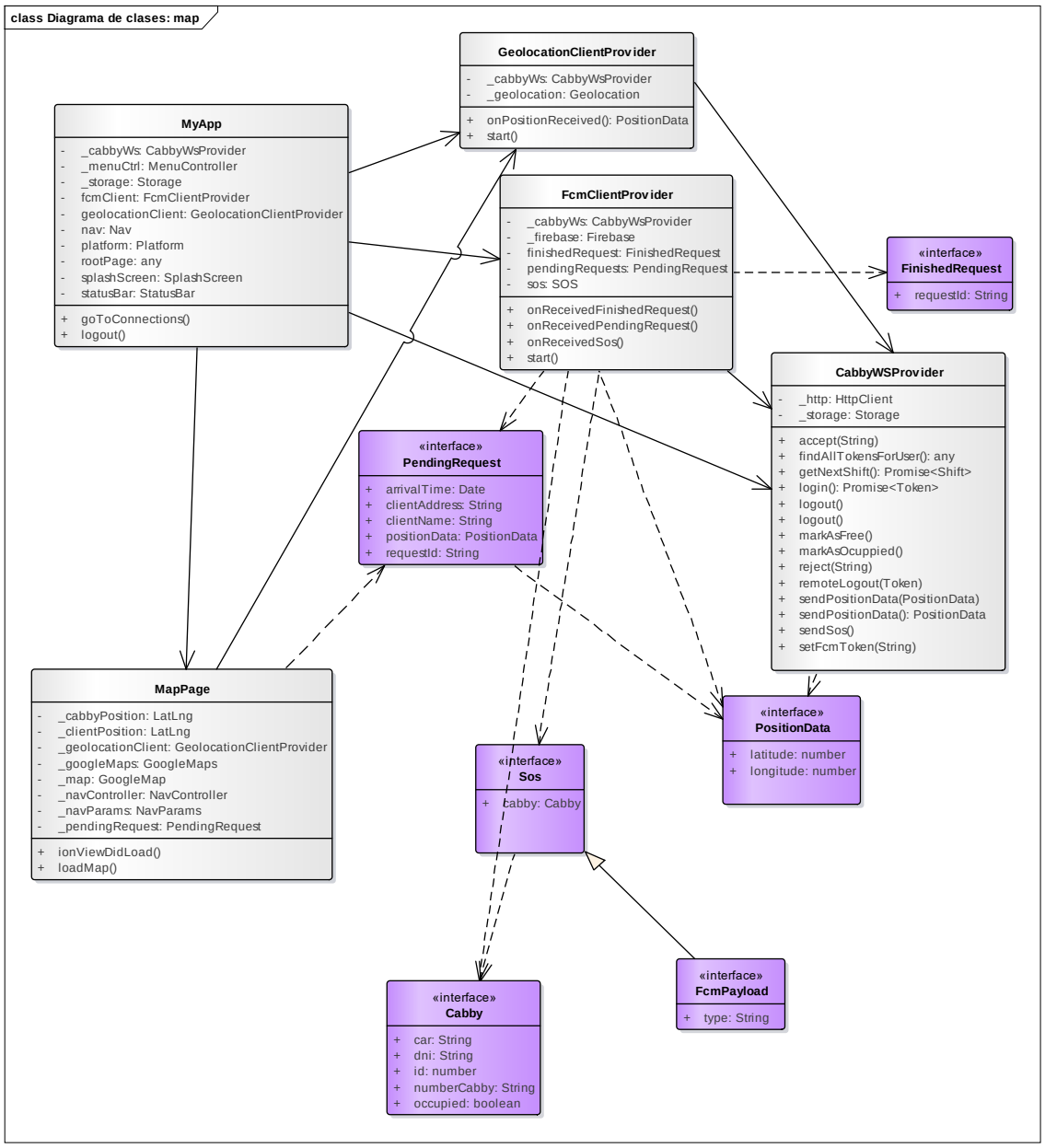


Ilustración 58- Diagrama de clases: map

5.2.1.3.4 Diagrama de clases: sos

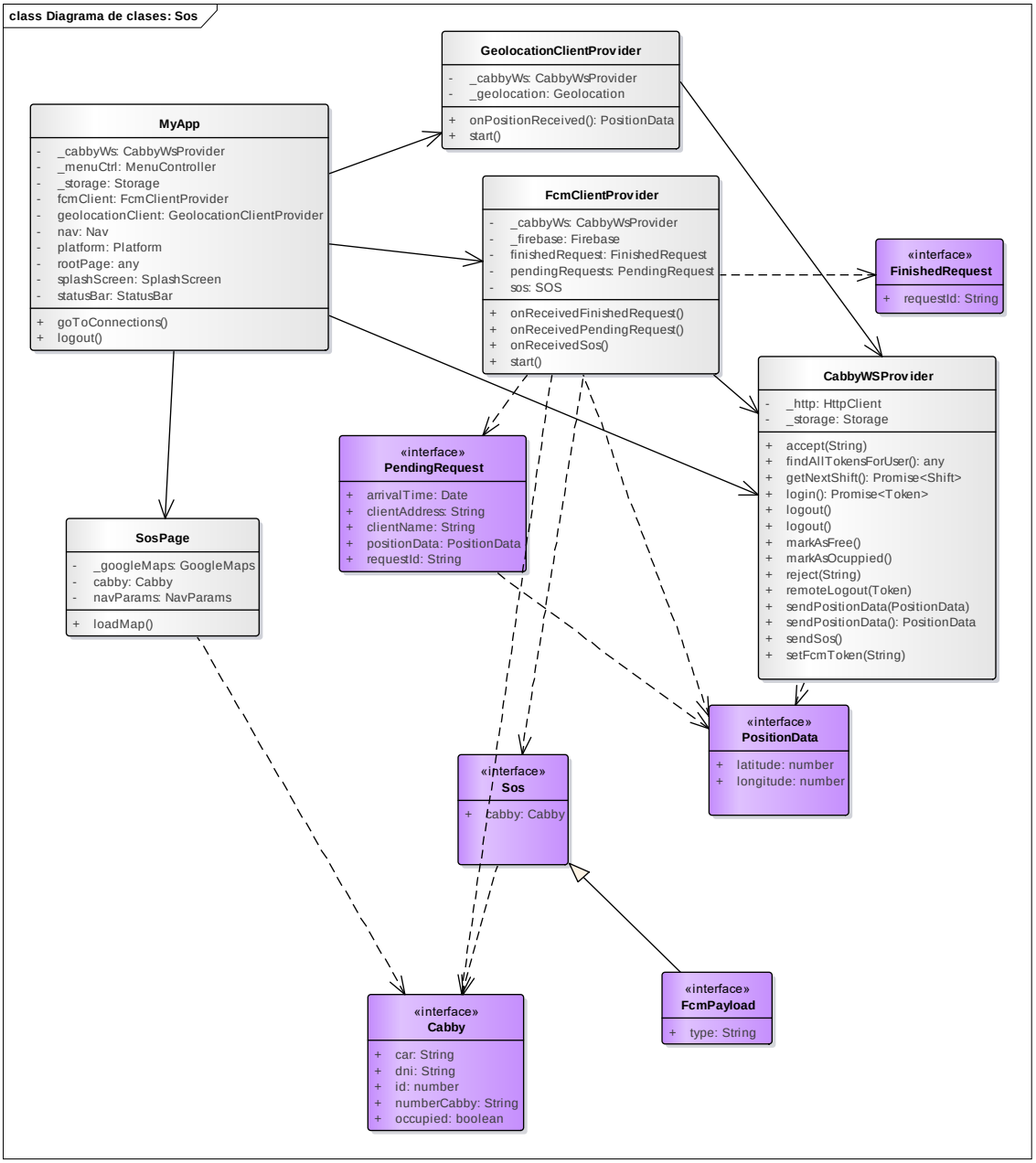


Ilustración 59- Diagrama de clases: Sos

5.2.2 Diagramas de Interacción y Estados

5.2.2.1 Diagrama de estados: Pedir taxi

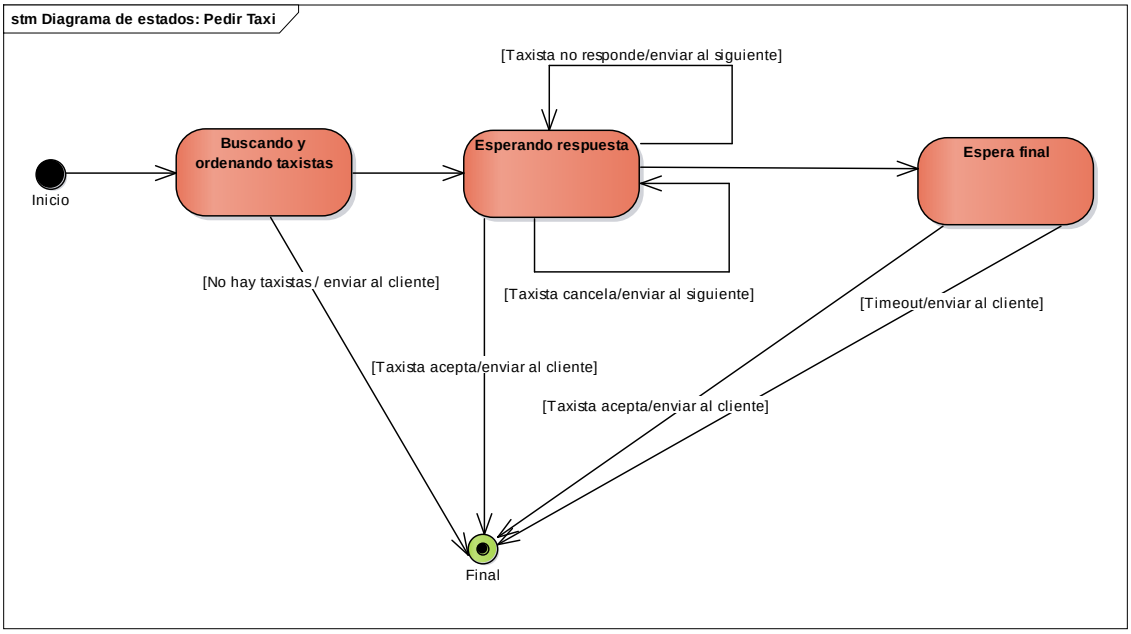


Ilustración 60 – Diagrama de estados: Pedir taxi

5.2.3 Diagramas de Interacción

5.2.3.1 Diagrama de secuencia: Pedir taxi

Debido al tamaño del diagrama y la complejidad del caso de uso, vamos a dividirlo en partes. La petición al servidor sigue el mismo proceso para todos los casos de uso, siendo manejada por el controlador correspondiente. En el caso de pedir taxi, está sería la secuencia:

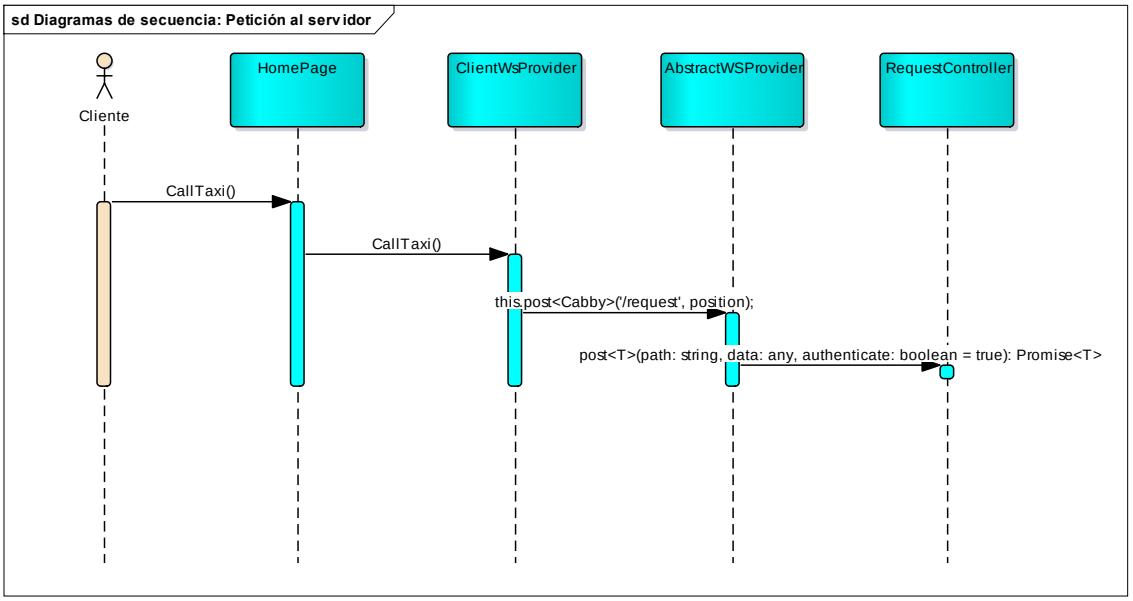


Ilustración 61 – Diagrama de secuencia: Pedir taxi (petición al servidor)

Cuando la petición llega al servidor, la secuencia que esta sigue se puede ver en el siguiente diagrama:

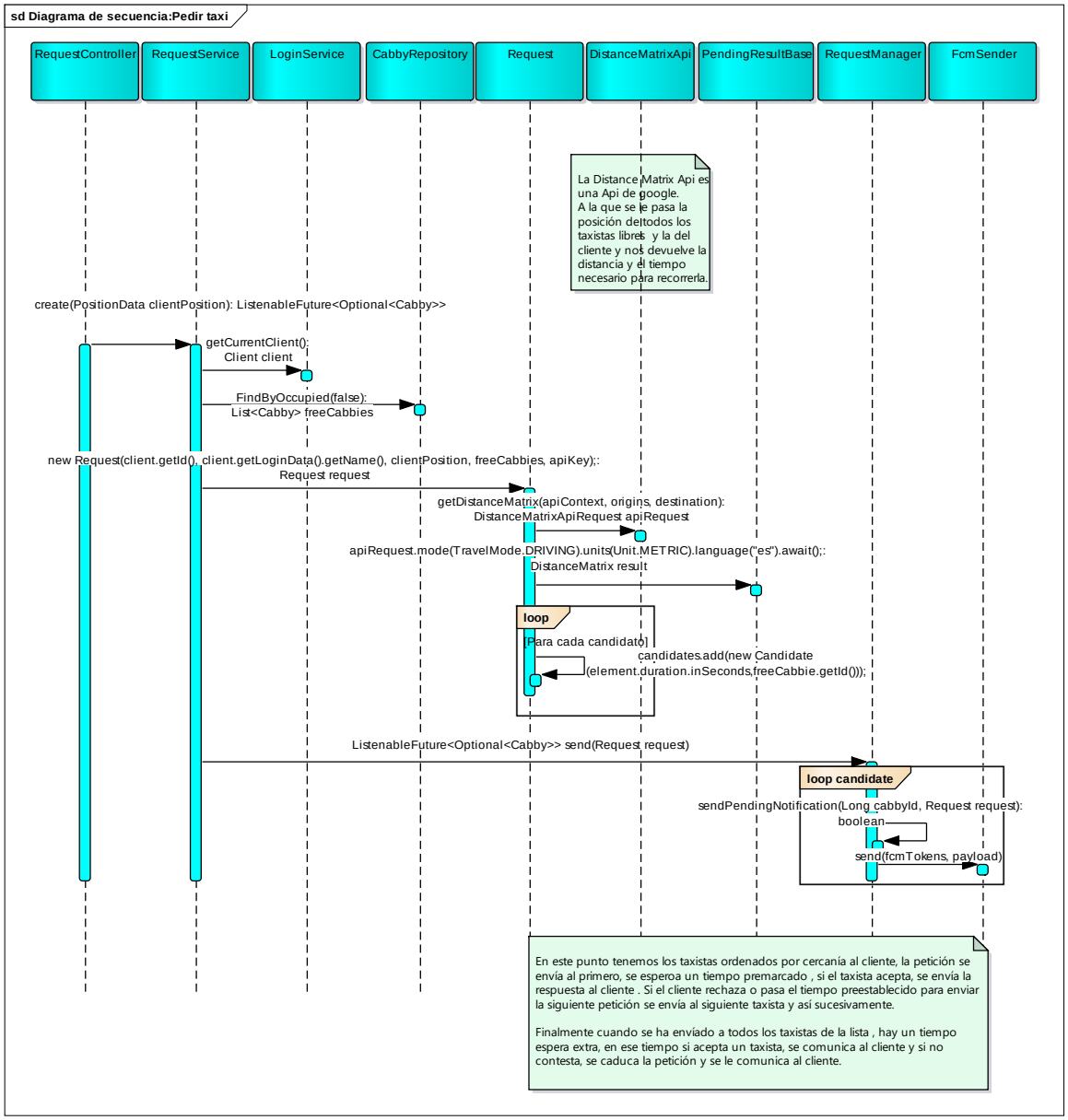


Ilustración 62 – Diagrama de secuencia pedir taxi (En el servidor)

Y por último quedaría el diagrama de respuesta del taxista:

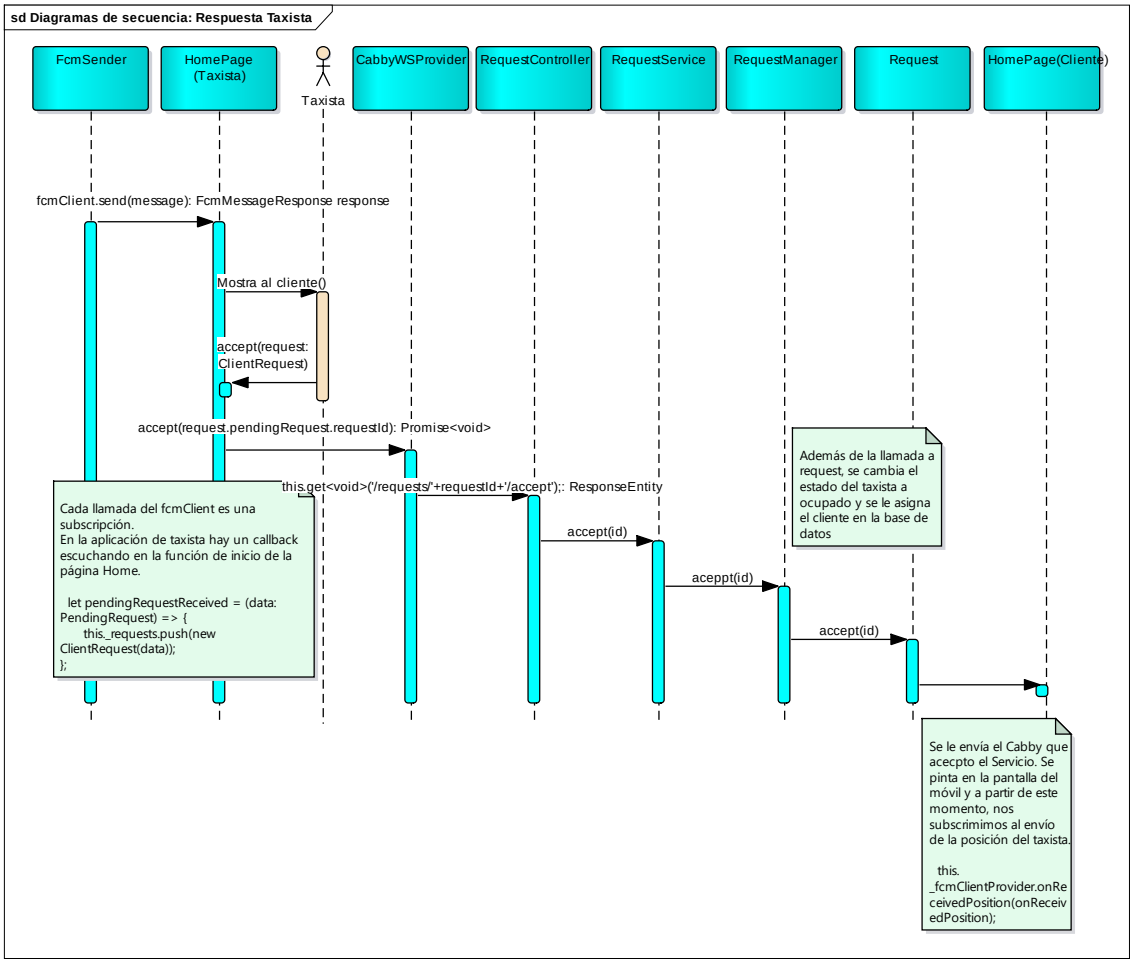


Ilustración 63- Diagrama se secuencia: Respuesta taxista (aceptación)

En cuanto al rechazo, se espera a que todos los taxistas rechacen o ninguno conteste para enviarle el mensaje de rechazo al cliente.

5.2.3.2 Diagrama de secuencia: Sos

El envío al servidor sigue el mismo proceso que el anterior, cambiando el nombre de los métodos y el punto de entrada.

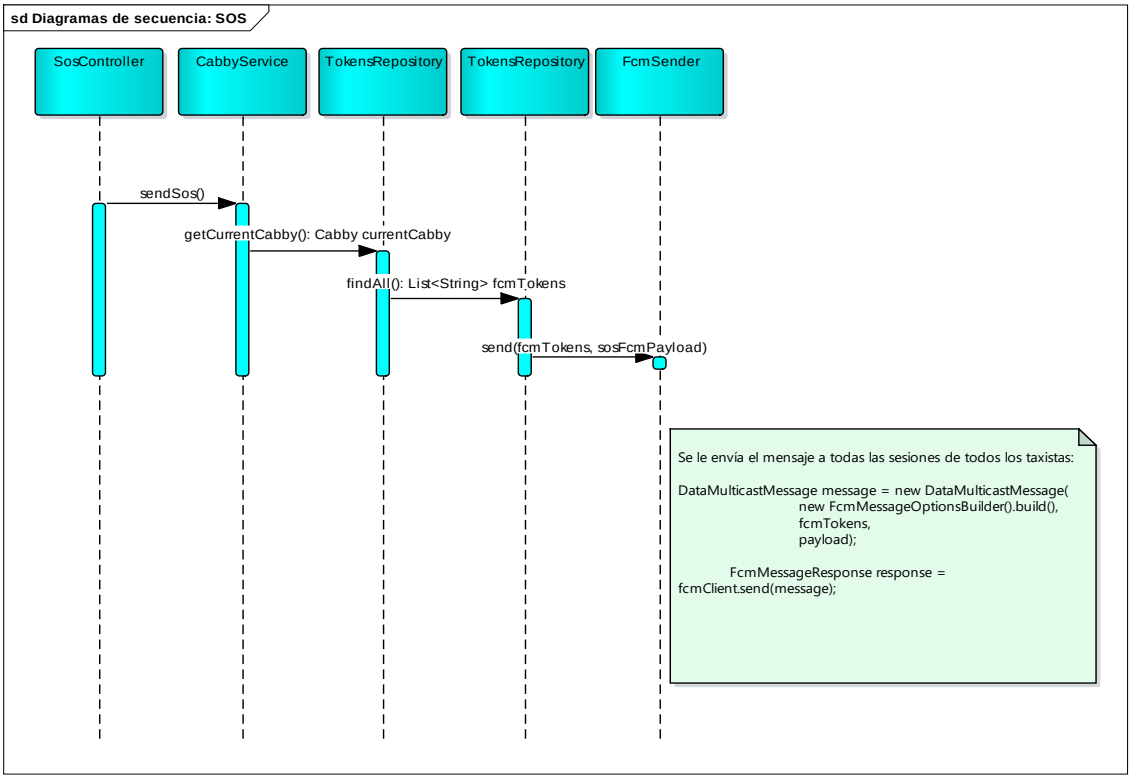


Ilustración 64- Diagrama de secuencia: SOS

5.2.3.3 Diagrama de secuencia: Registro

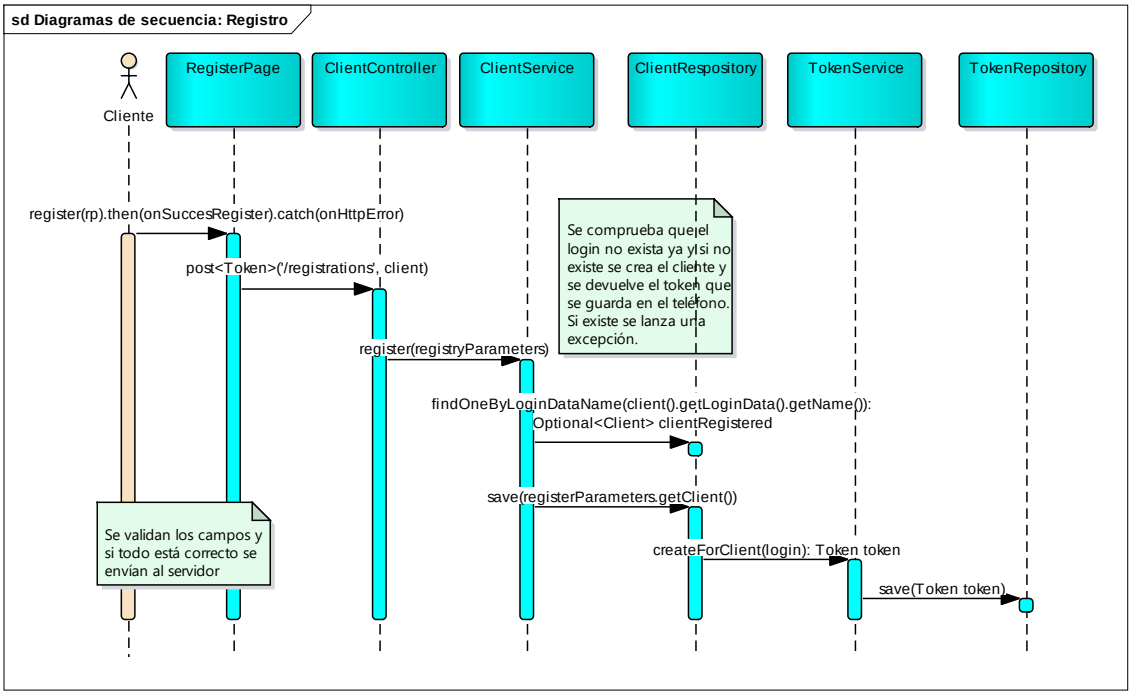


Ilustración 65- Diagrama de secuencia: Registro

5.2.3.4 Diagrama de actividad

Para que queda más clara la petición de taxi, ya que tiene bastante complejidad, en este diagrama se puede ver el flujo de la información a lo largo de todo el proceso:

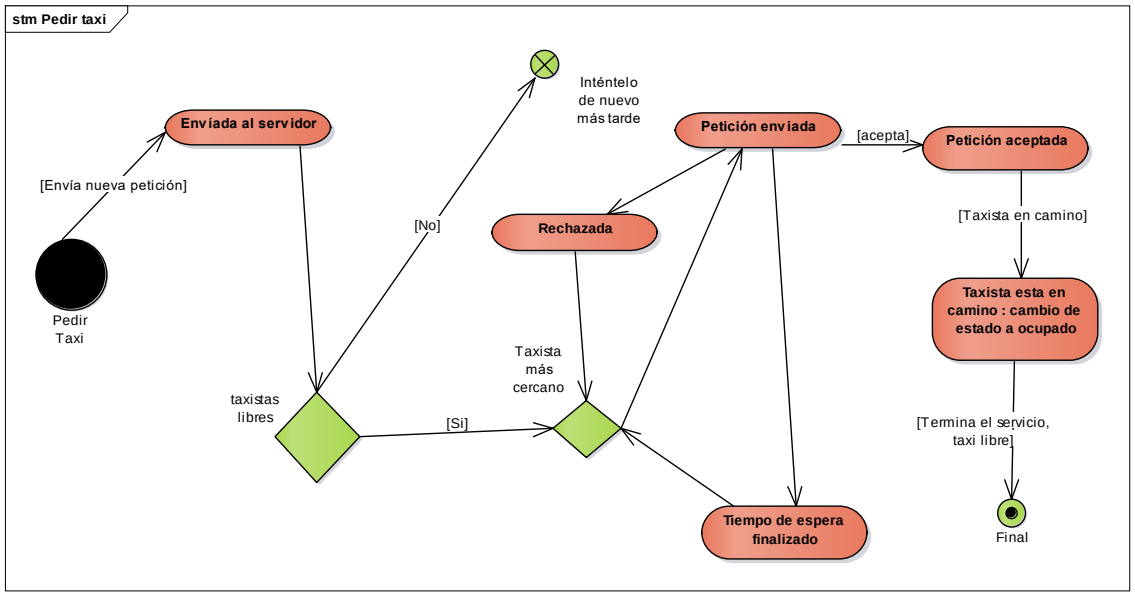


Ilustración 66- Diagrama pedir taxi

5.3 Diseño de la Base de Datos

5.3.1 Descripción del SGBD Usado

5.3.1.1 Almacenamiento en el teléfono

Storage es una forma fácil de guardar pares de clave/valor y objetos de JSON. Usa una gran variedad de ingenierías de almacenamiento utilizando la más conveniente dependiendo de la plataforma.

Cuando la aplicación está ejecutándose en un contexto de aplicación nativa, es decir, en el teléfono, este prioriza el uso de SQLite mientras que si lo hace en el navegador intenta usar IndexedDB, WebSQL y localstorage en ese orden.

Esto nos va a permitir guardar los tokens en las aplicaciones móviles, para poder mantener las sesiones abiertas mientras el usuario no desee cerrarlas, una forma más segura de guardas las sesiones y tener mayor control sobre las sesiones abiertas.

5.3.1.2 Almacenamiento en el servidor

MySQL es un sistema de bases relacional ampliamente usado y nos hemos decantado por él como gestor de bases de datos para este proyecto porque:

- Es una base de datos gratuita de código abierto.
- Es fácil de usar.
- Es rápida.
- Utiliza varias capas de seguridad. Contraseñas encriptadas, derechos de acceso y privilegios para los usuarios.
- Pocos requerimientos y eficiencia de memoria.
- Es multiplataforma.
- Tiene mucha documentación y fiable.

5.3.2 Integración del SGBD en Nuestro Sistema

Spring Data es un módulo de Spring cuyo objetivo es facilitar el acceso y exploración de datos en aplicaciones basadas en Spring. Pudiendo ser obtenidos estos datos de fuentes tan diferentes como RestFul, bases de datos relacionales a través de JPA o bases de Datos NoSQL como MongoDB.

En este proyecto, se ha utilizado el módulo de JPA utilizándose como implementación Hibernate.

Con esto se logra reducir y simplificar el código de acceso a bases de datos. Si ya de por sí JPA nos facilita los accesos y la gestión de los datos, Spring Data genera automáticamente las operaciones CRUD sin necesidad de escribirlas para cada entidad JPA.

5.3.3 Diagrama E-R

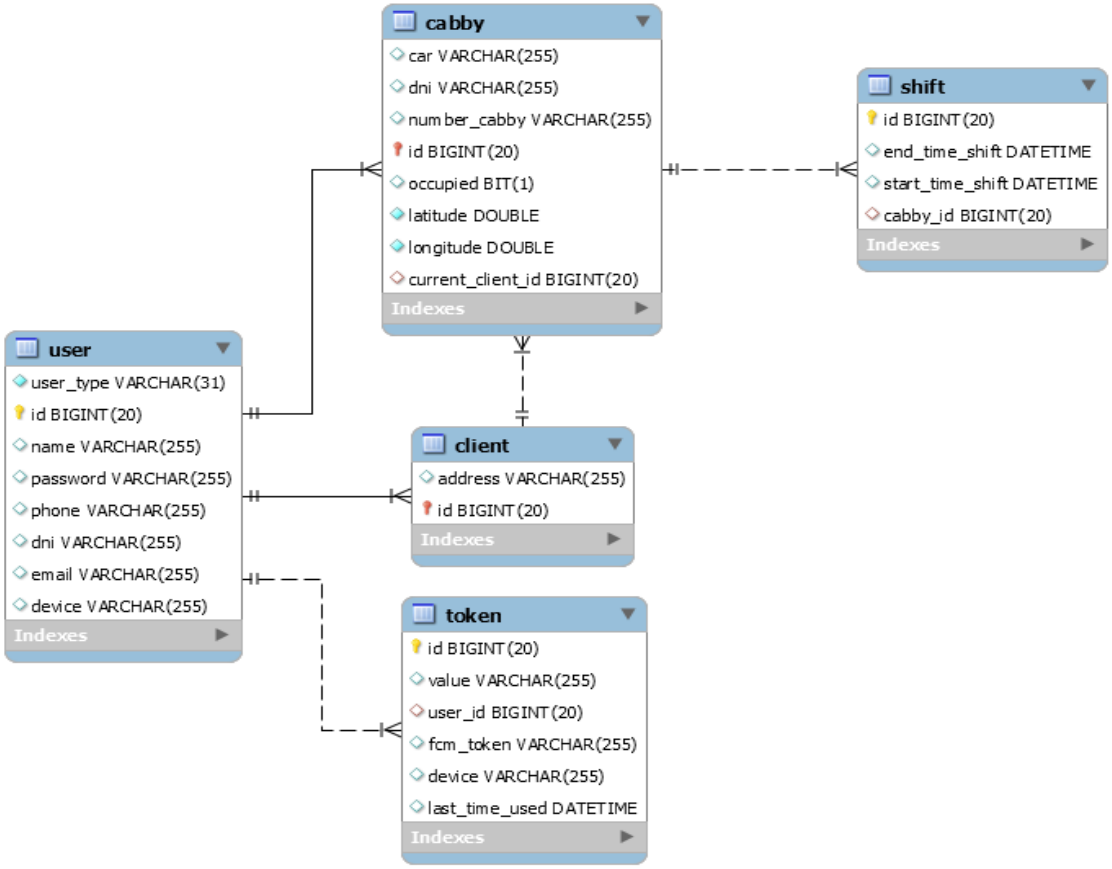


Ilustración 67 – Diagrama entidad relación

5.4 Diseño de la Interfaz

5.4.1 Diseño de la interfaz de Taxista

A continuación, se muestran las capturas de pantalla del resultado final del diseño de las mismas:

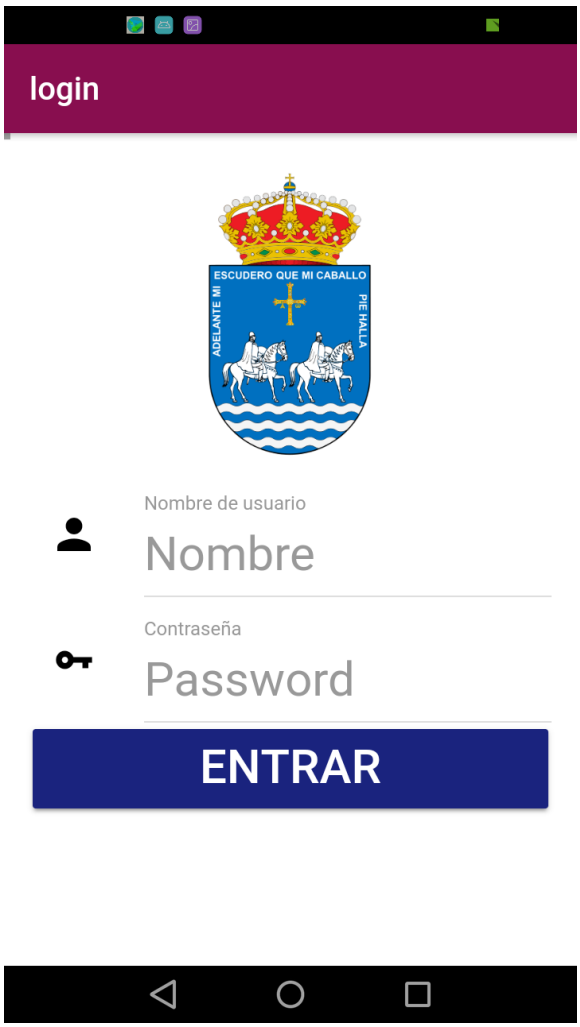


Ilustración 68- Pantalla de Login

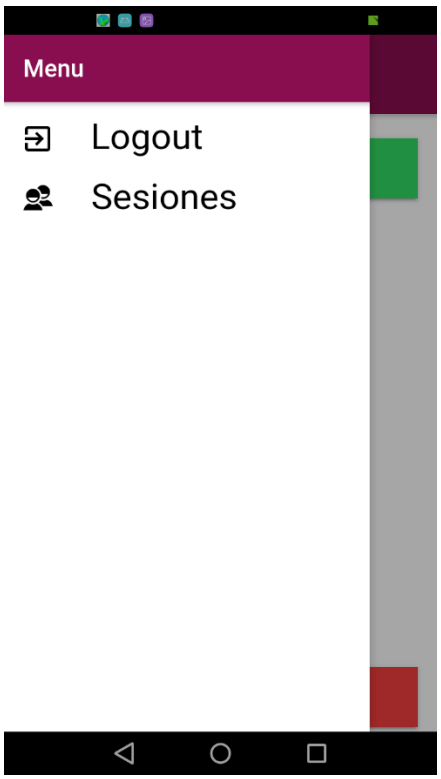


Ilustración 69- Pantalla menú

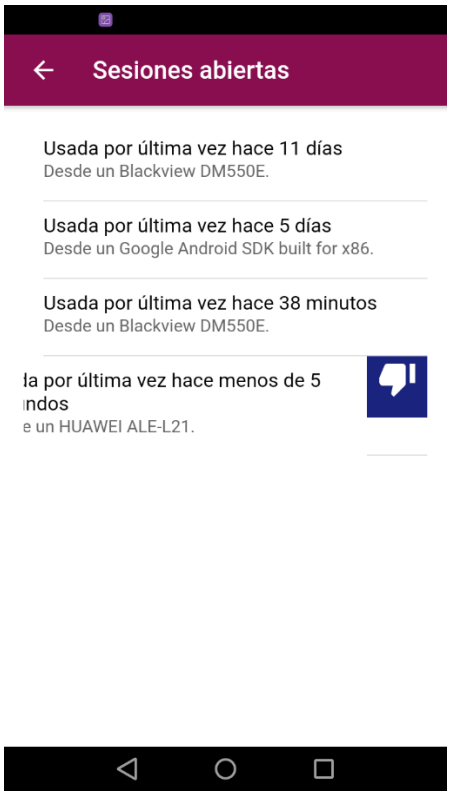


Ilustración 70- Pantalla sesiones abiertas

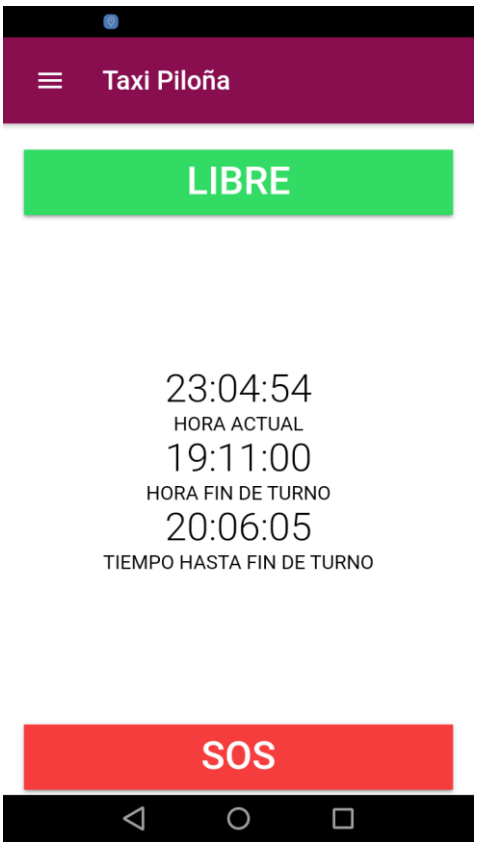


Ilustración 71- Pantalla principal

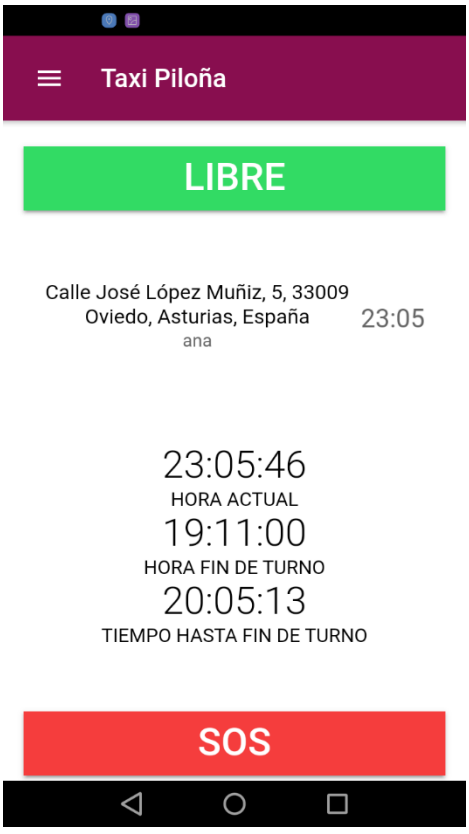


Ilustración 72- Pantalla principal con petición de taxi

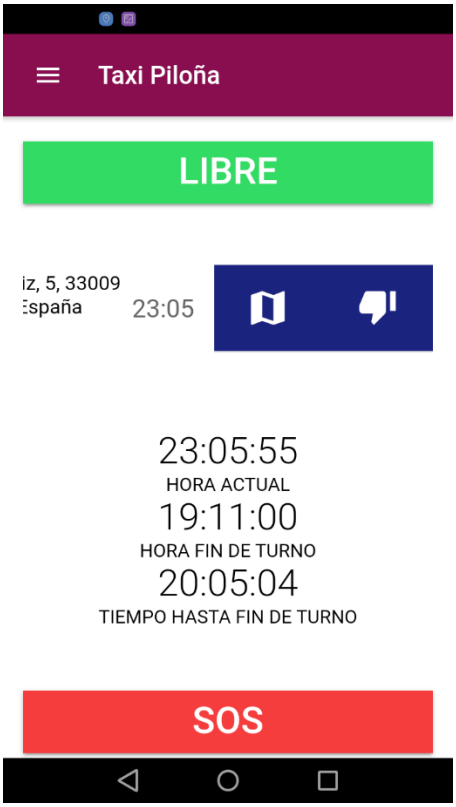


Ilustración 73Botones mapa y cancelar de petición de taxi

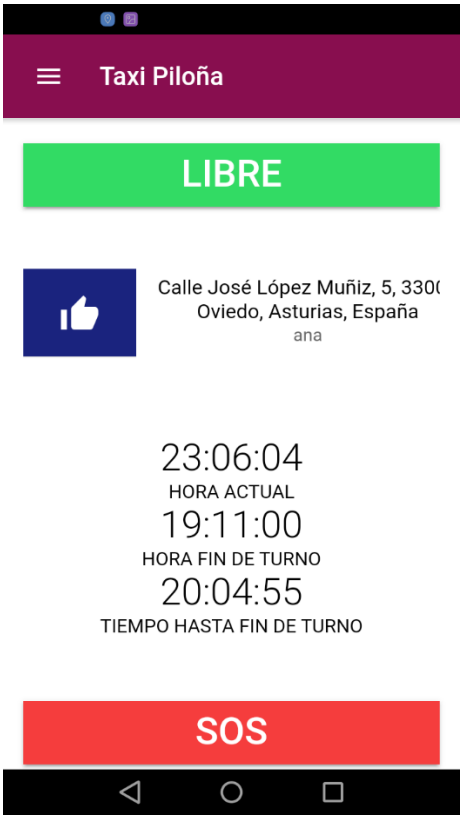


Ilustración 74- Botón aceptar petición de taxi

Ilustración 75- Mapa posición del cliente respecto taxista

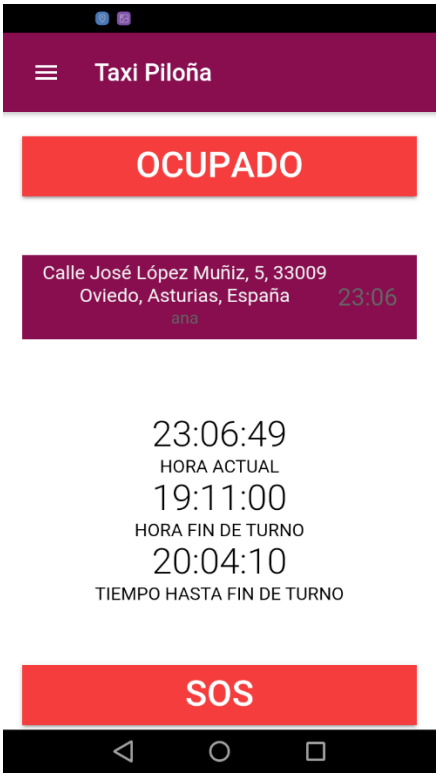


Ilustración 76- Pantalla principal petición de taxi en curso

Ilustración 77- Petición de SOS

5.4.2 Diseño de la interfaz de cliente

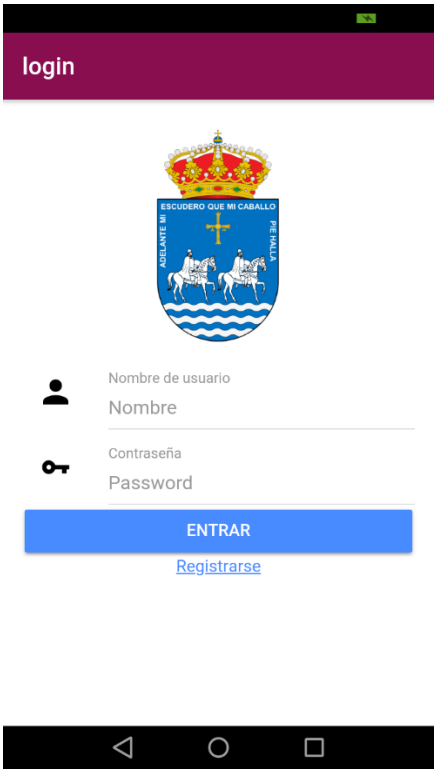


Ilustración 78 -Pantalla de Login

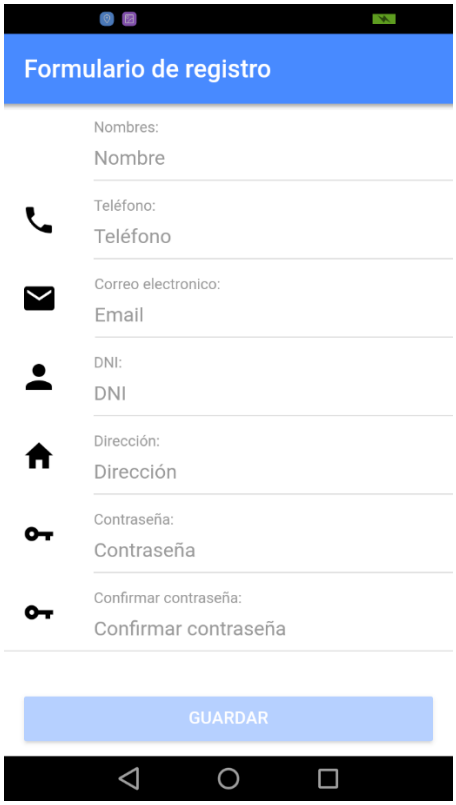


Ilustración 79-Pantalla de registro

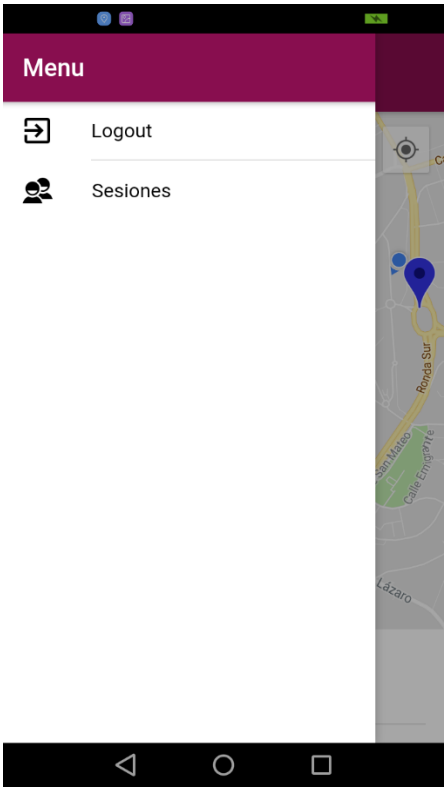


Ilustración 80- Menú

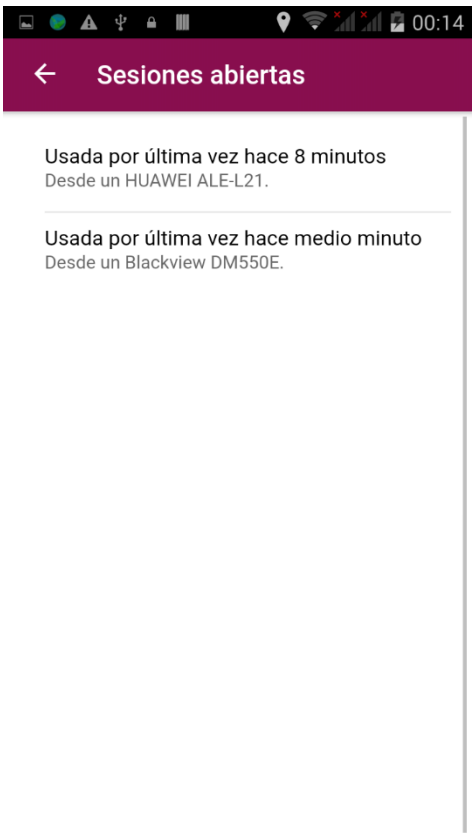


Ilustración 81- Sesiones abiertas

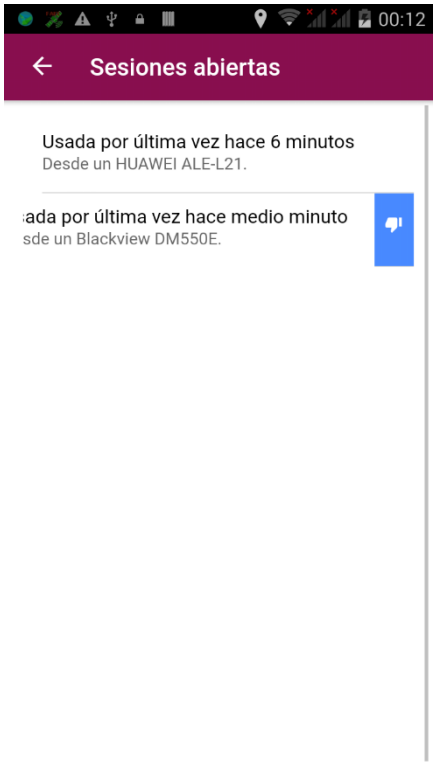


Ilustración 82- Cerrar sesión abierta

Ilustración 83 Pedir taxi

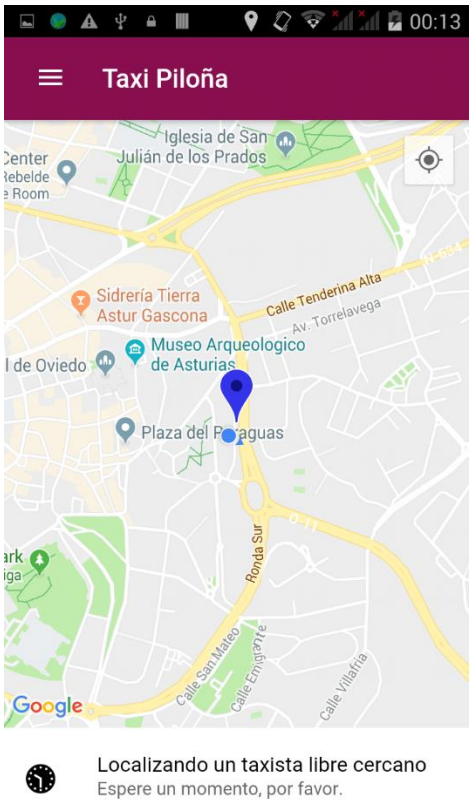


Ilustración 84- Localizando taxi libre cercano

Ilustración 85- Taxi en camino

Ilustración 86- No hay taxis disponibles

5.5 Especificación Técnica del Plan de Pruebas

5.5.1 Pruebas Unitarias

Para realizar los test unitarios en el servidor se utilizará Mockito y Junit.

Una prueba unitaria debe probar de forma aislada la funcionalidad de una unidad, sin los efectos secundarios de otras clases del sistema. Para eliminar esto se puede hacer utilizando reemplazos.

Mockito es un marco simulado que se puede usar con JUnit y permite crear y configurar objetos falsos. Su uso simplifica mucho el desarrollo de pruebas para clases con dependencias externas.

En caso de que los test no pasen, nos permitirán detectar errores y subsanarlos antes de que el código llegue a la fase de prueba por parte del cliente.

También permitirá detectar problemas en futuros cambios y ampliaciones de funcionalidades.

5.5.2 Pruebas de Integración y del Sistema

Se desplegará el servidor web en una máquina con una IP fija. Comprobando desde varios móviles que la funcionalidad es correcta:

Haciéndose:

- Pruebas de logueo y deslogueo.
- Conexión con varios dispositivos, cerrar sesión en dispositivos desde otro dispositivo.
- Comprobar que si se ha cerrado desde otro dispositivo cuando intentas volver a conectarte desde el primero no te deja conectarte.
- Hacer varias peticiones de taxi, aceptar, rechazar, mirar mapas.
- Petición de socorro.
- Comprobar que al aceptar una petición de taxi en la base de datos se asigna el cliente al taxista y se le cambia el estado.
- Comprobar que si el taxista cambia el estado al estado de ocupado sin cliente se hace correctamente.
- Comprobar que cuando se finaliza un taxista se ponga el taxista como libre en la base de datos y elimina la asignación al cliente.
- Comprobar que cuando te deslogueas se elimina el token de la base de datos.
- Comprobar que si te logueas como cliente en la aplicación de taxista te rechaza.
- Comprobar que si te logueas como taxista en una aplicación de cliente te rechaza.
- Registrarse y no poner todos los valores obligatorios.
- Registrarse correctamente.
- Loguearse con un usuario que no existe.
- Intentar loguearse sin poner contraseña.
- Intentar loguearse sin poner usuario.

5.5.3 Pruebas de Usabilidad y Accesibilidad

Las pruebas de usabilidad se realizarán con personas de diferentes perfiles (edad, nivel de estudios y género diferentes), y tras estas el prototipo pasará unos quince días en proceso de test, por los propios taxistas y quienes ellos consideren de la zona que pueden ser potenciales clientes, estos dispondrán de un cuestionario y aparte realizarán espontáneamente pruebas que ellos consideren oportunas, ya que la visión como usuario suele diferir de la de los especialistas pudiendo encontrar de esta manera dificultades que de manera guiada se nos pasarían por alto.

5.5.3.1 Actividades de las Pruebas de Usabilidad

5.5.3.1.1 Preguntas de carácter general

¿Usa el teléfono móvil frecuentemente?
1. Todos los días 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Utiliza aplicaciones específicas del Google Play Store?
1. Todos los días 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca
¿Qué tipo de aplicaciones?
1. Redes sociales 2. Ocio 3. Servicios 4. Varias de las anteriores
¿Qué busca Vd. Principalmente en una aplicación?
1. Que sea fácil de usar 2. Que sea intuitivo 3. Que sea rápido 4. Que tenga todas las funciones necesarias
¿Es usuario de taxis habitualmente?
1. Todos los días 2. Varias veces a la semana 3. Ocasionalmente 4. Nunca o casi nunca

5.5.3.1.2 Actividades guiadas

5.5.3.1.2.1 Usuarios aplicación cliente

Tarea	Tiempo en realizar la tarea	Errores	Observaciones
Registrase en el sistema			
Loguearse			
Ver las sesiones abiertas			
Pedir un taxi			
Desloguearse			

5.5.3.1.2.2 Usuarios aplicación taxista

Tarea	Tiempo en realizar la tarea	Errores	Observaciones
Loguearse			
Ver sesiones abiertas			
Ponerse como ocupado			
Aceptar petición de taxi			
Acceder al mapa			
Finalizar servicio			
Rechazar petición			
Acceder a mapa de la petición sin aceptar			
Pedir socorro			
Desloguearse			

5.5.3.1.3 Preguntas Cortas sobre la Aplicación y Observaciones

Facilidad de Uso	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Sabe dónde está dentro de la aplicación de cliente?				
¿Sabe dónde está siempre en la aplicación de taxista?				
¿Le resulta sencillo el uso de la aplicación de cliente?				
¿Le resulta sencillo el uso de la aplicación de taxista?				
Funcionalidad	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Funciona cada tarea como Vd. Espera?				
¿El tiempo de respuesta de la aplicación es muy grande?				
Calidad del Interfaz				
Aspectos gráficos	**Muy Adecuado**	**Adecuado**	**Poco Adecuado**	**Nada Adecuado**

El tipo y tamaño de letra es				
Los iconos e imágenes usados son				
Los colores empleados son				
Diseño de la Interfaz		**Si**	**No**	**A veces**
¿Le resulta fácil de usar?				
¿El diseño de las pantallas es claro y atractivo?				
¿Cree que el programa está bien estructurado?				
Observaciones				

5.5.3.1.4 Cuestionario para el responsable de las Pruebas

Aspecto Observado	**Notas**
El usuario comienza a trabajar de forma rápida por las tareas	
Tiempo en realizar cada tarea	
Errores leves cometidos	
Errores graves cometidos	
Necesita indicaciones para conseguir finalizar alguna tarea	

Los resultados de las pruebas permitirán valorar:

1. **Facilidad de aprendizaje**: Los usuarios se encuentran cómodos con las pantallas y encuentran de manera fácil los elementos necesarios para realizar las tareas de una manera adecuada, sin necesidad de indicaciones para finalizar estas y midiendo el tiempo empleado para llevar a cabo cada una.
2. **Eficiencia**: La petición es más rápida que la realización de una llamada a la centralita. Aquí no se puede medir hasta la fase de test en campo, que la elección del taxista es más adecuada que la de la centralita.
3. **Errores**: Cuantos errores cometen los usuarios en las distintas tareas, lo que decrementa la usabilidad de este.
4. **Satisfacción del usuario**: Impresión general de los usuarios al usar la aplicación.

5.5.4 Pruebas de accesibilidad

Debido a que el prototipo va a pasar por un periodo de test por el usuario final, estas pruebas se llevarán a cabo en ese ámbito, cumpliendo también con los requerimientos que el cliente crea oportunos.

5.5.5 Pruebas de Rendimiento

Para realizar las pruebas de rendimiento se va a utilizar Apache Jmeter, que es una aplicación de código abierto en java que permite hacer test. Es una herramienta que suele usarse para hacer pruebas de carga, aunque también soporta aserciones.

Capítulo 6. Implementación del Sistema

6.1 Estándares y Normas Seguidos

El proyecto sigue las convenciones de código y guías de estilo de Java, así como las de TypeScript.

También se utiliza el estándar de JSON para el intercambio de datos.

6.2 Lenguajes de Programación

6.2.1 Java

Java es un lenguaje de programación orientada a objetos ampliamente conocido. Este lenguaje ha ido evolucionando a lo largo del tiempo debido a los cambios en el entorno informático y a la mejora y evolución de la forma de programar.

En este proyecto se ha utilizado Java 8, que en esencia sigue siendo el mismo, pero a su vez difiere mucho de aquel Java 5, que estudie en la diplomatura de Ingeniería técnica informática en esta misma universidad.

Basándose en su herencia de C y C++, Java añade mejoras y funciones que reflejan el estado actual de la programación. Como respuesta a un aumento de los entornos online, Java ofrece características que racionalizan la programación de una arquitectura distribuida.

Java fue una idea de James Gosling, Patrick Naughton, Chris Warth, Ed Frank y Mike Sheridan de Sun Microsystems en 1991. Inicialmente se llamaba Oak y no comenzó a denominarse tal y como lo conocemos ahora hasta 1995.

Aunque hoy en día resulte curioso, el objetivo de Java inicialmente no era Internet, sino un lenguaje independiente de las plataformas para crear software para diferentes dispositivos. Pero mientras se ultimaban los detalles de Java, surgió Internet, que desempeñaría un papel fundamental en el futuro de Java: la Web.

Java contribuyó a simplificar la programación Web y presentó un tipo de programa de red, el applet, que cambio la forma de pensar en el contenido online. Un applet es un tipo especial de programa que está diseñado para su transmisión por internet y su ejecución automática en navegadores Web compatibles con Java. Estos applets están confinados al entorno de ejecución de Java impidiendo su acceso a otras partes del equipo.

Las aplicaciones de Java se compilan generalmente a bycode(clase Java) que puede ejecutarse en cualquier máquina virtual Java (JVM) , sin importar la arquitectura que la contenga. Como el proceso de compilación no genera código para un tipo específico de procesador, permite ejecutarlo en cualquier plataforma sin necesidad de modificar el código fuente ni recompilar.

Entre sus características destacaríamos que es un lenguaje simple, ya que dispone de un conjunto conciso y coherente de características que hace que sea fácil su aprendizaje y uso. Es seguro para crear aplicaciones en Internet, portable, pudiendo ejecutarse como hemos comentado en cualquier entorno que disponga de un sistema de tiempo de ejecución de Java. Aplica la filosofía de la programación orientada a objetos, es robusto con tipos estrictos y comprobaciones en tiempo de ejecución. Ofrece compatibilidad con programación de subprocesamiento múltiple. Es neutral arquitectónicamente hablando, ya que no se limita a un equipo o sistema operativo concreto como ya se ha mencionado. Es un lenguaje interpretado que permite código compatible entre plataformas gracias al uso del código de bytes. Proporciona un alto rendimiento ya que, este último (código de bytes), esta optimizado para acelerar la ejecución. Está orientado al entorno distribuido de Internet y por último vamos a destacar que es un lenguaje dinámico que incluye información en tiempo de ejecución que se usa para verificar y resolver el acceso a objetos en tiempo de ejecución.

6.2.2 TypeScript

Es un lenguaje de programación de código abierto con herramientas de programación orientada a objetos, desarrollado por Microsoft. Anders Hejlsberg, uno de los arquitectos principales en el desarrollo de C#, es el principal participante en el desarrollo de este lenguaje que convierte su código en Javascript común.

Se dice también que es un Superset de Javascript como ya hemos comentado anteriormente. Es decir, si el navegador está basado en Javascript, éste ejecutará el JavaScript como lenguaje original sin saber que el código original está en TypeScript.

Angular 2, uno de los frameworks más famosos de JavaScript, está siendo desarrollado en TypeScript.

Principalmente se caracteriza por implementar las ventajas de un lenguaje orientado a objetos y las ventajas de JavaScript, permitiéndonos usar ambos a la vez.

Algunas de las características de este lenguaje son:

- Cada programa de TypeScript tiene su correspondencia en JavaScript y viceversa.
- Opcionalmente permite un mapeado de las fuentes que permite su depuración.

- Incluye propuestas de ECMAScript en su versión 6, como las clases y los módulos.
- Define declaraciones ambientes que son visibles en el ámbito del archivo ts, pero no se propaga al archivo Js correspondiente.
- Es un lenguaje débilmente tipado, permitiendo declaraciones en tiempo de ejecución.
- Un módulo es totalmente anónimo, todo lo que se encuentra en su interior permanece oculto a menos que de manera explícita se desee que no lo sea.
- Una interfaz es un conjunto de declaraciones de tipo, sin implementaciones de contenido. Son de validez exclusiva en tiempo de compilación y su utilidad estriba en la documentación y validación del tipado.
- El soporte de clases abarca herencia, accesibilidad pública y privada y sobrecarga.

6.3 Herramientas y Programas Usados para el Desarrollo

6.3.1 Netbeans

Es un entorno de desarrollo principalmente hecho en Java. Existe un gran número de módulos para extenderlo. Es de código abierto, libre y gratuito sin restricciones de uso y estoy familiarizada con su uso.

6.3.2 Visual Studio Code

Es un editor de código fuente desarrollado por Microsoft y está muy preparado para TypeScript, quizá porque como hemos comentado antes Microsoft está tras el desarrollo de este lenguaje. Es una herramienta muy potente, es gratuita, de código abierto y dispone de gran variedad de plugins.

6.3.3 MySql Wokbench

Es una herramienta visual de diseño de bases de datos que integra desarrollo de software, administración de bases de datos diseño de las mismas y gestión y mantenimiento para sistemas de bases de datos MySQL.

6.3.4 Restlet Client

Facilita el desarrollo de servicios REST ya que permite comprobar su correcto funcionamiento sin necesidad de implementar el cliente real.

6.4 Creación del Sistema

6.4.1 Problemas Encontrados

Principalmente nos hemos encontrado con dos problemas destacables:

6.4.1.1 Error cross-site

Para el desarrollo era conveniente utilizar la posibilidad de la depuración en vivo, permite que mientras vas modificando el código se vean reflejados automáticamente los dichos cambios en el móvil, ya que facilita y ahorra mucho tiempo a la hora del desarrollo.

El problema es que la aplicación desplegada en el teléfono, que utiliza un navegador para ejecutar el código javascript y HTML del proyecto sobre él, necesita acceder a dos servidores diferentes. Uno es el empleado para obtener las nuevas versiones del código del proyecto, y el otro es el servicio REST. El navegador impide este acceso al considerarlo un problema de seguridad (acceso cross site).

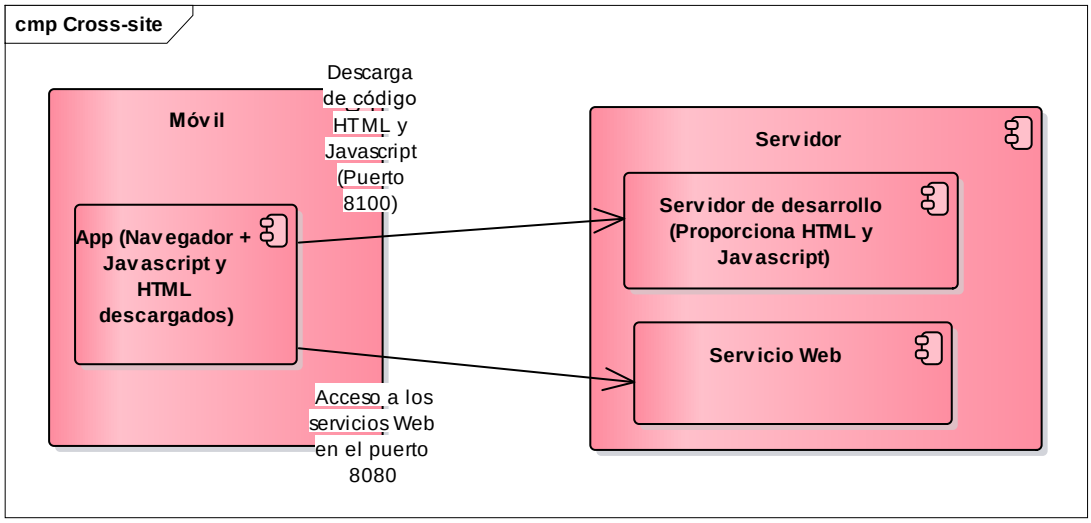


Ilustración 87- Problema Cross-site

Se ha solucionado usando una funcionalidad que ofrece ionic para el desarrollo que consiste en el uso de un proxy para el acceso a servicios web cuando se utiliza la depuración en vivo.

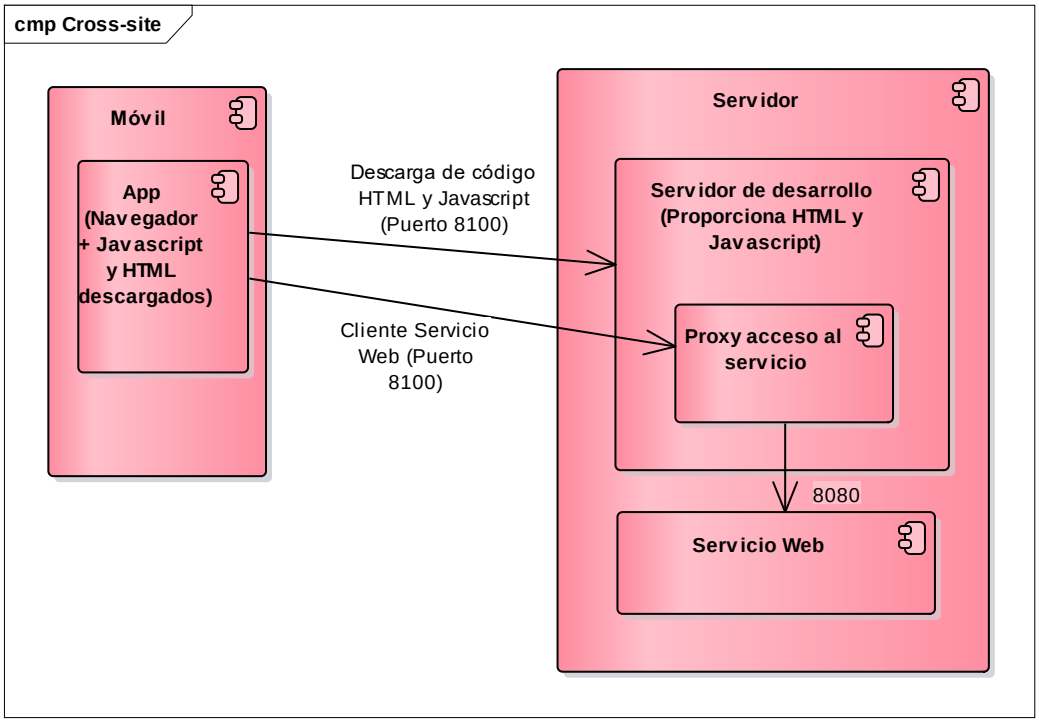


Ilustración 88 – Acceso a los servicios con un proxy para la depuración en vivo

6.4.1.2 Problemas de compatibilidad de plugins

El plugin de FCM resultó ser incompatible con el plugin de Google Maps, esto se debía a que estaba obsoleto y no era compatible con la última versión del sdk de Android provocando incompatibilidades con otros plugins.

Se solucionado desinstalando este plugin y utilizando el de Firebase, que nos ofrecía las mismas funcionalidades y esta actualizado.

6.4.2 Descripción detallada de las clases

Se adjunta a la documentación en una carpeta “Descripción clases”, el javadoc y los typedoc de las aplicaciones.

Capítulo 7. Desarrollo de las Pruebas

7.1 Pruebas Unitarias

7.1.1 Resultados test: CabbyService

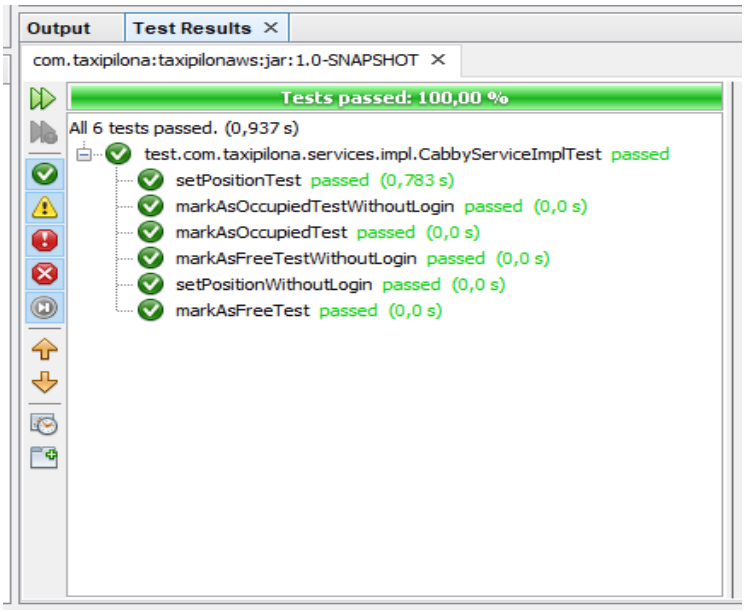


Ilustración 89 – Resultado test unitarios: CabbyService

7.1.2 Resultados test: ClientService

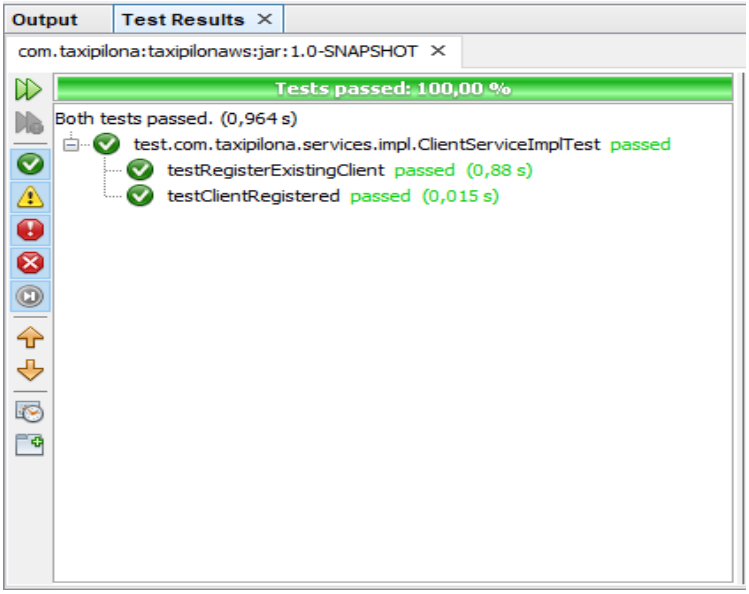


Ilustración 90 – Resultado test unitarios: ClientService

7.1.3 Resultados test: LoginService

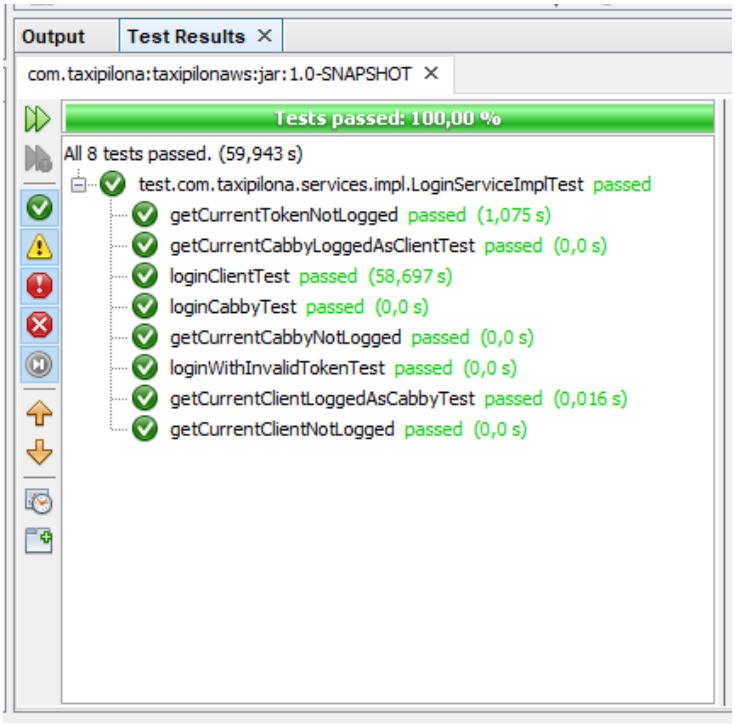


Ilustración 91 – Resultado test unitarios: LoginService

7.1.4 Resultados test: ShiftService

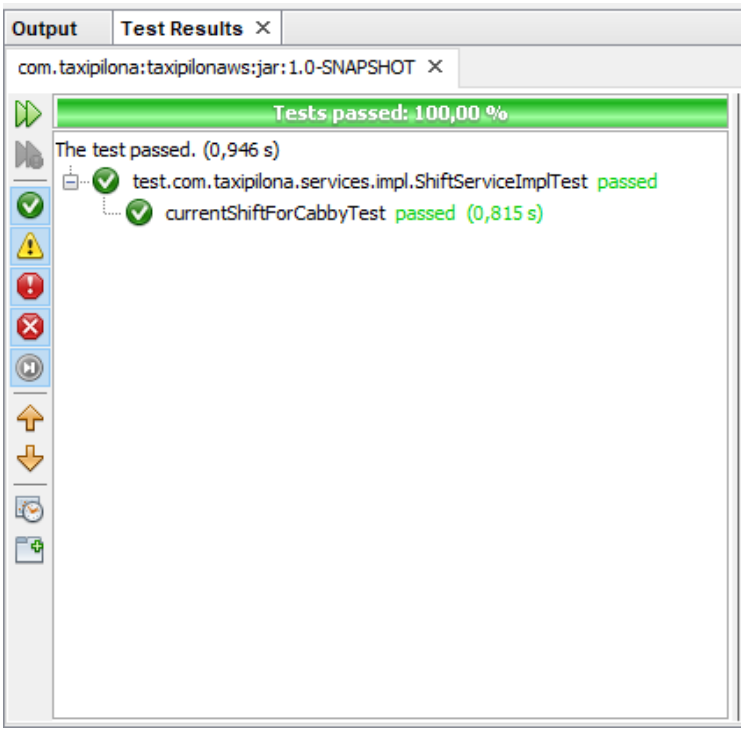


Ilustración 92- Resultados test unitarios: ShiftService

7.1.5 Resultados test: TokenService

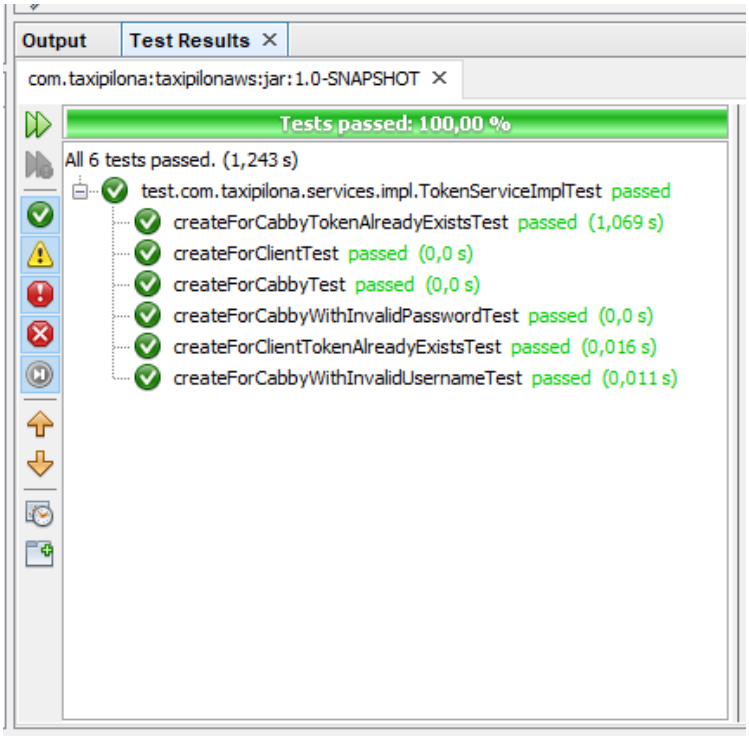


Ilustración 93 – Resultados test unitarios: Token Service

7.1.6 Resultados test: Cabby

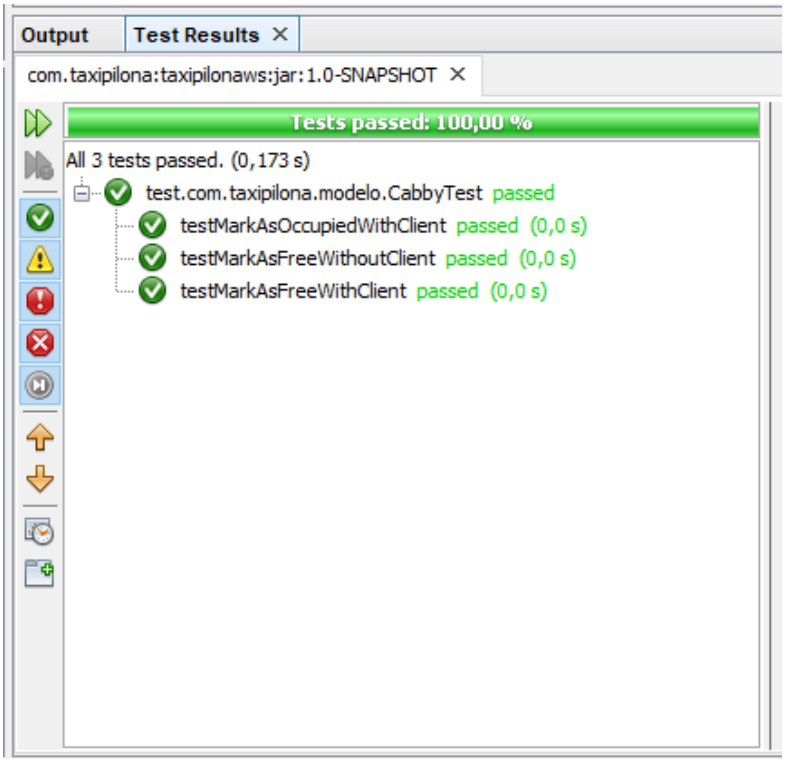


Ilustración 94 – Resultados test unitarios: Cabby

7.1.7 Resultados test: LoginData

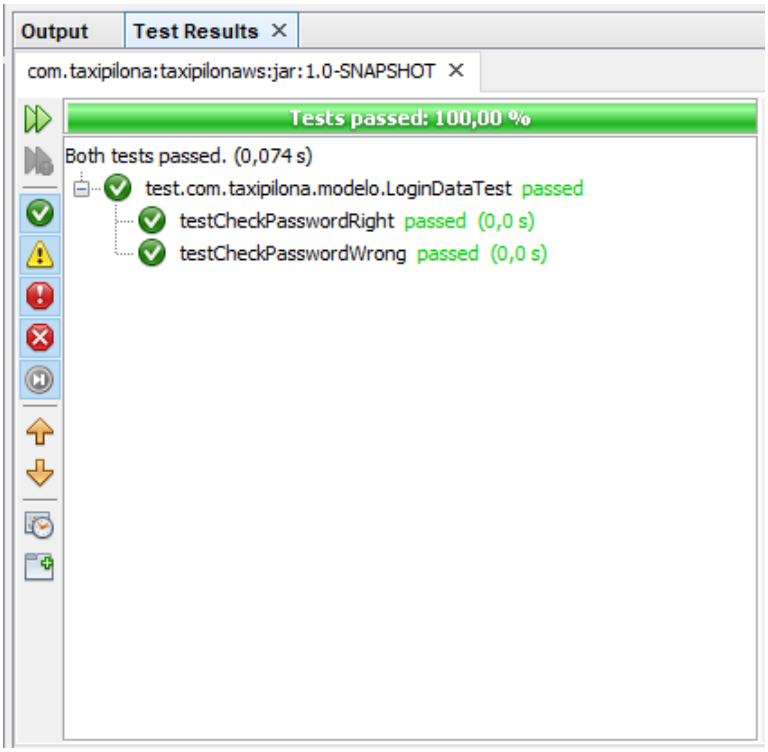


Ilustración 95 – Test unitarios:LoginData

7.1.8 Resultados de test: PositionData

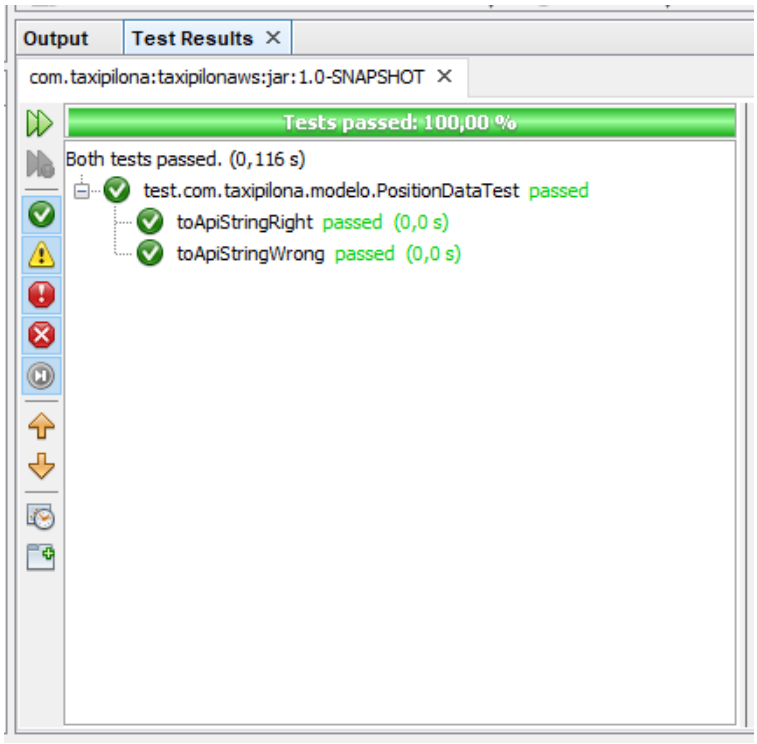


Ilustración 96 – Resultados test unitarios: PositionData

7.1.9 Resultados de Test: Token

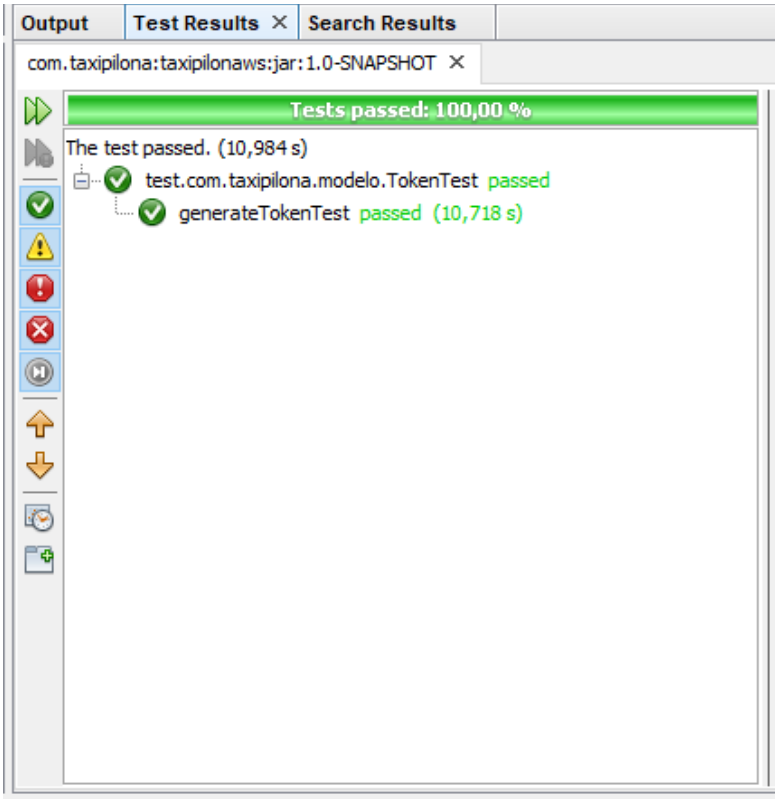


Ilustración 97 – Test unitarios : Token

7.2 Pruebas de Integración y del Sistema

Caso de Uso 1.1: Añadir Usuario Cliente mediante el registro	
Prueba	**Resultado Esperado**
Añadir un usuario no existente	El sistema posee un usuario más
	Resultado Obtenido
	El sistema efectivamente posee un usuario más
Prueba	**Resultado Esperado**
Añadir un usuario con un Login que ya existe	No se añade el usuario al sistema y se muestra un mensaje de error.
	Resultado Obtenido
	El usuario no está en la base de datos y ha aparecido el mensaje de error.

Caso de Uso 1.2: Loguearse	
Prueba	**Resultado Esperado**
Loguearse en la aplicación de cliente con un usuario que existe y es cliente	Se accederá a la página de Home.
	Resultado Obtenido
	Se accede a la página de Home.
Prueba	**Resultado Esperado**
Loguearse en la aplicación de taxista con un usuario que existe y es taxista	Se accederá a la página de Home.
	Resultado Obtenido
	Se accede a la página de Home.
Prueba	**Resultado Esperado**
Loguearse en la aplicación de cliente con un usuario que existe y es taxista	No nos dejará loguearnos en el sistema.
	Resultado Obtenido
	No nos deja loguearnos en el sistema.
Prueba	**Resultado Esperado**
Loguearse en la aplicación de taxista con un usuario que existe y es cliente	No nos dejará loguearnos en el sistema.
	Resultado Obtenido
	No nos deja loguearnos en el sistema.
Prueba	**Resultado Esperado**
Loguearse en la aplicación de taxista con un usuario que no existe	No nos dejará loguearnos en el sistema.
	Resultado Obtenido
	No nos deja loguearnos en el sistema.
Prueba	**Resultado Esperado**
Loguearse en la aplicación de cliente con un usuario que existe	No nos dejará loguearnos en el sistema.
	Resultado Obtenido
	No nos deja loguearnos en el sistema.

Caso de Uso 1.3: Pedir taxi	
Prueba	**Resultado Esperado**
Cliente pide un taxi para una ubicación concreta	Aparecerá en la pantalla del taxista la petición para esa ubicación. Aparecerá en la pantalla del cliente que se está procesando su petición.
	Resultado Obtenido
	Aparece la petición para el taxista en esa ubicación. Aparece en la pantalla de cliente el mensaje de que se está buscando taxi.
Prueba	**Resultado Esperado**
El taxista rechaza la petición de taxi.	Desaparece de la pantalla de taxista la petición. Aparecerá en la petición del cliente que no hay taxis en ese momento que lo intente más tarde.
	Resultado Obtenido
	Se le informa al cliente de que no se le puede atender en ese momento. Desaparece la petición de la pantalla del taxista.
Prueba	**Resultado Esperado**
El taxista acepta petición	Se quedará fijada la petición en la pantalla del taxista. Aparecerá el mensaje en la pantalla del cliente de que el taxista está en camino y su posición en el mapa. Aparecerá en la base de datos el cliente asociado al taxista.
	Resultado Obtenido
	Aparece la petición en otro color y fijada en la pantalla del taxista. Al cliente se le muestra el aviso de que hay un taxista en camino y su posición en el mapa. Aparece en la base de datos el cliente asociado al taxista.
Prueba	**Resultado Esperado**
Finaliza servicio	Desaparecerá la petición de la pantalla del taxista. Se borrará el cliente como asociado al taxista en la base de datos.
	Resultado Obtenido
	El cliente ya no aparece asociado al taxista en la base de datos. El mensaje a desaparecido.
Prueba	**Resultado Esperado**
Comprobar mapa en petición de taxi pendiente.	Se podrá ver la posición del cliente en un mapa.
	Resultado Obtenido
	Se puede ver la posición del cliente en un mapa.
Prueba	**Resultado Esperado**
Comprobar mapa en petición aceptada de taxi.	Se podrá ver la posición del cliente en un mapa.
	Resultado Obtenido
	Se ve la posición del cliente en un mapa.
Prueba	**Resultado Esperado**
Petición de taxi no hay taxis libres	Se le indicará al cliente que no se puede atender su petición en ese momento.

	Resultado Obtenido
	Se indica al cliente que no se puede atender su petición.

Caso de Uso 1.4: Comprobar turno	
Prueba	**Resultado Esperado**
Se deja al taxista sin turnos	En la pantalla aparecerá sólo la hora actual.
	Resultado Obtenido
	Aparece la hora actual
Prueba	**Resultado Esperado**
Se le añade varios turnos al taxista y uno engloba la hora actual.	Aparecerá en la pantalla la hora actual, la de inicio de turno y lo que falta para acabar el turno.
	Resultado Obtenido
	Aparece hora actual, la de inicio de turno y lo que falta para acabar el turno.
Prueba	**Resultado Esperado**
Se añade varios turnos, pero ninguno engloba la hora actual.	Sólo aparecerá la hora actual en la pantalla del taxista.
	Resultado Obtenido
	Aparece sólo la hora actual en la pantalla.

Caso de Uso 1.5: Pedir SOS	
Prueba	**Resultado Esperado**
Pulsar el botón de SOS	Aparecerá en la pantalla de otros taxistas información sobre el taxista y su ubicación.
	Resultado Obtenido
	Aparece el número de taxista y el coche del mismo en la pantalla y su ubicación en un mapa.

Caso de Uso 1.6: Sesiones abiertas	
Prueba	**Resultado Esperado**
Comprobamos las sesiones abiertas en la aplicación de taxista	Se mostrarán las mismas en la pantalla que hay en la base de datos. Son correctas.
	Resultado Obtenido
	Las sesiones abiertas son correctas. Son las mismas que hay en la base de datos.
Prueba	**Resultado Esperado**
Comprobamos las sesiones abiertas en la aplicación de cliente	Se mostrarán las mismas en la pantalla que hay en la base de datos. Son correctas.
	Resultado Obtenido
	Las sesiones abiertas son correctas. Son las mismas que hay en la base de datos.
Prueba	**Resultado Esperado**
Cerramos una sesión abierta en la aplicación de cliente	El token asociado a esta sesión desaparecerá de la base de datos. Se ha cerrado la sesión en el dispositivo.

	Resultado Obtenido
	Ha desaparecido el token en la base de datos y en el dispositivo cuando vas a la aplicación te aparece la página de Login.
Prueba	**Resultado Esperado**
Cerramos una sesión abierta en la aplicación de taxista	El token asociado a esta sesión desaparecerá de la base de datos. Se ha cerrado la sesión en el dispositivo.
	Resultado Obtenido
	Ha desaparecido el token en la base de datos y en el dispositivo cuando vas a la aplicación te aparece la página de Login.

Caso de Uso 1.7: Logout	
Prueba	**Resultado Esperado**
Nos deslogueamos de la aplicación de cliente	Nos saldremos de la pantalla de Home. Se borrará el token en la base de datos.
	Resultado Obtenido
	La aplicación se sale de la pantalla de Home. En la base de datos ya no está el token para este dispositivo.
Prueba	**Resultado Esperado**
Nos deslogueamos de la aplicación de taxista	Nos saldremos de la pantalla de Home. Se borrará el token en la base de datos.
	Resultado Obtenido
	La aplicación se sale de la pantalla de Home. En la base de datos ya no está el token para este dispositivo.

7.3 Pruebas de Usabilidad y Accesibilidad

7.3.1 Pruebas de Usabilidad

Para realizar las pruebas de usabilidad se ha contado con 10 personas de entre 16 y 65 años.

A todos ellos se les ha pedido realizar las mismas tareas guiadas para ver si la aplicación es usable intuitiva o por el contrario crea alguna confusión o problema para llevar a cabo dichas tareas.

A continuación, se muestran los resultados medios de cada prueba.

7.3.1.1 Preguntas de carácter general

¿Usa el teléfono móvil frecuentemente?	
1. Todos los días	80%
2. Varias veces a la semana	10%
3. Ocasionalmente	10%
4. Nunca o casi nunca	
¿Utiliza aplicaciones específicas del Google Play Store?	
5. Todos los días	30%
1. Varias veces a la semana	40%
2. Ocasionalmente	10%
3. Nunca o casi nunca	20%
¿Qué tipo de aplicaciones?	
1. Redes sociales	20%
2. Ocio	
3. Servicios	
4. Varias de las anteriores	60%
¿Qué busca Vd. Principalmente en una aplicación?	
5. Que sea fácil de usar	
6. Que sea intuitivo	
7. Que sea rápido	
8. Que tenga todas las funciones necesarias	100%
¿Es usuario de taxis habitualmente?	
5. Todos los días	
6. Varias veces a la semana	
7. Ocasionalmente	80%
8. Nunca o casi nunca	20%

7.3.1.2 Actividades guiadas

7.3.1.2.1.1 Usuarios aplicación cliente

Tarea	Tiempo en realizar la tarea (segundos)	Errores	Observaciones
Registrase en el sistema	85	No	
Loguearse	48	No	
Ver las sesiones abiertas	160	No	A las personas más mayores que no utilizan el móvil habitualmente les ha costado más darse cuenta de que está en el menú.
Pedir un taxi	25	No	Estamos en el mismo caso que el anterior, les llevo un poco más de tiempo a las personas menos familiarizadas con los dispositivos y aplicaciones móviles.
Desloguearse	40	No	

7.3.1.2.1.2 Usuarios aplicación taxista

Tarea	Tiempo en realizar la tarea	Errores	Observaciones
Loguearse	30	No	
Ver sesiones abiertas	100	No	
Ponerse como ocupado	Inmediato	No	
Aceptar petición de taxi	180	No	Algunos han tenido que recibir información para darse cuenta de dónde estaba. El manual de usuario para taxista va a ser útil y necesario.
Acceder al mapa	160	No	Algunos han tenido que recibir información para darse cuenta de dónde estaba. El manual de usuario para taxista va a ser útil y necesario.
Finalizar servicio	50	No	
Rechazar petición	45	No	

Acceder a mapa de la petición sin aceptar	75	No	
Pedir socorro	Inmediato	No	
Desloguearse	52	No	

7.3.1.2.2 Preguntas Cortas sobre la Aplicación y Observaciones

Facilidad de Uso	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Sabe dónde está dentro de la aplicación de cliente?	100%			
¿Sabe dónde está siempre en la aplicación de taxista?	80%	20%		
¿Le resulta sencillo el uso de la aplicación de cliente?	100%			
¿Le resulta sencillo el uso de la aplicación de taxista?		100%		
Funcionalidad	**Siempre**	**Frecuentemente**	**Ocasionalmente**	**Nunca**
¿Funciona cada tarea como Vd. Espera?		100%		
¿El tiempo de respuesta de la aplicación es muy grande?				100%

Calidad del Interfaz				
Aspectos gráficos	**Muy Adecuado**	**Adecuado**	**Poco Adecuado**	**Nada Adecuado**
El tipo y tamaño de letra es	100%			
Los iconos e imágenes usados son	80%	20%		
Los colores empleados son	80%	20%		
Diseño de la Interfaz		**Si**	**No**	**A veces**
¿Le resulta fácil de usar?		100%		
¿El diseño de las pantallas es claro y atractivo?		100%		
¿Cree que el programa está bien estructurado?		100%		
Observaciones				

Tras la realización de las pruebas, hemos comprobado que la aplicación es intuitiva y usable, además de atractiva y sencilla. Para mejorar esto tras las pruebas, en un inicio el menú era un deslizable hacia la derecha y se ha puesto arriba en la barra de menú también.

En cuanto a la dificultad para encontrar en un primer momento el menú de los botones deslizables no es preocupante ya que es la aplicación de taxista y no la de cliente, el cual va a tener un manual de usuario a su disposición.

7.4 Pruebas de Rendimiento

Para realizar las pruebas de rendimiento, como se ha indicado antes se utiliza la aplicación Apache Jmeter. La petición de taxi ha de comprobarse de forma empírica ya que depende de muchos factores, entre ellos cuánto tarde en aceptar o rechazar el taxista.

En todas las pruebas se han lanzado 20 hilos en bucles de 50 iteraciones.

Con lo que tras ver a la sobrecarga que se ha sometido, creemos que el sistema responde de manera adecuada.

7.4.1 Pruebas de rendimiento: Siguiente turno

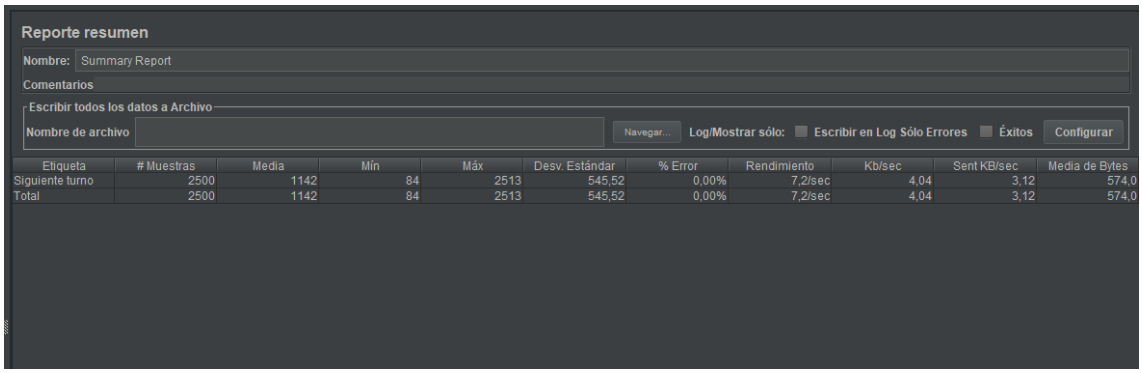


Ilustración 98 – Informe resumen pruebas rendimiento siguiente turno

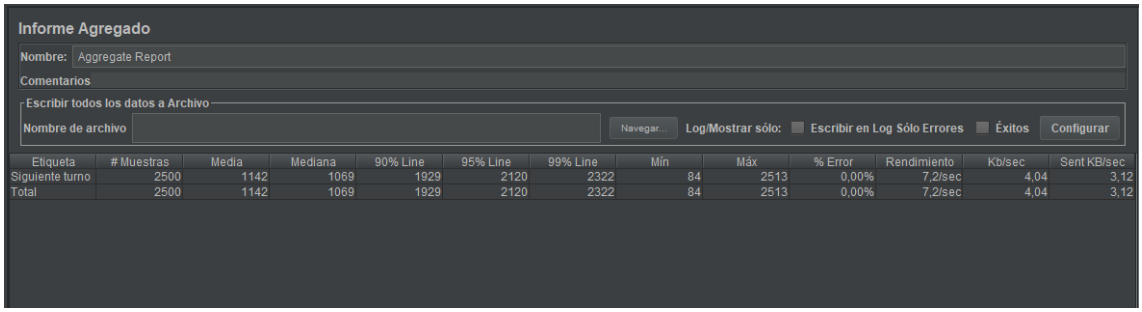


Ilustración 99 – Informe agregado pruebas de rendimiento siguiente turno

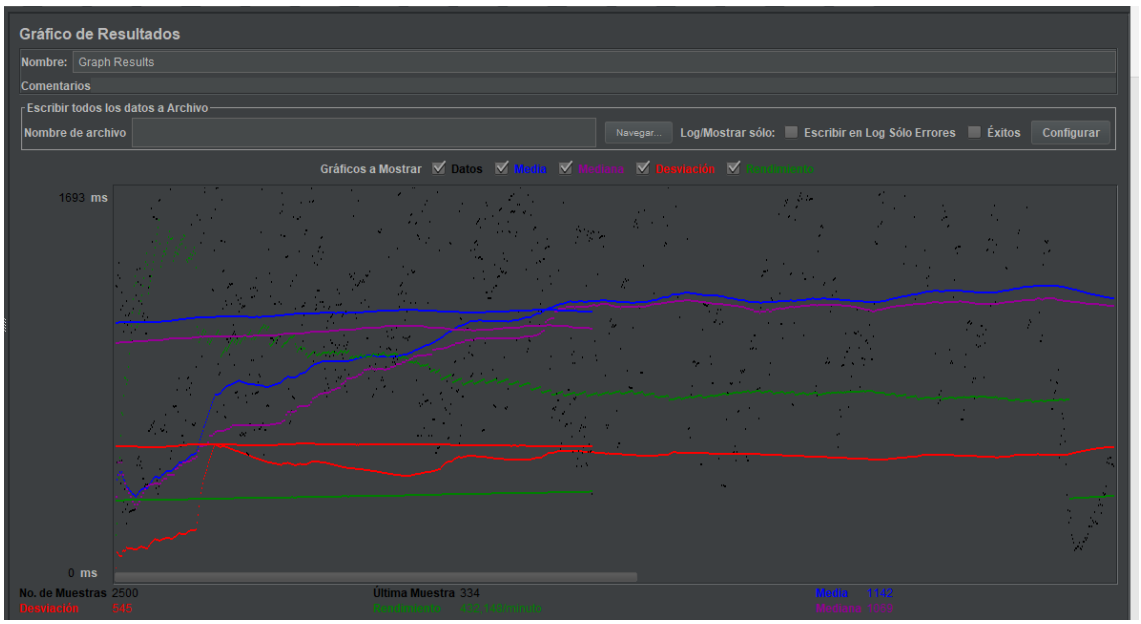
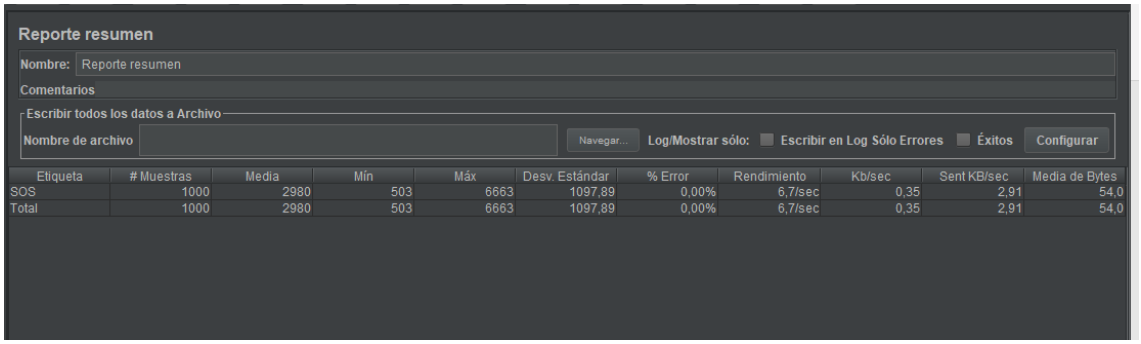


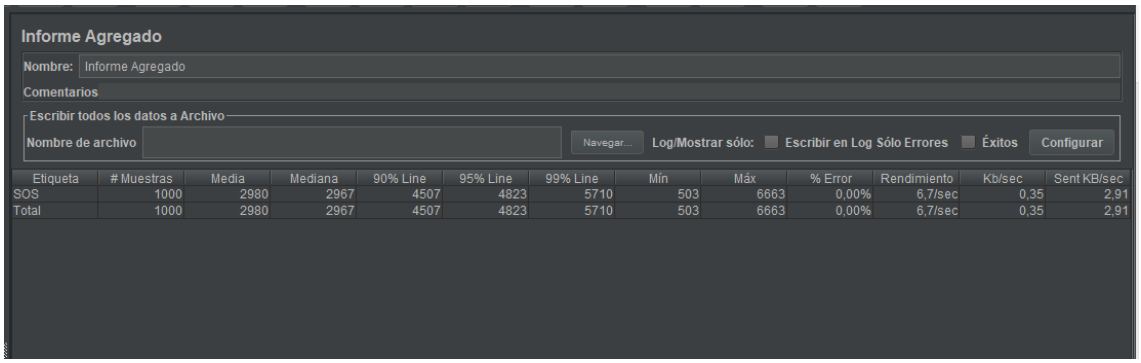
Ilustración 100 – Gráfico de resultados pruebas de rendimiento siguiente turno

7.4.2 Pruebas de rendimiento SOS



Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec	Media de Bytes
SOS	1000	2980	503	6663	1097,89	0,00%	6,7/sec	0,35	2,91	54,0
Total	1000	2980	503	6663	1097,89	0,00%	6,7/sec	0,35	2,91	54,0

Ilustración 101 – Informe resumen resultados pruebas de rendimiento SOS



Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimiento	Kb/sec	Sent KB/sec
SOS	1000	2980	2967	4507	4823	5710	503	6663	0,00%	6,7/sec	0,35	2,91
Total	1000	2980	2967	4507	4823	5710	503	6663	0,00%	6,7/sec	0,35	2,91

Ilustración 102 – Informe agregado resultado pruebas de rendimiento SOS

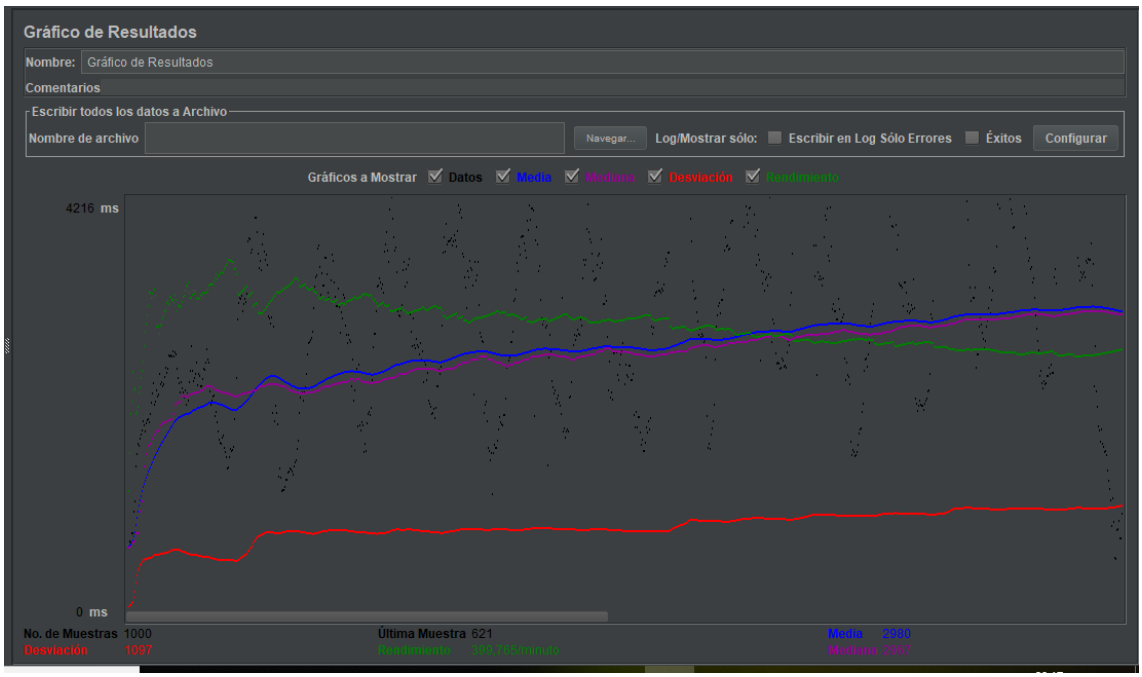
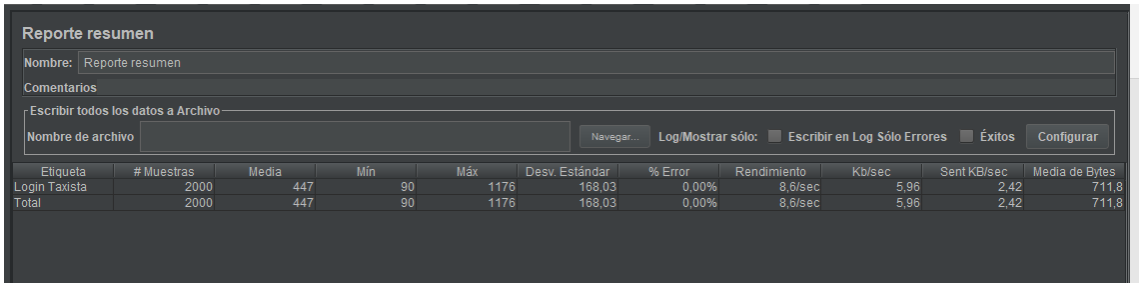


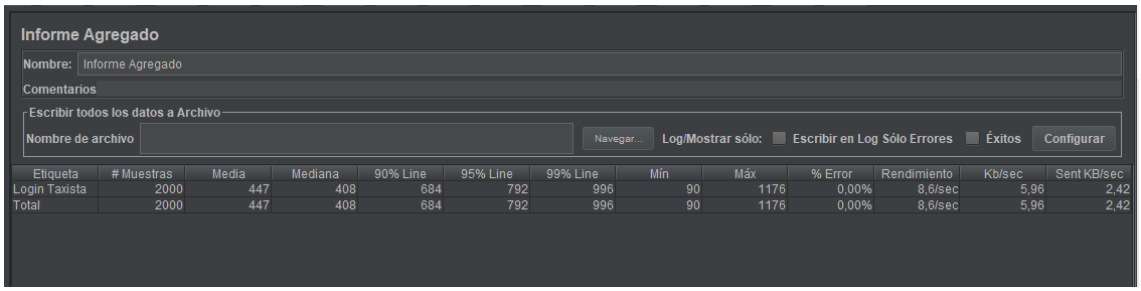
Ilustración 103- Gráfico resultados pruebas de rendimiento SOS

7.4.3 Pruebas de rendimiento Login taxista



Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec	Media de Bytes
Login Taxista	2000	447	90	1176	168,03	0,00%	8,6/sec	5,96	2,42	711,8
Total	2000	447	90	1176	168,03	0,00%	8,6/sec	5,96	2,42	711,8

Ilustración 104 – Informe resumen pruebas de rendimiento Login taxista.



Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimiento	Kb/sec	Sent KB/sec
Login Taxista	2000	447	408	684	792	996	90	1176	0,00%	8,6/sec	5,96	2,42
Total	2000	447	408	684	792	996	90	1176	0,00%	8,6/sec	5,96	2,42

Ilustración 105 -Informe agregado pruebas de rendimiento Login taxista

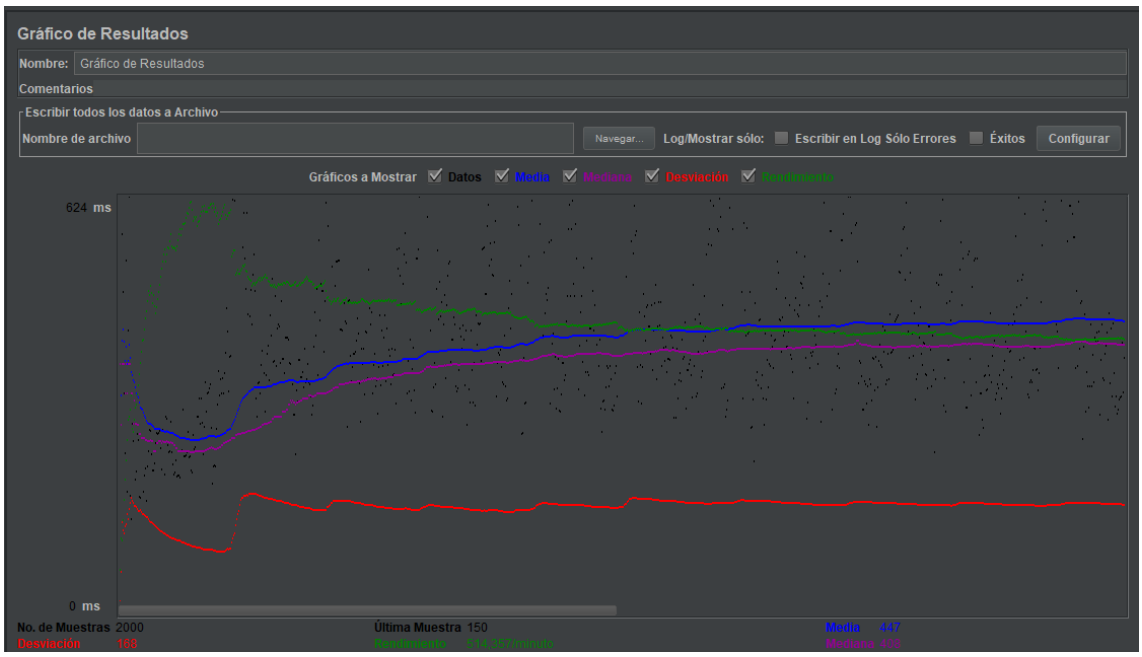
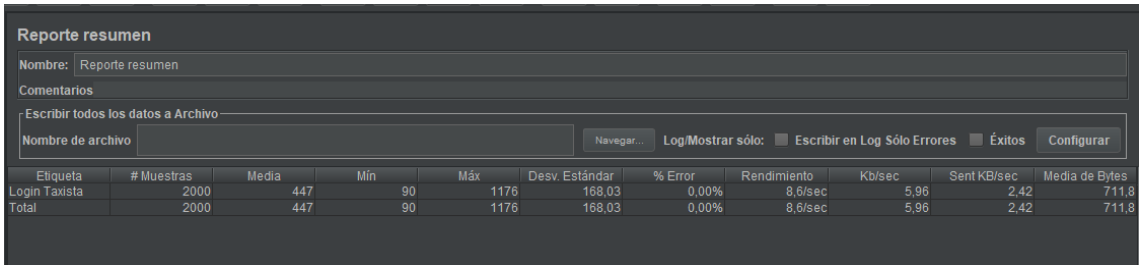


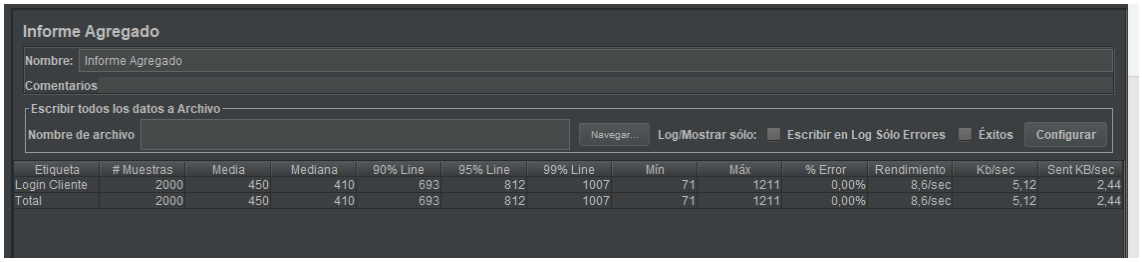
Ilustración 106 – Gráfico resultados pruebas de rendimiento Login taxista

7.4.4 Pruebas de rendimiento Login cliente



Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec	Media de Bytes
Login Taxista	2000	447	90	1176	168,03	0,00%	8,6/sec	5,96	2,42	711,8
Total	2000	447	90	1176	168,03	0,00%	8,6/sec	5,96	2,42	711,8

Ilustración 107-Informe resumen pruebas de rendimiento Login cliente



Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimiento	Kb/sec	Sent KB/sec
Login Cliente	2000	450	410	693	812	1007	71	1211	0,00%	8,6/sec	5,12	2,44
Total	2000	450	410	693	812	1007	71	1211	0,00%	8,6/sec	5,12	2,44

Ilustración 108- Informe agregado pruebas de rendimiento Login de cliente

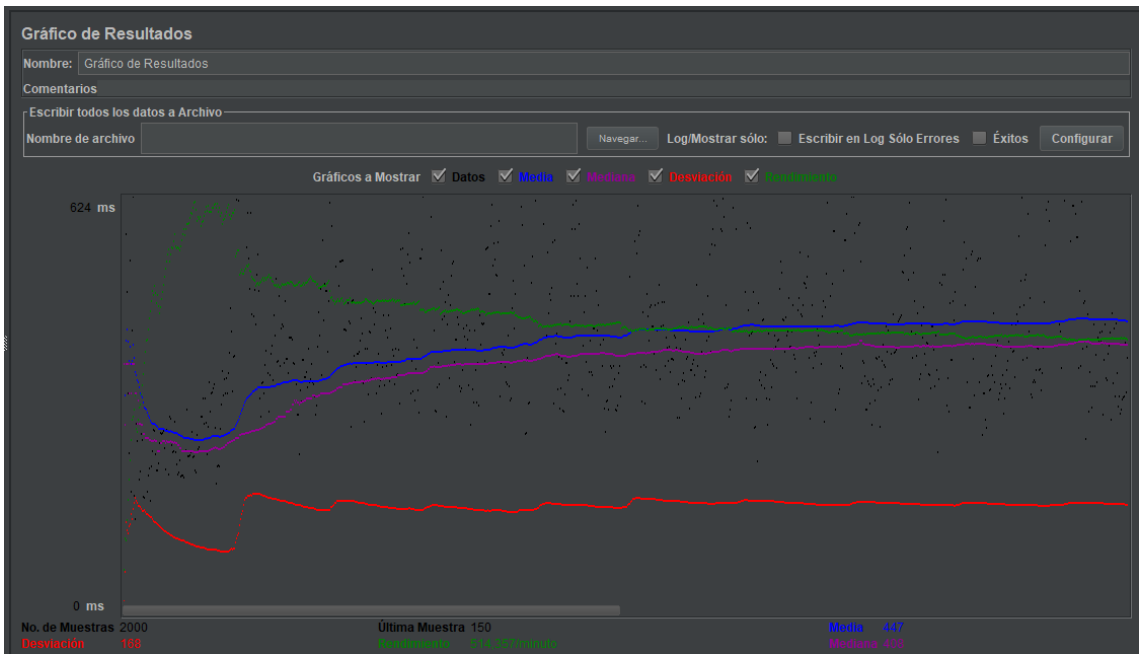


Ilustración 109 – Gráfico resultados pruebas rendimento Login de cliente

Capítulo 8. Manuales del Sistema

8.1 Manual de Instalación

Se debe tener instalado MySQL 8.0 y el Workbench. Se adjunta junto a la documentación en una carpeta “Estructura base de datos” el dump de la base de datos.

En la pantalla de Restore del Workbench cargamos los ficheros.

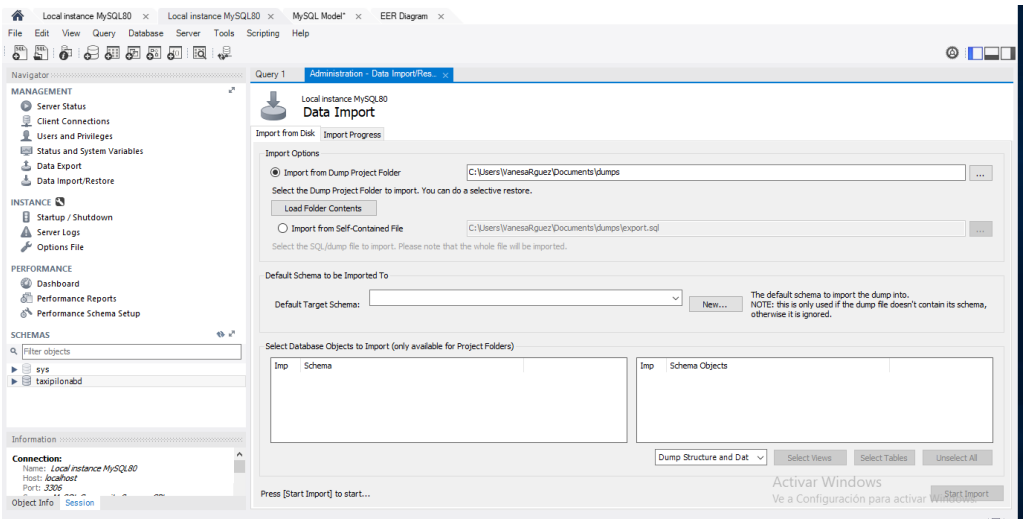


Ilustración 110- Pantalla Restore del Workbench

Se arranca el fichero .jar que contiene el servidor, con la base de datos arrancada.

8.2 Manual de Ejecución

En la ejecución lo único que hay que tener en cuenta es que hay que arrancar primero la base de datos y después la aplicación. En las aplicaciones móviles no es necesario más que descargárselas desde el Store.

8.3 Manual de Usuario

Estos manuales no serán los definitivos, ya que, tras la fase de test, se ampliará la funcionalidad y se modificarán aspectos que el cliente crea necesarios.

8.3.1 Manual de usuario taxista

En primer lugar y hasta que se construya la página web que va a gestionar el administrador, los taxistas los da de alta con el Workbench dicho administrador. Al ser una persona con amplios conocimientos informáticos no es necesario un manual a este respecto.

8.3.1.1 Loguearse en el sistema

El taxista le aparecerá en primer lugar al ejecutar la aplicación la pantalla de Login.

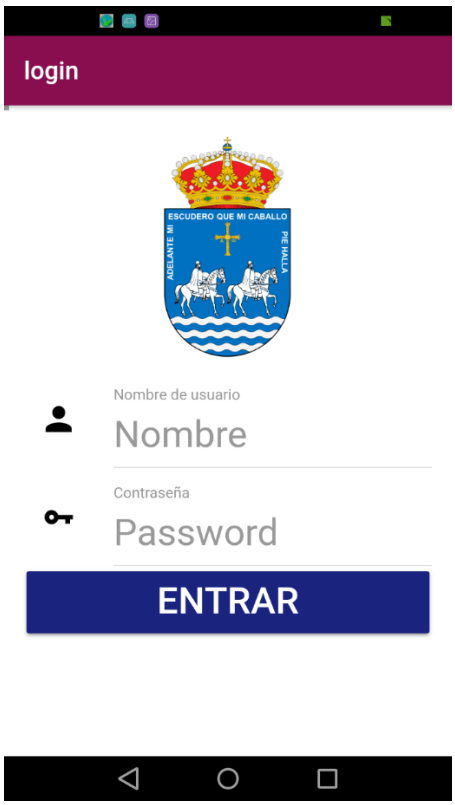


Ilustración 111- Pantalla de Login

Tran loguearse de manera correcta en el sistema aparecerá la pantalla principal en la cual se realizarán la mayor parte de las acciones de la aplicación.

8.3.1.2 Pantalla principal: libre/ocupado, turnos y SOS

Esta pantalla dispone de un botón Libre/Ocupado. Pulsándolo automáticamente el taxista cambiará su estado.

Un botón de SOS, en caso de un peligro con sólo pulsarlo, su número de taxista, coche que conduce y posicionamiento en un mapa les aparecerá a sus compañeros.

Y por último un reloj central con la hora actual, la hora de inicio de su turno y cuánto tiempo falta para su finalización. En caso de no estar dentro de un turno, sólo aparecerá la hora actual.

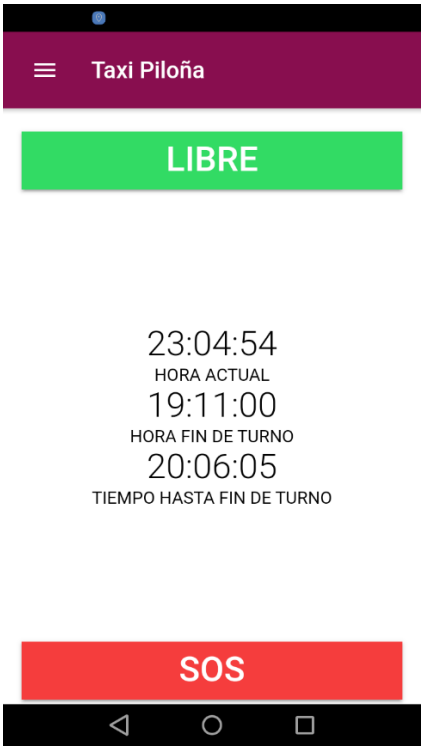


Ilustración 112- Pantalla principal taxista

Arriba a la izquierda se dispone de un menú, que aparte de pulsando el icono superior aparece deslizando con el dedo lateralmente hacia la derecha.

8.3.1.3 Menú: logout y sesiones abiertas

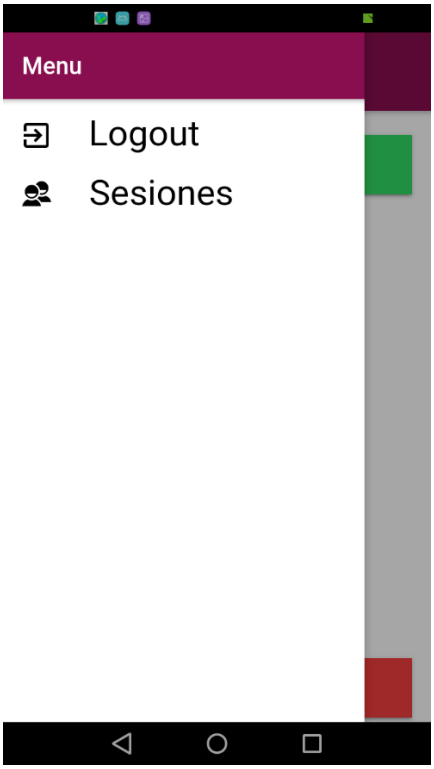


Ilustración 113- Menú

En este menú el usuario puede desloguearse y comprobar que sesiones tiene abiertas. Teniendo la posibilidad de cerrar alguna si lo desea.

Para cerrar una sesión sólo tiene que poner el dedo encima de la que desea eliminar deslizarlo a la izquierda y pulsar el botón habilitado para esta funcionalidad, lo distinguirá porque aparte de ser el único botón que hay, tiene un dedo pulgar hacia abajo dibujado.

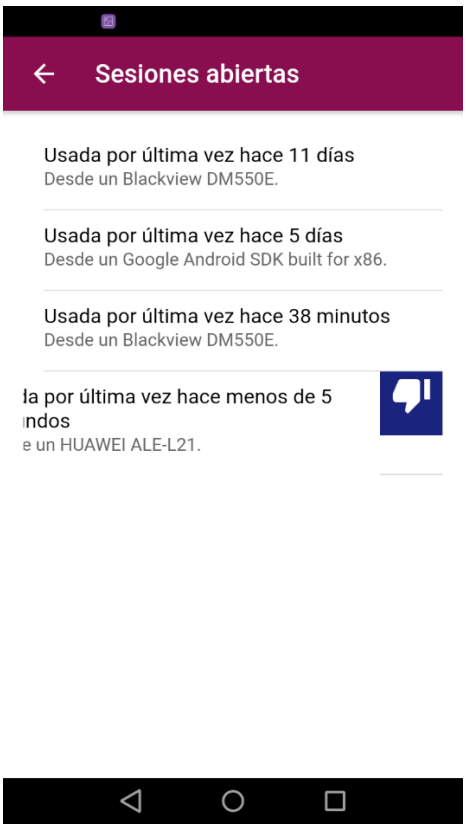


Ilustración 114- Sesiones abiertas

8.3.1.4 Pantalla de SOS

En caso de que algún compañero se encuentre en peligro le aparecerá la siguiente pantalla en su móvil interrumpiendo cualquier acción que esté realizando.

Es una pantalla con la ubicación del taxista en peligro en un mapa, así como su número de taxista y coche que está utilizando en el turno actual.

Ilustración 115- Pantalla de SOS

8.3.1.5 Petición de servicio

Las peticiones de servicio de los clientes se listarán en la pantalla principal.

Usted deslizando el dedo sobre la petición a la izquierda podrá ver en un mapa, cuál es la ubicación para la que se solicita el servicio, y rechazar la misma. Si desea aceptarla deberá deslizar la petición hacía la derecha.

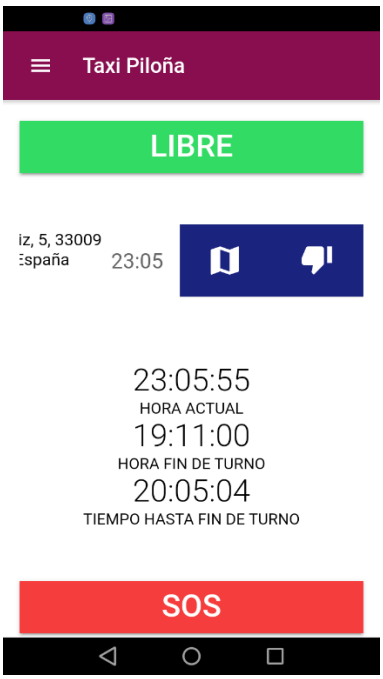


Ilustración 116- Petición de taxi, mapa y denegar

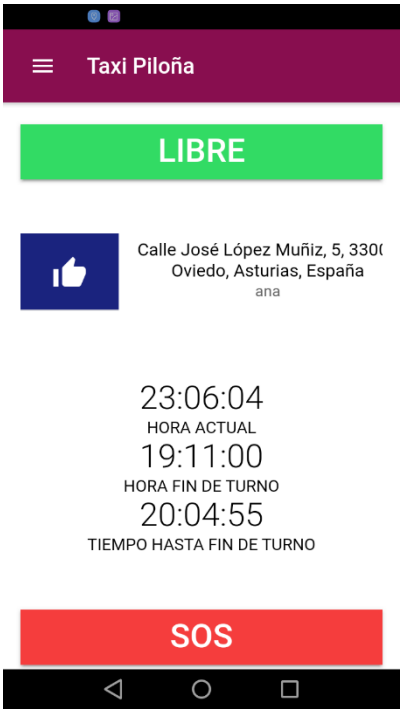


Ilustración 117- Petición de taxi : aceptar

En el momento que acepte una petición de taxi su estado pasará inmediatamente a ocupado.

Una vez finalizado el servicio sólo tendrá que pulsar el botón y pasará a estar libre para recibir nuevas peticiones.

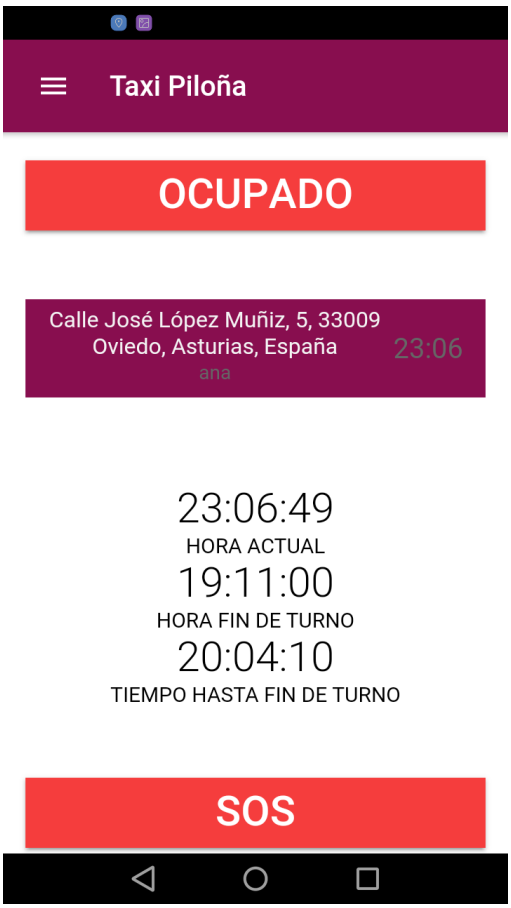


Ilustración 118- Petición de taxi aceptada

La petición de taxi aceptada sólo dispondrá de la opción de ver mapa, que se accederá a ella por un botón que aparecerá deslizando la misma hacia la izquierda.

En cuanto al mapa que usted podrá ver tanto en el botón antes de aceptarla como en el de después, aparecerá la posición de recogida del cliente, así como su propia posición.

Ilustración 119- Pantalla mapa

8.3.2 Manual de usuario de cliente

8.3.2.1 Registrarse y loguearse

El cliente para entrar en el sistema debe primero registrarse en la pantalla de registro, a esta se accede por un enlace en la pantalla de Login.

Tanto desde la pantalla de Login como desde la de registro , se accede a la pantalla principal.

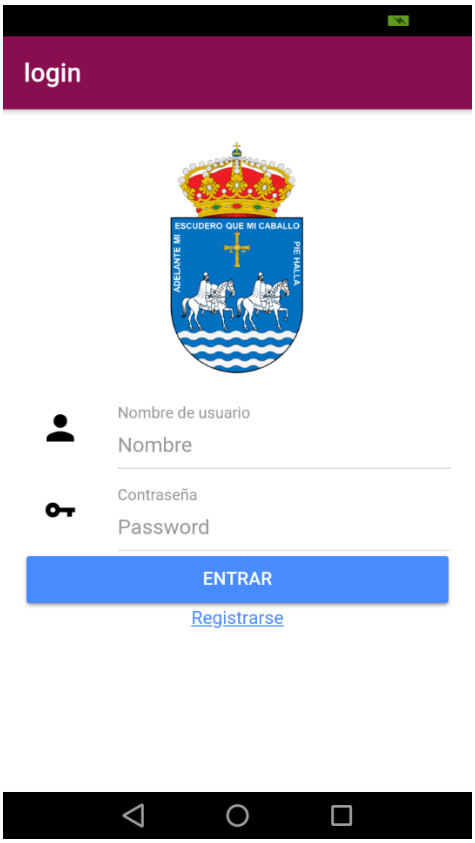


Ilustración 120- Pantalla de Login y enlace a registro

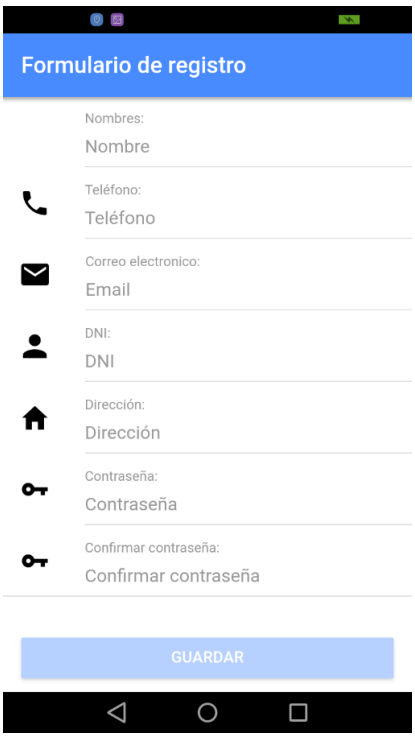


Ilustración 121- Pantalla de registro

8.3.2.2 Petición de taxi

Una vez dentro del sistema, podrá ver un mapa en el cual moviendo la marca que aparece centrada en su posición hacia donde quiere que el taxi le recoja y pulsando el botón de "Pedir taxi", su solicitud comenzará a gestionarse.

Ilustración 122- Pantalla pedir taxi

En todo momento usted sabrá el estado de su solicitud. Mientras el sistema busca un taxista cercano a la ubicación de recogida deseada, en la parte inferior de la pantalla le aparecerá un mensaje con esta información.

Ilustración 123 – Petición de taxi en curso

En caso de que no haya taxistas disponibles, se le informará en la misma ubicación dentro de la pantalla, disponiendo de un botón para cancelar la petición.

Ilustración 124 – Taxista no disponible

En caso de que un taxista acepte la petición. Se le informará en la parte baja de la pantalla de que su taxista está en camino y cual es el coche que va a recogerle, además, aparecerá la posición el mismo en el mapa y podrá hacer un seguimiento del progreso del mismo hacía usted.

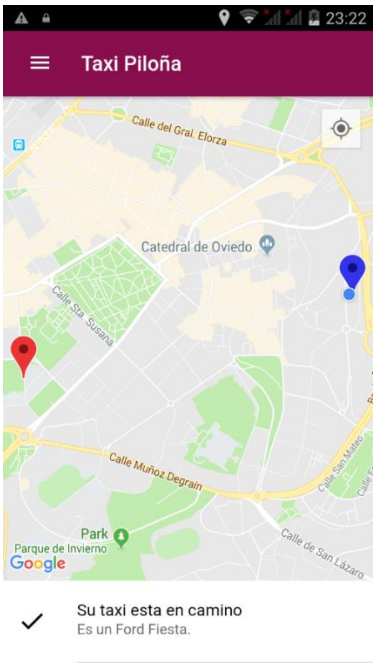


Ilustración 125- Taxi en camino

8.3.2.3 Sesiones abiertas y logout

En el menú que aparecerá deslizando con el dedo desde el lateral izquierdo hacia la derecha o pulsando las tres líneas que se encuentran en la barra superior. En el está la opción de desloguearse y ver las sesiones abiertas.

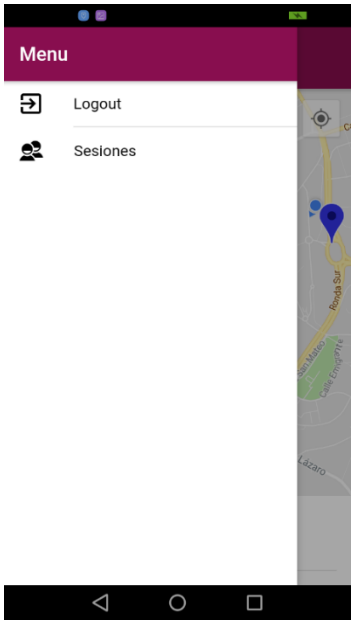


Ilustración 126- Menú

Las sesiones abiertas pueden eliminarse deslizando hacia la izquierda la que desea desconectar y pulsando el botón habilitado para este uso.

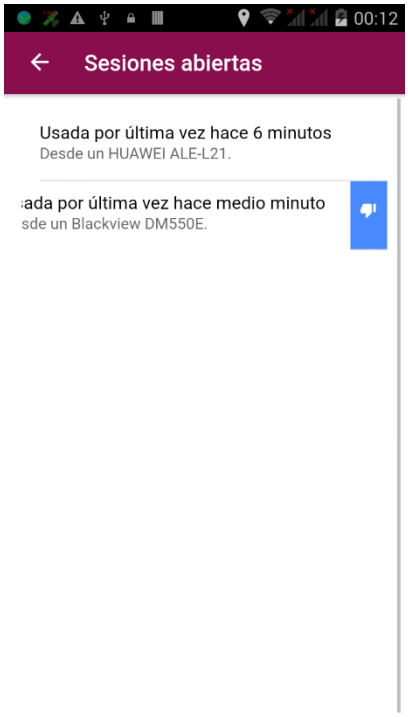


Ilustración 127- Sesiones abiertas

Capítulo 9. Conclusiones y Ampliaciones

9.1 Conclusiones

El proyecto me ha permitido afianzar conocimientos adquiridos en el máster y ampliar otros que no se tocan en el mismo.

Ha sido un reto ya que es para un cliente real que va a utilizarlo para actualizar su negocio.

Me ha permitido adaptar conocimientos y formas de trabajo a requisitos de la vida real y darme cuenta de que un proyecto de esta escala lleva un proceso largo, estudiado y complejo, en el que la comunicación entre el cliente y el desarrollador es imprescindible. Así como unas buenas prácticas de programación, test adecuados y un periodo de prueba antes del despliegue final son necesarios.

9.2 Ampliaciones

Al tratarse de un prototipo las ampliaciones están sujetas al consenso alcanzado con el cliente en la fase de pruebas. Así mismo en esta fase se harán las pruebas de accesibilidad que no aparecen en este documento.

Una de las ampliaciones que sí es seguro, es la necesidad de su desarrollo es una página web de gestión de la cooperativa.

Capítulo 10. Presupuesto

10.1 Resumen del presupuesto

Se tiene en cuenta el tiempo estimado de duración del proyecto, así como los materiales empleados.

El período de amortización para los recursos físicos se ha establecido en 5 años, mientras que las licencias de software, en caso de ser necesarias he considerado que han de fijarse a 3 años.

La duración del proyecto se estima que va a ser de 5 meses, por lo que el factor de amortización se va a establecer teniendo en cuenta este dato. Para el hardware se establecerá en base a la función ***(precio del recurso/5) *0,42*** y para los recursos software será ***(precio del recurso/3) *0.42***.

10.2 Recursos materiales

Utilizando la fórmula anterior calculamos el precio amortizado de los recursos hardware. El material de oficina es el que se prevé que se va a necesitar en un primer momento.

RECURSO	PRECIO (€)	AMORTIZACIÓN (€)
PORTÁTIL	600,00	50,4
PANTALLA	104,00	8,74
MATERIAL DE OFICINA	30,00	30,00
	Total	***8,14***

10.3 Recursos software

De momento se prevé que quizá se necesiten los siguientes paquetes y licencias, hasta estudiar el proyecto y analizarlo no se sabe si será necesario el pago de otras licencias, en principio se va a intentar utilizar software libre.

RECURSO	PRECIO (€)	AMORTIZACIÓN (€)
MICROSOFT OFFICE	539,00	75,46
MICROSOFT PROYECT	1369,00	191,66
	Total	***267,12***

10.4 Consumo

El consumo eléctrico suponiendo que el coste medio es de 0,14€/Kwh y se supone que se va a usar según la planificación durante:

RECURSO /CONSUMO	CONSUMO (KWH)	HORAS	COSTE (€)
PANTALLA	0,6	426	35,78
ORDENADOR PORTATIL	0,08	426	4,77
		Total	***40,55***

10.5 Coste total recursos físicos

Por último, se debe añadir un plus por mantenimiento y depreciación, equivalente al 20% del coste de la vida útil en referencia al proyecto. El coste final, por lo tanto, es el de su precio amortizado, su consumo, más el extra de mantenimiento y depreciación.

Todas las unidades de la tabla están en Euros.

RECURSO TECNOLÓGICO	COSTE AMORTIZADO	CONSUMO	COSTE MANT/DEPR	COSTE FINAL
PANTALLA	50,4	35,78	10,08	96,26
ORDENADOR PORTATIL	8,74	4,77	1,75	15,26
			Total	***111,52***

10.6 Gastos en recursos humanos

El proyecto como es realizado por una única persona que es a la vez, analista, programador, diseñador y el resto de los roles necesarios, se establecen unos precios por hora específicos por tareas.

- Formación: 16 €/h.
- Análisis: 15 €/h.
- Diseño: 15€/h.
- Implementación: 12€/h.
- Documentación: 10 €/h.

FASE DEL PROYECTO	PRECIO/HORA (€/H)	HORAS (H)	COSTE FINAL (€)
FORMACIÓN	16,00	40	640,00

ANÁLISIS	15,00	20	300,00
DISEÑO	15,00	20	300,00
IMPLEMENTACIÓN	12,00	290	3480,00
DOCUMENTACIÓN	10,00	68	680,00
		Total	***5400,00 €***

10.7 Presupuesto interno del proyecto

Con todo lo calculado en los apartados anteriores se construye el presupuesto interno.

DETALLE	COSTE (€)
RECURSOS MATERIALES	***408,64***
RECURSOS HARDWARE	141,52
RECURSOS SOFTWARE	267,12
RECURSOS HUMANOS	***5400,00***
FORMACIÓN	640,00
ANÁLISIS	300,00
DISEÑO	300,00
IMPLEMENTACIÓN	3480,00
DOCUMENTACIÓN	680,00
SUBTOTAL	***5808,64***
COSTES INDIRECTOS (5%)	290,43
BENEFICIO (18%)	1045,55
TOTAL (PRESUPUESTO INTERNO)	***7144,62***

10.8 Presupuesto para el cliente del proyecto

Partiendo el presupuesto interno, se hace el del cliente, prorrateando todo lo que no sean las fases del proyecto o entregables de seguimiento por el mismo.

DETALLE	COSTE (€)
FASES DEL PROYECTO	***7144,62***
FORMACIÓN	799,26
ANÁLISIS	379,70
DISEÑO	379,70
IMPLEMENTACIÓN	4634,32
DOCUMENTACIÓN	951,64

IVA (21%)	1500,37
PRESUPUESTO TOTAL	***8644,99***

Capítulo 11. Referencias Bibliográficas

11.1 Libros y Artículos

[Schildt14] Schildt, Herbert" Java 8 ". Anaya. 2014. ISBN: 978-84-415-3625-8

11.2 Referencias en Internet

[Spring] “Documentación de Spring”, https://spring.io/projects , 2018

[Ionic] “Documentación Ionic”, https://ionicframework.com/docs/components/, 2018

[Angular] “Angular, The Hero Editor”, https://angular.io/tutorial/toh-pt1, 2018

[RxJs] “Observable”,

http://reactivex.io/rxjs/class/es6/Observable.js~Observable.html#instance-method-subscribe , 2018

[RxJS] “Api Document”, https://rxjs-dev.firebaseapp.com/, 2008

[Spring] “Async in Spring”, http://www.baeldung.com/spring-async , 2018

[Firebase] “Firebase Cloud Messaging”, https://firebase.google.com/products/cloud-messaging/, 2018

[Mockito] “Class ArgumentMatchers”, https://static.javadoc.io/org.mockito/mockito-core/2.18.3/org/mockito/ArgumentMatchers.html, 2018

[Date-fns] “Date-fns”, https://date-fns.org/, 2018

[Java]” Guidelines, Patterns, and code for end-to-end Java applications”, http://www.oracle.com/technetwork/java/namingconventions-139351.html, 2018

[TypeScript]” TypeScript StyleGuide and Coding Conventions”, https://github.com/basarat/typescript-book/blob/master/docs/styleguide/styleguide.md, 2018

Capítulo 12. Apéndices

12.1 Código Fuente

Se adjunta con la documentación tres carpetas con el código:

- Servidor: “taxi-pilona-ws”
- Aplicación cliente: “taxi-pilona-cliente”
- Aplicación taxista: “taxi-pilona-taxista”