UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Graduado en Ingeniería del Software

NoLoneCommuter – Compartir coche nunca fue tan sencillo
NoLonoCommuter – Sharing car has never been so easy

Realizado por
González Doña, José Alonso

Tutorizado por
Luque Polo, Gabriel Jesús

Departamento
Lenguajes y Ciencias de la Computación
UNIVERSIDAD DE MÁLAGA

MÁLAGA, septiembre 2021

UNIVERSIDAD
DE MÁLAGA

E.T.S. INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADUADO EN INGENIERÍA DEL SOFTWARE

NoLoneCommuter – Compartir coche nunca fue tan sencillo

NoLoneCommuter – Sharing car has never been so easy

Realizado por
González Doña, José Alonso

Tutorizado por
Luque Polo, Gabriel Jesús

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, SEPTIEMBRE DE 2021

Fecha defensa: septiembre de 2021

Resumen

Hoy en día, el calentamiento global y la crisis económica conforman dos grandes amenazas, y sin duda el uso de los vehículos motorizados contribuye negativamente a su resolución. Es por ello que compartir vehículo se trata de una de las mejores formas de ayudar tanto a nuestro planeta como a nuestro bolsillo.

Por ello, este trabajo pretende desarrollar y documentar una aplicación móvil que permita a sus usuarios compartir vehículo de la forma más sencilla posible: pulsando únicamente un botón. NoLoneCommuter proporciona por ello un servicio distinto al de otras muchas aplicaciones que permiten compartir vehículo, ya que ninguna de ellas recoge información de tus rutas para que, simplemente cuando toques un botón, se te muestre una lista de personas que realizan viajes similares a los tuyos y podrán darte el servicio que necesitas, adaptándose a cualquier tipo de necesidad que requieras de cubrir; y se te permita contactar con ellos de forma sencilla.

La idea principal tras NoLoneCommuter es generar una comunidad de conductores de vehículos motorizados que compartan viajes entre ellos, fortaleciendo sus amistades y contribuyendo positivamente en el mantenimiento de nuestro planeta y en la mejora de su economía, ayudándose de las tecnologías más empleadas en el sector, como pueden ser *React Native + TypeScript*, *Google Firebase* o *Go Language*.

Palabras clave: NoLoneCommuter, Compartir Vehículo, Calentamiento Globlal, Crisis Económica, Aplicación Móvil.

Abstract

Nowadays, global warming and economic crisis shape two huge threats on our society, and without a doubt, the abusive use of motor vehicles is not helping to solve the problem. This is the main reason why sharing vehicles is one of the best ways to help both our planet and our wallet.

This project pretends to develop a mobile application that allows their users to share vehicle on the easiest and fastest way: simply touching a button. NoLoneCommuter presents, to the best of my knowledge, a never given before service, due to there are many other applications that allow you to share your vehicle, but none of them retrieve your location information so, with only one touch, you can see a list of people interested on sharing vehicles with you, who will give you the service you are looking for, adapting to meet all your need; and allows you to conctact with them easily.

The main idea behind NoLoneCommuter is to generate a motor-vehicle users comunity that want to share vehicles one another, forming new friendships and contributing positively on the planet and wallet maintenance, held by the newest and more used technologies like *React Native + TypeScript*, *Google Firebase* or *Go Language*.

Keywords: NoLoneCommuter, Car Sharing, Global Warming, Economic Crisis, Mobile App.

Índice

1 Introducción

En este primer capítulo de la memoria se describirán los motivos que me impulsaron a realizar este proyecto y los principales objetivos que he perseguido durante su desarrollo y futura defensa.

También se describirá en detalle la estructura de esta memoria. Por supuesto, en todo momento se justificarán las decisiones tomadas y se mostrará el avance de estas a los largo del tiempo de desarrollo del producto, hasta alcanzar su estado final.

1.1. Motivación

En general, todos tenemos asumido que cada cual tiene su vehículo que debe usar para las tareas más cotidianas, como asistir al trabajo, y que es extremadamente normal emplearlo en solitario día a día; pero esta idea desemboca en un mayor daño en el medio ambiente y en nuestra economía.

Existe una muy simple solución: compartir vehículo. Si encontrásemos a alguien que hiciese una ruta similar a la nuestra, los mismo días que nosotros la realizamos, podríamos compartir vehículo y gastos con él, contribuyendo a la conservación de nuestro planeta y de nuestra economía. Pero esto da lugar a una clara pregunta: ¿quién está dispuesto a compartir vehículo conmigo? Pues NoLoneCommuter facilita enormemente la respuesta a esta pregunta.

NoLoneCommuter permite a sus usuarios crear su propio perfil en el que añadir sus necesidades, y toma automáticamente datos de las rutas que sueles realizar, haciendo que simplemente tocando un botón, te aparezca un listado de usuarios que están dispuestos a compartir vehículo contigo y realizan rutas similares. A partir de ello, puedes seleccionar a quién veas más conveniente y contactar con él a través del chat interno de la aplicación para comenzar a ahorrar y reducir las emisiones contaminantes, contribuyendo a mejorar el medio ambiente.

1.2. Objetivos

El principal objetivo de la aplicación es facilitar a sus usuarios encontrar a otros compañeros que estén dispuestos a compartir vehículo con ellos de la forma más rápida y sencilla posible.

Para lograr el anterior objetivo, era necesarios cubrir otras metas menos ambiciosos, como permitir a los usuarios la gestión de sus perfiles, dándoles la posibilidad de añadir algunos datos personales y de contacto, y sobre todo de describir sus necesidades y qué necesidades pueden cubrir en caso de tomar el role de conductores. Con esta información, NoLoneCommuter podrá comprobar si dos usuarios son compatibles o no, evitando así malentendidos y problemas futuros.

El segundo objetivo genérico es capturar de forma eficiente y eficaz las rutas que recorren sus diferentes usuarios. Para ello, la aplicación trabaja en segundo plano enviando, bajo ciertos criterios, la ubicación del usuario. Con esos datos, el servidor web genera las rutas de los usuarios, que se compararán de forma eficiente para cumplir de nuevo el objetivo general del sistema: encontrar compañeros de viaje.

El tercer objetivo fue el de integrar un chat en la aplicación que permitiese a los usuarios comunicarse de forma sencilla y efectiva. Si bien los perfiles de los usuarios cuentan con una sección de información de contacto, el contar con un chat interno de la aplicación centraliza esta tarea, facilitando el proceso de compartir vehículo en general.

Otros objetivos relativos al desarrollo del proyecto podrían ser los siguientes:

En cuanto a la documentación, se pretende escribir una memoria clara y limpia, que plasme todo el trabajo tras el producto final, así como las distintas etapas de este y la evolución del producto conforme van avanzando las iteraciones.

También se quiere generar una documentación lo más completa posible, que refleje tanto cómo usar el producto, que abarca, como otras cuestiones más técnicas relativas a su modelo de datos, casos de uso o recopilación de requisitos.

En cuanto al código tanto de la aplicación móvil como del servidor, se quiere que este sea limpio, claro, fácil de leer, de adaptar y de expandir. Todo esto se consigue formateando siempre el código según los estándares de cada lenguaje, subdividiendo las distintas tareas en una o varias funciones de no más de veinte o treinta líneas, y siguiendo buenas prácticas como

organizar los ficheros en distintas carpetas o creando constantes al incio de los documentos para los valores por defecto.

Por último, tenía como objetivo general aprender y conseguir experiencia, tanto en el proceso de Ingeniería del Software como en el manejo de las tecnologías. Por eso me he decantado por emplear tecnologías con las que no me siento familiarizado que me empujen fuera de mi zona de confort.

1.3. Metodología

Para el desarrollo del proyecto se empleó una metodología semejante a SCRUM, pero con algunas variaciones. Más concretamente, se empleo una metodología iterativa que costó de varios ciclos en los que

1. se analizaban requisitos del sistema,
2. se desarrollaban los casos de uso,
3. se elegían uno o varios casos de uso,
4. se implementaban los casos de uso elegidos,
5. se comprobaba su correcto funcionamiento, y
6. se actualizaba la memoria del proyecto con los nuevos datos.

De esta forma, el producto fue evolucionando de forma casi continua hasta alcanzar el estado final.

Cabe mencionar que, en las primeras etapas, la mayor parte del tiempo y esfuerzo se invertía en analizar requisitos y desarrollar casos de uso; en las etapas intermedias, se abordó la mayor parte de la implementación; y las etapas finales se reservaron para añadir detalles al código y a la memoria del proyecto y para preparar la defensa.

En cuanto al contacto con el tutor *Luque Polo, Gabriel Jesús*, mayormente intercambiamos correos electrónicos con dudas sobre detalles de los algoritmos o de la documentación necesaria para el proyecto. También realizamos un total de cuatro reuniones en las que mostraba el estado del producto y proponíamos mejoras y cambios sutiles, que me fueron de gran ayuda para conocer los puntos fuertes y débiles de la aplicación.

1.4. Estructura del documento

El presente documento se divide en ocho capítulos, que describen los aspectos mencionados a continuación:

1. **Introducción**. En el capítulo de introducción se comentan aspectos generales del proyecto, incluyendo la motivación de este, sus objetivos principales, la metodología empleada y la actual estructura del documento.

2. **Mercado**. El capítulo de mercado sirve para comprender cómo se sitúa el producto desarrollado frente a sus competidores y productos similares, destacando qué lo hace único y competitivo.

3. **Tecnologías empleadas**. El capítulo de tecnologías empleadas detalla qué tecnologías se han empleado para el desarrollo del proyecto. Esto incluye desde lenguajes de programación y almacenamiento persistente, hasta herramientas de gestión de versiones, IDEs, editores de texto o herramientas de desarrollo de diagramas.

4. **Análisis**. El capítulo de análisis se centra en el análisis software del proyecto, listando los requisitos tanto funcionales como no funcionales, los distintos casos de uso y la matriz de trazabilidad.

5. **Modelado y diseño**. El capítulo modelado y diseño recoge tanto el proceso de modelado de ingeniería del software, como el proceso de diseño gráfico de la aplicación. Esto incluye la descripción completa de la base de datos y las estructuras auxiliares; y las descripción del proceso empleado para el desarrollo del diseño gráfico de la aplicación.

6. **Desarrollo**. El capítulo de desarrollo describe el funcionamiento concreto de cada uno de los algoritmos y módulos de la aplicación, así como las decisiones tomadas.

7. **Conclusiones y trabajos futuros**. El capítulo de conclusiones y trabajos futuros comparte las conclusiones finales del proyecto y presenta propuestas de mejora para la aplicación en forma de nuevas ideas con un enfoque claro y guiado.

También se han incluido dos apéndices:

1. **Manual de despliegue e instalación**. Este anexo detalla al usuario los pasos a seguir para poder desplegar el servidor de la aplicación e instalar la aplicación móvil en su *smartphone*.

2. Manual **de usuario**. Este anexo explica cómo emplear cada una de las funcionalidades que presenta la aplicación de forma detallada.

2
Mercado

En este capítulo se debatirá el lugar que ocupa el proyecto en el mercado, destacando tanto la plataforma sobre la que opera como su posición frente a otras aplicaciones similares, así como los puntos fuertes de NoLoneCommuter frente a su competencia más directa.

Al igual que en todos los epígrafes del documento, a lo largo de este capítulo también aparecerán aclaraciones sobre la toma de decisiones.

2.1. Plataforma

Como se comentó anteriormente, *NoLoneCommuter* se trata de una aplicación para dispositivos móviles con sistema operativo  *Android* [1].

La elección de crear una aplicación móvil frente a, por ejemplo, una aplicación web, se debe principalmente a que el sistema requería de obtener la ubicación en tiempo real de los usuarios de forma casi continua, lo cual es imposible en una aplicación web. Otro motivo de peso para decantarnos por desarrollar una aplicación móvil fue que más del 99 % de los españoles tiene un teléfono inteligente en casa [2], lo cual hace que *NoLoneCommuter* pueda llegar a mucha más gente.

También se barajó la posibilidad de realizar una aplicación para relojes inteligentes, pero la documentación sobre cómo desarrollar para estas tecnologías es más reducida y escueta, dado lo reciente de la tecnología. Además, sólo el 18 % de los españoles tiene un reloj inteligente [3], lo cual reduce mucho el potencial de crecimiento de la comunidad, y haría que *NoLoneCommuter* llegase a mucha menos gente.

Por otra parte, la elección de *Android* se debe principalmente a que más del 90 % de los dispositivos móviles en España emplean este sistema operativo [4], por lo que desarrollar una aplicación *Android* permite llegar a más personas en nuestro país, y formar una mayor comunidad. Además, también se trata del sistema operativo compatible con el mayor número de

marcas de teléfonos.

Otro motivo para desarrollar en *Android* es la abundante documentación que se puede encontrar en línea sobre este sistema operativo. En general, con introducir una duda en un buscador web se pueden encontrar decenas de soluciones simples y válidas al problema.

En un principio, se planteó que la aplicación también fuese compatible con dispositivos *iOS*, para aprovechar al máximo la multicompatibilidad que puede proporcionar *React Native*, pero finalmente se optó por hacer la aplicación para *Android* por los siguientes motivos:

1. Algunas librerías empleadas para el desarrollo del proyecto no eran compatibles con dispositivos *iOS*, lo cual implicaba buscar otras librerías similares para este sistema operativo. Otra opción era generar por cuenta propia los componentes que cubriesen las necesidades, pero eso rompía la esencia de *React Native* y consumiría mucho tiempo de desarrollo.

2. No tengo a mi alcance ningún dispositivo móvil con sistema operativo *iOS*, y los emuladores de esta plataforma no funcionan de forma rápida en mi computador. Esto entorpecía y ralentizaba mucho el probar el código desarrollado.

Por esos motivos, preferí desarrollar una aplicación Android que funcionase correctamente y se viese de forma agradable, a una aplicación para ambos sistemas operativos con, probablemente, un peor acabado.

2.2. Aplicaciones similares

En el mercado existen numerosas aplicaciones que permiten a sus usuarios compartir vehículo, como pueden ser *BlaBlaCar* [5] o *Carpooling* [6]; además de otras como *Uber* [7], *Cabify* [8] o el servicio de taxis clásico, que ponen un vehículo al servicio de sus clientes.

La principal diferencia entre estas aplicaciones y *NoLoneCommuter* es que esta última facilita a sus usuarios compartir vehículos sin que estos tengan que introducir la ruta a seguir o el punto de destino, sino que analiza las rutas de los usuarios para que estos, con un sólo clic, puedan encontrar una lista de usuarios dispuestos a compartir coche con ellos.

Otra característica a destacar de *NoLoneCommuter* es que los compañeros que la aplicación recomienda siempre podrán cumplir con las necesidades del usuario. Esto es posible gracias

a que en el perfil, podrás añadir información sobre las necesidades que tienes, así como las que puedes cubrir. Además, si algún usuario no desea hacer visible cierta información, pero sí quiere que la aplicación la tenga en cuenta al buscarle un compañero, puede ocultar la información que desee de su perfil y esta no será mostrada a ningún otro usuario

3

Tecnologías empleadas

En este apartado se analizarán todas las tecnologías empleadas para el desarrollo del trabajo. En cuanto a las tecnologías empleadas para el desarrollo de este trabajo, con el objetivo de aprender nuevos conceptos y crecer como profesional, opté por usar las tecnologías más recientes y punteras del mercado de software actual, dejando a un lado las cubiertas por la docencia. en el Grado en Ingeniería del Software.

3.1. Almacenamiento persistente

Para el almacenamiento persistente de la aplicación, decidí emplear los servicios que proporciona *Google Firebase* [9], concretamente la *Firestore Database* [10].

En concreto, me decanté por esta base de datos por ser de tipo NoSQL [11] y sencilla de emplear. El hecho de que sea una base de datos NoSQL permite alterar el modelo de datos de forma sencilla, sin tener que modificar implícitamente la estructura de datos, como sucede con las bases de datos relacionales. Esto me fue de gran ayuda durante el desarrollo del proyecto, ya que durante el proceso de desarrollo, se sucedieron muy numerosos cambios en el modelo de datos de la aplicación, y este tipo de bases de datos me permitía tener que despreocuparme por alterar el modelo de la base de datos, y simplemente centrarme en que la aplicación tuviese un modelo de datos le permitiese hacer lo que debe de forma eficaz y eficiente.

Por otra parte, destacar que *Firestore Database* es extremadamente sencillo de usar, pues Google proporciona una API muy intuitiva así como múltiples tutoriales, *FAQ* y foros de resolución de problemas.

3.2. Backend

En cuanto al lenguaje de programación empleado para el servidor de la aplicación, me decanté por  *Go Language* [12].

Existen cuatro motivos por los que decidí emplear *Go Language*: su eficiencia, la sencillez de sus expresiones, la librería *Gin* [13] y su integración con JSON.

Go Language se trata de un lenguaje de programación desarrollado en Noviembre de 2009 por *Robert Griesemer Rob Pike Ken Thompson*, empleado de Google, con el objetivo de crear un lenguaje similar a *C*, pero más absequible para nuevos usuarios y rápido.

Go Language cuenta con tipado estático, recolector de basura con opción de apagado, tipado estructural y *memory safety*, lo cual lo hace eficiente, rápido y seguro.

Otra característica que me atrajo de *Go Language* es la sencillez de sus expresiones, y su capacidad de expresar mucho en una línea, como la declaración de variables en el *scope* de una condición.

La existencia de la biblioteca *Gin* fue la que realmente me impulsó a emplear *Go Language* como lenguaje de programación del backend del proyecto, pues permite realizar un servidor HTTP [14] con unas pocas líneas de código, y se adapta perfectamente al modelo de router, controladores y gestores de almacenamientos típico de los servidores web. Cuenta con una gran variedad de opciones que permiten añadir middlewares, proxies y certificados HTTPS con extrema facilidad.

Por último, cabe destacar su integración con JSON. Esta característica fue de gran ayuda, ya que permite convertir objetos JSON enviados por el frontend en estructuras de *Go Language* de forma automática simplemente por añadir un *tag* a cada atributo cuando se define el tipo.

3.3. Frontend

Para el desarrollo de la cara visible del proyecto me decanté por emplear  *React Native* [15] +  *TypeScript* [16], empleando los componentes core [17] que este ofrece, y creando también los míos propios para personalizar la apariencia y funcionalidad de la aplicación.

En general, elegí *React Native* como *framework* para el desarrollo del frontend de la aplicación porque aporta más opciones de personalización gráfica y un mejor acabado en los componentes que otras tecnologías clásicas de desarrollo de aplicaciones móviles.

Además, *React Native* cuenta con otras características muy útiles, como la alta reutilización se sus componentes, su enorme capacidad de personalización, pudiendo crear elementos realmente agradables a la vista, personalizables y útiles que se adapten perfectamente a las necesidades; así como la capacidad de modificar la apariencia de los componentes cuando sea necesario, dotando de dinamismo a la aplicación móvil.

Por otro lado, cabe destacar el uso de la librería *Axios* [18] para la gestión de peticiones y respuestas al servidor. Axios es una biblioteca ligera para JavaScript [19] que permite realizar peticiones a servidores, recibir sus respuestas y facilitar el tratamiento de esa información, lo cual facilitó mucho la comunicación con el servidor web.

Finalmente, me decanté por emplear *TypeScript* como lenguaje de programación, ya que el poder dotar de tipado a *JavaScript* es extremadamente útil para dar mayor legibilidad y reusabilidad al código, y evitar los múltiples problemas y confusiones que *JavaScript* suele provocar.

Typescript permite generar código más limpio, claro, legible, entendible y reutilizable, así como permitir un mejor seguimiento de los errores y facilitar su resolución.

3.4. Control de versiones

En cuanto a herramienta de control de versiones, opté por *GitHub* [20].

Me decanté por *GitHub* por ser la que considero como la herramienta *Git* [21] más completa y sencilla de usar. Si bien es cierto que no he empleado muchas de las opciones que esta herramienta proporciona, me ha sido muy cómoda la gestión de commits y la creación y revisión de *Pull Requests*.

Otro punto muy a favor de *GitHub* es que, al ser tan conocido, tiene una muy buena integración con otras herramientas como *Visual Studio Code* [22].

3.5. IDE

Como IDE [23] decidí emplear *Visual Studio Code*.

El principal motivo es que se trata de la herramienta de desarrollo software más empleada a nivel mundial, y que, por ello, tiene integración con muchas otras herramientas muy útiles, así como mucha documentación en línea fácil de acceder.

El otro motivo que me incentivó a usarlo es que personalmente, me atrae su simpleza y apariencia, y me parece que con algunos plugins añadidos se trata de un IDE casi perfecto.

En cuanto a plugins, instalé los siguientes: *Code runner* [24], *Go* [25], *Go Nightly* [26], *Golang* [27], *json* [28] y *Prettier* [29].

3.6. Editores de texto

Como editores de texto, empleé dos: *Microsoft Word* [30] y LaTeX [31].

Decidí emplear *Microsoft Word* como editor de texto para documentos como los casos de uso o el documento de requisitos, ya que me parece más simple de emplear, y da igualmente un buen resultado. Además, es muy sencillo encontrar información para resolver errores en línea.

Por otra parte, usé *LATEX* como editor de texto para redactar la presente memoria, al ser mucho más potente y permitir un formateado mucho más elegante y adaptable. Cabe destacar que, pese a no haber empleado prácticamente esta herramienta con anterioridad, fue extremadamente sencillo adaptarse gracias a la plantilla proporcionada en la página web de la UMA [32] y a toda la documentación y resolución de problemas que hay disponible en la web.

3.7. Desarrollo de diagramas

Para desarrollar los distintos diagramas que anexan a este documento usé la herramienta *MagicDraw* [33].

MagicDraw se trata de un programa que permite confeccionar todo tipo de diagramas con un gran nivel de detalle y personalización, y garantizando que cumplan los estándares de modelado.

En general, he usado esta herramienta porque me he familiarizado mucho con ella a lo largo del grado y me parece extremadamente útil para generar diagramas UML muy completos.

Gestión de contenido multimedia Las imágenes de NoLoneCommuter se almacenan en *Cloudinary* [34].

Cloudinary se trata de una herramienta web que permite a sus usuarios subir imágenes y les proporciona una URL que se puede emplear para visualizar la imagen de forma sencilla. Es por ello que la base de datos de NoLoneCommuter nunca trabaja con fotos, sino con sus

enlaces, agilizando extremadamente el proceso.

El principal motivo por el que he empleado *Cloudinary* como gestor de contenido multimedia es su sencillez, pues con sólo una *key* que añadir a la cabecera HTTPS de la petición ya se sube la imagen y te devuelve el enlace. Es justo lo se necesitaba en un sólo paso y de forma simple.

3.8. Gestión de proyectos

Como herramienta de gestión del proyecto y sus tareas he utilizado *Trello* [35].

Me he decantado por esta herramienta porque es sencilla de usar y me permite crear varios tableros con diferentes columnas sobre los que organizar las distintas tareas a realizar.

Si bien es cierto que existen otras herramientas como *Jira* [36], las cuales considero superiores por permitir más tipos de tareas y niveles de jerarquía entre ellas, *Trello* se trata de una herramienta gratuita que ha suplido esa necesidad con creces.

De hecho, se trata de una de las tecnologías mencionadas que más he empleado y de las que más me enorgullezco de haber elegido, pues me ha permitido fácilmente gestionar las tareas en diferentes tableros y columnas para organizar el desarrollo del proyecto de forma extremadamente sencilla y eficaz.

4
Análisis

En este capítulo, veremos en detalle el análisis software del proyecto, es decir, cuáles son sus requisitos, sus casos de uso y la matriz de trazabilidad. También se puntualizarán decisiones tomadas.

Como es de suponer, los requisitos y casos de uso fueron evolucionando a lo largo del proceso de desarrollo, adaptándose a nuevos cambios y ampliándose o simplificándose según lo que se pretendía que fuese el producto final.

4.1. Requisitos

En esta sección se listarán los requisitos funcionales y no funcionales del sistema desarrollado. Para poder comprender correctamente los requisitos que se van a listar, es importante que se distingan claramente los conceptos de *ruta* y *viaje*. En el dominio de este sistema, se denomina como **ruta** a cualquier trayecto que haya realizado un usuario, sin que sea de ninguna forma necesario que esté compartiendo vehículo. Se denominará **viaje** a un trayecto que dos usuarios han realizado o planean realizar juntos, compartiendo vehículo.

4.1.1. Requisitos funcionales

1. Un usuario podrá crear una cuenta en el sistema.

2. Un usuario podrá iniciar sesión con su cuenta en el sistema.

3. Un usuario podrá actualizar la información de su cuenta. Esto incluye:

    a) Un usuario podrá modificar su información personal, incluyendo:

        1) su foto de perfil,

        2) su nombre completo, y

3) su descripción.

b) Un usuario podrá modificar su información de contacto, incluyendo:

1) su correo de contacto,
2) su teléfono de contacto, y
3) su cuenta de *Telegram*.

c) Un usuario podrá modificar sus preferencias, incluyendo:

1) si desea que se tomen datos de sus rutas,
2) si desea conducir, y
3) cuánta distancia está dispuesto a desviarse para compartir vehículo con otras personas, en caso de ser el conductor.

d) Un usuario podrá modificar la información de ayudas que necesita y que puede ofrecer, incluyendo:

1) si necesita ayuda por tener movilidad reducida, y si puede ofrecerla,
2) si está dispuesto a adaptar la música de su vehículo, y si necesita que se la adapten,
3) si está dispuesto a adaptar el aire acondicionado del vehículo, y si necesita que se lo adapten a él,
4) cuándo espacio tiene en el vehículo, y cuánto necesita, y
5) alguna otra información adicional sobre qué necesita o puede ofrecer.

e) Un usuario podrá elegir qué elementos de su cuenta desea que sean visibles para otros usuarios y cuáles no. Esto incluye toda la información personal, de contacto, de ayuda necesitada, de ayuda a ofrecer y de preferencias.

4. Un usuario podrá eliminar su cuenta de forma permanente.
5. El sistema podrá obtener información de la ubicación del usuario cada cierto tiempo, o cada vez que el usuario se desplace una cierta distancia.
6. Un usuario podrá decidir si desea que el sistema tome los datos de su ubicación o no.
7. El sistema mostrará al usuario una notificación permanente de que se está tomando su ubicación mientras esta se esté tomando.

8. Un usuario podrá ver la lista de sus rutas realizadas.

9. Un usuario podrá eliminar una ruta que haya realizado.

10. Un usuario podrá solicitar al sistema que le recomiende posibles compañeros de viaje para cierta ruta que tenga pensado realizar, mostrándole un listado de usuarios que suelen realizar una ruta similar y estarían dispuestos a compartir vehículo.

11. Un usuario podrá ver el perfil de los usuarios con los que el sistema les recomiende compartir vehículo.

12. Un usuario podrá solicitar un viaje de los que el sistema le recomendó, pudiendo fijar si desea o no conducir, la fecha y hora del viaje y un comentario.

13. Un usuario podrá ver la lista de sus viajes planeados.

14. Un usuario podrá marcar un viaje marcado como no visto como visto.

15. Un usuario podrá valorar los viajes que haya realizado, pero sólo una vez por viaje.

16. Un usuario podrá ver las conversaciones que tiene con otros usuarios.

17. Un usuario podrá ver los mensajes intercambiados con otro usuario en una conversación en concreto.

18. Un usuario podrá enviar mensajes a otros usuarios.

4.1.2. Requisitos no funcionales

1. El sistema cumplirá con la legislación de protección de datos vigente: *RGPD*.

2. El sistema se adherirá a los estándares dictados por la organización *ISO* sobre el desarrollo de software para dispositivos móviles.

3. El sistema solicitará el permiso de los usuarios para tomar datos sobre su ubicación.

4. El sistema responderá al usuario en un tiempo inferior a 1 minuto en cualquier petición.

5. El sistema estará disponible el 99 % de las ocasiones.

6. El sistema contará con un diseño software fácilmente reutilizable y adaptable.
7. El sistema será funcional desde cualquier lugar, siempre que se tenga acceso a Internet y se cuente con señal GPS.
8. El sistema tratará de minimizar el consumo de batería del teléfono móvil.
9. El sistema contará con un *manual de usuario* escrito en español.
10. El sistema contará con un *manual de instalación* escrito en español.
11. El sistema será funcional en dispositivos móviles con sistema operativo *Android*.
12. El sistema se comunicará con el servidor web mediante el protocolo *HTTP*.
13. La aplicación móvil estará desarrollada en *React Native*.
14. El servidor web estará desarrollado en *Go Language*.
15. El sistema empleará una base de datos NoSQL.

Cuando se pretenda referenciar alguno de los requisitos numerados anteriormente, se les nombrará como *RNF* seguido de - y en número del requisito. Por ejemplo, para referenciar al tercer requisito (*El sistema solicitará el permiso de los usuarios para tomar datos sobre su ubicación.*) se empleará el identificador *RNF-03*.

4.2. Casos de uso

Estos son los casos de uso:

1. Crear una cuenta.
2. Inciar sesión.
3. Eliminar cuenta.
4. Modificar información personal de la cuenta.
5. Modificar la información de preferencias de la cuenta.
6. Modificar la información de las ayudas que necesita o puede ofrecer.

7. Ocultar cierta información del perfil.
8. Tomar datos de la ruta del usuario.
9. Ver rutas.
10. Eliminar ruta.
11. Solicitar un compañero de ruta.
12. Ver el perfil de un compañero.
13. Crear un viaje con un compañero.
14. Ver viajes.
15. Marcar un viaje como visto.
16. Aceptar un viaje compartido.
17. Valorar un viaje.
18. Ver conversaciones.
19. Ver una conversación.
20. Responder a una conversación.
21. Crear una nueva conversación.

A continuación se detalla cada uno de ellos.

Identificador	CU01
Nombre	Crear una cuenta.
Actores	Usuario.
Precondiciones	La aplicación debe estar cargada e iniciada, mostrando la pantalla de inicio de sesión.
Postcondiciones	Se crea una cuenta en el sistema para el usuario o este es notificado de un error.
Descripción	El usuario crea una cuenta en el sistema.
Escenario de éxito	
1. El usuario selecciona *¿No tienes una cuenta aún?, crea una aquí.* 2. El sistema muestra el formulario de creación de nueva cuenta. 3. El usuario rellena el campo *correo electrónico* con una dirección válida. 4. El sistema muestra el campo *correo electrónico* con la información introducida por el usuario. 5. El usuario rellena el campo contraseña, introduciendo una contraseña válida, es decir, una que contenga al menos 8 caracteres, una letra minúscula y otra mayúscula. 6. El sistema muestra la contraseña en forma de caracteres ocultos (*). 7. El usuario selecciona el botón *Crear cuenta.* 8. El sistema crea una cuenta al usuario y lo redirige a la página principal.	
Escenarios alternativos	

Correo no válido:

- 3. El usuario rellena el campo correo electrónico con un correo inválido.
- 4. El sistema le indica al usuario que el correo no es válido.
- 5. Continúa con el paso 1 del escenario de éxito.

Contraseña no válida:

- 5. El usuario rellena el campo *contraseña*, introduciendo una contraseña no válida.
- 6. El sistema notifica al usuario que la contraseña introducida no es segura y destaca los criterios mencionados anteriormente.
- 7. Continúa con el paso 3 del escenario de éxito.

Orden variado:

- Este caso de uso permite que los pasos 3 y 4, y 4 y 5 cambien su orden.

Error en la creación de la cuenta:

- 8. El sistema no consigue crear la cuenta al usuario, le comunica lo ocurrido y le muestra la página de inicio de sesión de nuevo.
- 9. Continúa con el paso 1 del escenario de éxito.

Tabla 1: CU01 - Crear una cuenta

Identificador	CU02
Nombre	Iniciar sesión.
Actores	Usuario.
Precondiciones	La aplicación debe estar cargada e iniciada, mostrando la pantalla de inicio de sesión.
Postcondiciones	El usuario inicia sesión o recibe una notificación de error.
Descripción	El usuario inicia sesión en el sistema.
Escenario de éxito	
1. El usuario introduce en *correo electrónico* el correo que emplea en el sistema. 2. El sistema muestra el correo introducido. 3. El usuario introduce en contraseña la *contraseña* que emplea en el sistema. 4. El sistema muestra la contraseña en forma de caracteres ocultos (*). 5. El usuario selecciona *Iniciar sesión.* 6. El sistema verifica los datos introducidos, carga los datos del usuario y le muestra la pantalla principal.	
Escenarios alternativos	

Credenciales inválidas

- 1. El usuario introduce un correo electrónico y una contraseña que no se corresponden con ninguna cuenta del sistema.
- 2. El sistema notifica al usuario que el usuario o la contraseña introducidos son inválidos.
- 3. Continúa con el paso 1 del escenario de éxito.

Orden variado:

- Este caso de uso permite que los pasos 1 y 2, y 3 y 4 cambien su orden.

Error en el inicio de sesión:

- 6. El sistema no consigue iniciar la sesión del usuario, le comunica lo ocurrido y le muestra la página de inicio de sesión de nuevo.
- 7. Continúa con el paso 1 del escenario de éxito.

Tabla 2: CU02 - Iniciar sesión

Identificador	CU03
Nombre	Eliminar cuenta.
Actores	Usuario.
Precondiciones	La aplicación debe estar cargada e iniciada y la sesión del usuario debe estar activa. El sistema muestra la pantalla de *Otros*, de la sección de perfil.
Postcondiciones	El usuario elimina su cuenta o recibe una notificación de error.
Descripción	El usuario decide eliminar su cuenta en el sistema.
Escenario de éxito	
1. El usuario selecciona *Eliminar cuenta*. 2. El sistema muestra un mensaje de confirmación al usuario. 3. El usuario selecciona Aceptar. 4. El sistema borra los datos del usuario del sistema y cierra la aplicación.	
Escenarios alternativos	

Rechazo de la confirmación:

- 3. El usuario selecciona Cancelar.
- 4. El sistema cierra el mensaje.

Error en la eliminación de la cuenta:

- 4. El sistema no logra eliminar los datos de la cuenta del usuario y le muestra un mensaje.
- 5. El usuario acepta el mensaje.
- 6. El sistema cierra el mensaje.

Tabla 3: CU03 - Eliminar cuenta

Identificador	CU04
Nombre	Modificar información personal de la cuenta
Actores	Usuario.
Precondiciones	La aplicación debe estar cargada e iniciada y la sesión del usuario debe estar activa. El sistema muestra la pantalla de menú del perfil.
Postcondiciones	El usuario modifica correctamente sus datos personales o recibe una notificación de error.
Descripción	El usuario modifica su información personal: nombre completo, foto de perfil, correo electrónico de contacto, número de teléfono, cuenta de *Telegram* o descripción.
Escenario de éxito	
1. El usuario selecciona *Información Personal*. 2. El sistema muestra al usuario todos los campos citados anteriormente con los valores que tiene almacenados en ese momento. 3. El usuario modifica los datos que crea conveniente. 4. El sistema muestra los datos actualizados tras cada modificación. 5. El usuario selecciona *Guardar datos*. 6. El sistema almacena los nuevos datos y muestra la pantalla de perfil al usuario.	
Escenarios alternativos	

Error en la modificación de datos ▪ 6. El sistema no logra almacenar los nuevos datos, y muestra un mensaje de error al usuario. ▪ 7. El usuario acepta el mensaje. ▪ 8. El sistema cierra el mensaje.

Tabla 4: CU04 - Modificar información personal de la cuenta

Identificador	CU05
Nombre	Modificar la información de preferencias de la cuenta.
Actores	Usuario.
Precondiciones	La aplicación debe estar cargada e iniciada y la sesión del usuario debe estar activa. El sistema muestra la pantalla de menú del perfil.
Postcondiciones	El usuario modifica correctamente sus preferencias o recibe una notificación de error.
Descripción	El usuario modifica sus preferencias: si permite que la aplicación tome los datos de su posición, si está dispuesto a ser conductor en un viaje compartido, o sólo pasajero o cuánto desea desviarse de su trayecto principal para recoger a alguien.
Escenario de éxito	
1. El usuario selecciona *Preferencias*. 2. El sistema muestra al usuario todos los campos citados anteriormente con los valores que tiene almacenados actualmente. 3. El usuario modifica los datos que crea conveniente. 4. El sistema muestra los datos modificados tras cada modificación. 5. El usuario selecciona *Guardar datos*. 6. El sistema almacena los nuevos datos y muestra la pantalla de perfil al usuario.	
Escenarios alternativos	

Error en la modificación de datos:

- 6. El sistema no logra almacenar los nuevos datos, y muestra un mensaje de error al usuario.
- 7. El usuario acepta el mensaje.
- 8. El sistema cierra el mensaje.

Tabla 5: CU05 - Modificar la información de preferencias de la cuenta

Identificador	CU06
Nombre	Modificar la información de las ayudas que necesita o puede ofrecer.
Actores	Usuario
Precondiciones	La aplicación debe estar cargada e iniciada y la sesión del usuario debe estar activa. El sistema muestra la pantalla de menú del perfil.
Postcondiciones	El usuario modifica correctamente la información sobre sus ayudas o recibe una notificación de error.
Descripción	El usuario modifica la información relativa a las ayudas que necesita y puede ofrecer: si puede ayudar a personas con movilidad reducida, o necesita ayuda por tener movilidad reducida; si está dispuesto a adaptar la música de su vehículo, o si necesita de ello; si está dispuesto a adaptar el aire acondicionado del vehículo, o necesita de ello; el espacio que tiene disponible en el maletero, o el que necesita; y una descripción de las ayudas que puede ofrecer o necesita.
Escenario de éxito	
1. El usuario selecciona *Ayudas*. 2. El sistema muestra al usuario todos los campos citados anteriormente con los valores que tiene almacenados actualmente. 3. El usuario modifica los datos que crea conveniente. 4. El sistema muestra los datos modificados tras cada modificación. 5. El usuario selecciona *Guardar datos*. 6. El sistema almacena los nuevos datos y muestra la pantalla de perfil al usuario.	

Escenarios alternativos
Error en la modificación de datos: - 6. El sistema no logra almacenar los nuevos datos, y muestra un mensaje de error al usuario. - 7. El usuario acepta el mensaje. - 8. El sistema cierra el mensaje.

Tabla 6: CU06 - Modificar la información de las ayudas que necesita o puede ofrecer

Identificador	CU07
Nombre	Ocultar cierta información del perfil.
Actores	Usuario.
Precondiciones	La aplicación debe estar cargada e iniciada y la sesión del usuario debe estar activa. El sistema muestra la pantalla de menú del perfil.
Postcondiciones	El usuario oculta correctamente los datos o recibe una notificación de error.
Descripción	El usuario selecciona qué datos quiere que sean visibles y cuales no. Estos son los datos aceptados: nombre completo, foto de perfil, correo electrónico, descripción personal; correo de contacto, número de teléfono, cuenta de *Telegram*; si está dispuesto a ayudar a gente con movilidad reducida, a adaptar su música, el espacio que tiene disponible en el maletero, y la descripción de las ayudas que puede ofrecer; si necesita ayuda por movilidad reducida, que le adapten la música, el aire acondicionado, el espacio de maletero que necesita, y su descripción de ayudas; si está dispuesto a conducir y, de ser así, cuánta distancia está dispuesto a desviarse.
Escenario de éxito	

1. El usuario selecciona *Ocultar Información*.
2. El sistema muestra al usuario una pantalla con todos los datos que guarda sobre el usuario, con un botón para ocultar el campo para cada uno de ellos.
3. El usuario selecciona *Ocultar campo*con todos los campos que desee ocultar.
4. El sistema muestra los datos modificados tras cada modificación.
5. El usuario selecciona *Guardar*.
6. El sistema almacena los nuevos datos y muestra la pantalla de perfil al usuario.

Escenarios alternativos

Error en la modificación de datos:

- 6. El sistema no logra almacenar los nuevos datos, y muestra un mensaje de error al usuario.
- 7. El usuario acepta el mensaje.
- 8. El sistema cierra el mensaje.

Tabla 7: CU07 - Ocultar cierta información del perfil

Identificador	CU08
Nombre	Tomar datos de la ruta del usuario.
Actores	Sistema.
Precondiciones	La aplicación debe estar cargada e iniciada y la sesión del usuario debe estar activa. El usuario debe haber dado permiso al sistema para acceder a su ubicación, tanto a la aplicación, como en la configuración de su perfil.
Postcondiciones	El sistema almacena información sobre la ruta que sigue el usuario.
Descripción	El sistema toma datos de la ubicación del usuario para recrear la ruta que está siguiendo y poder recomendarle potenciales compañeros de viaje. Toma datos cada vez que el usuario se desplaza 100 metros.
Escenario de éxito	
1. El sistema toma datos de la ubicación del usuario y los envía al servidor para que los trate, generando rutas.	
Escenarios alternativos	
Error en la captura o el almacenamiento de los datos: ▪ 1. El sistema falla al captar o almacenar los datos, y ningún dato es almacenado.	

Tabla 8: CU08 - Tomar datos de la ruta del usuario

Identificador	CU09
Nombre	Ver rutas.
Actores	Usuario.
Precondiciones	La aplicación debe estar cargada e iniciada y la sesión del usuario debe estar activa. El sistema muestra el menú de rutas y viajes.
Postcondiciones	El usuario logra visualizar las rutas que el sistema ha captado, o recibe un mensaje de error.
Descripción	El usuario accede al listado de rutas que el sistema tiene sobre él.
Escenario de éxito	
1. El usuario selecciona *Mis Rutas*. 2. El sistema muestra al usuario un listado de las rutas que tiene almacenadas sobre él, indicando el recorrido y la hora de salida y de llegada, junto a un icono que permite su eliminación.	
Escenarios alternativos	
Error en la muestra de los datos: ▪ 2. El sistema no logra obtener las rutas del usuario y muestra un mensaje de error. ▪ 3. El usuario selecciona *Aceptar*. ▪ 4. El sistema cierra el mensaje y muestra al usuario la página principal.	

Tabla 9: CU09 - Ver rutas

Identificador	CU10
Nombre	Eliminar ruta.
Actores	Usuario.
Precondiciones	La aplicación debe estar cargada e iniciada y la sesión del usuario debe estar activa. El sistema muestra el menú de rutas y viajes.
Postcondiciones	El usuario visualiza las rutas que el sistema ha captado y elimina la que cree conveniente, o recibe un mensaje de error.
Descripción	El usuario accede al listado de rutas que el sistema tiene sobre él y elimina una de ellas.
Escenario de éxito	
1. El usuario selecciona Mis Rutas. 2. El sistema muestra al usuario un listado de las rutas que tiene almacenadas sobre él, indicando el recorrido y la hora de salida y de llegada, junto a un botón que permite su eliminación. 3. El usuario selecciona el botón de eliminar de una de las rutas. 4. El sistema muestra un mensaje de confirmación al usuario. 5. El usuario selecciona Aceptar. 6. El sistema elimina la ruta del usuario y le muestra la página de lista de rutas actualizada.	
Escenarios alternativos	

El usuario no tiene rutas:

- 2. El sistema muestra al usuario un mensaje indicando que no tiene rutas almacenadas.

Error en la muestra de los datos:

- 2. El sistema no logra obtener las rutas del usuario y muestra un mensaje de error.
- 3. El usuario selecciona *Aceptar*.
- 4. El sistema cierra el mensaje y muestra al usuario la página principal.

El usuario cancela la eliminación:

- 5. El usuario selecciona *Cancelar*.
- 6. El sistema cierra el mensaje.

Error en la eliminación de la ruta:

- 6. El sistema no logra eliminar la ruta que el usuario seleccionó y le muestra un mensaje de error.
- 7. El usuario selecciona *Aceptar*.

Tabla 10: CU10 - Eliminar ruta

Identificador	CU11
Nombre	Solicitar un compañero de ruta.
Actores	Usuario.
Precondiciones	La aplicación debe estar cargada e iniciada y la sesión del usuario debe estar activa. El sistema muestra el menú de rutas y viajes.
Postcondiciones	El usuario recibe una lista de posibles usuarios con los que compartir vehículo en sus trayectos más comunes, o recibe un mensaje de error.
Descripción	El usuario solicita una lista de usuarios que realicen rutas similares a la suya y estén dispuestos a compartir vehículo.
Escenario de éxito	
1. El usuario selecciona *Buscar compañero.* 2. El sistema analiza los datos del usuario y le muestra un listado de usuarios que podrían estar dispuestos a compartir coche con el usuario, facilitando acceso a su perfil e información de contacto, así como la ruta a compartir.	
Escenarios alternativos	

El sistema no encuentra usuario que estén dispuestos a compartir vehículo con el usuario:

- 2. El sistema muestra un mensaje al usuario indicando que no hay usuarios disponibles para compartir vehículo con él, sugiriéndole que lo intente en otro momento.

Error en la búsqueda de compañeros:

- 2. El sistema no logra obtener las recomendaciones de compañeros para el usuario y le muestra un mensaje de error.
- 3. El usuario acepta el mensaje.
- 4. El sistema cierra el mensaje.

Tabla 11: CU11 - Solicitar un compañero de ruta

Identificador	CU12
Nombre	Ver el perfil de un compañero.
Actores	Usuario.
Precondiciones	El usuario ha completado el caso de uso *CU11* con éxito.
Postcondiciones	El usuario observará el perfil de uno de sus posibles compañeros de viaje. Solamente se mostrarán los campos que el propietario del perfil no haya marcado como ocultos.
Descripción	El usuario selecciona el perfil de uno de los compañeros de ruta que se les muestra y ve sus datos.
Escenario de éxito	
1. El usuario selecciona el nombre del compañero cuyo perfil está interesado en visualizar. 2. El sistema muestra al usuario en una nueva pantalla toda la información del perfil del usuario seleccionado, incluyendo su valoración, salvo la que este haya marcado como oculta.	
Escenarios alternativos	
Error al extraer los datos del perfil. ▪ 2. El sistema no consigue obtener los datos del perfil del compañero y muestra un mensaje de error. ▪ 3. El usuario acepta el mensaje. ▪ 4. El sistema cierra el mensaje.	

Tabla 12: CU12 - Ver el perfil de un compañero

Identificador	CU13
Nombre	Crear un viaje con un compañero.
Actores	Usuario.
Precondiciones	El usuario ha completado el caso de uso *CU11* con éxito.
Postcondiciones	El usuario crea un nuevo viaje entre su compañero y él.
Descripción	El usuario crea un viaje a partir de una de las recomendaciones recibidas, y establece si desea conducir, la hora de salida y un mensaje opcional.
Escenario de éxito	
1. El usuario selecciona el botón *Solicitar viaje* de la proposición de viaje en la que esté interesado. 2. El sistema muestra al usuario una pantalla con un formulario a rellenar. El formulario contiene si desea o no conducir, la fecha de salida del viaje, y un campo de descripción. 3. El usuario cumplimenta el formulario de forma correcta y selecciona *Crear viaje*. 4. El sistema crea satisfactoriamente el viaje y muestra un mensaje de éxito al usuario. 5. El usuario acepta el mensaje.	
Escenarios alternativos	

Error al crear el viaje: ▪ 4. El sistema no consigue obtener los datos del perfil del compañero y muestra un mensaje de error. ▪ 5. El usuario acepta el mensaje. ▪ 6. El sistema cierra el mensaje.

Tabla 13: CU13 - Crear un viaje con un compañero

Identificador	CU14
Nombre	Ver viajes.
Actores	Usuario.
Precondiciones	La aplicación debe estar cargada e iniciada y la sesión del usuario debe estar activa. El sistema muestra el menú de rutas y viajes.
Postcondiciones	El usuario visualiza correctamente sus viajes o recibe un mensaje de error.
Descripción	El usuario visualiza los viajes que tiene planeados.
Escenario de éxito	
1. El usuario selecciona la opción *Mis viajes*. 2. El sistema muestra al usuario un listado de sus viajes, indicando para cada uno si lo ha marcado como visto, con quién comparte vehículo, la fecha y hora de inicio del viaje, quién ha aceptado el viaje, la valoración que le has asignado al viaje y la ruta a realizar.	
Escenarios alternativos	

El usuario no tiene viajes planeados:

- 2. El usuario no tiene viajes planeados, y el sistema muestra un mensaje que lo indica.

Error al cargar los viajes:

- 2. El sistema no consigue cargar los viajes del usuario y muestra un mensaje de error.
- 3. El usuario acepta el mensaje.
- 4. El sistema cierra el mensaje.

Tabla 14: CU14 - Ver viajes

Identificador	CU155
Nombre	Marcar un viaje como visto.
Actores	Usuario.
Precondiciones	El usuario ha completado con éxito el caso de uso *CU14*, y tiene al menos un viaje marcado como no leído.
Postcondiciones	El usuario marca el viaje como leído o recibe un mensaje de error.
Descripción	El usuario selecciona un viaje y lo marca como visto.
Escenario de éxito	
1. El usuario toca uno de los viajes que aparecen marcados como nuevos. 2. El sistema marca el viaje como visto y elimina la marca.	
Escenarios alternativos	
Error al marcar el viaje como visto: ▪ 2. El sistema no consigue marcar el viaje como visto y muestra al usuario un mensaje de error. ▪ 3. El usuario acepta el mensaje. ▪ 4. El sistema cierra el mensaje.	

Tabla 15: CU15 - Marcar un viaje como visto

Identificador	CU16
Nombre	Aceptar un viaje compartido.
Actores	Usuario.
Precondiciones	El usuario ha terminado el caso de uso *CU14* con éxito y tiene al menos un viaje propuesto sin aceptar.
Postcondiciones	El usuario marca el viaje como aceptado o recibe un mensaje de error.
Descripción	El usuario marca uno de sus viajes sin aceptar como aceptado.
Escenario de éxito	
1. El usuario selecciona *Aceptar viaje* en uno de los viajes no aceptado por él aún. 2. El sistema marca ese viaje como aceptado por parte del usuario y actualiza la información de la lista.	
Escenarios alternativos	
Error al marcar el viaje como aceptado. ▪ 2. El sistema no consigue marcar el viaje como aceptado por parte del usuario y muestra al usuario un mensaje de error. ▪ 3. El usuario acepta el mensaje. ▪ 4. El sistema cierra el mensaje.	

Tabla 16: CU16 - Aceptar un viaje compartido

Identificador	CU17
Nombre	Valorar un viaje.
Actores	Usuario.
Precondiciones	El usuario ha completado con éxito el caso de uso *CU14* y tiene al menos un viaje sin valorar.
Postcondiciones	El usuario añade su valoración al viaje o recibe un mensaje de error.
Descripción	El usuario valora uno de sus viajes compartidos que previamente no estaba valorado.
Escenario de éxito	
1. El usuario selecciona el número de *estrellas* de valoración que desea dar a la experiencia del viaje, indicando un mínimo de una estrella, y un máximo de cinco. 2. El sistema almacena el nuevo dato y actualiza los valores de la pantalla de viajes.	
Escenarios alternativos	
Error al añadir la valoración: ▪ 2. El sistema no consigue añadir la nueva valoración y muestra un mensaje de error al usuario. ▪ 3. El usuario acepta el mensaje. ▪ 4. El sistema cierra el mensaje.	

Tabla 17: CU17 - Valorar un viaje

Identificador	CU18
Nombre	Ver conversaciones.
Actores	Usuario.
Precondiciones	La aplicación debe estar cargada e iniciada y la sesión del usuario debe estar activa. El sistema muestra la pantalla de menú principal de la aplicación.
Postcondiciones	El usuario visualiza un extracto de sus conversaciones o recibe un mensaje de error.
Descripción	El usuario decide visualizar un extractos de sus conversaciones a través del chat interno de la aplicación. Cada conversación incluye el nombre del otro usuario y un resumen de los tres últimos mensajes que intercambiaron.
Escenario de éxito	
1. El usuario selecciona *Mis chats*. 2. El sistema muestra al usuario un listado de sus conversaciones con otros usuarios.	
Escenarios alternativos	

El usuario no tiene ninguna conversación

- 2. El usuario no tiene niguna conversación, y el sistema muestra un mensaje indicándolo.

Error al extraer los datos de las conversaciones:

- 2. El sistema no consigue extraer los datos de las conversaciones del usuario y le muestra un mensaje de error.
- 3. El usuario acepta el mensaje.
- 4. El sistema cierra el mensaje.

Tabla 18: CU18 - Ver conversaciones

Identificador	CU19
Nombre	Ver una conversación.
Actores	Usuario.
Precondiciones	El usuario debe haber completado con éxito el caso de uso *CU18*, y tener al menos una conversación.
Postcondiciones	El usuario visualiza una de sus conversaciones con otro usuario.
Descripción	El usuario selecciona una de las conversaciones y ve todos los mensajes que intercambió con el usuario. Se mostrarán los mensajes que ha enviado el usuario en alineados en el margen derecho, y los recibido en el izquierdos, ambos destacados en distintos colores. Los mensajes se mostrarán ordenados de más antiguos a más recientes. La aplicación hará *scroll down* de forma automática. Esta acción en sí mismo marcará los mensajes como leídos para la próxima vez que se consulten.
Escenario de éxito	
1. El usuario selecciona una de sus conversaciones. 2. El sistema muestra al usuario todos los mensajes que ha intercambiado con el otro usuario.	
Escenarios alternativos	

Error al cargar los mensajes: ▪ 2. El sistema no consigue extraer los mensajes de la conversación y muestra un mensaje de error al usuario. ▪ 3. El usuario acepta el mensaje. ▪ 4. El sistema cierra el mensaje.

Tabla 19: CU19 - Ver una conversación

Identificador	CU20
Nombre	Responder a una conversación.
Actores	Usuario.
Precondiciones	El usuario ha completado con éxito el caso de uso *CU19*.
Postcondiciones	El usuario envía un nuevo mensaje al otro usuario, o recibe un mensaje de error.
Descripción	El usuario responde al otro usuario en la conversación. El nuevo mensaje se marca como leído para el emisor y como no leído para el receptor.
Escenario de éxito	
1. El usuario escribe el mensaje que desee enviar en la barra inferior y pulsa la tecla *enter*. 2. El sistema envía el mensaje y actualiza el listado de mensajes.	
Escenarios alternativos	
Error al enviar el mensaje: ▪ 2. El sistema no consigue enviar el mensaje y muestra un mensaje de error al usuario. ▪ 3. El usuario acepta el mensaje. ▪ 4. El sistema cierra el mensaje.	

Tabla 20: CU20 - Responder a una conversación

Identificador	CU21
Nombre	Crear una nueva conversación.
Actores	Usuario.
Precondiciones	El usuario ha completado con éxito el caso de uso *CU12*.
Postcondiciones	El usuario inicia una nueva conversación con otro usuario, o le envía un nuevo mensaje, o recibe un mensaje de error.
Descripción	El usuario envía un nuevo mensaje a otro usuario desde su perfil. Si ya existía una conversación entre ambos usuarios, el nuevo mensaje se añadirá a ella y, si no existía, se creará una nueva conversación entre ambos con el nuevo mensaje como primer mensaje.
Escenario de éxito	
1. El usuario selecciona el botón *Enviar mensaje a "Nombre del usuario"*. 2. El sistema muestra una ventana emergente con una entrada para texto. 3. El usuario escribe el mensaje que desee enviar y pulsa la tecla *enter*. 4. El sistema envía el mensaje y muestra un mensaje de éxito al usuario.	
Escenarios alternativos	
Error al enviar el mensaje: ▪ 2. El sistema no consigue enviar el mensaje y muestra un mensaje de error al usuario. ▪ 3. El usuario acepta el mensaje. ▪ 4. El sistema cierra el mensaje.	

Tabla 21: CU21 - Crear una nueva conversación

4.3. Matriz de trazabilidad

Esta es la matriz de trazabilidad del proyecto, que relaciona qué requisitos cubre cada caso de uso.

	RF01	RF02	RF03	RF04	RF05	RF06	RF07	RF08	RF09	RF10	RF11	RF12	RF13	RF14	RF15	RF16	RF16	RF18
CU01	X																	
CU02		X																
CU03				X														
CU04			X															
CU05			X			X	X											
CU06			X															
CU07			X															
CU08					X													
CU09								X										
CU10									X									
CU11										X								
CU12											X							
CU13												X						
CU14													X					
CU15													X					
CU16														X				
CU17															X			
CU18																X		
CU19																	X	
CU20																		X
CU21																		X

5

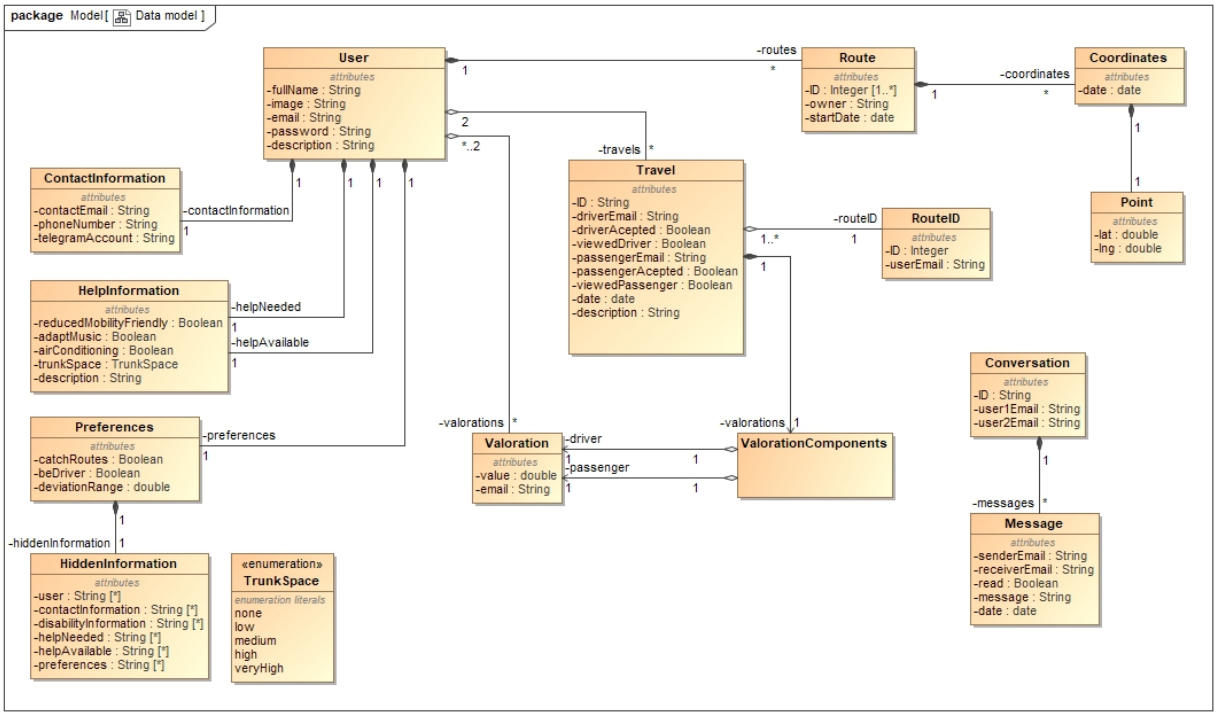
Modelado y Diseño

En este capítulo, analizaremos en profundidad el modelo de datos de la aplicación, y el proceso seguido hasta alcanzar el diseño final de la misma.

Go Language se trata de un lenguaje de programación que, si bien brinda la posibilidad de crear estructuras sobre las que se pueden definir métodos, emulando así el comportamiento de una clase, no suele ser muy empleado. De hecho, solamente se ha utilizado esa funcionalidad para confeccionar los distintos controladores y gestores de almacenamiento; mientras que la lógica de la aplicación se ha realizado empleando funciones.

5.1. Modelo de datos

Este es el modelo de datos que emplea la aplicación:



Para comprender mejor el modelo de datos de la aplicación, es necesario recordar que se

emplea una base de datos NoSQL (*Firestore Database*), por lo que algunas "*claves externas*" aparecen en forma de atributo. Esto puede apreciarse, por ejemplo, en que la entidad *Travel* tiene un *driverEmail* y un *passengerEmail* de tipo cadena de caracteres, en lugar de una referencia a un usuario; o en que las entidades *Conversation* y *Message* tengan atributos que representan correos de usuarios y tampoco sean claves externas, sino cadenas de caracteres. Esta es una práctica muy común en las bases de datos *NoSQL* que se emplea para optimizar las lecturas.

Como se puede observar, existen un total de quince entidades, de las cuales podemos considerar cuatro principales: *User*, *Route*, *Travel* y *Conversation*.

5.1.1. Entidad User

La entidad *User* contiene la información de los usuarios de la aplicación. Cada uno de ellos cuenta con

1. Un nombre completo
2. Una foto de perfil. En este caso se almacena la URL al recurso almacenado en *Cloudinary*.
3. Su contraseña. Por supuesto, en todo momento se almacena y viaja encriptada en SHA256.
4. Una descripción.
5. Su información de contacto.
6. La información de la ayuda que necesita.
7. La de la ayuda que puede proporcionar.
8. Sus preferencias.

La información de contacto simplemente contiene un email, un número de teléfono y una cuenta de *Telegram*.

La información tanto de ayuda que puede proporcionar como que necesita, sigue el mismo formato: si puede / necesita ayuda por tener movilidad reducida, si puede / necesita que se adapte la música del vehículo, si puede / necesita que adapten el aire acondicionado, el espacio disponible en el maletero que tiene / necesita (según las categorías: NADA, POCO, MEDIO, ALTO y MUY ALTO) y algún tipo de descripción adicional. Toda esta información se

emplea como vimos anteriormente para comprobar si dos usuarios son o no compatibles para compartir vehículo.

La información de preferencias simplemente se trata de un *flag* que indica si el usuario está dispuesto a compartir su ubicación, otro que indica si está dispuesto a hacer de conductor en un viaje compartido y la cantidad de kilómetros que está dispuesto a desviarse para recoger a su compañero, en caso de ser conductor. Por supuesto, en el momento en el que un usuario indica que no está dispuesto a compartir su ubicación, el sistema deja de enviarla. Esta información también contiene los datos que el usuario desea mantener oculto. Esta fue una de las decisiones más complejas en el desarrollo del trabajo, pues necesitábamos poder ocultar la información del usuario de cara al resto de usuarios, pero poder tenerla en cuenta para calcular la compatibilidad entre usuarios. Lo más difícil fue dar con una estructura de datos que lo permitiese.

En versiones anteriores, se probó con una serie de referencias, pero resultó demasiado complejo.

También se intentó hacer que cada atributo de la estructura *User* fuese de un tipo que conteniese el valor, y un *flag* que indicase si está visible o no, pero esta solución era demasiado compleja de implementar en *Go Language*, ya que este lenguaje no cuenta con *estructuras genéricas*. Por ello, finalmente se optó por una serie de *arrays* de cadenas de caracteres que contuviesen los nombres de los atributos que se desea ocultar. Por supuesto, esta información es tratada en el backend de la aplicación, de forma que el frontend sólo trabaja con los valores del perfil que se encuentran visibles, no recibiendo los ocultos en ningún momento.

5.1.2. Entidad Route

La entidad *Route* es la encargada de contener la información de las distintas rutas que realizan los usuarios.

Esta entidad cuenta con una colección de identificadores, el correo electrónico de su propietario, la fecha de inicio, y un array de coordenadas.

Puede resultar extraño que una entidad tenga un *array* de identificadores, pero esto es así porque, al *unificar rutas*, las rutas pueden eliminarse, pero seguir siendo referenciadas por otras entidades, como por ejemplo un *Travel*. Para solucionar este problema, barajamos dos opciones:

1. Editar los identificadores de todos los viajes que referenciasen a la ruta eliminada. Esta opción eliminaba duplicaciones y resultaba bastante evidente, pero era extremadamente ineficiente, ya que los datos de las *rutas* se almacenan en una colección distinta a la de los *viajes*, por lo que habría que recorrer la colección de *viajes completa*.

2. Es por ello, que nos decantamos por esta segunda opción: añadir un array de identificadores a las rutas. Como las rutas pertenecen a la colección de los usuarios, es decir, son un atributo de estos en la base de datos, era mucho más rápido añadir identificadores que recorrer la colección de *Travels* al completo. Además, no genera ninguna inconsistencia, porque simplemente pasamos de comparar identificadores, a comprobar si un cierto ID pertenece o no a una colección de identificadores.

Por otra parte, cada coordenada cuenta con una fecha, y un punto, que contienen la latitud y la longitud geográfica de la coordenada.

5.1.3. Entidad Travel

La entidad *Travel* tiene un identificador, la información del correo electrónico, si ha aceptado el viaje, y si ha visto la ruta tanto el conductor como el pasajero, la fecha prevista para el viaje, una descripción textual, el identificador de la ruta y las valoraciones de los usuarios.

El identificador de la ruta no se trata simplemente de un número o de una cadena de caracteres, sino de una estructura que cuenta con un ID numérico más un correo electrónico de un usuario. Esto nos permite solucionar un problema al unificar rutas, que veremos en detalle más adelante

La valoración de los usuarios se almacena gracias a dos estructuras que cuentan con dos atributos cada una: el valor de la puntuación y el email de quién realizó el voto.

5.1.4. Entidad Conversation

La entidad *Conversation* representa las conversaciones entre usuarios. Esta contiene un identificador, los correos electrónicos de los usuarios que intercambian mensajes y una lista de mensajes que han intercambiado.

Dichos mensajes contienen el correo del emisor y el del receptos, si el receptor ha leído el mensaje, la fecha y hora en la que se envió y el texto del mensaje en sí.

Como se comentó anteriormente, aparecen correos electrónicos de usuarios definidos y almacenados como cadenas de caracteres y no como claves externas, como suele ocurrir en las bases de datos NoSQL.

5.2. Organización NoSQL

Es importante destacar que los datos mostrados anteriormente no se almacenan tal como se muestran en el diagrama, si no que se organizan en tres colecciones: la de *Users*, la de *Travels* y la de *Conversations*.

La colección de *Users* contiene los datos del propio usuario y de sus rutas; mientras que la de *Travels* contiene simplemente a los viajes con su información adicional.

La colección de *Conversations* contiene las distintas conversaciones que los usuarios mantienen a través del chat de la aplicación.

En un principio, se tenían planeado emplear cuatro colecciones *Users*, *Travels*, *Routes* y *Conversations*, pero dado que cada ruta pertenece a uno y sólo un usuario, esta última colección no era necesaria.

Esta disposición es la que provoca, por ejemplo, que el identificador de las rutas para los viajes sea una estructura con un número y un correo electrónico, ya que los identificadores de las rutas son relativos para cada usuario, siendo que el identificador de una ruta es, simplemente, la posición que ocupa en el array de rutas del usuario esa ruta concreta.

También cabe destacar la duplicación de las valoraciones, y es que las valoraciones que los usuarios realizan sobre un viaje se almacena tanto en el viaje como en el usuario. Esto es así porque es mucho más rápido leer, por una parte la valoración de un viaje, y por otra la de los usuarios, que tener que recorrer una de las colecciones buscando el dato. En comparación con esto, el tener que escribir la valoración en dos colecciones distintas se trata de un mal menor.

Por último, cabe mencionar que en un principio la colección *Conversations* no iba a ser creada, y se iba a optar por añadir los mensajes que los usuarios compartiesen junto a su información en la colección *Users*, bien en un sólo usuario, o bien de forma duplicada en cada uno de los usuarios. Este enfoque es del todo correcto y funcional, y daría mejores resultados en las lecturas de mensajes, pero como prevemos que el chat será muy usado y que se realizarán también muchas escrituras, preferimos separar las conversaciones en su propia colección.

5.3. Proceso de diseño

En esta sección se comentará en detalle tanto el proceso seguido hasta alcanzar el diseño actual de la aplicación, como el funcionamiento de la navegación de la misma.

5.3.1. Diseño

En cuanto al proceso de diseño, decidí retrasarlo en gran parte hasta las últimas etapas del desarrollo. Esta decisión se tomó de forma involuntaria, ya que durante las primeras etapas del desarrollo, me centré en dejar claro qué producto se pretendía obtener y, posteriormente, me focalicé en desarrollar la idea, dedicando la mayor parte del tiempo al desarrollo de los algoritmos principales de la aplicación (explicados más adelante) y a asegurarme de que eran efectivos y eficientes, así como en crear una aplicación móvil que simplemente permitiese el uso de estos algoritmos mediante llamadas a un servidor.

Por ello, podemos clasificar el proceso de diseño en dos fases.

Una primera fase en la que se aclararon cuáles serían las pantallas que la aplicación necesitaba, llegando a la conclusión de que necesitaríamos un total de 14 pantallas:

1. Inicio de sesión
2. Creación de nueva cuenta
3. Pantalla principal
4. Menú del perfil
5. Información personal
6. Preferencias
7. Ocultar información
8. Ayudas
9. Otros
10. Menú de viajes y rutas

11. Gestionar rutas
12. Buscar compañero
13. Crear nuevo viaje
14. Gestionar viajes

Esta primera actividad se realizó antes de comenzar con el desarrollo del código, pese a sufrir algunas modificaciones como, por ejemplo, la no existencia de pantallas de menú en las primeras etapas.

En la segunda fase se realizó una mejora estética total a la aplicación.

En primer lugar, se eligió la plantilla de colores a emplear. Tras analizar varias posibilidades en la web, opté por emplear los siguientes colores:

1. *#EAE7DC* para el color de fondo de la aplicación. Elegí este tono porque no resalta mucho y da una muy buena apariencia suave y agradable al fondo de las pantallas de la aplicación.
2. *#D8C3A5* como color para las cabeceras de las pantallas. Este tono es similar al empleado para el fondo de la aplicación, pero resalta algo más, lo cual lo hace muy adecuado para esta función.
3. *#E98074* como color para los botones de la aplicación. Este tono rojizo destaca lo suficiente como para emplearse perfectamente como fondo para los distintos botones de la aplicación. No es ni demasiado llamativo, lo cual podría parecer como un parche; ni demasiado similar al color de fondo, lo cual podría hacer que el botón pasase desapercibido.
4. *#8E8D8A* para otros elementos a destacar de la aplicación, como por ejemplo el icono que indica el número de viajes que tiene el usuario sin leer. Se eligió este tono porque se necesitaba de uno que destacase sobre el *#E98074* empleado anteriormente, pero que siguiese en la gama de los tonos anteriores. Este tono grisáceo nos pareció adecuado.
5. *#0078FE* y *#DEDEDE* únicamente para los mensajes enviados y recibidos por el usuario a través del chat de la aplicación, respectivamente. Estos colores, si bien no

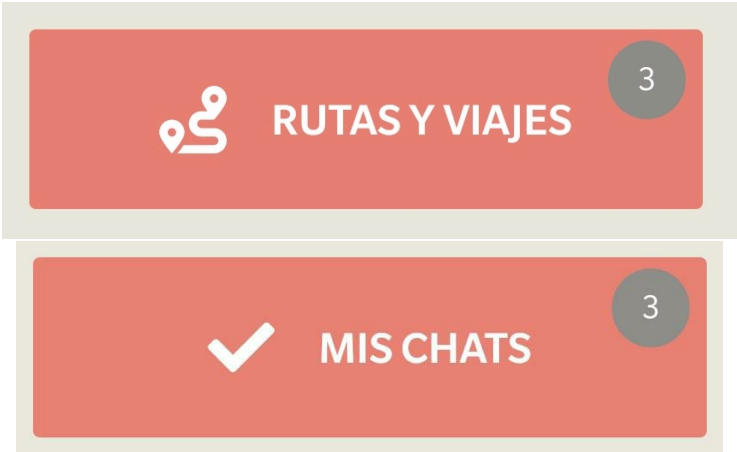
encajan del todo con el resto, son colores clásicos empleados en los chats para identificar mensajes, lo cual hace al chat de la aplicación más intuitivo y sencillo de usar.

Para las letras y los iconos, obtamos por emplear el color negro (*#000000*) si se encontraban sobre un fondo más claro, y el blanco (*#FFFFFF*) en fondos más oscuros.

Por supuesto, los códigos de los colores se encuentran en el fichero principal de la aplicación, recogidos en forma de constante, de forma que si se quisiese cambiar el color de los elementos de la aplicación, bastaría con modificarlos sólo en ese fichero.

La idea detrás de esta gama de colores con tonos tan claros y marrones como fondos, y rojizos para los elementos destacados es transmitir serenidad al usuario. Los colores claros y marrones suelen calmar a las personas, y los tonos rojizos, al no ser demasiado brillantes, hacen destacar los elementos de la aplicación pero sin romper con esa sensación de armonía. Al tratarse de una aplicación en la que buscas compartir vehículos, pensamos que esta gama de colores, con la sensación que la acompaña, transmiten la confianza necesaria para formar una comunidad de conductores.

Por último, se añadieron algunos detalles a la aplicación, como por ejemplo la adición de un icono que muestra el número de viajes que tienes sin ver en tu cuenta, o el número de mensajes sin leer que tiene el usuario:

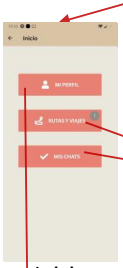


5.3.2. Navegación

La idea principal tras la navegación de la aplicación es que sea lo más sencilla posible. El objetivo es que cualquier usuario pueda desplazarse por todas las pantallas de la aplicación de forma prácticamente intuitiva.

En su versión final, la aplicación cuenta con 17 pantallas distintas, y este es su diagrama de navegación:

Inicio de sesión

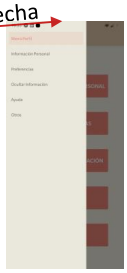


Inicio

Nueva cuenta

Desliza derecha

Principal perfil



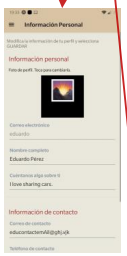
Menú perfil

Desliza derecha

Menú rutas

Principal rutas

Conversaciones



Info personal

Ocultar info

Otros

Ver rutas

Ver viajes

Conversación

Preferencias

Info ayudas

Buscar compañero

Nuevo viaje

Ver perfil

Téngase en cuenta que, para mejorar la legibilidad del diagrama, se han tomado las siguientes decisiones:

1. Todas las pantallas tienen en su esquina superior derecha un botón con un icono en forma de flecha que sirve para volver a la pantalla anterior. La flecha que indica dicha navegación se ha omitido.

2. Desde las pantallas de *Info personal*, *Preferencias*, *Ocultar info*, *Info ayudas* y *Otros*, se puede abrir el *Menú perfil* para facilitar la navegación entre los elementos de esa sección. Las flechas que indican dichas navegaciones se han omitido.

3. Desde las pantallas de *Ver rutas*, *Ver viajes*, *Buscar compañero*, *Crear cuenta* y *Ver perfil*, también se puede abrir el *Menú rutas* para facilitar la navegación entre los elementos de esa sección. Las flechas que indican dichas navegaciones se han omitido.

4. El *Menú perfil* permite navegar a cualquiera de las pantallas de la sección de perfil. Las flechas que indican dichas navegaciones se han omitido.

5. El *Menú rutas* permite navegar a cualquiera de las pantallas de la sección de rutas. Las flechas que indican dichas navegaciones se han omitido.

Como se aprecia en el diagrama, la navegación es bastante sencilla e intuitiva. En general, la aplicación se divide en cinco secciones:

1. Las pantallas de gestión de cuenta: *Inicio de sesión* y *Creación de nueva cuenta*.

2. La pantalla principal, que facilita la navegación entre las secciones.

3. La sección del perfil, que cuenta con las pantallas de *Información personal*, *Preferencias*, *Ocultar información*, *Ayuda* y *Otros*. Además, si deslizas a la derecha en cualquiera de estas pantallas, aparece un menú lateral que facilita la navegación entre las pantallas de esta sección.

4. La sección de rutas, que tiene las pantallas de *Mis rutas*, *Buscar compañero*, *Ver perfil*, *Crear nuevo viaje*, *Ver viajes*. Además, al igual que en la sección anterior, si deslizas hacia la derecha, aparece un menú que facilita la navegación.

5. La sección de *Conversaciones*, compuesta por las pantallas *Conversaciones* y *Conversación*. La primera muestra al usuario una lista de todas sus conversaciones, indicando quién es el otro usuario y los tres últimos mensajes. La pantalla de *Conversación* muestra una conversación en concreto.

Cabe destacar la existencia de los menús laterales que ayudan al usuario a navegar entre las distintas pantallas de cada sección.

6
Desarrollo

En general, la aplicación cuenta con siete funcionalidades diferenciadas, que trabajan en conjunto para lograr la búsqueda automática de compañeros de viaje: la gestión de cuentas, la gestión del perfil, la recolección de información de las rutas, el algoritmo de unificación de rutas, el propio algoritmo de búsqueda de rutas para compartir viajes, la gestión de los viajes y el chat.

Adicionalmente, en esta capítulo también se comentarán las pruebas realizadas sobre la aplicación.

6.1. Gestión de cuentas

La gestión de cuentas engloba tanto la creación de nuevas cuentas como el inicio de sesión.

Para el desarrollo del proyecto se ha intentado mantener este apartado lo más simple posible. Por ello, los usuarios pueden crearse cuentas con un nombre, un correo electrónico y una contraseña segura, es decir, que contenga al menos 8 caracteres, una mayúscula, una minúscula y un número; e iniciar sesión con su correo electrónico y su contraseña.

En cuanto al apartado de seguridad, la contraseña de almacena y transporta siempre encriptada, ya que es la aplicación móvil la que encripta según el proceso *SHA256* la contraseña antes de enviarla al servidor.

Cuando un usuario inicia sesión, se le proporciona un token de autenticación con una fecha de expiración de un día, el cual deben adjuntar en la cabecera HTTP de *Authentication* para que el servidor responda a sus peticiones, en caso contrario, recibirán un mensaje de error.

6.2. Gestión del perfil

La sección de gestión de perfil permite a los usuarios almacenar información sobre ellos que será empleada para comprobar la compatibilidad entre usuarios cuando alguno pretenda buscar un compañero con el que compartir vehículo para realizar una cierta ruta.

La gestión del perfil se divide en cinco apartados: información personal, de preferencias, de necesidades, opciones para ocultar la información y otras opciones.

La gestión de información personal permite al usuario guardar o modificar su foto de perfil, nombre completo, descripción e información de contacto, incluyendo un correo, un número de teléfono y una cuenta Telegram.

Cabe destacar que la base de datos no almacena la foto de perfil en sí, sino un enlace a la foto de perfil que proporciona la herramienta *Cloudinary*, para agilizar las búsquedas. *Cloudinary* se trata de una plataforma que permite que se le envíe una imagen, la almacena y te proporciona una URL desde la que puedes ver la imagen. Todo este proceso se realiza de forma segura mediante HTTPS a través de una clave de mi cuenta personal.

La gestión de preferencias permite al usuario indicar si desea que la aplicación tome o no datos de su localización, si está dispuesto a conducir en un viaje compartido y, de ser así, cuando está dispuesto a desviarse de su ruta para recoger a su compañero.

La gestión de información de ayudas permite al usuario indicar si cuenta con la posibilidad de adaptar su vehículo a otros usuarios, o si lo necesita para él. Las adaptaciones incluyen desde el poder ayudar a gente con movilidad reducida o el espacio libre del maletero, a si estás dispuesto a cambiar la temperatura del aire acondicionado del vehículo o a cambiar la música.

La opción de ocultar información se trata de una de las que más diferencia *NoLoneCommuter* de otras aplicaciones similares, y es que el usuario puede elegir si desea ocultar cualquier dato de su información personal, de contacto, de ayudas disponibles o necesitadas o de preferencias, para que no se le muestre a otros usuarios.

La sección de otras opciones permite al usuario eliminar su perfil de forma permanente. Esto borrará todos sus datos y su perfil, pero se quedarán almacenadas las rutas que haya realizado.

6.3. Recolección de información de las rutas

Para el correcto funcionamiento de la aplicación, es necesario recoger información sobre la ubicación del usuario, con el objetivo de formar rutas que el usuario realiza y encontrar así compañeros con rutas similares con los que compartir vehículos.

En primer lugar, el usuario deberá dar permiso a la aplicación para que esta acceda a su ubicación. Tras ello, la aplicación enviará al servidor la latitud y la longitud en la que se encuentra el usuario, junto con la fecha y hora en la que la información fue obtenida, cada 3 minutos o cada vez que el usuario se desplace 100 metros.

El objetivo de esa distancia y tiempo de espera es lograr formar rutas que se adapten a las reales pero sin agotar rápidamente la batería del dispositivo.

El valor de 3 minutos y 100 metros se aceptó como final ya que, sobre un mapa, 100 metros de desvío entre la ruta real y la formada en el servidor no suponen un gran problema, ya que los usuarios estarán dispuestos a desplazarse algunos kilómetros o, como mínimo, medio kilómetro, de forma que estos desvíos no causarán problemas en el algoritmo de búsqueda de compañeros de viaje.

En cuanto a la formación de rutas, el servidor trabaja del siguiente modo:

Cuando recibe una coordenada de la ubicación y la hora de un cierto usuario, la compara con la última coordenada y fecha que tenga del usuario anterior:

1. Si la distancia entre la nueva ruta y la última ruta almacenada es menor a 200 metros, entonces la nueva coordenada no se almacenará. Esta comprobación se trata de un añadido que no se tenía planeado implementar, pero ello estaba provocando que cada usuario tuviese una única ruta de gran longitud, y no varias rutas, ya que cada 3 minutos la aplicación enviaba una nueva coordenada y, pese a que el usuario no se moviese, esta se insertaba en su última ruta, haciendo imposible que las rutas acabasen.

2. Si la hora de la nueva coordenada difiere en más de una hora con la hora de la última coordenada almacenada, entonces se considera que se trata de una nueva ruta, en caso contrario será un punto más de la ruta anterior.

Si es se trata de un punto más de la ruta anterior, entonces simplemente se añade la nueva coordenada a la lista de coordenadas de la ruta, por supuesto, junto a la fecha en la que esta

fue tomada, para que el algoritmo se comporte de forma correcta en el futuro.

Si por el contrario, la coordenada es la primera de una nueva ruta, entonces se creará una nueva ruta con únicamente esa nueva coordenada, y con fecha de inicio igual a la fecha en la que se recogió la información de la nueva coordenada. Luego, se almacenan los nuevos datos en la base de datos y se lanza el *Algoritmo de Unificación de Rutas*, que se explica más adelante.

Téngase en cuenta que todos los valores fijos que se han comentado en este apartado, así como los que se comenten en apartados posteriores, están recogidos al principio de su archivo correspondiente en forma de constantes de código, por lo que son fácilmente modificables en caso de ser necesario.

6.4. Algoritmo de unificación de rutas

En las primeras versiones del producto, este algoritmo no existía, y se añadió posteriormente con el objetivo de solventar un sencillo problema: existían muchas rutas muy similares en la cuenta de cada usuario.

Como se aclaró anteriormente, este algoritmo se lanza cuando una nueva ruta es creada.

El algoritmo recibe como entrada el correo electrónico de un usuario, y la última ruta que este tenga almacenada, sin contar con nueva creada, por supuesto; y logra unificar esta ruta con alguna que sea similar, si es que la hay.

El criterio para que dos rutas sean similares es que compartan el 90 % de los puntos de la ruta más pequeña. Para realizar este cálculo, se emplea el *Algoritmo de compatibilidad de rutas*, que se detalla más adelante en el apartado *Algoritmo de búsqueda de rutas para compartir viajes*.

Entrando en detalle, el funcionamiento del algoritmo es el siguiente:

Primero, obtiene los datos del usuario cuyo email recibe como parámetro, y obtiene sus rutas. En segundo lugar, para cada una de sus rutas, salvo la que recibe como parámetro, se comprueba si son compatibles en un 90 %, es decir, si comparten el 90 % de las coordenadas de la ruta con menor número de puntos. En caso de encontrar una ruta compatible con esta nueva ruta, se añaden los identificadores y las fechas de la ruta más antigua a los de la nueva, se destruye la ruta más antigua y se actualiza la nueva.

Gracias a este algoritmo, podemos saber las rutas más frecuentes de un cierto usuario, y mostrar esta información a los usuarios que pretendan compartir ruta con éste, para que sepan si suele hacer la ruta o no, así como las últimas veces que la realizó y, para representar la hora,

se han considerado tres grandes franjas: MAÑANA, TARDE y NOCHE.

Cabe destacar que el algoritmo descrito anteriormente no fue así desde un principio. Esta fue su evolución:

1. En un primer momento, este algoritmo se ejecutaba siempre que se añadía un coordenada, ralentizando sumamente la respuesta del servidor, por lo que se decidió ejecutarlo solamente cuando se cerraba una ruta y se creaba otra nueva. El algoritmo comprobaba todas las rutas del usuario entre ellas, calculando si eran suficientemente similares como para unificarse, y unificando todas las rutas posibles.

2. Más adelante, nos dimos cuenta de que comparar todas las rutas no era necesario, ya que si este algoritmo se ejecutaba cada vez que se cerraba una ruta y se creaba otra nueva, sería suficiente con comparar la ruta recién cerrada con las anteriores, ya que todas las rutas anteriores habrán seguido ese proceso previamente, asegurándonos así que, tras la ejecución del algoritmo, todas las rutas estarían unificadas. Por ello, comparaba la ruta recién cerrada con todas las del usuario, unificando las que creía necesario.

3. Por último, se optó por la forma que se describió anteriormente, en la que si la ruta recién creada se unifica con otra, el algoritmo se detiene.

6.5. Algoritmo de búsqueda de rutas para compartir viaje

Este se trata del algoritmo encargado de realizar la principal funcionalidad de la aplicación: dado un cierto usuario, es capaz de devolver una lista de rutas y perfiles de usuario (con los datos ocultos) que son compatibles con el usuario recibido como parámetro.

El algoritmo actúa de la siguiente forma:

En primer lugar, obtiene la información del usuario cuyo email recibió como parámetro, así como la del resto de usuarios de la aplicación. Luego, compara la compatibilidad de cada uno de los otros usuarios con el usuario recibido. Para ello, analiza tres puntos distintos:

1. Comprueba si los usuarios tienen perfiles compatibles. Analiza si las necesidades del primer usuario pueden ser cubiertas por el segundo y viceversa. Si alguna necesidad no puede ser cubierta, entonces se determinará que esos usuarios no tienen rutas compatibles entre sí; en caso contrario, se continúa.

2. Obtiene la información de quién conducirá y cuánto está dispuesto a desviarse de su ruta principal. Aquí hay tres opciones:

    a) Ningún usuario quiere conducir. En este caso, se determinará que ambos usuarios no tienen rutas que compartir.

    b) Sólo un usuario quiere conducir. En este caso, el conductor será este usuario y la distancia que está dispuesto a desplazarse será la determinada por él.

    c) Ambos usuarios están dispuestos a conducir. En este caso, el conductor será el usuario que esté dispuesto a desviarse una distancia mayor.

3. Analiza si las rutas que realizan los usuarios son compatibles. Este algoritmo se conoce como *Algoritmo de compatibilidad de rutas*. Este compara las rutas de los usuarios dos a dos. Para cada par de rutas, se realizan las siguientes comparaciones:

    a) Primero, realiza un examen rápido. Esta funcionalidad no estaba implementada ni planeada en las primeras iteraciones del producto, pero surgió como método de optimización del algoritmo tras descubrir que muchos otros algoritmos que trabajaban con rutas cartográficas empleaban este método para saber si dos rutas eran o no similares. Este consiste en obtener el área de cada una de las rutas, y a analizar el área común entre ambas rutas. Si esta es igual o mayor al 20 % del área de la ruta con menor área, entonces se continúa el examen intensivo de la segunda parte del algoritmo; pero de no ser así, este subalgoritmo acabaría aquí indicando que las rutas recibidas no son compatibles.

    b) Por último, realiza un examen más a fondo, en el que se comparan las coordenadas de cada ruta, analizando si comparten, al menos, el 70 % de sus coordenadas. Esto determinará finalmente si las rutas son o no compatibles.

Si dos rutas son compatibles, se añade a la lista de rutas y perfiles compatibles que devolverá el algoritmo la ruta del usuario que conduzca y el perfil del usuario que no sea el que se recibió como parámetro. Una vez se termine el análisis, se devuelve dicha lista.

Por supuesto, todos los valores mencionados en este apartado se encuentran recogidos en forma de constantes al inicio del archivo que los use, por lo que son fácilmente editables si se descubriesen valores más óptimos en el futuro.

6.6. Gestión de viajes

Otra funcionalidad muy a destacar de NoLoneCommuter es la gestión de viajes.

Cuando un usuario selecciona la opción de *buscar a un compañero de viaje*, se le muestra una lista de rutas y usuarios dispuestos a compartir vehículo con ellos, indicando las veces que ha realizado las rutas y las fechas y el momento del día (MAÑANA, TARDE O NOCHE) en las que lo ha hecho. Si el usuario acepta el viaje, entonces se creará un viaje entre ambos usuarios, en el que se supone que el usuario que solicitó la ruta ha aceptado. El usuario deberá determinar la fecha y hora del viaje, si quiere o no conducir y algún comentario adicional que quiera que vea el otro usuario.

Cada usuario puede ver la lista de sus viajes, y aceptarlos si otros usuarios se los propusieron. Cada viaje tiene información sobre la ruta, la fecha, un comentario, una valoración y el perfil del compañero de viaje.

En cuanto a la valoración, cada usuario puede valorar del 1 al 5 el servicio del compañero de ruta en un cierto viaje. Esta información se empleará para mostrar cómo de buenos son los servicios de un usuario. Cada usuario tendrá por ello, una valoración, que será la media aritmética entre todos los votos recibidos por los usuarios. Si bien existen otras formas de calcular la valoración media de usuarios, que tienen en cuenta otros criterios de mayor complejidad como la severidad en las votaciones de cada usuario, finalmente se ha optado por simplemente emplear la media aritmética, por mayor simplicidad. De igual forma, como todo el código relativo a esto está recogido en un único fichero, muy desacoplado del resto del código y con un interfaz de uso muy claro, sería muy sencillo adaptar este comportamiento en un futuro.

En lo relativo al orden en que se muestran los viajes, se ha optado por mostrar primero los viajes más recientes, y más adelante los más antiguos, ya que normalmente, los usuarios están más interesados en los viajes presentes y futuros que en analizar los ya realizados.

6.7. Chat

NoLoneCommuter cuenta con un chat interno que permite a los usuarios comunicarse de forma sencilla para que acuerden los detalles de sus viajes.

El funcionamiento del chat es muy sencillo, simplemente existe una colección de conversaciones que almacena conversaciones entre dos usuarios, incluyendo algunos meta-datos ya

comentados anteriormente y los propios mensajes.

Cada usuario puede enviar un mensaje a cualquier otro usuario, bien sea a través de una conversación ya existente o a través del perfil del destinatario. El backend desconoce el origen del mensaje, por lo que comprueba si existe o no una conversación anterior entre los usuarios. Si es así, añade el nuevo mensaje, y en caso contrario, crea una nueva conversación entre los usuarios con el nuevo mensaje como primero de la lista.

La obtención de la lista de conversaciones de un usuario es simplemente una lectura de la base de datos.

La aplicación también muestra el número de mensajes sin leer que tiene el usuario en forma de burbuja, como se mostró en el capítulo de diseño. Para ello, el servidor cuenta con un punto de acceso que permite a los usuario preguntar por cuántos mensajes tienen sin leer, respondiendo únicamente con un número. Esto facilita la implementación de esa funcionalidad, y hace liviano el punto de acceso, evitando así cálculos en el frontend.

Por último, la gestión de si un mensaje está leído por el receptor o no se realiza en el backend. Cada vez que un usuario solicite una conversación en concreto, todos sus mensajes sin leer se marcan como leídos. De esta forma no es necesario añadir otro punto de acceso para marcar mensajes como leídos, ni hacer cálculos o peticiones adicionales en el frontend.

6.8. Pruebas

Durante el desarrollo del proyecto, únicamente se realizaron pruebas para comprobar el correcto funcionamiento de dos funciones del backend: el algoritmo de búsqueda de compañeros y el de unificación de rutas.

Más concretamente, se realizaron dos métodos de prueba, uno para cada uno de los algoritmos citados anteriormente, encargados de verificar su correcto funcionamiento, comprobando que los resultados devueltos eran correctos para un amplio abanico de posibilidades.

Se decidió no probar de forma sistemática los elementos del frontend, ya que se tratan de componentes simples que se basan en otras librerías o módulos que ya están probados, y considero que con ejecutarlos y comprobar de forma manual su correcto funcionamiento es suficiente.

De igual manera, se decidió no implementar más pruebas automatizadas en el backend ya que el resto de funcionalidades son muy simples, y ya fueron probadas de forma manual

durante las diferentes iteraciones.

7

Conclusiones y trabajos futuros

En este capítulo se proporcionarán las conclusiones finales del proyecto y se propondrán ideas y funcionalidades que podrían ser trabajadas y desarrolladas en un futuro para mejorar la aplicación actual.

7.1. Conclusiones

A modo de conclusión general, me gustaría destacar que el desarrollo de este trabajo ha cumplido con la gran mayoría de los objetivos que tenía planteados desde el inicio.

El producto entregado cumple con todos los requisitos que personalmente deseaba que cubriese, especialmente opino que los algoritmos de creación de nuevas rutas y de búsqueda de compañeros, además por supuesto de ser funcionales, son eficientes. Desde el principio del desarrollo del proyecto me preocupaba especialmente este apartado, ya que era relativamente sencillo tanto almacenar coordenadas, como comparar rutas; pero no lo era tanto el hacerlo consumiendo los mínimos recursos. Es por ello que considero un éxito la implementación de estos algoritmos, que constituyen el núcleo de la aplicación, haciendo posible las funciones más características del producto.

El principal reto que me planteaba este proyecto, y que considero haber solventado también con éxito, era el uso de tecnologías que jamás había empleado anteriormente. Principalmente caben destacar dos: *Go Language* y *React Native*. Como se comenta en el capítulo de tecnologías, estas dos son muy recientes y empeladas por numerosos grandes negocios, y es por ello y por familiarizarme con tecnologías que no fuesen explicadas en el grado que decidí emplearlas.

Por otra parte, también estoy muy satisfecho de las funcionalidades adicionales con las que cuenta la aplicación, como son el sistema de gestión de perfiles y el chat integrado, que

considero complementan muy bien a la funcionalidad principal.

Adicionalmente, también considero haber aplicado de forma correcta los conocimientos adquiridos en el grado. Esto se ve reflejado en la estructura de la aplicación, siendo que esta diferencia claramente entre back y frontend, y modulariza para cada sección las funcionalidades más importantes en paquetes de tamaño reducido y fácilmente legibles, entendibles, modificables y adaptables. Más allá del código, los conocimientos sobre el desarrollo de software en todas sus etapas también me han sido extremadamente útiles para tener claro qué metodología seguir y no estancarme en ningún momento.

Como es evidente, durante todo el proceso de desarrollo he encontrado diversas dificultades, mayormente atraídas por mi desconocimiento de las tecnologías empleadas y las decisiones que se deben tomar para diseñar los principales algoritmos de la aplicación. En general, para resolver estos problemas y dudas, opté por hacer búsquedas en la web y, en contadas ocasiones, debatir sobre algunas con mi tutor. Cabe destacar que en ningún momento quedé bloqueado por una duda durante un largo periodo de tiempo.

Para finalizar, cabe destacar que el desarrollo de este trabajo me ha ayudado enormemente a crecer como profesional, ya que con él he aprendido mucho el proceso de desarrollo software en general, y las tecnologías empeladas en particular; así como sobre cómo resolver problemas y dudas y cómo redactar documentación.

7.2. Trabajos futuros

En general, considero que la aplicación se encuentra preparada para ser lanzada al mercado sin ningún problema, y proporciona un servicio muy completo, pero siempre cabe margen de mejora, y es por ello que propongo estas cinco líneas de desarrollo que podrían desarrollarse en un futuro para complementar la aplicación:

1. **Organizar viajes entre varias personas**.

Una buena forma de ampliar la funcionalidad principal de la aplicación, sería la de permitir no sólo compartir vehículo entre dos personas, sino entre varias más. Un buen enfoque para esta tarea podría ser el de, justo antes de devolver las rutas y compañeros recomendados como se hace actualmente, se hiciese un examen que comprobase si existe algún otro viaje no completado similar al que se desea realizar y, de ser así, ofrecer al usuario la posibilidad de

incorporarse a este.

Por supuesto, en todo momento deberíamos asegurarnos de que todos los integrantes de un viaje están conformes, por lo que si un usuario elige un viaje con varios compañeros, todos deben ser notificados y dar su visto bueno, de lo contrario no se realizará el viaje y quedará cancelado.

Para hacer esto posible serían necesarios algunos cambios en los modelos de datos de tanto los perfiles, para incluir más información sobre preferencias; como en los viajes, para que puedan soportar tener más de un pasajero. También serían necesarios cambios en el servidor web, principalmente en el algoritmo de búsqueda de compañero, y en el frontend, para mostrar de forma clara las nuevas opciones.

2. **Notificaciones**.

   Dada la naturaleza interactiva de la aplicación, sería también conveniente incluir un sistema de notificaciones en la aplicación.

   Lo más interesante sería que un usuario fuese notificado cuando recibe un mensaje a través del chat interno o cuando alguien le propone un viaje compartido.

   También sería interesante que los usuarios pudiesen personalizar qué notificaciones recibir y cuáles no, así como desactivarlas por completo si lo desean.

3. **Usar HTTPS**.

   También sería interesante dejar atrás *HTTP* y emplear *HTTPS* como protocolo de comunicación entre la aplicación y el servidor web. Pese a que los datos más sensibles, como las contraseñas, se envía encriptados, siempre es conveniente emplear *HTTPS* para proveer de más seguridad a la aplicación.

   Dada la modularización de la aplicación y las facilidades que ofrece *Go Language* para la generación y renovación de *cerificados TLS* [37], con la existencia de librerías como *certmagic* [38] esto sería muy sencillo.

4. **Optimizar el envío de coordenadas**.

   Actualmente, la aplicación envía al servidor información sobre la ubicación del usuario cada 3 minutos o si este se desplaza 100 metros. Si bien esto no se traduce en un consumo excesivo de batería o tráfico web, podría modificarse la aplicación para que esta almacenase coordenadas durante un cierto periodo de tiempo y, en cuanto tuviese conexión a internet las

enviase al servidor. De esta forma se reduciría el tráfico web y el consumo de batería.

Implementar este cambio sería sencillo, pues todo lo relativo al envío de coordenadas se encuentra en un paquete independiente tanto el el servidor como en la aplicación móvil.

5. **Añadir estadísticas de los usuarios**.

Otra posible adición a la aplicación podría ser una sección en la que se mostrasen estadísticas sobre un usuario concreto. Algunas de ellas podría ser el número total de viajes realizados, el número de viajes realizados ese mes, el número de viajes recibidos, sus rutas más realizadas, si suele compartir vehículo, un resumen de su actividad semanal, y un largo etcétera.

Por supuesto, el usuario podría elegir qué estadísticas de su perfil mostrar y cuáles mantener ocultas.

Referencias

[1] *Android | La plataforma que amplía los límites de lo posible.* (s. f.). Android. Recuperado 1 de noviembre de 2020, de https://www.android.com/intl/es_es/

[2] Statista. (2020, 27 noviembre). *Porcentaje de viviendas teléfono móvil España 2006–2020.* https://es.statista.com/estadisticas/539901/porcentaje-de-viviendas-con-aparatos-de-telefonia-movil-en-espana/#:%7E:text=La%20penetraci%C3%B3n%20de%20este%20tipo,con%20terminales%20de%20telefon%C3%ADa%20m%C3%B3vil.

[3] M. (2016, 29 septiembre). *El 18 % de los españoles tiene un smartwatch.* MuyCanal.https://www.muycanal.com/2016/09/29/pulsera-deportiva-smartwatch-espana

[4] Fernández, S. (2019, 16 abril). *Android supera el 90 % de cuota en España mientras que iOS cae por debajo del 9 %, según Kantar.* Xataka Móvil. https://www.xatakamovil.com/mercado/android-supera-90-cuota-espana-ios-cae-debajo-9-kantar

[5] *blablacar.es.* (s. f.). BlaBlaCar. Recuperado 1 de noviembre de 2020, de https://www.blablacar.es/

[6] *Compartir coche en España y Europa, carpool spain.* (s. f.). Carpooling. Recuperado 1 de noviembre de 2020, de https://www.europe-carpooling.es/

[7] Explorar la plataforma. (s. f.). *Uber.* Recuperado 1 de noviembre de 2020, de https://www.uber.com/es/es-es/

[8] *La ciudad es tuya. Viaja con nosotros | Cabify.* (s. f.). Cabify. Recuperado 1 de noviembre de 2020, de https://cabify.com/es

[9] Google. (s. f.). *Google Firebase.* Google Firebase. Recuperado 20 de noviembre de 2020, de https://firebase.google.com/

[10] Google. (s. f.-a). *Cloud Firestore.* Firebase. Recuperado 21 de noviembre de 2020, de https://firebase.google.com/docs/firestore

[11] colaboradores de Wikipedia. (2021, 17 enero). *NoSQL*. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/NoSQL#:%7E:text=En%20inform%C3%A1tica%2C%20NoSQL%20(a%20veces,como%20lenguaje%20principal%20de%20consultas

[12] *The Go Programming Language*. (s. f.). Go Language. Recuperado 10 de diciembre de 2020, de https://golang.org/

[13] G. (s. f.). *gin-gonic/gin*. GitHub. Recuperado 29 de mayo de 2021, de https://github.com/gin-gonic/gin

[14] colaboradores de Wikipedia. (2021b, julio 15). *Protocolo de transferencia de hipertexto*. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Protocolo_de_transferencia_de_hipertexto

[15] *React Native · Learn once, write anywhere*. (s. f.). React Native. Recuperado 29 de mayo de 2021, de https://reactnative.dev/

[16] *Typed JavaScript at Any Scale*. (s. f.). TypeScript. Recuperado 29 de mayo de 2021, de https://www.typescriptlang.org/

[17] *Core Components and APIs · React Native*. (s. f.). React Native Core Components. Recuperado 29 de mayo de 2021, de https://reactnative.dev/docs/components-and-apis

[18] A. (s. f.-a). *axios/axios*. Axios. Recuperado 29 de mayo de 2021, de https://github.com/axios/axios

[19] colaboradores de Wikipedia. (2021b, mayo 23). *JavaScript*. JavaScript. https://es.wikipedia.org/wiki/JavaScript

[20] *GitHub: Where the world builds software*. (s. f.). GitHub. Recuperado 1 de noviembre de 2020, de https://github.com/

[21] *Git*. (s. f.). Git. Recuperado 1 de noviembre de 2020, de https://git-scm.com/

[22] Microsoft. (s. f.). *Visual Studio Code - Code Editing. Redefined*. Visual Studio Code. Recuperado 1 de noviembre de 2020, de https://code.visualstudio.com/

[23] colaboradores de Wikipedia. (s. f.). *Entorno de desarrollo integrado.* IDE. Recuperado 1 de noviembre de 2020, de https://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado

[24] *Code Runner - Visual Studio Marketplace.* (s. f.). Code Runner VSCode. Recuperado 29 de mayo de 2021, de https://marketplace.visualstudio.com/items?itemName=formulahendry.code-runner

[25] Microsoft. (s. f.-a). *Go with Visual Studio Code. Go VSCode.* Recuperado 29 de mayo de 2021, de https://code.visualstudio.com/docs/languages/go

[26] Go *Nightly - Visual Studio Marketplace.* (s. f.). Go Nightly VSCode. Recuperado 29 de mayo de 2021, de https://marketplace.visualstudio.com/items?itemName=golang.go-nightly

[27] Go - *Visual Studio Marketplace.* (s. f.). Golang VSCode. Recuperado 29 de mayo de 2021, de https://marketplace.visualstudio.com/items?itemName=golang.Go

[28] Microsoft. (s. f.-b). *JSON editing in Visual Studio Code. JSON VSCode.* Recuperado 29 de mayo de 2021, de https://code.visualstudio.com/docs/languages/json

[29] *Prettier - Code formatter - Visual Studio Marketplace.* (s. f.). Prettier VSCode. Recuperado 29 de mayo de 2021, de https://marketplace.visualstudio.com/items?itemName=esbenp.prettier-vscode

[30] *Microsoft. (s. f.-c). Microsoft Word: software de procesamiento de texto | Microsoft 365. Microsoft Word.* Recuperado 1 de noviembre de 2020, de https://www.microsoft.com/es-es/microsoft-365/word

[31] colaboradores de Wikipedia. (s. f.-b). *LaTeX.* Latex. Recuperado 1 de noviembre de 2020, de https://es.wikipedia.org/wiki/LaTeX

[32] *Inicio - Universidad de Málaga.* (s. f.). UMA. Recuperado 19 de julio de 2021, de https://www.uma.es/

[33] *No Magic, Inc - Unified Modeling Language (UML), SYSML, UPDM, SOA, Business Process Modeling Tools - CATIA - Dassault Systèmes®.* (s. f.). MagicDraw. Recuperado 1 de noviembre de 2020, de https://www.3ds.com/products-services/catia/products/no-magic/

[34] *Cloudinary. (s. f.). Image and Video Upload, Storage, Optimization and CDN.* Recuperado 1 de noviembre de 2020, de https://cloudinary.com/

[35] *Trello.* (s. f.). Trello. Recuperado 1 de noviembre de 2020, de https://trello.com/

[36] Atlassian. (s. f.). *Jira | Software de seguimiento de proyectos e incidencias. Jira.* Recuperado 1 de noviembre de 2020, de https://www.atlassian.com/es/software/jira

[37] Team, P. (2021, 18 agosto). *What is a TLS/SSL certificate, and how does it work?* ProtonMail Blog. https://protonmail.com/blog/tls-ssl-certificate/

[38] C. (s. f.-b). *GitHub - caddyserver/certmagic: Automatic HTTPS for any Go program: fully-managed TLS certificate issuance and renewal.* GitHub. Recuperado 1 de mayo de 2021, de https://github.com/caddyserver/certmagic

Apéndice A
Manual de Despliege e Instalación

A.1. Introducción

Este documento describe el proceso a seguir para instalar la aplicación en su teléfono móvil y el servidor de esta en su computador.

En este manual se detallan de forma clara y ordenada los pasos a seguir para instalar el servidor web que emplea la aplicación en su computador, y para instalar la aplicación en su teléfono móvil con sistema operativo Android.

Para completar los pasos que se describen posteriormente, es necesario que usted posea un computador con sistema operativo Windows o Linux, y un teléfono móvil con sistema operativo Android.

Si le surge cualquier duda o comentario, por favor no dude en contactar con nosotros a vía correo electrónico.

A.2. Primeros pasos

En primer lugar, descargue todos los archivos que se adjuntan en la entrega del trabajo.

Entre estos archivos, localice uno llamado *source-code.zip*, y descomprímalo. El nuevo directorio contiene el código fuente de tanto el servidor web como la aplicación móvil.

Finalmente, instale las siguientes herramientas en su computador:

1. Go language: https://golang.org/doc/install.
2. React Native: https://reactnative.dev/docs/environment-setup.
3. Expo: https://docs.expo.dev/get-started/installation/.

4. yarn: https://classic.yarnpkg.com/en/docs/install/#windows-stable.

A.3. Instalación del servidor

Para hacer funcionar el servidor en su máquina, abra un terminal en la ubicación en la que haya descomprimido los archivos. Asegúrese de que el archivo *main.go* se encuentra en ese directorio. Para ello puede ejecutar el comando *dir* en Windows o *ls* en Linux y comprobar que el archivo aparece entre los listados.

A continuación, ejecute el comando *go mod download* para instalar las dependencias del servidor y, posteriormente, *go run main.go* para ejecutarlo.

Téngase en cuenta que el servidor se ejecutará en el puerto *8080*, así que si tiene algún servicio ocupando ese puerto, asegúrse de detenerlo.

Finalmente, es necesario conocer en qué dirección IP está corriendo el servidor, por lo que debe abrir un nuevo terminal y ejecutar *ipconfig* si dispone de un computador con sistema operativo Windows, o *ifconfig* si este es Linux, y buscar cuál es la IP de su computador. Normalmente, estas tienen el formato *192.168.XXX.YYY*. Una vez tenga esta información, guárdela, será de utilidad en el siguiente paso.

A.4. Construir la aplicación móvil

El segundo paso es construir la aplicación móvil.

Primeramente, diríjase al fichero *app/api/base.ts* y sustituya la indicación <*su-ip*> de la cuarta línea por la dirección IP en la que esté corriendo su servidor, que obtuvo y guardó en el paso anterior.

A continuación, abra una nueva terminal en el directorio en el que extrajo el código fuente y ejecute *cd app*.

Ejecute el comando *expo build:android -t apk* para generar el instalador de aplicaciones Android con extensión *apk*. Este proceso tomará varios minutos. Al final del proceso se le proporcionará un enlace, acceda a él para descargar el archivo *apk*.

A.5. Transferir el archivo al teléfono

El siguiente paso es transferir el archivo descargado anteriormente a su teléfono Android. Para ello, puede enviárselo por correo electrónico, conectar el dispositivo vía USB a su computador o emplear cualquier otro método que vea conveniente.

A.6. Instalar la aplicación

Para poder instalar finalmente la aplicación en su dispositivo móvil, primero debe configurar su dispositivo para poder instalar aplicaciones de terceros. Para ello, puede seguir esta guía o una similar: https://www.xatakandroid.com/tutoriales/como-instalar-aplicaciones-en-apk-en-un-movil-android.

Tras ello, simplemente toque la aplicación y la instalación comenzará.

Finalmente, la aplicación estará instalada y lista para ser usada. Si tiene dudas sobre su uso, consulte el *Apéndice B: Manual de Usuario* de este mismo documento.

Apéndice B

Manual de Usuario

B.1. Introducción

Este documento tiene como objetivo facilitar a los usuarios de la aplicación web su uso correcto, resolviendo las posibles dudas que puedan surgir en el momento de su utilización.

Este manual explica paso a paso qué acciones se deben realizar para completar las distintas tareas que la aplicación ofrece, partiendo del lanzamiento de la aplicación en un teléfono móvil con sistema operativo Android que la tuviese previamente instalada.

Si le surge cualquier duda o comentario, por favor no dude en contactar con nosotros a vía correo electrónico.

B.2. Creación de una cuenta

Cuando inicie la aplicación verá la pantalla de inicio de sesión (*Figura 1*). Una vez allí, seleccione la opción *¿No tiene una cuenta aún?, crea una aquí.*

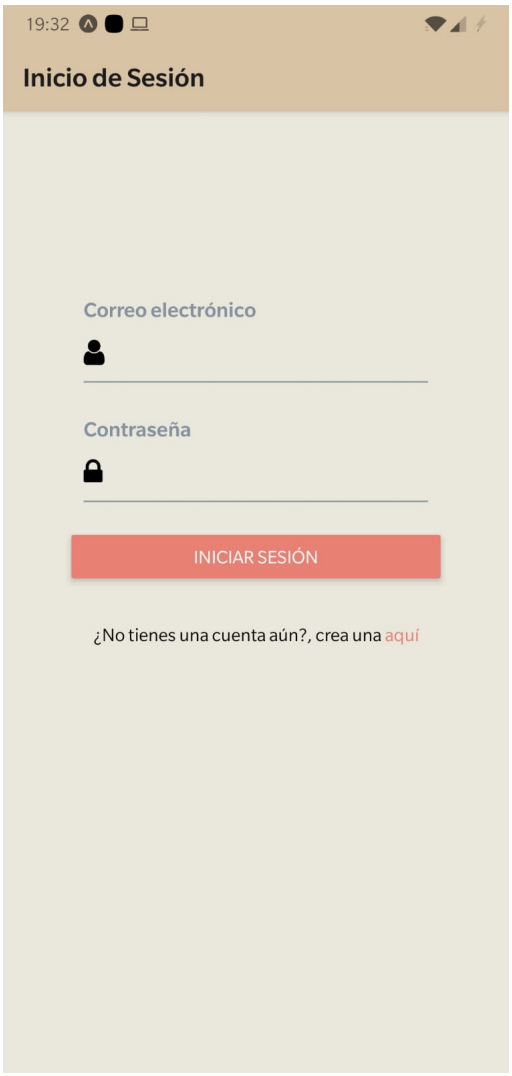


Figura 1: Pantalla de inicio de sesión.

Esto le mostrará la pantalla de creación de nueva cuenta (*Figura 2*), la cual consiste en un simple formulario que deberá rellenar con los siguientes datos:

1. Su nombre completo.

2. Un correo electrónico válido que no esté registrado anteriormente en la aplicación.

3. Una contraseña segura. En este caso, debe contener al menos ocho caracteres, con mínimo un carácter en mayúsculas y otro en minúsculas, como se indica en el cuadro de la pantalla.

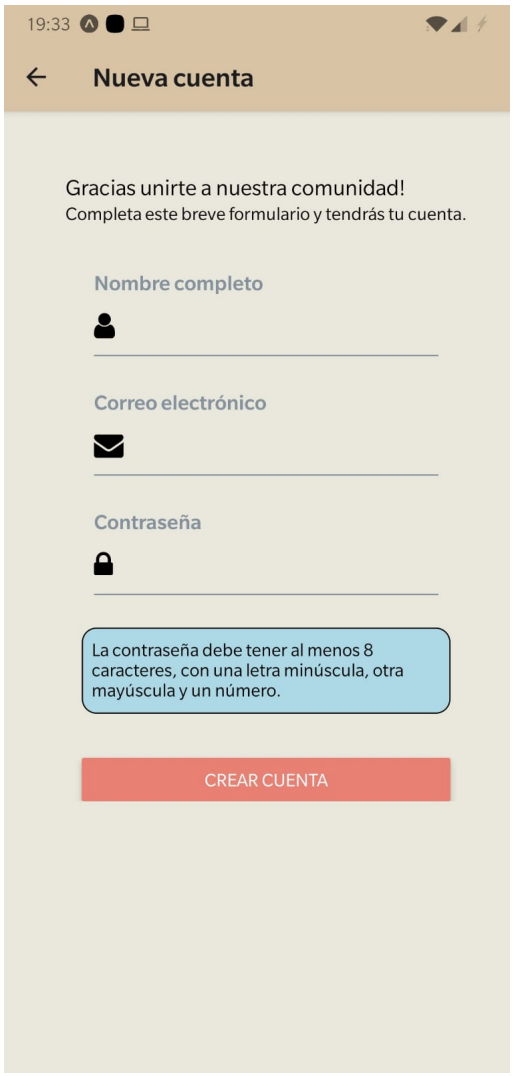


Figura 2: Pantalla de nueva cuenta.

Finalmente, seleccione el botón *Crear Cuenta*. Su cuenta será creada y se le dirigirá a la pantalla de inicio de sesión de nuevo (*Figura 1*)

En caso de introducir datos inválidos o de que surja un problema en la creación de su cuenta, se le notificará el error.

B.3. Inicio de sesión

Para poder iniciar sesión en el sistema, deberá tener una cuenta creada. Si no la tiene, consulte el apartado anterior.

Para iniciar sesión con su cuenta, simplemente deberá introducir su correo electrónico y su contraseña en el formulario de inicio se sesión que se muestra al arrancar la aplicación (*Figura*

1) y seleccionar el botón *Iniciar Sesión*.

Si los datos son correctos, se le mostrará la página principal de la aplicación (*Figura 3*). En caso contrario, se le mostrará un mensaje indicando el error.

Si usted tiene viajes sin ver o mensajes sin leer, aparecerá un indicador sobre los botones *Rutas y Viajes* y *Mis Chats*, respectivamente, indicando el número de nuevos elementos.

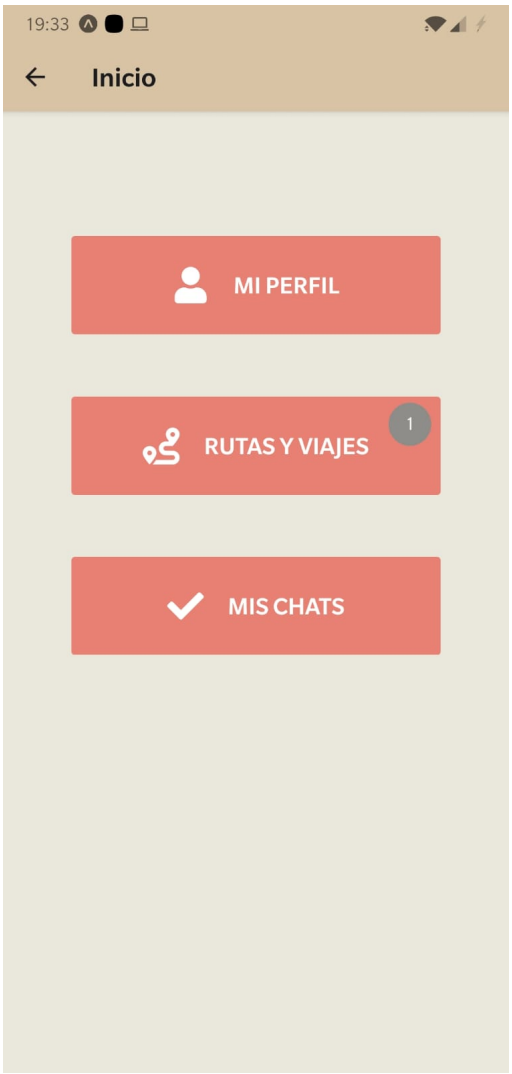


Figura 3: Pantalla de pantalla principal.

B.4. Gestionar su perfil

En esta sección se mostrará al usuario cómo modificar los datos de su perfil, incluyendo:

1. Su información personal.
2. Sus preferencias.

3. La información que desee ocultar.

4. Las ayudas que puede proporcionar y las que necesita.

5. Otras opciones.

Para poder realizar las acciones que se describen en esta sección, el usuario debe encontrarse en la pantalla de inicio (*Figura 3*) y seleccionar el botón *Mi Perfil*. Esto le mostrará el menú del perfil (*Figura 4*).

Figura 4: Pantalla de mi perfil.

B.4.1. Modificar información personal

Para modificar su información personal, seleccione el botón *Información Personal*. Tras ello, se le mostrará la pantalla de su información personal (*Figura 5*).

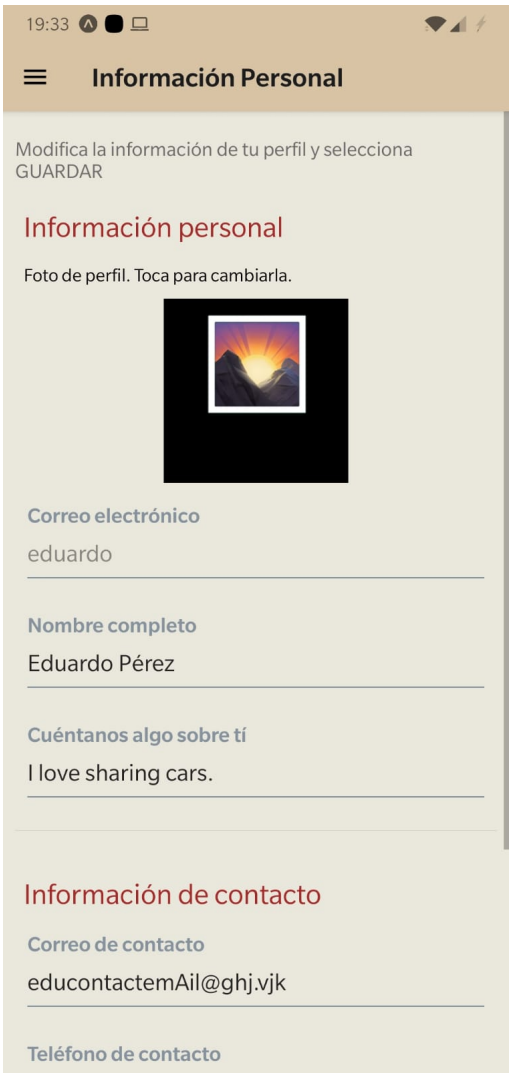


Figura 5: Pantalla de datos personales.

En esta pantalla podrá modificar todos los datos personales que se muestran. Si desea cambiar su imagen de perfil, simplemente debe tocarla y se abrirá su galería para que seleccione la que vea más conveniente.

Tras realizar todos los cambios, seleccione el botón *Guardar* y se actualizarán sus datos.

En caso de que suceda algún error o algún campo introducido no sea válido, se le mostrará un mensaje de error.

B.4.2. Modificar preferencias

Para modificar sus preferencias en la aplicación, seleccione el botón *Preferencias*. Tras ello, se le mostrará la pantalla de sus preferencias (*Figura 6*),

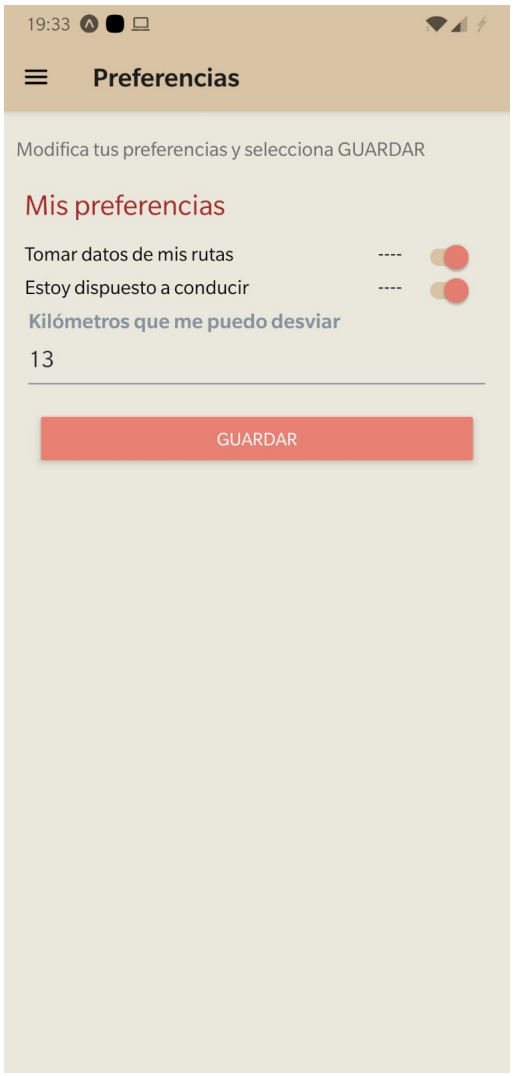


Figura 6: Pantalla de preferencias.

En esta pantalla, podrá modificar todos los datos relativos a sus preferencias en la aplicación. Tenga en cuanta que si selecciona que no está dispuesto a conducir, se ocultará la sección que le pregunta cuánta distancia está dispuesto a desviarse para recoger a un compañero, pues esta información no tiene sentido en ese caso.

Tras realizar todos los cambios, seleccione el botón *Guardar* y se actualizarán sus datos.

En caso de que suceda algún error o algún campo introducido no sea válido, se le mostrará un mensaje de error.

B.4.3. Ocultar información

Para ocultar datos de su perfil de modo que otros usuarios no puedan verlos, selecciona el botón *Ocultar Información*. Esto le mostrará la pantalla de ocultar información (*Figura 7*).

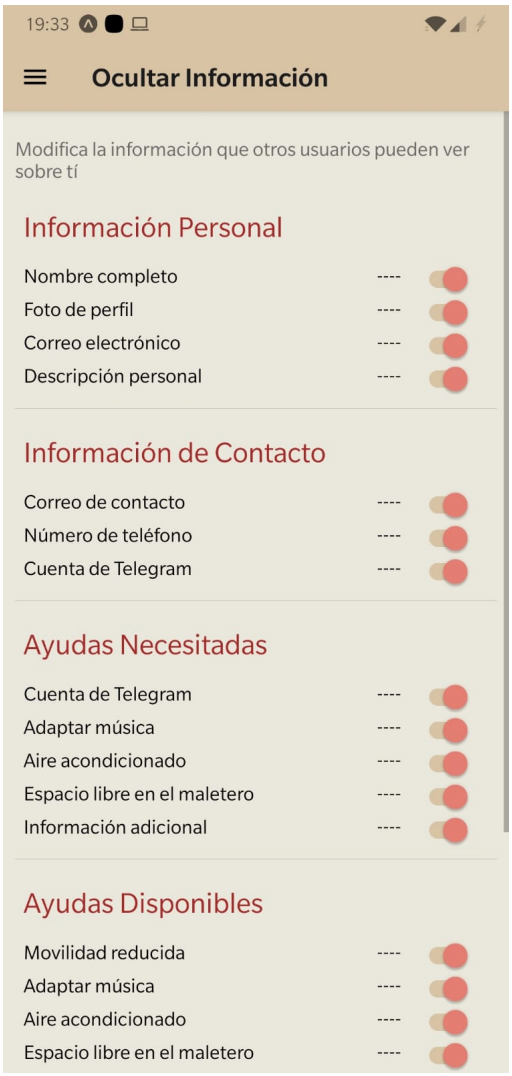


Figura 7: Pantalla de ocultar información.

En esta pantalla podrá seleccionar qué elementos desea ocultar y cuáles desea mantener visibles a otros usuarios. Los datos se encuentran agrupados por categorías para que sea más simple localizarlos.

Tras realizar todos los cambios, seleccione el botón *Guardar* y se actualizarán sus datos.

En caso de que suceda algún error o algún campo introducido no sea válido, se le mostrará un mensaje de error.

B.4.4. Modificar ayudas

Para configurar qué ayudas puede aportar y de cuáles requiere, seleccione el botón *Ayuda*. Esto le mostrará la pantalla de ayudas (*Figura 8*).

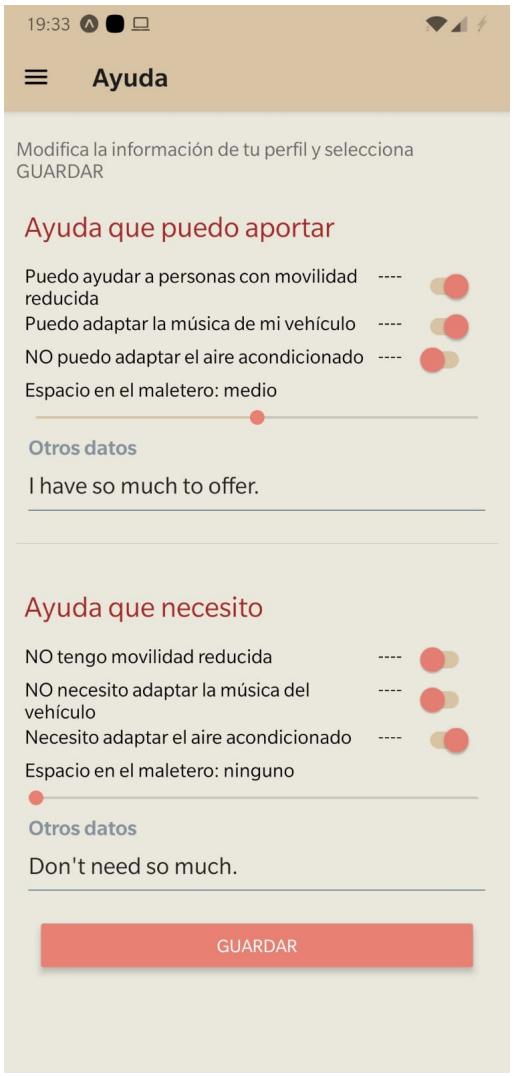


Figura 8: Pantalla de ayudas a aportar y necesitadas.

En esta pantalla podrá seleccionar tanto las ayudas que puede aportar como las que necesita.

Tras realizar todos los cambios, seleccione el botón *Guardar* y se actualizarán sus datos.

En caso de que suceda algún error o algún campo introducido no sea válido, se le mostrará un mensaje de error.

B.4.5. Eliminar perfil

Si desea eliminar permanentemente su perfil del sistema, seleccione el botón *Otros*. Esto le mostrará la pantalla de otros cambios (*Figura 9*).

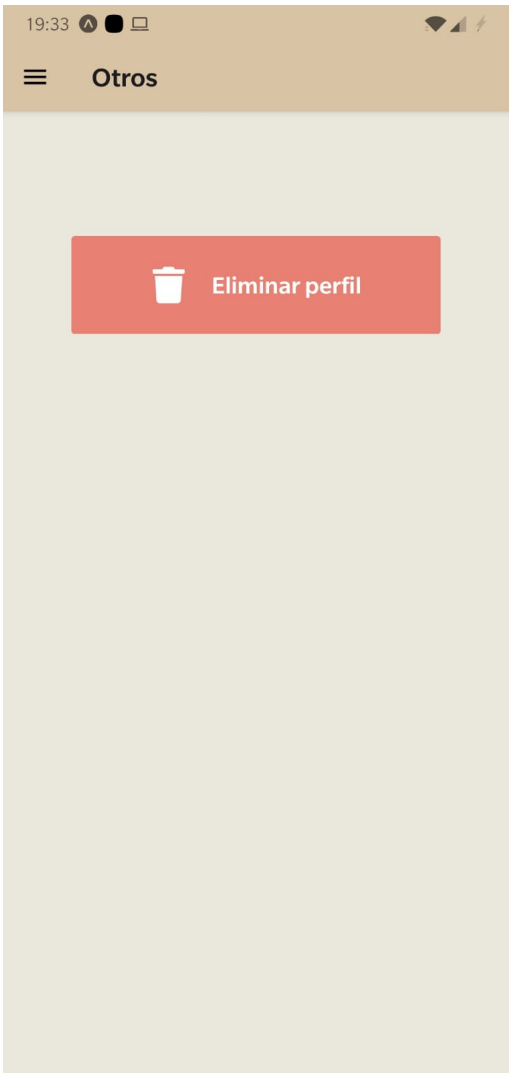


Figura 9: Pantalla de otras opciones.

Aquí, seleccione el botón *Eliminar perfil*. Se le mostrará un mensaje de confirmación. Si lo acepta, se eliminarán permanentemente los datos de su perfil, no pudiendo recuperarlos de ninguna forma.

B.5. Gestionar rutas y viajes

En esta sección se le mostrará cómo:

1. Ver las rutas que el sistema almacena sobre usted.
2. Eliminar rutas que el sistema haya almacenado.
3. Buscar un compañero de viaje.

4. Ver el perfil de un compañero de viaje.
5. Enviar un mensaje a un compañero.
6. Solicitar un viaje con un compañero.
7. Ver la lista de sus viajes.
8. Aceptar un viaje.
9. Valorar un viaje.

Para realizar las acciones que se detallan a continuación, debe seleccionar el botón *Rutas y Viajes* en la pantalla de inicio (*Figura 3*). Tras ello, se le mostrará la pantalla de menú de rutas y viajes (*Figura 10*).

Figura 10: Pantalla de viajes y rutas.

B.5.1. Ver sus rutas

Para ver la lista de las rutas que el sistema ha obtenido sobre usted, seleccione el botón *Mis Rutas*. Esto le mostrará un listado de sus rutas (*Figura 11*).

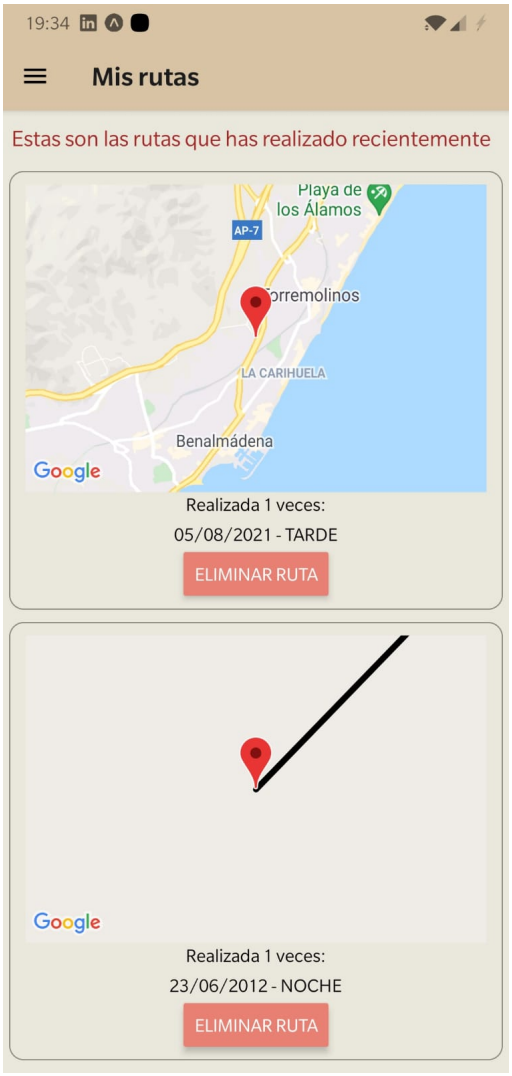


Figura 11: Pantalla de ver mis rutas.

B.5.2. Eliminar ruta

Si desea eliminar una ruta de las que se listaron en el apartado anterior, simplemente debe seleccionar el botón *Eliminar Ruta* que aparece junto a cada una de las rutas, como se aprecia en la *Figura 11*.

Se le mostrará un mensaje de confirmación y, si lo acepta, la ruta quedará eliminada de forma permanente.

B.5.3. Buscar compañero de viaje

Si desea buscar un compañero de viaje que se adapte a sus necesidades y realice alguna ruta similar a las tuyas, selecciona el botón *Buscar compañero*.

Tras un tiempo de carga, se le mostrará un listado de elementos formados por una ruta y un usuario, con información sobre cuándo suele realizar la ruta (*Figura 12*).

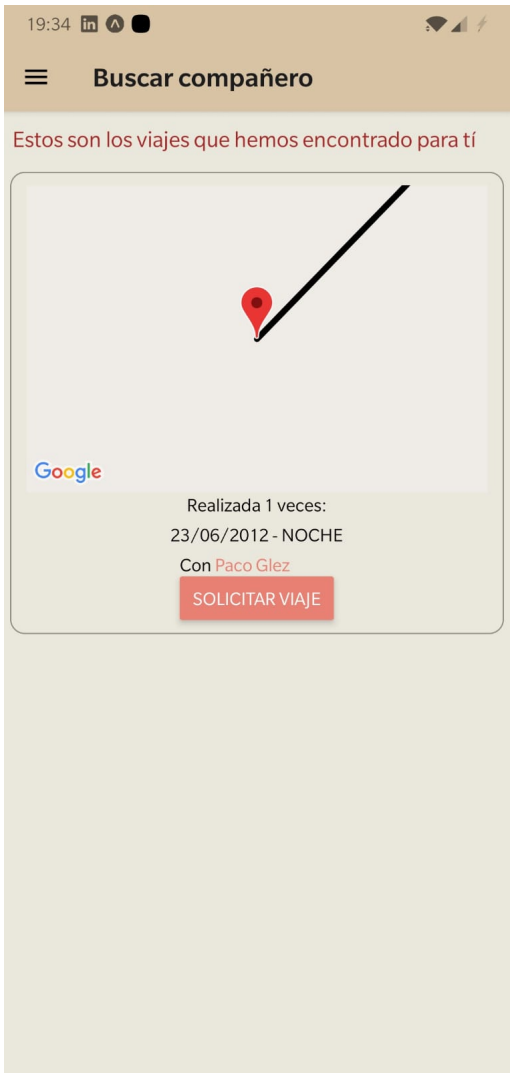


Figura 12: Pantalla de buscar compañero de viaje.

B.5.4. Ver perfil de un compañero de viaje

Si desea ver la información de perfil de uno de los compañeros de viaje listados en el apartado anterior, simplemente seleccione su nombre que aparece en la *Figura 13*.

Esto mostrará la pantalla de perfil del usuario, por supuesto, ocultando los datos que él

haya decidido mantener ocultos (*Figura 13*).

Figura 13: Pantalla de perfil de un compañero.

B.5.5. Enviar mensaje a un compañero de viaje

Si desea enviar un mensaje a través del chat interno de la aplicación a un compañero cuyo perfil visualizaste en el apartado anterior (*Figura 13*), simplemente seleccione el botón *Enviar Mensaje a <Usuario>*.

Esto abrirá una ventana emergente en la que podrá redactar su mensaje y enviarlo.

B.5.6. Solicitar un viaje con un compañero

Si alguno de los viajes que vio en el apartado *Buscar compañero de viaje* le convenció, puede solicitarlo seleccionando el botón *Solicitar Viaje*.

Esto le mostrará la pantalla de creación de viaje (*Figura 14*), que consiste en un simple formulario en el que introducirá si desea o no conducir, la fecha y hora de salida y algún dato adicional que desee que vea el compañero.

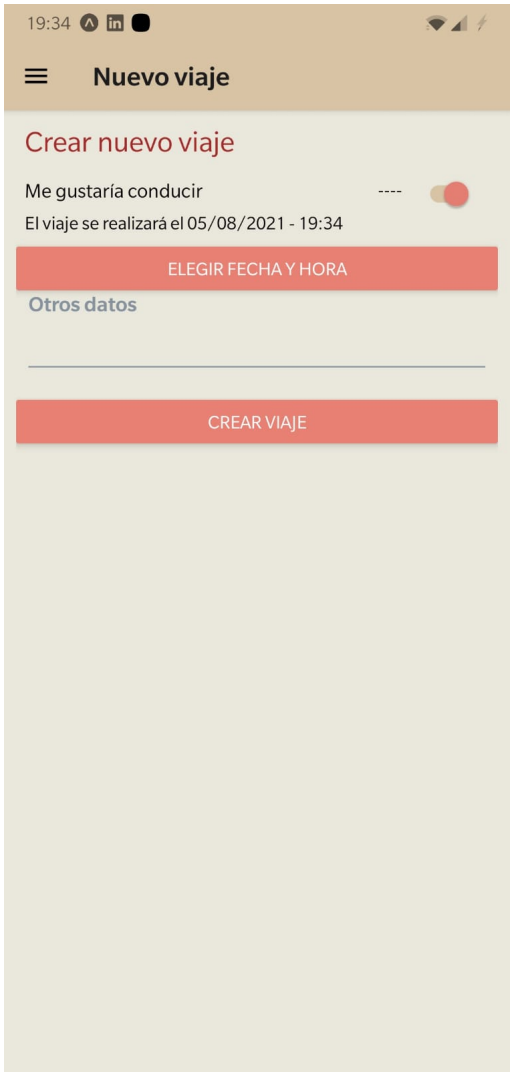


Figura 14: Pantalla de crear viaje.

Finalmente, seleccione el botón *Crear Viaje* y se creará un nuevo viaje entre ambos, aceptado por usted y pendiente de confirmación por parte del otro usuario.

B.5.7. Ver sus viajes

Si desea ver el listado de sus viajes, seleccione el botón *Mis Viajes*.

Se le mostrará un listado de viajes que tiene planeados, tanto creados por usted como por otros usuarios, mostrando primero los viajes más recientes (*Figura 15*).

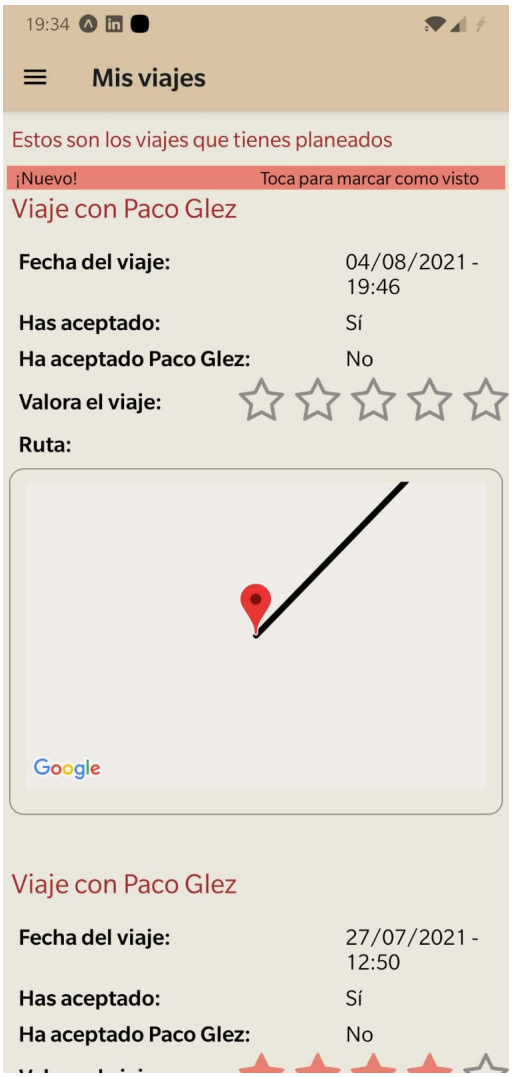


Figura 15: Pantalla de ver mis viajes.

Aquí podrá ver, para cada viaje, la ruta que tiene asignada, su compañero, la fecha de salida, quiénes aceptaron, la valoración actual y la información adicional en caso de haberla.

B.5.8. Aceptar un viaje

Si desea aceptar un viaje de los que vio en el apartado anterior, simplemente seleccione *Aceptar Viaje* (*Figura 15*).

Esto marcará el viaje como aceptado, haciéndoselo saber al otro usuarios.

B.5.9. Valorar un viaje

Si desea valorar un viaje de los que vio en el aparatado *Ver sus viajes*, simplemente toque el número de estrellas que desea asignarle (*Figura 15*).

Esto actualizará la valoración de viaje, y alterará la valoración media de su compañero de viaje.

B.6. Mis chats

En esta sección se le mostrará cómo mantener conversaciones con otros usuarios a través del chat interno de la aplicación.

Esto incluye:

1. Ver la lista de sus conversaciones.

2. Ver una conversación concreta.

3. Enviar un mensaje.

Para realizar las acciones que se describen posteriormente, debe seleccionar el botón *Mis Chats* en la pantalla de inicio (*Figura 3*).

B.6.1. Ver la lista de sus conversaciones

Tras seleccionar el botón *Mis Chats* en la pantalla de inicio (*Figura 3*) se le mostrará el lista de sus conversaciones (*Figura 16*).

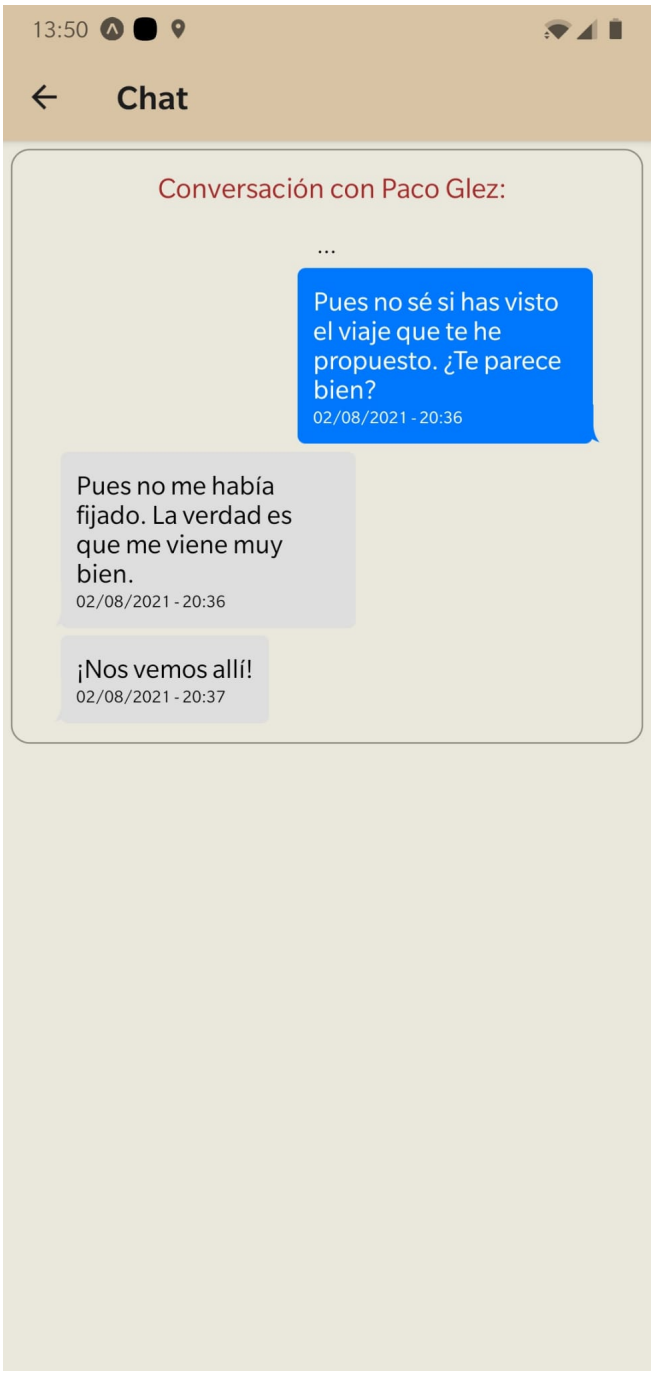


Figura 16: Pantalla de ver conversaciones.

Para cada conversación se muestra quién es el otro usuario y los últimos tres mensajes.

B.6.2. Ver una conversación

Si desea ver una conversación en concreto de las listadas en el apartado anterior, simplemente tóquela.

Esto mostrará la conversación en concreto (*Figura 17*).

Figura 17: Pantalla de ver conversación.

En la conversación se mostrará un separador que indica qué mensajes no ha leído aún, así como la fecha y hora en la que se envió cada mensaje y quién lo envió, siendo que aparecerán a la derecha y en azul los que envió usted, y en el margen izquierdo y en blanco los mensajes recibidos.

B.6.3. Enviar un nuevo mensaje

Para enviar un nuevo mensaje en una conversación que seleccionó en el apartado anterior, simplemente escriba en mensaje en la sección inferior de la pantalla y seleccione la tecla *enter* para enviarlo.

Si desea enviar un mensaje a un usuario con el que no tiene previa conversación, deberá hacerlo según se indica en el apartado *Enviar mensaje a un compañero de viaje*.

UNIVERSIDAD DE MÁLAGA | uma.es

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

E.T.S. DE INGENIERÍA INFORMÁTICA