UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

INGENIERÍA DEL SOFTWARE

UNICAR: UNA PLATAFORMA PARA COMPARTIR VEHÍCULOS ENTRE ALUMNOS Y PERSONAL DE LA UMA

UNICAR: A VEHICLE SHARE PLATFORM BETWEEN UMA STUDENTS AND STAFF

Realizado por
Marcos Sepúlveda Romero

Tutorizado por
Gabriel Jesús Luque Polo

Departamento
Lenguajes y Ciencias de la Computación
UNIVERSIDAD DE MÁLAGA

MÁLAGA, SEPTIEMBRE DE 2020

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADO EN INGENIERÍA DEL SOFTWARE

UNICAR: UNA PLATAFORMA PARA COMPARTIR VEHÍCULOS ENTRE ALUMNOS Y PERSONAL DE LA UMA

UNICAR: A VEHICLE SHARE PLATFORM BETWEEN UMA STUDENTS AND STAFF

Realizado por
Marcos Sepúlveda Romero

Tutorizado por
Gabriel Jesús Luque Polo

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, SEPTIEMBRE DE 2020

Fecha defensa:

Resumen

Este trabajo fin de grado (TFG) titulado “UNICAR: Una plataforma para compartir vehículos entre alumnos y personal de la UMA” es una plataforma digital para compartir los trayectos que realizan los vehículos entre alumnos y personal de la UMA. Ofrece a los usuarios de la Universidad de Málaga (UMA) una aplicación en la que los conductores con destino a alguno de los emplazamientos universitarios de la UMA la oportunidad de ofrecer las plazas libres de su vehículo a otros usuarios en base a sus horarios y su ubicación, y a otros usuarios que no tengan vehículo y que tengan que ir a la UMA encontrar conductores que ofrezcan plazas libres. Este TFG se ha desarrollado pensando en la sostenibilidad y la reducción de emisiones de carbono poniendo en contacto a los usuarios de la UMA entre ellos, ofreciendo una aplicación multiplataforma. De esta forma se reduce la exposición de los usuarios al coronavirus al no utilizar el transporte público donde hay mayor probabilidad de contagio.

Palabras clave: Carpooling, Ciudades inteligentes, App, Ionic, Angular.

Abstract

This Final Degree Project (FDP) titled “UNICAR: A vehicle share platform between UMA students and staff” is a digital platform for sharing vehicles between users of our university (both, students and staff). Our system provides a web application to car drivers going to one of the UMA university campus, in which they have the opportunity of offering free places in their vehicle to other users based on their schedules and their location: Our platform can be also used oppositely: users who do not own a personal vehicle and they need to go to the UMA, can use our application to find car drivers with free places. This FDP has been developed keeping in mind sustainability and the reduction of carbon emissions by keeping UMA users in contact with each other, offering a multiplatform application. Also, this kind of application is very useful currently to reduce the exposure of users to the coronavirus since it allows us to avoid public transport where there is a greater probability of contagion.

Keywords: Carpooling, Smart Cities, App, Ionic, Angular.

Índice

1. Introducción

En este primer capítulo mostraremos las razones que llevaron a la realización de este trabajo fin de grado y sus principales objetivos y, para finalizar, mostraremos la organización de esta memoria.

1.1 Motivación

En Málaga, las opciones de transporte público para poder ir a la Universidad de Málaga (de ahora en adelante UMA) son varias: autobús, metro, tren de cercanías o el alquiler de bicis entre otras. Sin embargo, el acceso por parte de los usuarios a tanto los autobuses urbanos e interurbanos como el Metro y el tren de cercanías está limitado tanto por la localidad como por horarios y por eso no es una opción para todo el mundo.

La alternativa principal al transporte público es utilizar un vehículo propio, pero esto no está al alcance de todo el mundo, y los que lo pueden utilizar solos afrontan un gasto económico mayor que si cogieran el transporte público.

Ponerse en contacto con personas que dispongan de un vehículo y acordar una cuantía por el desplazamiento hasta el lugar de destino, en este caso la UMA, es una tarea complicada, pues no todas las personas conocen a gente interesada cercana o con horarios parecidos que puedan coincidir, o simplemente no se fían de que alguien ajeno a la UMA los lleve.

Cabe destacar que, con todo el revuelo ocasionado por la crisis del coronavirus, es preferible limitar los vectores de transmisión del virus, por lo que utilizar el transporte público no es la elección más idónea.

El objetivo de este TFG es el de crear una aplicación que permita realizar carpooling (Gallo & Buonocore, 2017) entre los alumnos y el personal de la UMA de una forma automática, fiable y segura, en base a la localización y los horarios de estos, para así mejorar la comunicación entre los alumnos, reducir la aglomeración de vehículos en las carreteras y las emisiones de CO_2 en el medio ambiente (Bruck et al., 2017) además de reducir la probabilidad de contagio del coronavirus.

1.2 Objetivos

El objetivo principal de este TFG consistirá en desarrollar una aplicación que ofrezca a los usuarios de la UMA una plataforma digital en la que los conductores con destino la UMA puedan ofrecer las plazas libres de su vehículo a otros usuarios en base a sus horarios y su ubicación.

Hay que tener en cuenta que este proyecto facilita enormemente la comunicación entre los usuarios, pues:

- No todos los usuarios de la UMA se conocen entre sí, por lo que ofrecer un punto de encuentro para estos facilita la comunicación y fortalece sus vínculos.
- Los horarios entre los usuarios de la UMA son muy diversos, por lo que casar estos puede llegar a ser complicado.
- Es posible que no haya pasajeros cerca del punto de partida del conductor, pero sí de camino por la ruta que toma este.

En resumen, la aplicación será un punto de encuentro digital enfocado en sugerir a usuarios, tanto conductores como pasajeros, otros posibles usuarios en base a sus horarios y ubicaciones, para realizar carpooling de una forma cómoda, todo eso en una única plataforma en la que también puedan comunicarse entre ellos.

Para ello se utilizará una arquitectura cliente-servidor donde el servidor almacenará los datos y la lógica de negocio y, el cliente, mediante una interfaz

web responsive, permitirá el acceso a los servicios del sistema tanto en sistemas móviles como en ordenadores personales.

En todo momento se seguirá una metodología ágil para tener listo en cada iteración un nuevo conjunto de funcionalidades y realizar un seguimiento adecuado. En la Sección 2.5 se explicará con más detalle los procedimientos seguidos.

1.3 Estructura de la memoria

Ahora expondremos brevemente el contenido de cada apartado de la memoria:

1. **Introducción**

En este capítulo presentaremos las motivaciones y objetivos de nuestro proyecto junto a un breve resumen de este documento.

2. **Tecnologías, herramientas y metodología**

Este segundo capítulo describiremos las tecnologías, recursos y metodologías utilizadas para la realización de este proyecto.

3. **Análisis y modelado**

Aquí comentaremos los requisitos y las decisiones de diseño de este proyecto software, junto a los diagramas pertinentes.

4. **Desarrollo e implementación**

En esta parte del documento se mostrarán las distintas iteraciones que componen el desarrollo de la aplicación.

5. **Conclusiones y líneas futuras**

Por último, comentaremos las conclusiones a las que hemos llegado a lo largo del proyecto y posibles mejoras que se pueden aplicar en futuros desarrollos.

Finalmente, la memoria acaba con las referencias bibliográficas consultadas durante la realización de este trabajo fin de grado y un apéndice que sirve como manual de uso de la aplicación desarrollada.

2. Tecnologías, herramientas y metodología

Tras explicar el objetivo de este trabajo, en este capítulo vamos a introducir de forma breve los diferentes elementos (tanto tecnologías, herramientas como metodologías de desarrollo) que se han necesitado para la realización del sistema.

2.1 Tecnologías utilizadas

Empezamos explicando las tecnologías utilizadas para conseguir el objetivo de este trabajo fin de grado y por qué se han utilizado. Las podemos dividir en tecnologías de backend y frontend.

2.1.1 Frontend

Para el frontend se han elegido las siguientes tecnologías:

2.1.1.1 Ionic Framework

Figura 2.1.1: Logo de Ionic Framework.

Ionic (Drifty Company, 2013) es un framework de interfaz de usuario móvil open source para crear experiencias de aplicaciones web y nativas multiplataforma de

alta calidad. Este framework provee una extensa librería de componentes de interfaz de usuario, gestos y herramientas para crear rápidamente aplicaciones.

Ionic permite a los desarrolladores web desarrollar una sola aplicación y desplegarla en diferentes plataformas ya sea como una app o como una Progressive Web App (PWA) (Goswami & Chatterjee, 2019) utilizando el mismo código para ambas.

También incluye una línea de comandos (CLI) propia que facilita el crear, probar y desplegar la aplicación, que incluye funcionalidades como Live Reload, para recargar automáticamente la aplicación en desarrollo cuando se realicen cambios en el código o despliegues en múltiples plataformas.

Ionic está diseñado para funcionar y ejecutarse rápidamente en todos los dispositivos móviles más recientes. Las aplicaciones creadas utilizando este framework están listas para ser usadas con un tamaño reducido y con las mejores prácticas integradas, como transiciones aceleradas por hardware, gestos táctiles optimizados, renderizado previo y compilación AOT (Ahead-of-Time compiler) (Ahead-of-time compilation, 2020).

Además, Ionic soporta diferentes frameworks frontend como Angular2 (Google, 2016), React (Facebook, Inc., 2013) o Vue (Vue.js, 2014) o simplemente utilizando JavaScript (Netscape Communications & Fundación Mozilla, 1995).

Los componentes de Ionic están escritos en HTML (W3C, 1992), CSS (W3C, 1996) y JavaScript, lo que facilita la creación de interfaces de usuario de alta calidad que funcionan muy bien en las diferentes plataformas.

Ionic facilita el acceso a las funciones nativas del dispositivo mediante JavaScript. Permite elegir entre una librería de más de 120 complementos de

dispositivos nativos para acceder a la cámara, la ubicación geográfica, Bluetooth y más, o también utilizar el SDK completo cuando se necesite.

Por último, posee una librería de plugins creados por la comunidad o por empresas que facilitan la integración de ciertas funcionalidades en la aplicación.

2.1.1.2 Angular2

Figura 2.1.2: Logo de Angular2.

Como se comentó en la sección previa, Ionic permite frameworks para desarrollar el frontend y en concreto se ha seleccionado Angular2.

Angular2 es un framework para el desarrollo de aplicaciones web de código abierto basado en TypeScript (Microsoft, 2018) y utiliza el patrón de arquitectura software Modelo Vista Controlador (MVC). Su desarrollo está liderado por el Angular Team en Google y está soportado por la comunidad y otras empresas.

Angular está basado en el uso de componentes, su bloque de construcción básico, dichos componentes están formados a su vez por tres elementos, la vista, formada por HTML y CSS y la lógica, que se desarrolla en un archivo TypeScript. Angular también provee otras herramientas que facilitan el desarrollo y andamiaje de la aplicación tales como los módulos y el routing.

Se eligió este framework para el desarrollo pues, aunque posee una curva de aprendizaje pronunciada, es bastante versátil y es también unos de los lenguajes de programación que más se demandan en los perfiles de trabajo (Barot, 2019).

2.1.2 Backend

Para el backend se ha optado por Firebase (Google, 2014), un servicio PaaS (Platform as a Service) de Google ya establecido para así facilitar el desarrollo de la aplicación.

Figura 2.1.3: Logo de Firebase by Google.

Firebase posee una suite de herramientas y funciones muy poderosa. Algunas de las que incluye son:

- Firebase Authentication, un sistema de autenticación de usuario, con el cual se provee una serie de diferentes métodos de acceso a nuestra aplicación, entre ellos utilizar email y contraseña, cuenta de Google, Facebook, etc. además de un sistema de cambio de contraseña, verificación de email entre otros servicios.
- Cloud Firestore, una base de datos NoSQL flexible, escalable y en la nube a fin de almacenar y sincronizar datos para la programación en el lado del cliente y del servidor.
- Almacenamiento para guardar archivos subidos por usuarios tales como imágenes de perfil.
- Hosting, desde el cual puedes configurar el hosting y dominio del servidor.
- Cloud Functions, que permite que el programador desarrolle funciones y métodos que sólo quiere que se ejecute en el servidor.
- Una serie de aplicaciones analíticas y estadísticas que permiten evaluar el rendimiento de la aplicación.

En las próximas subsecciones se describirán las principales características de cada una de esas herramientas.

2.1.2.1 Firebase Authentication

Firebase Authentication proporciona servicios de backend, SDK fáciles de usar y librerías de IU listas para autenticar a los usuarios en la aplicación. Admite la autenticación mediante contraseñas, números de teléfono, proveedores de identidad federados populares como Google, Facebook y Twitter, entre otros.

Firebase Authentication se integra estrechamente con otros servicios de Firebase e implementa los estándares como OAuth 2.0 (*OAuth 2.0*, 2006) y OpenID Connect, por lo que se puede integrar fácilmente con el backend.

Utilizando así esta funcionalidad que nos proporciona Firebase podemos conocer la identidad de un usuario de forma sencilla, además de permitir que una aplicación guarde de forma segura los datos del usuario en la nube y brinde la misma experiencia personalizada en todos los dispositivos del usuario.

2.1.2.1 Cloud Firestore

Cloud Firestore es una base de datos NoSQL (Strozzi.it, 2007) flexible y escalable para el desarrollo móvil, web y de servidores para Firebase y Google Cloud Platform. Mantiene los datos sincronizados entre el cliente gracias a realtime listeners y ofrece soporte offline para móviles y web. Cloud Firestore también ofrece una integración perfecta con otros productos de Firebase y Google Cloud Platform, incluido Cloud Functions.

El modelo de datos de Cloud Firestore admite estructuras de datos jerárquicas y flexibles. Los datos se almacenan en documentos y estos a su vez en colecciones. Los documentos pueden contener objetos anidados complejos además de subcolecciones.

En Cloud Firestore, se usan consultas para recuperar documentos individuales o para recuperar todos los documentos de una colección que coincidan con los parámetros de consulta. Las consultas pueden incluir varios filtros encadenados

además de filtrado y el orden de los datos obtenidos. Los datos están indexados de forma predeterminada, por lo que el rendimiento de las consultas es proporcional al tamaño del conjunto de resultados, no a su conjunto total de datos.

Cloud Firestore también ofrece algunas de las mejores características de la infraestructura de Google Cloud Platform, como replicación automática de datos para servidores en diferentes regiones u operaciones atómicas por lotes y soporte para transacciones reales.

Está diseñado para aguantar las cargas de trabajo más pesadas sobre las bases de datos de las aplicaciones más grandes del mundo.

2.1.2.1.1 Modelo de datos

El modelo de datos NoSQL de Cloud Firestore (Google, 2020) almacena datos en documentos que contienen campos que se asignan a valores. Estos documentos se almacenan en colecciones, estos son contenedores de documentos que se pueden utilizar para organizar los datos de diferentes maneras al igual que ser utilizados en las consultas.

Cada documento contiene un conjunto de pares clave-valor. Cloud Firestore está optimizado para almacenar grandes colecciones de documentos pequeños.

Los documentos pueden contener subcolecciones y objetos anidados, los cuales pueden incluir campos primitivos como cadenas u objetos complejos como listas, construyendo así estructuras de datos jerárquicas que se escalan a medida que crece la base de datos.

Cada documento en Cloud Firestore se identifica de forma única por su ubicación dentro de la base de datos.

Los tipos de datos primitivos soportados por Cloud Firestore son los siguientes:

- Array
- Boolean
- Bytes
- Date, como un valor Timestamp en microsegundos.
- Float
- Integer
- Map, representa un objeto formado por pares clave-valor. Cuando está indexado, se pueden realizar consultas sobre subcampos.
- Null
- String

Hay más información sobre los tipos de datos que acepta Cloud Firestore en la página del proyecto (Google, 2020).

2.1.2.1.2 Seguridad

Cloud Firestore ofrece autenticación y administración de acceso a través de dos métodos diferentes, según las librerías que se utilicen:

- Para clientes web y móvil, Firebase Authentication y las reglas de seguridad de Cloud Firestore para manejar la autenticación, autorización y validación de datos. Las reglas de seguridad proporcionan control de acceso y validación de datos en un formato simple pero expresivo. Con ellas se pueden llegar a crear sistemas de acceso basados en roles y usuarios que mantengan seguros los datos de los usuarios.

- En el servidor se utilizan la Identity and Access Management (IAM) para administrar el acceso a la base de datos.

2.1.2.1.3 Usos, límites y precios

Cloud Firestore ofrece diferentes planes al desarrollador, para este proyecto se ha elegido el plan Spark, el cual es gratuito, pero conlleva una serie de límites:

Activo	Cuota límite
Datos almacenados	1 GiB
Lecturas de documentos	50.000 por día
Escrituras de documentos	20.000 por día
Eliminaciones de documentos	20.000 por día
Salida de red	10 GiB por mes

Por defecto Cloud Firestore tiene aplicados otros límites de uso no asociados al plan que estés suscrito (Google, 2020).

Además de límites por plan elegido hay que tener en cuenta otros (Google, 2020):

- Cada documento tiene un límite de tamaño máximo de 1 MB.
- Un documento no puede tener más de 20.000 campos.
- Solo se permite una escritura en un documento por segundo, por lo que no se recomienda que varios usuarios escriban a la vez en el mismo documento, pues se puede llegar a perder información.
- No se puede obtener un documento parcialmente.
- Cuando se requiere un documento que apunta a una subcolección solo se devuelve el documento.
- Firebase te cobra por el número de lecturas y escrituras que se realizan sobre un documento, por lo que hay que tener en cuenta los documentos que se utilizarán más frecuentemente.
- Las consultas encuentran los documentos que están dentro de una sola colección. Se pueden usar consultas en grupos de colecciones para buscar documentos dentro de subcolecciones, solo que está limitado a grupos de 200. Otra opción es utilizar un mapa o meterlo en una colección en el documento raíz.

- Si varios usuarios modifican a la vez un array en un documento se pueden crear problemas, es por eso por lo que no se pueden insertar, borrar o modificar elementos individuales de un array, ni siquiera se pueden utilizar consultas para comprobar si un elemento contiene cierto valor. Es por eso por lo que se puede usar la función arrayContains("keyword") para comprobar si un valor está contenido en el array.
- No se pueden utilizar dos funciones arrayContains() concatenadas en una misma consulta.
- Se pueden utilizar consultas por lotes o transacciones para realizar de forma atómica un proceso que requiera varias consultas y/o modificaciones en la base de datos.

Estos límites se han tenido que tener en cuenta a la hora del desarrollo de la aplicación para no superarlos, por lo que se han tomado una serie de decisiones en la estructura de la base de datos en base a estos. En el Capítulo 4 se ahondará en las decisiones tomadas y sus repercusiones.

2.1.2.2 Cloud Functions

Cloud Functions (Google, 2020) es un framework que provee Firebase que permite ejecutar automáticamente código en el backend en respuesta a eventos ocasionados por peticiones HTTPS y otras características de Firebase.

Los lenguajes aceptados por Cloud Functions son JavaScript y TypeScript, ya que se ejecutan en un entorno con NodeJS.

El código se sube a los servidores con un simple comando a través de la consola de comandos. Tras eso, Firebase automáticamente se encarga de escalar los recursos computacionales necesarios dependiendo de los patrones de uso generados por los usuarios al usar la aplicación. Cada función subida a la nube se ejecuta de forma aislada, en su propio entorno con su propia configuración.

En el Capítulo 4 se hablará más detenidamente sobre qué funciones se ejecutan en el backend.

2.1.3 Librerías

2.1.3.1 Google Maps Platform

Figura 2.1.4: Logo de Google Maps Platform.

Para facilitar el desarrollo se ha optado por utilizar la API Javascript de Google Maps, que proporciona métodos y funciones que nos serán de gran ayuda.

La API de JavaScript de Maps permite personalizar mapas con contenido propio e imágenes para mostrarlos en páginas web y dispositivos móviles. La API de JavaScript de Maps presenta cuatro tipos de mapas básicos (mapa de ruta, satélite, híbrido y terreno) que se pueden modificar utilizando capas y estilos, controles y eventos, y varios servicios y bibliotecas.

A lo largo de este documento se irán comentando las funciones y métodos utilizados al igual que ciertas explicaciones sobre su funcionamiento.

2.1.3.1.1 APIs utilizadas

Los servicios de Google Maps Platform (Google, 2005) están divididos en diferentes APIs cada una con un propósito concreto. En esta aplicación se han utilizado las siguientes APIs:

- Maps JavaScript API. Permite personalizar mapas con contenido propio e imágenes para mostrarlas en páginas web y dispositivos móviles.
- Places API. Es un servicio que devuelve información sobre lugares mediante solicitudes HTTP. Los lugares se definen dentro de esta API

como establecimientos, ubicaciones geográficas o puntos de interés destacados.

- Geocoding API. Esta API es la que se encarga del proceso de conversión de direcciones en coordenadas geográficas y viceversa. También se puede utilizar para encontrar la dirección perteneciente a un ID de un lugar determinado.
- Directions API. Es un servicio que calcula la ruta entre ubicaciones mediante una solicitud HTTP.

2.1.3.1.2 Límites y facturación

La API de Google Maps no es de uso gratuito (Google, 2020), sino que se necesita registrar una cuenta o tarjeta de crédito para la facturación, que se cobra cada mes.

Los precios varían dependiendo del uso y de la API que utilices, la mayoría siguen un modelo de Pay-as-you-go (pagar por el uso de cada API), aquí vamos a mostrar la lista de los precios en dólares de las APIs utilizadas en este proyecto:

API	**Stock Keeping Units (SKUs)**	**Tipo de cargo**	**Precio por número de peticiones**	
			0–100.000	100.001–500.000
Maps JavaScript API	Dynamic Maps	Por carga de mapa	0,007 USD	0,0056 USD
Places API	Basic, Autocomplete	Por cada petición	0,00283 USD	0,00227 USD
Geocoding API	Geocoding	Por cada petición	0,005 USD	0,004 USD
Directions API	Directions, Directions Advanced	Por cada consulta	0,01 USD	0,008 USD

Una vez que se ha creado una cuenta de facturación y un proyecto, Google proporciona un crédito inicial de prueba gratuito de 300 USD para Google Cloud Platform y un crédito mensual de 200 USD para Google Maps Platform.

En los capítulos 4 y 5 se hablará más a fondo y en contexto sobre las peticiones realizadas a la API JavaScript de Google Maps.

2.1.3.2 Librerías de NodeJS

Para facilitar el desarrollo se han utilizado las siguientes librerías externas:

- Angular Google Maps (AGM) (SebastianM, 2020). Son componentes de Angular para Google Maps.
- Agm-Direction (explooosion, 2020). Ofrece componentes de dibujo de direcciones para AGM.
- ngx-color (scttcper, 2020). Ofrece una serie de componentes para seleccionar colores de una paleta.
- spherical-geometry-js (NotWoods, 2018). Es un conjunto de funciones simplificadas del SDK de JavaScript de Google Maps.

Todas estas librerías utilizan la licencia MIT.

2.2 Herramientas de desarrollo

Para desarrollar la aplicación se ha optado por el uso de Visual Studio Code (Microsoft, 2015), un editor de código optimizado con soporte para operaciones de desarrollo como depuración, ejecución de tareas y control de versiones.

Su objetivo es proporcionar solo las herramientas que un desarrollador necesita para un ciclo rápido de código, compilación y depuración. Es gratuito y muy versátil gracias al soporte de las extensiones que aporta la comunidad.

2.3 Herramientas para la gestión

En este apartado vamos a exponer todas las herramientas utilizadas para gestionar el proyecto a lo largo de las iteraciones.

2.3.1 Git y GitHub

Git (Git, 2005) se ha utilizado para llevar un seguimiento de la actividad del desarrollo y la aplicación junto con GitHub (GitHub, 2008) donde se ha creado un repositorio privado para almacenar el proyecto y realizar los cambios del código de forma progresiva.

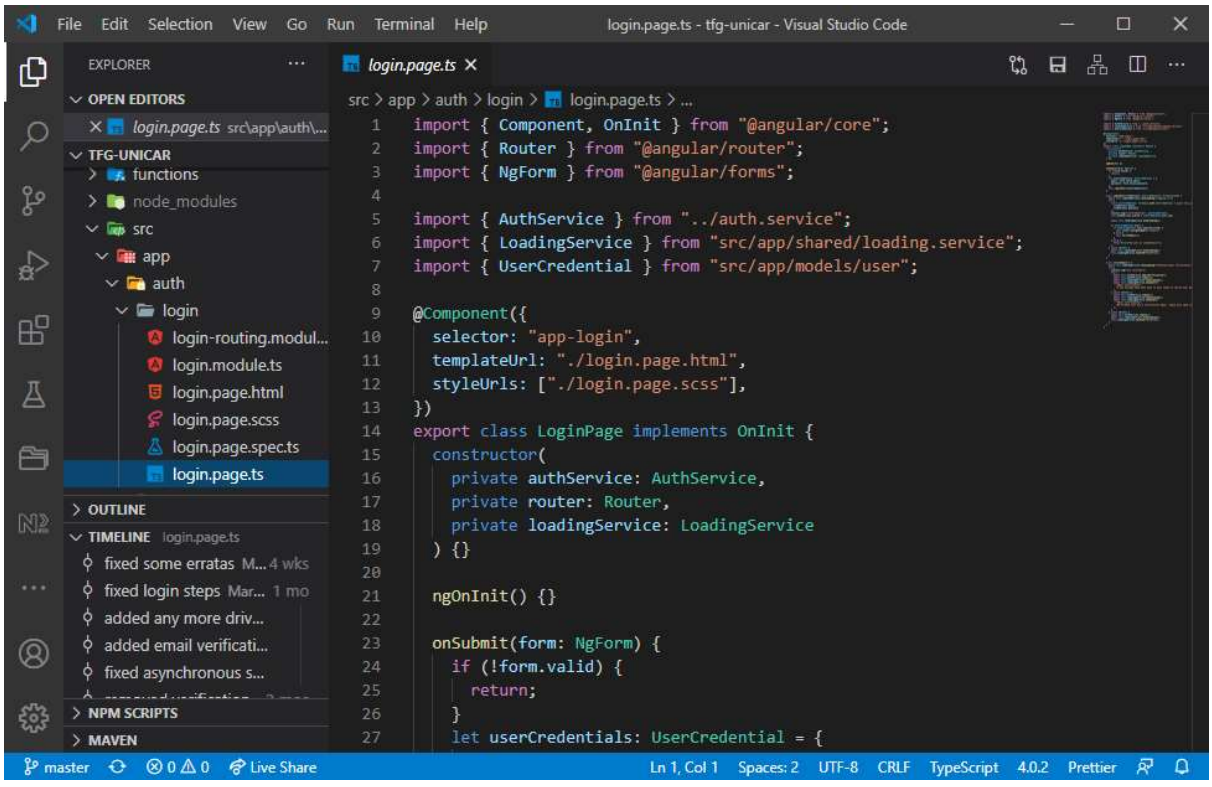


Figura 2.3.1: Edición de la página de inicio de sesión en Visual Studio Code.

Además, estas herramientas nos permiten restaurar una versión anterior si fuera necesario o desarrollar funcionalidades en paralelo sin que haya conflictos, entre otras funcionalidades.

2.3.2 Trello

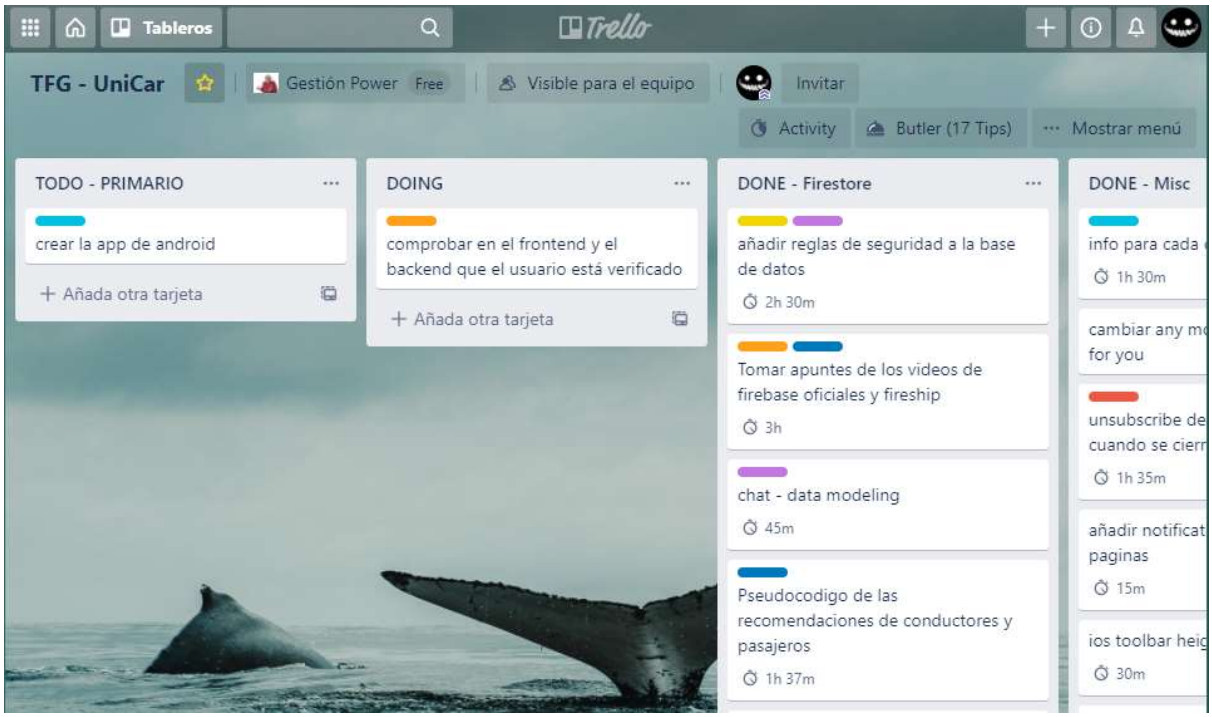


Figura 2.3.1: Captura de pantalla del tablero de Trello del proyecto fin de grado.

A la hora de seguir una metodología ágil de desarrollo, es necesario una aplicación donde pueda uno estructurarse bien las tareas y procedimientos a seguir, por eso se ha elegido Trello (Atlassian, 2011).

Trello es una aplicación web diseñada para gestionar las tareas, está estructurada de forma que se tiene un tablero, dentro de este se tienen listas y dentro de estas están las tareas a realizar.

2.4 Herramientas para la documentación

Para documentar los cambios y pasos que se han seguido para el desarrollo de este trabajo fin de grado han sido los siguientes:

2.4.2 Google Docs

Gracias a Google Docs (Google, 2006) se ha provisto el desarrollo tanto de la memoria, como de otros apuntes y notas que se han ido tomando a medida que se avanzaba en la investigación y el desarrollo.

Google Docs es un procesador de texto incluido como parte de un paquete ofimático de software gratuito web que ofrece Google dentro de su servicio Google Drive. Este servicio también incluye Google Sheets y Google Slides, una hoja de cálculo y un programa de presentación respectivamente.

2.4.2 diagrams.net

Diagrams.net (diagrams.net, 2005) es un conjunto de herramientas de código abierto para crear diagramas y es también uno de los software de desarrollo de diagramas más utilizado en el mundo a nivel de navegadores.

Esta página ha sido utilizada para crear, prototipar y desarrollar los diferentes diagramas de la aplicación.

2.5 Metodología utilizada

Este trabajo fin de grado se ha dividido en las siguientes fases de desarrollo:

1. Aprender las herramientas y tecnologías a utilizar (Ionic con Angular2 y Firebase) y entender cómo funcionan por dentro.
2. Análisis e investigación de la factibilidad de las ideas en las que se van a basar los algoritmos de la aplicación. Aquí también se han desarrollado algunos prototipos de algunas funcionalidades como el chat y la identificación única de coordenadas.
3. Diseñar los mockups de la aplicación del cliente, con lo que se obtiene una idea de cómo será el producto final.
4. Diseñar la estructura de la base de datos teniendo en cuenta las restricciones de precios y límites de Cloud Firestore.

5. Preparar las herramientas, programas, librerías y configuraciones necesarias para el proyecto.
6. Dividir la aplicación en diferentes iteraciones, de tal forma que el desarrollo sea incremental. En cada iteración se han ido tomando apuntes para facilitar la escritura de la memoria. Tras cada iteración se ha realizado una pequeña iteración con el objetivo de corregir la mayor cantidad de bugs posibles ocasionados por la integración del contenido nuevo.

   A la hora de establecer el orden de desarrollo de cada iteración hemos seguido el enfoque “Lo crítico primero”, por lo que el orden final ha sido:

   a. Iteración Auth, en la que se incluye todo el conjunto de páginas, servicios y configuración en el servidor necesario relacionado con la creación de usuarios, inicio de sesión, autenticación, registro y gestión de usuarios.
   b. Iteración Passenger, en esta se incluyen las páginas de selección de ruta, coche del usuario y listado de recomendaciones de pasajeros a conductores, al igual que los servicios y componentes necesarios para su funcionamiento, y todo lo relacionado con mapas.
   c. Iteración Driver, aquí las páginas de selección de punto de recogida y recomendaciones de conductores a pasajeros junto a los componentes y servicios necesarios, al igual que todo lo relacionado con mapas.
   d. Iteración de recomendaciones, aquí se incluyen los algoritmos en el servidor necesarios para el emparejamiento entre conductores y pasajeros y viceversa.
   e. Iteración Chat, formada por la creación de la página de lista de chats, el componente chat, el servicio de mensajería, las funciones y algoritmos necesarios para su funcionamiento en el servidor, y la integración en otras páginas como en las listas de recomendaciones.
   f. Iteración Timetable, en el que se incluyen la página de modificación de turnos, selectores de hora y días, su servicio correspondiente y su integración en las páginas de recomendaciones.

g. Iteración sobre la interfaz, donde se han diseñado el logo, colores del tema y temas relacionados con el pulido de la interfaz de usuario.
h. Iteración de corrección de bugs final, donde, como su propio nombre indica se han realizado todos los cambios necesarios para solucionar todos los problemas y errores posibles.

7. Escritura de la memoria del trabajo fin de grado.
8. Preparación de la presentación del trabajo fin de grado.

Para llevar a cabo todas estas fases e iteraciones se ha utilizado la herramienta Trello. Como hemos explicado previamente en la Sección 2.3.2, Trello permite organizar en listas tarjetas en las que se pueden especificar tareas, en nuestro caso cada tarjeta se ha utilizado como un objetivo pequeño a realizar, en cuyo caso es una pequeña funcionalidad perteneciente a la iteración correspondiente.

El ciclo de vida de una tarjeta pasa por tres listas diferentes:

- TO DO, aquí se almacenan todos los objetivos que nos quedan por realizar.
- DOING, aquí solamente se ponen los objetivos que se estén desarrollando en ese momento.
- DONE, que puede haber más de una si tenemos diferentes listas para cada iteración o módulo que estemos desarrollando. Aquí se almacenan todos los objetivos que hemos completado en una iteración.

Cada iteración se ha llevado a cabo aproximadamente en una semana, donde se ha tenido en cuenta temas como: resolución de problemas, inexperiencia sobre la tecnología utilizada, consultas con el tutor, etc...

En resumen, se podría decir que la metodología utilizada ha sido una simplificación de la metodología de desarrollo Kanban (APD, 2019).

3. Análisis y modelado

3.1 Requisitos funcionales

En este apartado definiremos los requisitos funcionales de la aplicación.

ID	Descripción
RF-USER-01	El sistema debe permitir al usuario iniciar sesión en la aplicación.
RF-USER-02	El sistema debe permitir al usuario cerrar sesión en la aplicación.
RF-USER-03	El sistema debe permitir crear una cuenta a un usuario nuevo.
RF-USER-04	El sistema debe permitir recuperar la contraseña a un usuario.
RF-USER-05	El usuario debe poder modificar la información de su perfil.
RF-PASS-01	El usuario debe poder seleccionar o modificar una ruta en el mapa entre un punto de origen y una de las universidades destino.
RF-PASS-02	El usuario debe poder introducir información sobre su coche.
RF-PASS-03	El sistema debe recomendar en base a la información del conductor un conjunto de usuarios pasajeros.
RF-DRIV-01	El usuario debe poder seleccionar o modificar un punto de recogida en el mapa.
RF-DRIV-02	El sistema debe recomendar en base a la información del pasajero un conjunto de usuarios conductor.
RF-TIME-01	El sistema debe permitir al usuario registrar la información de sus horarios de entrada y salida.
RF-TIME-02	Un usuario debe poder consultar los horarios de sus usuarios recomendados.
RF-CHAT-01	Un usuario debe poder iniciar una conversación con un usuario recomendado.
RF-CHAT-02	El sistema debe permitir a dos usuarios enviar y recibir mensajes en un chat.
RF-CHAT-03	El sistema debe notificar al usuario cuando reciba un mensaje nuevo.

3.2 Requisitos no funcionales

En este apartado incluiremos los requisitos no funcionales:

ID	Descripción
RNF-SEGU-01	Un usuario solo puede leer y escribir en la base de datos si está autenticado.
RNF-SEGU-02	Un usuario solo puede modificar su propio perfil.
RNF-SEGU-03	Un usuario solo puede actualizar su propio horario.
RNF-SEGU-04	Un usuario solo puede leer y escribir mensajes de un chat si dicho usuario pertenece al chat.
RNF-SEGU-05	El correo del usuario debe de ser válido.
RNF-ACCE-01	La aplicación cambiará a un tema oscuro o un tema claro dependiendo del tema del sistema en el que se ejecute.
RNF-ACCE-02	La interfaz de la aplicación se redistribuirá dependiendo del tamaño de la pantalla donde se ejecute.
RNF-ESCA-01	Los recursos del sistema serán proporcionales al número de usuarios que estén activos.
RNF-DOCU-01	El sistema debe ofrecer información al usuario sobre los términos y condiciones de uso de la aplicación.
RNF-DOCU-02	El sistema debe ofrecer información sobre los distintos módulos y herramientas que componen la aplicación.
RNF-PORT-01	El desarrollo debe favorecer que la aplicación sea fácil de portar a diferentes plataformas.

3.3 Casos de uso

3.3.1 Registro de un usuario en la aplicación

ID	CU-01
Título	Registro de un usuario en la aplicación.
Requisitos	RF-USER-03, RNF-SEGU-01, RNF-SEGU-05
Actores	Usuario, Cliente, Servidor, Navegador web
Descripción	Partiendo de la página de inicio de sesión se accede a la página de registro donde el usuario se registrará y después se vuelve a la página de iniciar sesión.
Precondición	El usuario debe de comenzar en la página de inicio de sesión.
Secuencia	1. El usuario pulsa el botón CREATE ACCOUNT. 2. El cliente carga la página de registro. 3. El usuario introduce su email en el campo Email. 4. El usuario introduce una contraseña en el campo Password. 5. El usuario introduce su nombre en el campo Name. 6. El usuario pulsa en un color para seleccionar el fondo de perfil. 7. El usuario pulsa en el botón CREATE ACCOUNT. 8. El cliente envía la información del formulario al servidor. 9. El servidor comprueba la validez de los datos introducidos. 10. El servidor registra las credenciales en el servidor de autenticación. 11. El servidor introduce los datos del perfil de usuario en la base de datos. 12. El servidor envía la confirmación de que se ha creado la cuenta al cliente. 13. El cliente muestra un aviso diciendo que se ha creado correctamente su cuenta. 14. El cliente carga la página de inicio de sesión. 15. El cliente envía un correo de verificación a la dirección de correo que ha proporcionado. 16. El cliente muestra un aviso en el que dice que se ha enviado un correo de verificación. 17. El usuario accede a su correo en el navegador. 18. El usuario abre el correo enviado y pulsa en el enlace para validar su correo. 19. El enlace abre una página de verificación en una nueva ventana del navegador web y ésta contacta con el servidor. 20. El servidor verifica que el correo es válido.

	21. El servidor actualiza la página abierta y muestra que el correo ha sido verificado.
Postcondición	El usuario ha creado una cuenta en la aplicación.
Secuencia alternativa 1	10. El servidor comprueba que el correo ya está registrado.

3.3.2 Acceder a la página principal de la aplicación

ID	CU-02
Título	Acceder a la página principal de la aplicación.
Requisitos	RF-USER-01, RNF-SEGU-05
Actores	Usuario, Cliente, Servidor
Descripción	Partiendo de la página de inicio de sesión, el usuario debe introducir sus credenciales y acabar en la página principal de la aplicación.
Precondición	1. El usuario debe de estar registrado en la aplicación. 2. El usuario debe de comenzar en la página de inicio de sesión.
Secuencia	1. El usuario introduce su email en el campo Email. 2. El usuario introduce su contraseña en el campo Password. 3. El usuario pulsa el botón LOGIN. 4. El cliente envía los datos de acceso del usuario al servidor. 5. El servidor recibe los datos de acceso. 6. El servidor comprueba la validez de los datos de acceso. 7. El servidor confirma al cliente que los datos de acceso son válidos. 8. El cliente carga la página principal de la aplicación.
Postcondición	El usuario se halla en la página principal de la aplicación.
Secuencia alternativa 1	5. El servidor no recibe las credenciales.
Secuencia alternativa 2	7. El servidor notifica al cliente que el email no ha sido verificado.

3.3.3 Restablecer contraseña de una cuenta de usuario

ID	CU-03
Título	Restablecer contraseña de una cuenta de usuario.
Requisitos	RF-USER-02, RF-USER-04, RNF-SEGU-05
Actores	Usuario, Cliente, Servidor, Navegador web
Descripción	Partiendo de la página de inicio de sesión, el usuario debe acceder a la página de recuperación de contraseña e introducir su email para restablecer su contraseña y acabar en la página de inicio de sesión con la contraseña restablecida.
Precondición	1. El usuario debe de estar registrado en la aplicación. 2. El usuario debe de comenzar en la página de inicio de sesión. 3. El usuario no debe haber iniciado sesión.
Secuencia	1. El usuario pulsa el botón RESET MY PASSWORD. 2. El cliente carga la página de reseteo de contraseña. 3. El usuario introduce su email en el campo Email. 4. El usuario pulsa el botón RESET PASSWORD. 5. El cliente comunica al servidor el restablecimiento de contraseña del correo introducido. 6. El servidor envía al correo del usuario un email para que restablezca la contraseña. 7. El cliente carga la página de inicio de sesión. 8. El usuario accede a su correo y pulsa el enlace del correo enviado. 9. El navegador web abre una nueva página donde el usuario podrá restablecer la contraseña. 10. El usuario introduce la nueva contraseña. 11. El usuario pulsa en el botón confirmar. 12. El servidor actualiza la contraseña.
Postcondición	El usuario ha restablecido su contraseña.
Secuencia alternativa 1	5. El servidor comprueba que el correo introducido no existe.

3.3.4 Modificar el perfil de usuario

ID	CU-04
Título	Modificar el perfil de usuario.
Requisitos	RF-USER-01, RF-USER-05, RNF-SEGU-01, RNF-SEGU-02, RNF-SEGU-05
Actores	Usuario, Cliente, Servidor
Descripción	El usuario debe poder modificar con nueva información su perfil y acabar en la página de opciones.
Precondición	1. El usuario debe haber iniciado sesión en la aplicación. 2. El usuario debe de comenzar en la página principal.
Secuencia	1. El usuario pulsa en el botón Settings. 2. El cliente carga la página Settings. 3. El usuario pulsa en el botón Edit profile info. 4. El cliente carga la página del perfil del usuario. 5. El cliente solicita al servidor la información del perfil. 6. El servidor busca en la base de datos la información del perfil. 7. El servidor envía al cliente la información solicitada. 8. El cliente carga la información recibida. 9. El usuario modifica el nombre en el campo Name. 10. El usuario selecciona un nuevo color de perfil. 11. El usuario pulsa el botón SAVE CHANGES. 12. El cliente envía al servidor la información modificada. 13. El servidor recibe la información modificada. 14. El servidor actualiza la información modificada. 15. El servidor notifica al cliente que la información se ha modificado correctamente. 16. El cliente carga la página anterior (Settings).
Postcondición	El usuario se halla en la página Settings de la aplicación y su perfil ha sido actualizado correctamente.

3.3.5 Seleccionar la ruta del conductor

ID	CU-05
Título	Seleccionar la ruta del conductor.
Requisitos	RF-USER-01, RF-PASS-01, RNF-SEGU-01, RNF-SEGU-02, RNF-DOCU-02
Actores	Usuario, Cliente, Servidor, API de Google Maps
Descripción	El usuario debe acceder a la página de selección de ruta, seleccionarla y acabar en la página de recomendaciones de pasajeros.
Precondición	1. El usuario debe haber iniciado sesión en la aplicación. 2. El usuario debe de comenzar en la página principal.
Secuencia	1. El usuario pulsa el botón I'm the driver. 2. El cliente carga la página de recomendaciones de pasajeros. 3. El usuario pulsa el botón con el icono de editar a la derecha del título del apartado Route. 4. El cliente carga la página de selección de ruta. 5. El cliente solicita al servidor los datos de la ruta anterior. 6. El servidor busca la información correspondiente. 7. El servidor envía la información al cliente. 8. El cliente carga la información correspondiente en cada campo. 9. El usuario introduce en el campo Search una dirección de origen. 10. La API Places de Google Maps autocompletará la dirección introducida. 11. El usuario selecciona la dirección de las sugerencias de Google Maps. 12. El usuario selecciona una universidad en el desplegable Destination. 13. La API Directions de Google Maps dibuja en el mapa la ruta desde el punto de origen a la universidad de destino. 14. El usuario selecciona una distancia máxima en el desplegable Radius. 15. El usuario pulsa el botón SET THIS ROUTE. 16. El cliente recolecta toda la información introducida y generada por la API de Google Maps. 17. El cliente envía al servidor la información. 18. El servidor recibe la información. 19. El servidor calcula las coordenadas de región visitadas. 20. El servidor almacena toda la información en la base de datos. 21. El servidor notifica al cliente que se ha almacenado la

	información correctamente. 22. El cliente carga la página anterior (recomendaciones de pasajeros).
Postcondición	El usuario se halla en la página de recomendaciones de pasajeros y tiene seleccionada una ruta.
Secuencia alternativa 1	9. El usuario pulsa en el mapa.

3.3.6 Introducir o modificar los datos del coche del conductor

ID	CU-06
Título	Introducir o modificar los datos del coche del conductor.
Requisitos	RF-USER-01, RF-PASS-02, RNF-SEGU-01, RNF-SEGU-02, RNF-DOCU-02
Actores	Usuario, Cliente, Servidor
Descripción	El usuario debe acceder a la página del coche, modificar los datos y acabar en la página de recomendaciones de pasajeros.
Precondición	1. El usuario debe haber iniciado sesión en la aplicación. 2. El usuario debe de comenzar en la página principal.
Secuencia	1. El usuario pulsa el botón I'm the driver. 2. El cliente carga la página de recomendaciones de pasajeros. 3. El usuario pulsa el botón con el icono de editar a la derecha del título del apartado Your Car. 4. El cliente carga la página del coche. 5. El cliente solicita al servidor los datos del coche. 6. El servidor busca la información correspondiente. 7. El servidor envía la información al cliente. 8. El cliente carga la información correspondiente en cada campo. 9. El usuario introduce en el campo Brand la marca del coche. 10. El usuario introduce en el campo Model el modelo del coche. 11. El usuario introduce en el campo Plate la matrícula del coche. 12. El usuario selecciona el color del coche.

	13. El usuario pulsa el botón SAVE. 14. El cliente envía al servidor la información introducida. 15. El servidor recibe la información. 16. El servidor almacena toda la información en la base de datos. 17. El servidor notifica al cliente que se ha almacenado la información correctamente. 18. El cliente carga la página anterior (recomendaciones de pasajeros).
Postcondición	El usuario se halla en la página de recomendaciones de pasajeros y ha modificado la información de su coche.

3.3.7 Comenzar una conversación con un pasajero recomendado

ID	CU-07
Título	Comenzar una conversación con un pasajero recomendado.
Requisitos	RF-USER-01, RF-PASS-01, RF-PASS-02, RF-PASS-03, RF-CHAT-01, RF-CHAT-02, RF-CHAT-03, RNF-SEGU-01, RNF-SEGU-02, RNF-SEGU-04, RNF-DOCU-02
Actores	Usuario, Cliente, Servidor
Descripción	El usuario debe acceder a la página de recomendación de pasajeros, seleccionar un pasajero recomendado y enviarle un mensaje.
Precondición	1. El usuario debe haber iniciado sesión en la aplicación. 2. El usuario debe de comenzar en la página principal. 3. El usuario debe tener seleccionado una ruta. 4. El usuario debe tener registrado un coche. 5. Las recomendaciones de pasajeros se deben cargar correctamente.
Secuencia	1. El usuario pulsa el botón I'm the driver. 2. El cliente carga la página de recomendaciones de pasajeros. 3. El cliente envía al servidor una petición para encontrar pasajeros que coincidan con la ruta seleccionada por el usuario. 4. El servidor busca recomendaciones de pasajeros cercanos. 5. El servidor encuentra los pasajeros.

	6. El servidor envía las recomendaciones de pasajeros al cliente 7. El cliente carga las recomendaciones. 8. El usuario pulsa en el icono de chat en el primer pasajero recomendado. 9. El cliente envía al servidor una petición para crear una nueva sala de chat. 10. El servidor recibe la petición. 11. El servidor crea la sala de chat en la colección chat. 12. El servidor notifica al cliente. 13. El cliente carga en una nueva ventana un chat. 14. El usuario escribe un mensaje en el campo Write your message here. 15. El usuario pulsa el icono de enviar mensaje. 16. El cliente muestra el mensaje enviado en el chat con un icono de carga. 17. El cliente envía al servidor la información del mensaje. 18. El servidor recibe la información del mensaje nuevo. 19. El servidor almacena el mensaje en la subcolección mensajes de la sala de chat. 20. El servidor notifica al cliente que se ha guardado el mensaje. 21. El cliente muestra el mensaje enviado sin el icono de carga.
Postcondición	El usuario se halla en la página del chat con el pasajero recomendado y le ha enviado un mensaje.

3.3.8 Seleccionar el punto de recogida del pasajero

ID	CU-08
Título	Seleccionar el punto de recogida del pasajero.
Requisitos	RF-USER-01, RF-DRIV-01, RNF-SEGU-01, RNF-SEGU-02
Actores	Usuario, Cliente, Servidor
Descripción	El usuario debe acceder a la página de selección del punto de recogida, seleccionarlo y acabar en la página de recomendaciones de conductores.
Precondición	1. El usuario debe haber iniciado sesión en la aplicación. 2. El usuario debe de comenzar en la página principal.

Secuencia	1. El usuario pulsa el botón I'm the passenger. 2. El cliente carga la página de recomendaciones de conductores. 3. El usuario pulsa el botón con el icono de editar a la derecha del título del apartado Pickup Point. 4. El cliente carga la página de selección del punto de recogida. 5. El cliente solicita al servidor los datos del punto de recogida anterior. 6. El servidor busca la información correspondiente. 7. El servidor envía la información al cliente. 8. El cliente carga la información correspondiente en cada campo. 9. El usuario introduce en el campo Search la dirección del punto de recogida. 10. La API Places de Google Maps autocompletará la dirección introducida. 11. El usuario selecciona la dirección de las sugerencias de Google Maps. 12. El usuario selecciona una distancia máxima en el desplegable Radius. 13. La API de Google Maps dibuja en el mapa el punto de recogida con una circunferencia indicando el radio máximo. 14. El usuario selecciona una universidad en el desplegable Destination. 15. El usuario pulsa el botón SET PICK UP POINT. 16. El cliente recolecta toda la información introducida y generada por la API de Google Maps. 17. El cliente envía al servidor la información. 18. El servidor recibe la información. 19. El servidor calcula las coordenadas de región visitadas. 20. El servidor almacena toda la información en la base de datos. 21. El servidor notifica al cliente que se ha almacenado la información correctamente. 22. El cliente carga la página anterior (recomendaciones de conductores).
Postcondición	El usuario se halla en la página de recomendaciones de conductores y tiene seleccionada un punto de recogida.
Secuencia alternativa 1	9. El usuario pulsa en el mapa.

3.3.9 Comprobar el horario de un conductor recomendado

ID	CU-09
Título	Comprobar el horario de un conductor recomendado.
Requisitos	RF-USER-01, RF-DRIV-02, RF-TIME-01, RF-TIME-02, RNF-SEGU-01, RNF-SEGU-03
Actores	Usuario, Cliente, Servidor
Descripción	El usuario debe acceder a la página de recomendación de conductores, seleccionar el primer conductor recomendado con horario registrado y ver sus turnos.
Precondición	1. El usuario debe haber iniciado sesión en la aplicación. 2. El usuario debe de comenzar en la página principal. 3. El usuario debe tener seleccionado un punto de recogida. 4. Las recomendaciones de conductores se deben cargar correctamente.
Secuencia	1. El usuario pulsa el botón I'm the passenger. 2. El cliente carga la página de recomendaciones de conductores. 3. El cliente envía al servidor una petición para encontrar conductores que coincidan con el punto de recogida seleccionado por el usuario. 4. El servidor busca recomendaciones de conductores cercanos. 5. El servidor encuentra los conductores. 6. El servidor envía las recomendaciones de conductores al cliente 7. El cliente carga las recomendaciones. 8. El usuario pulsa en el icono de horario en el primer conductor recomendado que muestre uno. 9. El cliente envía al servidor una petición para obtener el horario del conductor. 10. El servidor recibe la petición. 11. El servidor busca el horario del usuario. 12. El servidor notifica al cliente. 13. El cliente carga en una nueva ventana el horario del conductor.
Postcondición	El usuario se halla en la página del horario del conductor recomendado.

3.3.10 Introducir o modificar los horarios de entrada y salida

ID	CU-10
Título	Introducir o modificar los horarios de entrada y salida.
Requisitos	RF-USER-01, RF-TIME-01, RF-TIME-02, RNF-SEGU-01, RNF-SEGU-03
Actores	Usuario, Cliente, Servidor
Descripción	El usuario debe acceder a la página de horarios, seleccionar un turno y guardar los cambios.
Precondición	1. El usuario debe haber iniciado sesión en la aplicación. 2. El usuario debe de comenzar en la página principal.
Secuencia	1. El usuario pulsa en el botón Timetable. 2. El cliente carga la página de horarios. 3. El cliente envía al servidor una petición para obtener el horario del usuario. 4. El servidor recibe la petición. 5. El servidor busca el horario del usuario. 6. El servidor notifica al cliente. 7. El cliente carga el horario del usuario. 8. El usuario pulsa el botón ADD OR UPDATE SHIFTS. 9. El cliente carga una página para seleccionar las horas de entrada y salida y los días. 10. El usuario pulsa en la hora de la entrada y selecciona una hora 11. El usuario pulsa en la hora de la salida y selecciona una hora. 12. El usuario pulsa las casillas de verificación para marcar los días. 13. El usuario pulsa el botón SET. 14. El cliente cierra la página de selección de hora y días. 15. El usuario pulsa el botón SAVE CHANGES. 16. El cliente envía al servidor la nueva información del turno. 17. El servidor recibe la información. 18. El servidor almacena el nuevo horario del usuario. 19. El servidor notifica al cliente.
Postcondición	El usuario se halla en la página del horario y con los horarios actualizados.

3.3.11 Leer mensajes recibidos de un chat

ID	CU-11
Título	Leer mensajes recibidos de un chat.
Requisitos	RF-USER-01, RF-CHAT-01, RF-CHAT-02, RF-CHAT-03, RNF-SEGU-01, RNF-SEGU-02, RNF-SEGU-04
Actores	Usuario, Cliente, Servidor
Descripción	Partiendo de la página principal, el usuario debe acceder a la lista de mensajes recibidos y abrir el chat para leerlos.
Precondición	1. El usuario debe haber iniciado sesión en la aplicación. 2. El usuario debe de comenzar en la página principal. 3. El usuario debe de tener un chat con mensajes no leídos.
Secuencia	1. El usuario pulsa el botón Messages. 2. El cliente carga la página de la lista de chats. 3. El cliente envía al servidor una petición para obtener la lista de salas de chat activas del usuario. 4. El servidor recibe la petición. 5. El servidor busca las salas de chat del usuario. 6. El servidor envía las salas de chat al cliente. 7. El cliente carga las salas de chat del usuario. 8. El usuario hace click en la primera sala de chat. 9. El cliente carga en una nueva ventana chat. 10. El cliente envía al servidor una petición para obtener la lista de mensajes del chat. 11. El servidor recibe la petición. 12. El servidor busca los mensajes de la sala del chat. 13. El servidor envía los mensajes de la sala del chat al cliente. 14. El servidor marca como leídos los mensajes de esa sala de chat. 15. El cliente carga los mensajes en el chat.
Postcondición	El usuario se halla con el primer chat de su lista de salas de chat abierto con los mensajes cargados en el cliente.

3.3.12 Leer los términos y condiciones de la aplicación

ID	CU-12
Título	Leer los términos y condiciones de la aplicación.
Requisitos	RF-USER-02, RNF-DOCU-01
Actores	Usuario, Cliente
Descripción	Partiendo de la página de inicio de sesión, el usuario debe acabar en la página de términos y condiciones.
Precondición	1. El usuario no debe haber iniciado sesión en la aplicación. 2. El usuario debe de comenzar en la página de inicio de sesión.
Secuencia	1. El usuario pulsa sobre el banner inferior que pone “By using our website, you accept that we use cookies to perform analytics and produce content tailored to your interests. Read our policy on cookies usage”. 2. El cliente carga la página de términos y condiciones de la aplicación.
Postcondición	El usuario se halla en la página términos y condiciones de la aplicación.

3.4 Diagramas de navegación

En este apartado vamos a mostrar un diagrama de navegación de la aplicación para conocer a alto nivel cómo se navega en general por ella. Los módulos más característicos de los diagramas de navegación están pintados de colores diferentes para que sea más intuitivo.

En la Figura 3.4.1 se muestra el esquema general de navegación de la aplicación. El usuario primero encuentra la página de acceso Login junto a las relacionadas a este servicio (Restore, Register, Terms...). Una vez accedido al sistema se pasa a la página Home desde la que se puede acceder al resto de

funcionalidades (horario, chat, listas de conductores y pasajeros o la configuración de la herramienta).

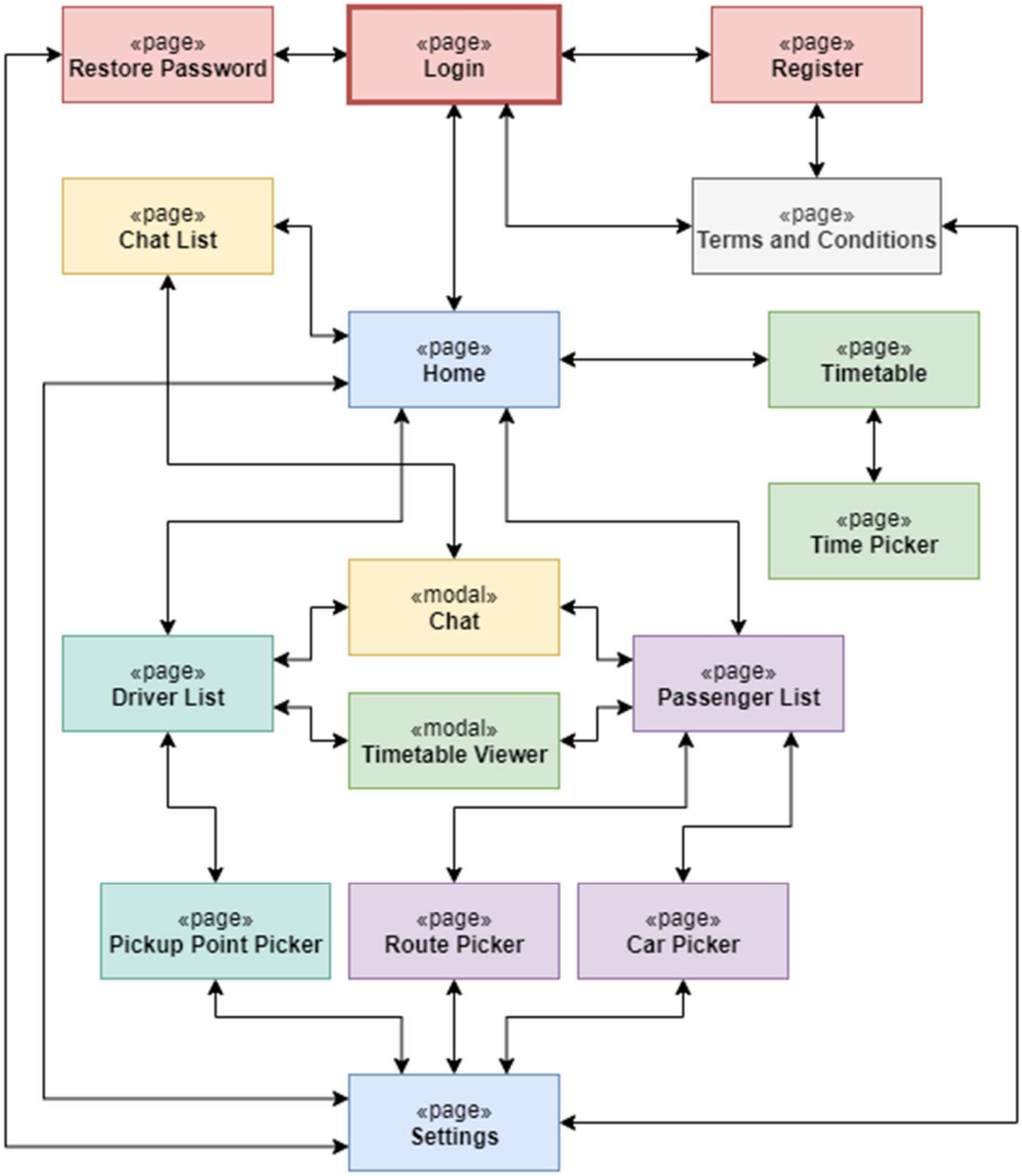


Figura 3.4.1: Diagrama de navegación de la aplicación simplificado.

Ahora vamos a mostrar los diagramas de navegación más detallados divididos en los principales módulos por los que está formado la aplicación.

En la Figura 3.4.2 estamos en la página de login y desde ahí tenemos la opción de crearnos una cuenta, leer los términos y condiciones de la aplicación e iniciar sesión, en cuyo caso accederíamos a la página principal.

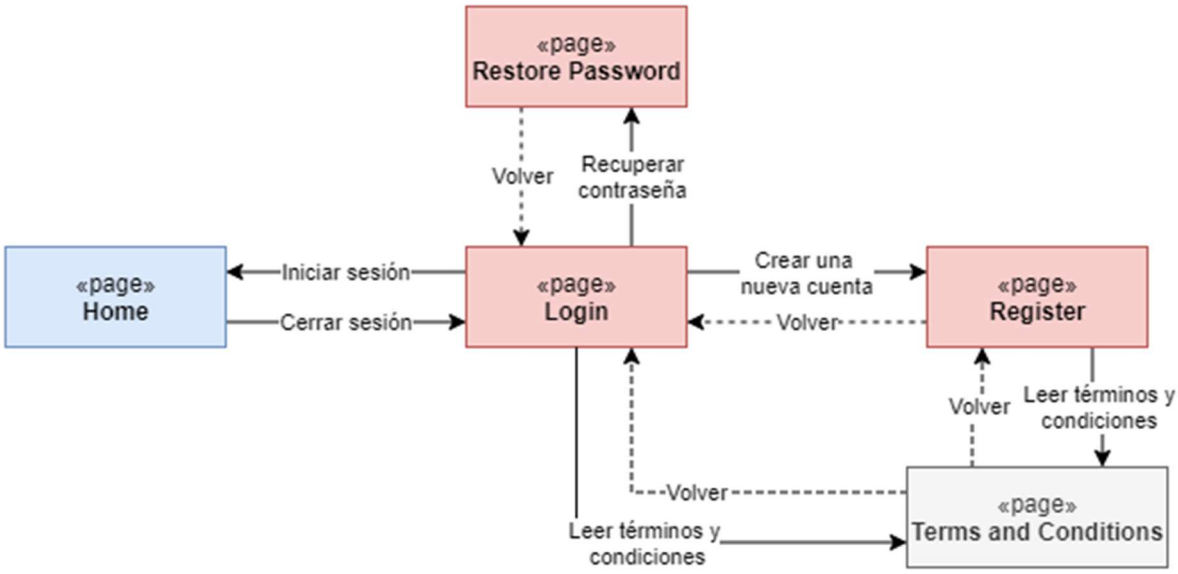


Figura 3.4.2: Diagrama de navegación del módulo de autenticación.

En la Figura 3.4.3 se muestra cómo se pueden seleccionar la ruta y el coche del usuario, ver las recomendaciones de pasajeros e incluso entablar una conversación con ellos o comprobar sus horarios.

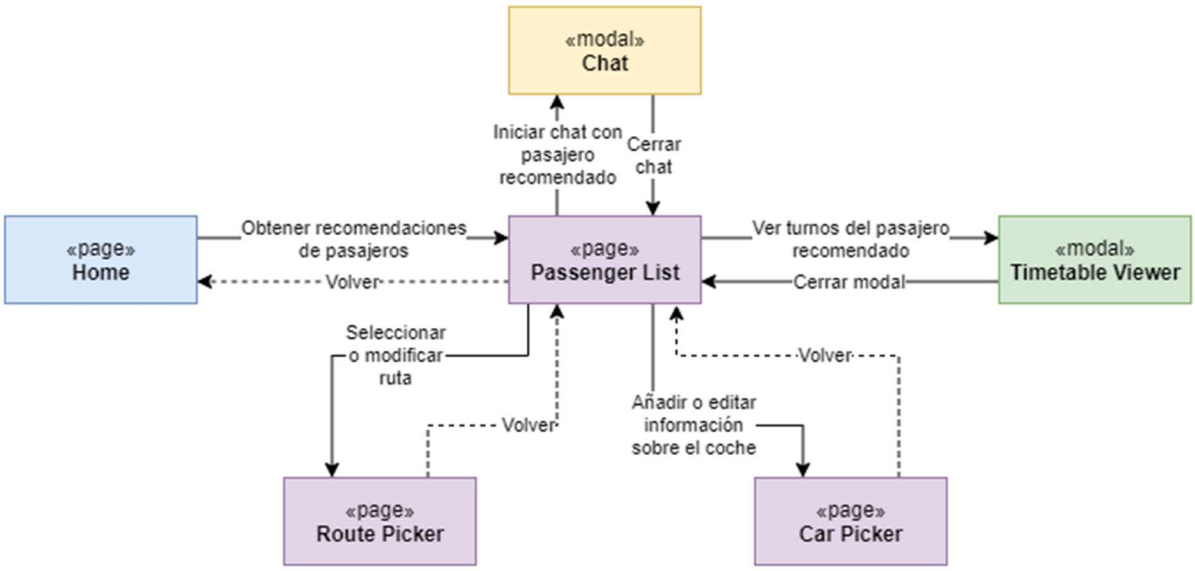


Figura 3.4.3: Diagrama de navegación del módulo de recomendaciones de pasajeros a conductores.

En la Figura 3.4.4 se muestra cómo se pueden seleccionar el punto de recogida, ver las recomendaciones de conductores o iniciar una conversación con ellos.

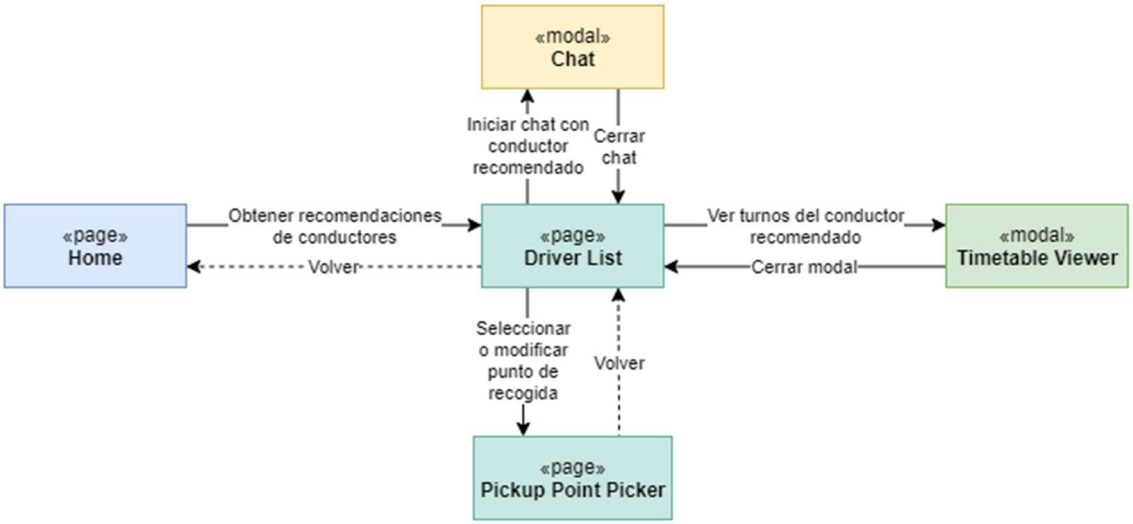


Figura 3.4.4: Diagrama de navegación del módulo de recomendaciones de conductores a pasajeros.

En la Figura 3.4.5 vemos la lista de chats de la aplicación y cómo las demás páginas están conectadas con ésta en caso de recibir notificaciones de mensajes nuevos.

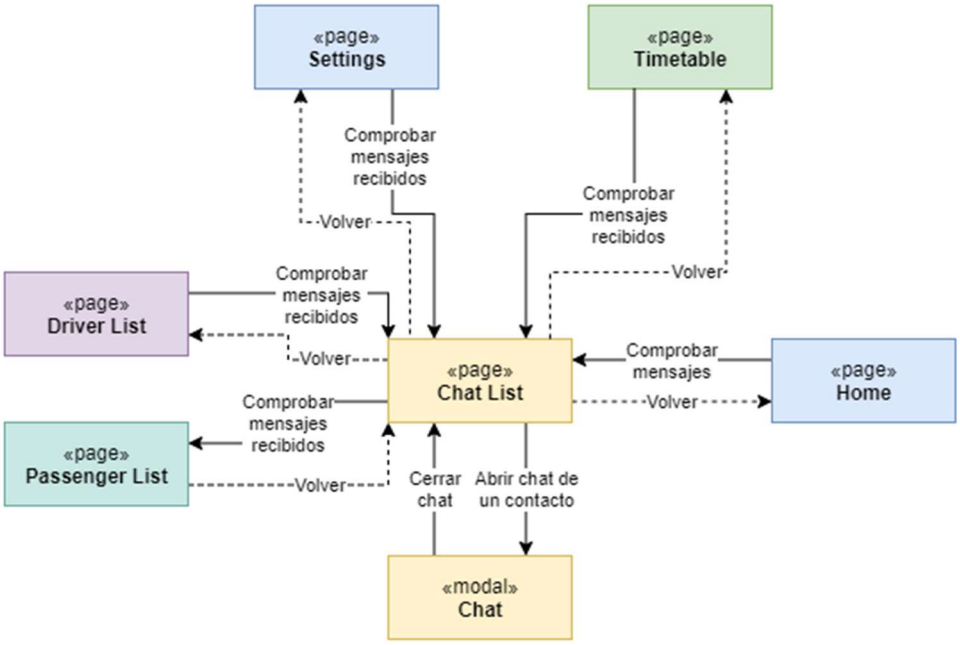


Figura 3.4.5: Diagrama de navegación del módulo de mensajería.

En la Figura 3.4.6 se pueden modificar los horarios del usuario y en la Figura 3.4.7 se observa como desde la página de opciones se pueden acceder a muchas funcionalidades de la aplicación.

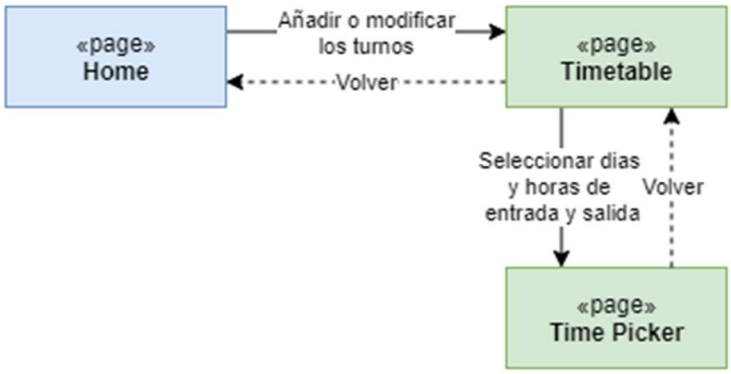


Figura 3.4.6: Diagrama de navegación del módulo de horarios.

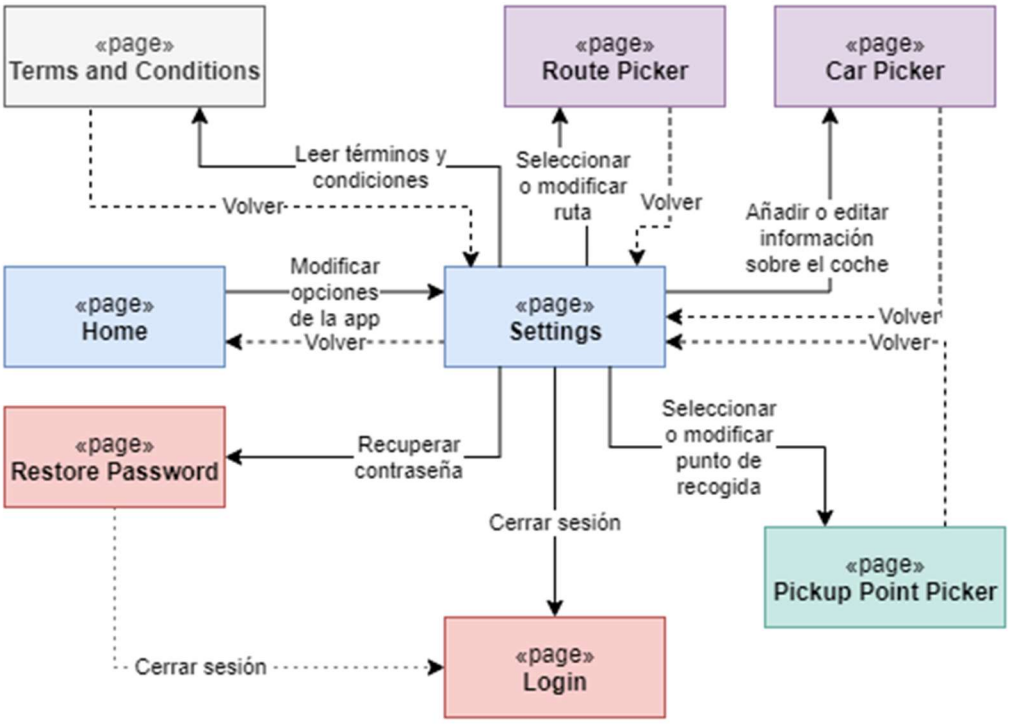


Figura 3.4.7: Diagrama de navegación de la página de opciones.

3.5 Diagramas de clase

En este apartado se van a exponer los diferentes diagramas de clase de la aplicación divididos entre los tipos personalizados de datos que se han creado para facilitar el desarrollo, los diagramas de cada página del cliente y, por último, los diagramas de las funciones en el servidor (Cloud Functions).

3.5.1 Tipos y clases personalizadas

Para facilitar el desarrollo de la aplicación se han utilizado tipos personalizados de datos, todos estos aparecen en la Figura *3.5.1*.

Para los datos del perfil se ha utilizado la interfaz User, que está compuesta por las interfaces Profile, que tiene información del perfil, Driver, que almacenan la información sobre la ruta, puntos de destino y origen, entre otros, la interfaz Car que contiene información sobre el coche del conductor y la interfaz Passenger, donde se guarda la información sobre el punto de recogida y el punto de origen y destino.

También se crearon interfaces para las peticiones y respuestas con el servidor, como MatchmakingRequest y MatchmakingResponse, ambas encargadas de estructurar la información de las recomendaciones de usuarios.

Estas interfaces hacen uso de otras interfaces más simples todavía: Point para una coordenada con valores latitud y longitud, Tile para una coordenada de región, Location para un par origen destino, que a su vez utiliza la interfaz Place que contiene direcciones e información de Google Maps, y por último Match, que sirve para estructurar los datos de cada usuario recomendado.

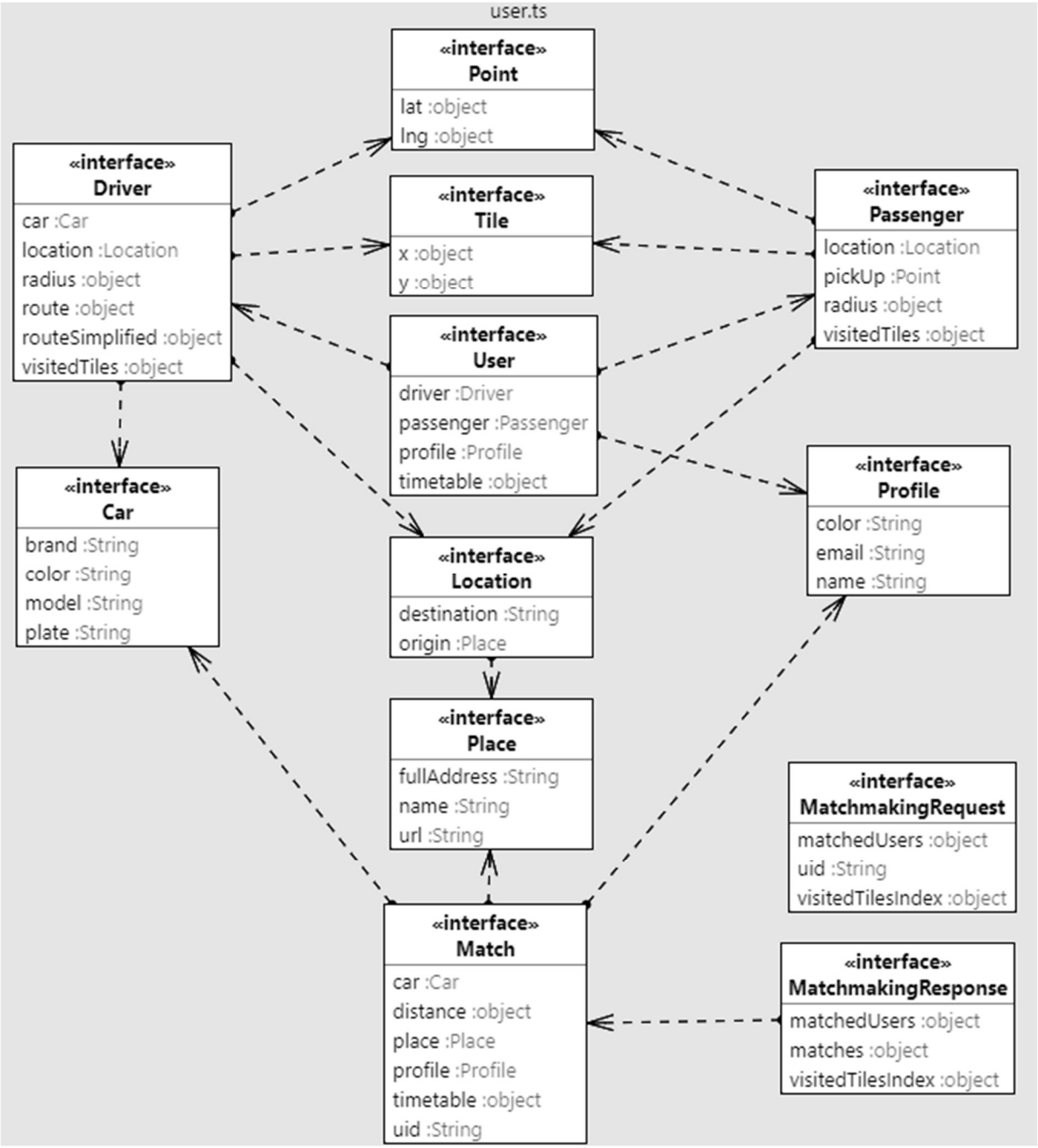


Figura 3.5.1: Diagrama de clases de las interfaces relacionadas con el usuario utilizadas.

En la Figura *3.5.2* se observan las clases que tienen que ver con los mensajes y chats. Cabe destacar que las interfaces como ChatRoomRequest y MessageRequest se utilizan para las peticiones enviadas al servidor.

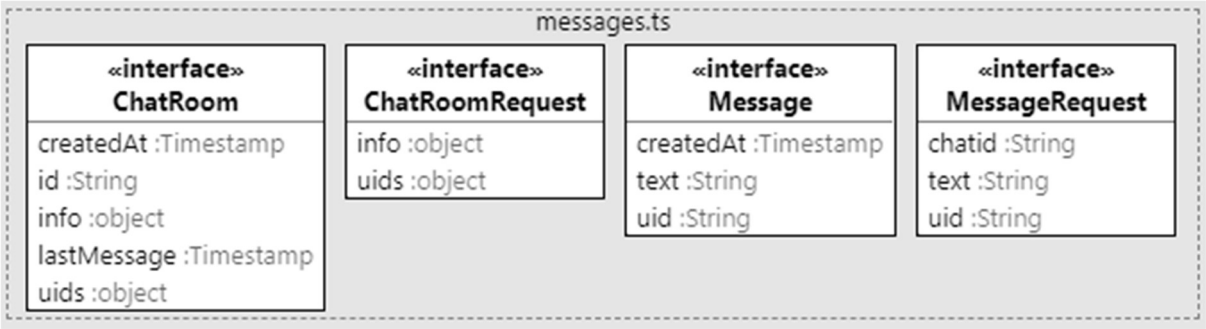


Figura 3.5.2: Diagrama de clases de las interfaces relacionadas con el chat.

Para crear la selección de turnos con diferentes horarios en diferentes días se creó la clase TimetableClass que centralizaba las funciones necesarias para su desarrollo como aparece en la Figura 3.5.3. La mayoría de la aplicación ha sido desarrollada sobre interfaces porque no se requería de clases extra aparte de las que conforman cada componente de la aplicación como se puede apreciar en casi todos los diagramas de clase.

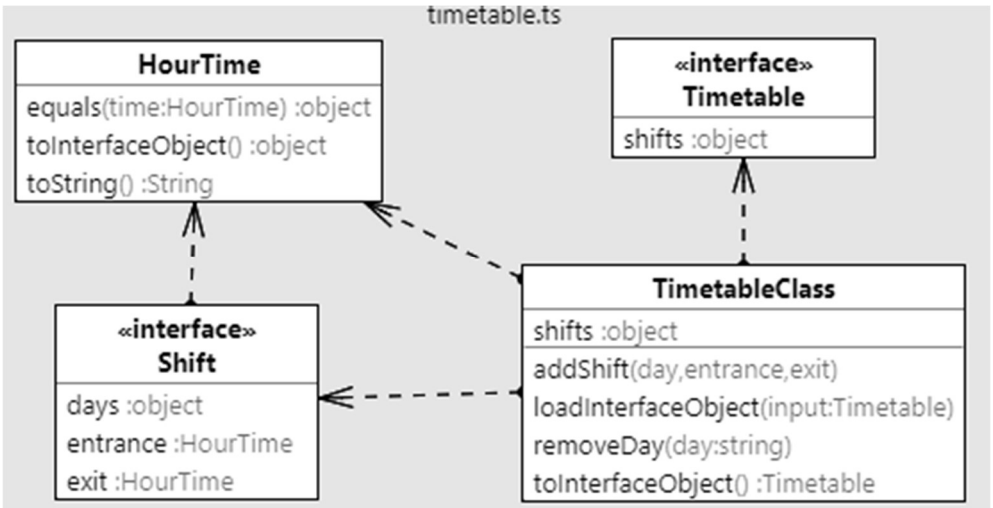


Figura 3.5.3: Diagrama de clases de las interfaces y clases relacionadas con los horarios.

Los conjuntos (Set) en TypeScript solamente funcionan para tipos de datos primitivos, por lo que se ha tenido que crear una clase MatchSet que funciona como un conjunto de recomendaciones (Figura 3.5.4), pues en la lógica de la aplicación no pueden repetirse los usuarios recomendados.

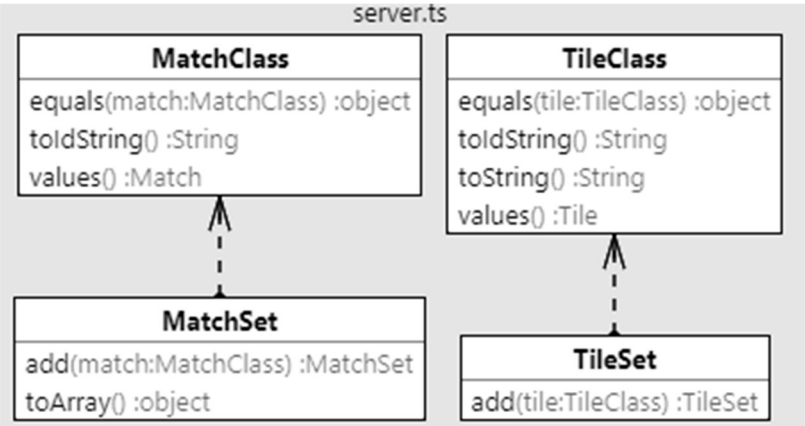


Figura 3.5.4: Diagrama de clases de las clases y conjuntos que se utilizan en el servidor para el cálculo de coordenadas de región y generación de recomendaciones.

3.5.2 Cliente

Las aplicaciones hechas en Ionic están basadas en bloques llamados Componentes, que permiten construir rápidamente la interfaz de usuario. Las páginas de la aplicación como tal son componentes, por lo que cada página de la aplicación contiene los métodos necesarios para el funcionamiento de la misma.

Los componentes no deben buscar o guardar datos directamente, deben centrarse en presentar datos y delegar el acceso a los datos a un servicio. Los servicios son clases que permiten compartir información entre otras clases que no se conocen entre sí.

El enrutador (route) de Angular2 es una de las bibliotecas más importantes en una aplicación Angular. Sin él, las aplicaciones serían aplicaciones de una sola vista o no podrían mantener su estado de navegación en las recargas del navegador. Con él podemos estructurar la navegación de la aplicación. Para lograr esto cada página importa una clase ModuleRouting en la que se especifica las direcciones de las páginas hijas. Así se consigue una modularización total de cada módulo de la aplicación.

En el módulo Auth (Figura 3.5.5) se pueden observar diferentes tipos de componentes: el servicio AuthService que se encarga de las principales funciones de inicio de sesión, registro, etc., LoginPage, RegisterPage y RestorePasswordPage que son las páginas y por último AuthGuard y LoggedGuard, estos son *guards*, los *guards* se encargan de controlar el acceso a diferentes páginas en base a una serie de condiciones especificadas en su lógica.

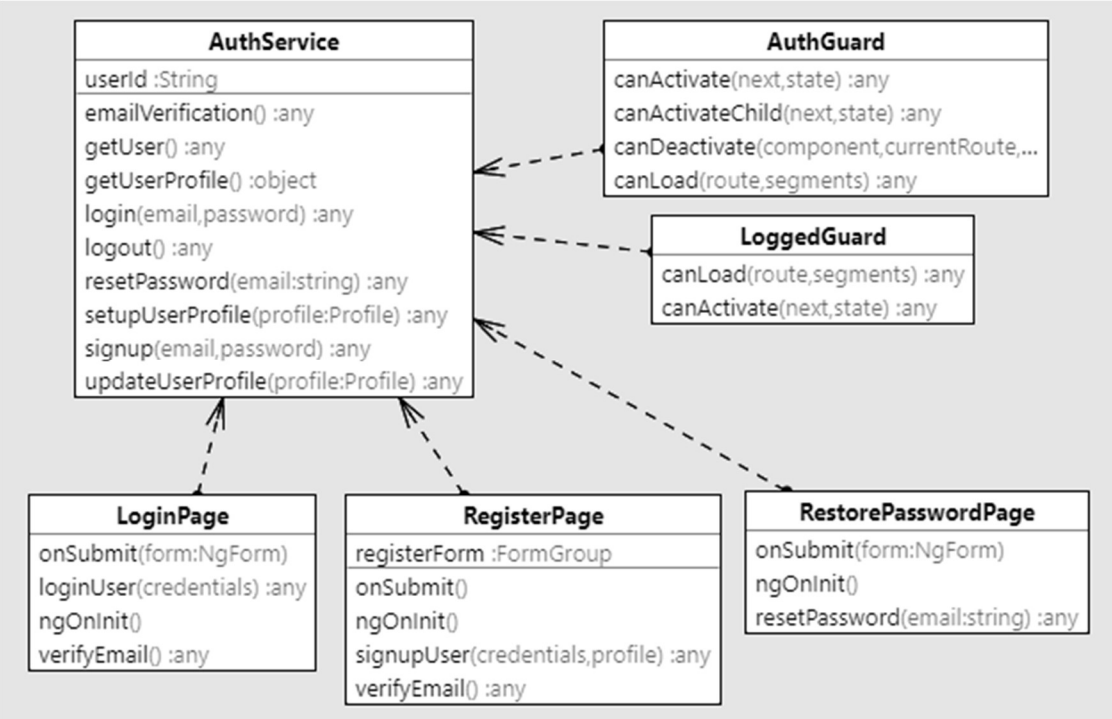


Figura 3.5.5: Diagrama de clases del módulo Auth de la aplicación.

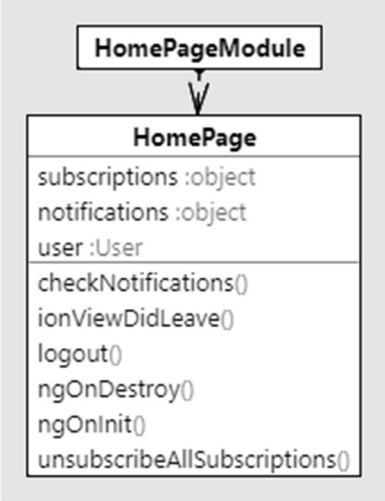


Figura 3.5.6: Diagrama de clases de la página principal de la aplicación.

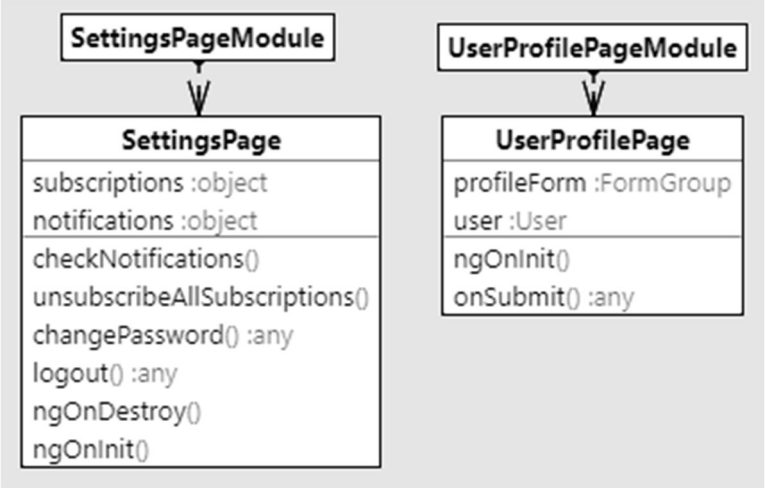


Figura 3.5.7: Diagrama de clases del módulo de opciones de la aplicación.

Hay páginas muy simples como la página de opciones (Figura 3.5.7) o la página principal (Figura 3.5.6), pero todo cambia cuando hay que hablar sobre los módulos Driver (Figura 3.5.8) y Passenger (Figura 3.5.9). Estos módulos son muy parecidos entre ellos, ambos hacen uso del servicio DriverPassengerService que se encarga de compartir la información entre los diferentes componentes y las funciones que comunican con el servidor. También poseen ambos los módulos de información, formados por componentes diferentes que simplemente muestran la información del usuario como el punto de recogida o la ruta. Ambos poseen páginas para seleccionar esta información, ya sea la página de selección de ruta o punto de recogida, en la que se realizan la carga de los mapas.

.

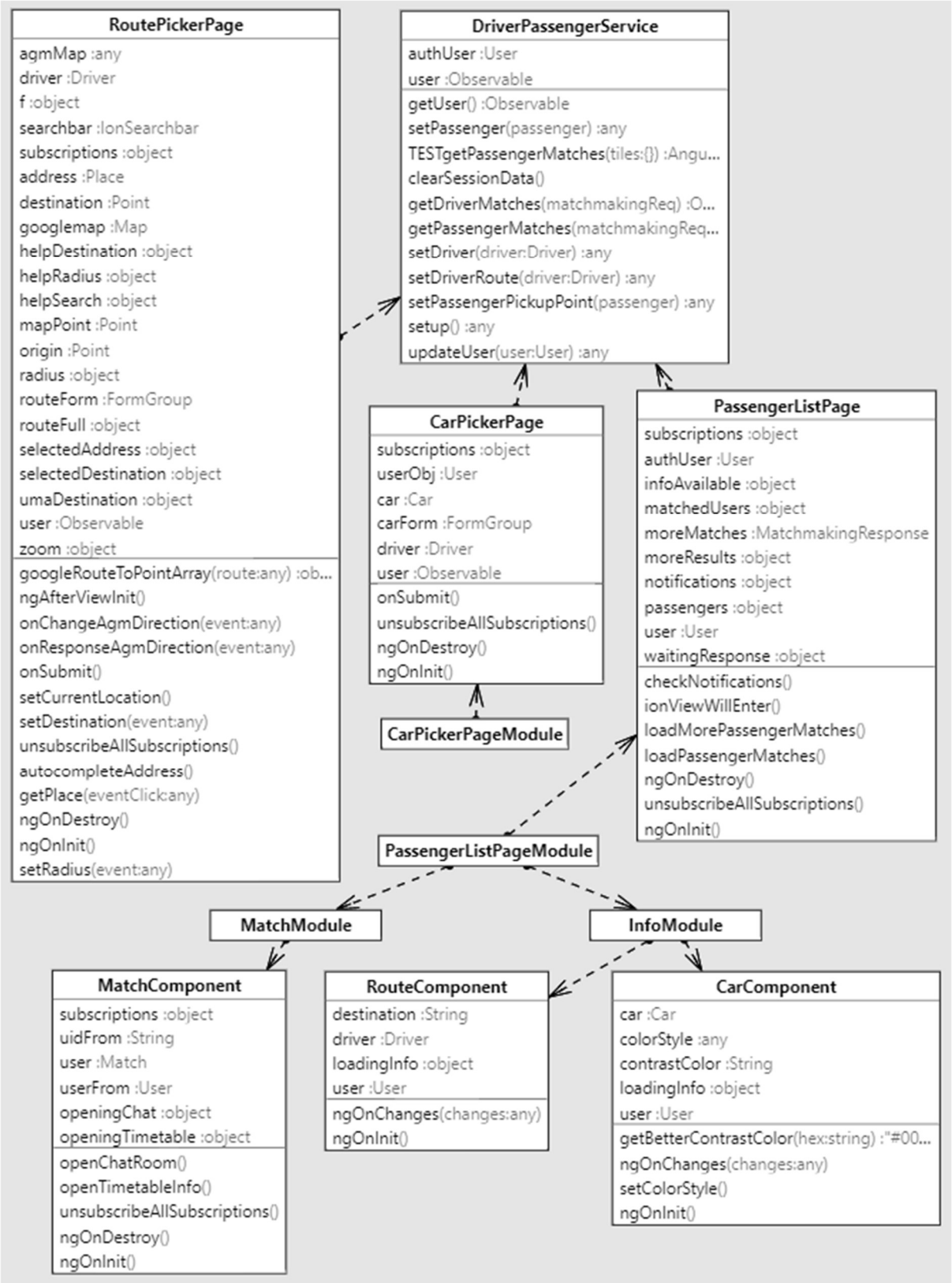


Figura 3.5.8: Diagrama de clases del módulo del conductor de la aplicación.

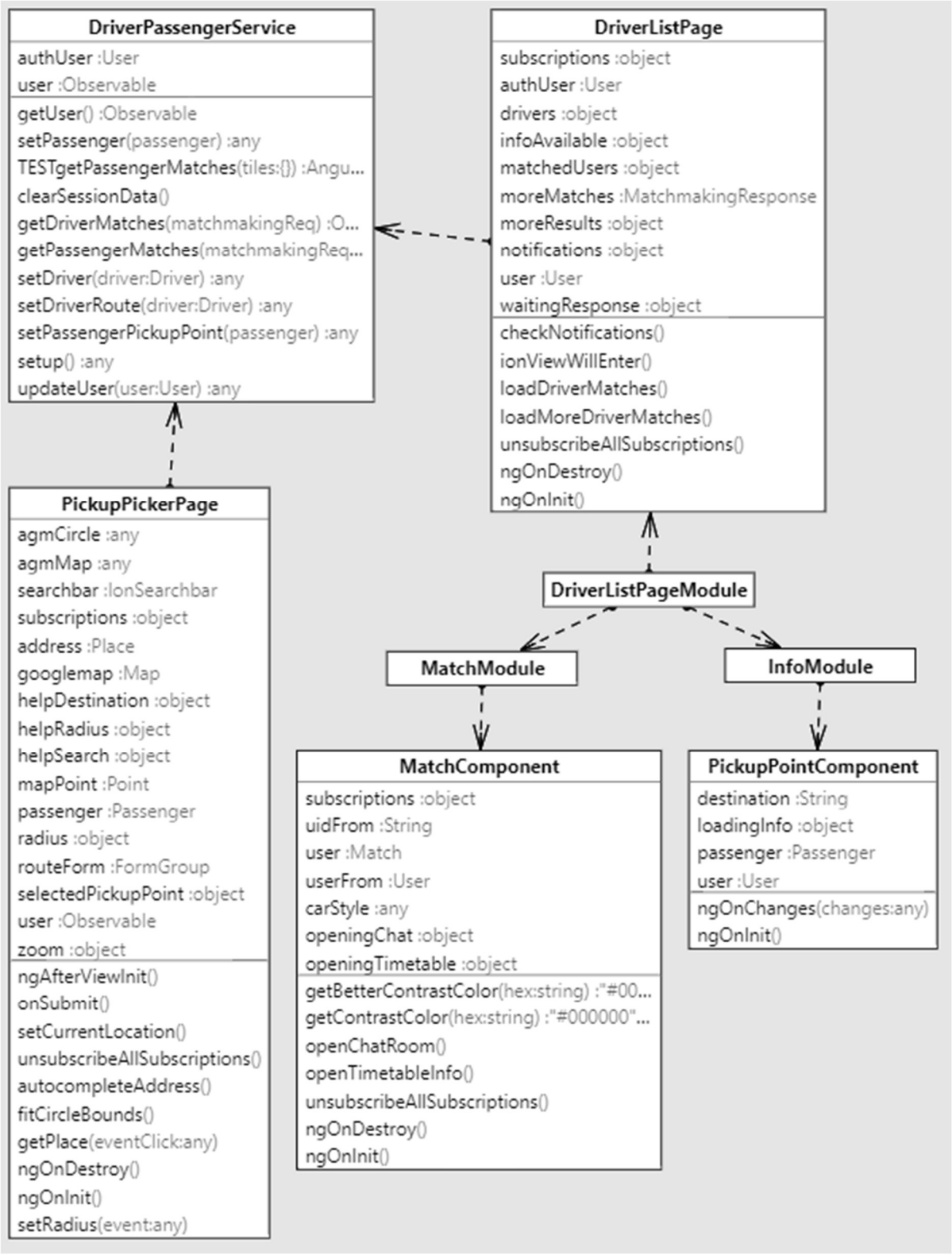


Figura 3.5.9: Diagrama de clases del módulo del pasajero de la aplicación.

Para el módulo de mensajería (Figura 3.5.10) se ha utilizado un componente modal para el chat. Los componentes modal son ventanas emergentes en las

que se puede cargar un componente, como por ejemplo una página, en este caso la página del chat. De esta forma cuando en una página de recomendaciones se pulsa el icono de iniciar chat, el chat se muestra por encima de la página y esta no tiene que perder los usuarios recomendados hasta entonces.

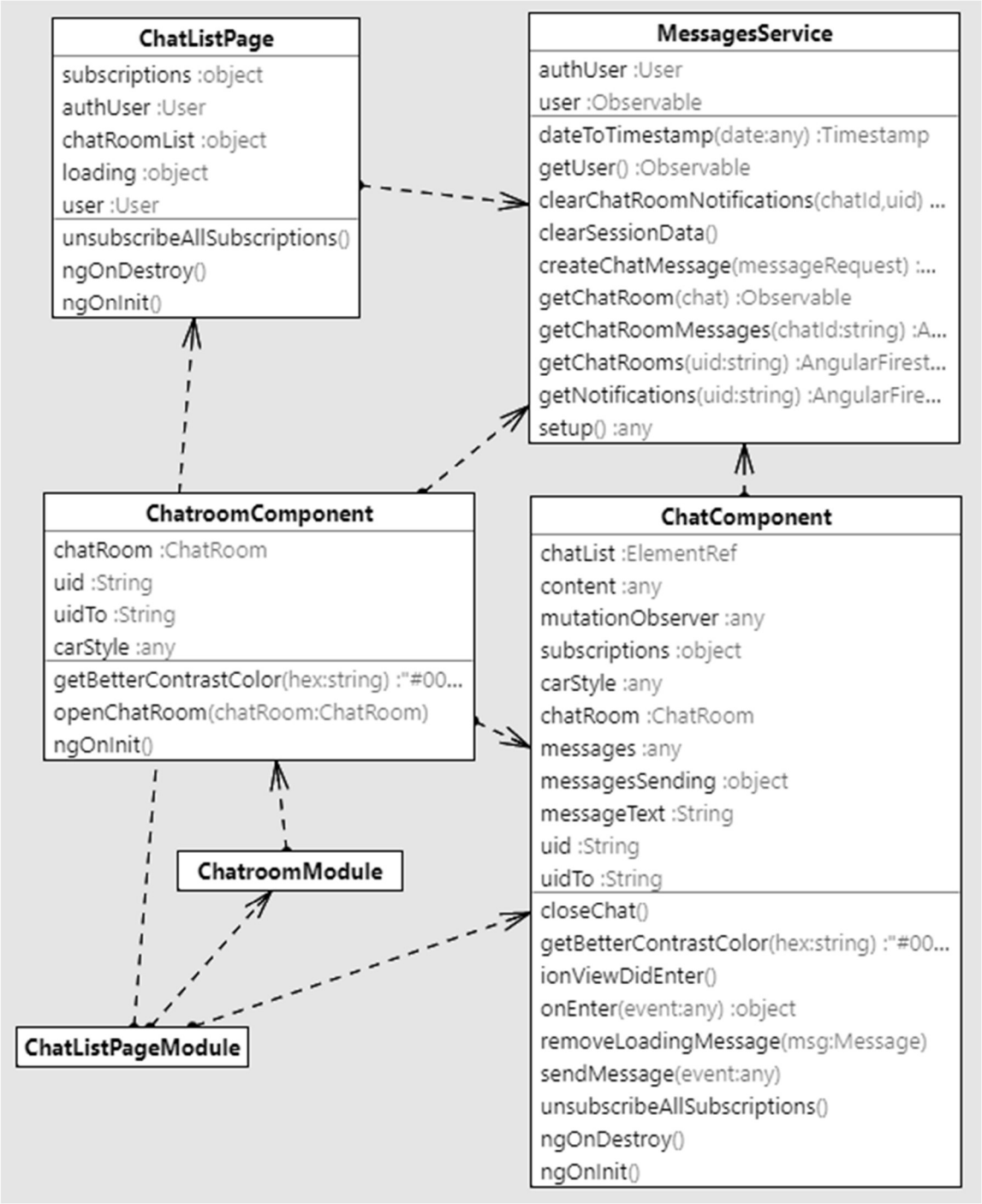


Figura 3.5.10: Diagrama de clases del módulo de mensajería de la aplicación.

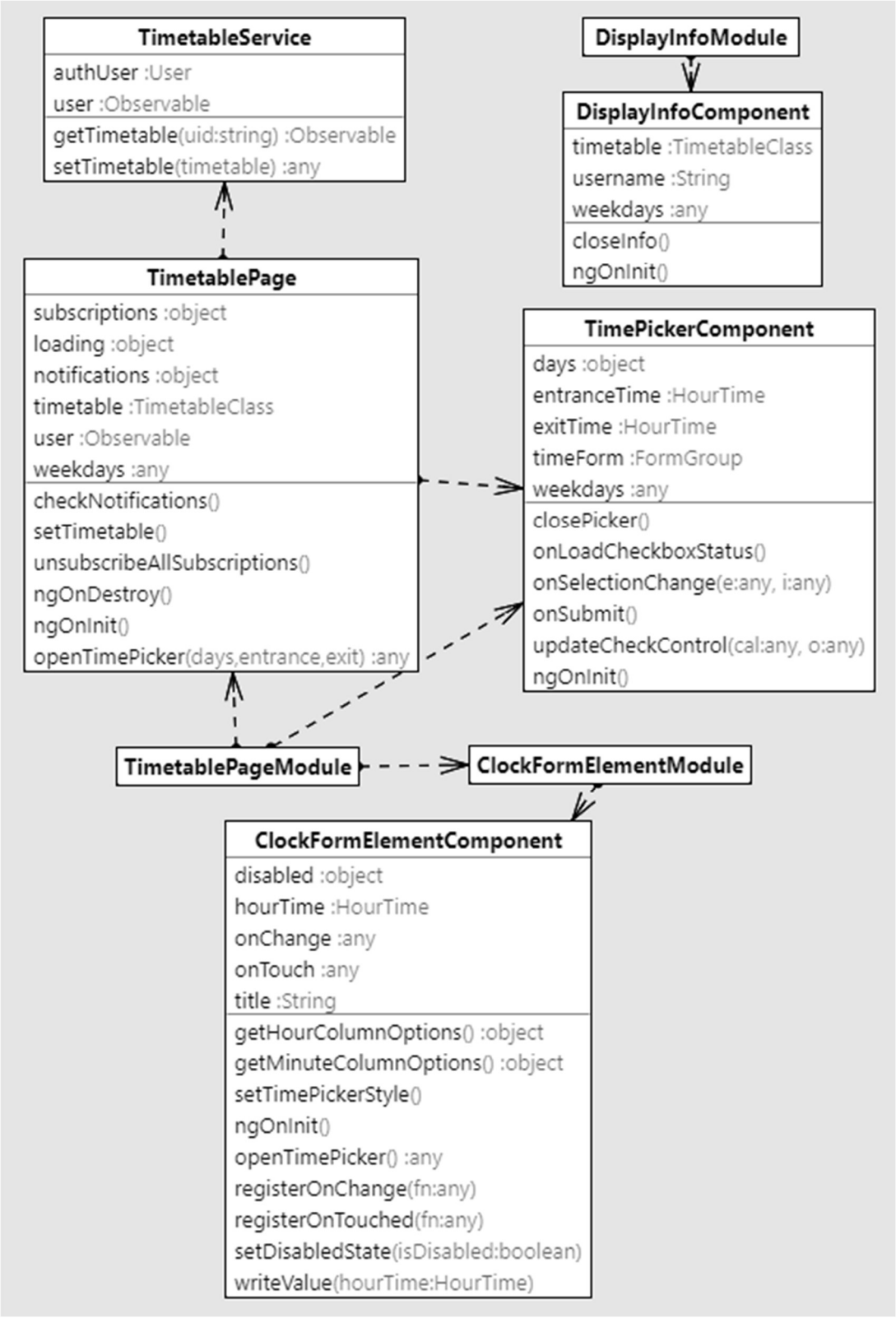


Figura 3.5.11: Diagrama de clases del módulo de horarios de la aplicación.

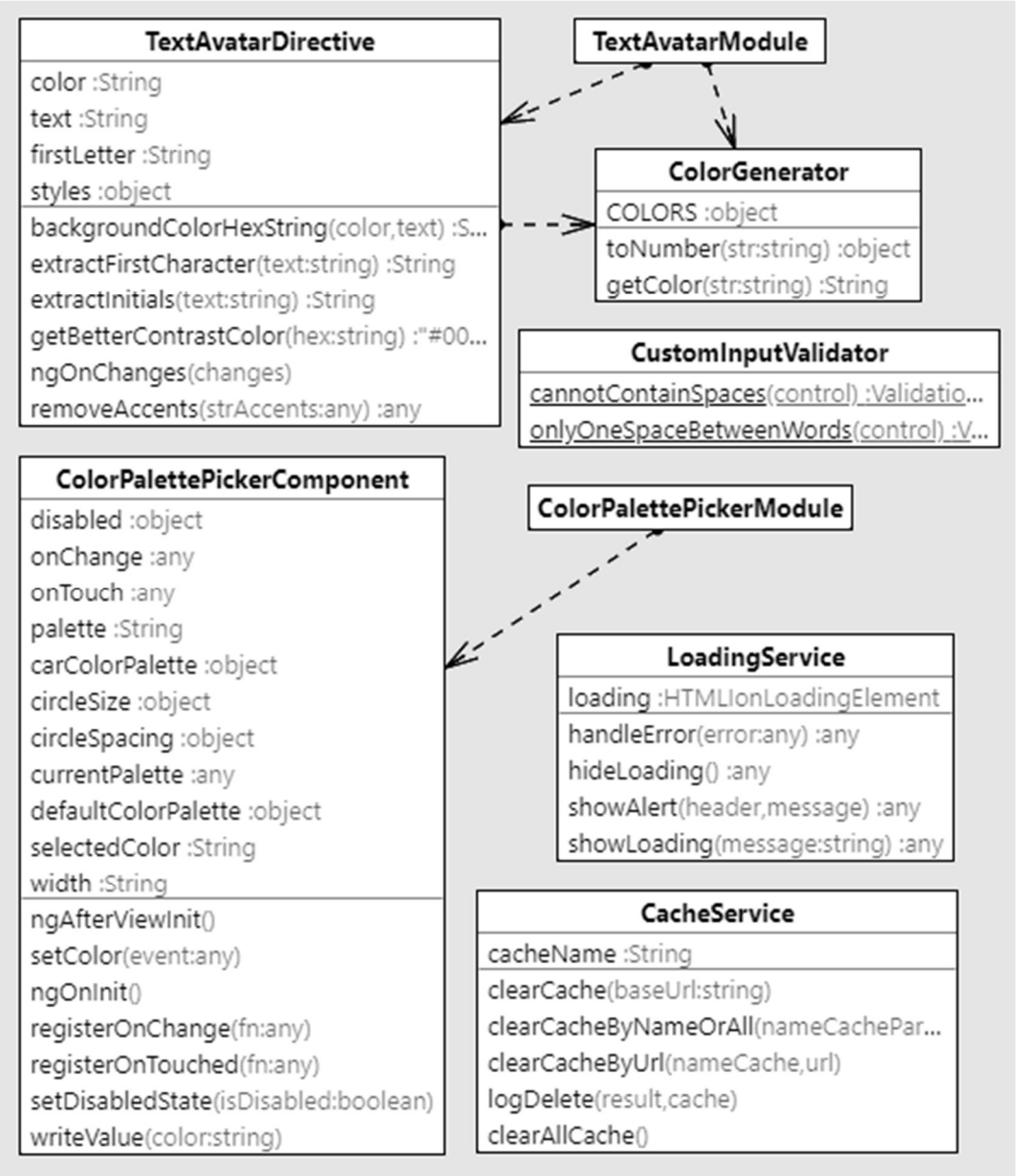


Figura 3.5.12: Diagrama de clases del módulo de recursos compartidos de la aplicación.

Para finalizar la aplicación posee módulos, componentes y servicios compartidos entre diferentes páginas (Figura 3.5.12), como por ejemplo el selector de color, el avatar de cada usuario o el servicio LoadingService que se encarga de centralizar las funciones de carga y ventanas de alerta.

3.5.3 Servidor

El servidor está dividido en diferentes módulos, chat, map y matching, como se puede ver en la Figura 3.5.13.

En chat están todas las funciones y métodos relacionados con la mensajería, como enviar mensajes, obtener las salas de chat, o actualizar las notificaciones de mensajes sin leer. Para el correcto formato de los valores de fecha de los campos se utiliza la función dateToTimestamp que recibe un objeto date y genera una fecha en el formato de Firebase que es un Timestamp.

Map se encarga de las conversiones entre tipos de coordenada, ya sea pasar de puntos en latitud y longitud a coordenadas de región o pasar de un TileSet a un array de coordenadas de región, y simplificación de rutas. También posee una función para calcular la mínima distancia entre un punto de recogida y una ruta.

Por último, matching, esta contiene los algoritmos de emparejamiento para conductores y pasajeros. En cada uno de ellos hacen uso de la clase utils.ts que se encarga de almacenar funciones útiles de carácter general como por ejemplo pasar el objeto Passenger a Match u obtener los ids de usuario de cada Match en un MatchSet.

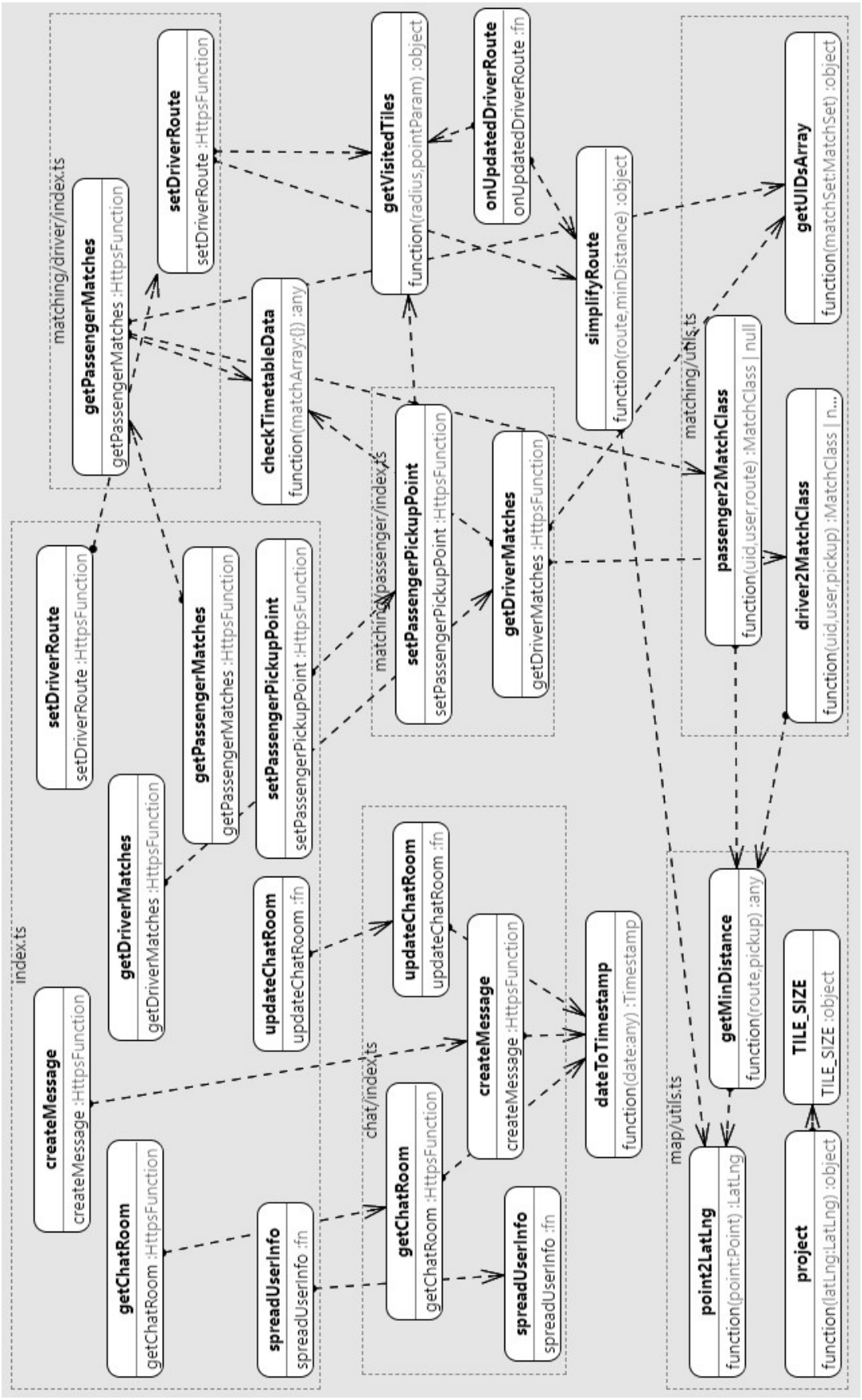


Figura 3.5.13: Diagrama de clases de las funciones del servidor.

3.6 Diagramas de secuencia

En este apartado se van a mostrar los diagramas de secuencia simplificados más representativos de las características de la aplicación.

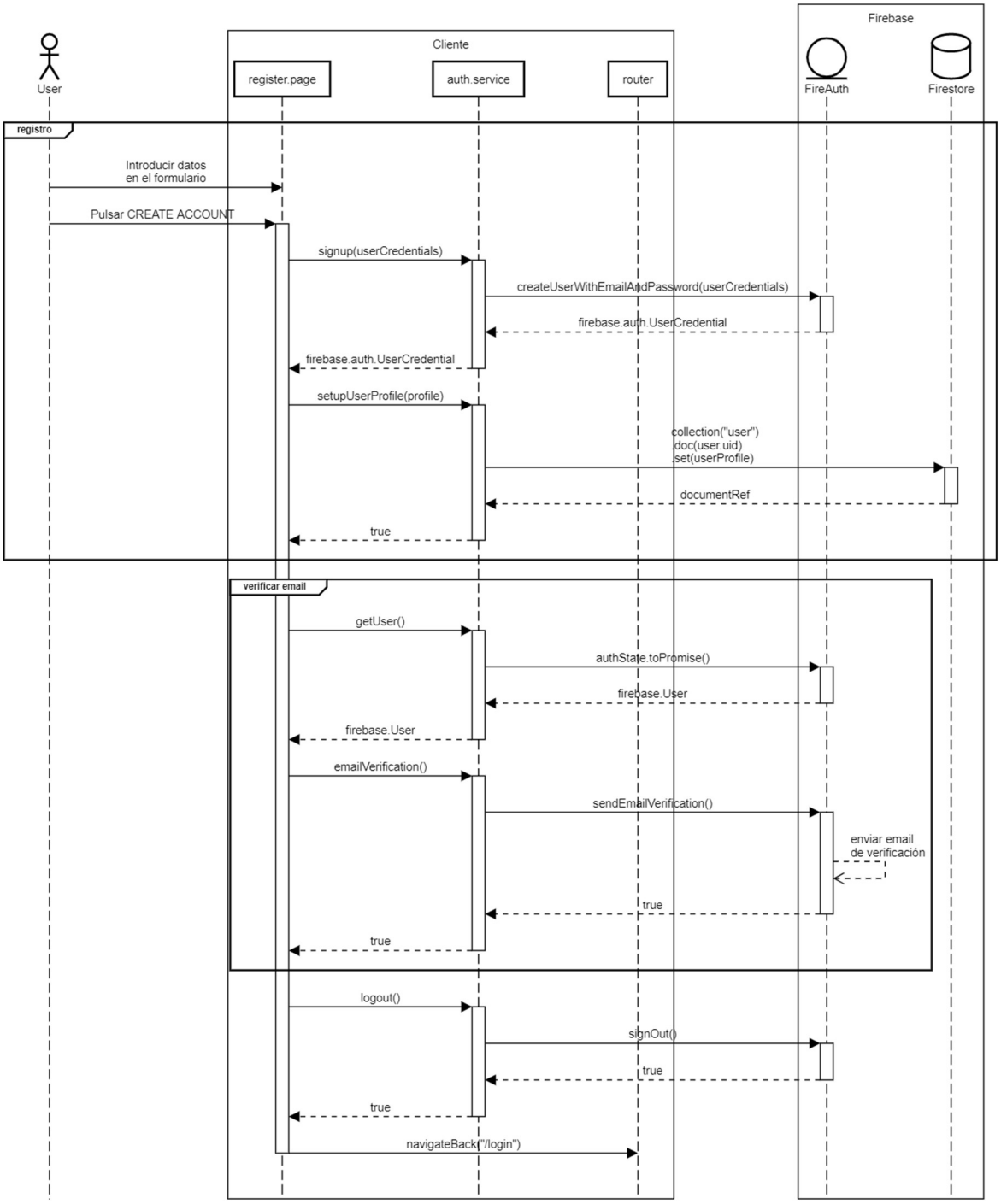


Figura 3.6.1: Diagrama de secuencia de registro de un usuario en la aplicación.

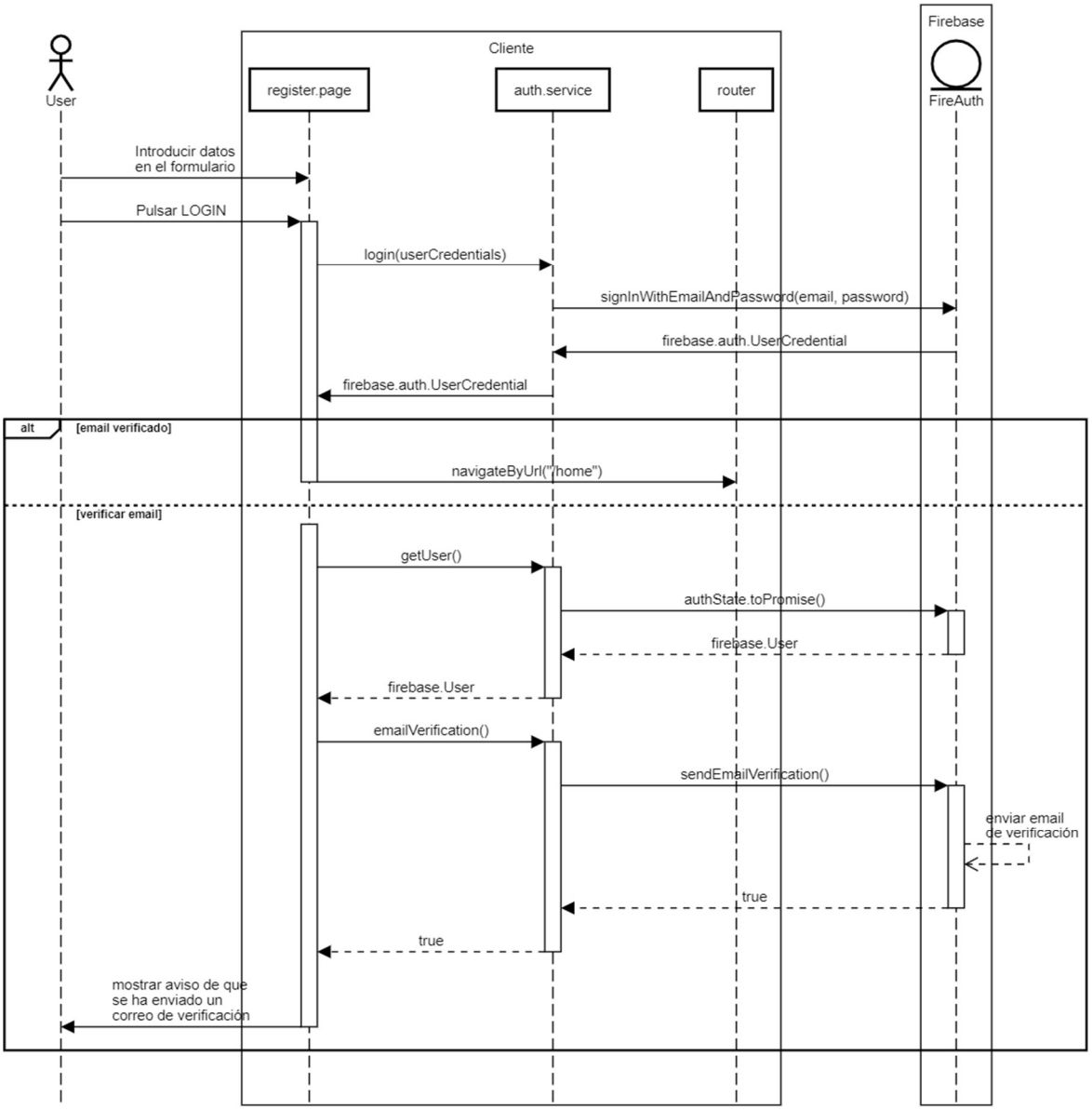


Figura 3.6.2: Diagrama de secuencia de inicio de sesión de un usuario en la aplicación.

En la Figura 3.6.1 se muestran las secuencias de llamadas que se realizan al registrar un usuario en el sistema. Primero se registran el email y la contraseña en Firebase Authentication, luego se inicializa en la base de datos el perfil del usuario y por último se envía un email de verificación a su correo.

El inicio de sesión en la aplicación está reflejado en la Figura 3.6.2. Aquí el orden es sencillo, primero se intenta autenticar en Firebase Authentication y si tiene el

email verificado automáticamente el cliente carga la página principal, si no es así se comprueba si ha enviado el mensaje de verificación a su correo y si no, se vuelve a enviar.

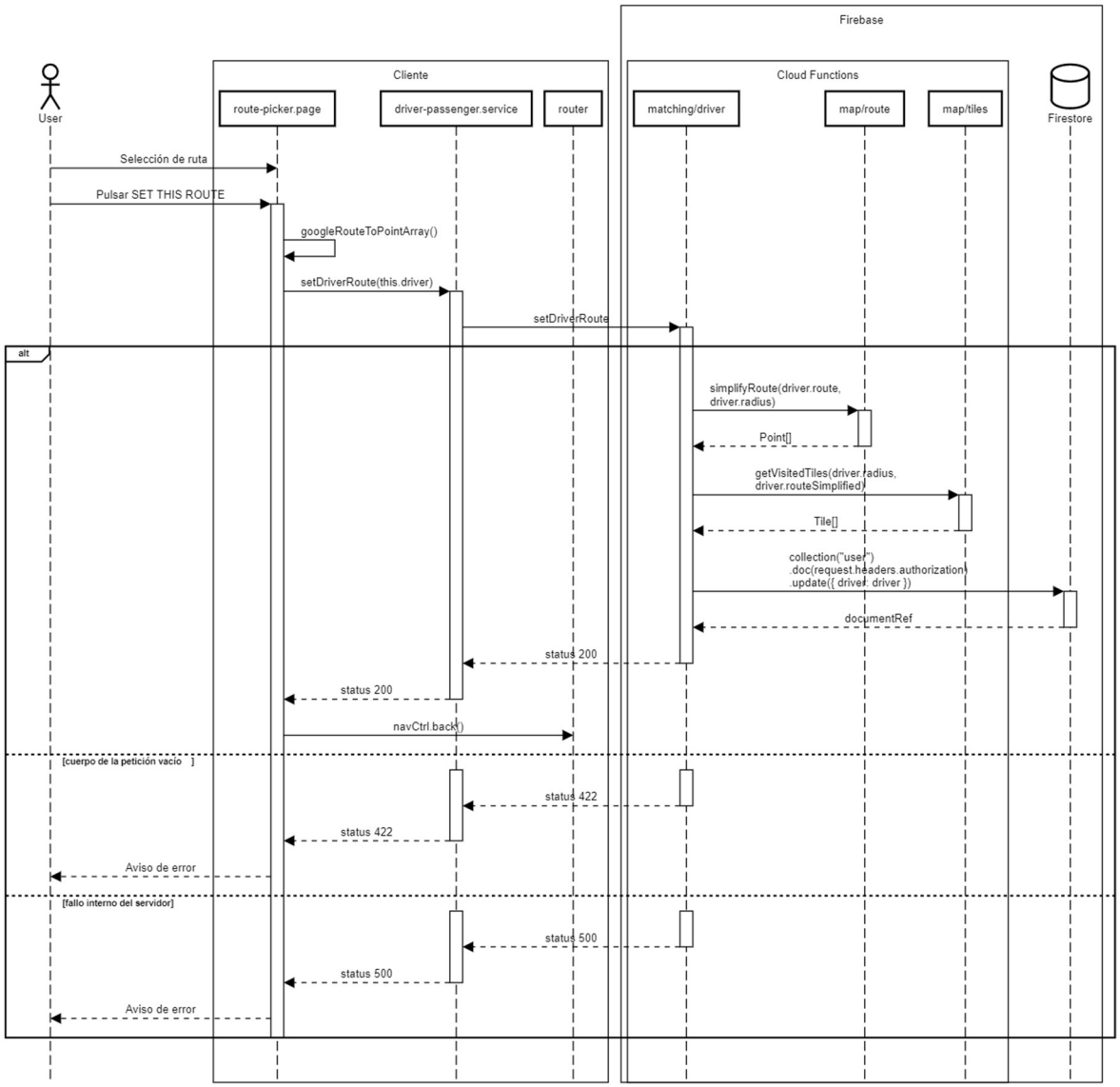


Figura 3.6.3: Diagrama de secuencia de la selección de ruta del conductor.

En la selección de ruta del conductor que aparece en la Figura 3.6.3 primero se transforma la línea de puntos generada por el mapa de Google en un array de puntos, luego se envían al servidor estos datos para que en este se calcule la simplificación de la ruta y el cálculo de las coordenadas de región, que son cálculos más complejos. Tras guardar estos datos calculados en el servidor, este responde al cliente con diferentes tipos de códigos de error dependiendo de cómo haya transcurrido la operación.

- 200. Todo ha ocurrido como se esperaba y no ha habido ningún error.
- 422. Se ha enviado al servidor una petición vacía o que no cumple los requisitos de los parámetros de la función.
- 500. Ha ocurrido algún problema en la función.

Una vez registrada la ruta ya se pueden solicitar las recomendaciones en la página de recomendaciones de pasajeros, como aparece en la Figura 3.6.4. Para ello es necesario los datos de perfil del usuario, luego es tan simple como hacer una petición a la función de Cloud Functions y esperamos que busque los pasajeros que cumplen con las condiciones. Además de las búsquedas de usuarios en la base de datos también se calcula la distancia mínima entre la ruta y los puntos de recogida de cada uno de ellos y, por último, transforman los resultados a objetos Match, con lo que se consigue que, al enviarlos al cliente, el usuario no pueda obtener más información de la que debería de cada usuario.

Por último, en la Figura 3.6.5 se muestra la secuencia de llamadas que se realizan cuando un usuario accede a la página de mensajes de la aplicación. Primero se buscan las salas de chat que tenga abiertas el usuario y se ordenan en base al último mensaje recibido. Luego cuando el usuario accede a uno de los chats se abre un modal en el que se cargan los mensajes del chat y luego se quitan las notificaciones de mensajes no leídos de esa sala de chat.

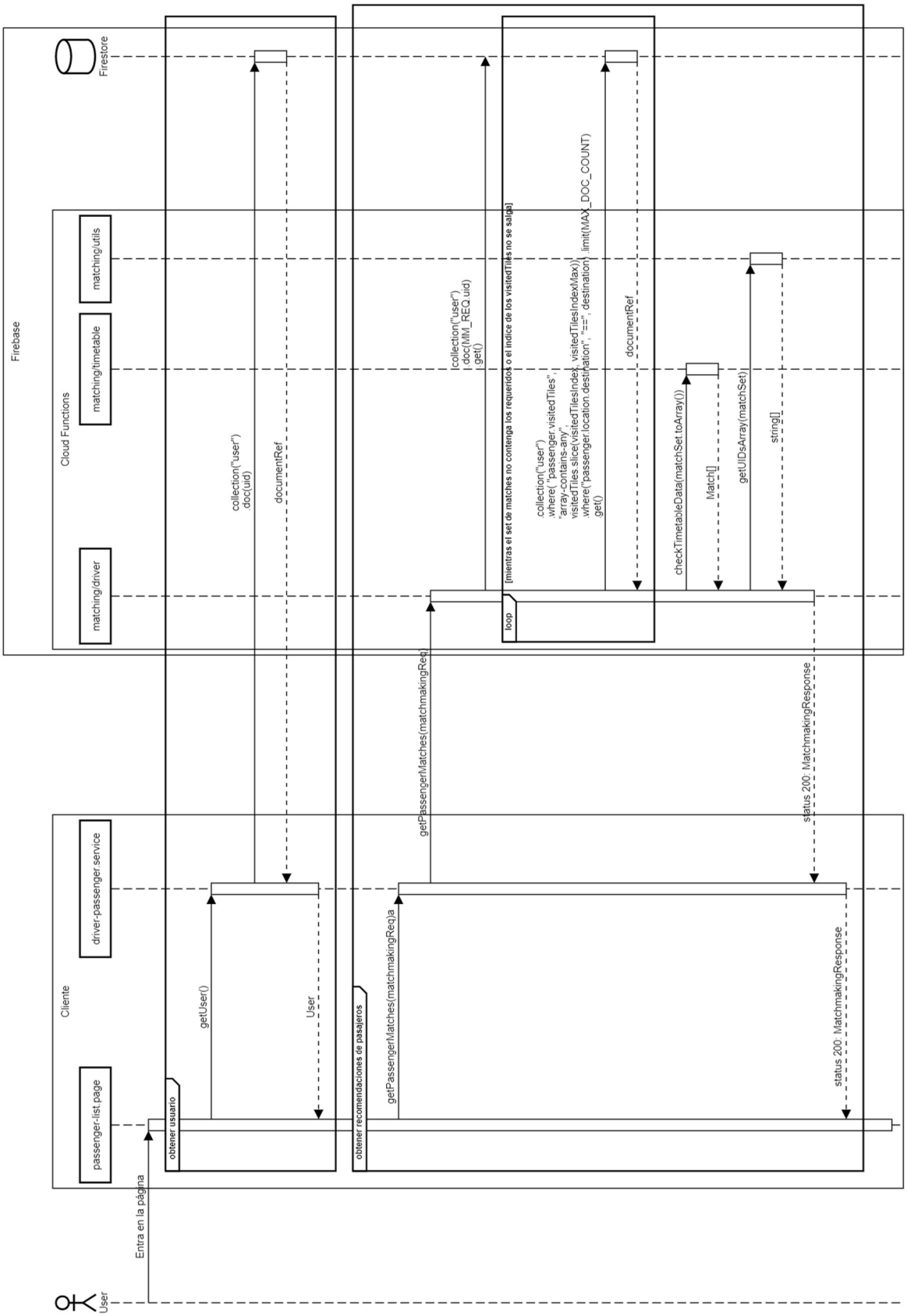


Figura 3.6.4: Diagrama de secuencia de recomendación de pasajeros a un conductor.

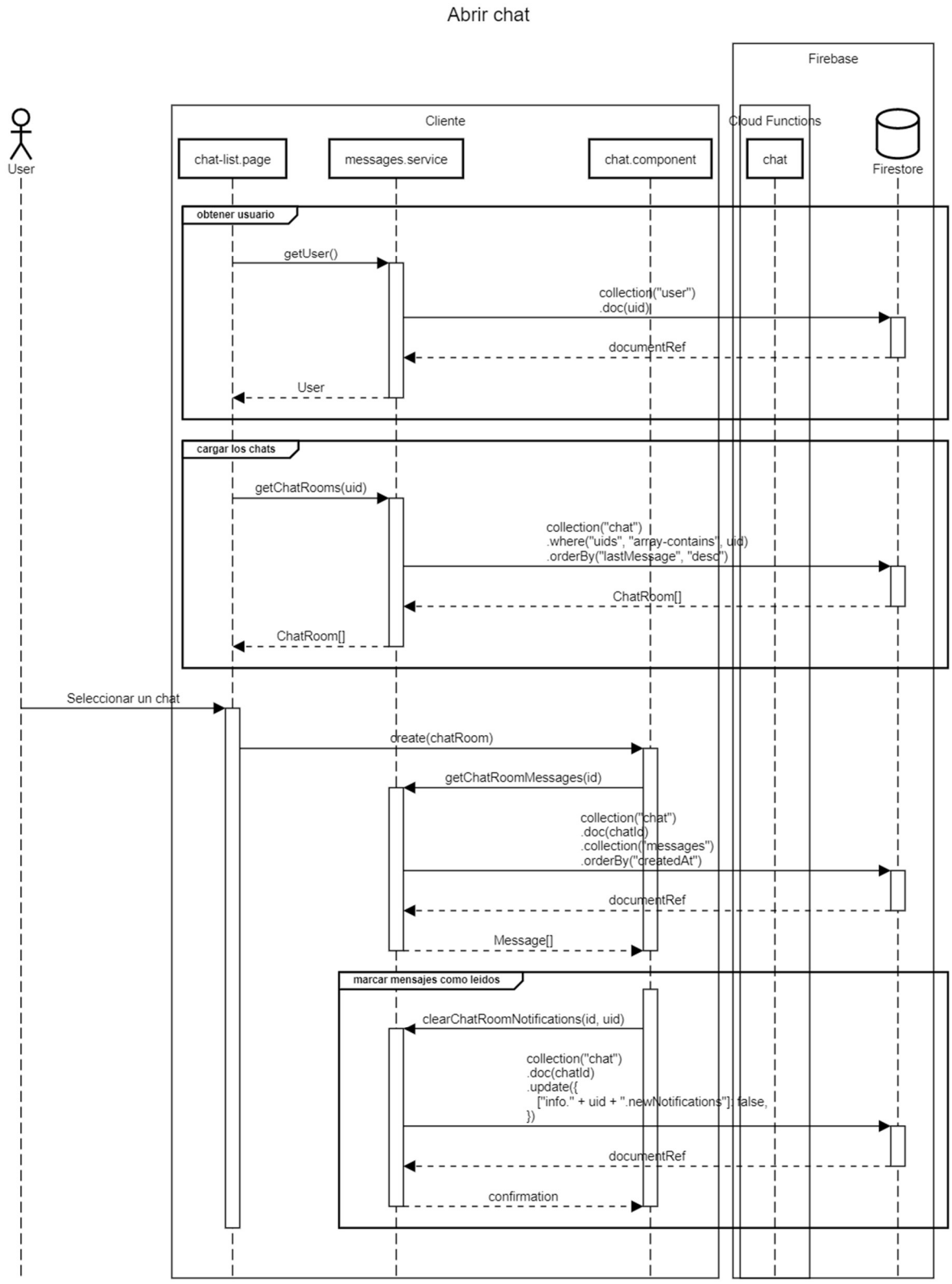


Figura 3.6.5: Diagrama de secuencia de abrir un chat en la aplicación.

4. Desarrollo e implementación

Siguiendo el análisis y modelado del capítulo previo, en este se definen los detalles más relevantes de la implementación del sistema desarrollado.

4.1 Modelo de la base de datos

Teniendo en cuenta lo visto en el Capítulo 2 sobre la estructura de los datos y las limitaciones de Cloud Firestore, la base de datos se ha dividido en tres grandes colecciones: user, chat y timetable.

Puesto que las lecturas y escrituras sobre documentos en Cloud Firestore están limitadas y en planes de pago superior conllevan un coste, suponiendo que un futuro la aplicación necesitará utilizar un plan mejor, se ha optado por limitar el número de documentos creados, comprimiendo todo lo posible la información, aunque en algunos casos no sea lógica o legible la agrupación de los datos.

Por lo general, la documentación de Cloud Firestore sugiere crear los documentos en base a la información que se va a consultar en una página. En la distribución de las páginas que conforman la aplicación se ha tenido en cuenta esta sugerencia de tal forma que en las páginas para los módulos driver y passenger se ha comprimido los datos en un solo documento pertenecientes a la colección user, la colección chat está formada por documentos independientes que contienen información única y duplicada de user y poseen una subcolección de mensajes y por último la colección timetable que contiene en cada documento los turnos que tiene un usuario.

4.1.1 Colección user

Esta colección representada en la Figura 4.1.1 se encarga de guardar la información del perfil, conductor y pasajero del usuario.

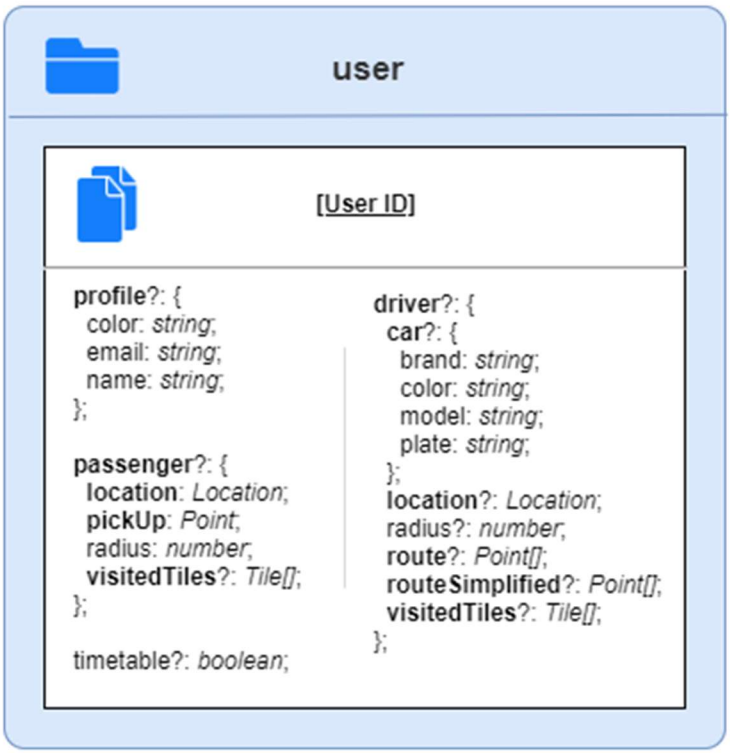


Figura 4.1.1: Diagrama de la colección user de la base de datos.

Los documentos que están contenidos en la colección user se pueden dividir en cuatro partes:

- Profile. Aquí se guarda la Información del perfil de usuario, entre ellos cabe destacar el clave color, que indica el color del perfil que tendrá visible para otros usuarios cuando aparezca en las recomendaciones o en los chats.
- Passenger. En passenger se guarda toda la información referente al punto de recogida del pasajero, donde se incluyen las coordenadas de región de dicho punto.
- Driver. Aquí se almacena toda la información relacionada con la ruta del conductor, ruta completa, simplificada, coordenadas de región a las que pertenecen e información del coche.

- Timetable. Hay una clave booleana que determina si el usuario ha registrado o no los turnos de entrada y salida en la aplicación.

4.1.2 Colección chat

En esta colección, como se puede observar en la Figura 4.1.2, se almacenan las salas de chats con su respectiva subcolección de mensajes.

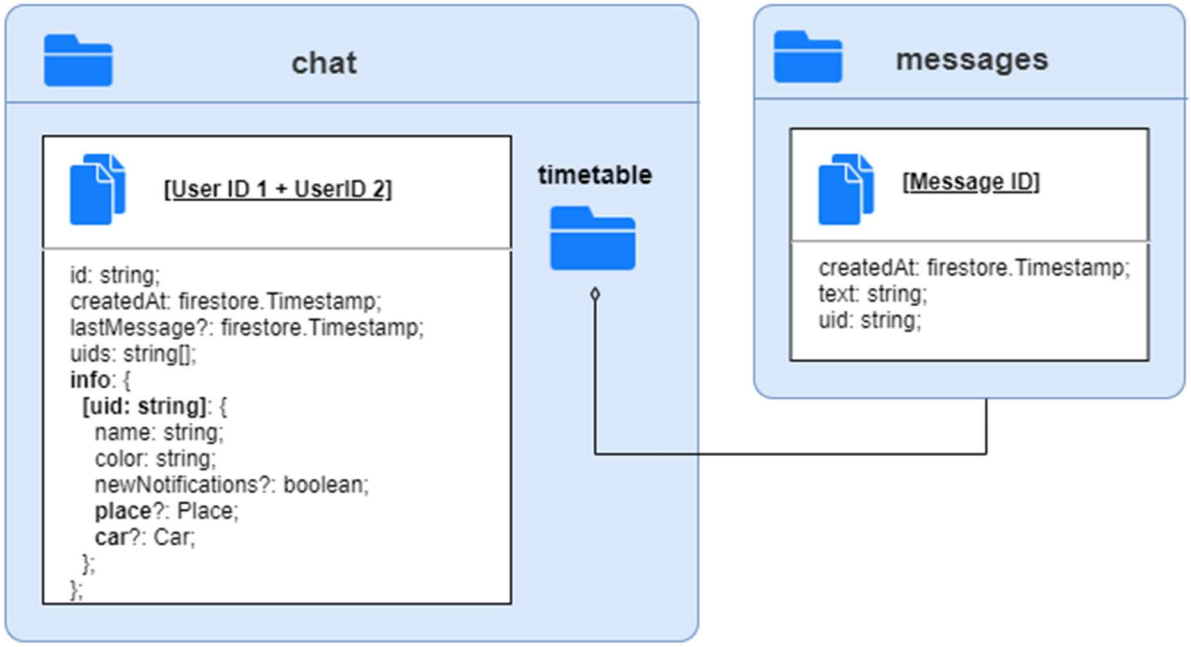


Figura 4.1.2: Diagrama de la colección chat de la base de datos.

Los ids de los documentos pertenecientes a esta colección se forman a partir de la concatenación de los ids de los dos usuarios que entablan la conversación.

Estos documentos contienen información del perfil de cada usuario, duplicada a partir de sus documentos pertenecientes a la colección user. Este es un problema de las bases de datos NoSQL y es que la información puede estar duplicada por cómo se estructuran los datos en ella. Para evitar datos inconsistentes (por ejemplo, que un usuario modifique el color de su perfil en el documento de la colección user y no se actualice en los documentos de la colección chat), se han creado una serie de funciones en el servidor que se

encargan de propagar la información modificada a otros documentos de otras colecciones que contengan la misma información.

Aunque se ha comentado previamente que se ha intentado limitar al máximo el número de documentos, en cuanto a la subcolección messages, por cada mensaje enviado se crea un nuevo documento, esto es así puesto que es posible que en un chat se envíen mensajes al mismo tiempo. Si se tuviera un array de mensajes en un solo documento es posible que hubiera colisiones al intentar añadir dos nuevos mensajes de cada usuario respectivamente a la vez, de esa forma se perdería uno de los mensajes, lo cual no es aceptable.

4.1.3 Colección timetable

En esta colección se almacenan los datos referentes a los horarios del usuario.

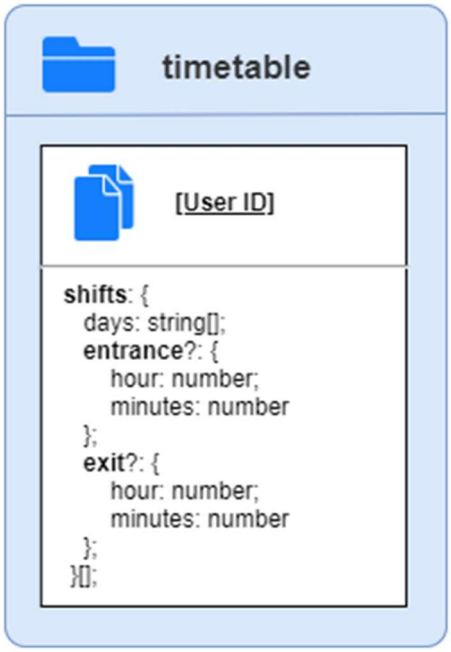


Figura 4.1.3: Diagrama de la colección timetable de la base de datos.

Los documentos de esta colección son los más simples puesto que solo tienen que almacenar un array de pares entrada, salida y los días correspondiente a esas horas, como se puede apreciar en la Figura 4.1.3.

4.2 Emparejamiento de usuarios

A la hora de sugerir las recomendaciones entre usuarios, ya fuese conductores a pasajeros o pasajeros a conductores, se ha utilizado un sistema para identificar de forma única por regiones las coordenadas de los puntos de recogida y de las coordenadas que forman cada ruta.

Antes de obtener estas regiones tenemos que realizar una serie de pasos para convertir las componentes latitud y longitud de una coordenada geográfica en una componente X e Y perteneciente a una coordenada de *región*. Para ello, utilizando las herramientas de la API JavaScript de Google Maps y otras funciones, primero se trasladan los valores de latitud y longitud de la coordenada a una *coordenada mundial* y de esta a una *región*. Ahora explicaremos qué significan cada uno de estos tipos de coordenadas, cómo se han trasladado de unos a otros y los principios en los que se basan.

4.2.1 Tipos de coordenadas

La API JavaScript de Google Maps utiliza los siguientes sistemas de coordenadas (Google, 2020):

- Valores de latitud y longitud, que hacen referencia a un punto del mundo de forma única. (Google usa el estándar WGS84 del Sistema geodésico mundial (World Geodetic System) (Department of Defense, 1984)).
- Coordenadas de píxeles, que hacen referencia a un píxel específico en el mapa a un nivel de zoom específico.
- Coordenadas mundiales, que hacen referencia a un punto en el mapa de manera única.
- Coordenadas de región, que hacen referencia a una región específica en el mapa a un nivel de zoom específico.

Puesto que primero hace falta pasar los valores de latitud y longitud a una coordenada mundial. La API utiliza la proyección de Mercator (Mercator, 1569) para realizar este tipo de traslaciones, ahora veremos en qué consiste.

4.2.2 Proyección cartográfica de Mercator

La API JavaScript de Google Maps utiliza la proyección de Mercator siempre que realiza una traslación de latitud y longitud a una coordenada mundial.

La proyección de Mercator es un tipo de proyección cartográfica que es un sistema de representación gráfica que establece una relación ordenada entre los puntos de la superficie curva de la Tierra y los de una superficie plana, en este caso un mapa. Estos puntos se localizan ayudándose de una red de meridianos y paralelos, en forma de malla.

La proyección de Mercator es una proyección cilíndrica tangente al ecuador. Al ser una proyección cilíndrica como tal, deforma las distancias entre los meridianos en líneas paralelas, aumentando su ancho real, por lo que en zonas de latitud elevada la proyección está distorsionada, lo que impide apreciar las regiones en su verdadera proporción, como por ejemplo los polos.

Este es uno de los problemas que presentan las proyecciones en general ya que según la propiedad que posea una proyección puede distinguirse entre:

- Proyecciones equidistantes, si conserva las distancias.
- Proyecciones equivalentes, si conservan las superficies.
- Proyecciones conformes, si conservan los ángulos.

No es posible tener todas las propiedades anteriores a la vez, por lo que es necesario optar por soluciones de compromiso que dependerán de la utilidad a la que sea destinado el mapa (Wikipedia, 2020).

4.2.3 Coordenadas mundiales

Una coordenada mundial es un valor de punto flotante tomado desde el origen de la proyección del mapa hasta la ubicación de un punto específico. Hay que tener en cuenta que, dado que este valor es un número de punto flotante, puede ser mucho más preciso que la resolución de la imagen del mapa en el que se muestra, en otras palabras, una coordenada mundial es independiente del nivel de zoom.

Las coordenadas mundiales en Google Maps se miden desde el origen de la proyección de Mercator (la esquina noroeste del mapa a 180 grados de longitud y aproximadamente 85 grados de latitud) y aumentan en la dirección X hacia el este y aumentan en la dirección Y hacia el sur. Dado que el tamaño básico de una región de la proyección de Mercator en Google Maps es de 256 x 256 píxeles, el espacio de coordenadas mundial utilizable es {0-256}, {0-256}.

Una proyección de Mercator tiene un ancho finito longitudinalmente, pero una altura infinita latitudinalmente. El mapa que se utiliza es recortado utilizando la proyección de Mercator a aproximadamente ±85 grados para hacer que la forma del mapa resultante sea cuadrada, lo que permite una lógica más sencilla para la selección de regiones como se aprecia en la Figura 4.2.1. Hay que tener en cuenta que una proyección puede producir coordenadas mundiales fuera del espacio de coordenadas utilizable del mapa base, por ejemplo, si es trazada muy cerca de los polos.

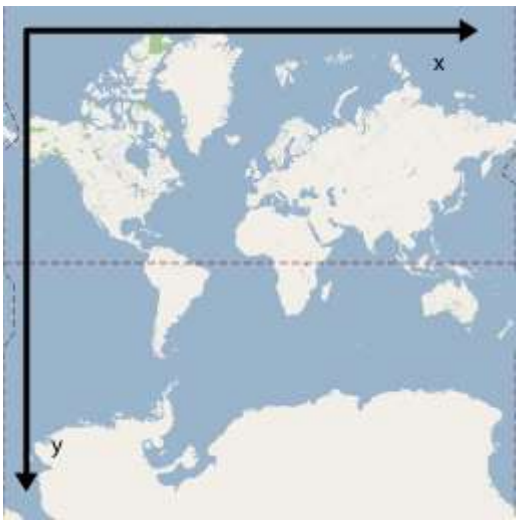


Figura 4.2.1: Mapa recortado utilizando la proyección de Mercator.

Dicho esto, las componentes de una coordenada mundial a partir de los valores de latitud y longitud se obtienen utilizando las siguientes fórmulas (Google, 2020). Para la componente X:

$$WX = 256 \cdot (\frac{1}{2} + \frac{long}{360})$$

Para la componente Y:

$$senY = sen(lat \cdot \frac{\pi}{180})$$

$$WY = 256 \cdot (\frac{1}{2} - ln(\frac{1 + senY}{1 - senY}) \cdot \frac{1}{4\pi})$$

Tras realizar esta serie de cálculos tenemos por fin la coordenada mundial equivalente a los valores de latitud y longitud. Se necesita ahora trasladar estas a coordenadas de región.

4.2.4 Coordenadas de región

La API de Google Maps no puede cargar todas las imágenes del mapa a la vez para los niveles de zoom más altos. En su lugar, la API divide las imágenes en cada nivel de zoom en un conjunto de regiones de mapas, que están organizadas lógicamente en un orden que la aplicación comprende y formadas por dos componentes X e Y. Cuando se realiza un desplazamiento en el mapa o se modifica el nivel de zoom, la API determina qué regiones se necesitan usando coordenadas de píxeles y traduce esos valores en un conjunto de regiones a recuperar. Estas coordenadas de regiones se asignan mediante un esquema que hace que sea lógicamente fácil determinar qué región contiene las imágenes para cualquier punto dado.

Las regiones en Google Maps están numeradas desde el mismo origen que el de los píxeles. Para la implementación de Google de la proyección de Mercator, la región de origen siempre está en la esquina noroeste del mapa, con los valores

de X aumentando de oeste a este y los valores de Y aumentando de norte a sur. Las regiones se indexan utilizando las coordenadas X e Y desde ese origen. Por ejemplo, en la Figura 4.2.2 a un nivel de zoom 2, cuando la tierra se divide en 16 mosaicos, cada mosaico puede ser referenciado por un par (X, Y) único.

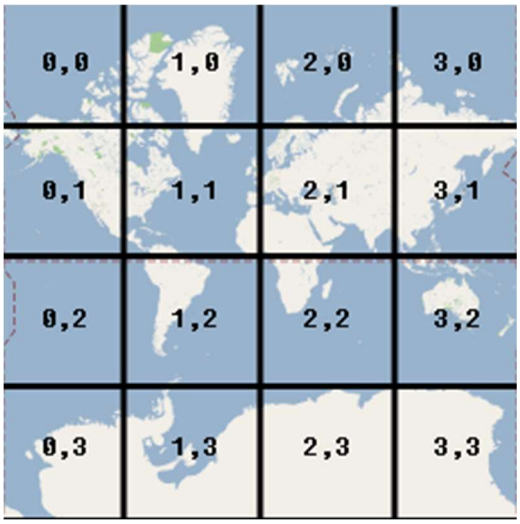


Figura 4.2.2: División del mapa del mundo en coordenadas de región con un nivel de zoom 2.

Hay que tener en cuenta que dividiendo las coordenadas de píxeles por el tamaño del mosaico y tomando las partes enteras del resultado, produce como subproducto la coordenada de la región en el nivel de zoom actual.

Habiendo explicado en qué consiste una coordenada de región, la fórmula necesaria para obtener la coordenada de región que contiene una coordenada mundial a un determinado nivel de zoom es la siguiente:

$$RX = \frac{2^{zoom} \cdot WX}{256}$$

$$RY = \frac{2^{zoom} \cdot WY}{256}$$

En RX, WX/256 normaliza WX entre 0 y 1 (ya que hay 256 regiones y WX estaba entre 0 y 256) y luego se multiplica por la cantidad de regiones definidas por el nivel de zoom. El razonamiento es equivalente para el caso de RY.

Con estas ecuaciones ya podemos identificar de forma única a qué región pertenece un punto en el mapa, y, por tanto, podemos comprobar si un punto de recogida y algún punto perteneciente a una ruta están comprendidos en una misma región.

4.2.5 Punto de recogida del pasajero

En la aplicación un pasajero puede establecer un punto de recogida, dicho punto de recogida está determinado por un par de valores latitud y longitud y por un radio. Dicho radio establece la distancia máxima que el pasajero está dispuesto a desplazarse para ser recogido por un conductor.

Para calcular las regiones por las que pasa dicho punto teniendo en cuenta el radio hay que obtener los puntos noroeste y sureste a la distancia del radio respecto al punto definido por el usuario, por lo que aquí hay que te tener en cuenta dos consideraciones:

1. El nivel de zoom, puesto que dependiendo del nivel de zoom el tamaño de ancho por alto de la coordenada de región varía, por ejemplo, para un valor para el zoom de 16 el tamaño es de 490m x 490m aproximadamente. Esta longitud se puede obtener a partir de la fórmula:

$$longitud = EC \cdot \frac{cos(LAT \cdot \frac{\pi}{180})}{2^{ZOOM}}$$

   Donde EC es la longitud en metros del perímetro de la circunferencia mayor (Wikipedia, 2020) de la tierra (círculo resultante de una sección realizada a una esfera mediante un plano que pase por su centro y la divida en dos hemisferios) que pasa por el ecuador. Esta longitud puede

obtenerse fácilmente conociendo el radio de dicha circunferencia a partir de la ecuación:

$$perimetro = 2\pi \cdot radio$$

Puesto que, si el radio de dicha circunferencia es aproximadamente 6.371 Km, el perímetro es aproximadamente 40.030 Km.

2. La distancia del punto al radio. Un punto siempre pertenecerá a una coordenada de región única, pero al establecer un radio desde el punto seleccionado por el usuario, es posible que los puntos correspondientes al noroeste y sureste pertenezcan a diferentes coordenadas de región.

Para los cálculos de las regiones en la aplicación se ha optado por un nivel de zoom de 16 por varias razones:

1. El tamaño de cada coordenada de región es aproximadamente de medio kilómetro (490m x 490m), con lo que se consigue una precisión aceptable a la hora de emparejar los usuarios, ni demasiado pequeña ni excesivamente grande.

2. El número total de coordenadas a dicho nivel de zoom es de:

$$2^{16} \times 2^{16} = 65.536 \times 65.536 = 4.294.967.296$$

Por lo que las componentes de la coordenada de región serán pares (X, Y) donde la mayor coordenada será (65.535, 65.535) y cada componente no excederá el tamaño máximo de un entero de 32 bits ni la combinación de ambas (pero si de su producto), aunque en Cloud Firestore los enteros tienen tamaño de 64 bits (Google, 2020), al igual que en TypeScript los enteros están contenidos por el tipo básico number (Mozilla, 2020) que tiene formato binario de 64 bits de doble precisión y cumple con el estándar IEEE 754 (Wikipedia, 2020).

Quedando explicadas las consideraciones a tener en cuenta antes de calcular las coordenadas de región perteneciente a un punto de recogida con su radio, solamente necesitamos obtener las coordenadas de región correspondientes a los puntos al noroeste y al sureste respecto al punto de recogida puesto que podemos sacar fácilmente cuales son las coordenadas de región intermedias por lo explicado en la Sección 4.2.3.

Y por lo tanto en la base de datos, cada usuario que haya establecido un punto de recogida tendrá guardado en ella un array de coordenadas de región.

4.2.6 Ruta del conductor

En la aplicación el conductor puede establecer en el mapa la ruta que seguirá con el coche desde el origen que desee con destino a una de las universidades de Málaga y con un radio máximo de desplazamiento que determina cuán lejos estaría el conductor dispuesto a desviarse de su ruta.

Para averiguar la ruta inicial se ha utilizado la API Directions de Google Maps, esta genera la ruta a partir de los puntos de origen y destino, dicha ruta está formada por un array de pares latitud longitud. La cuestión es que hay demasiados pares en este array que nos genera la API puesto que está diseñado para seguir dicha ruta con el coche, por lo que hay que simplificarla.

Lo que se ha hecho ha sido tener en cuenta la longitud de cada coordenada de región (en nuestro caso al elegir un nivel de zoom 16 son aproximadamente 490m, como aparece en la Figura 4.2.3) e ir seleccionando los pares que estén separados entre sí 490m utilizando la función computeDistanceBetween de la librería Geometry de la API JavaScript de Google Maps, dicha función calcula la distancia entre dos pares latitud, longitud en metros. De esta forma reducimos los datos y simplificamos los cálculos calculando las coordenadas de región justas para mapear toda la ruta.

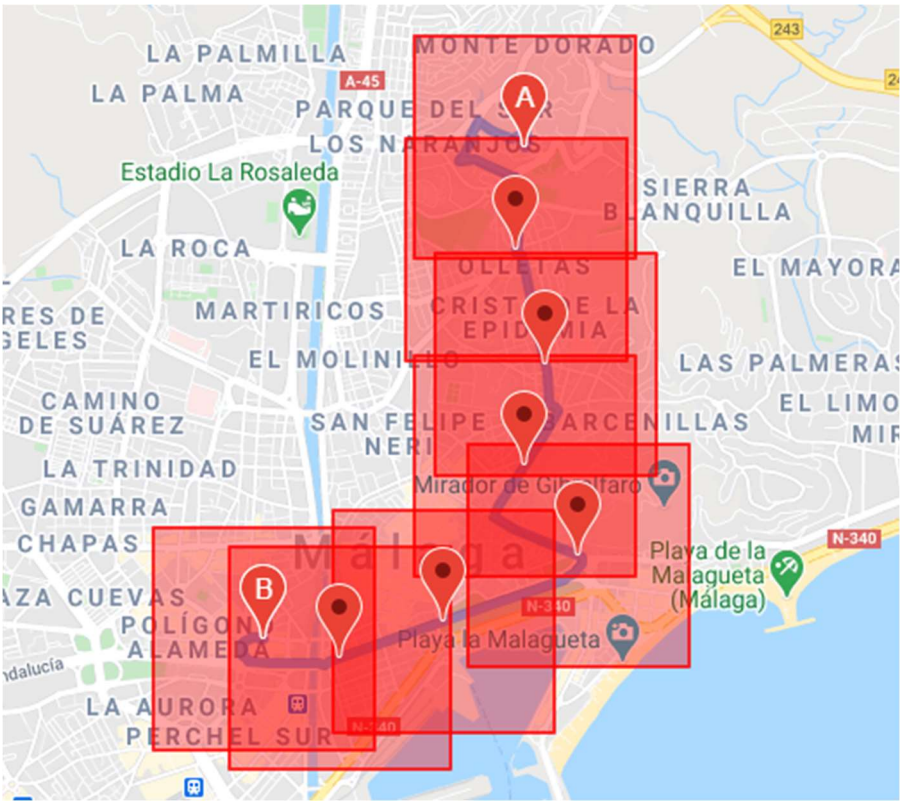


Figura 4.2.3: Representación del cálculo de coordenadas de región de una ruta simplificada.

Y por lo tanto en la base de datos, cada usuario que haya establecido una ruta tendrá guardado en ella un array de coordenadas de región.

4.2.7 Emparejamiento

Tras haber mostrado cómo se van a clasificar cada punto de recogida y cada ruta, ahora explicaremos cómo nos hemos ocupado de recomendar usuarios.

Tanto para las recomendaciones para los pasajeros como para los conductores el algoritmo se ha basado en la rapidez que ofrece Cloud Firestore para obtener los resultados de una consulta (Sharma, 2018).

Como hemos comentado en los anteriores apartados, tras establecer una ruta o un punto de recogida el servidor analiza los valores de latitud y longitud y obtiene las coordenadas de región por las que pasan y acto seguido las almacena en un

array. Para emparejar las coordenadas de un usuario con otro, es tan simple como realizar una consulta sobre los documentos de la colección user utilizando el operador array-contains-any en el que comprobaremos si las coordenadas de región visitadas del usuario coinciden con las coordenadas de región visitadas de los otros usuarios en la colección user (Google, 2020).

Hay que mencionar un problema que conlleva utilizar el operando array-contains-any: no permite utilizar como parámetro un array de más de diez elementos. Por lo que, si el array de coordenadas de región visitados es mayor, en el algoritmo de recomendación se van realizando consultas progresivas tomando los elementos del array de diez en diez.

Otro problema que acarrea realizar tantas consultas progresivas sobre la base de datos es que cada vez que se realiza una consulta se devuelven todos los documentos que coinciden con los parámetros, por lo que hay que limitar la cantidad de documentos leídos. En nuestro caso lo hemos reducido a ocho documentos, puesto que es una cantidad de usuarios que al aparecer en la aplicación del cliente ocupa toda la altura de la pantalla.

Al realizar una consulta utilizando un array de coordenadas de región es posible que que, sin contar el límite, haya más de ocho documentos, por lo que hay que tener en cuenta los documentos ya enviados al cliente, para ello se ha llevado una cuenta de los ids de los documentos devueltos, que se utilizan para no volver a devolverlos en siguientes consultas.

Además de las precauciones tomadas mencionadas previamente, en próximas versiones del sistema, se podrían considerar escenarios menos habituales como la utilización de la referencia de un documento devuelto como parámetro en una consulta para que obtener los documentos que empiecen a partir de dicho documento. En la Figura 4.2.4, se especifica el diagrama de estados que debería de seguir el algoritmo en ese caso.

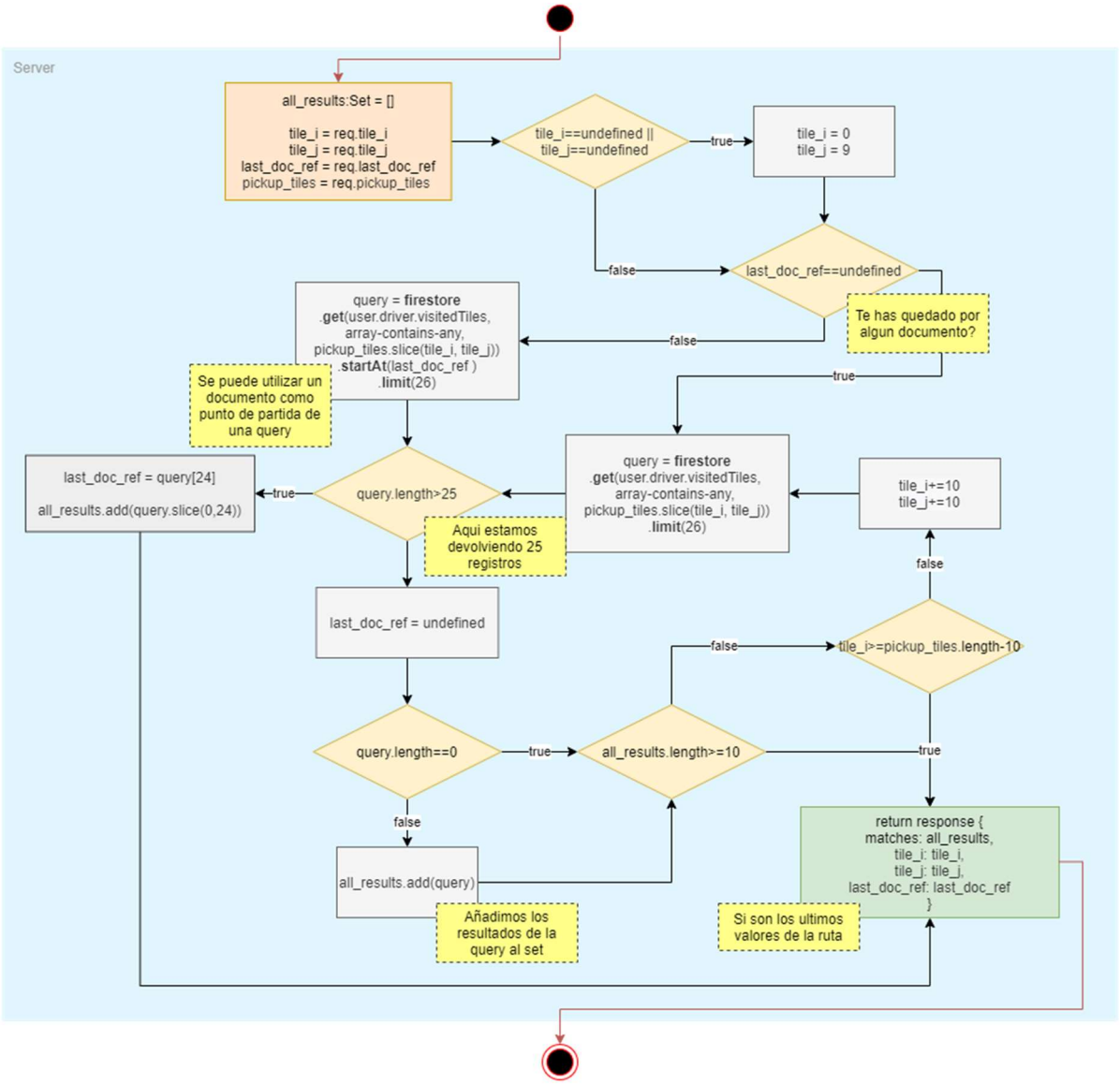


Figura 4.2.4: Diagrama de estados del algoritmo de recomendaciones de pasajeros a conductores en el servidor.

4.3 Mapas

A la hora de utilizar las páginas donde se seleccionan las rutas o los puntos de recogida se ha tenido en cuenta el gasto que supone la carga del mapa, crear la ruta, autocompletar la dirección u obtener la dirección de un punto del mapa.

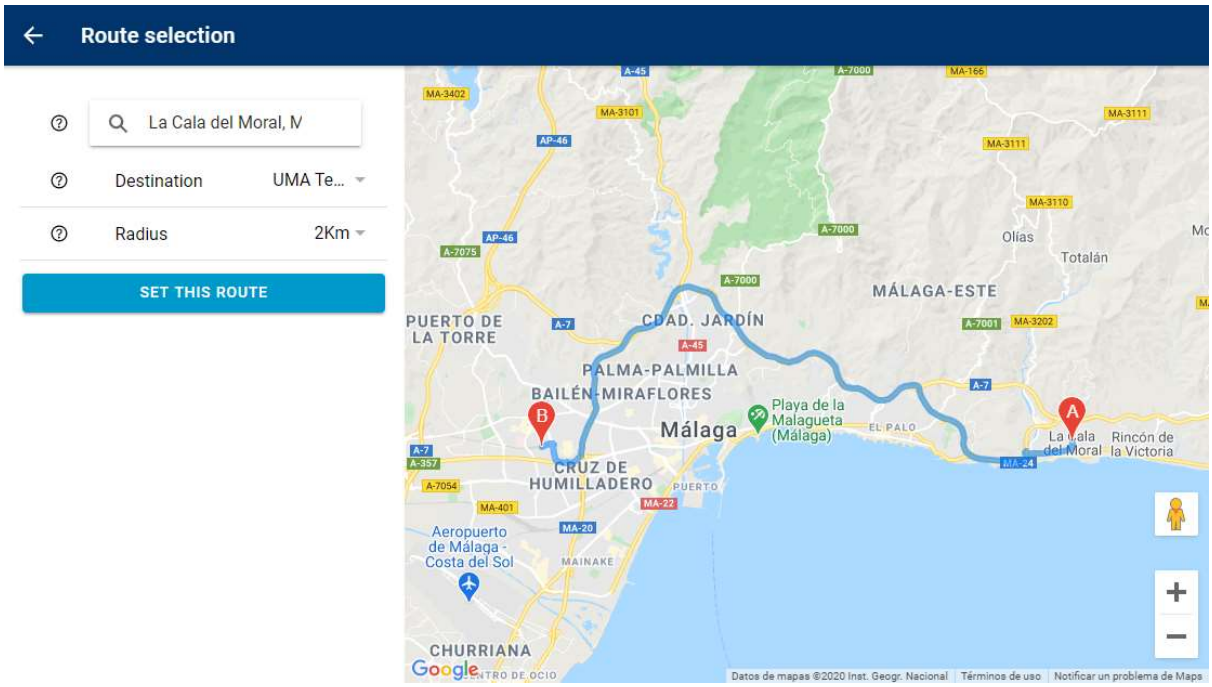


Figura 4.3.1: Página de selección de ruta del conductor.

En la UMA hay aproximadamente 35.500 alumnos matriculados, 1300 empleados y 2500 profesores, por lo que, habría alrededor de 40.000 personas utilizando la aplicación en el peor de los casos (Wikipedia, 2017). Un usuario que quiera utilizar tanto la funcionalidad de recomendación de pasajeros como de conductores requeriría realizar las siguientes peticiones:

- Cargas del mapa, una para la selección de ruta y otra para el punto de recogida, cada carga supone un coste de 0,007 USD.
- Autocompletado de la dirección (0,00283 USD) u obtención de la información del lugar al pulsar en el mapa (0,005 USD) para ambas páginas.
- Cálculo de la ruta entre el punto de origen y la universidad de destino en el caso de selección de ruta (0,01 USD).

Si hacemos los cálculos vemos que el coste aproximado por el uso completo de ambas páginas sería aproximadamente 0,03466 USD por usuario, por lo que si tenemos 40.000 personas el coste total sería de 1.386,4 USD al mes. Para que los costes no superasen la cuota gratuita de 200 USD que ofrece la plataforma al mes, mediante una regla de tres, el número adecuado de usuarios sería de 5.770.

Aun conociendo los gastos que supone utilizar Google Maps Platform el desarrollo es bastante simple ya que proporciona muchas herramientas que facilitan las cosas como el cálculo de distancias, renderizado de mapas, dibujado de figuras, etc. Además de que la instalación es bastante simple ya que solamente hay que añadir la librería de JavaScript al HTML de la página en la que se vaya a implementar.

Para el servidor sin embargo hubo problemas para utilizar esta librería ya que en este no se utiliza ningún HTML desde el que se pueda cargar. Por eso se utilizó la librería spherical-geometry-js (NotWoods, 2018), esta biblioteca proporciona clases y funciones para el cálculo de datos geométricos en la superficie de la Tierra. También incluye un pequeño subconjunto de clases de la versión 3 de la API de JavaScript de Google Maps, para usar como un módulo separado o en NodeJS.

4.4 Interfaz de usuario del cliente

Para el desarrollo del cliente se utilizó Ionic por dos sencillas razones. La primera es su portabilidad, de esta forma desarrollando un solo proyecto tendríamos una aplicación lista para diferentes plataformas y la segunda es que Ionic ya tiene implementados una serie de componentes, plugins y características que permiten, por un lado, crear interfaces muy rápidamente y por otro aprovechar al máximo las capacidades nativas del dispositivo en el que se ejecute la aplicación.

La interfaz gráfica que hemos diseñado buscaba ser simple y eficiente para que al usuario no le costara adaptarse a su uso. Los estilos de los componentes que proporciona Ionic cumplen con estos principios, por lo que se ha minimizado el desarrollo de los estilos, modificando solamente algunos componentes como por ejemplo el estilo de las etiquetas de coche o la organización del selector de colores. De la misma manera la personalización del tema de la aplicación está centralizado en un solo archivo con lo que modificando los valores de este los

cambios se aplican en todas las páginas, esto incluye tanto el tema normal como el tema oscuro. El tipo de tema de la aplicación se ha programado para que se aplique dependiendo del tema del sistema en el que se esté ejecutando como se puede apreciar en la Figura 4.4.1.

Una de las características de las que nos hemos aprovechado en el desarrollo es que dependiendo de la plataforma en la que se ejecute la aplicación, los componentes se adaptan al estilo de cada una. Por ejemplo, si la aplicación se ejecuta en una plataforma como iOS, los estilos de los componentes de la aplicación cambian como se ve en la Figura 4.4.2.

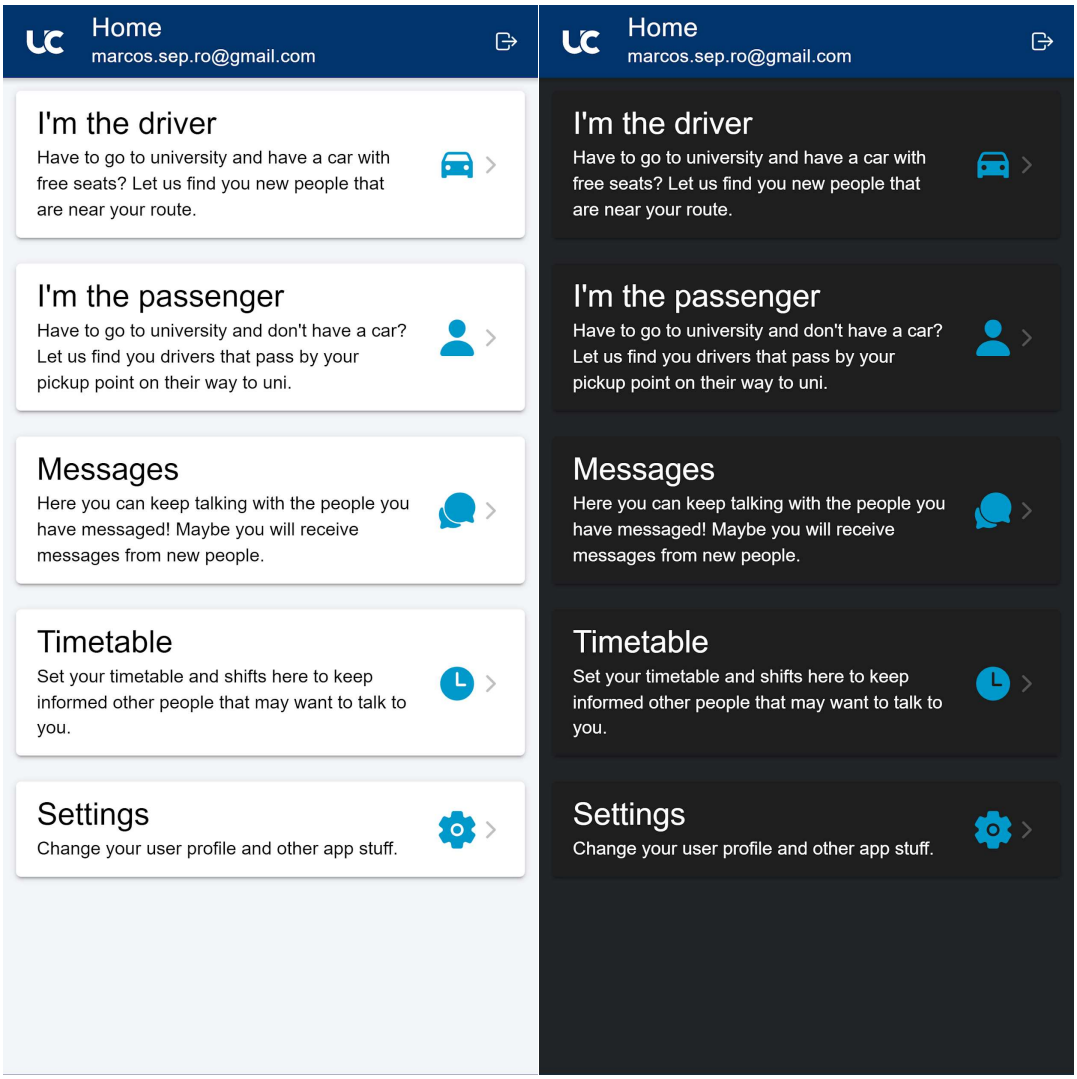


Figura 4.4.1: Capturas de pantalla de la página principal con diferentes temas aplicados.

También buscábamos que la interfaz se adaptase a diferentes tipos de pantallas, esto ha sido fácil de implementar pues Ionic proporciona un componente llamado ion-grid que permite organizar la disposición de los componentes distribuyendolos en filas y columnas. Las columnas se expandirán para llenar la fila y cambiarán de tamaño para adaptarse a columnas adicionales como se puede observar en la Figura 4.4.3. Están basadas en un diseño de 12 columnas con diferentes puntos de interrupción según el tamaño de la pantalla. Estos puntos de interrupción son los que se encargan de adaptar el componente a la pantalla dependiendo del tamaño de esta (Ionic Framework, 2020).

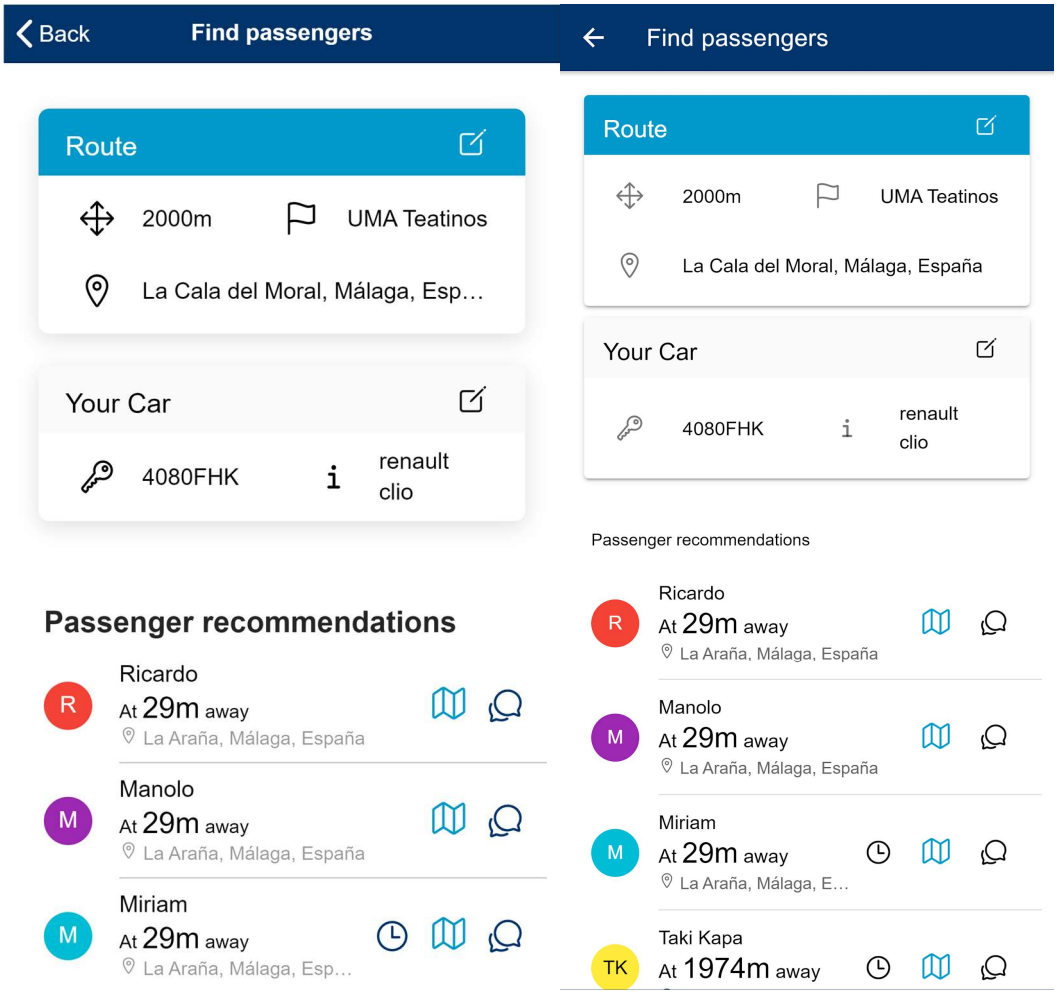


Figura 4.4.2: Comparación de la página de recomendaciones de pasajeros con el tema de iOS y tema Material Design aplicados.

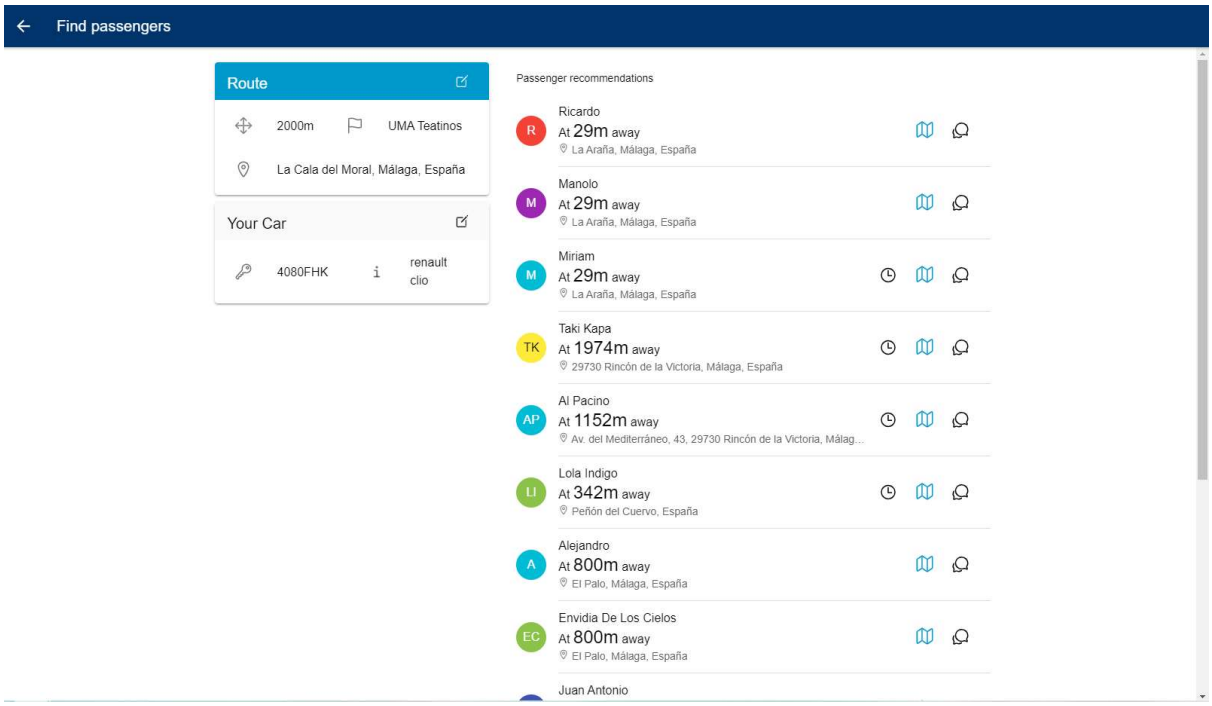


Figura 4.4.3: Captura de pantalla de la página de recomendación de pasajeros adaptada a una pantalla en 1080p.

Para darle un toque más personalizado a la aplicación se ha creado un logo a medida (Figura 4.4.4) utilizando Adobe Illustrator. La idea que se ha intentado plasmar en él es la de combinar dos círculos que representan las ruedas de un vehículo.

Figura 4.4.4: Logo de UniCar.

Por último, mencionar que el esquema de colores utilizados en el tema de la aplicación y el logo está sacado de los colores principales del campus virtual los cuales son en hexadecimal:

- Color primario #00346b.
- Color secundario #0099cc.
- Color terciario #275484.

4.5 Conexión del cliente con el servidor

Para la comunicación del cliente con el servidor se han utilizado las funciones que ofrece el SDK de Firebase para JavaScript, con ellas se pueden realizar consultas en la base de datos o manejar la identificación del usuario entre otros, y también se han creado algunas funciones en Cloud Functions como si fuera una API REST.

La instalación del SDK es muy sencilla y solo requiere seguir el tutorial que ofrece Firebase en su página web (Google, 2020).

Las funciones del SDK se ejecutan en el cliente, por lo que aquí es donde las reglas de seguridad de la base de datos entran en juego (Figura 4.5.1). Estas administran el acceso a los recursos en la base de datos y aseguran que no se realicen operaciones no permitidas, como por ejemplo que un usuario acceda a una sala de chat en la que no esté o crear colecciones y documentos no permitidos.

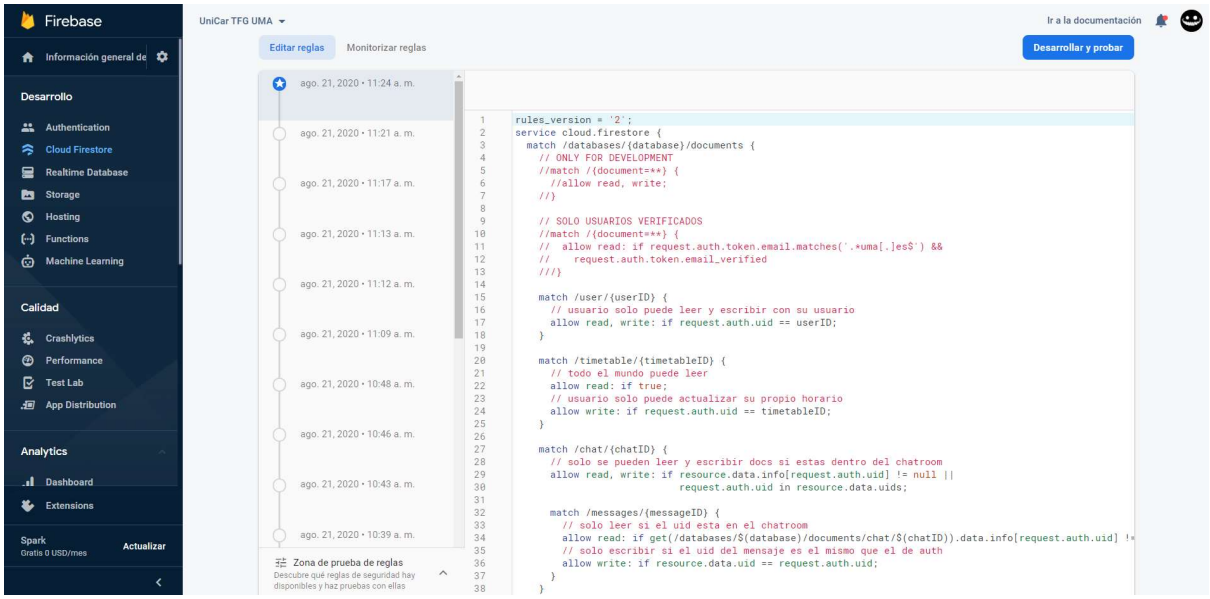


Figura 4.5.1: Captura de pantalla de la configuración de reglas de Cloud Firestore.

Las funciones implementadas como una API REST han sido principalmente las de obtención de recomendaciones, de esta forma enviando una petición con la estructura determinada de los datos de entrada, el servidor hace todos los cálculos pesados (búsqueda de usuarios, conversión a objetos Match, cálculo de la distancia mínima entre los puntos de recogida y rutas, etc.) y devuelve los resultados sin que el cliente pueda intervenir en todo el proceso evitando comportamientos extraños no deseados. Lo mismo ocurre con el envío de mensajes, ya que en cada documento de un mensaje se almacena la información de cuándo se envió, que se tiene en cuenta para el orden en el que se muestran en la conversación. Por eso el registro de la hora se hace en el servidor sin que el cliente pueda modificar de forma fraudulenta este valor.

Estas funciones están implementadas mayormente en los servicios de la aplicación pues de esta forma dan “servicio” a los componentes que soliciten estas funcionalidades. Para ello se ha hecho uso del paradigma de programación reactiva utilizando la librería RxJS (ReactiveX, 2020).

La programación reactiva es un paradigma de programación asíncrona que se ocupa de los flujos de datos y la propagación de cambios. RxJS (Reactive Extensions for JavaScript) es una biblioteca para programación reactiva que utiliza objetos observables lo que facilita la composición de código asincrónico o basado en funciones callback.

RxJS combina el patrón Observer con el patrón Iterator y la programación funcional con colecciones para satisfacer la necesidad de gestionar secuencias de eventos. Los conceptos esenciales de RxJS que se encargan de la gestión de eventos asíncronos y los que se han utilizado en esta aplicación son:

- Observable. Representa la idea de una colección invocable de valores o eventos futuros.
- Observer. Es una colección de funciones callbacks que conoce cómo escuchar los valores entregados por el Observable.

- Suscription. Representa la ejecución de un Observable. También útil para cancelar la ejecución.

4.6 Servidor

Al haber utilizado Firebase se ha simplificado en gran parte el desarrollo de la parte del servidor pues la infraestructura ya está desarrollada, por lo que solamente se ha tenido que tener en cuenta el aprendizaje de las herramientas que proporciona Firebase como la gestión de usuarios, el uso de la base de datos (consultas, creación, eliminación de colecciones y documentos y las reglas de seguridad) y el uso de las funciones de Cloud Functions.

4.4.1 Cloud Functions

Aquí cabe destacar lo ya comentado en la Sección 4.2 sobre el emparejamiento de usuarios, que se han diseñado una serie de algoritmos que permiten las recomendaciones de usuarios en base a: o bien la ruta del conductor o bien el punto de recogida. Estas funciones han sido realizadas como una API REST (Wikipedia, 2020) donde se ha establecido que la función se active cuando reciba una petición por parte del cliente.

Los datos en las bases de datos NoSQL no requieren de estructuras fijas para almacenarlos, como tablas, por lo que es posible que exista información repetida. Por esta razón se han de crear mecanismos que mantengan actualizados los datos para tener así una base de datos consistente. Se han desarrollado una serie de funciones de propagación de la información actualizada en la base de datos, de esta forma se mantiene la cohesión de los datos en esta. Estas funciones han tenido que ser implementadas porque, por ejemplo, en los documentos de la colección chat se almacenan datos copiados directamente de los documentos de la colección user.

5. Conclusiones y líneas futuras

5.1 Conclusiones

Al final del desarrollo hemos obtenido una aplicación que cumple los objetivos de partida y será útil a los estudiantes, profesores y personal de la Universidad de Málaga. Una aplicación de aspecto profesional, totalmente funcional, que cubre todos los aspectos de su funcionalidad principal pero también otros secundarios (como el sistema de mensajería).

Durante el desarrollo he aprendido nuevas tecnologías punteras como Angular, Ionic, Typescript, NodeJS o Firebase, todas ellas se utilizan hoy en día y se piden en muchos perfiles de trabajo como por ejemplo desarrollador FullStack. Además, me he peleado con todas estas para hacerlas funcionar correctamente utilizando solamente la documentación proporcionada por los desarrolladores y videos de plataformas como YouTube. También se han desarrollado algoritmos eficientes y funcionales, como los algoritmos de emparejamiento de usuarios.

Se han tenido en cuenta otro tipo de cuestiones que durante el grado no se tuvieron en cuenta como el estético o cuestiones económicas relacionadas con las tecnologías utilizadas.

También he cumplido un objetivo personal que era el desarrollo de esta idea, pues yo, al igual que otros alumnos de la universidad, no tengo muchas facilidades para utilizar el transporte público para llegar a la universidad y no conozco a mucha gente que tenga vehículo propio. Cabe destacar el dominio

adquirido en las tecnologías utilizadas que fueron elegidas por mí mismo como reto personal como preparación para el mundo laboral.

Como conclusión diré que el aprendizaje desde cero de una tecnología puede llegar a ser muy difícil para una sola persona, pero con tiempo y dedicación se pueden lograr buenos resultados, como este proyecto.

5.2 Líneas futuras

En cuanto a posibles continuaciones del mismo, tenemos bastante las siguientes características a aplicar:

- Por las limitaciones y costes que supone Google Maps Platform se utilizará OpenMaps como sustituto en futuras actualizaciones de la aplicación.
- Las funciones desplegadas en el servidor se ejecutan bajo NodeJS versión 8. Cloud Functions ofrece tanto la versión 8 como la 12, pero esta última está solamente disponible para planes de pago superior y hace poco han anunciado el fin del soporte de la versión 8, por lo que en un futuro habrá que contratar un plan superior de Firebase para que funcionen.
- La traducción de la aplicación al español también se aplicará en una futura actualización al igual que otro idioma que se viera necesario integrar.
- La aplicación podría ser utilizada para cualquier universidad del mundo, por lo que se tendrá en cuenta el registro de diferentes usuarios de otras universidades a la de Málaga.
- Aplicar una batería de pruebas de forma intensiva de la aplicación para comprobar que está preparada para un uso estable.
- Exportar la aplicación como una app de iOS y Android para colgarlas en las respectivas páginas de la tienda.

Bibliografía

- *Ahead-of-time compilation*. (2020, 4 17). Ahead-of-time compilation. https://en.wikipedia.org/wiki/Ahead-of-time_compilation
- APD. (2019). *En qué consiste la metodología Kanban y cómo utilizarla*. ¿En qué consiste la metodología Kanban y cómo utilizarla? | APD. https://www.apd.es/metodologia-kanban/
- Atlassian. (2011). *Trello*. Trello. https://trello.com/
- Barot, S. (2019, 6 19). *Why Developers Love to Use TypeScript in 2020?* Why Developers Love to Use TypeScript in 2020? | Aglowid IT Solutions. https://aglowiditsolutions.com/blog/why-use-typescript/#:~:text=Popularity%20of%20TypeScript,is%20prone%20to%20fewer%20errors.
- Bruck, B. P., Incerti, V., Iori, M., & Vignoli, M. (2017). Minimizing CO2 emissions in a practical daily carpooling problem. *Computers & Operations Research*, *81*, 40-50.
- Department of Defense. (1984). *Department of Defense World Geodetic System 1984, Its Definition and Relationships With Local Geodetic Systems*. NGA: DoD World Geodetic System 1984. https://earth-info.nga.mil/GandG/publications/tr8350.2/tr8350_2.html
- diagrams.net. (2005). *diagrams.net*. Diagram Software and Flowchart Maker. https://www.diagrams.net/
- Drifty Company. (2013). *Ionic - Cross-Platform Mobile App Development*. Ionic - Cross-Platform Mobile App Development. https://ionicframework.com/

- explooosion. (2020). *Agm-Direction*. GitHub - explooosion/Agm-Direction: This is the directive for @agm/core (not official). https://github.com/explooosion/agm-direction
- Facebook, Inc. (2013). *React*. React – Una biblioteca de JavaScript para construir interfaces de usuario. https://es.reactjs.org/
- Gallo, M., & Buonocore, C. (2017). The Inclination of University Students Towards Carpooling: Critical Aspects and Opportunities. *International Journal of Education and Learning Systems*, *2*, 407-412. https://www.iaras.org/iaras/home/caijels/the-inclination-of-university-students-towards-carpooling-critical-aspects-and-opportunities
- *Git*. (2005). Git. https://git-scm.com/
- *GitHub*. (2008). GitHub: The Largest and Most Advanced Open Source Development Platform - Secure Public & Private Repositories - Code Review - Codespaces - Actions - CI CD - Project Management - DevOps · GitHub. https://github.com/
- Google. (2005). *Google Maps Platform*. APIs de geolocalización | Google Maps Platform | Google Cloud. https://cloud.google.com/maps-platform?hl=es
- Google. (2006). *Documentos de Google*. Documentos de Google: crea y edita documentos online de forma gratuita. https://www.google.es/intl/es/docs/about/
- Google. (2014). *Firebase*. Firebase - Google. https://firebase.google.com/
- Google. (2016). *Angular*. Angular. https://angular.io/

- Google. (2020). *Agrega Firebase a tu proyecto de JavaScript*. Agrega Firebase a tu proyecto de JavaScript. https://firebase.google.com/docs/web/setup
- Google. (2020). *Planes y precios*. Planes y precios | Google Maps Platform | Google Cloud. https://cloud.google.com/maps-platform/pricing/?hl=es
- Google. (2020, 6 19). *Map and Tile Coordinates*. Map and Tile Coordinates | Maps JavaScript API | Google Developers. https://developers.google.com/maps/documentation/javascript/coordinates
- Google. (2020, 6 22). *Supported data types*. Supported data types | Firebase. https://firebase.google.com/docs/firestore/manage-data/data-types
- Google. (2020, 6 30). *Supported data types*. Supported data types | Firebase. https://cloud.google.com/firestore/docs/concepts/data-types
- Google. (2020, 7 27). *Get to know Cloud Firestore*. Get to know Cloud Firestore | YouTube. https://www.youtube.com/playlist?list=PLl-K7zZEsYLluG5MCVEzXAQ7ACZBCuZgZ
- Google. (2020, 8 12). *Usage and limits*. Usage and limits | Firebase. https://firebase.google.com/docs/firestore/quotas
- Google. (2020, 9 1). *Cloud Functions for Firebase*. Cloud Functions for Firebase. https://firebase.google.com/docs/functions
- Google. (2020, 9 10). *Showing Pixel and Tile Coordinates*. Showing Pixel and Tile Coordinates | Maps JavaScript API. https://developers.google.com/maps/documentation/javascript/examples/map-coordinates
- Google. (2020, 9 22). *Cloud Firestore Data model*. Cloud Firestore Data model | Firebase. https://firebase.google.com/docs/firestore/data-model

- Google. (2020, 9 23). *Perform simple and compound queries in Cloud Firestore*. Perform simple and compound queries in Cloud Firestore | Firestore. https://firebase.google.com/docs/firestore/query-data/queries
- Goswami, B., & Chatterjee, R. (2019). *Progressive Web Apps for Social Development*. Independently published.
- Ionic Framework. (2020). *ion-grid*. ion-grid - Ionic Documentation. https://ionicframework.com/docs/api/grid
- Mercator, G. (1569). *Mercator projection*. Mercator projection - Wikipedia. https://en.wikipedia.org/wiki/Mercator_projection
- Microsoft. (2015). *Visual Studio Code*. Visual Studio Code - Code Editing. Redefined. https://code.visualstudio.com/
- Microsoft. (2018). *TypeScript*. TypeScript: Typed JavaScript at Any Scale. https://www.typescriptlang.org/
- Mozilla. (2020, 7 15). *Number*. Number - JavaScript | MDN. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number
- Netscape Communications & Fundación Mozilla. (1995). *JavaScript*. JavaScript | MDN. https://developer.mozilla.org/es/docs/Web/JavaScript
- NotWoods. (2018). *Spherical Geometry Library*. spherical-geometry-js - npm. https://www.npmjs.com/package/spherical-geometry-js
- *OAuth 2.0*. (2006). OAuth 2.0 — OAuth. https://oauth.net/2/
- ReactiveX. (2020). *RxJS*. RxJS Introduction. https://rxjs-dev.firebaseapp.com/guide/overview

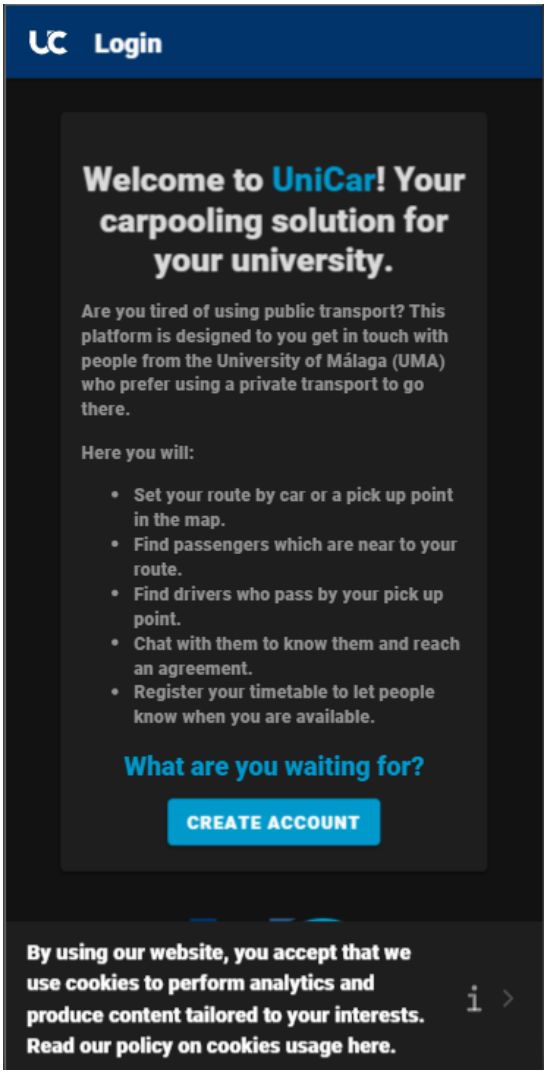
- scttcper. (2020). *Angular Color*. GitHub - scttcper/ngx-color: 🎨 Color Pickers from Sketch, Photoshop, Chrome, Github, Twitter & more. https://github.com/scttcper/ngx-color
- SebastianM. (2020). *AGM*. GitHub - SebastianM/angular-google-maps: Angular 2+ Google Maps Components. https://github.com/SebastianM/angular-google-maps
- Sharma, A. (2018, 9 29). *Realtime Database vs. Cloud Firestore — Which Database is Suitable for your Mobile App*. Realtime Database vs. Cloud Firestore — Which Database is Suitable for your Mobile App | by Ashish Sharma | Data Driven Investor | Medium. https://medium.com/datadriveninvestor/realtime-database-vs-cloud-firestore-which-database-is-suitable-for-your-mobile-app-87e11b56f50f
- Strozzi.it. (2007). *NoSQL*. NoSQL Relational Database Management System: Home Page. http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/nosql/Home%20Page
- *Vue.js*. (2014). Vue.js. https://vuejs.org/
- W3C. (1992). *HTML*. HTML. https://es.wikipedia.org/wiki/HTML
- W3C. (1996). *CSS*. Hojas de estilos en cascada. https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada
- Wikipedia. (2017). *UMA Datos generales*. Universidad de Málaga - Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Universidad_de_M%C3%A1laga
- Wikipedia. (2020). *Proyección cartográfica*. Proyección cartográfica - Wikipedia, la enciclopedia libre.

https://es.wikipedia.org/wiki/Proyecci%C3%B3n_cartogr%C3%A1fica#cite_note-2

- Wikipedia. (2020, 9 8). *Representational state transfer*. Representational state transfer | Wikipedia. https://en.wikipedia.org/wiki/Representational_state_transfer
- Wikipedia. (2020, 9 9). *Gran círculo*. Gran círculo - Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Gran_c%C3%ADrculo#Geograf%C3%ADa_y_cartograf%C3%ADa
- Wikipedia. (2020, 9 13). *IEEE 754*. IEEE 754 - Wikipedia. https://en.wikipedia.org/wiki/IEEE_754

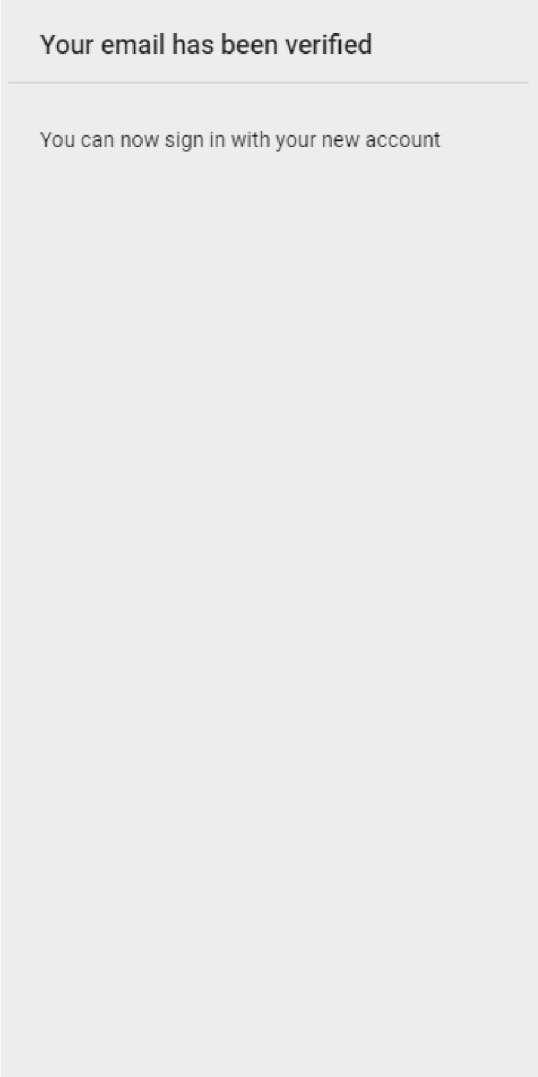
Apéndice A:
Manual de usuario

Crear una cuenta nueva

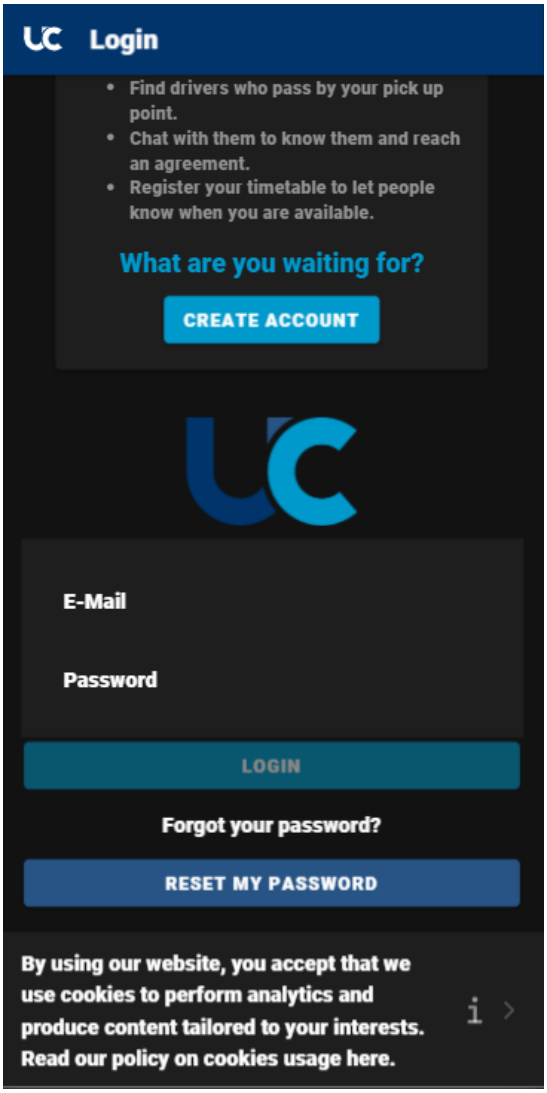


1. Para utilizar la aplicación UniCar necesitas una cuenta. Para ello debes pulsar en el botón CREATE ACCOUNT.

2. Rellena el email, la contraseña, tu nombre y un color para el fondo de tu perfil. Después pulsa CREATE ACCOUNT para crear la cuenta.

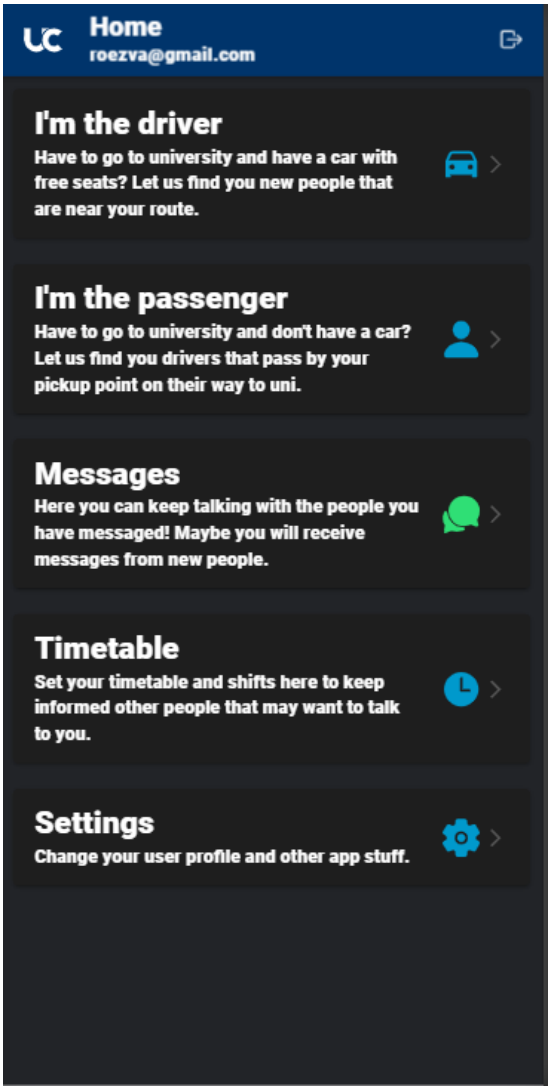


3. Se te enviará un correo electrónico para confirmar tu cuenta de correo. Puedes comprobar la carpeta de spam si ves que no llega el correo de confirmación.

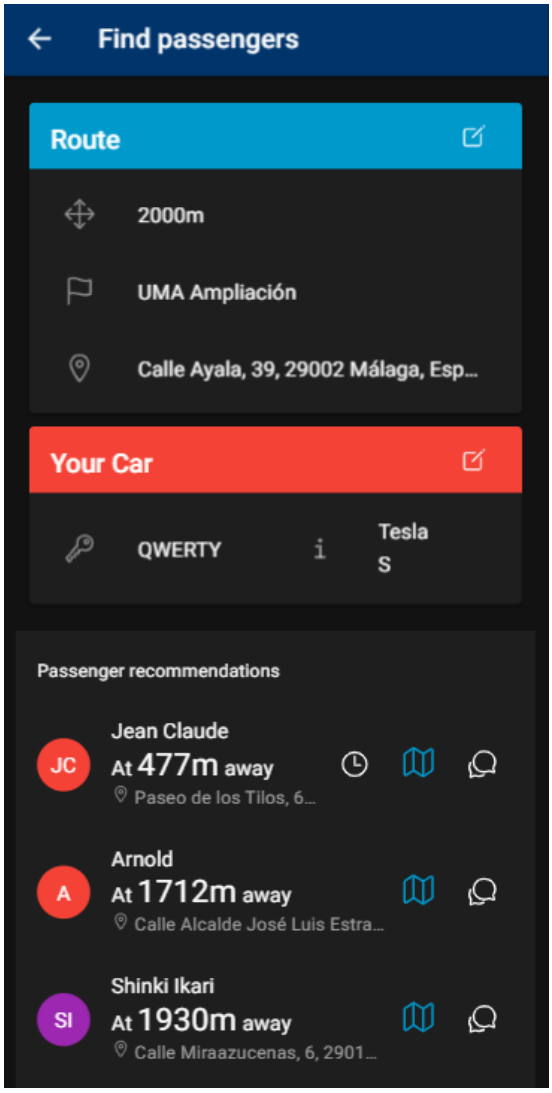


4. Una vez que hayas confirmado tu cuenta, podrás acceder a la aplicación desde la pantalla de inicio de sesión introduciendo tu email y contraseña y pulsando el botón LOGIN.

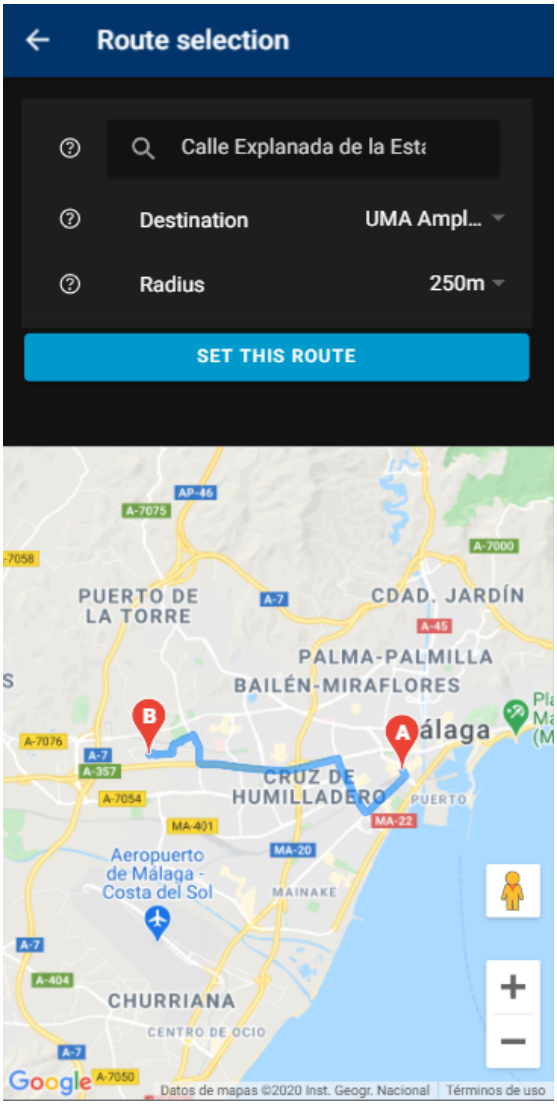
Obtener recomendaciones de pasajeros



1. Desde la pantalla principal, pulsa I'M A DRIVER

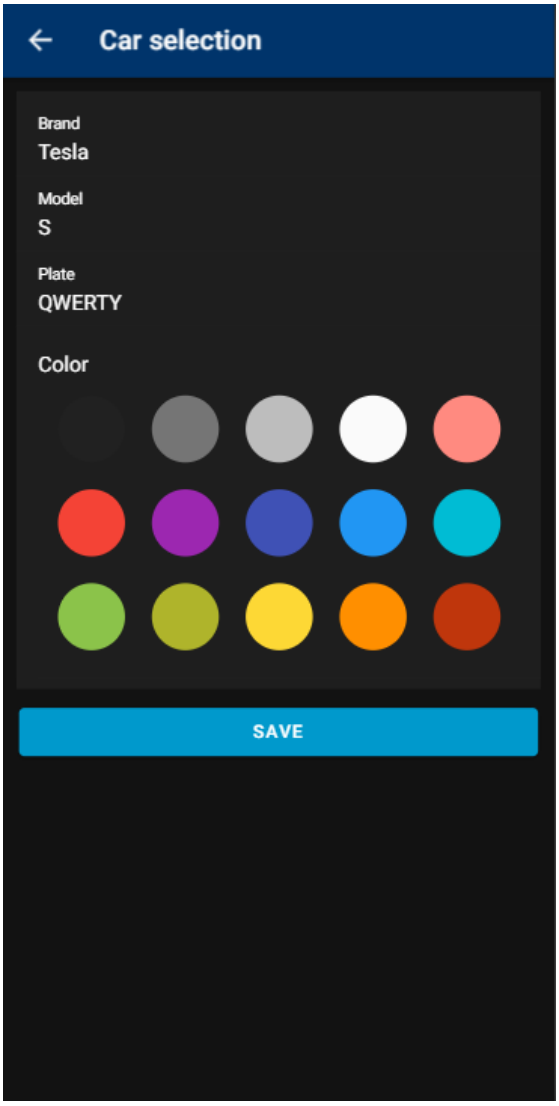


2. Aparecerá información sobre tu ruta y tu vehículo. Para establecer o modificar la información en ambas, pulsa en el botón de editar a la derecha en cada título.



3. En la pantalla de selección de ruta introduce el punto de salida, el destino de tu viaje y la distancia máxima de desvío de tu ruta para recoger pasajeros.

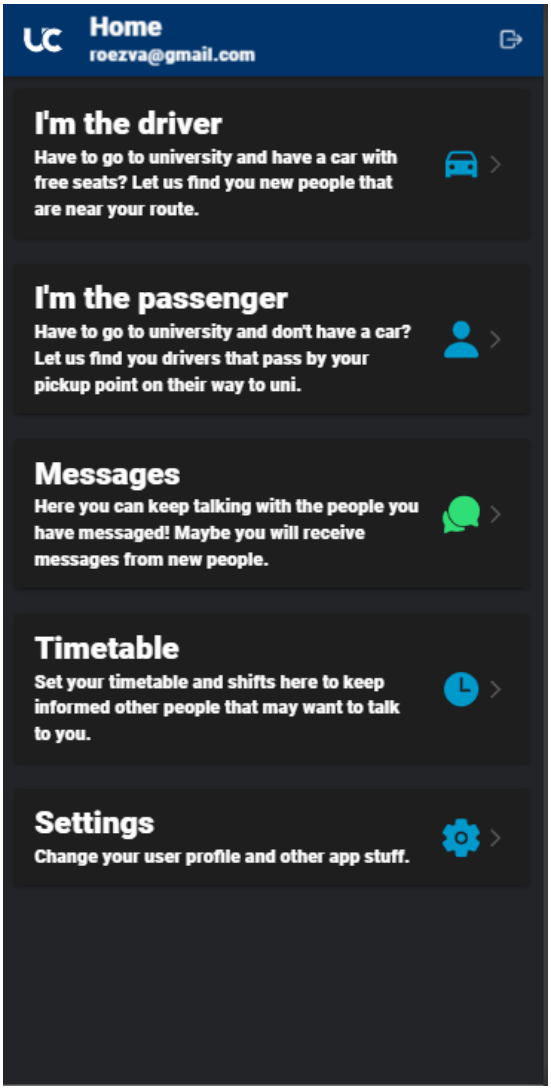
   Puedes seleccionar un punto de origen en el mapa pulsando en él o introduciendo una dirección en el buscador. Para guardar los cambios pulsa el botón SET THIS ROUTE, el cual te llevará a la pantalla anterior.



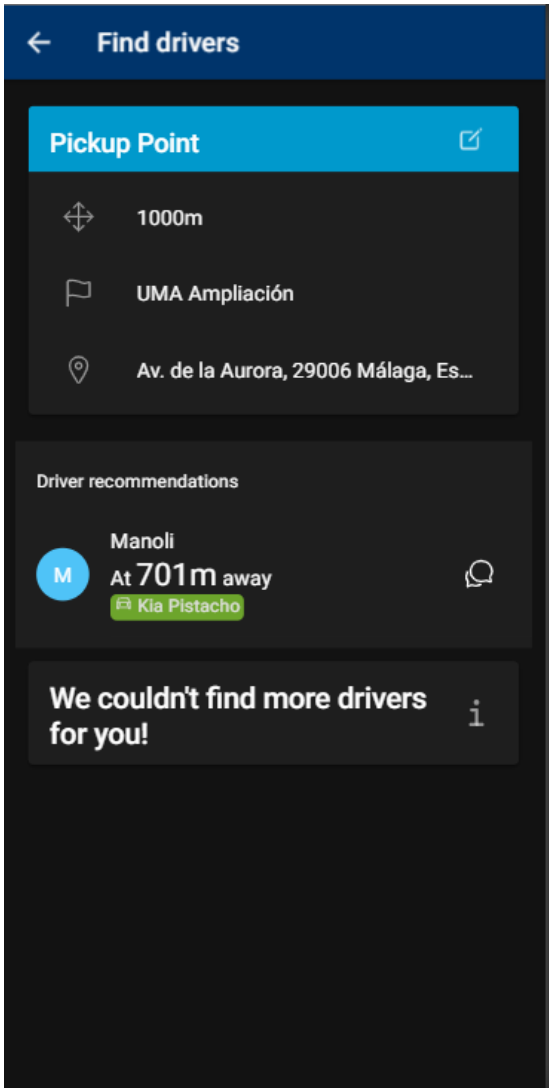
4. En la pantalla de información del vehículo puedes establecer la marca, el modelo, la matrícula y el color de este. Esta información será mostrada a otros usuarios cuando te busquen como conductor.

   Para guardar los cambios, pulsa el botón SAVE, el cual te llevará a la pantalla anterior.

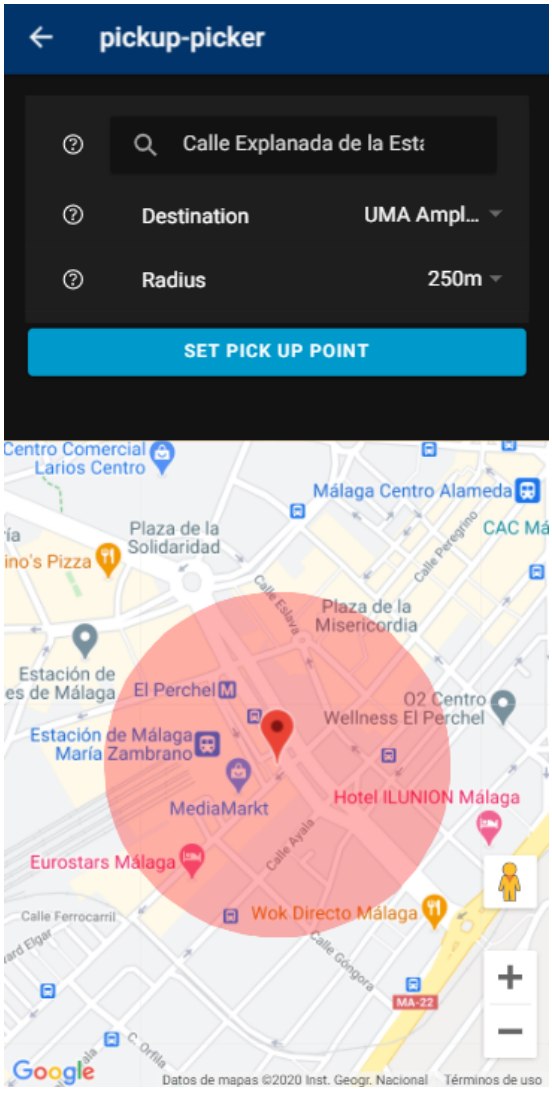
Obtener recomendaciones de conductores



1. Desde la pantalla principal, pulsa en I'M THE PASSENGER.

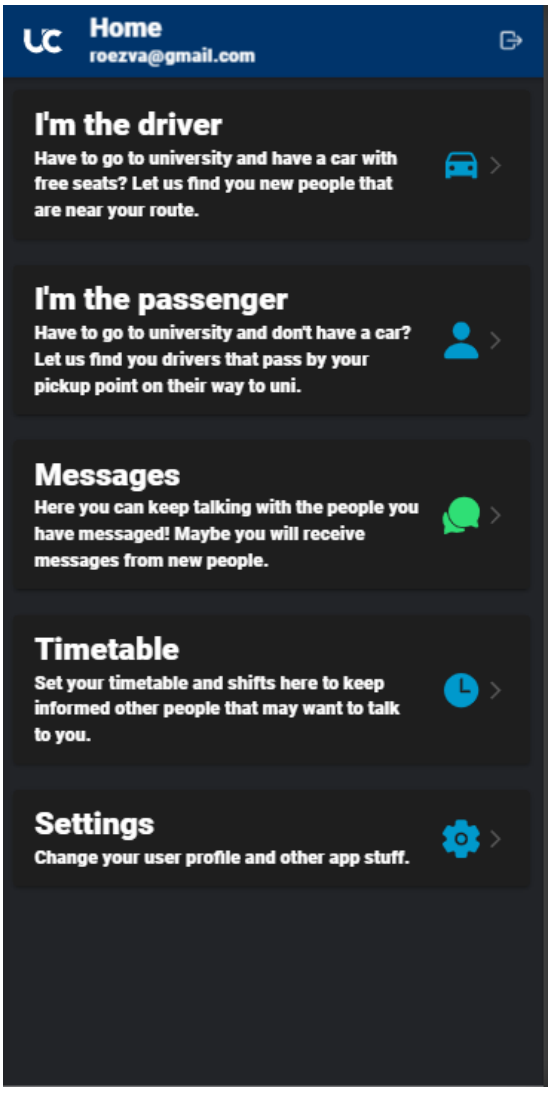


2. Aquí aparecerá la información de tu punto de recogida, así como los posibles conductores que pasan cerca de ese punto.
   Para seleccionar un punto de recogida pulsa en botón a la derecha de PICKUP POINT.

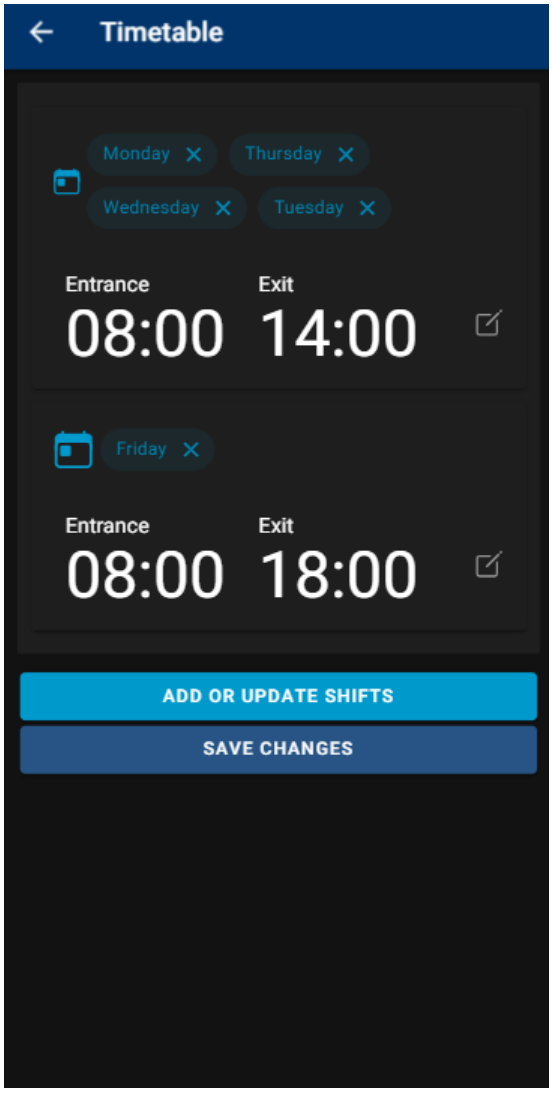


3. En la pantalla de selección de punto de recogida deberás indicar un punto de recogida pulsando en el mapa o introduciendo una dirección y la distancia máxima a la que estás dispuesto a desplazarte para que te recoja el conductor. Para guardar los cambios pulsa en SET PICK UP POINT, el cual te llevará a la pantalla anterior.

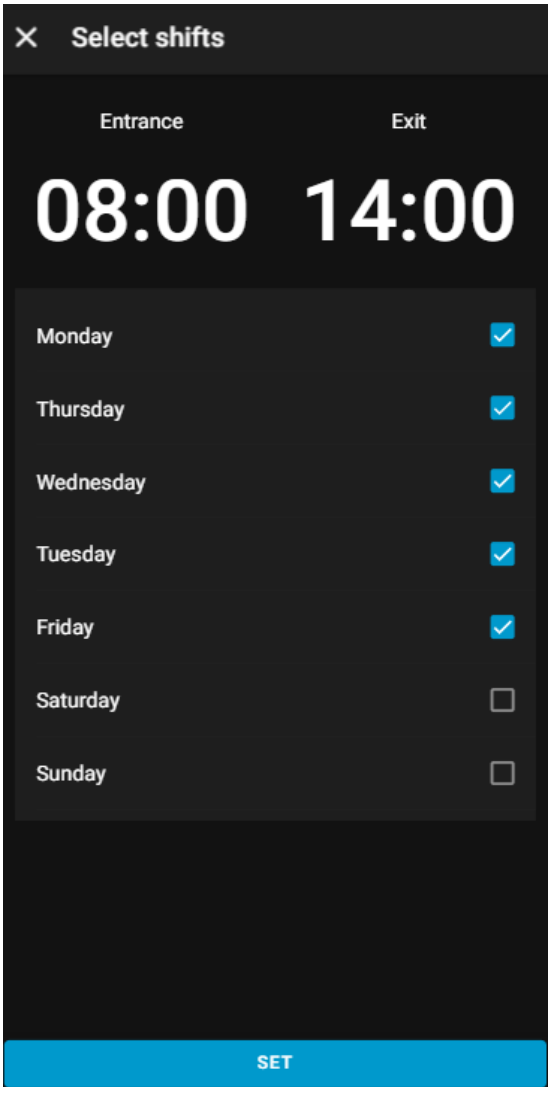
Establecer los horarios de entrada y salida a la universidad



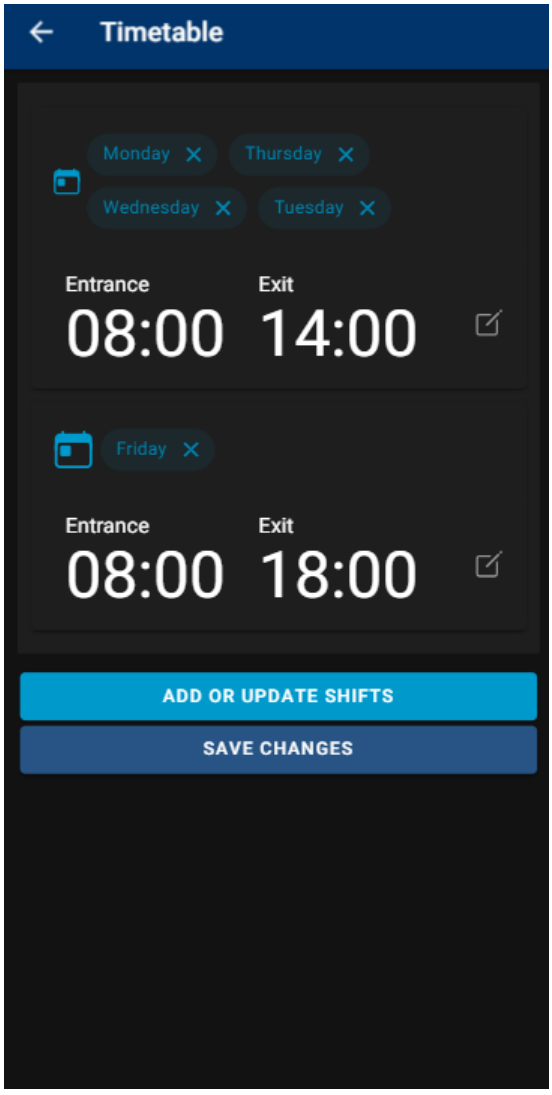
1. Desde la pantalla principal, pulsa TIMETABLE



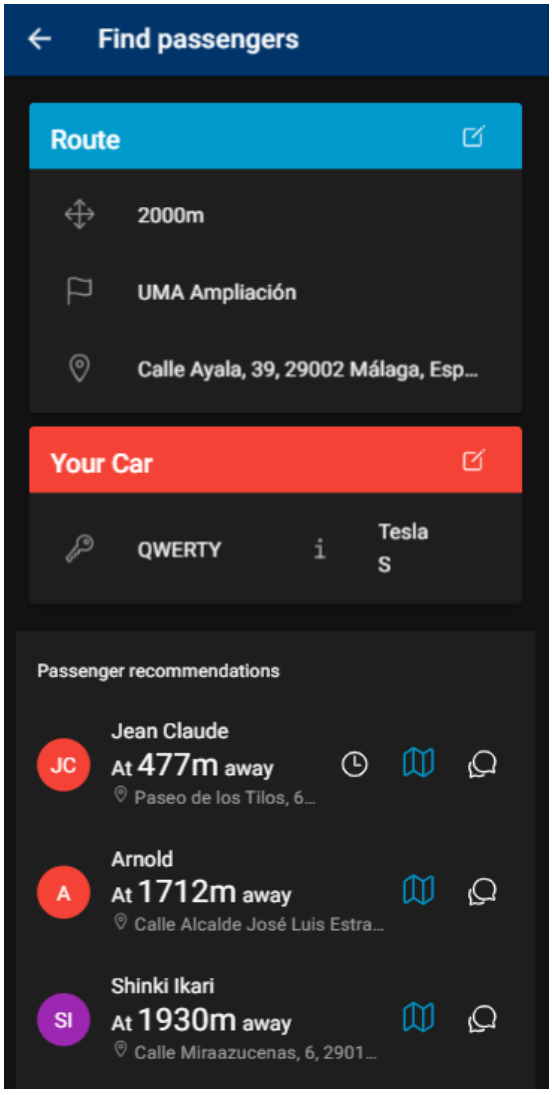
2. Aquí aparece la información de tus horarios de entrada y salida. Esta información se mostrará a otros usuarios para que comprueben si tu horario coincide con el suyo. Para añadir un turno pulsa ADD OR UPDATE SHIFTS.



3. Desde esta pantalla podrás seleccionar la hora de entrada y salida pulsando en cada hora y los días que tienes ese horario. Para guardar los cambios pulsa SET.

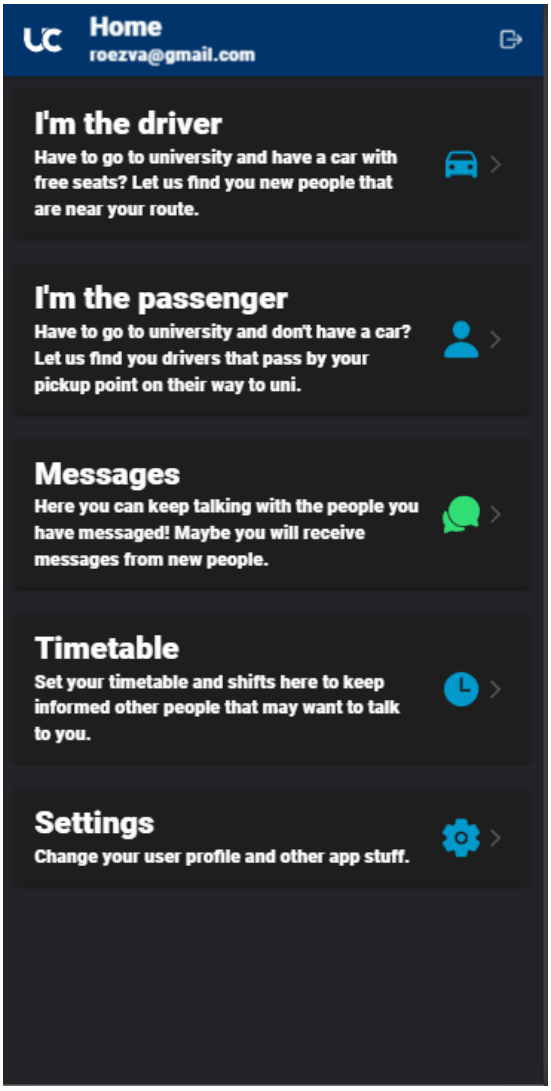


4. Cuando hayas terminado de seleccionar los turnos, pulsa en SAVE CHANGES para guardar los cambios.

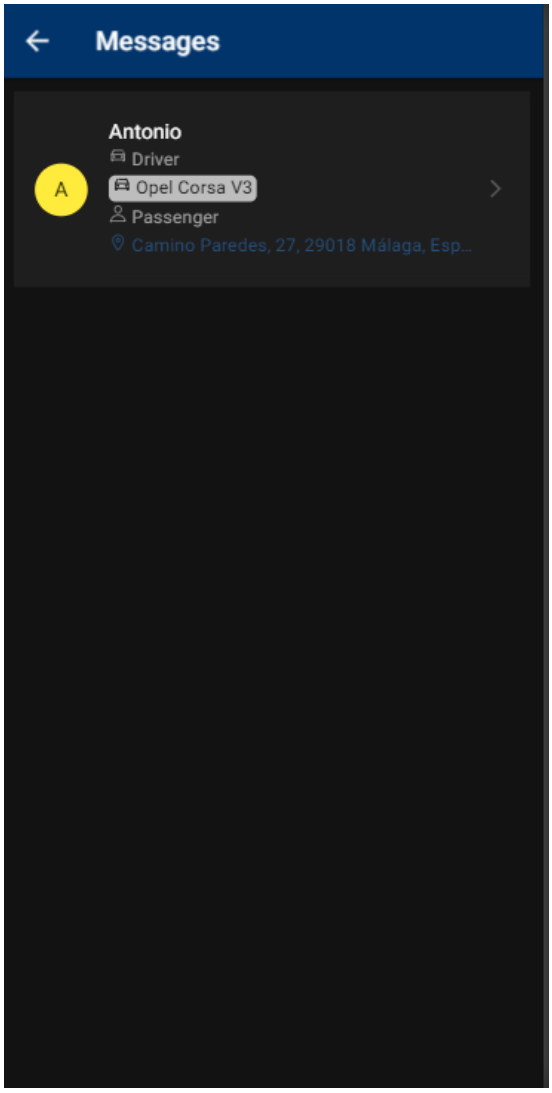


5. Cuando se rellenan los horarios, en las recomendaciones de pasajeros que cumplan ese horario aparecerá el icono de un reloj al lado del usuario, de modo que el conductor sabrá que coincide en horarios con ese pasajero

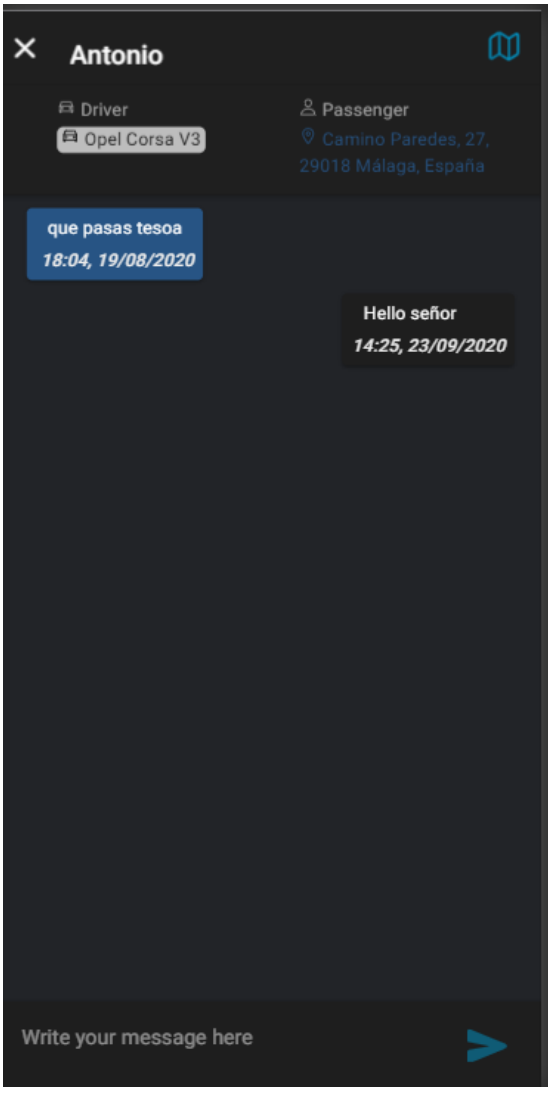
Entablar una conversación con un usuario



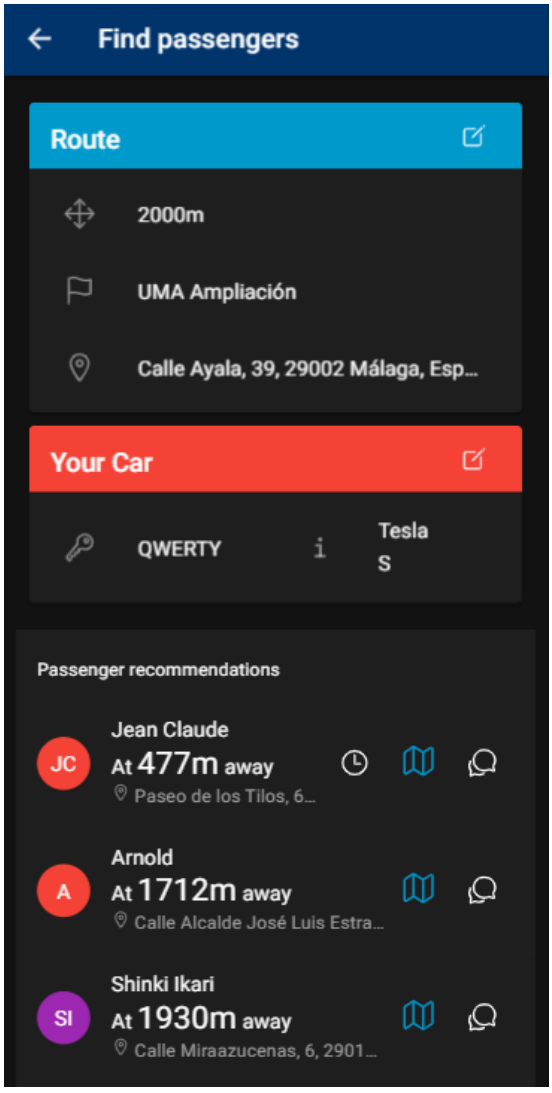
1. Si eres pasajero y has encontrado conductor o si eres conductor y has encontrado pasajero, se abrirá un chat entre ambos. Para acceder a todos los chats, pulsa sobre MESSAGES.



2. Aquí aparecerán todos los mensajes que tengas con otras personas de la aplicación. Para acceder a un chat en concreto, pulsa en el mensaje correspondiente.

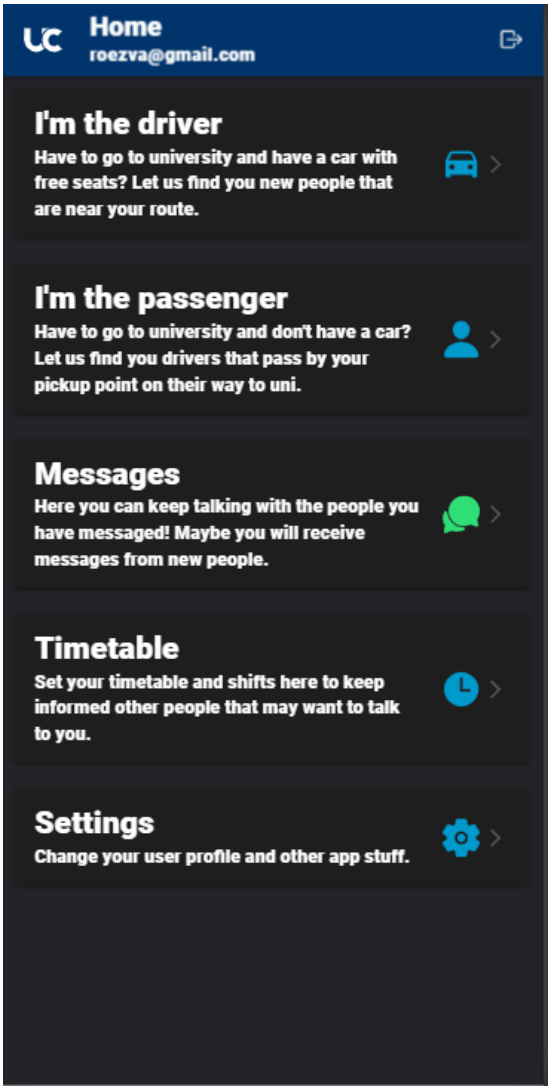


3. Como conductor, desde la pantalla de mensajes podrás ver la información del pasajero. Como pasajero, podrás ver la información del conductor y el resto de sus pasajeros.

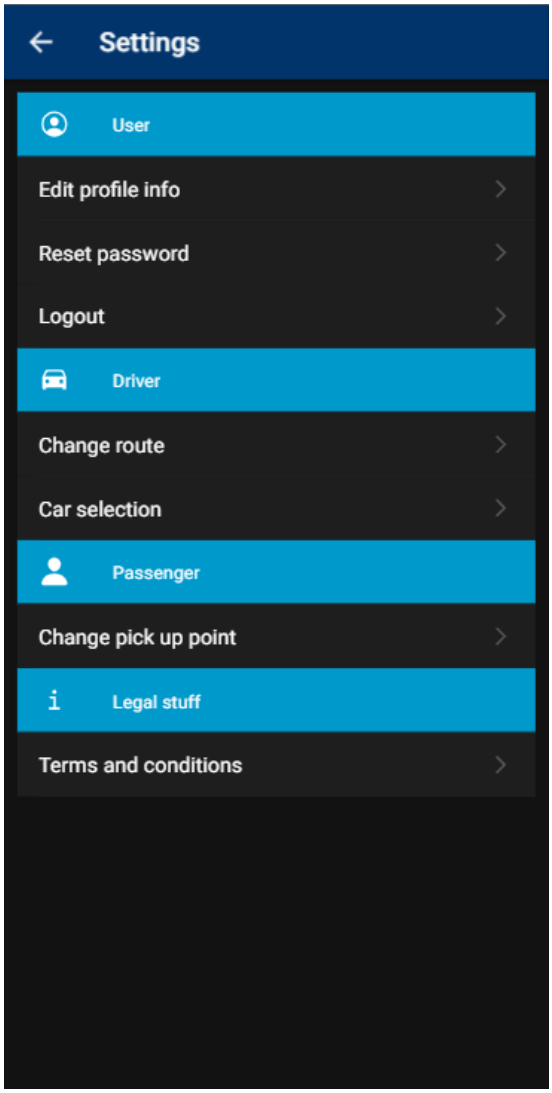


4. Puedes acceder a una conversación directamente en cualquier momento pulsando el icono del bocadillo de texto a la derecha de cualquier recomendación de usuario.

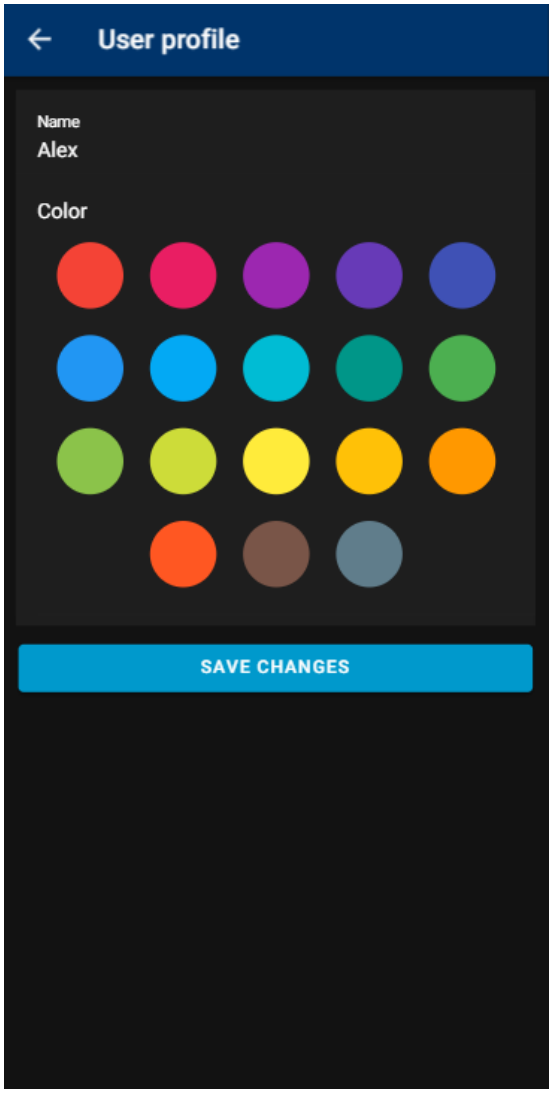
Modificar mi perfil



1. Para modificar tu perfil, pulsa en SETTINGS.

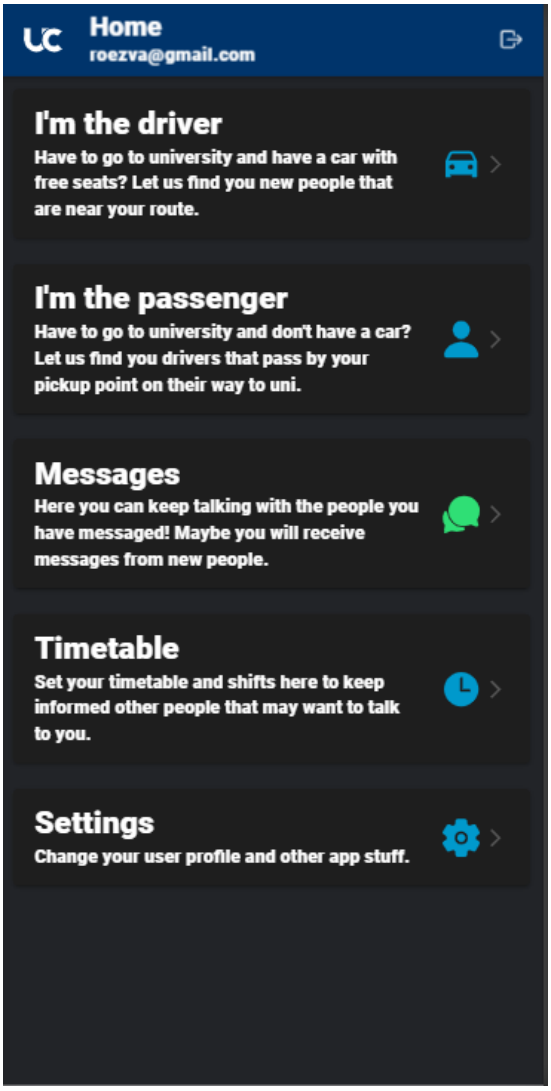


2. Pulsa en EDIT PROFILE INFO.

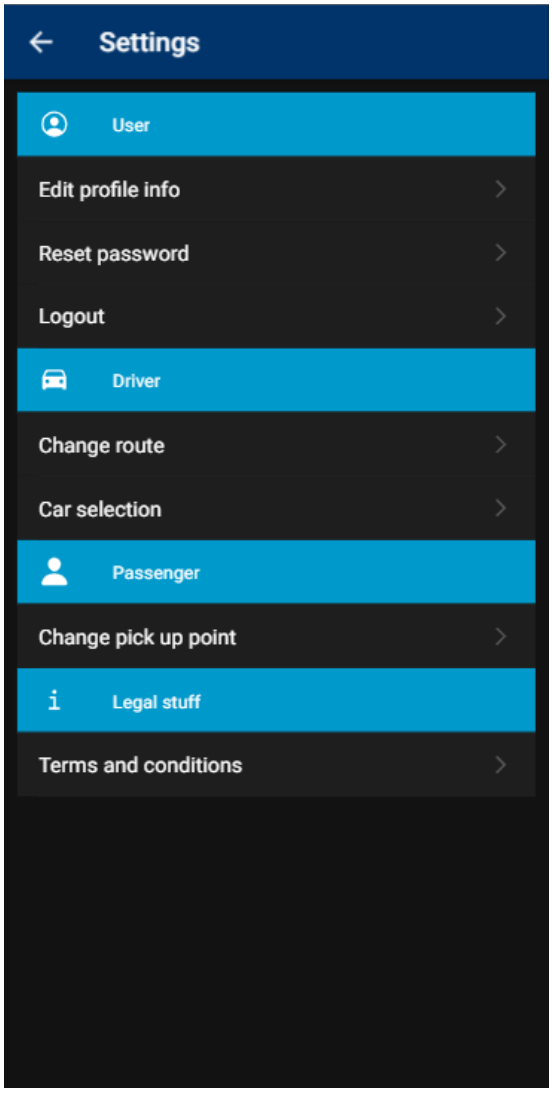


3. Aquí puedes modificar la información de tu perfil. Para guardar los cambios pulsa en SAVE CHANGES.

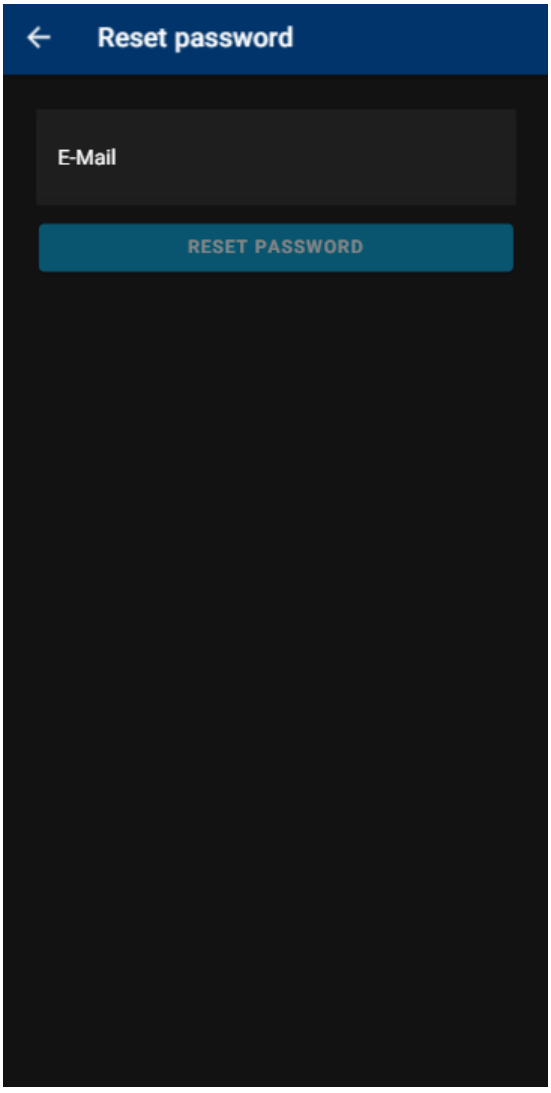
Resetear mi contraseña



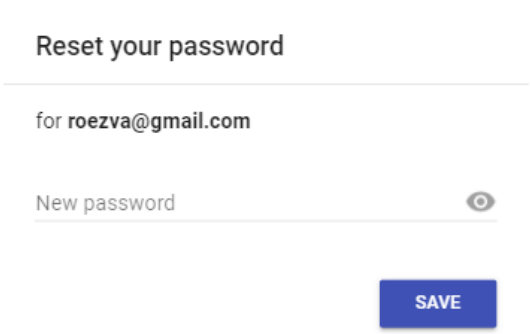
1. Pulsa en SETTINGS.



2. Pulsa en RESET PASSWORD.

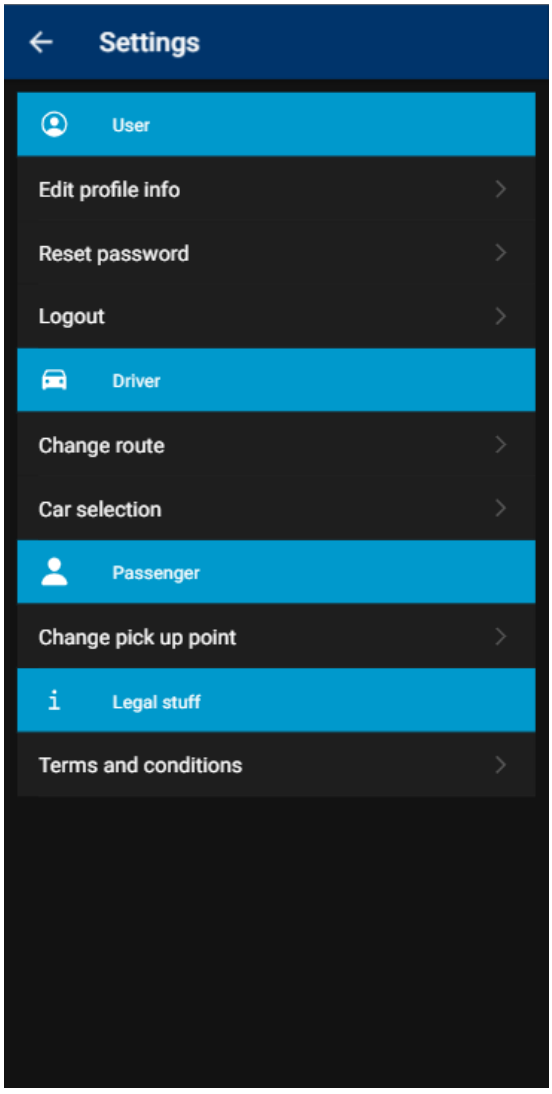


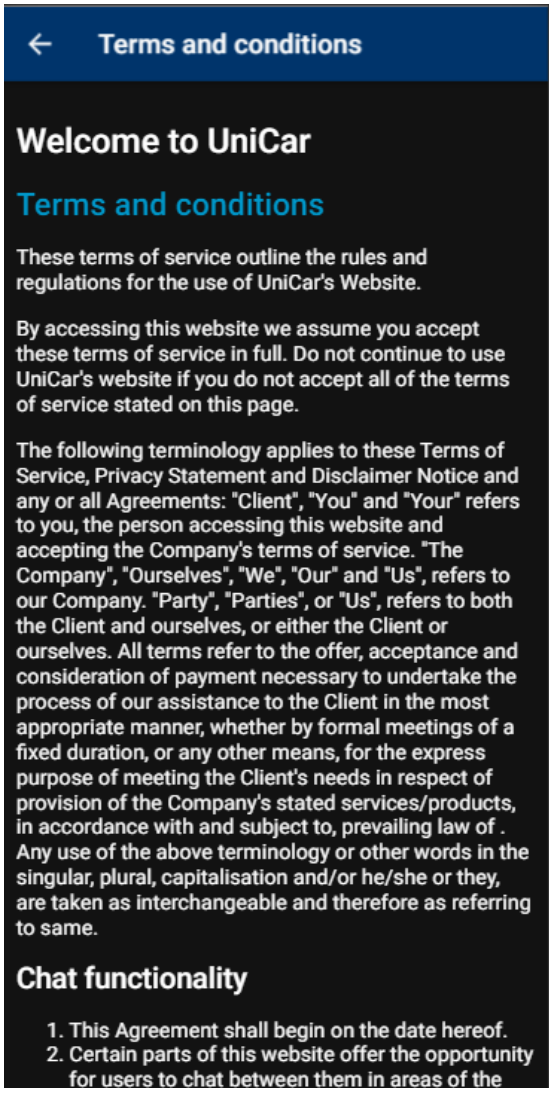
3. Escribe el email con el que te has registrado y pulsa en RESET PASSWORD. Se te enviará un mensaje de correo con un enlace de recuperación.



4. Haz click en el enlace, escribe tu nueva contraseña y luego pulsa en SAVE para guardar tu nueva contraseña..

Comprobar los términos y condiciones y otras opciones





Desde la página Settings podrás editar la información de tu perfil y las diferentes configuraciones de pasajero o conductor desde un mismo sitio. También puedes cerrar sesión pulsando en LOGOUT.

Desde esta página o desde la página de inicio de sesión o en la página de registro puedes revisar la información legal pulsando en el botón TERMS AND CONDITIONS.

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S. DE INGENIERÍA INFORMÁTICA

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga