Trabajo de Fin de Grado

CoDrive

Grado en Ingeniería Informática

Guillermo Mauro Marion López

TUTORIZADO POR:
María Dolores Afonso Suárez

Fecha 07/24

Agradecimientos

Quiero dar mi más sincero agradecimiento a la Universidad de Las Palmas de Gran Canaria (ULPGC) por ofrecerme la oportunidad de formarme y crecer tanto personal como profesionalmente durante estos años de estudio en la carrera de Ingeniería Informática. La calidad de la educación impartida y el apoyo constante han sido fundamentales para mi desarrollo académico.

Agradezco especialmente a mi tutora, María Dolores Afonso Suárez, por su inestimable guía, paciencia y dedicación en la supervisión de mi Trabajo de Fin de Titulación (TFT). Su conocimiento, consejos y apoyo han sido cruciales para la realización de este proyecto. Su compromiso y esfuerzo han sido una fuente constante de inspiración y motivación.

También quiero dedicar unas palabras de agradecimiento a mis padres, cuya confianza, amor y apoyo incondicional han sido la base sobre la que he construido mi camino en esta carrera. Gracias por estar siempre a mi lado, por creer en mis capacidades y por brindarme los recursos necesarios para alcanzar mis metas. Sin su constante aliento y sacrificio, este logro no habría sido posible.

A todos, gracias por haber contribuido de manera significativa a mi éxito académico y personal. Este logro es tan suyo como mío.

La educación es el arma más poderosa que puedes usar para cambiar el mundo.
– Nelson Mandela

Resumen

El proyecto Codrive, propone desarrollar una aplicación móvil para dispositivos Android que gestione el acceso y uso de vehículos ofrecidos en préstamo. La aplicación tiene como objetivo mejorar la movilidad al ofrecer una plataforma segura y eficiente para compartir vehículos, promoviendo un modelo de transporte sostenible. Se utilizará una metodología ágil para el diseño, enfocada en la experiencia del usuario, la seguridad y la facilidad de uso. Los objetivos clave incluyen facilitar el acceso a vehículos compartidos, optimizar su uso para reducir tiempos de inactividad, contribuir a una movilidad más sostenible, y ofrecer una experiencia segura mediante tecnologías avanzadas y protocolos estrictos.

Abstract

The Codrive project aims to develop a mobile application for Android devices to manage the access and usage of vehicles offered for lending. The application seeks to improve mobility by providing a secure and efficient platform for sharing vehicles, promoting a sustainable transportation model. An agile methodology will be used for the design, focusing on user experience, security, and ease of use. The key objectives include facilitating access to shared vehicles, optimizing their use to reduce idle time, contributing to more sustainable mobility, and providing a secure experience through advanced technologies and strict protocols.

Índice general

Índice de figuras

Capítulo 1

Introducción

En esta sección se describe la aplicación CoDrive como propuesta de Trabajo de Fin de Título (TFT), destacando las razones que motivaron su desarrollo. CoDrive tiene como objetivo principal facilitar el alquiler y préstamo de automóviles, promoviendo un uso más eficiente y creando un entorno de co-family para los usuarios. Además, esta plataforma ofrece a los propietarios la oportunidad de generar ingresos adicionales con sus vehículos, fomentando así la competitividad en el sector. Se aprovecha el crecimiento del uso de aplicaciones móviles para alcanzar un público más amplio y mejorar la visibilidad de la plataforma.

1.1. Descripción

En la era de la movilidad, presentamos nuestra aplicación desarrollada en Kotlin utilizando Jetpack Compose para ofrecer una experiencia fluida. CoDriving está diseñada para dispositivos Android y transforma el proceso de alquiler de coches al eliminar intermediarios, optimizando así el tiempo y esfuerzo involucrados en el proceso.

Inspirados en el modelo de Airbnb, CoDriving permite a particulares alquilar sus coches de manera fácil y segura. Por necesidades de tener un coche para un viaje de fin de semana o un vehículo para realizar diligencias diarias, esta app te ofrece la flexibilidad de buscar coches según las fechas deseadas, que cuenta con un diseño intuitivo y atractivo.

Por otro lado, no solo simplifica la búsqueda y alquiler de vehículos, sino que también integra funciones fundamentales como chat en tiempo real y un sistema de notificaciones, garantizando una comunicación eficiente entre propietarios y arrendatarios. La integración de CoDrive con las herramientas de Google permite una experiencia completa y conectada para los usuarios.

El desarrollo de CoDriving se lleva a cabo siguiendo metodologías ágiles, lo que nos permite adaptarnos rápidamente a las necesidades de nuestros usuarios y mejorar continuamente.

En conclusión, CoDriving redefine el alquiler de automóviles al combinar tecnología avanzada con un enfoque centrado en el usuario. Nuestra dedicación a la innovación y mejora

continua garantiza una plataforma confiable que satisface las necesidades de movilidad de manera eficiente y segura. Con CoDriving, facilitamos el acceso a vehículos y fomentamos una comunidad de confianza y colaboración, transformando la forma en que nos movemos y conectamos en la era digital.

1.1.1. Motivación

La motivación para desarrollar un Trabajo Final de Grado (TFT) sobre una aplicación de alquiler de automóviles para dispositivos Android se basa en la necesidad creciente de simplificar y modernizar los procesos tradicionales mediante el uso de tecnologías avanzadas. En la era digital actual, en la que los dispositivos móviles se han convertido en una extensión esencial de nuestras vidas, la ingeniería informática desempeña un papel crucial en la mejora y accesibilidad de los servicios para un público más amplio.

El interés en la utilización de tecnologías modernas, como el desarrollo de aplicaciones móviles, el uso de bases de datos en la nube y la implementación de funcionalidades como chats en tiempo real y notificaciones instantáneas, es fundamental para este proyecto. La ingeniería informática ofrece herramientas poderosas para crear soluciones innovadoras capaces de transformar sectores enteros. En este caso, el objetivo es simplificar el proceso de alquiler de automóviles, haciéndolo tan sencillo como realizar unas pocas interacciones desde un dispositivo móvil.

Por otra parte, el proyecto se alinea con las tendencias actuales de movilidad y economía compartida, donde la posesión de bienes da paso al acceso compartido. Facilitar el alquiler de vehículos no solo responde a una necesidad práctica, sino que también promueve el uso responsable de recursos y contribuye a una economía más sostenible.

En resumen, el desarrollo de una aplicación de alquiler de automóviles para dispositivos Android representa una significativa aportación al campo de la informática aplicada. Al aprovechar la tecnología para mejorar y simplificar un servicio esencial, esta propuesta no solo hace que el alquiler de automóviles sea más accesible y eficiente para todos, sino que también tiene el potencial de generar un impacto positivo en la vida diaria de las personas. Promueve una mayor movilidad y flexibilidad en un mundo cada vez más digitalizado, destacando así su relevancia y contribución a la sociedad contemporánea.

1.1.2. Objetivo principal

El objetivo principal de este trabajo es mejorar mis habilidades en el desarrollo de aplicaciones para Android. Aspiro a profundizar en el uso de herramientas específicas, dominar mejores prácticas de diseño y explorar nuevas funcionalidades de la plataforma. Además, pretendo adquirir un conocimiento más sólido sobre la arquitectura de aplicaciones móviles y la gestión eficiente de recursos. Este enfoque integral me permitirá crear aplicaciones más eficientes, innovadoras y de alto rendimiento, así como prepararme de manera óptima para enfrentar los desafíos futuros en el campo del desarrollo móvil.

Capítulo 2

Estado actual y objetivos

En esta sección, se discutirá el estado actual de la aplicación en el mercado, analizando otras aplicaciones y proyectos similares. Se explorarán las características comunes que comparten estas aplicaciones, así como las principales diferencias que distinguen a esta aplicación de las demás.

2.1. Aplicaciones

En primer lugar, Free2Move [4] Share Now es una plataforma global de carsharing que permite a los usuarios reservar vehículos fácilmente mediante una aplicación móvil. Destaca por su compromiso con la sostenibilidad urbana al ofrecer una flota que incluye vehículos eléctricos, promoviendo así el uso eficiente de recursos y una movilidad ecoamigable.

Por otro lado, Car2Go [16] es un servicio de carsharing que permite a los usuarios alquilar automóviles por minutos en múltiples ciudades globales. A través de su aplicación móvil, los usuarios pueden localizar y desbloquear vehículos disponibles para desplazamientos cortos dentro de la ciudad. La flota de Car2Go incluye una variedad de vehículos compactos y eléctricos, facilitando una alternativa flexible y conveniente al transporte público y privado tradicional, sin compromisos a largo plazo y con soluciones prácticas para el estacionamiento.

Asimismo, Zipcar [20] es otro líder en el mercado de carsharing, que ofrece a los usuarios la posibilidad de alquilar vehículos por horas o días. Zipcar se distingue por su amplia cobertura geográfica y la diversidad de su flota, que incluye desde vehículos compactos hasta camionetas y SUVs. La plataforma se enfoca en proporcionar una opción de transporte flexible para aquellos que necesitan un vehículo ocasionalmente, eliminando la necesidad de poseer uno propio y promoviendo así una movilidad más sostenible y accesible.

Finalmente, Getaround [5] ofrece un enfoque innovador al carsharing, permitiendo a los propietarios de vehículos alquilar sus coches a otros usuarios cuando no los están utilizando. Esta plataforma peer-to-peer maximiza el uso de los vehículos privados, reduciendo la cantidad de coches en las calles y ayudando a disminuir la congestión y las emisiones. Con su

aplicación móvil intuitiva, los usuarios pueden fácilmente encontrar y alquilar vehículos en su área, ofreciendo una solución flexible y económica para diversas necesidades de transporte.

A diferencia de otras plataformas mencionadas anteriormente, CoDrive se enfoca en promover prácticas eco-amigables y garantizar la accesibilidad, además de no contar con una flota propia, lo que causaría un incremento significativo del mantenimiento.

2.2. Objetivos

El propósito primordial de CoDrive, es diseñar y desarrollar una aplicación móvil destinada a la gestión de préstamos de vehículos. Esta aplicación busca ser un reflejo de la integración y aplicación práctica de los conocimientos y habilidades adquiridos en áreas clave como la programación, el diseño de sistemas, y la interfaz y experiencia de usuario (UI/UX). Como parte integral de este objetivo, se aspira a crear una solución profesional que garantice la fiabilidad del sistema, cumpliendo rigurosamente con los estándares de seguridad establecidos.

Los objetivos específicos de CoDrive, como una aplicación para la gestión de vehículos compartidos, que representan el resultado esperado de este proyecto, son los siguientes:

- ✔ **Ofrecer una plataforma accesible**: Facilitar y optimizar la gestión del uso de vehículos compartidos, asegurando que sea intuitiva y de fácil acceso para todos los usuarios.
- ✔ **Fomentar prácticas de transporte eco-amigables**: Contribuir a la reducción de la huella de carbono mediante la promoción del uso compartido de vehículos, apoyando así la sostenibilidad y el aprovechamiento eficiente de los recursos.
- ✔ **Garantizar la confiabilidad**: Asegurar que el sistema sea fiable y funcional, proporcionando una experiencia de usuario consistente y de calidad.

Así, CoDrive no solo busca ser una herramienta funcional, sino intentar promover un cambio hacia un modelo de transporte más sostenible y seguro, reflejando un compromiso con la innovación y la responsabilidad ambiental.

Capítulo 3

Competencias específicas y aportaciones del trabajo

En el desarrollo de CoDrive, se han abordado competencias específicas que se discutirán en detalle en esta sección. Este Trabajo Final de Título también examinará el impacto socioeconómico de CoDrive, destacando sus aportaciones al sector del alquiler de automóviles. Se analizará su potencial para promover el uso eficiente de recursos y generar nuevas oportunidades de ingresos para los propietarios de vehículos, contribuyendo así al desarrollo económico y social de nuestra comunidad.

3.1. Competencias cubiertas

CoDrive cumple con la competencia CI17, que asegura la accesibilidad y usabilidad de los sistemas, servicios y aplicaciones informáticas. Esta competencia se manifiesta a través de una cuidadosa consideración en el diseño y desarrollo de la aplicación, como son: Enfoque centrado en el Usuario y accesibilidad.

CoDrive cumple con la competencia CI16. Este cumplimiento se manifiesta a través de varios aspectos clave en el diseño y desarrollo de la aplicación: Metodologías de desarrollo, Ciclo de vida del software, análisis y diseño

CoDrive cumple con la competencia CI14. Este cumplimiento se manifiesta a través del uso delos estados de la app, haciendo que se lancen funciones en paralelo e hilos dependiendo del estado.

CoDrive cumple con la competencia CI7, que se refiere al conocimiento, diseño y utilización eficiente de los tipos y estructuras de datos más adecuados para resolver un problema específico. Para lograr esta competencia se ha intentado cumplir las 8 reglas de oro.

Este trabajo de TFT cumple con la competencia CI1 al abordar de forma integral el diseño, desarrollo, selección y evaluación de una aplicación informática, garantizando su fiabilidad,

seguridad y calidad. Implementando medidas de seguridad para proteger los datos, y realizado pruebas exhaustivas para garantizar su fiabilidad.

1. Selección de Estructuras de Datos Adecuadas:

    a) En CoDrive, la información necesaria para el sistema está estructurada de manera óptima.

1. Eficiencia en el Acceso y Manipulación de Datos:

    a) Se han tenido en cuenta los requisitos de rendimiento y escalabilidad al seleccionar las estructuras de datos. Se han utilizado estructuras, como hashmaps, que permiten un acceso rápido a los datos y una gestión eficiente de grandes volúmenes de información, asegurando que la aplicación sea capaz de manejar una carga de trabajo creciente.

3.2. Aportación al entorno socioeconómico

La aplicación CoDrive contribuye a la sociedad y al medio ambiente de varias maneras, lo que se traduce en una importante aportación socioeconómica y en el ahorro de recursos:

1. Plataforma Accesible para la Gestión del Uso de Vehículos Compartidos:

    a) CoDrive proporciona una plataforma accesible que facilita la gestión eficiente del uso de vehículos compartidos. Esto permite a los usuarios acceder fácilmente a vehículos disponibles cuando los necesiten, lo que optimiza el uso de recursos automovilísticos y fomenta una movilidad más flexible y sostenible.

2. Promoción de Prácticas de Transporte Eco-Amigables:

    a) Al promover el uso compartido de vehículos, CoDrive ayuda a reducir la cantidad de automóviles en circulación, lo que contribuye a disminuir la huella de carbono y a mitigar los efectos del cambio climático. Al fomentar prácticas de transporte más sostenibles y eco-amigables, la aplicación apoya los esfuerzos para proteger el medio ambiente y promover un futuro más verde.

3. Ahorro de Recursos y Optimización del Uso de Vehículos:

    a) La gestión eficiente del uso de vehículos compartidos a través de CoDrive conlleva un ahorro significativo de recursos, como combustible, mantenimiento y espacio de estacionamiento. Al permitir que múltiples usuarios utilicen un mismo vehículo en lugar de poseer vehículos individuales, se optimiza el uso de recursos automovilísticos y se reduce la congestión vial y la contaminación asociada.

En resumen, CoDrive no solo garantiza la seguridad y confiabilidad en el uso de vehículos compartidos, sino que también ofrece una solución innovadora que contribuye positivamente a la sociedad y al medio ambiente al promover la movilidad sostenible, la eficiencia en el uso de recursos y la reducción de la huella de carbono. Esta aportación socioeconómica y el ahorro

de recursos hacen de CoDrive una herramienta valiosa para las comunidades y empresas que buscan adoptar prácticas de transporte más responsables y sostenibles.

Capítulo 4

Análisis

En esta sección se llevará a cabo un análisis técnico del proyecto. Se describirán las deferentes herramientas, tecnologías, procedimientos para llevar acabo este TFT. Con este fin, se pretende que el lector se familiarizará con los diversos aspectos técnicos que se mostrarán a lo largo del documento.

4.1. Herramientas de desarrollo

4.1.1. Android Studio

Este proyecto ha sido desarrollado con Android Studio [10], una herramienta especializada en la creación de aplicaciones Android, utilizando un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés). Aunque la programación podría haberse realizado directamente con un editor de texto, el uso de estas herramientas simplifica significativamente la tarea del desarrollador, proporcionando una serie de funcionalidades que agilizan el proceso de creación de programas para Android, incluyendo la construcción de interfaces gráficas de usuario.

Anunciado por el 16 de mayo de 2013, Android Studio se basa en IntelliJ [14] IDEA de JetBrains y está disponible de forma gratuita para los desarrolladores. Compatible con Windows, Mac OS X y Linux, ofrece una variedad de herramientas y recursos que hacen que el desarrollo para Android sea más eficiente y productivo.

1. **Código Abierto:** Android es un sistema operativo de código abierto basado en el núcleo de Linux, permitiendo la personalización y modificación por parte de desarrolladores y fabricantes.
2. **Adaptabilidad Multiplataforma:** Compatible con una amplia gama de dispositivos, incluyendo teléfonos inteligentes, tabletas, relojes inteligentes, televisores y electrodomésticos, lo que lo hace versátil para diversas aplicaciones.

3. **Portabilidad de Aplicaciones:** Las aplicaciones desarrolladas en Java (o Kotlin) se ejecutan en una máquina virtual, facilitando su portabilidad a diferentes plataformas de hardware.
4. **Conectividad Permanente:** Diseñado para mantener los dispositivos conectados a Internet de manera continua, lo que facilita la sincronización de datos y el acceso a servicios en la nube.
5. **Servicios Incorporados:** Ofrece una amplia gama de servicios integrados, como localización mediante GPS, bases de datos SQL, y servicios de mensajería y notificaciones.
6. **Seguridad:** Implementa un modelo de seguridad que aísla aplicaciones en entornos separados, utilizando permisos específicos para limitar el acceso y las capacidades de cada aplicación.
7. **Optimización para Recursos:** Utiliza la Máquina Virtual Dalvik y su sucesora ART, optimizadas para operar en dispositivos con recursos limitados, garantizando eficiencia en el consumo de energía y memoria.
8. **Interfaz de Usuario:** Proporciona una interfaz gráfica rica con soporte para gráficos vectoriales, animaciones avanzadas y gráficos en 3D mediante OpenGL, mejorando la experiencia visual.
9. **Compatibilidad con Aplicaciones:** Soporta una amplia variedad de aplicaciones disponibles a través de Google Play Store y otras tiendas de aplicaciones, cubriendo un amplio rango de funcionalidades.
10. **Personalización:** Permite una considerable personalización tanto del sistema como de la interfaz de usuario, incluyendo la posibilidad de cambiar lanzadores, temas y widgets.

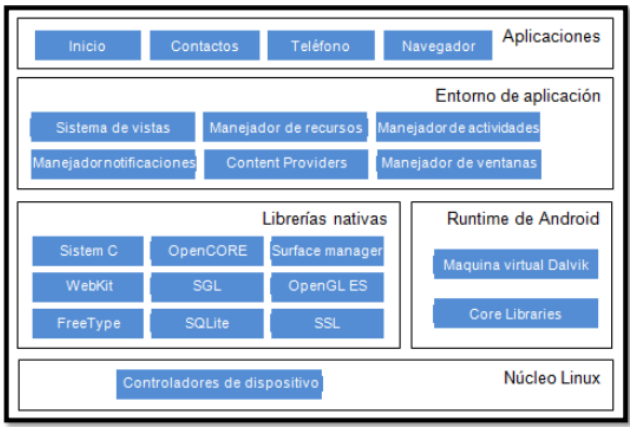

Ilustración 4.1: Estructura de S.O Android [17]

Android se estructura en capas como una cebolla como se muestra en la figura 4.1. En el centro está el núcleo Linux, que gestiona el hardware. Luego vienen las bibliotecas nativas y los servicios del sistema. Por encima, se encuentra el framework que proporciona las APIs para

desarrollar aplicaciones. Finalmente, en la capa superior están las aplicaciones que usamos a diario. Esta arquitectura modular permite una gran flexibilidad y personalización.

4.1.2. Figma

Figma [3] es una plataforma de diseño basada en la nube que permite a los equipos trabajar juntos en la creación de interfaces de usuario. Su interfaz intuitiva y sus potentes características facilitan la creación de prototipos interactivos que pueden ser compartidos y comentados por todos los miembros del equipo.

4.1.3. GitHub

GitHub [6] se basa en Git, un sistema de control de versiones distribuido que permite llevar un seguimiento detallado de todos los cambios realizados en el código fuente. Esto es crucial para cualquier proyecto de software, ya que permite:

1. Registrar el historial de cambios: Permite ver quién hizo qué cambios y cuándo, lo que facilita el seguimiento del progreso y la identificación de errores introducidos en versiones anteriores.
2. Facilita la creación de ramas crear ramas para trabajar en nuevas características o corregir errores sin afectar la rama principal del proyecto, y luego fusionar los cambios cuando estén listos.

GitHub ofrece robustas características de seguridad y respaldo de datos:

1. Repositorios privados: Con los que controlar el acceso al código.
2. Backups automáticos: GitHub mantiene copias de seguridad de los repositorios, asegurando que el trabajo no se pierda.

4.1.4. Firebase

Firebase[12] es una plataforma de desarrollo de aplicaciones móviles y web ofrecida por Google. Proporciona una variedad de servicios en la nube que incluyen autenticación de usuarios, bases de datos en tiempo realy almacenamiento de archivos, entre otros. Es ampliamente utilizada por desarrolladores para construir aplicaciones escalables y seguras sin preocuparse por la gestión de infraestructuras complejas.

4.1.5. Firestore

Firestore[13] es un servicio de bases de datos en tiempo real proporcionado por Firebase. Firestore, también conocido como Cloud Firestore, es la evolución de la base de datos Firebase

Realtime Database y ofrece una serie de características avanzadas que lo hacen ideal para el desarrollo de aplicaciones modernas.

4.1.6. Firebase Authenticator

Optamos por utilizar Firebase Authenticator[11] para la creación de usuarios en CoDrive debido a su robusta seguridad, facilidad de implementación y perfecta integración con otros servicios de Firebase. Esta herramienta nos permite ofrecer múltiples métodos de autenticación seguros y gestionar usuarios de manera eficiente, lo que mejora la experiencia del usuario al proporcionar opciones flexibles de inicio de sesión y recuperación de cuentas. Además, su integración nativa con Firestore y Firebase Database garantiza una sincronización segura y coherente de los datos, lo cual es crucial para la integridad y confiabilidad de esta aplicación.

4.1.7. Trello

Trello[2] es una herramienta de gestión de proyectos basada en tableros que utiliza un sistema visual de tarjetas para organizar tareas y flujos de trabajo. Entre sus características principales se encuentran:

- Interfaz Intuitiva y Visual: Trello se adapta fácilmente a diferentes métodos de gestión de proyectos, como es en mi caso Cascada.
- Gestión de Tareas y Proyectos: Trello es eficaz para gestionar tanto tareas individuales como proyectos completos
- Accesibilidad y Disponibilidad: Trello está disponible en múltiples plataformas, lo que facilita el acceso y la gestión del proyecto desde cualquier lugar
- Costo y Escalabilidad: Trello ofrece una opción gratuita con características robustas, lo que lo hace accesible para proyectos de todos los tamaños

4.1.8. Librerías nativas

A continuación se mostrará información a cerca de las librerías nativas de Android, sacados de [19] .

- ✔ **Gestor de superficies (Surface Manager)**: Realiza la composición de escenas 2D y 3D a través de un proceso de renderizado basado en la manipulación de mapas de bits en memoria.
- ✔ **SGL (Scalable Graphics Library)**: Esta biblioteca se utiliza en Android y en el navegador web Chrome para representar elementos en dos dimensiones. Es el motor gráfico 2D de Android.

- ✔ **OpenGL — ES (OpenGL for Embedded Systems)**: Motor gráfico 3D acelerado: OpenGL ES, adaptado a dispositivos móviles, ofrece renderizado 3D de alta calidad, aprovechando al máximo las capacidades gráficas del hardware disponible.
- ✔ **Bibliotecas multimedia**: Construido sobre OpenCORE, este conjunto de herramientas permite manejar una amplia gama de formatos multimedia, desde imágenes estáticas hasta vídeos en alta definición y audio de alta calidad.
- ✔ **WebKit**: El motor de renderización de páginas web que impulsa el navegador de Android. Además de ser utilizado como una aplicación independiente, WebKit se integra seamlessly en otras apps, ofreciendo una experiencia de navegación web consistente y eficiente. Este mismo motor es el corazón de navegadores tan populares como Google Chrome y Safari.
- ✔ **SSL (Secure Sockets Layer)**: Proporciona seguridad en las conexiones a Internet mediante criptografía.
- ✔ **FreeType**: Motor que se encarga del manejo del tipo de letra. Se encarga de transformar principalmente imágenes vectoriales de las tipografías en mapas de bits.
- ✔ **SQLite**: Se encarga de la gestión de las bases de datos relacionales.
- ✔ **Biblioteca C de sistema (libc)**: Se encarga de la ejecución de funcionalidad básica para ejecutar las aplicaciones

4.2. Las versiones de Android y niveles de API

Antes de iniciar un proyecto en Android, es esencial elegir la versión del sistema en la que se desea desarrollar la aplicación. Es importante recordar que ciertas clases y métodos solo están disponibles a partir de versiones específicas, por lo que es imprescindible saber cuál es la versión mínima necesaria si se pretende emplearlos.

Cada vez que se lanza una nueva plataforma, se mantiene la compatibilidad con versiones anteriores. Esto significa que se añaden nuevas funcionalidades y, haciendo que el ya existente pasen a ser obsoleta, pero aún se pueden seguir utilizando.

4.2.1. Distribución en el mercado de las APIs Android

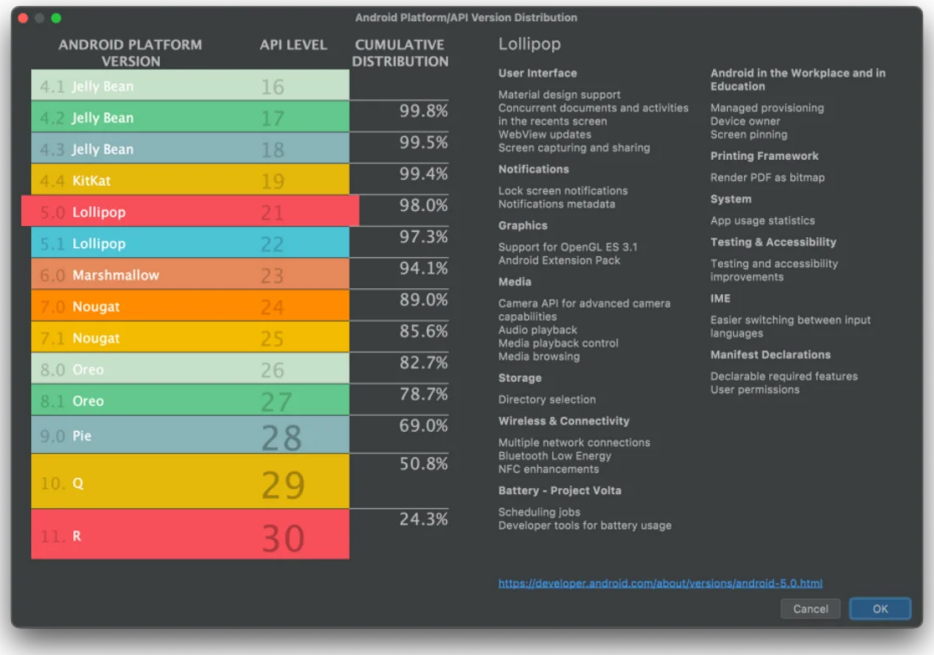

Ilustración 4.2: Estadística sobre cada versión de Android

Como se puede observar en la ilustración 4.2 Jelly Bean, KitKat y Lollipop dominan el mercado. Inicialmente, la aplicación CoDrive se empezó a desarrollar con la API 17 para garantizar soporte para el mayor número de usuarios posible, con el objetivo de maximizar su usabilidad. Sin embargo, al introducir aspectos relacionados con la interfaz de usuario para cumplir con requisitos no funcionales, se decidió migrar la aplicación a la API 5.0, a pesar de perder una pequeña cuota de mercado según se refleja en el gráfico. Esta actualización permitió obtener una versión más moderna y mejorada del software.

4.3. Requisitos

4.3.1. Requisitos funcionales

Un requisito funcional define una función del sistema de software o sus componentes. En CoDrive se han documentado los siguientes:

✔ Registro y Autenticación de Usuarios

- **Registro de usuarios:** La aplicación debe permitir a los usuarios registrarse proporcionando información básica como nombre, dirección de correo electrónico, número de teléfono y contraseña.

- **Autenticación:** La aplicación debe permitir a los usuarios iniciar sesión utilizando su dirección de correo electrónico y contraseña.
- **Cambiar contraseña:** La aplicación debe permitir a los usuarios cambiar su contraseña

✔ Perfil de Usuario

- **Gestión de perfil:**Los usuarios deben poder ver y editar su perfil, incluyendo detalles personales y datos de contacto.
- **Ver historial de alquileres:**Los usuarios deben poder ver un historial de todas sus alquileres anteriores.

✔ Búsqueda y Selección de Vehículos

- **Búsqueda de vehículos**: La aplicación debe permitir a los usuarios buscar vehículos disponibles utilizando filtros como hora de recogida, fecha y hora de devolución.
- **Visualización de detalles del vehículo:** Los usuarios deben poder ver detalles específicos de cada vehículo, incluyendo marca, modelo, año, características, disponibilidad y precio.
- **Comparación de vehículos:**Los usuarios deben poder comparar varios vehículos para tomar una decisión informada.

✔ Gestión de Reservas

1. **Notificaciones:** La aplicación debe enviar notificaciones a los usuarios sobre el estado de sus reservas, recordatorios de recogida y devolución, y cualquier cambio en la reserva.

✔ Seguridad y Privacidad

- **Protección de datos**: La aplicación debe cumplir con las regulaciones de protección de datos y asegurar la privacidad de la información personal y financiera de los usuarios.

4.3.2. Requisitos no Funcionales

Los requisitos no funcionales de CoDrive son cruciales para garantizar su rendimiento, seguridad, usabilidad y mantenimiento. A continuación se detallan algunos de los requisitos no funcionales clave:

1. **Rendimiento:**

✔ **Tiempo de respuesta:** La aplicación debe cargar y responder a las solicitudes del usuario en menos de 3 segundos.

- ✔ **Escalabilidad:** La aplicación debe poder manejar un aumento en el número de usuarios y transacciones sin degradar el rendimiento, aprovechando la capacidad de escalabilidad de Firebase.

2. **Disponibilidad:**
   - ✔ **Tiempo de actividad:** La aplicación debe estar disponible y operativa al menos el 99.9 % del tiempo.
   - ✔ **Mantenimiento planificado:** Las ventanas de mantenimiento deben ser comunicadas con antelación y programadas en horarios de baja actividad.
3. **Seguridad:**
   - ✔ **Autenticación y autorización:** Implementar métodos seguros de autenticación utilizando Firebase Authentication, como OAuth , y asegurar que solo los usuarios autorizados puedan acceder a determinadas funcionalidades.
   - ✔ **Protección de datos:** Los datos personales y financieros deben ser cifrados tanto en tránsito como en reposo utilizando Firebase Security Rules.
4. **Usabilidad:**
   - ✔ **Interfaz de usuario (UI) intuitiva:** La aplicación debe ser fácil de navegar y usar, con una interfaz amigable para el usuario diseñada en Android Studio.
5. **Compatibilidad:**
   - ✔ **Multiplataforma:** La aplicación debe ser compatible con las versiones recientes de Android.
6. **Mantenimiento:**
   - ✔ **Gestión de errores:** La aplicación debe incluir un sistema robusto para la captura, registro y notificación de errores.
7. **Fiabilidad:**
   - ✔ **Recuperación ante fallos:** La aplicación debe contar con mecanismos de recuperación para garantizar la continuidad del servicio en caso de fallos, aprovechando las capacidades de recuperación de Firebase.
   - ✔ **Pruebas automatizadas:** Implementar un conjunto completo de pruebas automatizadas en Android Studio para asegurar que las funcionalidades existentes no se vean afectadas por nuevas actualizaciones.
8. **Capacidad de configuración:**
   - ✔ **Personalización:** Permitir a los administradores configurar aspectos de la aplicación sin necesidad de cambios en el código fuente.
9. **Experiencia del usuario (UX):**

- ✔ **Rapidez de interacción:** Minimizar los pasos necesarios para completar una reserva.
- ✔ **Feedback del usuario:** Proveer retroalimentación clara y rápida sobre las acciones del usuario, como confirmaciones de reserva y notificaciones de errores.

4.4. Caso de uso

A continuación se mostrará los conceptos básicos de un diagrama de casos, como se muestra en la ilustración 4.3 :

- ✔ **Actor:** En los casos de uso, un actor es una entidad externa que interactúa con el sistema. Representa un rol, no una persona específica, y puede ser un usuario, otro sistema o un dispositivo. Los actores están fuera del sistema y realizan acciones para lograr objetivos específicos mediante la interacción con el mismo. Existen actores primarios, que inician la interacción, y secundarios, que apoyan el proceso. Por ejemplo, un Çliente "que realiza una compra o un "Sistema de Pago"que procesa la transacción. En esencia, los actores son entidades que intercambian información con el sistema para cumplir sus metas.

  En nuestro caso, el actor tiene un único rol: el de usuario. Este usuario puede desempeñar tanto el papel de arrendatario como de arrendador. La aplicación no distingue entre estos dos roles, permitiendo al usuario realizar las acciones de ambos con un solo rol.

- ✔ **Caso de uso, o comportamiento:** Un caso de uso representa una unidad funcional coherente de un sistema, subsistema o clase, donde uno o más actores interactúan con el sistema para llevar a cabo ciertas acciones.

- ✔ **Asociación entre actores y casos de uso:** Asociación es una línea que conecta un actor con un caso de uso, indicando la interacción entre el actor y el sistema. Representa una relación donde el actor participa en el caso de uso. Un caso de uso describe una secuencia de acciones que el sistema realiza para proporcionar un resultado observable y valioso para un actor. En resumen, la asociación muestra quién interactúa con el sistema, mientras que el caso de uso detalla cómo el sistema cumple con una necesidad o objetivo del actor.

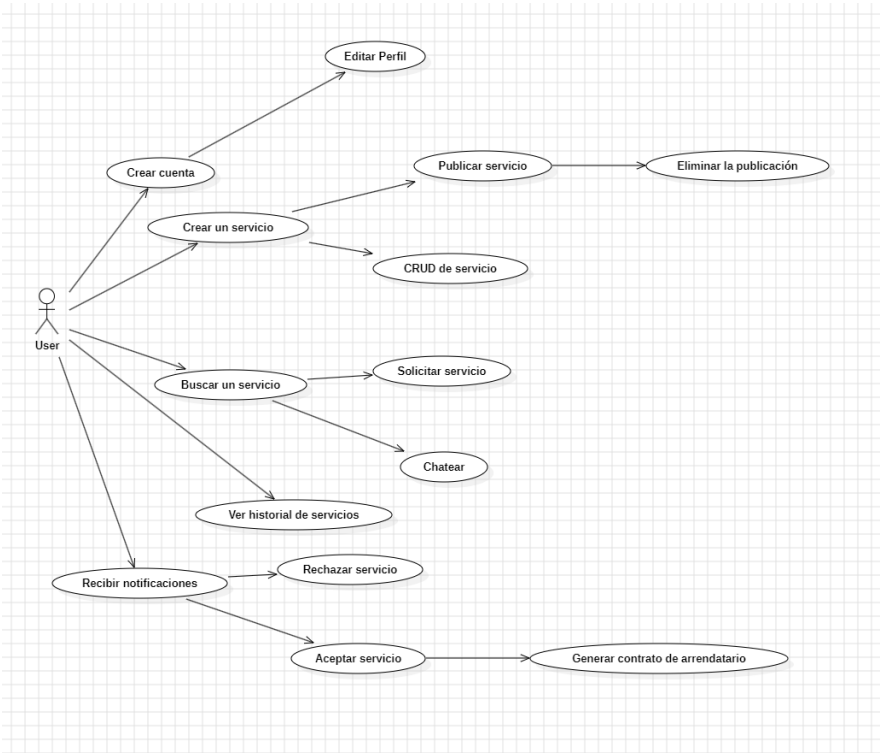

Ilustración 4.3: Diagrama de caso de uso

A continuación, se presentarán algunos casos de uso detallados. Para cada caso de uso se proporcionará una descripción completa, identificando el actor implicado, el flujo de eventos, las precondiciones y las postcondiciones.

4.4.1. Crear cuenta

1. Breve Descripción
    - El actor podrá crearse una cuenta con su correo, nombre, contraseña, localización, teléfono
- Actor
    - Usuario
- Flujo de eventos
    - El actor en vez de loguearse decide crearse una cuenta
- Precondiciones
    - El actor no tendrá ninguna cuenta asociada a los datos introducidos
- Postcondiciones
    - El actor se crea una cuenta con éxito

4.4.2. Crear servicio

1. Breve Descripción
    - El usuario podrá añadir servicios de coches de alquiler
- Actor
    - Usuario
- Flujo de eventos
    - El actor después de loguearse tendrá la opción de agregar publicaciones de coches de alquiler
- Precondiciones
    - El actor necesita estar registrado
- Postcondiciones
    - El actor crea una publicación exitosamente

4.4.3. Búsqueda de servicio

1. Breve Descripción
    - El usuario buscar automóviles según su localización fecha y hora de recogida, dejada y rango de días
- Actor
    - Usuario
- Flujo de eventos
    - El actor irá la homepage donde se encontrará los distintos filtros de búsqueda.
- Precondiciones
    - El actor necesita estar registrado
- Postcondiciones
    - El actor realizará una búsqueda exitosa

4.4.4. Editar Perfil

1. Breve Descripción

    a) El actor podrá editar los distintos campos de perfil, nombre, localización, foto de perfil, contraseña.

2. Actor.
    a) Usuario.
3. Flujo de eventos.
    a) El actor accede a tu perfil y cambia los campos de cada atributo.
4. Precondiciones
    a) El actor tiene que estar registrado previamente para poder modificar su perfil
5. Postcondiciones
    a) El actor modifica su perfil con éxito

4.4.5. Recibir notificaciones

1. Breve Descripción
    a) El actor podrá recibir notificaciones acerca de la app
2. Actor
    a) Usuario
3. Flujo de eventos
    a) En caso de que tenga alguna noticia acerca del usuario, este recibirá una notificación
4. Precondiciones
    a) Tener algún evento asociado al este usuario
5. Postcondiciones
    a) El usuario visualizará la notificación

4.4.6. Rechazar notificación

1. Breve Descripción
    a) El actor podrá rechazar o eliminar notificaciones
2. Actor
    a) Usuario
3. Flujo de eventos
    a) El usuario al tener notificaciones podrá eliminar la o rechazarla en caso de petición de servicio.

4. Precondiciones
    a) Tener alguna notificación
5. Postcondiciones
    a) El usuario elimina o rechaza con éxito la notificación, en caso de rechazarla le enviará un mensaje del motivo al emisor.

4.5. Metología

En el desarrollo de este Trabajo de Fin de Grado (TFT) se ha empleado la metodología en cascada, un enfoque de gestión de proyectos caracterizado por una secuencia lineal y estructurada de fases. Esta metodología se compone de etapas claramente definidas que incluyen la especificación de requisitos, el diseño del sistema, la implementación, las pruebas, el despliegue y el mantenimiento.

Inicialmente, se realizó un análisis de los requisitos del sistema, asegurando una comprensión detallada de las necesidades y expectativas del usuario final. Posteriormente, se procedió al diseño del sistema, donde se elaboraron los modelos y arquitecturas necesarios para guiar la implementación.

La fase de implementación consistió en el desarrollo del código y la integración de los diferentes componentes del sistema, seguido por una etapa de pruebas rigurosas para verificar la funcionalidad y corregir posibles errores. Finalmente, el sistema fue desplegado y se estableció un plan de mantenimiento para asegurar su correcto funcionamiento a lo largo del tiempo.

La adopción de la metodología en cascada ha permitido un enfoque ordenado y sistemático en el desarrollo del proyecto, facilitando el control y la gestión de cada una de las fases y garantizando la calidad y la integridad del producto final.

4.5.1. Implementación de la Metodología en Cascada en el Proyecto

El desarrollo de este proyecto se ha dividido en un conjunto de fases:

1. **Fase de Requisitos:**
    - ✔ **Especificación de Requisitos:** Se realiza una recopilación y análisis de los requisitos del sistema. Esta fase incluye la definición de todas las características y funcionalidades necesarias para completar la aplicación.
2. **Fase de Diseño:**
    - ✔ **Diseño del Sistema:** Con base en los requisitos especificados, se elabora el diseño del sistema, que incluye diagramas de arquitectura, diseño de la base de datos, y

modelos de interfaz de usuario. Esta fase asegura una planificación detallada de cómo se implementará cada funcionalidad.

3. **Fase de Implementación:**
   - ✔ **Desarrollo del Sistema:** Se lleva a cabo la codificación del sistema conforme al diseño previamente elaborado. Se siguen buenas prácticas de desarrollo.
4. **Fase de Despliegue:**
   - ✔ **Implementación del Sistema:** Una vez que el sistema ha sido probado y aprobado, se despliega en el entorno de producción. Esta fase también puede incluir la capacitación de los usuarios finales y la preparación de la documentación necesaria.
5. **Fase de Mantenimiento:**
   - ✔ **Mantenimiento y Soporte:** Después del despliegue, realizar actualizaciones y mejoras, y asegurar el correcto funcionamiento del sistema a lo largo del tiempo.

4.6. Planificación

4.6.1. Primera fase del desarrollo del TFT (Semana 1-2)

4.6.1.1. Familiarización con Jetpack Compose

La primera fase del desarrollo de este Trabajo de Fin de Título (TFT) se ha centrado en la familiarización con Jetpack Compose. Hasta ahora, he desarrollado aplicaciones en Android Studio utilizando XML para el diseño de interfaces de usuario. No obstante, con la evolución de las tecnologías y las mejoras significativas que Jetpack Compose ofrece, he decidido migrar a este nuevo enfoque declarativo para crear interfaces más dinámicas y modernas.

Durante esta fase, me dediqué a aprender y entender los principios y componentes básicos de Jetpack Compose. Esto incluyó la creación de pequeñas aplicaciones de prueba para experimentar con los elementos de la interfaz de usuario, como `Text`, `Button`, `Row`, `Column`, entre otros que Compose ofrece. Además, investigué y apliqué las mejores prácticas para gestionar estados y efectos secundarios en Compose, garantizando que las interfaces no solo sean atractivas, sino también eficientes y reactivas.

4.6.1.2. Diseño de Páginas con Figma

Simultáneamente, utilicé Figma para diseñar las páginas de la aplicación. En esta etapa, esbocé las distintas pantallas y flujos de usuarios que la aplicación tendrá. Desde la pantalla de inicio hasta las diversas secciones y funcionalidades específicas del TFT, cada elemento visual fue cuidadosamente diseñado para asegurar una experiencia de usuario intuitiva y agradable.

Asimismo, aproveché las capacidades de Figma para crear prototipos interactivos que simulan la navegación dentro de la aplicación. Esto no solo facilitó la visualización del producto final, sino que también permitió realizar.

4.6.1.3. Planificación y Organización con Trello

Finalmente, una parte crucial de esta primera fase fue la planificación y organización del desarrollo utilizando Trello. La creación de un tablero en Trello me permitió dividir el proyecto en etapas y tareas manejables, asegurando un seguimiento claro y estructurado del progreso. Cada tarjeta en el tablero representaba una tarea o funcionalidad específica que necesitaba ser completada, y las listas organizaban estas tarjetas según el estado de su desarrollo: *To Do*, *In Progress*, y *Done*.

Algunas de las tareas planificadas incluyeron:

- ✔ Investigación y aprendizaje de Jetpack Compose.
- ✔ Diseño de mockups y prototipos en Figma.
- ✔ Implementación de las primeras pantallas en Jetpack Compose.

4.6.2. Segunda fase del desarrollo del TFT (Semana 3-10)

Después de finalizar la fase de investigación y diseño preliminar, se procederá al desarrollo de la aplicación. Esta etapa incluirá:

4.6.2.1. Preparación del entorno de desarrollo

Se configurará un entorno de desarrollo óptimo utilizando Android Studio y las herramientas específicas necesarias. Se procederá con la instalación y ajuste de dichas herramientas, asegurando que el entorno esté completamente preparado para el uso de Jetpack Compose y Firestore.

4.6.2.2. Creación de la Base de Datos

Se diseñará la estructura de la base de datos en Firestore y se desarrollarán los scripts necesarios para la creación de colecciones, documentos y consultas. Esto abarcará la definición de esquemas de datos y la configuración de reglas de seguridad para asegurar un acceso controlado y eficiente.

4.6.2.3. Creación de la interfaz de usuario

Se realizará el desarrollo de la UI para la aplicación utilizando la biblioteca Jetpack Compose. Este proceso incluirá la creación de gráficos interactivos para la visualización de datos y la integración de diversas funcionalidades interactivas. Se asegurará que la interfaz sea intuitiva y fácil de usar, además de contar con un diseño moderno y funcional.

4.6.3. Tercera Fase: Mejoras (Semana 11-13)

Tras terminar la segunda fase de desarrollo, se procederá la optimización en la medida de lo posible y mejora. Esta fase incluirá:

4.6.3.1. Optimización del rendimiento

Se detectarán y solucionarán posibles inconvenientes de rendimiento en la aplicación, optimizando las consultas a la base de datos y refactorizando el código cuando sea necesario. Se garantizará que la aplicación mantenga un buen rendimiento y unos tiempos de respuesta razonable. Además, se implementarán pruebas de rendimiento periódicas para asegurar que cualquier nuevo desarrollo no afecte negativamente la eficiencia de la aplicación, gracias a las herramientas que nos aporta el IDE.

4.6.3.2. Ajustes de diseño

Se realizarán modificaciones en el diseño y optimizaciones en la interfaz de usuario, tomando en cuenta los comentarios de los usuarios y las mejores prácticas en diseño. Se asegurará que la interfaz sea estéticamente atractiva y que cumpla con las expectativas de los usuarios, ofreciendo una experiencia intuitiva y satisfactoria.

4.6.3.3. Añadido de funcionalidades adicionales

Se agregarán funcionalidades adicionales o mejoras identificadas durante las pruebas y revisiones del proyecto. Se asegurará de que la aplicación satisfaga todas las necesidades y expectativas de los usuarios.

Una vez completadas las fases anteriores del desarrollo de la aplicación, se procederá a la fase final, que incluirá la evaluación de la aplicación, la documentación final y el cierre del proyecto.

4.6.4. Cuarta Fase: Evaluación, Documentación Final y Cierre del Proyecto (Semana 14-15)

4.6.4.1. Evaluación de la aplicación

Se llevará a cabo una evaluación de la aplicación para asegurar que todas las funcionalidades cumplan con los requisitos especificados.

4.6.4.2. Documentación Final

Se preparará la documentación final del proyecto, que incluirá un informe detallado de todas las etapas del desarrollo, desde la investigación inicial hasta el refinamiento y mejora. Este documento final detallará el diseño de la aplicación, decisiones técnicas, problemas encontrados y cómo se resolvieron, así como los resultados de las pruebas realizadas.

4.6.4.3. Cierre del Proyecto

Se llevará a cabo el cierre formal del proyecto, que incluirá una revisión final para garantizar el cumplimiento de los objetivos del proyecto. Se realizará una reunión de cierre con el tutor del TFG para evaluar el desempeño del proyecto, discutir lecciones aprendidas y compartir las mejores prácticas identificadas. Finalmente, se archivarán todos los documentos y recursos del proyecto para futuras referencias y se formalizará la entrega de la aplicación a los usuarios o clientes finales.

A continuación, se muestra una imagen del resultado final del proceso iterativo llevado a cabo, utilizando Trello como herramienta de gestión, tal como se ilustra en la Figura 4.4.

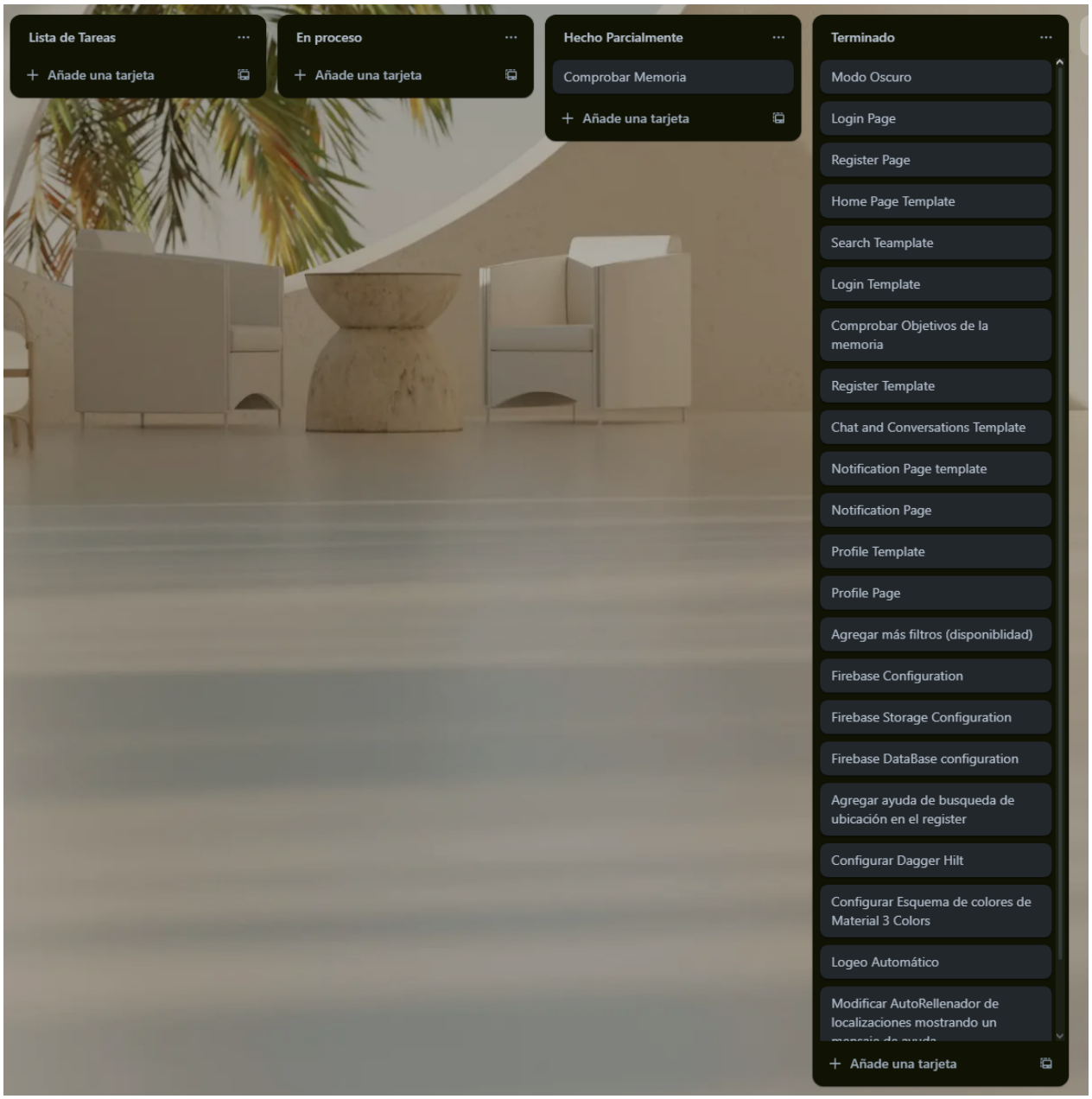

Ilustración 4.4: Resultado final del proceso iterativo en Trello

Como se puede observar, el proceso iterativo de cada tarea se dividió en cuatro partes: Lista de Tareas, En Proceso, Hecho Parcialmente y Terminado. En la imagen, solo se muestran las tareas en la columna de Terminado, ya que no se disponía de un histórico de imágenes del Trello.

Capítulo 5

Diseño

5.1. Instalación del entorno de trabajo

Antes de comenzar, es crucial configurar adecuadamente el entorno de desarrollo con Android Studio. Este entorno de desarrollo integrado (IDE) oficial de Android proporciona todas las herramientas necesarias para diseñar, programar, probar y depurar aplicaciones móviles.

Antes de proceder, es necesario asegurarse de cumplir con los requisitos previos: un sistema operativo compatible (Windows 7/8/10 de 64 bits, macOS 10.10 o superior, o una distribución de Linux como Ubuntu o Fedora), al menos 4 GB de memoria RAM (se recomiendan 8 GB), un mínimo de 2 GB de espacio libre en disco para Android Studio y 4 GB adicionales para el Android SDK, una resolución de pantalla mínima de 1280x800 y el Java Development Kit (JDK), incluido en el paquete de instalación de Android Studio.

Es importante destacar que se recomienda utilizar la versión más reciente de Android Studio, conocida como Android Studio Giraffe [1], para aprovechar las últimas mejoras y características. Además, se sugiere instalar plugins útiles como GsonFormat, que facilita la manipulación de JSON, y otros plugins que pueden optimizar el flujo de trabajo y la eficiencia del desarrollo.

5.2. Instalación y configuración del backend

Para el backend, utilizaremos Firebase. Para ello, crearemos un proyecto en Firebase Console [9]. Visita la Firebase Console y crea un nuevo proyecto. Asigna un nombre a tu proyecto, acepta los términos y condiciones, y haz clic en Çrear proyecto".

Una vez creado el proyecto, agregamos la aplicación Android. En la consola de Firebase, haz clic en *Agregar aplicación* y selecciona Android. Introduce el nombre del paquete de tu aplicación, el apodo de la aplicación (opcional) y la huella digital SHA-1. Descarga el

archivo `google-services.json` proporcionado y colócalo en el directorio `app` de tu proyecto Android.

Abre el archivo `build.gradle` en el nivel del proyecto (nivel superior) y añade las siguientes líneas en el bloque `dependencies`:

```
classpath 'com.google.gms:google-services:{última versión}'
```

Luego, en el archivo `build.gradle` del módulo (generalmente `app`), añade:

```
apply plugin: 'com.google.gms.google-services'

dependencies {
    implementation platform('com.google.firebase:firebase-bom:26.2.0')
    implementation 'com.google.firebase:firebase-firestore'
    implementation 'com.google.firebase:firebase-storage'
    implementation 'com.google.firebase:firebase-auth'
    implementation 'com.google.firebase:firebase-database'
}
```

5.2.1. Configuración del CMD para Backups

Para gestionar backups de los datos en Firebase, podemos utilizar la Firebase CLI [8]. Primero, instalamos Node.js [15] si aún no lo tenemos. Luego, abrimos el Command Prompt (CMD) y ejecutamos el siguiente comando para instalar la Firebase CLI:

```
npm install -g firebase-tools
```

Una vez instalada, iniciamos sesión en Firebase utilizando:

```
firebase login
```

Navegamos al directorio raíz de mi proyecto y ejecutamos:

```
firebase init
```

Seleccionamos Firestore, Storage, Authentication y Realtime Database para la configuración. Esto creará los archivos necesarios (`firebase.json` y `.firebaserc`) en el proyecto.

Para realizar backups de Firestore, ejecutamos el siguiente comando:

```
firebase firestore:export <path-to-backup>
```

De manera similar, podemos exportar e importar datos de Realtime Database utilizando:

```
firebase database:get / > <path-to-backup>.json
firebase database:set / <path-to-backup>.json
```

5.3. Implementación

En esta sección, se mostrarán las principales ventanas de la aplicación desarrollada, proporcionando una visión general de su diseño y apariencia. La aplicación ha sido creada para ofrecer una experiencia de usuario intuitiva y eficiente, facilitando el acceso y la gestión de la información de manera ágil. A continuación, se presenta una muestra de las principales ventanas que conforman la aplicación. Cabe destacar que, para el diseño de la app, se ha priorizado el uso de componentes de Material 3 [7] de Google, garantizando un ecosistema coherente basado en los estándares de Google:

5.3.1. Pantalla de Login y Register

A continuación se mostrarán las pantalla de inicio de sesión y registro, Ilustración 5.1 y ilustración 5.2

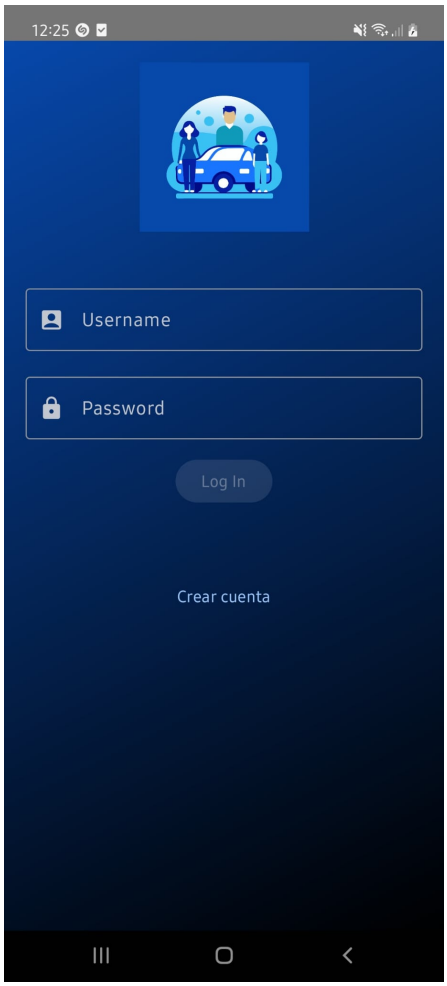

Ilustración 5.1: Pantalla de Inicio de sesión

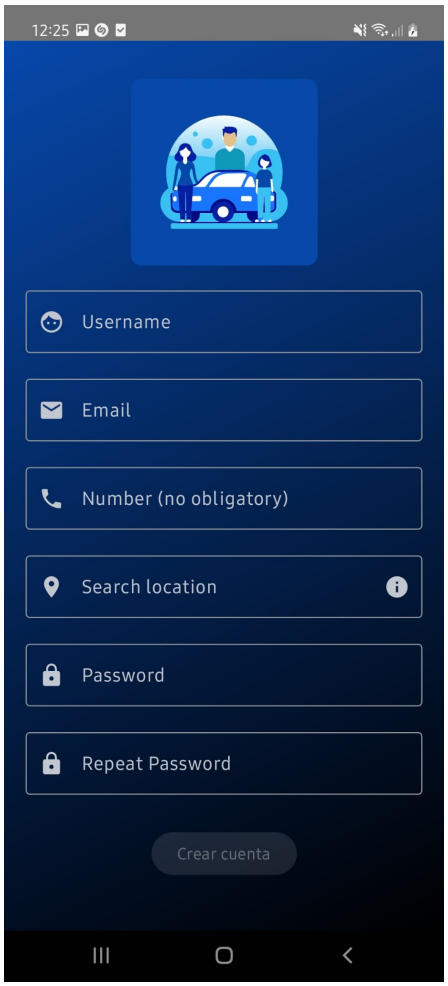

Ilustración 5.2: Pantalla de registro

5.3.2. Pantalla de Inicio

Pantalla de inicio nada más logearte o registrarte, el cual tiene filtros de búsqueda, como se muestra en las siguientes ilustraciones 5.3 y con filtros 5.4

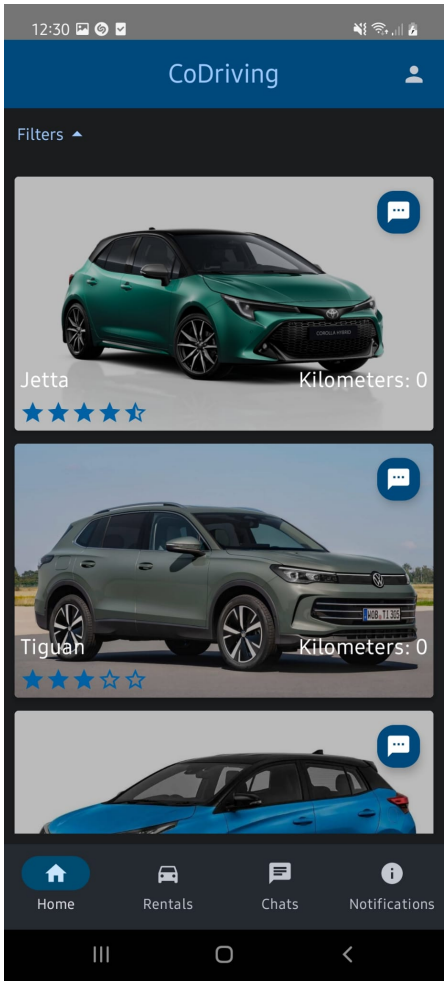

Ilustración 5.3: Pantalla de Inicio sin filtros desplegados

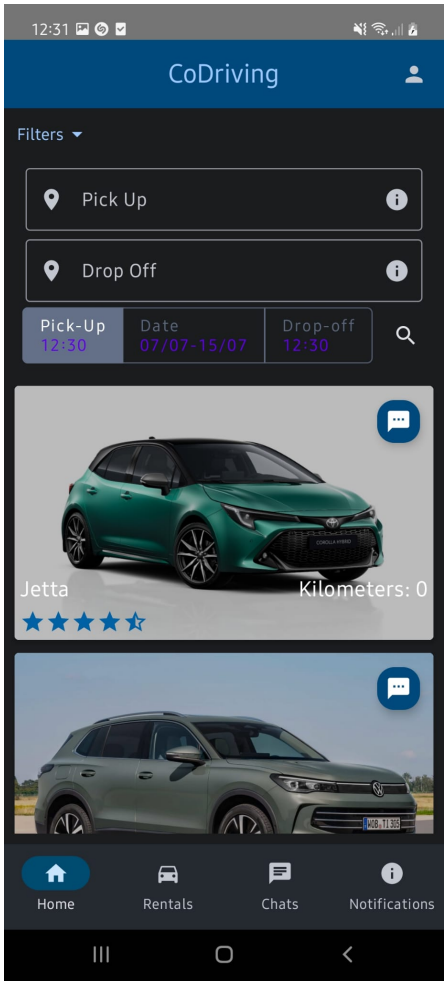

Ilustración 5.4: Pantalla de Inicio con filtros desplegados

5.3.3. Mis automóviles

Pantalla donde guardamos nuestros automóviles para posteriormente publicarlos, Ilustración 5.5

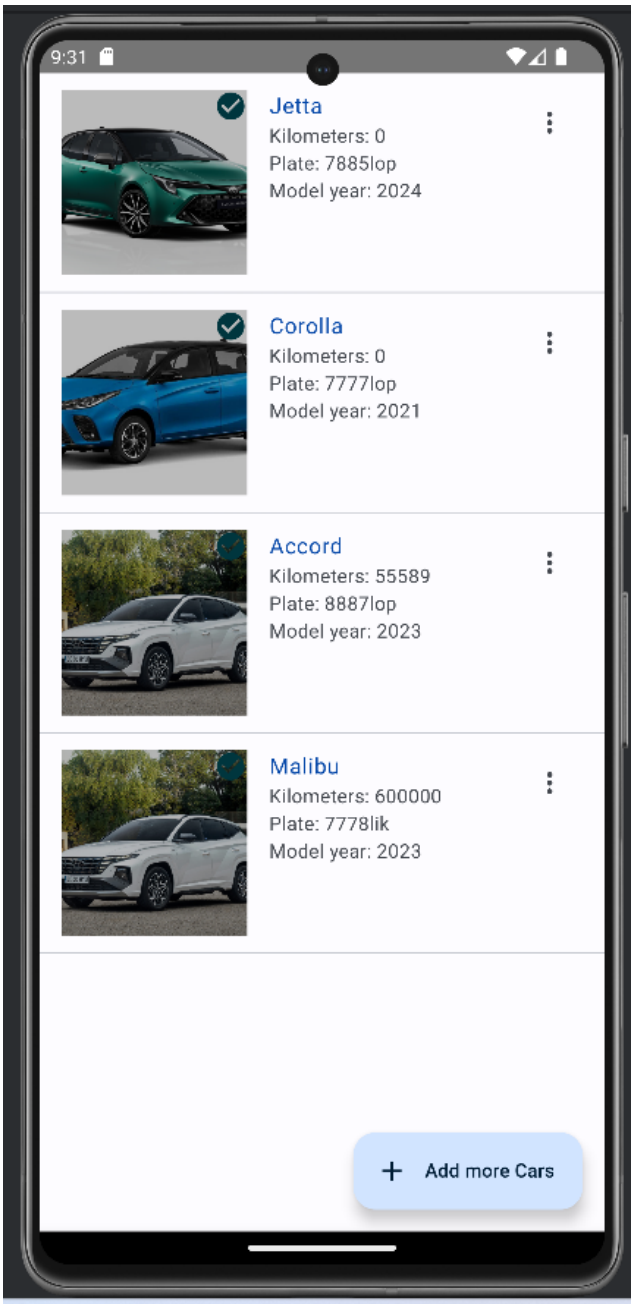

Ilustración 5.5: Mis Servicios

5.3.4. Notificaciones

Pantalla donde veremos las notificaciones informativas como peticiones de alquiler. Ilustración 5.6

Ilustración 5.6: Notificaciones

5.3.5. Chat y conversaciones

Pantalla donde veremos las conversaciones que tengamos con usuarios y dentro de cada uno el chat. Conversaciones ilustración 5.7 y chat 5.8

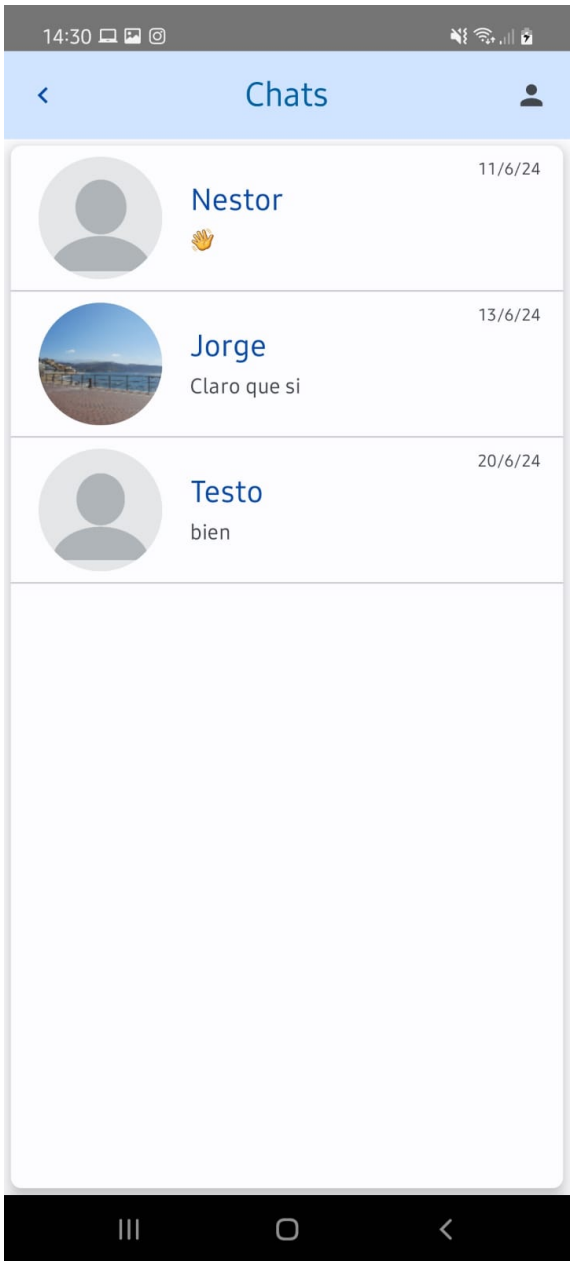

Ilustración 5.7: Pantalla de conversaciones

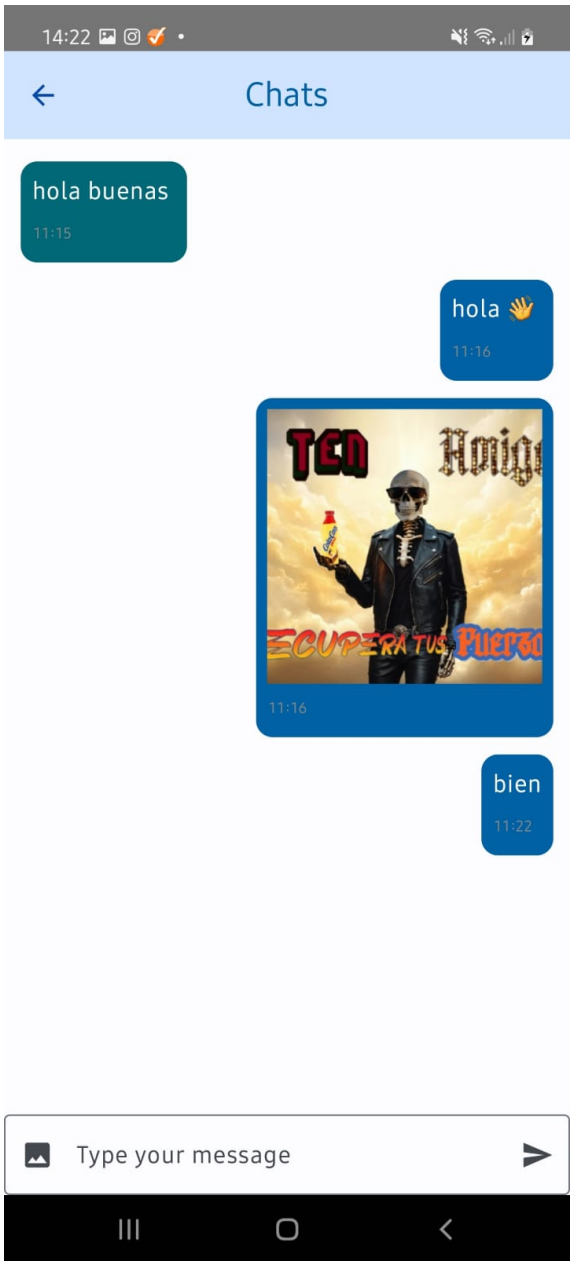

Ilustración 5.8: Pantalla de chat

5.3.6. Perfil

A continuación se mostrarán la pantalla de perfil, donde veremos más abajo nuestro historial de rentas si es que hay. Perfil ilustración 5.9 y historial de rentas 5.10

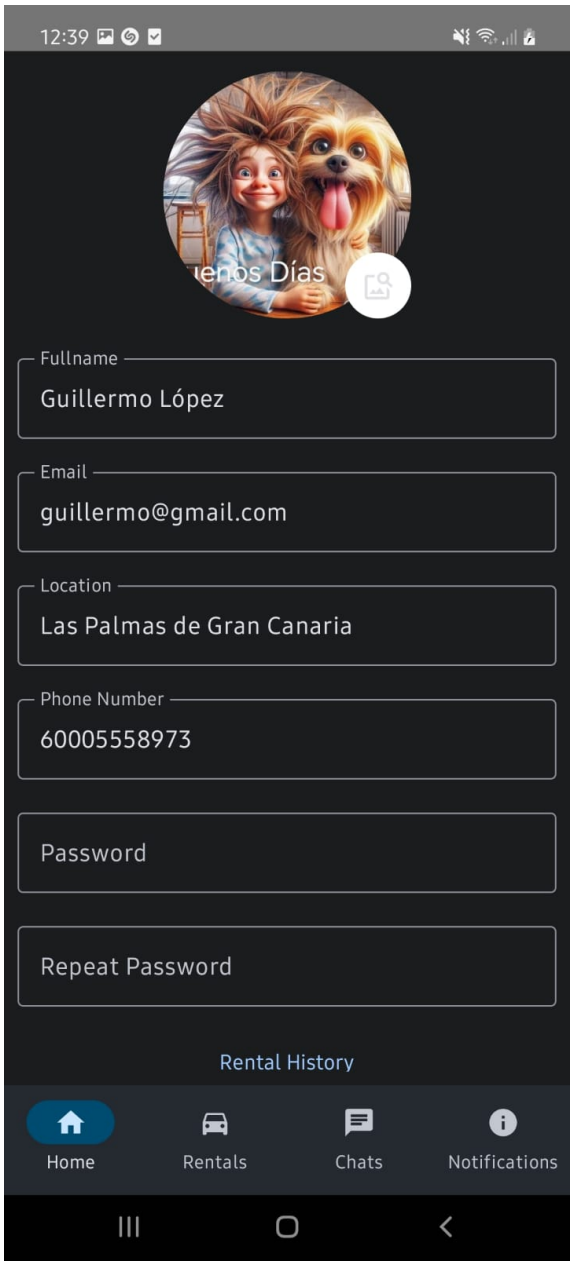

Ilustración 5.9: Pantalla de perfil

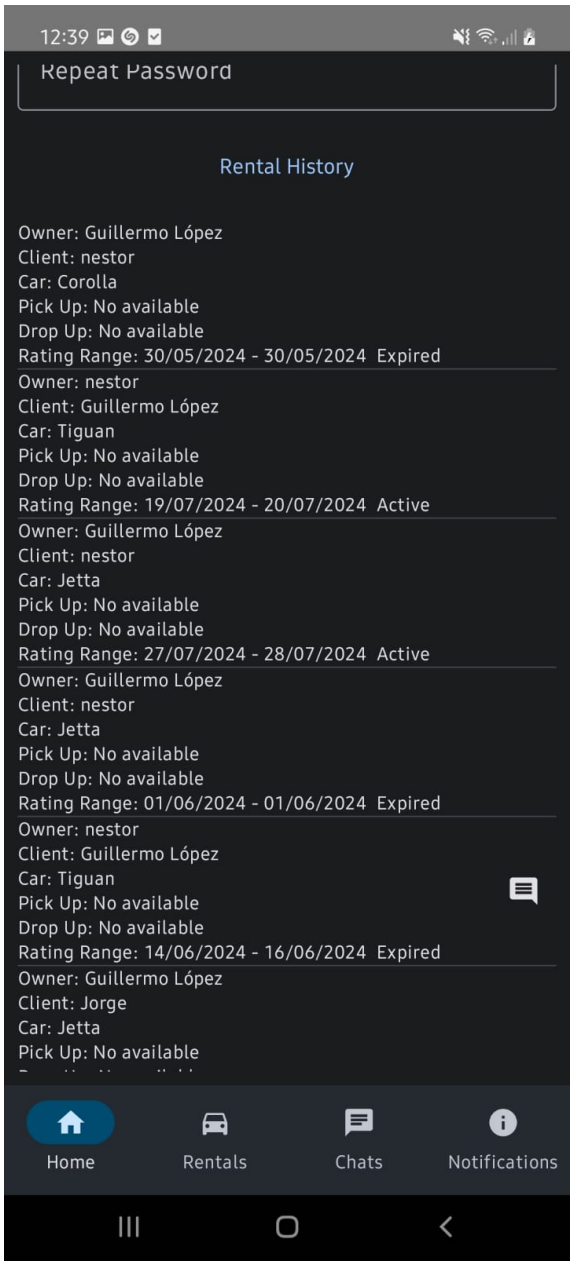

Ilustración 5.10: Historial de rentas

5.3.7. Agregar vehículo

A continuación se mostrará la página para agregar vehículos a ”Mis vehículos”5.5, como se muestra en la ilustración 5.11

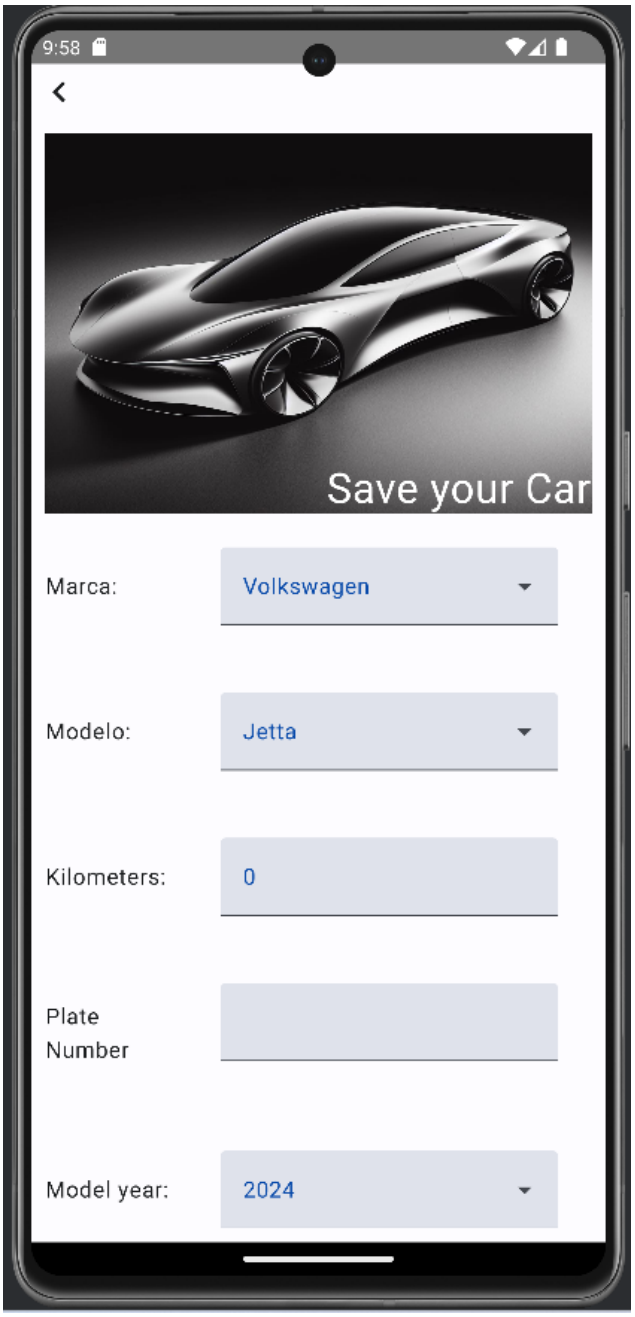

Ilustración 5.11: Pantalla de agregar servicio

5.4. Modelo de datos

En CoDrive, hemos implementado un modelo de base de datos utilizando Firebase. Este modelo ha sido cuidadosamente diseñado para optimizar el rendimiento, la escalabilidad y la facilidad de acceso a los datos. A continuación, se describe en detalle la estructura de nuestra base de datos en Firebase.

5.4.1. Estructura base de datos

La base de datos de Firestore contiene las siguientes colecciones principales: `Cars`, `RentCar`, `Users`, `carNotifications`, y `conversations`. A continuación, se describe la estructura de cada colección:

Colección Cars

Colección de Cars ilustración 5.12

✔ **Campos de la colección**:

- `owner`: Referencia al documento en la colección `Users`.
- `image`: Lista de URL de imágenes.
- `year`: Año del modelo del coche.
- `kilometers`: Kilometraje del coche.
- `model`: Modelo del coche.
- `plate`: Matrícula del coche.
- `brand`: Marca del coche.
- `enable`: Estado de habilitación del coche (booleano).
- `numberOfReviews`: Número de reseñas recibidas.
- `rating`: Calificación promedio.
- `rentCars`: Lista de referencias a documentos en la colección `RentCar`.

✔ **Subcolecciones dentro de cada documento de coche**:

- `reviews`: Campos de la colección:
  - `rating`: Calificación otorgada.
  - `comment`: Comentario de la reseña.
  - `reviewSender`: Referencia al documento del usuario que envió la reseña.

5.4.2. Colección RentCar

A continuación se mostrará la estructura de la colección Cars con sus cambios y subcolecciones, ilustración 5.13

✔ **Campos de la colección**:

- `ownerName`: Nombre del propietario.

- `endDate`: Fecha de finalización del alquiler (tipo timestamp).
- `pricePerDay`: Precio por día de alquiler.
- `startDate`: Fecha de inicio del alquiler (tipo timestamp).
- `carId`: Referencia al documento en la colección `Cars`.
- `busy`: Estado de ocupación del coche (booleano).
- `pickUpLocation` (opcional): Lugar de recogida.
- `dropOffLocation` (opcional): Lugar de devolución.

Colección Users

A continuación se mostrará la estructura de la colección users con sus campos, ilustración 5.14

✔ **Campos de la colección**:

- `email`: Correo electrónico del usuario.
- `name`: Nombre del usuario.
- `phone`: Número de teléfono del usuario.
- `profileImage`: URL de la imagen de perfil del usuario.

Colección carNotifications

A continuación se mostrará la estructura de la colección carNotifications 5.15

✔ **Campos de la colección**:

- `idNotification`: Identificador único de cada notificación.
- `idProduct`: Id del automóvil
- `idSender`: Id del emisor.
- `message`: Mensaje de la notificación
- List¡RentCars

  : Lista de alquileres interesados del automóvil
- timestamp: fecha de la creación de la notificación
- title: Titulo de la notificación
- type: tipo de notificación, si es informativa o una petición

Colección conversations

Estructura del modelo de datos conversations con subcolección chat, ilustración 5.18

✔ **Campos de la colección**:

- `lastMessage`: Contenido del último mensaje.
- `date`: Fecha y hora del último mensaje (tipo timestamp).
- `userIds`: Array que contiene los identificadores de los miembros de la conversación.

✔ **Subcolección `messages`**:

- Documentos que contienen los mensajes intercambiados, cada uno con los campos:
  - `sender`: Referencia al usuario que envió el mensaje.
  - `message`: Contenido del mensaje.
  - `timestamp`: Fecha y hora del mensaje (tipo timestamp).

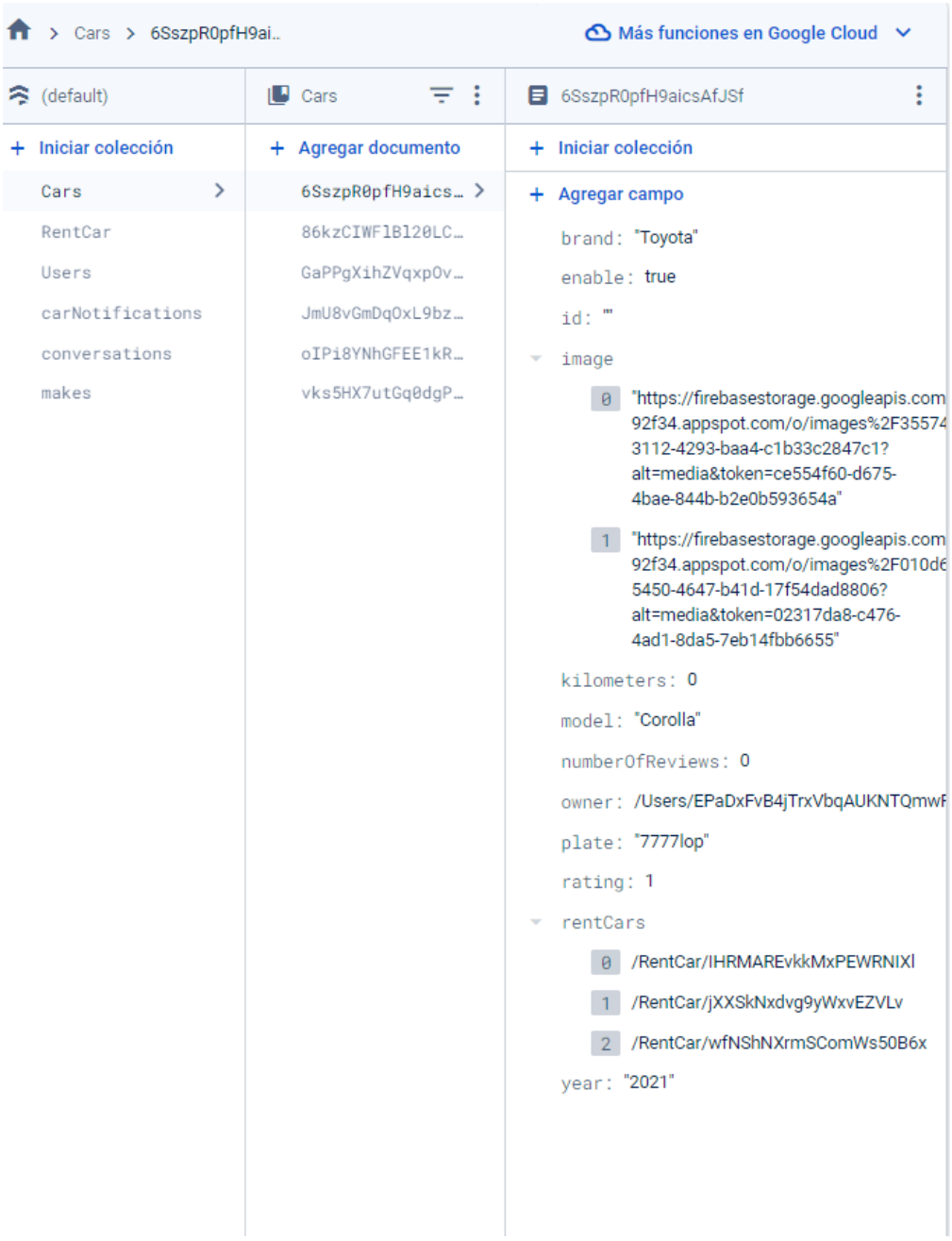

Ilustración 5.12: Modelo de los autos

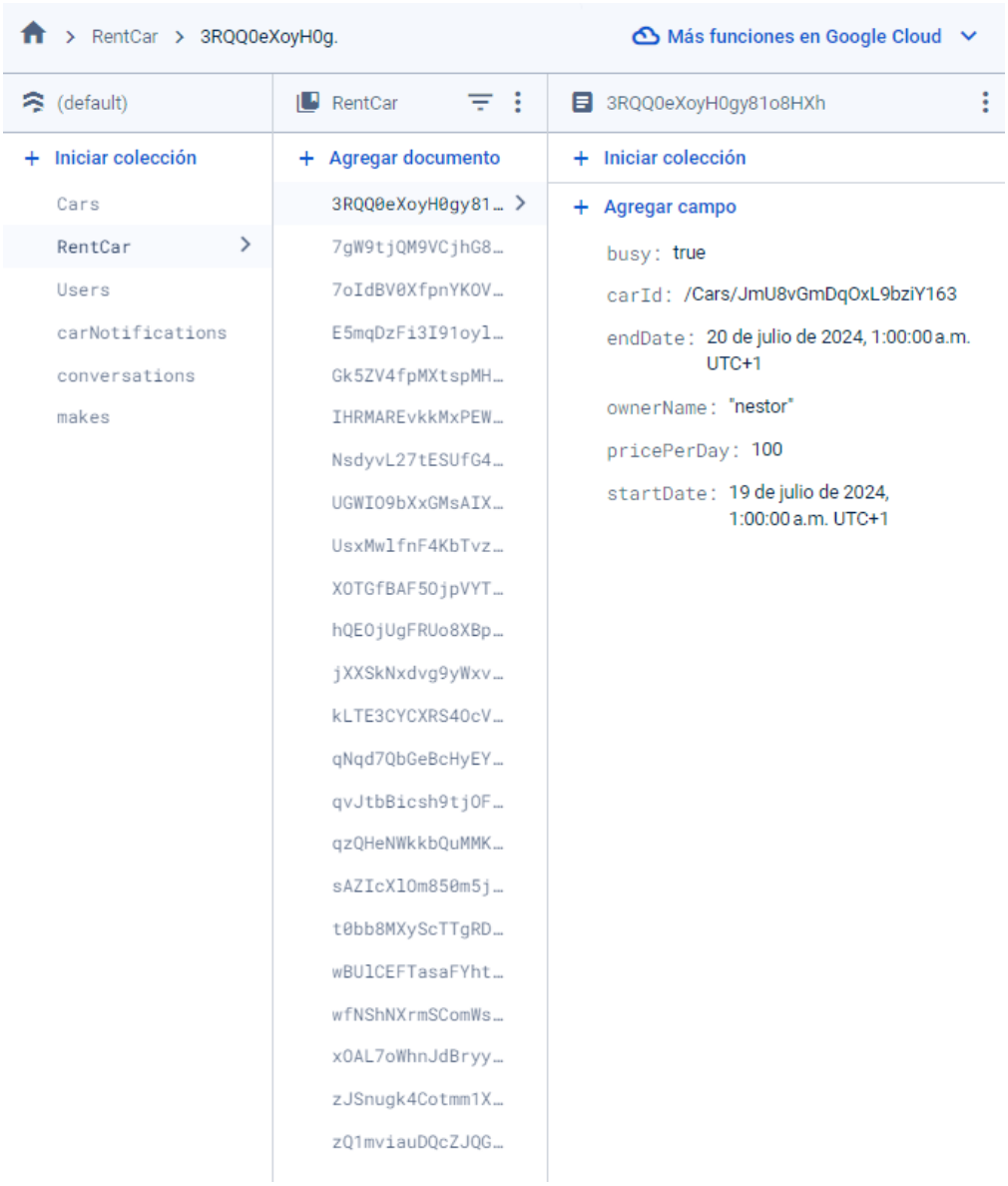

Ilustración 5.13: Modelo de los alquileres

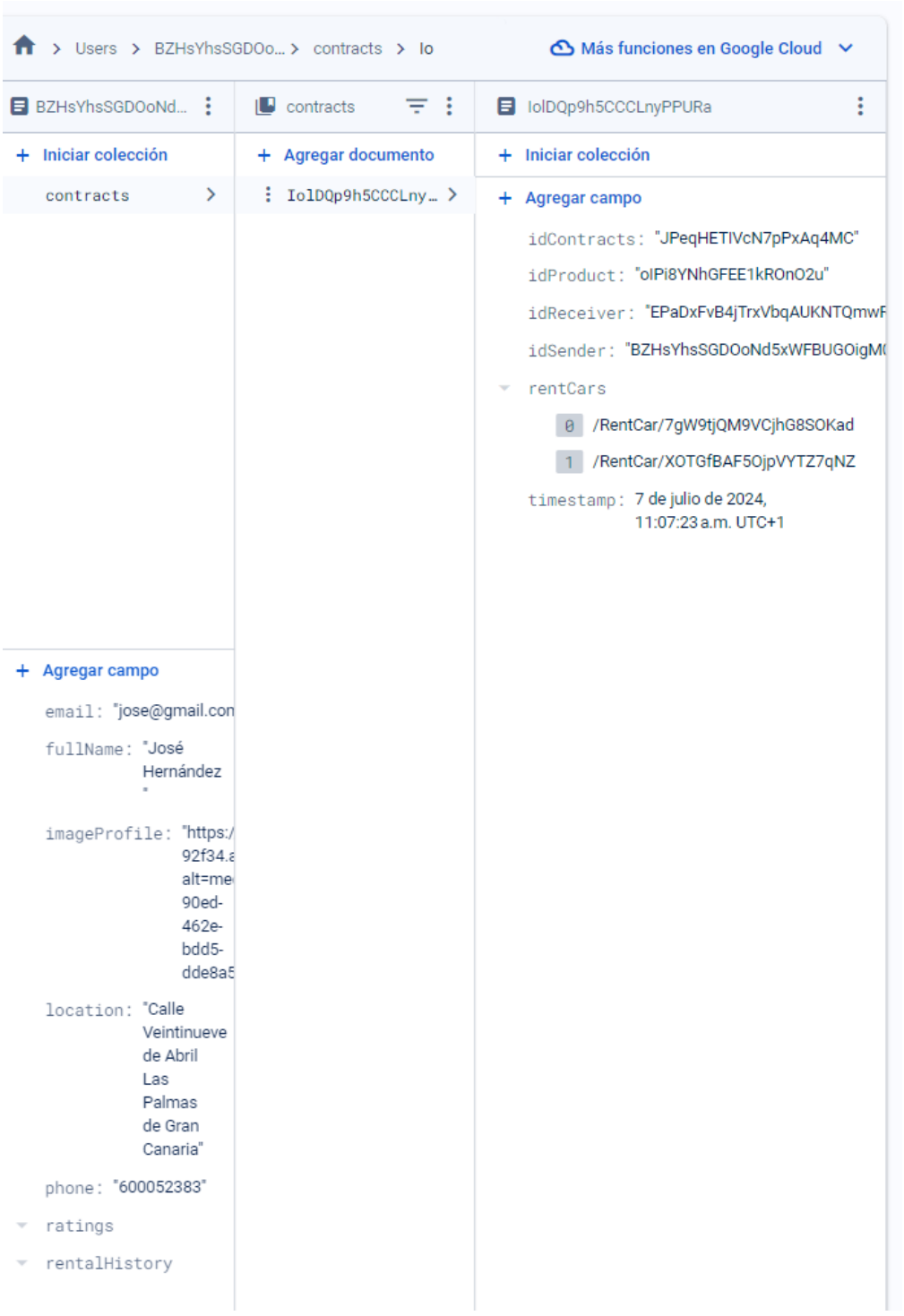

Ilustración 5.14: Modelo de los usuarios con contratos

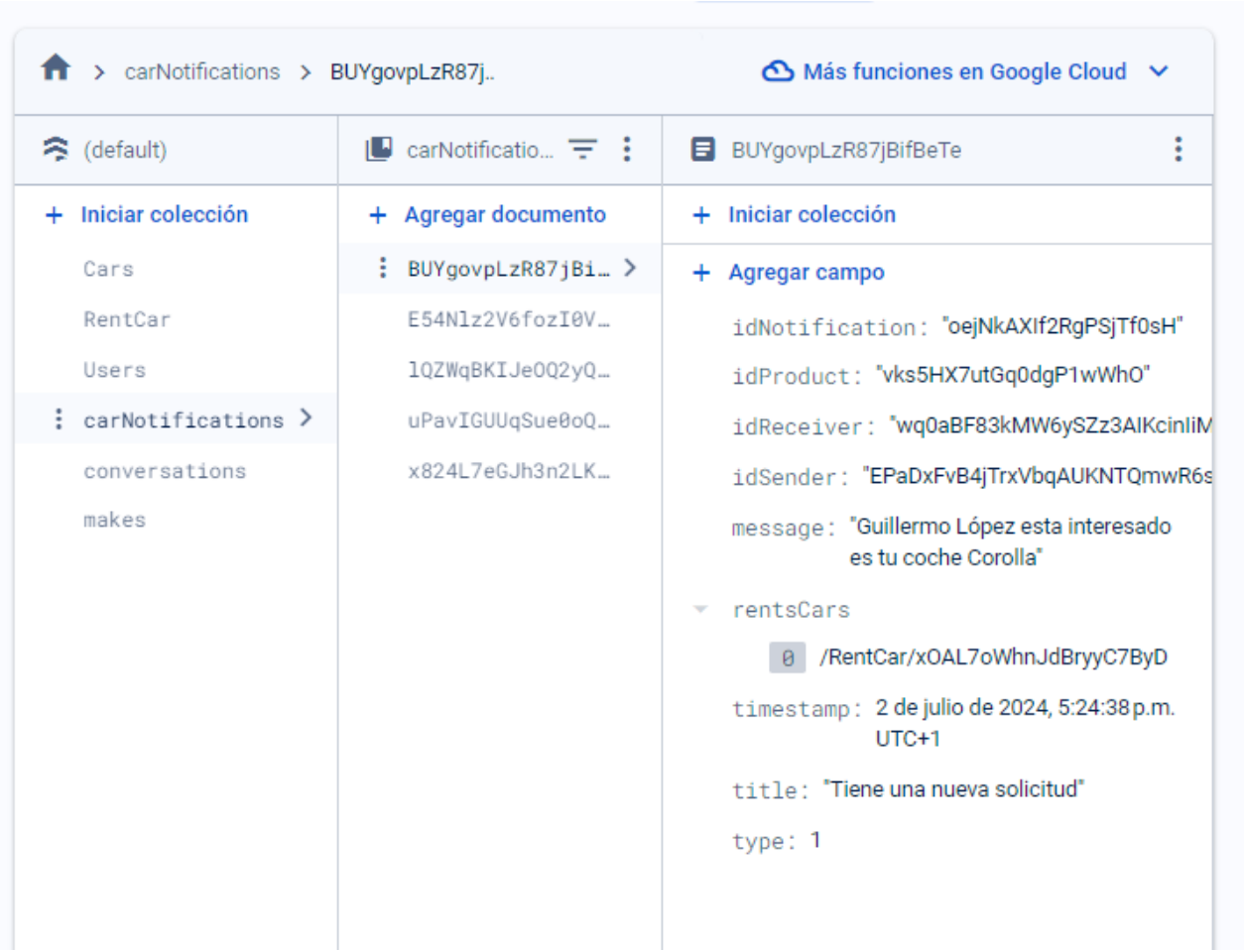

Ilustración 5.15: Modelo de las notificaciones

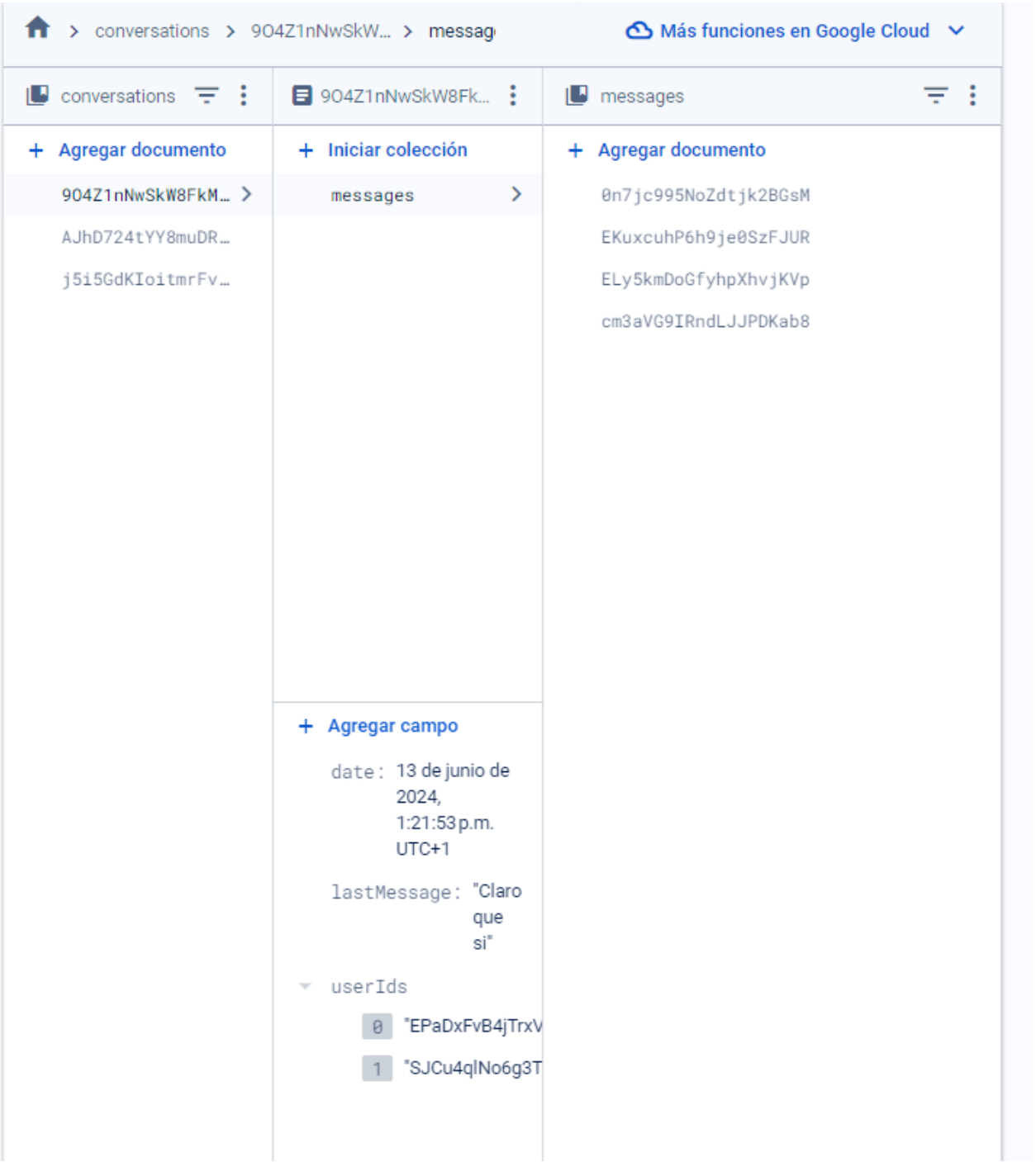

Ilustración 5.16: Modelo de conversaciones con mensajes

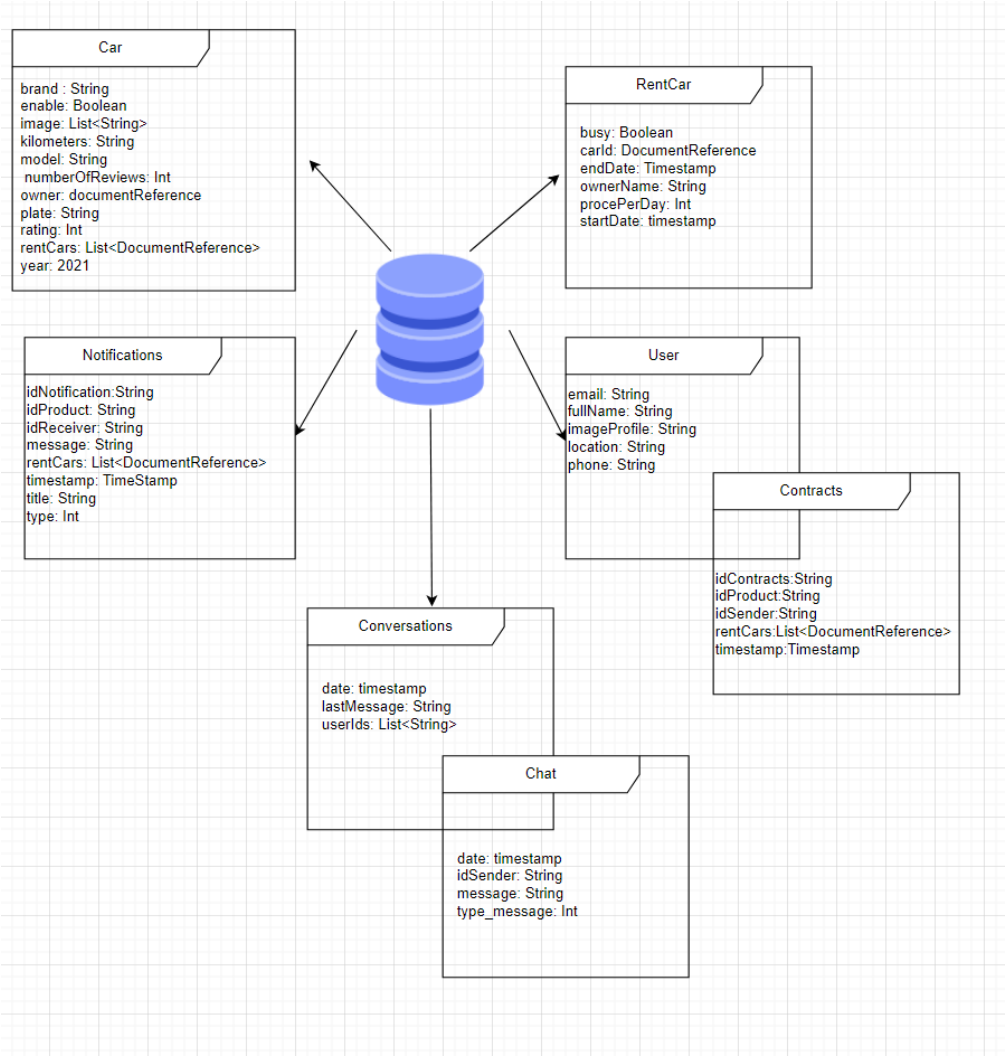

Ilustración 5.17: Estructura genérica de los modelos de la base de datos

5.4.3. Diagrama de Clases

A continuación se mostrará una estructura de los diagrama de clases.5.18:

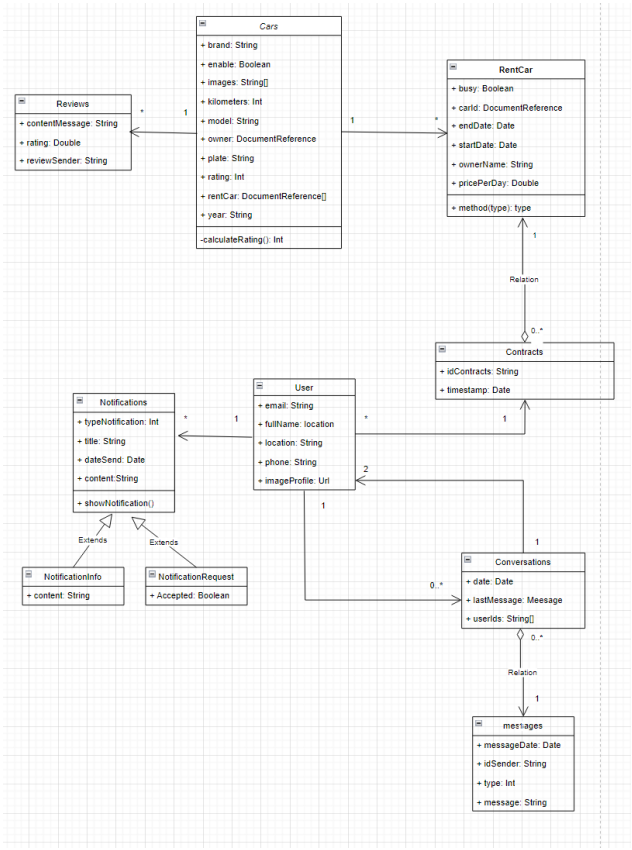

Ilustración 5.18: Diagrama de Clases

Este diagrama de clases muestra una representación básica del sistema de gestión de alquiler de autos del backend de CoDrive. Las entidades, relaciones y atributos clave se identifican y explican de manera resumida. Su organización responde a criterios de modularidad, reutilización de código, mantenibilidad y escalabilidad del sistema.

5.4.3.1. Reglas de la base de datos

Las reglas establecidas son permisos de lectura y escritura para cada entidad, controlando el acceso a la información según el estado de autenticación del usuario y la propiedad de los datos. Se prioriza la seguridad de la información confidencial mediante la restricción del acceso y la implementación de autenticación.

5.4.3.2. Desafíos encontrados

Durante el desarrollo de los modelos de datos para el backend, enfrenté el reto de diseñar una relación que permitiera a los contratos incluir múltiples alquileres de un mismo auto. El objetivo era simplificar la experiencia del usuario, permitiéndole solicitar varias instancias de alquiler en una sola petición, en lugar de tener que realizar múltiples solicitudes individuales.

Este desafío presentó varias dificultades. Al mostrar un historial de contratos, sería necesario desglosar las fechas de alquiler, complicando la presentación y gestión de la información. En contraste, si se optara por un enfoque en el que cada alquiler requiriera una solicitud independiente, se reduciría la complejidad del procesamiento y la estructura de datos, aunque esto aumentaría el número de solicitudes que el usuario debe realizar.

5.5. Patrón Modelo-Vista-VistaModelo

El patrón MVVM (Model-View-ViewModel) divide la aplicación en tres capas distintas: la Vista, que representa la interfaz gráfica; el Modelo, que maneja el tratamiento de datos; y el ViewModel, que contiene la lógica para preparar y gestionar los datos para la Vista, maneja las entradas de la Vista y actualiza los datos en el Modelo. Esta división facilita una mayor portabilidad de la aplicación y simplifica su mantenimiento. Ilustración 5.19.

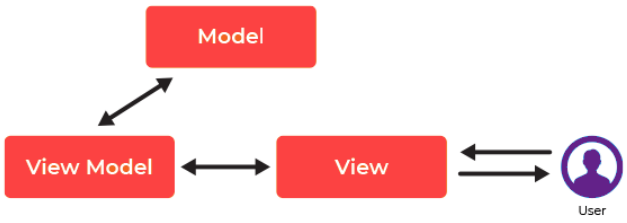

Ilustración 5.19: Diagrama MVVM

Como vemos en la ilustración 5.19, la forma en que debe trabajar este patrón es conectando la Vista, que es la parte visible para el usuario de la aplicación, con un ViewModel, que gestionará los datos de la Vista, procesará las interacciones del usuario y actualizará la interfaz de usuario de manera reactiva. A su vez, el ViewModel será el que conecte con la capa Modelo, para poder obtener y almacenar los datos necesarios. La Vista está vinculada directamente al ViewModel mediante data binding, de manera que cualquier cambio en los datos del ViewModel se refleja automáticamente en la Vista, y las acciones del usuario en la Vista se comunican al ViewModel para que este las procese y actualice el Modelo si es necesario.

5.6. Estructura planteada para CoDrive

En conclusión, la imagen.5.20 representaría la estructura de la aplicación.

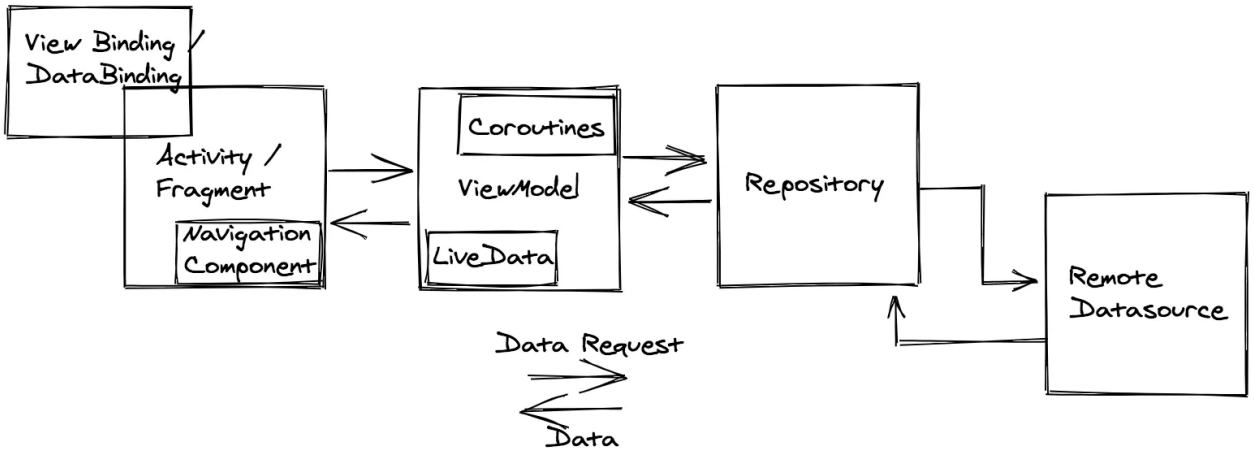

Ilustración 5.20: Arquitectura MVVM

La arquitectura Model-View-ViewModel facilita controlar el acoplamiento entre los componentes, así la app se encarga de eliminar el alto acoplamiento entre cada componente, entonces la app tendrá una capa de presentación formada conjuntamente por una actividad/fragmento (View) y un ViewModel por pantalla (o feature) cuando sea necesario. Se utilizan Coroutines para las operaciones de background (llamados al API), UI reactiva usando LiveData, gestión del estado de la UI usando Sealed classes , además de Data Binding / View Binding (sí, podemos usar ambos dentro del mismo módulo). Además se utiliza una capa de datos con un repositorio el cual se encargará de almacenar las funciones que actúan con el servidor remoto.

Capítulo 6

Desarrollo

6.1. Organización del código

Teniendo en cuenta que hemos utilizado un patrón de diseño MVVM, hemos organizado el código en una estructura de carpetas por nivel. Ilustración 6.1

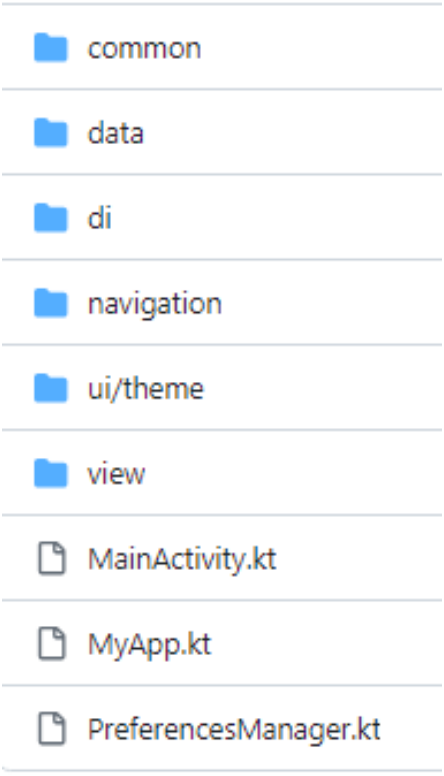

Ilustración 6.1: Estructura del código

1. **common** se encargará de almacenar aquellos componentes que se utilicen de manera

concurrente en el código

2. **data** se encargará de la capa de Modelo, el cual almacenará todos los modelos de la app y repositorios
3. di es utilizado para almacenar los modelos de Dagger Hilt el cual se encargará de la inyección de dependencias
4. **navigation** contiene la lógica de navegación entre pantallas
5. **screens** se almacena cada vista con su ViewModel correspondiente
6. **ui/theme** es una carpeta generada automáticamente que almacena los temas y colores principales de la app

6.2. Descripción Funcionalidades

En este documento se describen las funcionalidades de la aplicación desarrollada, divididas en secciones para una mejor comprensión. Se ha eliminado el contenido relacionado con las pruebas de software, centrándose únicamente en las características y beneficios que ofrece la aplicación a los usuarios.

6.2.1. Funcionalidades de la Página de Inicio de Sesión y Registro

A continuación se mostrará las funcionalidades de la pagina de inicio de sesión y registrarse ,ilustración 6.2 y 6.3

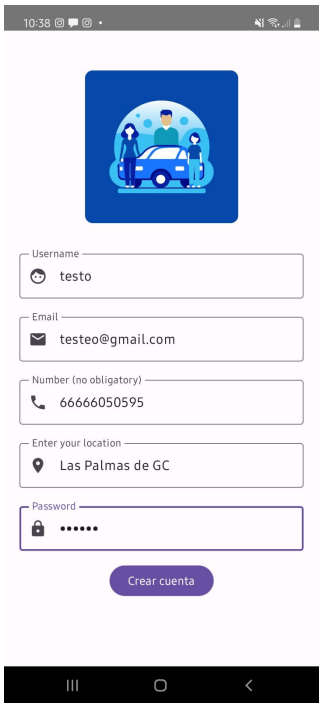

Ilustración 6.2: Pantalla de creación de usuario

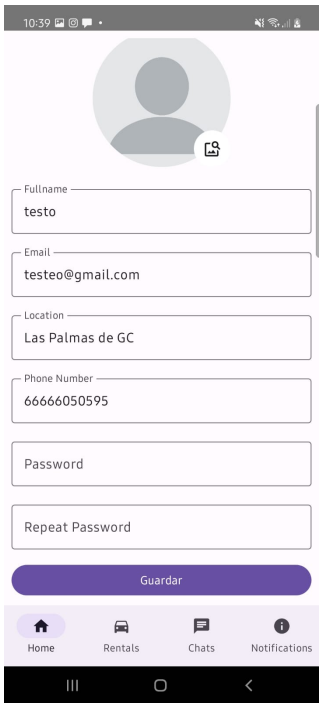

Ilustración 6.3: Pantalla de perfil una vez creado el usuario

6.2.1.1. Funcionalidades del inicio de sesión

- ✔ **Inicio de sesión y autenticación:** El usuario podrá iniciar sesión y ser autenticado correctamente.
- ✔ **Acceso a la página de registro:** El usuario podrá navegar a la página de registro si aún no tiene una cuenta.
- ✔ **Inicio de sesión automático:** Una vez que el usuario se haya autenticado, la próxima vez que inicie sesión será de forma automática.

6.2.1.2. Funcionalidades de la página de registro:

- ✔ **Registro de nuevos usuarios:** El usuario podrá registrarse si sus credenciales (correo electrónico) no existen en el sistema.
- ✔ **Modificación de perfil:** El usuario podrá actualizar su nombre, correo electrónico, localización, número de teléfono y foto de perfil.

6.2.2. Funcionalidades del página de Inicio

Funcionalidades de la Home page como se muestra en la ilustración 6.4

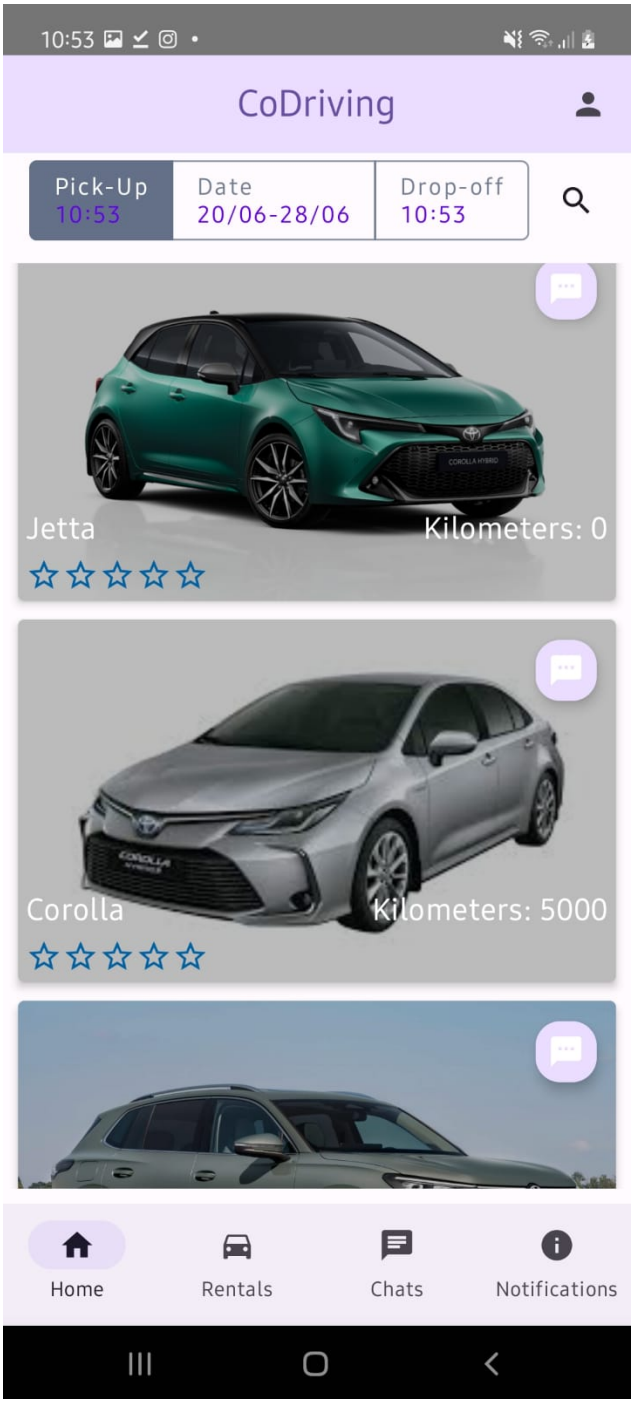

Ilustración 6.4: El auto publicado sale en el homepage

✔ **Visualización de automóviles con mejores valoraciones:** Los usuarios podrán ver los automóviles con las mejores valoraciones.

- ✔ **Búsquedas por filtros:** Los usuarios podrán buscar automóviles utilizando filtros como zona de recogida, zona de entrega, hora de recogida, hora de entrega y días de alquiler.
- ✔ **Acceso rápido a las ventanas principales:** Los usuarios tendrán acceso rápido a las distintas ventanas principales de la aplicación.
- ✔ **Accesos a la cabecera:** Los usuarios tendrán acceso rápido a elementos de la cabecera como son: perfil, cerrar sesión y cambiar el estado de la app (modo claro, modo oscuro)

6.2.3. Funcionalidades de la Ventana de Guardado de Automóviles

Funcionalidades de la ventana de guardado de automóviles, como se muestra en la siguiente ilustración: 6.5

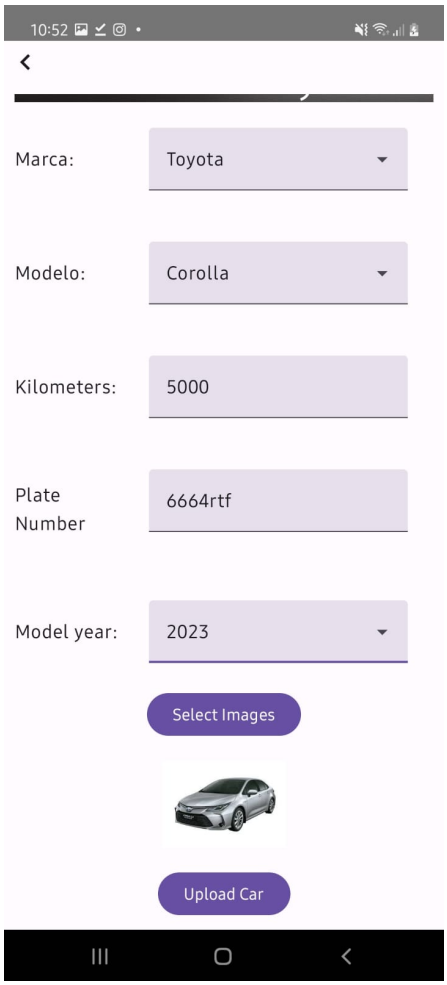

Ilustración 6.5: Pantalla de guardar un auto

Ilustración 6.6: Pantalla de visualización de autos guardados

- ✔ **Añadir automóviles:** El usuario podrá agregar automóviles de forma indefinida re-

llenando los siguientes campos: marca, modelo, kilómetros, número de placa, año e imágenes.

- ✔ **CRUD de automóviles:** El usuario podrá agregar, eliminar y editar todos los automóviles guardados.
- ✔ **Publicar automóviles:** El usuario, una vez guardado el automóvil, podrá publicarlo indicando el precio por día, lugar de recogida y entrega.

6.2.4. Funcionalidades de pagina de conversaciones y chat

Funcionalidades de la pagina de conversaciones y chat 6.7 y 6.8

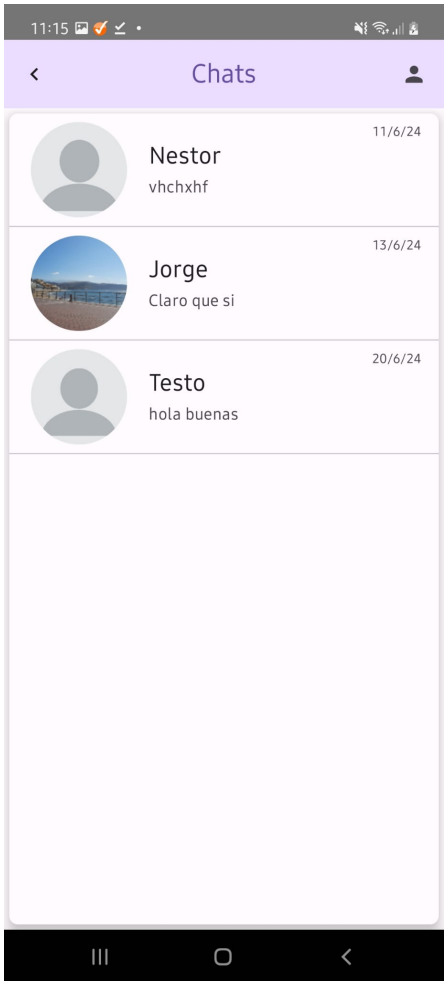

Ilustración 6.7: Pagina de conversaciones

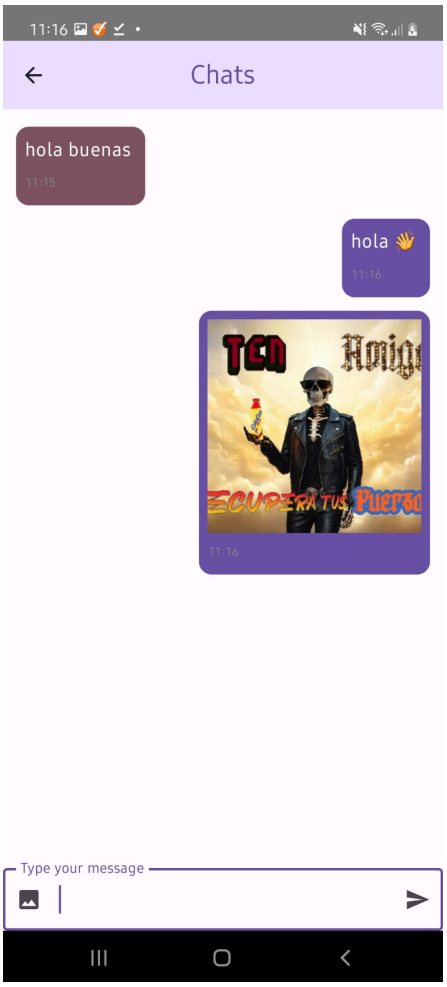

Ilustración 6.8: Chat en tempo real y visualización de imagen adjunto

- ✔ **Lista de conversaciones:** El usuario podrá ver una lista de todas sus conversaciones
- ✔ **Inicio de nuevas conversaciones:** El usuario podrá iniciar una nueva conversación buscando su publicaciones
- ✔ **Mensajería en tiempo real:** El usuario podrá enviar y recibir mensajes instantáneamente.
- ✔ **Adjuntar imágenes:** El usuario podrá adjuntar y enviar imágenes.
- ✔ **Historial de mensajes:** El usuario podrá ver el historial completo de mensajes dentro de cada conversación.
- ✔ **Eliminar mensajes:** El usuario podrá conversaciones completas.

Como podemos observar en la primera ilustración hemos entrado con otro usuario, el cual tenía varias conversaciones abiertas entre ellas, la del usuario para la fase de prueba ”testo”donde se previsualiza el último mensaje, una vez dentro vemos que se realiza una conversación en tiempo real y donde se pueden visualizar imágenes adjuntas

6.2.5. Funcionalidades del página de notificaciones

Funcionalidades de la pagina de notificaciones, como se muestran en las siguientes ilustraciones 6.9, 6.10 y 6.11

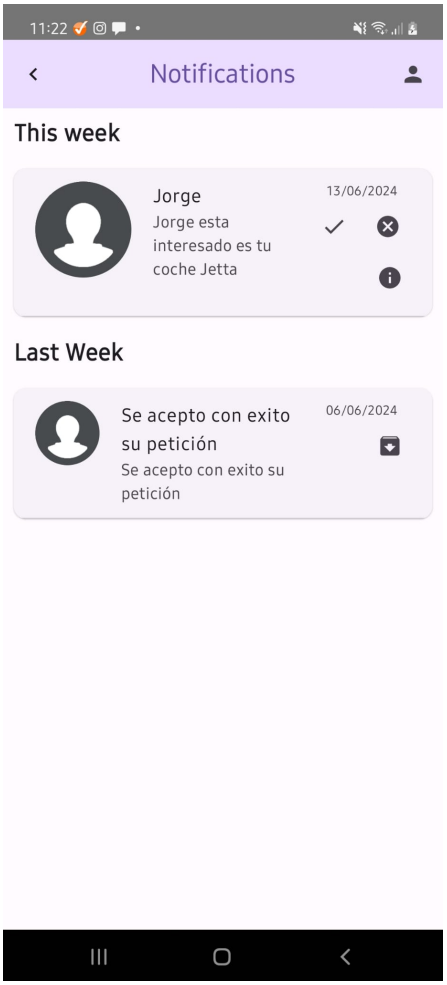

Ilustración 6.9: Primera notificación es la de la petición

Ilustración 6.10: Visualización de días de alquiler

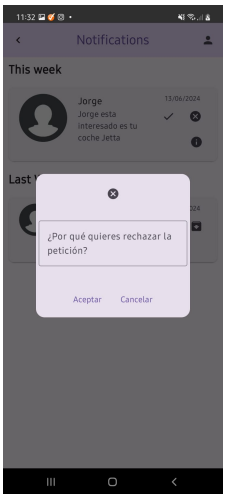

Ilustración 6.11: Visualización de rechazar una petición

- ✔ **Visualizar notificaciones:** El usuario podrá ver una lista de todas sus notificaciones.
- ✔ **Aceptar petición:** El usuario podrá aceptar peticiones de alquiler.
- ✔ **Rechazar peticiones:** El usuario podrá rechazar peticiones de alquiler dejando un mensaje.
- ✔ **Enviar notificaciones de aceptar y rechazar peticiones:** El usuario, al rechazar o aceptar peticiones, enviará la respuesta como notificación al otro usuario.
- ✔ **Ver días de alquiler:** El usuario podrá ver en las notificaciones de peticiones qué días están alquilando.
- ✔ **Eliminar notificaciones:** El usuario podrá eliminar las notificaciones informativas (las que no son peticiones).

6.3. Despliegue

Dado que el backend está realizado en Firebase, solo nos tendremos que preocupar de arrancar la aplicación. Para ello, necesitaremos Android Studio y clonar el repositorio. Si queremos ejecutarla en un emulador, no necesitamos nada adicional, pero si queremos probarla en dispositivos móviles físicos, tendremos que habilitar las opciones de desarrollador. Para ello, seguiremos los siguientes pasos, teniendo en cuenta que dependiendo del modelo y marca del dispositivo, puede variar ligeramente [18]:

1. En el dispositivo Android, abre **Configuración** y busca **Acerca del dispositivo**.
2. Elige **Acerca del dispositivo** y, luego, presiona **Número de compilación** siete veces. Ingresa la contraseña o el PIN del dispositivo (si se te solicita).
3. Regresa a **Configuración** y presiona **Sistema**. **Opciones para desarrolladores** debería aparecer en la lista. Es posible que esta opción se encuentre en **Avanzada**.
4. Presiona **Opciones para desarrolladores** y, luego, activa **Depuración por USB**.

Una vez realizados estos pasos, solo tendremos que ir a Android Studio y elegir el dispositivo en el mismo lugar donde aparece el emulador.

Capítulo 7

Conclusiones y trabajo futuro

7.1. Conclusiones

El desarrollo de la aplicación CoDrive ha representado una experiencia profundamente enriquecedora y formativa. Este proyecto, orientado al alquiler de coches, ha sido creado utilizando Kotlin en Android Studio, empleando Jetpack Compose para la construcción de la interfaz de usuario. Destaco que es la primera vez que utilizo Jetpack Compose, lo que ha implicado un desafío y, al mismo tiempo, una oportunidad de aprendizaje significativo.

La integración de tecnologías avanzadas de Google, como Firebase para la gestión completa del backend y Material3 para el diseño de la interfaz de usuario, ha sido fundamental en el desarrollo de esta aplicación. Estas herramientas han proporcionado una base sólida y versátil, permitiendo una implementación eficaz y moderna de las funcionalidades requeridas.

Jetpack Compose se revela como una herramienta poderosa y eficiente, transformando la creación de interfaces de usuario en un proceso más intuitivo y productivo en comparación con el enfoque tradicional basado en XML. Aunque esta es mi primera incursión en esta tecnología, he podido apreciar su potencial y las ventajas significativas que ofrece en términos de simplicidad y eficiencia durante el desarrollo.

El proceso de desarrollo en la plataforma Android ha sido altamente gratificante. Android ofrece una gran flexibilidad y una amplia gama de posibilidades para la innovación en el desarrollo de aplicaciones móviles. Estoy convencido de que tanto Android como Jetpack Compose tienen un futuro prometedor, proporcionando un entorno robusto y versátil para los desarrolladores.

Finalmente, considero que CoDrive tiene un potencial considerablemente mayor que el actualmente explotado. Las funcionalidades presentes en esta versión inicial son solo el comienzo. Existe un amplio margen para continuar mejorando y añadiendo características que enriquezcan la experiencia del usuario y optimicen el servicio de alquiler de coches. Estoy muy interesado en las posibilidades futuras y las innovaciones que se pueden implementar en las próximas versiones de la aplicación.

7.2. Trabajo futuro

Con el objetivo de continuar con el desarrollo y perfeccionamiento de la aplicación CoDrive, se han identificado diversas áreas de mejora y expansión que, al implementarse, potenciarán significativamente la funcionalidad y la experiencia del usuario. Las siguientes propuestas delinean los objetivos estratégicos futuros para la aplicación:

7.2.1. Integración con Google Maps API

Se propone integrar Google Maps API en la aplicación CoDrive. Esta funcionalidad permitirá mostrar un mapa interactivo en el cual los usuarios podrán buscar y localizar coches de alquiler en su área, de manera similar a la experiencia de búsqueda en plataformas como Airbnb. Esta característica no solo mejorará la experiencia del usuario, sino que también facilitará la localización y selección de vehículos disponibles en tiempo real, optimizando el proceso de alquiler.

7.2.2. Seguimiento en Tiempo Real del Vehículo

Implementar un sistema de seguimiento en tiempo real permitirá a los usuarios conocer la ubicación exacta de los vehículos alquilados en cualquier momento. Esta funcionalidad incrementará la transparencia y seguridad del servicio, ofreciendo tranquilidad tanto a los usuarios como a los proveedores del servicio. Además, permitirá gestionar de manera eficiente la logística y operación del alquiler de vehículos.

7.2.3. Conectividad Constante con el Vehículo para Análisis de Datos

Se plantea la posibilidad de que la aplicación CoDrive se conecte continuamente con los vehículos alquilados para recolectar y analizar datos en tiempo real. Esta conectividad permitirá monitorear el estado del vehículo, detectar posibles problemas de manera anticipada y ofrecer servicios adicionales basados en el análisis de datos, como recomendaciones de mantenimiento y mejoras en el rendimiento del vehículo.

7.2.4. Mejora del Buscador con Filtros Avanzados

Es crucial mejorar el sistema de búsqueda dentro de la aplicación, incorporando filtros más efectivos y avanzados. Los usuarios podrán buscar vehículos basándose en una variedad de criterios adicionales como tipo de vehículo, precio, características específicas del coche, disponibilidad, entre otros. Esto hará que el proceso de búsqueda sea más preciso y alineado con las necesidades específicas de cada usuario.

Bibliografía

[1] Android Developers (julio de 2023). *Notas de la versión de Android Studio: Giraffe*. Android Studios. Recuperado 08/07/2024 de https://developer.android.com/studio/releases/past-releases/as-giraffe-release-notes?hl=es-419.

[2] Atlassian, Inc. (2024). *Trello: Organize anything, together*. Recuperado el 4 de julio de 2024 de https://trello.com.

[3] Figma, Inc. (2024). collaborative interface design tool. Figma.Recuperado el 4 de julio de 2024 de https://www.figma.com.

[4] Free2Move (2024). *Free2Move*. Recuperado el 8 de julio de 2024 Recuperado en https://www.free2move.com.

[5] Getaround (2024). *Getaround*. Consultado el 8 de julio de 2024 Recuperado en https://www.getaround.com.

[6] GitHub (2023). *GitHub: A guide to using repositories for software development*. Recuperado 8 de julio de 2024 en https://docs.github.com/en/get-started.

[7] Google (2023). *Material Design 3*. Consultado el 8 de julio de 2024 Recuperado en https://m3.material.io/.

[8] Google Firebase (2023). *Firebase CLI*. Google Firebase. Recuperado el 8 de julio de 2024 de https://firebase.google.com/docs/cli?hl=es-419.

[9] Google Firebase (2024). *Firebase Console*. Google Firebase. Recuperado el 8 de julio de 2024 de https://console.firebase.google.com/u/0/.

[10] Google, Inc. (2024a). Android studio: The official ide for android. Recuperado : 2024-05-12 https://developer.android.com/studio.

[11] Google, Inc. (2024b). *Firebase Authenticator: Firebase Authentication for apps*. Recuperado el 25 de junio de 2024 de https://firebase.google.com/products/auth.

[12] Google, Inc. (2024c). *Firebase Database: Realtime database for apps*. Recuperado el 15 de abril de 2024 de https://firebase.google.com/products/realtime-database.

[13] Google, Inc. (2024d). *Firebase Storage: Cloud storage for apps*. Firabase. Recuperado el 4 de julio de 2024 de https://firebase.google.com/products/storage.

[14] JetBrains (2023). *IntelliJ IDEA*. Consultado el 8 de julio de 2024 Recuperado en https://www.jetbrains.com/idea/.

[15] Node.js Foundation (Sin fecha). *Node.js*. Node.js Foundation. Recuperado el 8 de julio de 2024 de https://nodejs.org/en.

[16] NOW, S. (2024). *SHARE NOW*. Recuperado el 8 de julio de 2024 Recuperado en https://www.share-now.com/es/es/.

[17] Statista (2024). *Alemania: Estadísticas y hechos*. Consultado el 8 de julio de 2024 Recuperado en https://es.statista.com/sectores/1175/tema/1849/alemania/.

[18] Studios, A. (2023). *Cómo activar la depuración por USB*. Recuperado en https://developer.android.com/codelabs/basic-android-kotlin-training-run-on-mobile-device?hl=es-419#1.

[19] Vico", A. J. (2011). Librerías de android. Column80 wordpress. Recuperado el 8 de julio de 2024 de https://columna80.wordpress.com/2011/02/17/arquitectura-de-android/.

[20] Zipcar (2024). *Zipcar*. Consultado el 8 de julio de 2024 Recuperado en https://www.zipcar.com.