I.E.S RAFAEL ALBERTI
Curso 2024/2025

TRACKMYRIDE

Memoria del proyecto final de 2º Desarrollo Aplicaciones Multiplataforma. App Android

Adrián Arroyo Gómez
12/06/2025

ÍNDICE

1. Introducción, expectativas/objetivos y antecedentes

1.1 Idea inicial y motivación personal

La idea original de esta aplicación, **TrackMyRide**, surge de una necesidad real y cotidiana vivida en primera persona: poder salir a la carretera sin un destino concreto, disfrutar del trayecto, dejarse llevar por el momento y, al finalizar, **conservar un registro detallado del recorrido realizado**. Esta necesidad nace del estilo de conducción de muchos aficionados al mundo del motor o aventura mismamente, especialmente en el ámbito de las motocicletas, donde salir a rodar sin una ruta planificada es algo habitual. Sin embargo, tras una jornada de conducción espontánea, muchas veces resulta imposible recordar exactamente por dónde se ha pasado, qué carreteras se han disfrutado o qué paisajes se han encontrado por el camino. **TrackMyRide** nace como una solución a este problema: **una herramienta que registra automáticamente el trayecto y permite al usuario revivirlo cuando quiera, consultarlo, compartirlo o incluso repetirlo en el futuro.**

A partir de esta premisa tan sencilla, la idea comenzó a evolucionar con rapidez. Además de servir como un diario de rutas, la aplicación también resultó tener utilidad en contextos completamente distintos. Por ejemplo, en el proceso de **obtención del permiso de conducir**, esta aplicación se ha utilizado durante clases prácticas en autoescuela, activándola al inicio de cada sesión para registrar los trayectos realizados con el instructor. Posteriormente, esos recorridos podían repetirse como copiloto junto a un amigo o familiar, permitiendo **repasar las rutas más frecuentes del examen y familiarizarse con las calles clave de la ciudad**. Este caso de uso inesperado reforzó la idea de que la aplicación podría tener múltiples aplicaciones, más allá de un uso exclusivamente recreativo.

1.2 Objetivos del proyecto

A partir de la idea inicial, se definieron los siguientes objetivos como base del proyecto:

- **Permitir al usuario grabar sus rutas mediante el GPS del dispositivo**, de forma automática y sin distracciones durante la conducción.
- **Almacenar el historial de rutas**, permitiendo visualizar los trayectos realizados sobre un mapa con detalles como título, descripción, fecha o duración.
- **Ofrecer herramientas de personalización** sobre cada ruta, incluyendo la posibilidad de añadir imágenes, colocar marcadores o “pines” en puntos clave y editar la información descriptiva.
- **Permitir la clasificación del vehículo utilizado** (moto, coche o bicicleta) y añadir datos técnicos como consumo medio o eficiencia para generar estadísticas útiles.
- **Ofrecer una experiencia ampliada a los usuarios Premium**, con ventajas como la posibilidad de almacenar más rutas, más imágenes, compartirlas con otros usuarios o exportarlas para su uso externo.
- **Facilitar el acceso y registro de nuevos usuarios** mediante un sistema de cuentas personalizado, así como garantizar la seguridad y privacidad de sus datos personales y rutas almacenadas.

En esencia, **TrackMyRide** busca ser **una extensión inteligente del cuaderno de viaje clásico**, adaptado a la tecnología actual y pensado para todo tipo de conductores o aficionados al desplazamiento libre.

1.3 Evolución de la idea y ampliación del público objetivo

En sus inicios, TrackMyRide estaba concebido como una aplicación específicamente pensada para motociclistas. Como aficionado a las rutas en moto, centre mi diseño y funcionalidad inicial en este colectivo, con un enfoque claro: cubrir las necesidades de quienes salen sin un rumbo fijo, pero desean recordar y compartir sus recorridos más satisfactorios.

Sin embargo, con el tiempo y a medida que se iban esbozando las funcionalidades, quedó claro que **las necesidades que cubría TrackMyRide no eran exclusivas del mundo motero**. Ciclistas urbanos y de carretera, conductores de coche aficionados a los viajes, podrían aprovechar sus funciones. Por ello, la app fue adaptada para permitir seleccionar el tipo de vehículo utilizado (coche, moto o bici) y para **generar estadísticas para cada modalidad**, ampliando su público objetivo de forma significativa.

De este modo, **TrackMyRide** pasó de ser una ide de aplicación “para moteros” a una solución versátil, pensada para cualquier persona interesada en **registrar, consultar, repetir o compartir sus rutas personales**, sin importar el medio de transporte utilizado.

1.4 Tecnologías consideradas y elección final

Desde el inicio del proyecto, uno de los pilares fundamentales fue la elección de un stack tecnológico moderno, robusto y que permitiera tanto un desarrollo ágil como una experiencia de usuario satisfactoria. La aplicación fue concebida desde el principio como una **aplicación móvil nativa para Android**, y en consecuencia, se optó por utilizar **Jetpack Compose** como tecnología base para el desarrollo de la interfaz de usuario.

Jetpack Compose es una herramienta declarativa de UI desarrollada por Google, que representa una evolución respecto a los tradicionales XML en Android. Esta tecnología permite una mayor fluidez en el diseño de pantallas, facilita el mantenimiento del código y proporciona una estructura más reactiva, donde los componentes se actualizan automáticamente ante los cambios de estado. Gracias a esto, fue posible implementar interfaces limpias, modernas y responsivas, lo cual es fundamental para una aplicación que se centra en la experiencia visual del usuario (visualización de rutas, imágenes, mapas, etc.).

Persistencia y backend: de SQLite local a una solución client-server

En las primeras fases del desarrollo, se consideró usar una base de datos local como **SQLite** para el almacenamiento de rutas, vehículos y usuarios. Esta opción tenía la ventaja de su simplicidad e independencia de conexión a internet, y habría permitido un desarrollo rápido para una versión inicial o prototipo. Sin embargo, tras una evaluación más profunda, se decidió descartar esta opción en favor de una arquitectura más profesional basada en cliente-servidor.

El principal motivo de este cambio fue la **seguridad y escalabilidad**. Usar almacenamiento local no permitía implementar un sistema real de autenticación de usuarios, control de suscripciones Premium, ni sincronización de datos entre dispositivos. Tampoco era viable gestionar el acceso a funcionalidades según el tipo de usuario (Premium o gratuito). Por ello, se optó por desarrollar un backend propio utilizando **Spring Boot**, una de las tecnologías más utilizadas actualmente para construir APIs REST seguras y escalables.

Spring Boot, junto con **Spring Security y JWT (JSON Web Tokens)**, permite controlar de forma precisa el acceso a los recursos según el rol del usuario. Se implementó un sistema de autenticación mediante tokens que garantiza que solo los usuarios registrados puedan acceder a su información, y que las funcionalidades Premium estén protegidas correctamente.

Firebase Authentication

Para simplificar y robustecer el proceso de registro y login, se integró **Firebase Authentication**, una herramienta proporcionada por Google que permite gestionar el acceso de los usuarios de forma segura, sencilla y personalizable. Firebase maneja toda la lógica de autenticación en segundo plano, incluyendo recuperación de contraseñas, validación de correos electrónicos y almacenamiento cifrado de credenciales.

Esta integración permitió acelerar el desarrollo del sistema de usuarios y evitar implementar desde cero una lógica compleja de gestión de contraseñas, correos duplicados, validación, etc. Además, Firebase ofrece compatibilidad total con Android y se integra fácilmente con otros servicios si en el futuro se quisiera añadir login con Google o redes sociales.

Servicios en la nube: Railway y Render

Con el backend desarrollado en Spring Boot y una base de datos MySQL como núcleo del almacenamiento de datos, fue necesario desplegar ambos componentes en servicios accesibles por internet. Para ello, se utilizaron las siguientes plataformas:

- **Railway**: se usó como servicio de hosting para la **base de datos MySQL**. Railway permite desplegar y escalar bases de datos en la nube con facilidad, ofreciendo una interfaz amigable, backups automáticos y entornos de desarrollo rápidos. Esta solución facilita el acceso desde la aplicación móvil sin preocuparse por la infraestructura, lo cual es ideal en entornos de desarrollo y producción ligeros.
- **Render**: se utilizó como plataforma para el **hosting del backend Spring Boot**. Render permite desplegar aplicaciones de backend con un sistema de CI/CD integrado, lo cual significa que cada vez que se realiza un cambio en el código fuente, el servidor se actualiza automáticamente. Esto garantiza una continuidad en el desarrollo y facilita la integración con GitHub como repositorio de control de versiones. Render también permite configurar entornos seguros, variables de entorno y acceso HTTPS sin costes adicionales en sus planes básicos.

Ambas plataformas, Railway y Render, ofrecen soluciones freemium que permiten mantener la aplicación operativa sin costes elevados en fases de desarrollo, lo que fue especialmente valioso para este proyecto académico. Además, su facilidad de configuración y compatibilidad con tecnologías modernas las convirtió en la mejor opción frente a otras alternativas más complejas como AWS o Azure, que requerían una infraestructura más pesada.

Otras herramientas y librerías

Durante el desarrollo de la aplicación se han empleado otras tecnologías auxiliares que también merecen mención:

- **Google Maps SDK**: para la visualización de las rutas en el mapa, el posicionamiento GPS en tiempo real y la colocación de pines personalizados.
- **Retrofit**: para la comunicación entre la app móvil y el backend Spring Boot mediante peticiones HTTP.
- **Coil**: librería moderna para la carga eficiente de imágenes en Compose.
- **Cloudinary**: para gestionar las imágenes subiéndolas a la nube.
- **Paypal Sandbox**: para gestionar el pago simulado de la suscrpción premium.

1.5 Aplicaciones similares y diferenciación

En el mercado existen algunas aplicaciones que comparten ciertas similitudes con TrackMyRide, como **GeoTracker**, **Komoot**, **Strava** o **Relive**. Estas apps permiten registrar recorridos y consultar estadísticas, aunque cada una presenta limitaciones específicas. GeoTracker, por ejemplo, si bien cumple con el objetivo de registrar rutas GPS, **ofrece una experiencia de usuario poco intuitiva, con una interfaz más anticuada y una presencia intrusiva de anuncios**, lo cual afecta negativamente a la experiencia general. Otras, como Strava, están muy orientadas al deporte competitivo y no cubren bien el perfil de usuario que simplemente desea registrar y revivir rutas de ocio o transporte diario.

La principal ventaja competitiva de TrackMyRide reside en su **sencillez, versatilidad y enfoque personalizado**. A diferencia de las aplicaciones existentes, esta herramienta no está pensada exclusivamente para deportistas, ni requiere conocimientos técnicos avanzados. Se adapta tanto a un usuario que desea **revivir su trayecto en moto un domingo por la sierra**, como a otro que quiere repetir la ruta hecha en coche durante unas prácticas de autoescuela, o simplemente guardar el camino a casa tras perderse voluntariamente explorando una nueva ciudad. Además, funciones como **el registro visual con imágenes, la adición de pines personalizados o la simulación de pagos Premium** aportan un valor añadido que pocas aplicaciones ofrecen en conjunto.

2. Descripción del resultado obtenido

El resultado final del desarrollo ha sido una aplicación móvil funcional, intuitiva y visualmente atractiva, orientada principalmente a usuarios que realizan rutas en distintos tipos de vehículos (moto, coche o bicicleta) y desean registrar, conservar, visualizar y compartir sus recorridos con una serie de funcionalidades avanzadas. Esta sección detalla todas las funcionalidades desarrolladas, así como la organización interna de la aplicación y su estructura general.

2.1 Estructura por capas del proyecto

data/: Contiene todo lo relacionado con el acceso a datos, tanto locales como remotos.

- **local/**: Aquí se almacenan las preferencias locales del usuario como autenticación persistente, tema de la aplicación, y opción de recordar sesión.
- **remote/**:
    - **API/**: Interfaces de Retrofit que gestionan las peticiones HTTP a los distintos endpoints del backend (usuarios, rutas, vehículos, imágenes, autenticación, etc).
    - **DTO/**: Objetos de transferencia de datos que representan las estructuras esperadas por la API.
    - **firebase/**: Encapsula toda la lógica de autenticación mediante Firebase.
    - **mappers/**: Mapeadores para convertir entre DTOs y modelos de dominio.
- **repository/**: Implementaciones concretas de los repositorios que actúan como puente entre los datos (API/local) y la lógica de negocio.

di/ (Dependency Injection): Módulos de Hilt que proveen las dependencias necesarias como Retrofit, ubicación, repositorios, sesión de usuario, etc.

domain/: Define el núcleo de la lógica del negocio.

- **model/**: Contiene las entidades principales del sistema (Ruta, Usuario, Vehículo, Imagen, etc.).
- **usecase/**: Cada caso de uso de la app está encapsulado aquí: iniciar sesión, registrar usuario, obtener rutas, etc.
- **repository/**: Interfaces de los repositorios utilizados en los casos de uso.
- **tracker/**: Contiene la lógica principal de seguimiento de rutas mediante GPS.

navigation/: Gestiona toda la navegación entre pantallas utilizando Jetpack Navigation, con soporte para AppBar y menú lateral.

ui/: Todo lo relacionado con la interfaz gráfica.

- **screens/**: Composables separados por pantallas: inicio de sesión, registro, home/mapa, detalles de ruta, perfil, premium, etc.
- **components/**: Componentes reutilizables como botones, diálogos, headers...
- **theme/**: Paleta de colores, tipografías y estilos generales.
- **permissions/**: Manejo de permisos de ubicación y almacenamiento.

utils/: Funciones y herramientas auxiliares, adaptadores de fecha, simplificador de rutas, llamadas seguras a la API, etc.

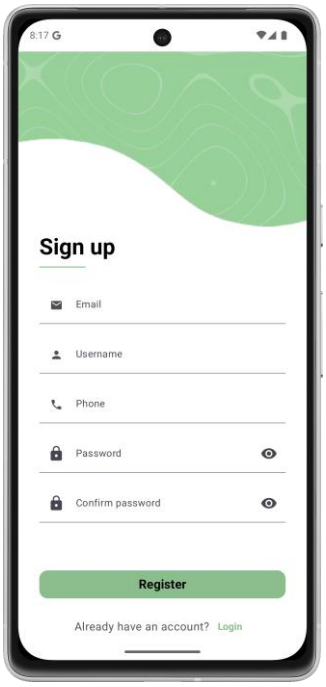
MainActivity.kt: Punto de entrada principal de la aplicación.

AppGlobals: Valores globales y constantes instanciados para usarlos en toda la aplicación

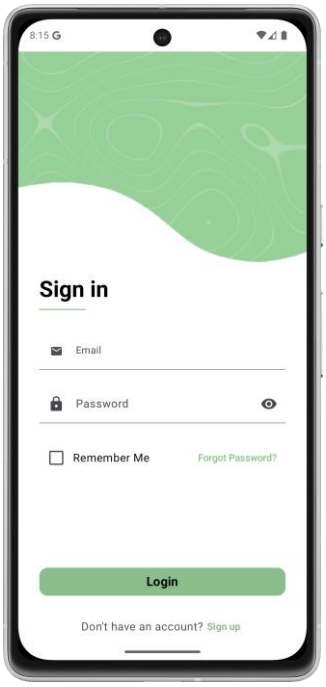
2.2 Pantallas implementadas

La aplicación cuenta con un conjunto de pantallas cuidadosamente diseñadas para ofrecer una experiencia intuitiva, funcional y adaptada al uso real por parte de los usuarios. A continuación, se describen cada una de ellas:

- **Pantalla de inicio de sesión (Login)**
  Permite al usuario acceder a la aplicación introduciendo su correo y contraseña. Se conecta con **Firebase Authentication** para validar los datos y establecer una sesión segura. También ofrece acceso a la pantalla de registro y recuperación de contraseña.
- **Pantalla de registro (Register)**
  Ofrece un formulario para nuevos usuarios donde pueden crear una cuenta introduciendo nombre, correo, contraseña y confirmación. Tras registrarse, se redirige automáticamente a la pantalla principal.



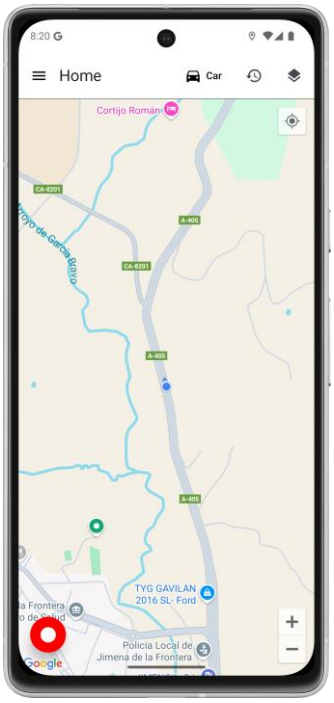
Img 1.Registro



Img 2.Login

- **Pantalla de recuperación de contraseña**
  Permite al usuario introducir su email y recibir un enlace de recuperación a través de **Firebase**. Esto mejora la accesibilidad y control de cuentas.

- **Pantalla principal (Home - Mapa)**
  Es la pantalla central de la app, donde el usuario puede comenzar el seguimiento de su ruta mediante el botón principal de grabación. Utiliza la **API de Google Maps** para mostrar el mapa en tiempo real con la localización actual y el recorrido en vivo. Desde aquí se accede al menú lateral y otras funciones rápidas.



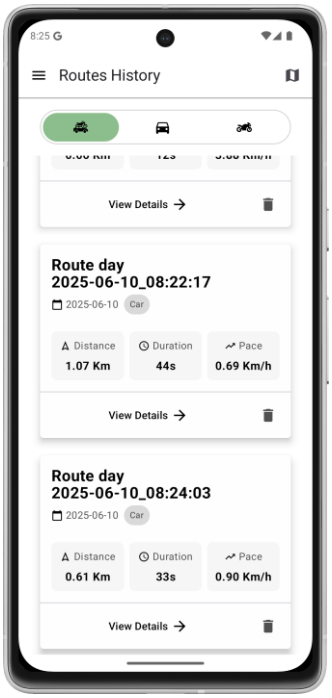
Img 4.Home

Img 3.Home grabando

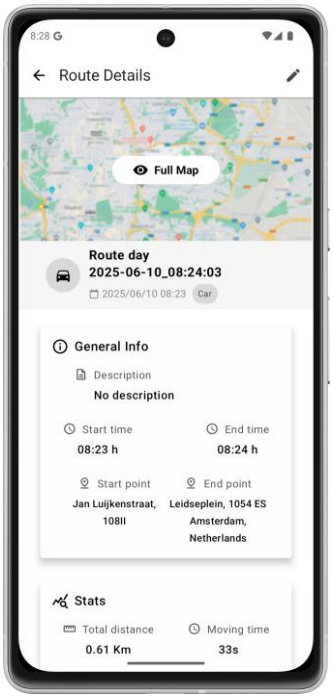
- **Pantalla de perfil**
  Muestra los datos del usuario, su correo, nombre y foto de perfil. Permite cambiar la imagen del usuario, nombre de usuario, teléfono y contraseña. Es una sección sencilla, pero muy útil para personalización básica.

- **Historial de rutas**
  Muestra una lista de las rutas grabadas por el usuario. Si es usuario gratuito, solo se muestran las 5 más recientes; los usuarios premium tienen acceso completo. Cada ruta muestra su título, duración, distancia, ritmo, y permite eliminarla y acceder a los detalles completos.

- **Detalle de ruta**
  Aquí se pueden consultar todos los datos de una ruta en particular: trazado en el mapa, estadísticas (consumo, eficiencia, velocidad media, etc.), imágenes asociadas, pins personalizados y notas. También se puede editar el título, descripción, o eliminar la ruta.

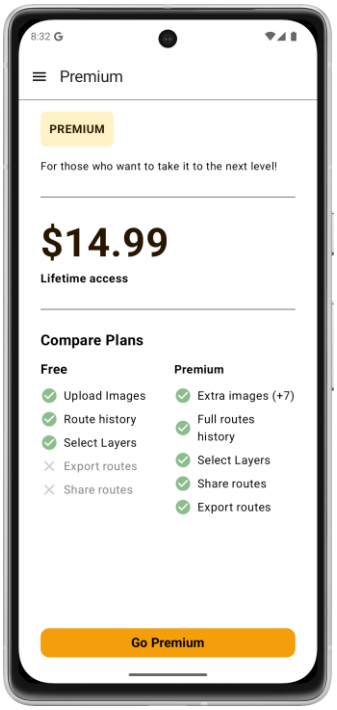


Img 5.Historial rutas



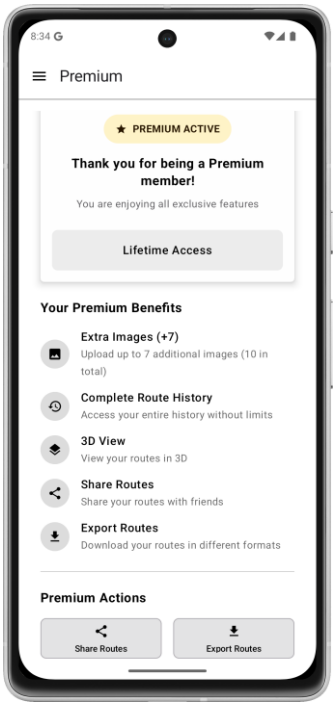
Img 6.Detalles ruta

- **Pantalla de Premium**
  Explica las ventajas de la cuenta premium: historial ilimitado, exportar rutas, más imágenes por ruta, etc. Desde aquí se lanza la pantalla de pago.
- **Pantalla de pago Premium**
  Simula un proceso de pago mediante **PayPal Sandbox**, ofreciendo una experiencia cercana a una compra real sin necesidad de transacción real. Si el pago se simula correctamente, el usuario pasa a tener estado premium.



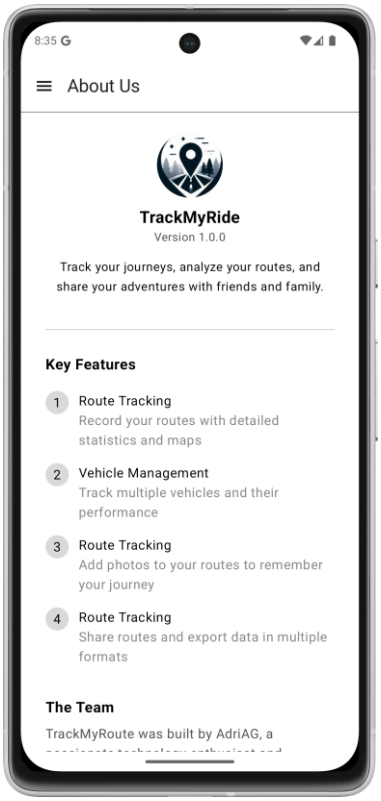
Img 8.No Premium

Img 7.Paypal

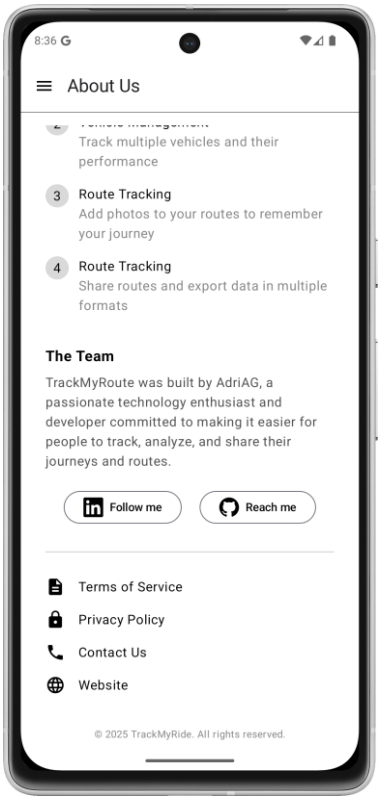


Img 9.Premium

- **Pantalla "Sobre Nosotros" (About Us)**
  Pantalla informativa donde se muestra una breve descripción del proyecto, el autor y los objetivos de la aplicación.



Img 10.About Us



Img 11.About Us 2

3. Proceso de instalación, despliegue y configuración

El correcto funcionamiento de la aplicación requiere tanto el despliegue del backend (API REST) como de la aplicación móvil (cliente Android). Ambos deben ser configurados de forma adecuada, tanto en entorno **local** como en producción (nube). A continuación, se describen todos los pasos necesarios para cada uno de los entornos.

3.1 Despliegue y configuración en local

3.1.1 Requisitos previos

Antes de comenzar, es necesario contar con:

- Tener instalado **XAMPP** o un servidor local con MySQL y Apache.
- Git para clonar repositorios
- Tener instalado **Android Studio** para compilar la app móvil.
- Cuenta en **Cloudinary**.
- Cuenta en **Firebase**.
- Cuenta de **PayPal Developer** (para simulación de pago).
- Obtener token **Google Maps API**

3.1.2 Configuración y despliegue del backend (API)

Paso 1: Clonar el repositorio

Clonar el repositorio que contiene el proyecto del backend:

git clone https://github.com/agadrian/TrackMyRideAPI

Paso 2: Levantar el servidor de base de datos

- Abrir **XAMPP**.
- Activar **Apache** y **MySQL**.
- Entrar a phpMyAdmin y **crear la base de datos**: *CREATE DATABASE trackmyridedb;*
- Importar la estructura desde tu archivo .sql, que se encuentra en el repositorio:

  Abrir .SQL

Paso 3: Configurar el .env

Para ello tendrás que obtener las credenciales de **Cloudinary.** También tendrás que obtener el serviceAccount.json que proporciona **Firebase** y guardarlo en la siguiente ruta de la API: **src/main/resources/firebase/serviceAccount.json.**

```
DB_NAME =trackmyridedb
DB_USER =
DB_PASSWORD =
cloudinary.cloud_name=
cloudinary.api_key=
cloudinary.api_secret=
```

Img 12.Archivo .env API

Luego en el **application.properties** referenciamos a estos valores:

```
# Datos de conexion con la base de datos MySQL
spring.jpa.database-platform=org.hibernate.dialect.MySQLDialect
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/${DB_NAME}
spring.datasource.username=${DB_USER}
spring.datasource.password=${DB_PASSWORD}
```

Img 13.Archivo local.properties API

Una vez realizado esto, simplemente ejecutamos en local la API para que corra en localhost.

3.1.3 Configuración y despliegue de la App Android

Paso 1: Clonar el repositorio de la App:
git clone https://github.com/agadrian/TrackMyRide

Paso 2: Modificar NetworkModule.kt
Debemos de poner la IP local para que la app detecte la conexión:

```
@Module
@InstallIn(SingletonComponent::class)
object NetworkModule {
    private const val BASE_URL = "http://10.0.2.2:8080"
```

Img 14.Object NetwokModule App

Paso 3: Añadir el serviceAccount.json de Firebase
Tendremos que añadir en el módulo de la App este .json también para que a la hora de generar la Apk, se incluya y permita el uso de **Firebase** para la autenticación.

Paso 4: Configurar el local.properties
Tendremos que configurar este archivo (si no lo tenemos hay que crearlo en la raíz del proyecto) para establecer diferentes parámetros, los cuales obtenemos en las diferentes plataformas previamente mencionadas:

```
MAPS_API_KEY=
cloudinary_cloud_name=
cloudinary_api_key=
cloudinary_api_secret=
cloudinary_unsigned_preset=
paypal_client_id=
```

Img 15.Archivo local.properties App

Una vez realizadas estas configuraciones, podremos compilar la App usando Android Studio, ya sea en el emulador o generando una Apk en nuestro dispositivo móvil, y ya podremos usarla de forma normal, siempre y cuando estemos corriendo el backend (API).

3.2 Despliegue y configuración en la nube

3.2.1 Requisitos previos

- Cuenta en un proveedor de despliegue de API, en este caso **Render**
- Base de datos en la nube, en este caso **Railway**
- Proyecto subido a **GitHub**
- Cuenta en **Dropbox** o similar para subir el **.war** del proyecto API
- Las mismas cuentas de **Firebase**, **Cloudinary** y **PayPal Developer** (crearlas si fuera necesario)
- Obtener token **Google Maps API**

3.2.2 Configuración y despliegue del backend (API)

Paso 1: Clonar el repositorio

Clonar el repositorio que contiene el proyecto del backend:

git clone https://github.com/agadrian/TrackMyRideAPI

Paso 2: Subir la base de datos a Railway

Primero deberemos de crear un contenedor **MySql** en un proveedor web, **Railway** en este caso. Importaremos el archivo .sql que contiene el repositorio para crear la estructura inicial de la base de datos.

Una vez subido, deberemos de copiar los datos que nos ofrece **Railway** para conectarnos en el siguiente paso desde la API.

Paso 3: Modificar variables entorno

Modificamos el archivo **application.properties** para incluir los valores adecuados:

```
spring.datasource.url=jdbc:mysql://${MYSQL_HOST}:${MYSQL_PORT}/${MYSQL_DATABASE}?
useSSL=false&serverTimezone=UTC
spring.datasource.username=${MYSQL_USER}
spring.datasource.password=${MYSQL_PASSWORD}
```

Img 16.Archivo application.properties API

Y en el **.env** añadimos esos campos, configurándolo con los valores que nos da el proveedor de la base de datos que hayamos elegido, **Railway** en este caso:

```
cloudinary.cloud_name=
cloudinary.api_key=
cloudinary.api_secret=
MYSQL_HOST=
MYSQL_PORT=
MYSQL_DATABASE=
MYSQL_USER=
MYSQL_PASSWORD=
```

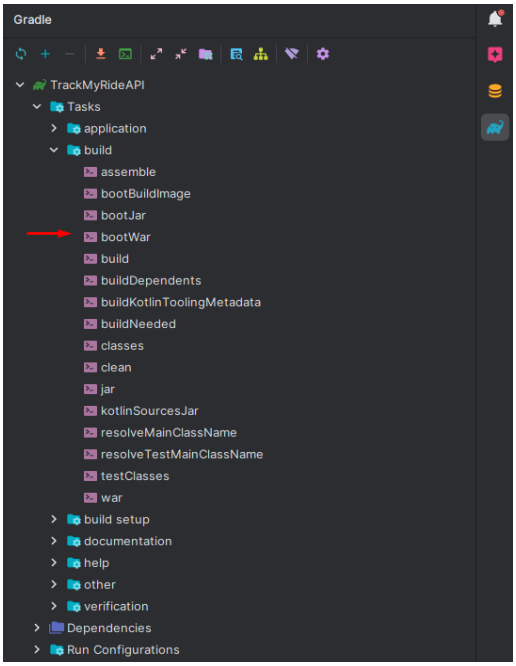
Img 17.Archivo .env API

Este paso recomiendo hacerlo, aunque no sea estrictamente necesario, ya que en Render ya le decimos cuales son las variables de entorno, pero no está de más ponerlas aquí también.

Paso 4: Crear y subir .war

Ahora debemos de crear un .war de nuestra App, y subirlo a **Dropbox** por ejemplo.

En caso de usar **InteliJ** como IDE, se haría desde el apartado de **Gradle:**



Img 18.Archivo .war api

Nos creará un archivo con el nombre de nuestra app .war, el cual se encontrará en build/Libs. Este archivo lo subimos a **Dropbox** y copiamos el enlace que usaremos mas adelante.

Paso 5: Editar dockerfile

Ahora tendremos que editar en la raíz del proyecto el archivo **dockerfile**, y poner nuestro enlace generado por **Dropbox**. Esto lo hacemos para que se descargue el .war automáticamente en la plataforma que despleguemos la API, **Render** en este caso.

```
FROM azul/zulu-openjdk:17-latest

# Instala curl
RUN apt-get update && apt-get install -y curl

# Crea el directorio de trabajo
WORKDIR /app

# Descarga directa desde Dropbox (con dl=1)
RUN curl -L -o app.war "enlacedropbox"

# Ejecuta el .war
CMD ["java", "-jar", "app.war"]
```

Img 19.Archivo dockerfile API

Paso 6: Enlazar el repositorio en Render

Por último, debemos de subir los cambios si no lo hemos hecho aún, a un repositorio.

En Render, creamos un nuevo despliegue, y vinculamos nuestro repositorio.

También tendremos que crear las mismas variables de entorno que hemos configurado en local, pero en render.

Una vez configuradas, simplemente iniciamos la API desde Render y se desplegará correctamente.

3.2.3 Configuración y despliegue de la App Android

Paso 1: Clonar el repositorio de la App si aún no lo hemos hecho:
git clone https://github.com/agadrian/TrackMyRide

Paso 2: Modificar NetworkModule.kt
Debemos de cambiar la IP a la que nos conectamos, estableciendo la que nos ofrece Render:

```
@Module
@InstallIn(SingletonComponent::class)
object NetworkModule {

    private const val BASE_URL = "https://trackmyrideapi.onrender.com"
```

Img 20.Object NetworkModule App

Paso 3: Añadir el serviceAccount.json de Firebase
Tendremos que añadir en el módulo de la App el .json que ofrece **Firebase** para el SDK de Android para que a la hora de generar la Apk, se incluya y permita el uso de **Firebase** para la autenticación.

Paso 4: Configurar el local.properties
Tendremos que configurar este archivo (si no lo tenemos hay que crearlo en la raíz del proyecto) para establecer diferentes parámetros, los cuales obtenemos en las diferentes plataformas previamente mencionadas:

```
MAPS_API_KEY=
cloudinary_cloud_name=
cloudinary_api_key=
cloudinary_api_secret=
cloudinary_unsigned_preset=
paypal_client_id=
```

Img 21.Archivo local.properties App

Una vez realizadas estas configuraciones, podremos compilar la App usando Android Studio, ya sea en el emulador o creado una Apk en nuestro dispositivo móvil, y ya podremos usarla de forma normal, siempre y cuando hayamos iniciado el backend (API) en **Render** correctamente.

4. Prototipado

El prototipo de la App se ha realizado como primer paso haciendo uso de **Figma**.

Este prototipo es la versión inicial de la App, por lo que no contiene algunas pantallas añadidas más tarde en el proceso de creación de la App. Esto es debido a que han ido surgiendo nuevas necesidades de implementarlas a medida que se iba desarrollando.

5. Diseño funcional

5.1 Diagrama de Casos de Uso

5.1.1 Caso de uso Registro / Inicio de sesión

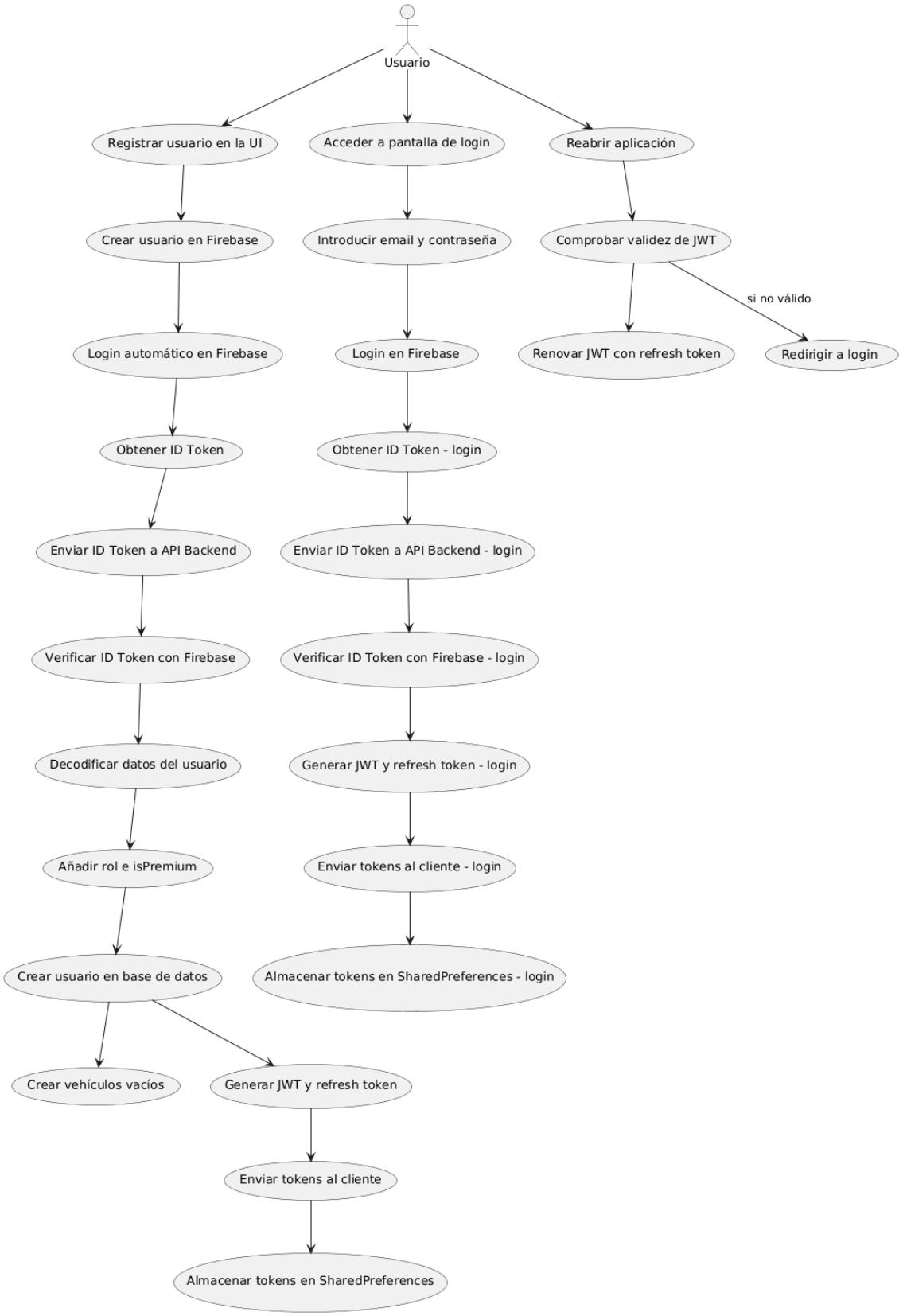
Cuando un usuario se registra en la aplicación, se ejecuta una serie de pasos para garantizar la correcta autenticación y vinculación del usuario en Firebase y en la API propia. Este caso de uso también contempla la funcionalidad de "Recordarme", usando refresh tokens.

Flujo resumido:

1. El usuario introduce su **email** y **contraseña** en la UI de registro.
2. El cliente envía estos datos a **Firebase Authentication**, que crea el usuario.
3. Automáticamente se hace login en Firebase para obtener un **ID Token** válido.
4. Este token se envía a la **API backend**, que verifica su validez con Firebase y decodifica sus datos.
5. La API crea un nuevo usuario con atributos adicionales como isPremium y rol (user/admin).
6. La API genera un **JWT** propio y un **refresh token**, que se devuelven al cliente.
7. El cliente los guarda en **SharedPreferences**.
8. Se llama a un endpoint adicional que crea los **vehículos iniciales vacíos** (coche, moto, bici).
9. Si el usuario entra de nuevo a la app y el JWT ha expirado, pero tiene activado "Recordarme", se utiliza el refresh token para obtener un nuevo JWT sin necesidad de loguearse manualmente.
10. Si el refresh token también ha expirado o no es válido, se redirige al usuario al login.

Aplicaciones:

- Registro de usuario
- Autenticación segura con verificación cruzada entre Firebase y backend
- Gestión de sesión persistente y renovación automática con refresh tokens
- Inicialización de vehículos del perfil

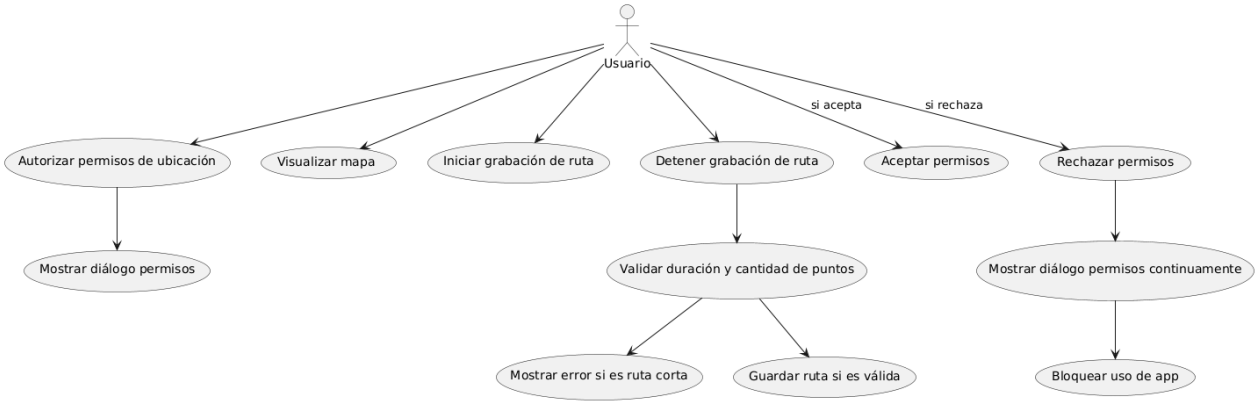


Img 22.Usecase auth

5.1.2 Caso de uso grabación ruta

El usuario debe autorizar permisos de ubicación para usar la app; si no los acepta, no podrá acceder a las funcionalidades y se le mostrará un diálogo persistente. Al aceptar, se muestra el mapa y el usuario puede comenzar a grabar su ruta pulsando un botón. Cuando decide detener la grabación, la app valida que la ruta tenga una duración y cantidad mínima de puntos. Si no cumple, muestra un error y no guarda la ruta; si sí cumple, se guarda correctamente.

Esta funcionalidad garantiza que el usuario siempre otorgue permisos esenciales para la ubicación y que solo se guarden rutas con datos suficientes, mejorando la calidad de la información y experiencia.



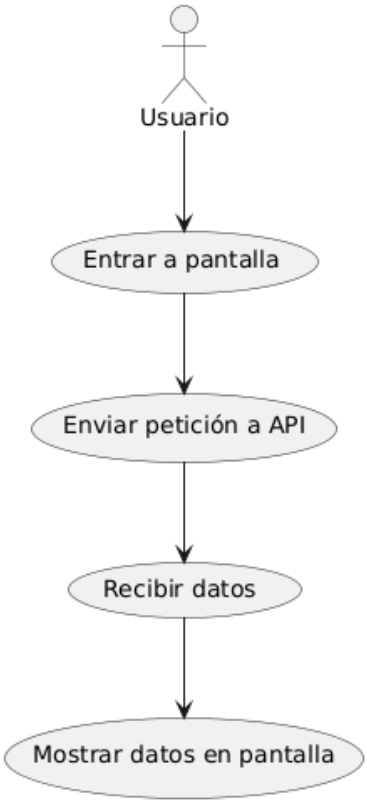
Img 23.Usecase grabación ruta

5.1.3 Caso de uso de carga de datos

Cuando el usuario accede a una pantalla de la aplicación la cual implique mostrar datos, automáticamente se realiza una petición a la API para obtener los datos correspondientes. Mientras carga los datos, se muestra un estado de carga para informar en todo momento al usuario. Una vez recibidos, estos datos se muestran en la pantalla para que el usuario pueda visualizarlos.

Este caso de uso aplica al entrar a las pantallas:

- Historial de rutas
- Detalles ruta
- Perfil



Img 24.Usecase carga datos

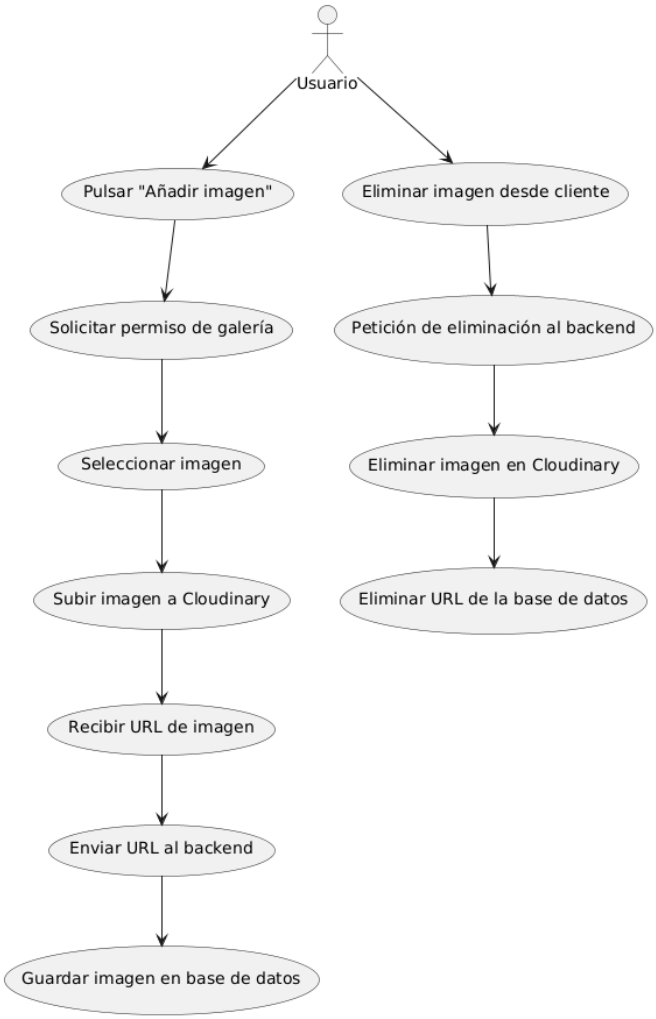
5.1.4 Caso de uso de subida / eliminación de imágenes

Cuando el usuario desea añadir una imagen (a una ruta o al perfil), el proceso sigue una secuencia estándar. Se solicita permiso de acceso a la galería si aún no se ha concedido. Tras seleccionar la imagen, esta se sube a **Cloudinary**. Si la subida es exitosa, se obtiene una URL que se envía al backend de la aplicación, donde se almacena en la base de datos.

En el caso de eliminación, el cliente realiza una petición a la API propia, que se encarga de eliminar tanto de la base de datos como de **Cloudinary**.

Este caso aplica a:

- Añadir imagen a ruta
- Añadir o editar imagen de perfil
- Eliminar imagen de ruta o de perfil



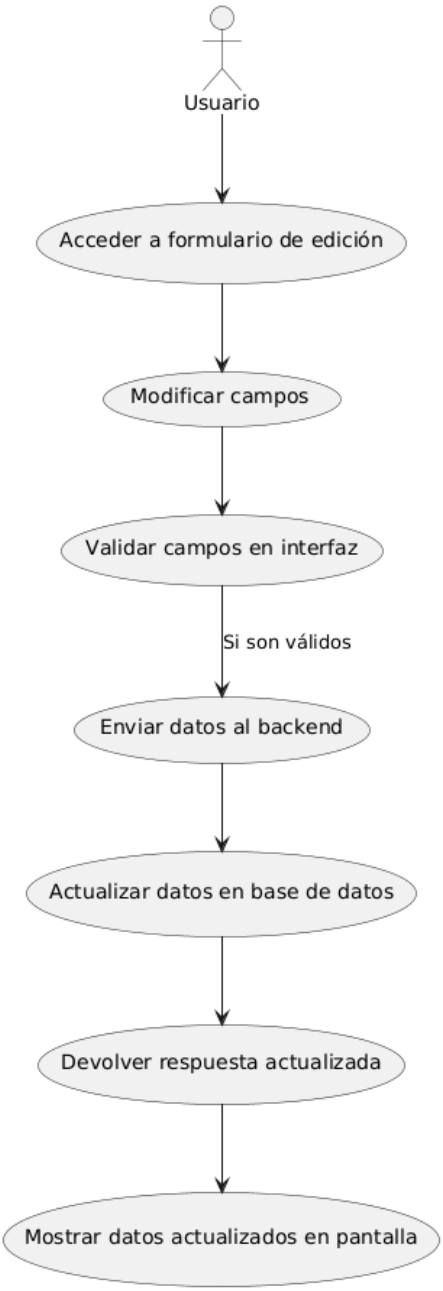
Img 25.Usecase imágenes

5.1.5 Caso de uso de edición de datos

Cuando un usuario desea modificar información dentro de la aplicación (como su perfil, los detalles de una ruta o los datos del vehículo), se siguen los mismos pasos generales. El usuario edita los campos, se valida la entrada desde la interfaz, y si los datos son válidos, se envía una petición al backend para actualizar la información. La respuesta actualizada es devuelta y mostrada en pantalla.

Este caso aplica a:

- Editar perfil de usuario
- Editar detalles de una ruta
- Editar información del vehículo



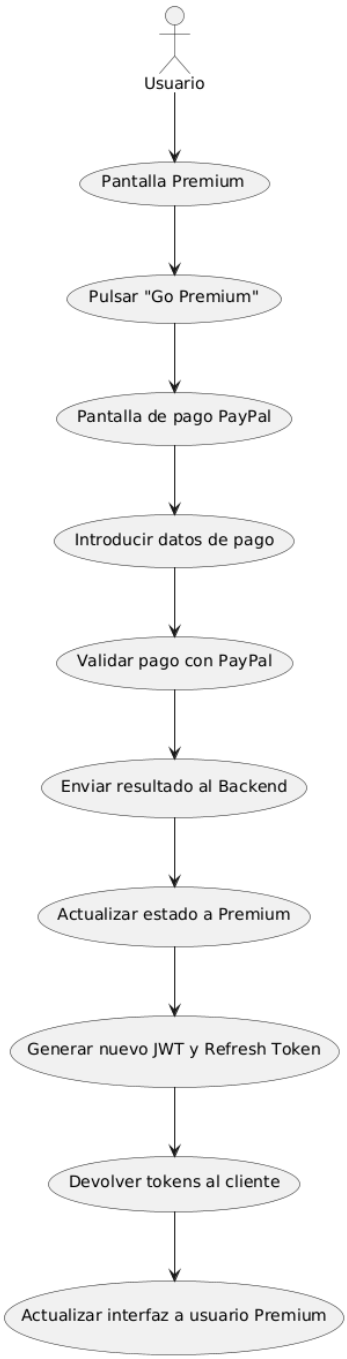
Img 26.Usecase edición datos

5.1.6 Caso de uso de suscripción premium

Cuando un usuario desea suscribirse a la versión Premium de la aplicación, realiza los siguientes pasos. Desde la pantalla Premium pulsa el botón de suscripción, que abre una pantalla integrada con la API de PayPal en entorno de pruebas (sandbox). Una vez validado correctamente el pago, se informa al backend para actualizar el estado del usuario. Este genera un nuevo JWT y refresh token y los envía al cliente. Finalmente, la pantalla de suscripción se actualiza para mostrar que el usuario ya es Premium.

Esto aplica a:

- Acceso a funciones exclusivas
- Proceso de renovación automática de token tras cambio de rol



Img 27.Usecase premium

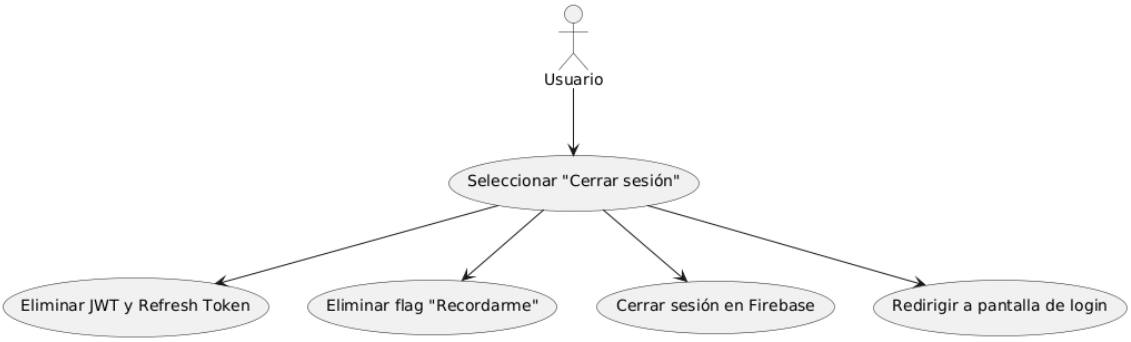
5.1.7 Caso de uso Cerrar Sesión

Cuando el usuario decide cerrar sesión desde la aplicación, se lleva a cabo una serie de acciones tanto a nivel de cliente como de autenticación:

- Se eliminan el JWT y el refresh token almacenados en SharedPreferences.
- Se elimina la opción “Recordarme” si estaba activada.
- Se realiza el signOut() de Firebase.
- Se redirige al usuario a la pantalla de login o inicio.

Aplicaciones:

- Garantiza seguridad del usuario
- Limpia completamente cualquier rastro de sesión activa



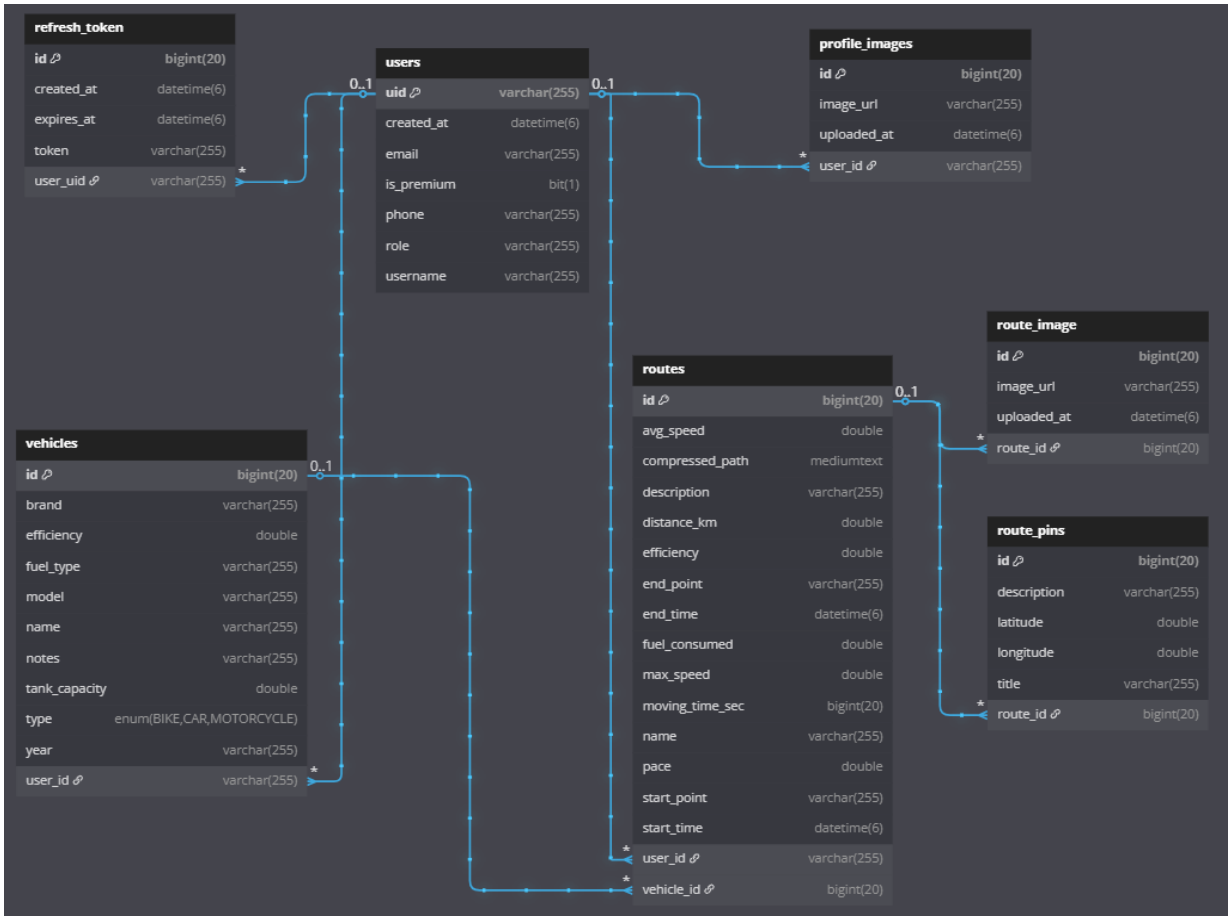
Img 28.Usecase logout

5.2 Diagrama E-R

El modelo E-R de la base de datos refleja las principales entidades del sistema **TrackMyRide** y sus relaciones:

- **User**: Representa a los usuarios registrados, identificados por un UID único proporcionado por Firebase.
- **ProfileImage**: Imagen de perfil asociada a un usuario, con una relación **1:1** con User.
- **Vehicle**: Vehículos registrados por cada usuario, con una relación **N:1** hacia User, ya que un usuario puede tener varios vehículos.
- **Route**: Rutas creadas y registradas por los usuarios, vinculadas tanto a un vehículo como a un usuario, estableciendo relaciones **N:1** con Vehicle y User.
- **RoutePin**: Puntos específicos dentro de una ruta, con una relación **1:N** con Route, ya que una ruta puede tener múltiples puntos.
- **RouteImage**: Imágenes asociadas a una ruta, con una relación **N:1** hacia Route.
- **RefreshToken**: Tokens de actualización para la gestión de sesiones y seguridad, asociados a un usuario con una relación **N:1** hacia User.

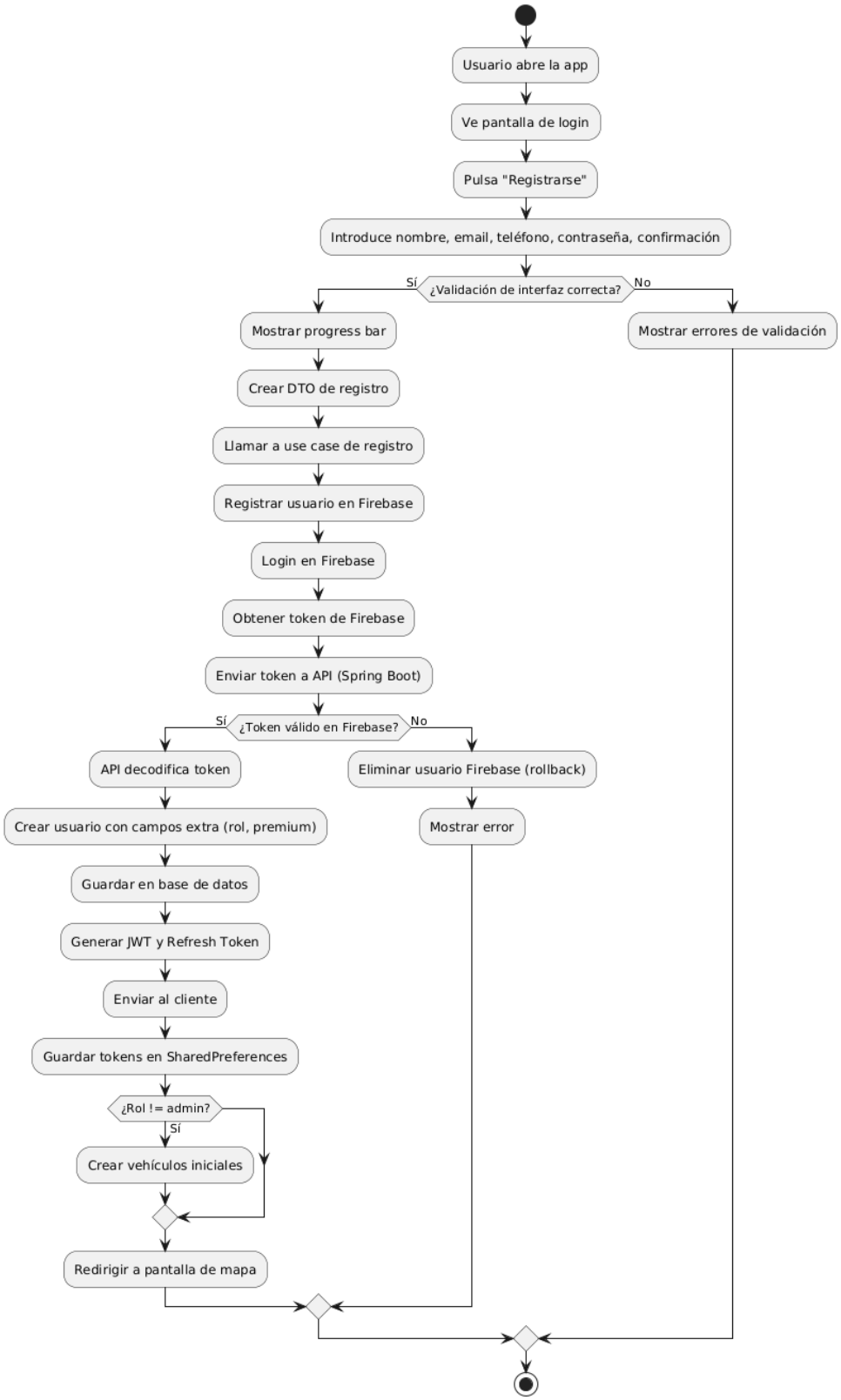
Este esquema garantiza la integridad referencial y una estructura clara para el manejo de la información.



Img 29.Diagrama E-R

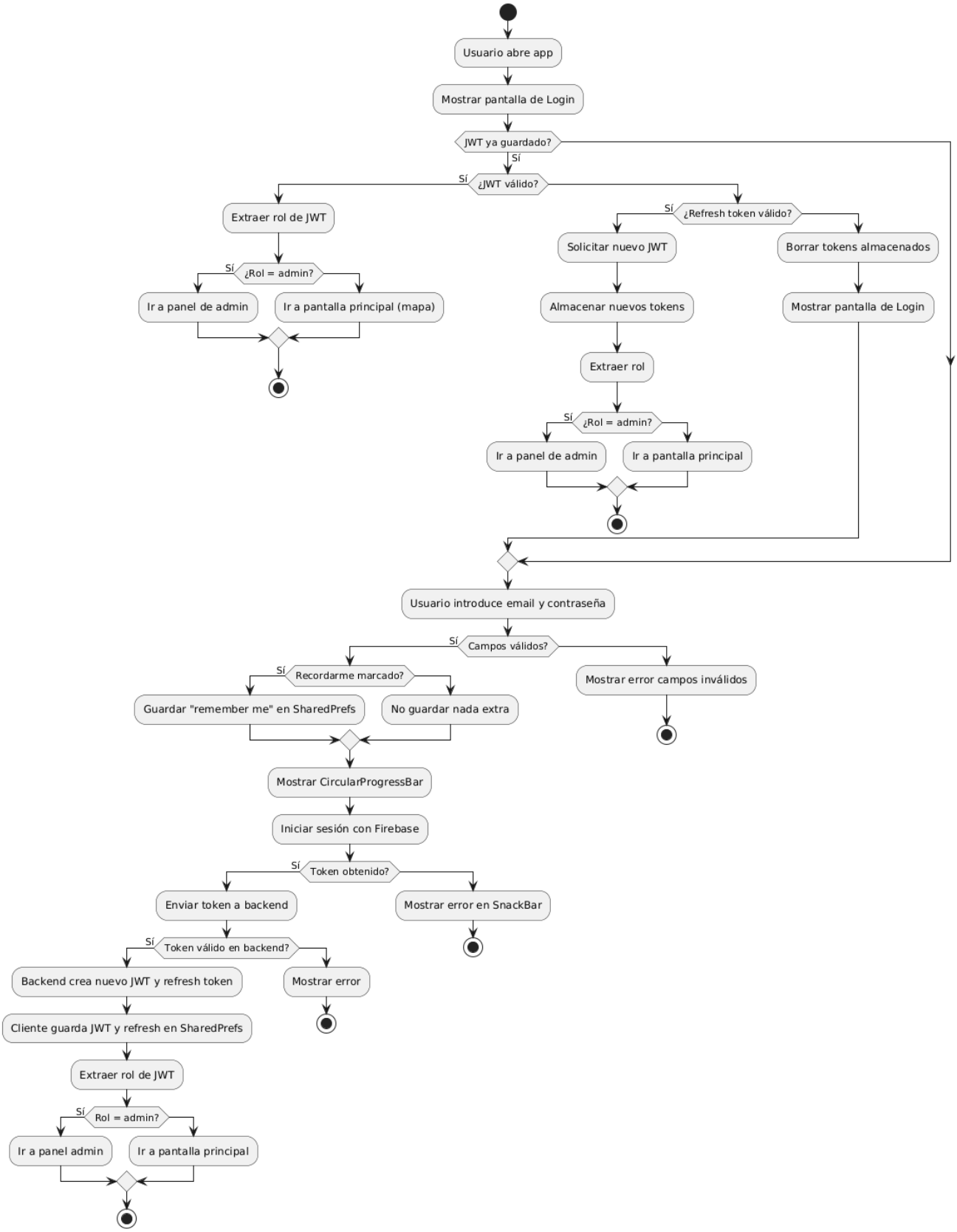
5.3 Diagrama de Flujo / Actividad

5.3.1 Diagrama UML de flujo Registro Usuario



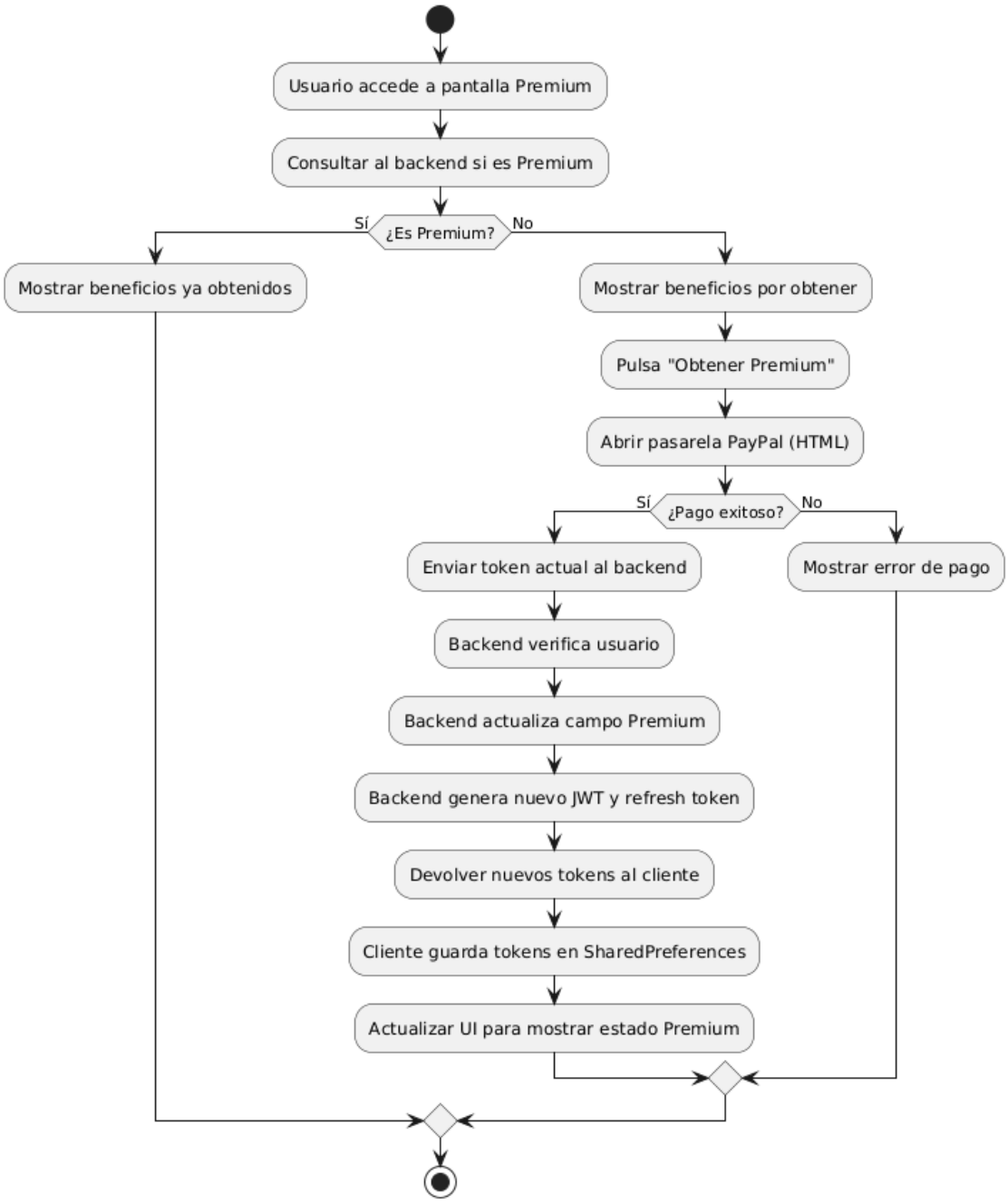
Img 30.Flujo registro

5.3.2 Diagrama UML de flujo Inicio de Sesión Usuario



Img 31Flujo inicio sesión

5.3.3 Diagrama UML de flujo Premium



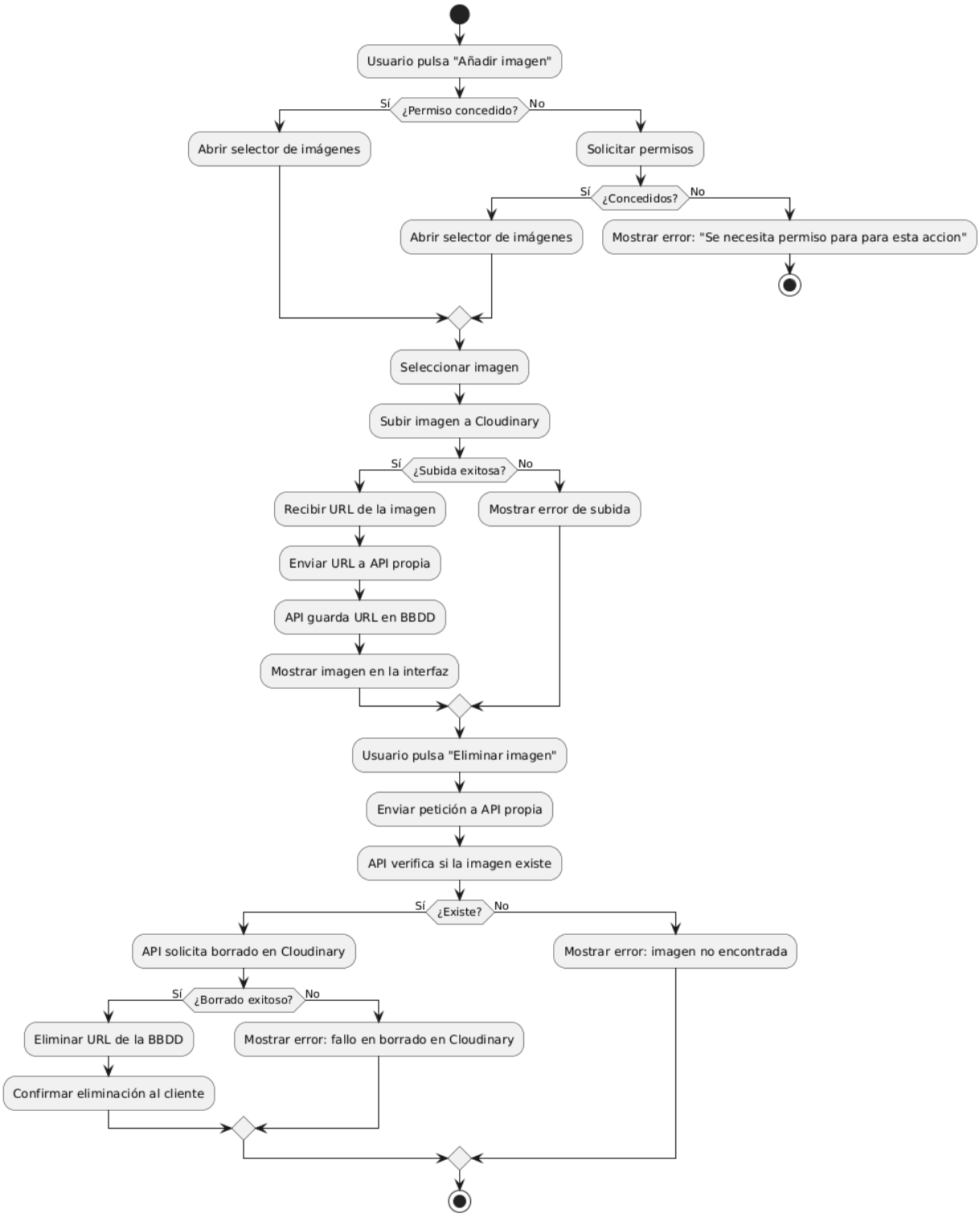
Img 32.Flujo premium

Este flujo representa una mejora significativa de experiencia: el usuario no tiene que cerrar sesión ni reiniciar la app tras el pago. La arquitectura garantiza seguridad (con nuevos tokens) y sincronización con el backend en tiempo real.

5.3.4 Diagrama UML de flujo Grabación Ruta

Img 33.Flujo grabación ruta

5.3.5 Diagrama UML carga y borrado imágenes



Img 34.Flujo imágenes

6. Desarrollo

6.1 Secuencia de Desarrollo

El desarrollo de la aplicación siguió un enfoque incremental, comenzando con la fase de diseño y evolucionando progresivamente hasta alcanzar una versión funcional completa (v1.0), sin versiones intermedias públicas ni despliegue de betas. A continuación, se detallan las fases principales:

- **Diseño y prototipado (marzo - abril)**
  El primer paso fue la creación de un prototipo visual completo de la aplicación en Figma, donde se definieron todas las pantallas necesarias y su navegación. Este prototipo sirvió como referencia base para la posterior implementación en Jetpack Compose.
  Una vez finalizado el diseño en Figma, se procedió a trasladarlo a código, implementando inicialmente todas las pantallas diseñadas con Jetpack Compose. Durante el desarrollo, surgieron nuevas necesidades que obligaron a añadir pantallas adicionales sobre la marcha.

- **Navegación y estructura inicial (marzo - abril)**
  Con las pantallas creadas, se integró un drawer de navegación lateral junto con su correspondiente topbar, estableciendo la arquitectura visual y la lógica de navegación entre vistas. Esta etapa se desarrolló entre el 22 de marzo y el 22 de abril, representando el esqueleto base de la aplicación.

- **Integración inicial de Firebase (abril)**
  El 30 de abril se implementó de forma básica la autenticación con Firebase, aunque en esta fase todavía no se utilizaba la API propia. Fue un paso inicial para validar el uso de Firebase Auth como proveedor de identidad.

- **Inyección de dependencias y persistencia de sesión (mayo)**
  El 4 de mayo se introdujo Dagger Hilt en el proyecto para la inyección de dependencias. Además, se configuró Firebase Auth correctamente y se añadió la funcionalidad de "Recordar sesión" mediante SharedPreferences. También se implementó el mecanismo de logout de usuario.

- **Permisos, trackeo y compresión de rutas (7 - 13 mayo)**
  Durante esta semana, se desarrolló el sistema de gestión de permisos general, sobre todo el de ubicación, y se implementó el trackeo de rutas en tiempo real. Asimismo, se añadió un sistema de compresión y simplificación de rutas para minimizar el peso de los datos almacenados.

- **Integración de backend propio con Spring Boot (14 - 16 mayo)**
  Se crearon las interfaces de Retrofit y los *use cases* para conectar la app con la API REST desarrollada en Spring Boot. En esta etapa también se finalizó el login y el registro con JWT, combinando Firebase Auth con el backend propio para un sistema de autenticación híbrido y seguro.

- **Estadísticas, panel de administrador y mejoras (18 - 22 mayo)**
  Se desarrolló la funcionalidad de cálculo de estadísticas asociadas a las rutas (distancia, tiempo, velocidad media, etc.) y se implementó una vista de administración con capacidades básicas. Posteriormente se incorporó el

sistema de pagos en modo *sandbox* con PayPal y se añadió la opción de exportar y compartir rutas.

- **Optimización y mejoras finales (30 mayo - 3 junio)**
  En esta fase se llevaron a cabo múltiples tareas de refactorización y mejora tanto visual como lógica. Se introdujo la funcionalidad de añadir pines en rutas, y se refactorizaron las llamadas a la API para realizarse en hilos separados, evitando bloqueos en la interfaz principal.
- **Versión final 1.0**
  El resultado de este proceso es la versión actual (v1.0), plenamente funcional, con autenticación híbrida, grabación y visualización de rutas, gestión de sesión, estadísticas, navegación completa, sistema de pago simulado para obtención premium, sistema de premium que obtiene funcionalidad extra, y visualización de mayor cantidad de datos.

Durante todo el desarrollo, la app fue probada manualmente tanto en emulador como en dispositivo real, grabando rutas en exteriores con coche y moto para validar la precisión del trackeo y la estabilidad de la aplicación en condiciones reales.

6.2 Dificultades encontradas y soluciones

Durante el desarrollo del proyecto surgieron varios retos técnicos, que se resolvieron aplicando diferentes estrategias y herramientas. A continuación, se describen las más relevantes:

Permisos de ubicación y grabación de rutas

Una de las principales dificultades encontradas fue la gestión de los permisos de ubicación. La aplicación depende completamente de la localización del usuario, por lo que si este no concede el permiso, la funcionalidad principal queda completamente inutilizada.
Esto obligó a diseñar un flujo claro para el usuario, mostrando mensajes explicativos que justificaran la necesidad del permiso. Sin embargo incluso tras denegar el permiso dos veces como permite Android, la única forma de concederlo es accediendo manualmente desde los ajustes del sistema, opción que también se facilita con un botón para que el usuario acceda directamente a los ajustes de la App. Esta restricción complicó la experiencia de usuario y supuso un reto a nivel de diseño y comunicación en la interfaz.

Además, otro reto relacionado fue el de capturar la ubicación del usuario de forma constante, precisa y eficiente. Debido a las limitaciones de recursos del dispositivo y al potencial elevado número de puntos de GPS que se podrían registrar en rutas largas, se decidió implementar un sistema de obtención inteligente de posiciones.

En lugar de capturar una localización en intervalos de tiempo fijos, se optó por registrar un nuevo punto solo si se ha avanzado una determinada cantidad de metros desde el último punto válido. Esto permitió reducir significativamente el número de posiciones almacenadas sin comprometer la fidelidad del trazado de la ruta.

Por otro lado, fue necesario sincronizar esta lógica con la parte visual de la aplicación. La pantalla del mapa debía reflejar el avance del usuario en tiempo real, por lo que se diseñó un sistema que actualiza suavemente la posición del marcador en pantalla mediante animaciones, logrando una experiencia fluida y continua para el usuario, sin saltos bruscos ni sobrecarga gráfica.

Sincronización cliente-servidor y refresh de tokens

Otro reto fue el sistema híbrido de autenticación: se utilizó Firebase para la gestión de usuarios, pero se necesitaba generar un JWT en el backend propio para acceder a recursos protegidos, ya que añade el rol y el estado de premium.
Esto implicaba obtener el token de Firebase desde la app y enviarlo al backend, donde se validaba y generaba un JWT adicional.

Además, se implementó un sistema de **refresh tokens** para renovar los tokens expirados, si el usuario usó “remember me” al iniciar sesión sin forzarlo a iniciar sesión de nuevo. Esto implicó lógica tanto en la app como en el backend, incluyendo control de expiración, validaciones y persistencia de tokens seguros.

Tests y depuración de mapas en Android

La depuración de la funcionalidad de mapas (Google Maps) presentó dificultades, especialmente al tratarse de trackeo en condiciones reales (movimiento, cambios de red, etc.).

Se optó por hacer pruebas reales directamente con un dispositivo Android, instalando la APK y grabando rutas reales en entornos urbanos y rurales. Esto permitió identificar problemas como pérdidas de señal GPS, fallos en el refresco visual del mapa o imprecisiones al comprimir rutas, que se solucionaron ajustando los algoritmos de simplificación y mejorando el sistema de gestión de eventos del mapa.

También se implementaron logs detallados y herramientas de depuración interna para poder analizar el comportamiento del sistema durante el trackeo.

6.3 Decisiones de diseño y arquitectura

Durante el desarrollo del proyecto se tomaron una serie de decisiones estratégicas en cuanto al diseño visual, la estructura interna de la app y la arquitectura del sistema completo, tanto del lado del cliente como del servidor. Estas decisiones no solo afectaron al rendimiento y mantenibilidad del código, sino que también marcaron la forma en que se desarrollaron nuevas funcionalidades y se organizó el trabajo técnico.

Arquitectura de la App Android: MVVM limpio y escalable

Desde el inicio se decidió implementar una arquitectura basada en el patrón **MVVM (Model-View-ViewModel)**, siguiendo principios de **clean architecture** adaptados al desarrollo móvil. Esta decisión se fundamentó en la necesidad de mantener el código desacoplado, fácilmente testeable y preparado para escalar en el futuro con nuevas funcionalidades.

La estructura general siguió un enfoque en capas claramente definidas:

- **View (UI en Jetpack Compose)**: se encargan únicamente de mostrar los datos y de recoger las interacciones del usuario. La UI es declarativa y reactiva, adaptándose automáticamente a los cambios de estado.
- **ViewModel**: gestiona el estado de la interfaz, contiene lógica de presentación y expone estados que la vista observa. No tiene referencias a la UI.
- **Use Cases (Casos de uso)**: encapsulan lógica de negocio concreta (como “guardar una ruta”, “cargar estadísticas del usuario”, “enviar token a backend”, etc.), permitiendo mantener una única fuente de verdad de cada operación.
- **Repositories (Interfaces)**: actúan como punto intermedio entre los casos de uso y las fuentes de datos. Se inyectan en los use cases y permiten alternar entre distintas implementaciones (por ejemplo, datos locales o remotos).
- **Data Layer**:
    - **Remote**: Implementaciones de los repositorios que hacen llamadas HTTP usando Retrofit a la API Spring Boot, con validación de tokens.
    - **Local**: Uso de SharedPreferences para funcionalidades como "Recordarme" o persistencia simple sin base de datos.
- **Dependency Injection con Dagger Hilt**: se utilizó para inyectar ViewModels, Repositorios, Retrofit, casos de uso y otros componentes, lo que redujo el acoplamiento entre clases y facilitó la escalabilidad y testeo.

Este enfoque modular y organizado permitió que las funcionalidades fueran fácilmente testeables, mantenibles y reutilizables.

Jetpack Compose vs XML tradicional

En cuanto a la construcción de la interfaz gráfica, se optó desde el inicio por **Jetpack Compose** en lugar del enfoque clásico de XML. Esta decisión se tomó tras valorar varias ventajas claras:

- **UI declarativa y reactiva**: el estado de la aplicación se refleja directamente en la interfaz sin necesidad de manejar manualmente actualizaciones de vistas.
- **Reutilización de componentes**: Compose permite construir componentes personalizados y reutilizables con facilidad, como la TopBar, botones comunes, tarjetas de ruta, campos de texto, etc.
- **Menos errores y más productividad**: La integración con Kotlin permite escribir menos código, más limpio y con menos posibilidad de errores de tipo.
- **Compatibilidad con herramientas modernas**: Compose se integra naturalmente con ViewModel, Flows, Navigation, etc., y se mantiene alineado con la evolución del ecosistema Android.

Además, Compose resultó ser una herramienta clave durante el diseño y prototipado de pantallas, permitiendo iterar rápidamente en las vistas antes de tener completamente desarrollada la lógica detrás.

Backend propio con Spring Boot + JWT vs Firebase puro

Inicialmente, se utilizó **Firebase Authentication** de forma sencilla para validar usuarios y gestionar el inicio de sesión con correo y contraseña. Sin embargo, con el tiempo se optó por una arquitectura más robusta y personalizada, basada en:

- **API propia desarrollada en Kotlin + Spring Boot**
- **Validación del token de Firebase y emisión de JWT personalizados**
- **Control total sobre usuarios, roles, rutas, imágenes, pagos, etc.**

Esta transición tuvo como objetivo ampliar el control, la seguridad y la escalabilidad del sistema, permitiendo implementar funcionalidades avanzadas sin depender completamente del ecosistema **Firebase**.

Las razones principales para esta decisión fueron:

- **Escalabilidad**: La API permite extender el sistema con módulos como estadísticas detalladas, sistema de pagos premium, gestión de imágenes o funcionalidades exclusivas, sin limitaciones impuestas por Firebase.
- **Seguridad personalizada**: Se diseñó un sistema de autenticación robusto utilizando JWT firmados y refresh tokens, garantizando sesiones seguras, controladas y renovables desde el servidor.
- **Control sobre roles y permisos**: Se implementó un sistema básico de roles como "usuario" y "admin", permitiendo validar privilegios directamente en los endpoints protegidos. Además, se añadió la posibilidad de asignar y revocar permisos o modificar atributos del usuario desde un **panel de administración** propio.
- **Gestión segura de usuarios premium**: Uno de los principales motivos para contar con backend propio fue la necesidad de controlar de forma segura y centralizada el sistema de **suscripción premium**. Desde la API, se puede:
    - Comprobar si un usuario es premium o no antes de permitir el acceso a ciertas rutas o funcionalidades.
    - Actualizar el estado premium desde el servidor tras recibir confirmación de pago (por ejemplo, vía PayPal).
    - **Revocar privilegios premium manualmente** en caso de errores, fraude o comportamiento indebido, a través del panel de administración.
    - Registrar y auditar qué usuarios han accedido a contenido premium.
- **Gestión de imágenes con Cloudinary**: La API se encargó también de la subida, borrado y recuperación de imágenes de rutas y perfiles, delegando esta responsabilidad a Cloudinary pero manteniendo el control desde el backend.

Este enfoque híbrido, que combina la agilidad inicial de Firebase con la potencia y personalización de una API propia, permitió **comenzar rápido** el desarrollo y luego evolucionar hacia una **solución profesional y segura** sin comprometer el control ni la flexibilidad.

Comunicación cliente-servidor: Retrofit + Token Management

La app se comunica con la API utilizando **Retrofit** como cliente HTTP, siguiendo una arquitectura robusta de peticiones autenticadas. Algunos puntos clave de esta implementación fueron:

- **Interceptors personalizados**: Se añadieron interceptores que incluyen automáticamente el JWT en cada petición.
- **Manejo de expiración del token**: Si el servidor responde con un error de token expirado, se intenta renovar automáticamente usando el refresh token y se reintenta la operación.
- **Petición segura de rutas, estadísticas y subida de imágenes**: Se aplicaron encabezados de autenticación y validación de roles en los endpoints.
- **Separación clara de interfaces (APIs)**: Las llamadas HTTP están organizadas por áreas funcionales (autenticación, rutas, usuario…), y su lógica encapsulada en los repositorios.

En conjunto, todas estas decisiones permitieron que tanto la app móvil como el backend fueran **mantenibles, modulares y escalables**, facilitando tanto el desarrollo actual como posibles evoluciones futuras, como versiones web, incorporación de nuevos métodos de pago o funcionalidades premium más complejas.

6.4 Control de versiones y revisión de código

Durante todo el proceso de desarrollo, se utilizó Git como sistema de control de versiones, con repositorios alojados en GitHub. Se mantuvieron dos repositorios independientes, uno para la aplicación móvil y otro para la API backend, lo que permitió una gestión clara y separada de las dos partes principales del proyecto.

Aunque se trabajó en solitario, se mantuvo una disciplina rigurosa en el manejo de versiones. Desde el inicio, todos los cambios significativos se registraron mediante commits claros y descriptivos, seguidos de push hacia la rama principal (master). En este proyecto no se implementó una estrategia formal de ramas —como GitFlow o desarrollo trunk-based—, sino que todo el trabajo se realizó directamente sobre la rama principal. Esta simplicidad facilitó la fluidez en el desarrollo, evitando sobrecargas de gestión que no eran necesarias para un proyecto individual.

La revisión de código, en ausencia de otros desarrolladores, fue un proceso autoimpuesto. Antes de subir cualquier cambio, se realizó una inspección minuciosa del código para garantizar que no existieran errores evidentes ni conflictos. De esta forma, se aseguraba la calidad del código sin la necesidad de pull requests o revisiones externas. Esta práctica fomentó la responsabilidad sobre el código propio y evitó posibles regresiones o bugs innecesarios.

Respecto a la automatización y control de calidad, el proyecto no incorporó herramientas automáticas de integración continua (CI) ni hooks específicos de Git para pre-commit o pre-push. También se escribieron tests unitarios, especialmente en la API, que se ejecutaban localmente para asegurar la estabilidad antes de cada actualización.

Para organizar las tareas y prioridades, se empleó la herramienta Notion, específicamente su módulo de to-do. Este espacio sirvió como lista personal de pendientes donde se registraban las funcionalidades a implementar, correcciones, mejoras o ideas, clasificándolas según su prioridad (alta, media o baja). Además, se anotaban las fechas de finalización de cada tarea, lo que permitió llevar un seguimiento ordenado y metódico del progreso. Esta gestión mental y documental ayudó a mantener el foco y a evitar olvidos, reduciendo el esfuerzo cognitivo diario y permitiendo un desarrollo más eficiente.

En resumen, el control de versiones y revisión de código estuvo orientado a un desarrollo ágil y eficiente para un proyecto individual, apoyado en buenas prácticas básicas, autoevaluación constante y una organización personal clara. Esto garantizó un código estable, entendible y con un historial transparente de evolución, facilitando futuras ampliaciones y mantenimiento.

7. Pruebas

7.1 Pruebas unitarias en la aplicación Android

Durante el desarrollo de la aplicación Android, se implementaron pruebas unitarias centradas en el comportamiento de la interfaz de usuario y la lógica asociada a pantallas clave como la de login. Estas pruebas se desarrollaron utilizando **Jetpack Compose Test**, el framework de pruebas específico para interfaces desarrolladas con Compose, junto con **ViewModels falsos (FakeViewModels)** para simular distintos estados sin necesidad de conexión a internet ni dependencias externas.

Objetivos de las pruebas

Las pruebas se enfocaron en garantizar:

- Que la lógica del login se comporta correctamente ante entradas válidas e inválidas.
- Que la navegación responde de manera adecuada al tipo de usuario (normal o administrador).
- Que se muestra el mensaje de error correspondiente en caso de fallo.
- Que los elementos visuales reaccionan a las interacciones del usuario (checkbox de "Recordarme", visibilidad de contraseña…).
- Que los ViewModels se actualizan correctamente cuando el usuario interactúa con los componentes de UI.

Metodología y enfoque

Se utilizó una clase de prueba principal LoginScreenTest, en la cual se simulan diferentes flujos de uso mediante la inyección de un FakeLoginViewModel y un FakeSessionViewModel. De esta forma, se puede controlar de forma programada el estado de la lógica interna y validar su repercusión en la interfaz.

Se cubrieron múltiples casos, entre los que se incluyen:

- **Inicio de sesión exitoso con usuario normal**: Se verifica que se navega a la pantalla principal.
- **Inicio de sesión exitoso con usuario administrador**: Se comprueba que se accede al panel de administración.
- **Inicio de sesión fallido**: Se valida que se muestra un mensaje en el snackbar y no hay navegación.
- **Actualización del campo de contraseña**: Se asegura que los cambios en el campo de texto se reflejan en el ViewModel.
- **Interacción con el checkbox “Recordarme”**: Se comprueba que se alterna correctamente el estado booleano.
- **Botón de mostrar/ocultar contraseña**: Se valida que cambia el estado de visibilidad en el ViewModel.

- **Ausencia de errores al iniciar sesión con credenciales válidas**: Se asegura que no se muestra el snackbar si el login es correcto.

En total, se han desarrollado al menos 6 pruebas de comportamiento representativas de la interacción usuario-interfaz, cubriendo de forma efectiva los escenarios esperables más relevantes.

Se pueden ver los ejemplos concretos en código en el siguiente enlace: Tests

Comprobación en dispositivos reales

Además de las pruebas automatizadas, se realizaron **pruebas manuales exhaustivas en distintos dispositivos físicos** con diferentes versiones de Android, resoluciones y configuraciones. Esto permitió verificar el correcto funcionamiento de:

- Animaciones de seguimiento en tiempo real de rutas.
- Solicitud y gestión de permisos de ubicación y acceso archivos.
- Diferencias en comportamiento entre modo claro/oscuro.
- Usabilidad general en pantallas pequeñas y grandes.
- Flujo completo de inicio de sesión, registro, y uso de rutas.

También se contó con la ayuda de **usuarios externos (amigos y familiares)** que probaron la aplicación, permitiendo detectar errores de usabilidad o pequeños bugs no contemplados inicialmente.

7.2 Pruebas en la API

La API desarrollada en Spring Boot fue sometida a pruebas exhaustivas para asegurar la robustez de la lógica de negocio, el correcto control de acceso a los recursos, y el comportamiento esperado de los endpoints bajo distintos escenarios.

Tipos de pruebas realizadas

Se implementaron dos tipos principales de pruebas:

1. **Pruebas unitarias de servicios (Service)**
2. **Pruebas de integración de controladores (Controller)**

Pruebas unitarias

Cada clase de servicio (como RouteService, UserService, etc.) fue probada de forma aislada mediante **mocking de las dependencias** usando la librería MockK, con el objetivo de:

- Validar la lógica interna sin depender de la base de datos ni de Firebase.
- Comprobar el lanzamiento de excepciones personalizadas como NotFoundException o ForbiddenException.
- Verificar que se llaman los métodos esperados en los repositorios o servicios auxiliares.
- Probar rutas protegidas en función del UID del token JWT y el rol del usuario.

Ejemplo: En el test RouteServiceTest, se valida que:

- Se devuelve una ruta si el usuario está autorizado.
- Se lanza una excepción NotFoundException si la ruta no existe.
- Se lanza ForbiddenException si el usuario no tiene permiso.

Esto garantiza que el control de acceso y la lógica de negocio están bien desacopladas y probadas.

Pruebas de integración

Además de pruebas unitarias, se crearon **pruebas de integración completas** con MockMvc para simular peticiones HTTP reales a los endpoints expuestos en los controladores. Estas pruebas corren en un contexto de Spring Boot real (anotación @SpringBootTest) y permiten:

- Verificar que los endpoints aceptan peticiones correctamente formadas.
- Comprobar el flujo de autenticación vía Firebase.
- Validar respuestas HTTP (como códigos 200 OK, 401 Unauthorized, etc.).
- Testear la correcta serialización/deserialización de DTOs.

Importante: Para estas pruebas de integración se utiliza una **base de datos en memoria H2**, configurada específicamente para el perfil de pruebas (@ActiveProfiles("test")). Esta elección permite:

- Aislar completamente los tests del entorno de producción o desarrollo.
- Ejecutar las pruebas con rapidez y sin necesidad de dependencias externas.
- Garantizar un estado limpio de base de datos antes de cada ejecución, gracias al soporte de Spring para la inicialización automática de esquemas y datos.

Ejemplo: En el test AuthControllerIntegrationTest, se simula el proceso completo de registro de un nuevo usuario:

- Se mockea FirebaseAuth para devolver un FirebaseToken válido.
- Se envía una petición POST a /auth/register con el header Authorization.
- Se verifica que la respuesta del endpoint es 200 OK.

Este tipo de pruebas asegura que el sistema completo funciona correctamente en condiciones reales, incluyendo validación, seguridad y formato de entrada/salida.

Se pueden ver los ejemplos concretos en código en los siguientes enlaces:

Unitarios, Integración

7.3 Supervisión y gestión de errores con Firebase Crashlytics

Durante el proceso de desarrollo y prueba de la aplicación, se consideró fundamental contar desde un principio con una herramienta que permitiera detectar y registrar errores que se produjeran tanto en la fase de pruebas como en producción. Por ello, se decidió implementar **Firebase Crashlytics**, un servicio ofrecido por Firebase que proporciona informes detallados sobre errores en tiempo real, facilitando así la identificación de problemas y la mejora continua del proyecto.

Crashlytics permite recopilar automáticamente información sobre fallos que ocurren en la aplicación, incluyendo:

- El tipo de error o excepción generada.
- El dispositivo y versión del sistema operativo donde ocurrió.
- El hilo (thread) afectado.
- Un historial (stack trace) del error que ayuda a entender el contexto del fallo.

Objetivos de uso de Crashlytics

El principal objetivo al integrar Crashlytics fue mejorar la calidad del producto detectando errores que no aparecían durante las pruebas locales o manuales. Al recibir informes automáticos en tiempo real, fue posible:

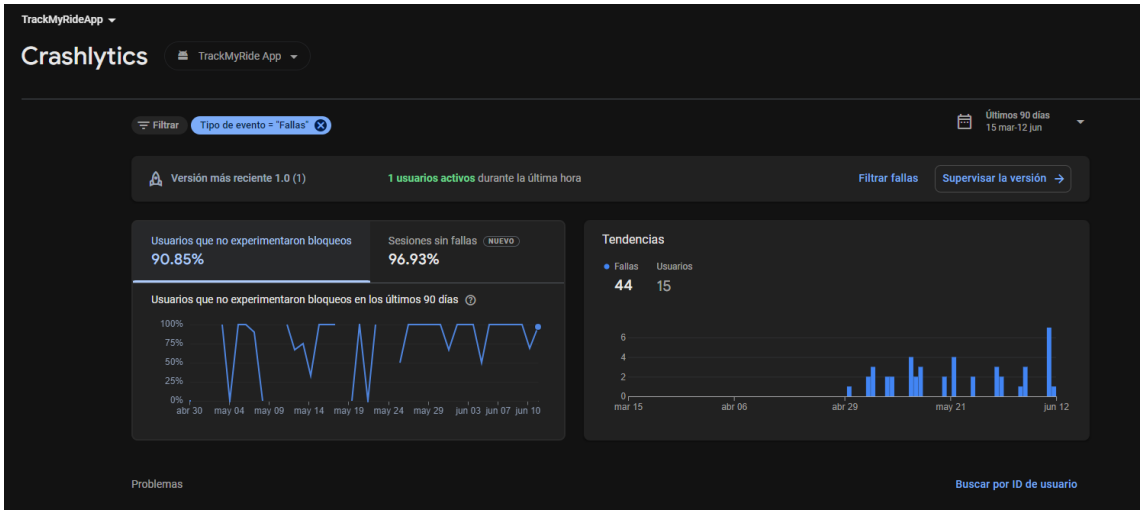
- Corregir errores que solo ocurrían en ciertos dispositivos o versiones del sistema operativo.
- Reducir el tiempo de diagnóstico al recibir descripciones claras y trazas del error.
- Obtener métricas sobre la estabilidad general de la aplicación, como el número de errores por sesión.

Integración en el proyecto

La integración se realizó utilizando el SDK oficial de Firebase para Android. Fue necesario:

- Añadir las dependencias necesarias en el archivo de configuración del proyecto.
- Configurar la clave de Firebase correspondiente.
- Verificar que los informes de errores se estuvieran enviando correctamente mediante pruebas de error forzadas.

A continuación, muestro una imagen del panel de Crashlytics y los datos que muestra (siendo estos datos de todo el proceso de desarrollo no únicamente del testeo final)



Img 35. Panel Crashlytics

7.4 Conclusión de las pruebas

El proceso de pruebas, tanto en la aplicación móvil como en la API, ha sido fundamental para garantizar la calidad, estabilidad y seguridad del sistema completo. A través de la combinación de **pruebas automatizadas y validaciones manuales**, se ha logrado un nivel de confianza elevado en el correcto funcionamiento de los distintos componentes que conforman la solución.

Fiabilidad alcanzada a través de pruebas unitarias

Tanto en el frontend (app Android Jetpack Compose) como en el backend (API Spring Boot), las pruebas unitarias han permitido validar la lógica específica de cada módulo o función de forma aislada. Este enfoque ha facilitado:

- Detectar errores tempranamente durante el desarrollo.
- Asegurar que los cambios realizados no introducen regresiones.
- Mantener un diseño desacoplado, facilitando el testeo y mantenimiento.

En la app, se probaron interacciones con la interfaz, navegación condicional según roles (USER vs ADMIN), visibilidad de la contraseña, estados del ViewModel, etc.

En la API, las pruebas de los servicios garantizaron que:

- Se controlan adecuadamente los permisos de acceso mediante el Jwt.
- Se lanza la excepción correcta ante errores de negocio.
- El comportamiento ante datos nulos o incorrectos es predecible y seguro.

Garantía de funcionamiento end-to-end mediante pruebas de integración

Las pruebas de integración en la API, han sido clave para comprobar que el sistema completo responde correctamente bajo un flujo realista de ejecución. Por ejemplo:

- El proceso de autenticación vía Firebase fue completamente simulado, lo que permitió validar que el sistema reacciona correctamente ante tokens válidos o inválidos.
- El registro de usuarios, y el acceso protegido a rutas u otros recursos, pudo validarse sin depender de un entorno de producción.

Este tipo de pruebas ayuda a identificar errores que solo emergen cuando varios componentes interactúan entre sí, como inconsistencias entre capas o fallos en la serialización de objetos.

Pruebas manuales y validación por terceros

Además de las pruebas automatizadas, se realizaron **pruebas manuales exhaustivas** tanto en la app como en la API. Estas incluyeron:

- Uso intensivo de la aplicación en distintos dispositivos Android (teléfonos y emuladores) con diferentes tamaños de pantalla y versiones del sistema operativo.

- Pruebas de escenarios de uso extremo: campos vacíos, conexiones inestables, errores de autenticación, datos inconsistentes.
- Validación por parte de diferentes usuarios (compañeros y usuarios no técnicos), quienes detectaron detalles de usabilidad o errores en flujos que no se habían considerado inicialmente.

Estas pruebas fueron esenciales para comprobar aspectos como:

- Experiencia de usuario (UX) real.
- Comportamiento visual y funcional bajo condiciones reales.
- Corrección de errores poco frecuentes o específicos de dispositivos.

Cobertura y resultados generales

En términos generales:

- Se implementaron **más de 60 pruebas unitarias** para la API.
- Se desarrollaron **6 pruebas específicas para la pantalla de login en la app**, cubriendo interacción, visibilidad, estados y validaciones.
- Se verificaron **funcionalidades clave de forma manual** por parte del desarrollador y testers externos.
- Todas las pruebas automatizadas pasan con éxito al momento del despliegue.

8. Distribución

La distribución de la aplicación TrackMyRide abarca tanto la publicación del backend como la entrega del cliente móvil Android. El objetivo de este proceso es garantizar que el sistema pueda ser ejecutado de forma autónoma por usuarios, testers o evaluadores sin necesidad de un entorno de desarrollo local.

Aunque el proceso de instalación y despliegue ya se ha explicado detalladamente en el punto 5, en este apartado se describe específicamente el enfoque adoptado para la **distribución final del software**, así como las **tecnologías utilizadas para su empaquetado y publicación**.

8.1 Distribución del backend (API REST)

El backend de TrackMyRide, desarrollado con **Spring Boot**, es empaquetado y distribuido como un archivo .war, el cual se genera mediante el sistema de construcción **Gradle**. Este archivo contiene la aplicación lista para ejecutarse en un contenedor servlet como Tomcat.

Tecnología de distribución:

- **Render**: se utiliza como plataforma de despliegue continuo. Render permite configurar el entorno, definir las variables de entorno necesarias, y lanzar el servicio web automáticamente tras recibir cambios desde GitHub.
- **Dropbox** (o servicios similares): se emplea como almacenamiento temporal del archivo .war. Este archivo es posteriormente descargado desde Render utilizando un Dockerfile personalizado.
- **Railway**: sirve como proveedor de base de datos en la nube (MySQL), ofreciendo una solución rápida para la persistencia de datos remota y accesible desde el backend desplegado.

Esta combinación de herramientas permite una **distribución cloud completamente funcional**, replicando el entorno local, y facilitando el acceso al sistema desde cualquier parte del mundo sin configuraciones manuales.

8.2 Distribución de la app móvil (Cliente Android)

La aplicación móvil está desarrollada con **Jetpack Compose**, el moderno framework declarativo de interfaces de usuario para Android, lo que facilita la creación de interfaces reactivas y altamente personalizables.

Formato de distribución:

La aplicación se distribuye en forma de archivo **APK (Android Package)**, el cual puede instalarse manualmente en cualquier dispositivo Android compatible. Este archivo contiene todo lo necesario para ejecutar la app sin necesidad de Android Studio ni conexión al backend local (si se usa la versión desplegada en Render).

Tecnologías implicadas:

- **Android Studio**: utilizado tanto para el desarrollo como para la generación del .apk de distribución.
- **Firebase**: empleado para la autenticación de usuarios y otras funcionalidades en la nube (como el control de sesiones).
- **Cloudinary, PayPal SDK y Google Maps**: integrados mediante claves API configuradas en el archivo local.properties.

Proceso de exportación del APK:

Durante la fase de desarrollo, se suele ejecutar la app directamente en un dispositivo conectado por USB (modo *debug*). Sin embargo, ese APK no es apto para distribución, ya que no está firmado correctamente y muchos dispositivos impiden su instalación por seguridad.

Para generar un APK listo para enviar a testers o publicar, debe realizarse un build en modo release y estar correctamente firmado digitalmente. A continuación, se detalla el proceso:

La generación del APK final no se hace simplemente ejecutando la app en un dispositivo conectado (como suele hacerse en fase de desarrollo). Para obtener un APK **firmado y distribuible**, se debe seguir el proceso de "build release" en Android Studio. A continuación, se explica cómo hacerlo:

1. En Android Studio, ve al menú superior: **Build > Generate Signed Bundle / APK…**
2. Selecciona **APK** y haz clic en **Next**.
3. En la siguiente pantalla:
4. En **Key store path**, selecciona un archivo .jks (Java Key Store). Si no tienes uno, puedes crearlo ahí mismo con una contraseña.
5. Introduce el **Key alias** y las **contraseñas** correspondientes.

6. Asegúrate de seleccionar **Build type: release**.
7. Pulsa **Finish**. Android Studio comenzará a generar el archivo .apk.
8. Cuando termine, encontrarás el archivo en la ruta: **App-TrackMyRide/app/release/app-release.apk**

Este archivo se puede compartir por **correo electrónico, Google Drive, Telegram**, etc., y se puede instalar en cualquier móvil Android (teniendo activada la opción de "Instalación desde fuentes desconocidas").

9. Manual de instalación y uso

9.1 Introducción

TrackMyRide es una aplicación móvil diseñada para aquellos que desean registrar, gestionar y compartir sus rutas de manera sencilla y segura. La aplicación permite a los usuarios planificar sus trayectos, almacenar información detallada sobre cada recorrido, incluir imágenes relacionadas y hacer seguimiento de sus progresos y estadísticas personales. Además, la plataforma cuenta con una API REST segura que facilita la interacción entre la aplicación móvil y el backend, garantizando la integridad y disponibilidad de los datos.

Propósito del manual

El presente manual tiene como objetivo proporcionar una guía completa para los usuarios de TrackMyRide, facilitando la comprensión y el uso correcto de la aplicación. En este documento, los usuarios encontrarán desde los requisitos previos para su instalación, hasta tutoriales y explicaciones detalladas sobre las funcionalidades básicas y avanzadas de la aplicación. Este manual pretende ser una referencia útil tanto para nuevos usuarios que se inician en la aplicación como para usuarios experimentados que desean sacar el máximo provecho de todas las herramientas disponibles.

Requisitos previos (conocimientos o herramientas necesarias)

Para utilizar TrackMyRide, los usuarios deberán contar con un dispositivo móvil compatible con el sistema operativo Android, preferiblemente actualizado a versiones recientes para asegurar la mejor experiencia y compatibilidad. Es recomendable disponer de una conexión a internet estable para la sincronización de datos y la gestión de imágenes a través del sistema en la nube. Aunque la aplicación es intuitiva y accesible, es útil tener conocimientos básicos sobre el manejo de aplicaciones móviles, navegación por menús, y gestión de permisos en el dispositivo para optimizar el uso de todas las funcionalidades.

9.2 Descripción general

Funciones principales de la Aplicación

Las funciones principales incluyen:

- **Grabación de rutas mediante GPS**: el usuario puede iniciar el seguimiento de su ubicación con solo pulsar un botón. La aplicación registra el recorrido en tiempo real, permitiendo su posterior visualización y análisis.
- **Selección de tipo de vehículo**: antes de comenzar una ruta, el usuario puede elegir entre coche, moto o bicicleta, adaptando así los parámetros de análisis del trayecto.
- **Configuración personalizada de consumo**: en una sección dedicada, el usuario puede establecer datos como la eficiencia y consumo del vehículo seleccionado, con el objetivo de generar estadísticas detalladas y precisas por cada ruta.
- **Historial de rutas**: cada ruta realizada queda almacenada en un historial, indicando el nombre de la ruta, fecha y hora, tipo de vehículo, distancia, etc. En la versión gratuita, se pueden consultar hasta 5 rutas. Los usuarios premium tienen acceso completo a todo su historial.
- **Visualización del recorrido**: al acceder a una ruta guardada, el usuario puede ver el recorrido trazado en un mapa interactivo, facilitando su revisión o análisis posterior.
- **Gestión de puntos en el mapa**: posibilidad de añadir marcadores (pins) manualmente en la visualización del recorrido, por ejemplo, para destacar puntos de interés, paradas, incidencias, etc.
- **Edición de rutas**: permite modificar el título y la descripción de cada ruta una vez registrada.
- **Gestión de imágenes**: los usuarios pueden añadir imágenes desde la galería del dispositivo a cada ruta para personalizar o documentar mejor sus trayectos. Los usuarios gratuitos tienen un límite de imágenes por ruta, mientras que los usuarios premium pueden subir una mayor cantidad de imágenes por ruta.
- **Modo premium (simulado con PayPal Sandbox)**: mediante un sistema de pago actualmente simulado, los usuarios pueden activar una cuenta premium que desbloquea funcionalidades avanzadas:
    - Acceso completo al historial de rutas.
    - Subida ilimitada de imágenes por ruta.
    - Exportación y **compartición de rutas** con otros usuarios o plataformas.
- **Pantalla de perfil personalizable**: incluye la opción de cambiar la foto de perfil, nombre de usuario, cambio de contraseña y consultar los datos del usuario actual.

Beneficios para el usuario

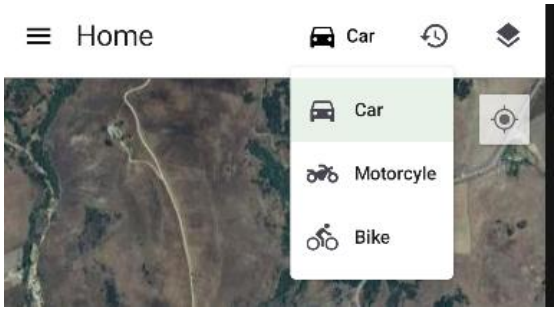
TrackMyRide proporciona un entorno cómodo, eficiente y seguro para cualquier usuario interesado en registrar y gestionar sus trayectos, ya sea con fines deportivos, logísticos o simplemente por ocio. Algunos de los principales beneficios incluyen:

- **Seguimiento preciso y automático** del recorrido mediante el GPS del dispositivo, sin intervención manual durante el trayecto.
- **Adaptación a distintos tipos de vehículo**, lo que permite a ciclistas, motoristas y conductores guardar estadísticas específicas y más realistas. Esto también da una mayor facilidad a la hora de buscar en el historial de rutas, pudiendo filtrar por vehiculos, para asi ver unicamente las rutas que te interesan.
- **Control sobre el consumo y eficiencia**, gracias a la introducción de parámetros personalizados para cada tipo de transporte.
- **Acceso visual e intuitivo a los recorridos**, con mapas interactivos que muestran con claridad la ruta realizada y los puntos relevantes que el propio usuario puede marcar en cualquier lugar del mapa.
- **Personalización completa de rutas**, con posibilidad de añadir descripciones, imágenes y marcadores.
- **Privacidad y seguridad**, ya que cada usuario solo tiene acceso a sus propios datos, protegidos mediante autenticación JWT.
- **Funcionalidades extra para usuarios premium**, como eliminar restricciones en la cantidad de rutas guardadas o imágenes subidas, y compartir recorridos de forma sencilla.

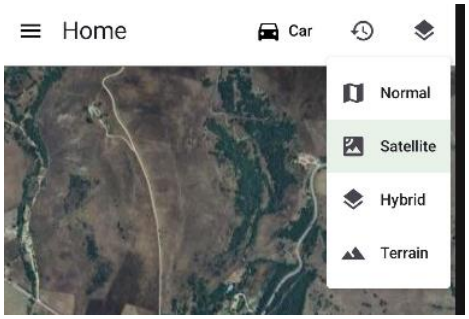
Breve descripción de la interfaz

La interfaz de usuario de TrackMyRide ha sido diseñada para ser intuitiva y fácil de usar incluso para personas sin experiencia técnica. A continuación, se describen brevemente las principales pantallas:

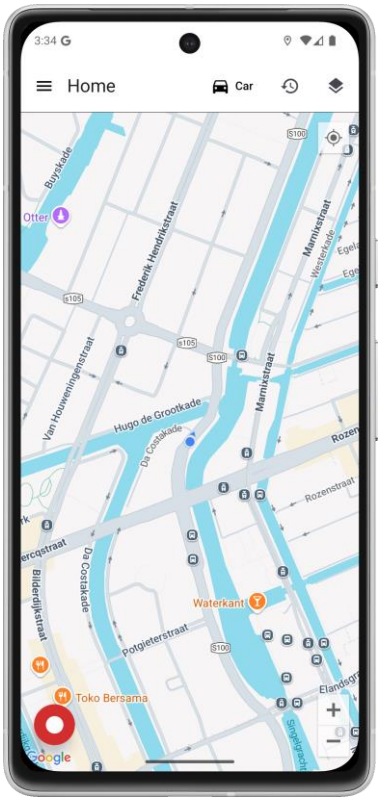
- **Pantalla principal de grabación**: incluye un botón central para iniciar o detener la grabación del recorrido, y una opción para seleccionar el tipo de vehículo. Tambien contiene un menu desplegable para elegir que tipo de capa deseas ver en el mapa. Al mismo tiempo, se ha añadido un icono para acceder al historial de rutas de una manera mas rápida e intuitiva, sin tener que abrir el menú hamburguesa lateral para ello.



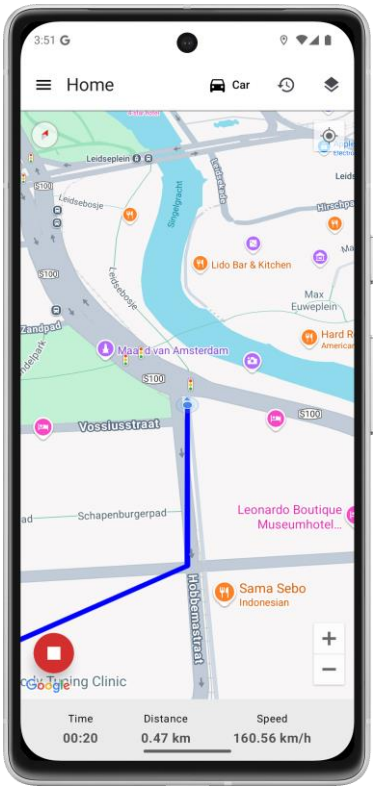
Img 37.selector vehículo



Img 36.selector capa mapa

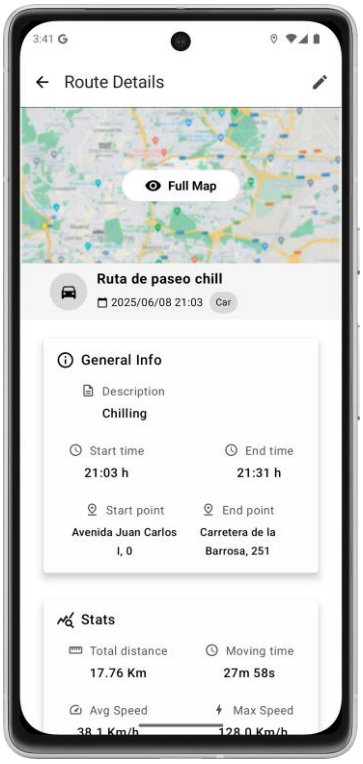


Img 39.Home

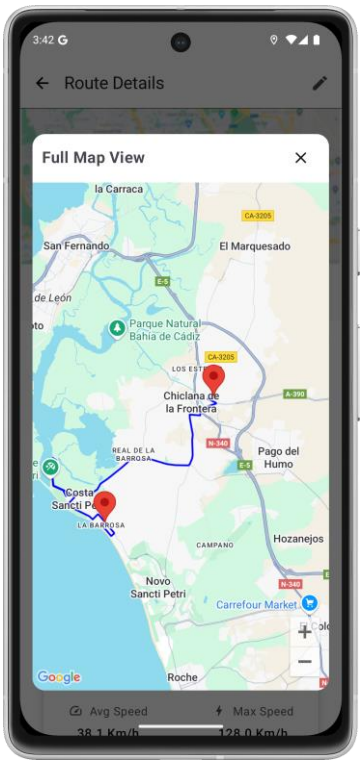


Img 38.Home grabando

- **Detalles ruta**: muestra todos los datos de la ruta en cuestión, nombre, descripción, estadísticas, lugar de inicio y fin, fechas y hora, vehículo, etc. Además, muestra un mapa interactivo con el recorrido trazado, e incluye botones para editar información, añadir imágenes o marcar puntos en el mapa.

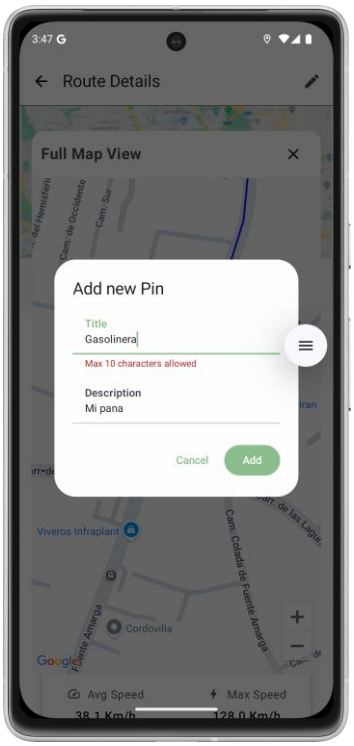


Img 41.Detalle ruta

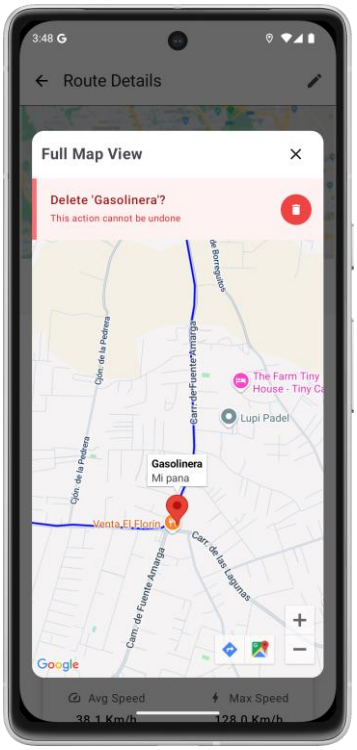


Img 40.Detalle ruta mapa

Como vemos, dándole al botón de Full Map se abre este dialogo, con el mapa completo, en el cual si dejamos presionado en cualquier lugar, podemos crear un pin:

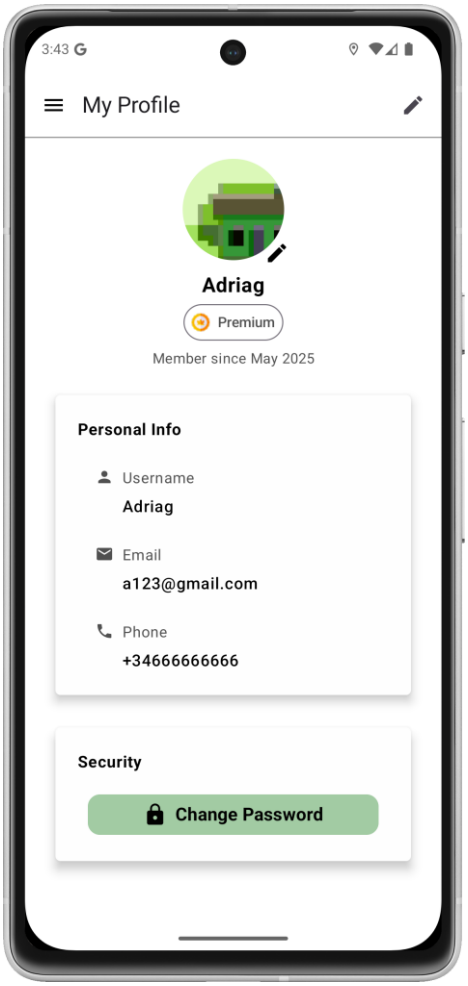


Img 43.Dialogo Pin

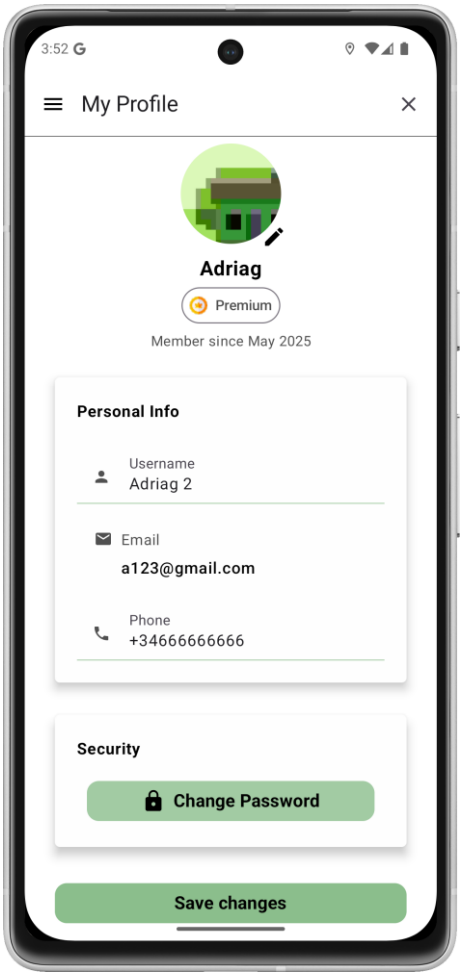


Img 42.Dialogo mapa con pin

- **Pantalla de perfil**: donde el usuario puede ver sus datos personales y modificarlos. Tanto imagen de perfil, como usuario, contraseña y teléfono.



Img 45.Perfil

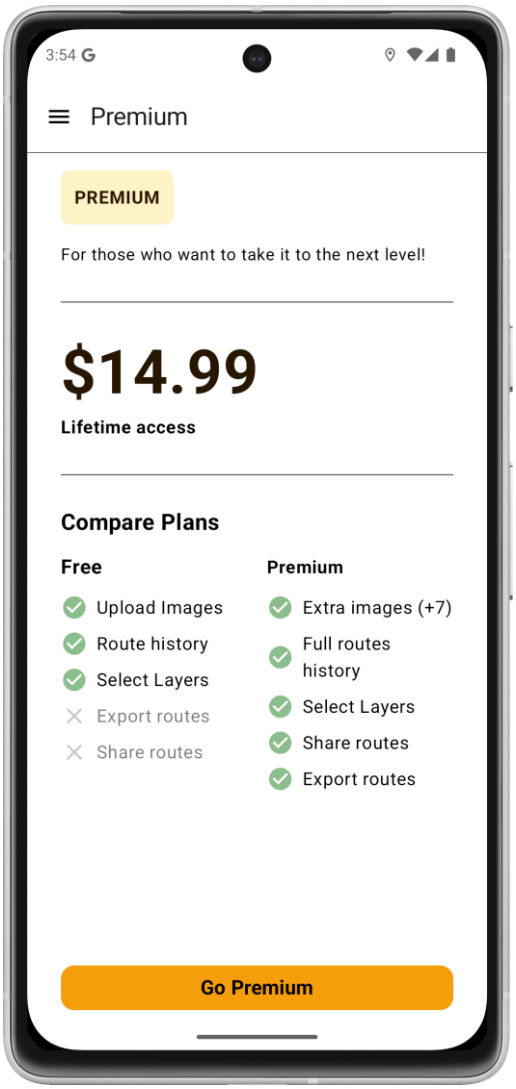


Img 44.Perfil editando

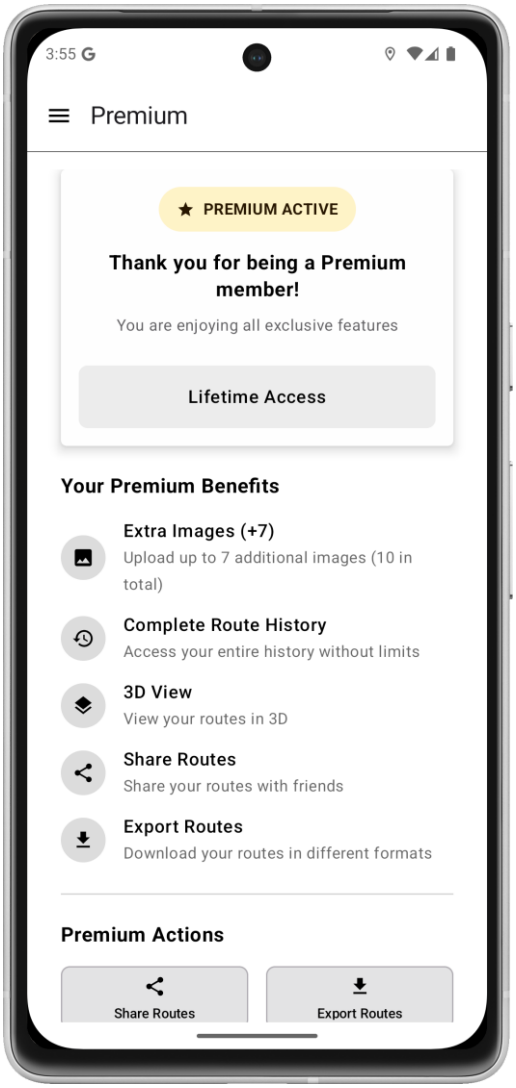
Como vemos si le damos al icono del lápiz, se activan los campos editables y el botón para guardar cambios.

- **Pantalla de premium**: en esta sección se dan a conocer al usuario las mejoras que obtiene si decide pagar y ser premium. Al pulsar el botón, te lleva a la pantalla de pago de prueba que usa paypal sandbox para simularla.

  Si ya eres premium, te muestra tus ventajas:

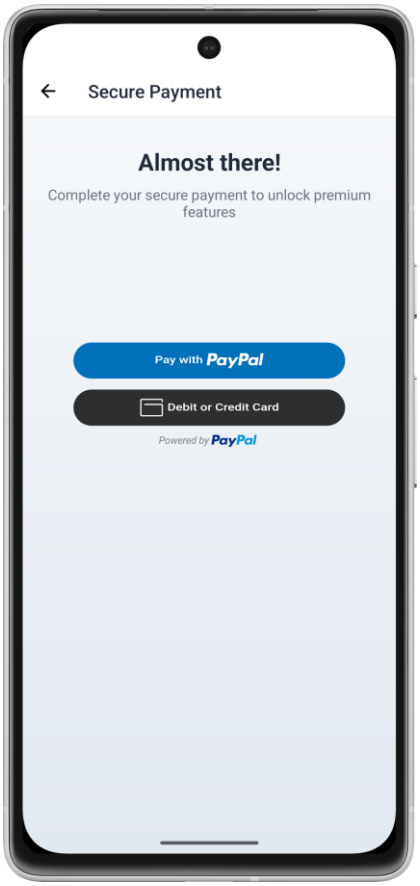


Img 46.No premium



Img 47.Premium

- **Pantalla de pago (sandbox)**: simula el proceso de pago con PayPal para activar el modo premium. Cuando se ponen los datos de la tarjeta de prueba y se valida el pago, vuelve a la pantalla de premium, la cual cambia indicando que ya es premium.



Img 48.Pago paypal

- Pantalla de **About Us:** en esta pantalla puedes encontrar información relevante sobre el desarrollo de esta aplicación, términos, contacto etc.

Cada sección está claramente diferenciada mediante iconos, menús y títulos visibles, manteniendo una navegación fluida y accesible desde cualquier pantalla.

9.3 Requisitos del sistema

Para garantizar un correcto funcionamiento de la aplicación **TrackMyRide**, es importante tener en cuenta una serie de requisitos tanto a nivel de hardware como de software. Esta sección proporciona una guía detallada sobre los recursos mínimos y recomendados necesarios para ejecutar la aplicación de forma fluida, así como otros aspectos que influyen en la experiencia de uso, como la conectividad o la disponibilidad de sensores específicos del dispositivo.

Requisitos de Hardware

Aunque **TrackMyRide** está diseñada para funcionar en una amplia gama de dispositivos Android, es recomendable contar con un equipo que cumpla con ciertos criterios mínimos para asegurar una experiencia de usuario óptima:

Requisitos mínimos:

- **Procesador**: CPU de 4 núcleos (Quad-Core) a 1.5 GHz o superior. La aplicación realiza operaciones en segundo plano, como el registro GPS y procesamiento de datos, por lo que un procesador básico pero moderno es esencial.
- **Memoria RAM**: 2 GB de RAM. Esto permite ejecutar la app de forma estable y sin cierres inesperados durante el seguimiento de rutas largas.
- **Almacenamiento interno**: Al menos 200 **MB** de espacio libre para instalar la app y guardar datos temporales como imágenes, rutas o configuraciones locales. Si el usuario añade muchas imágenes o realiza rutas frecuentes, es recomendable contar con al menos **1 GB disponible** para evitar restricciones.
- **Pantalla**: Resolución mínima de **720x1280 (HD)**. Esto asegura que todos los elementos de la interfaz se visualicen correctamente sin recortes o sobreposiciones.
- **GPS**: Módulo de localización obligatorio. La funcionalidad principal de la aplicación depende del acceso a la ubicación en tiempo real. Se recomienda un dispositivo con **soporte para GPS lo mas moderno posible.**
- **Conectividad**: Conexión Wi-Fi o datos móviles. Las funciones principales de la App requieren de conexión para enviar datos al servidor, como subir imágenes, sincronización, autenticación, visualización de mapas o activación de premium.

Requisitos de Software

Para poder instalar y utilizar **TrackMyRide** correctamente, el dispositivo debe contar con un sistema operativo Android actualizado y con ciertos servicios activos

Sistema operativo:

- **Versión mínima compatible**: Android 7**.0**. Esta versión garantiza la compatibilidad con librerías modernas utilizadas para acceso a ubicación, gestión de permisos, y conexión con servidor.
- **Versión recomendada**: Android **10 o superior**. A partir de esta versión, se mejoran aspectos como el manejo en segundo plano del GPS, ahorro energético, y mayor seguridad para la gestión de tokens de autenticación.

Servicios necesarios:

- **Servicios de Google Play**: deben estar actualizados para el correcto funcionamiento del sistema de localización, autenticación y acceso al almacenamiento en la nube.
- **Permisos requeridos por la app**:
    - Acceso a la ubicación precisa (durante el tracking).
    - Acceso a almacenamiento multimedia (para cargar imágenes).
    - Acceso a internet (para todas las funciones conectadas al servidor).

Requisitos de Conectividad y Otros Componentes

Las características de esta App dependen de una conexión estable y activa, por lo que se recomienda:

Conexión a internet:

- **Necesaria para**:
    - Autenticación de usuarios.
    - Envío y sincronización de rutas con el servidor.
    - Visualización del historial de rutas remoto.
    - Subida y eliminación de imágenes en **servicios de terceros (Cloudinary)**.
    - Activación y validación del estado **Premium** mediante simulación de pago con **PayPal Sandbox**.
    - Acceso a mapas dinámicos y previsualización de rutas.
- **Velocidad recomendada**:
    - Conexión **3G o superior** (WiFi o 4G/5G preferibles para subida de imágenes).

Compatibilidad y otros entornos

- **Dispositivos compatibles**:
    - Smartphones Android con pantalla táctil.
    - Algunas **tablets Android** también pueden ejecutar la app, aunque la experiencia está optimizada para móviles.
- **No compatible con**:
    - Dispositivos iOS (iPhone, iPad).
    - Emuladores sin servicios de localización activos.
    - Dispositivos Android sin soporte GPS.

Recomendaciones adicionales

- Mantener el dispositivo **actualizado** con los últimos parches de seguridad de Android.
- Evitar cerrar la app o bloquear excesivamente el sistema durante el tracking para no interrumpir el registro.
- Activar el **modo de alta precisión** del GPS en las configuraciones del dispositivo.

9.4 Instalación y configuración inicial

La correcta instalación de **TrackMyRide** es el primer paso fundamental para comenzar a disfrutar de todas sus funcionalidades. Este apartado describe, con todo detalle, el proceso necesario para instalar la aplicación en un dispositivo Android, incluyendo la descarga de la APK, los pasos previos, la gestión de permisos y el registro inicial obligatorio.

Aunque en futuras versiones la aplicación podría distribuirse a través de plataformas oficiales como Google Play, en esta etapa del desarrollo se proporciona el archivo APK directamente al usuario, lo que implica una serie de pasos manuales que es importante seguir cuidadosamente.

Descarga de la aplicación

Actualmente, la aplicación **TrackMyRide** se distribuye en formato **APK**. Un archivo APK es el paquete de instalación estándar para aplicaciones Android, y puede instalarse manualmente en cualquier dispositivo compatible que tenga habilitada la opción de **instalación desde fuentes desconocidas**.

Pasos para instalar la aplicación

Una vez tengas el archivo APK descargado en tu móvil, sigue estos pasos para instalar correctamente **TrackMyRide**:

1. **Habilita la instalación desde fuentes desconocidas** Android no permite por defecto instalar aplicaciones que no vienen de Google Play, así que debes activar esta opción:
    - Ve a **Configuración > Seguridad** (o “Privacidad”, según tu modelo de móvil).
    - Busca la opción **“Fuentes desconocidas”** o **“Instalación de apps desconocidas”**.
    - Habilita el permiso para instalar desde el navegador o la aplicación de archivos que hayas usado.
2. **Instala la APK**
    - Una vez descargada la APK, abre el archivo desde las notificaciones o usando el gestor de archivos de tu móvil.
    - El sistema te preguntará si deseas instalar la aplicación. Acepta y espera unos segundos mientras se completa la instalación.
3. **Accede a la aplicación**
    - Cuando la instalación termine, verás el icono de TrackMyRide en tu menú de aplicaciones. Pulsa sobre él para abrir la app.

Registro y acceso

Para poder utilizar **TrackMyRide**, es necesario que te registres con una cuenta. Es un proceso muy sencillo que solo requiere unos pocos datos.

¿Cómo registrarse?

1. Al abrir la aplicación por primera vez, verás una pantalla de inicio de sesión.
2. Si no tienes cuenta todavía, pulsa en **"Sign Up"**.
3. Completa el formulario con los siguientes datos:
    - Nombre de usuario.
    - Correo electrónico.
    - Número de telefono (Opcional)
    - Contraseña segura (elige una que solo tú conozcas).
4. Pulsa en **"Register"**.

¡Listo! Una vez registrado, si todos los datos introducidos son válidos, accederás directamente a la aplicación y podrás comenzar a usar todas sus funciones.

En caso de ya poseer una cuenta, simplemente inicia sesión con tus datos.

Primera configuración recomendada

Aunque la aplicación ya está lista para usarse tras el inicio de sesión, hay algunas configuraciones que te recomendamos revisar para personalizar tu experiencia y sacarle el máximo provecho:

- **Foto de perfil**: En la sección de perfil puedes cambiar tu imagen de usuario.
- **Tipo de vehículo**: Puedes añadir datos sobre tus vehículos, como el consumo, eficiencia, cantidad de combustible, etc.
- **Permisos**: La aplicación te pedirá acceso a ciertas funciones del teléfono, como:
    - **Ubicación** (para grabar tus rutas). Esta te la pedirá expresamente una vez hagas **Login,** ya que sin esto, la aplicación no tiene utilidad.
    - **Almacenamiento/Galería** (para subir imágenes). Asegúrate de aceptar estos permisos, cuando se soliciten, para poder utilizar las funcionalidades requeridas correctamente.

Con estos pasos, tendrás **TrackMyRide** completamente instalada y lista para usar. Desde aquí, podrás comenzar a grabar tus rutas, consultar tu historial y explorar todas las funcionalidades, tanto en su versión gratuita como premium.

¿Nada de lo anterior te ha funcionado?

Diríjase aquí para obtener más información

9.5 Navegación y funcionalidades básicas

TrackMyRide ha sido diseñada pensando en ofrecer una experiencia de usuario intuitiva, clara y visualmente amigable. A lo largo de esta sección aprenderás cómo moverte dentro de la aplicación, qué hace cada botón o sección principal, y cómo empezar a utilizar las funciones esenciales paso a paso. No necesitas conocimientos técnicos ni experiencia previa en Apps similares: con esta guía estarás listo para sacarle el máximo partido desde el primer uso.

Pantalla	Descripción
Home	Es la pantalla principal de la App. Desde aquí puedes ver el mapa, cambiar la capa, elegir que vehículo vas a usar para grabar y comenzar a grabar tus rutas. Es el centro de actividad.
My Vehicles	En esta sección puedes introducir los datos técnicos de los vehículos que utilizas: coche, moto o bicicleta. Estos datos son importantes para que el sistema calcule correctamente las estadísticas energéticas de tus rutas. Aún así, no son obligatorios para usar la aplicación y su core principal.
Routes History	Aquí puedes consultar el listado de rutas que ha realizado, pudiendo filtrar por vehículo, eliminar, y ver detalles básicos.
Route Details	En esta pantalla se muestran los detalles de las rutas, que incluyen información detallada como distancia, duración, modo de transporte, imágenes asociadas, descripción, y estadísticas (si el vehículo está configurado).
Profile	Sección donde puedes gestionar tu cuenta personal, cambiar tu foto de perfil y ver tu tipo de cuenta (gratuita o premium). También podrás ver tus datos básicos registrados, cambiar tu contraseña.
Premium	Desde aquí puedes convertirte en usuario premium, haciendo el pago de la suscripción mediante **paypa**l. Si decides hacerlo, en esta pantalla pasarás a poder ver tus mejoras obtenidas para que las consultes en cualquier momento.

About Us	En esta sección podrás obtener un poco más de contexto de la App, tanto de su desarrollador, como su origen. También encontraras información relevante como políticas varias, y contacto directo.

Img 49.Tabla Navegación Pantallas

Los botones e iconos de la aplicación están diseñados con un estilo moderno, sencillo y con colores reconocibles: verde para iniciar, rojo para detener, y tonos neutros para acciones secundarias. Esto hace que sea fácil identificar qué hacer en cada momento.

Tutorial básico: cómo comenzar a usar TrackMyRide

A continuación, se detalla el recorrido que se recomienda seguir para empezar a utilizar la aplicación de forma correcta y aprovechar todas sus funcionalidades desde el principio.

Paso 1: Registro de usuario

Cuando abras la aplicación por primera vez, lo primero que verás será la pantalla de **login**. Si no posees una cuenta, podrás navegar a el apartado de registro para crear una. Deberás rellenar un pequeño formulario con los siguientes datos:

- Nombre de usuario
- Correo electrónico
- Número de teléfono (Opcional)
- Contraseña segura

Una vez registrado, se creará tu cuenta automáticamente y se iniciará sesión en ella. Este proceso es muy rápido y no requiere validación externa, por lo que en pocos segundos ya estarás dentro.

Paso 2: Permisos necesarios

Una vez hayas accedido a la app por primera vez, se te pedirá aceptar los **permisos de ubicación**. Este permiso es fundamental para que la aplicación pueda seguir tus rutas a través del GPS del dispositivo.

Importante: La app solo utiliza la ubicación mientras estás grabando una ruta. No registra tu posición en segundo plano fuera de ese contexto.

Asegúrate de aceptar los permisos cuando se te soliciten. Si no los aceptas, no podrás hacer uso de la App.

Paso 3: Configuración de vehículos

Aunque no es obligatorio, **se recomienda encarecidamente** que el siguiente paso sea acceder a la sección de **Vehículos** y añadir los datos de al menos un medio de transporte (coche, moto o bici).

Aquí puedes introducir:

- Tipo de vehículo
- Consumo energético o eficiencia
- Nombre o alias del vehículo
- Resto de información de interés

¿Por qué es importante esto? Porque si introduces estos datos, la aplicación podrá calcular estadísticas detalladas al final de cada ruta, como:

- Consumo total estimado
- Coste energético
- Eficiencia por kilómetro

Si no configuras ningún vehículo, podrás igualmente grabar rutas, pero **las estadísticas estarán limitadas**, y no obtendrás todos los datos que la aplicación puede ofrecerte.

Paso 4: ¡Listo para grabar rutas!

Una vez creada tu cuenta, concedidos los permisos y añadidos tus vehículos (si lo deseas), ya puedes empezar a usar la funcionalidad principal: grabar rutas.

Desde la pantalla de **Inicio**, pulsa el botón **"Iniciar ruta"** y la aplicación comenzará a seguir tu trayecto. Podrás detener o guardar la ruta cuando termines. Los datos quedarán almacenados y accesibles desde el historial, desde el cual podrás modificar tanto nombre, descripción, ver la ruta, añadir pines, y añadir imágenes que quedarán en el recuerdo de la ruta.

9.6 Funciones avanzadas

Además de las funciones básicas que permiten grabar y consultar rutas, **TrackMyRide** ofrece una serie de funcionalidades avanzadas pensadas para usuarios que desean personalizar su experiencia, gestionar sus rutas de manera más eficiente y sacar el máximo partido a la aplicación. Algunas de estas funciones están disponibles solo para usuarios con cuenta **premium**, lo que permite mantener una app optimizada y sostenible, mientras se ofrece valor añadido a quienes necesitan más potencia y flexibilidad.

A continuación, se describen en detalle estas funciones avanzadas y cómo usarlas correctamente.

Gestión avanzada del historial de rutas

TrackMyRide guarda automáticamente cada ruta que grabas. Aunque cualquier usuario puede acceder a su historial, los usuarios gratuitos tienen un límite de **5 rutas visibles** en el historial. Esto permite a todos disfrutar de la funcionalidad sin saturar el dispositivo.

Sin embargo, si eres usuario **premium**, podrás:

- Consultar **todo tu historial sin límite de rutas**.
- Ver estadísticas completas de rutas antiguas, sin restricciones.
- Mantener un registro cronológico completo de tus desplazamientos.

Recomendación: Si usas la App con frecuencia o para fines profesionales, se recomienda activar el modo premium para conservar todas tus rutas sin perder información.

Añadir imágenes a las rutas

Una de las funciones más útiles y atractivas es la posibilidad de añadir imágenes a tus rutas desde la galería del dispositivo. Esto te permite documentar momentos clave del trayecto, paisajes, incidencias, puntos de interés, etc.

Usuarios gratuitos pueden añadir hasta 3 imágenes por ruta.

Usuarios premium, aumenta este límite a 10, cantidad más que suficiente por ruta.

Cómo hacerlo:

1. Ve al historial.
2. Selecciona una ruta.
3. Pulsa el botón “+ Imagen”.
4. Elige la imagen desde tu galería.
5. Esta se subirá y quedará vinculada a la ruta.

Las imágenes se almacenan de forma segura en la nube, por lo que no ocupan espacio adicional en tu móvil.

Edición completa de rutas

Después de grabar una ruta, puedes personalizarla completamente para dejar un registro más detallado. Desde la pantalla de detalle de cada ruta, podrás:

- Cambiar el **título** de la ruta.
- Añadir o modificar la **descripción**.
- **Insertar pins o marcadores** en puntos específicos del mapa para señalar ubicaciones relevantes (lugares de descanso, problemas, miradores, etc.).
- Eliminar imágenes antiguas o no deseadas.

Este nivel de edición te permite transformar cada ruta en una pequeña bitácora personal, ideal para quienes usan la app para viajar, explorar o documentar desplazamientos importantes.

Exportar y compartir rutas (solo Premium)

Para los usuarios que desean guardar sus rutas en otros formatos o compartirlas con otras personas, **TrackMyRide Premium** ofrece la función de **exportación de rutas**.

Esto incluye:

- Descargar la ruta en formato estándar (.gpx).
- Enviar el recorrido a través de redes sociales, correo o aplicaciones de mensajería.

Esta función es especialmente útil para:

- Ciclistas que desean compartir sus rutas con otros.
- Conductores que necesitan registrar trayectos profesionales.
- Usuarios que deseen guardar un backup o visualizar la ruta en otros programas de mapas.
- Moteros que quieran compartir su ruta de aventura con amigos para que las hagan.

Personalización del perfil

Todos los usuarios, independientemente de su tipo de cuenta, pueden **editar su perfil**. Esto incluye:

- Cambiar la **foto de perfil** seleccionando una imagen desde la galería del dispositivo.
- Ver el correo con el que están registrados.
- Saber si su cuenta es **gratuita o premium** (el estado aparece reflejado en el perfil).

Un perfil bien configurado no solo hace la experiencia más agradable, sino que permite identificar visualmente al usuario.

Simulación de pago con PayPal Sandbox

Para activar las funcionalidades premium, **TrackMyRide** permite realizar un pago simulado a través del sistema **PayPal Sandbox**. Este sistema actualmente es solo una simulación, pensada para mostrar distintas funcionalidades que dependen del tipo de suscripción.

Esto significa que **no se realiza ningún cargo real** ni se utilizan cuentas bancarias reales. El sistema actúa como si el pago fuera real, permitiendo activar el modo premium de forma segura durante la fase de evaluación.

Una vez activado el modo premium, el usuario accede a todas las funciones avanzadas inmediatamente, sin necesidad de reiniciar la app.

Estadísticas post-rutas

Una de las características de TrackMyRide es su capacidad para calcular **estadísticas personalizadas** en base al vehículo que utilices y los datos que hayas introducido.

Después de cada ruta, si has configurado correctamente tu vehículo (tipo, consumo, eficiencia), la app podrá mostrarte datos como el consumo en esa ruta, el ritmo que has tenido, velocidad máxima y mínima entre otros.

Esto ofrece una mayor experiencia al usuario, que puede ver todos estos datos sin necesidad de estar pendiente durante la propia ruta.

Casos de uso recomendados

A continuación, algunos ejemplos reales donde estas funciones avanzadas pueden marcar la diferencia:

- Un **motero**, que sale a la aventura por las carreteras sin un rumbo en específico, y quiere más tarde recordar por donde ha ido, para poder regresar o compartirla con otros moteros.
- A la hora de hacer prácticas del **carné de conducir**, ya que puedes ponerlo a grabar, y cuando acabes un día de prácticas, puedes volver a realizar el recorrido de copiloto o con otro vehículo que puedas conducir, facilitándote el aprendizaje de la zona para mejorar.
- **Un ciclista profesional** que quiere registrar y comparar sus rutas de entrenamiento diarias.
- **Un repartidor autónomo** que necesita un historial completo de rutas para comprobar recorridos y estimar gastos.
- **Una pareja de viajeros** que usa la App para documentar sus rutas por carretera con imágenes y descripciones.
- **Un usuario ecológico** que desea calcular su impacto energético para tomar decisiones más sostenibles.

9.7 Resolución de Problemas (FAQ)

En esta sección se abordan las dudas, errores y situaciones más comunes que los usuarios pueden encontrar al utilizar **TrackMyRide**, así como sus soluciones recomendadas. La finalidad es que cualquier persona, independientemente de su nivel de experiencia, pueda resolver los problemas de manera rápida, sin necesidad de conocimientos técnicos.

Si encuentras un problema que no está listado aquí, puedes acudir a la sección de contacto y soporte al final del manual.

Pregunta / Problema	Solución sugerida
No puedo iniciar sesión después de registrarme.	Asegúrate de haber introducido correctamente tu correo electrónico y contraseña. Revisa si tienes conexión a internet. Si el problema persiste, intenta reiniciar la app.
Olvidé mi contraseña.	En la versión actual se ha habilitado una pantalla específica para recuperar la contraseña intriduciendo tu correo. Contacta con soporte a través del correo, si aún sigues teniendo dificultades.
La aplicación no graba mi ubicación al iniciar una ruta.	Verifica que hayas concedido los permisos de ubicación cuando se solicitaron. También asegúrate de que el GPS del dispositivo esté activado. Reinicia el móvil si es necesario.
No puedo subir imágenes desde la galería.	Comprueba que hayas concedido permisos de acceso a archivos/fotos. También verifica que el formato de imagen sea compatible (preferiblemente .jpg o .png).
La app se cierra inesperadamente.	Asegúrate de tener instalada la última versión de la APK. Reinicia el dispositivo y vuelve a intentarlo. Si el error persiste, contacta con soporte.
No puedo ver mis rutas antiguas en el historial.	Si eres usuario gratuito, solo podrás visualizar las 5 rutas más recientes. Para ver todas las rutas, debes activar el modo premium desde la sección correspondiente.
He pagado con PayPal Sandbox pero no se activó el premium.	A veces la simulación puede tardar unos segundos. Reinicia la app y verifica en tu perfil si aparece el estado “Usuario premium”. Si no, contacta al soporte.
No puedo compartir mi ruta con otros.	Verifica que estés utilizando una cuenta premium. Solo los usuarios premium tienen acceso a la funcionalidad de exportación y compartición de rutas.

El botón de iniciar ruta no funciona.	Asegúrate de tener conexión GPS y datos móviles activos. También verifica que no haya una ruta en curso ya activa.
Mis estadísticas aparecen vacías o incompletas.	Esto ocurre si no se ha configurado previamente un vehículo con datos de consumo y eficiencia. Ve a la sección "Vehículos" y añade esta información.
No me deja añadir más imágenes a una ruta.	Los usuarios gratuitos solo pueden añadir 3 imágenes por ruta. Si necesitas más capacidad, activa el modo premium.

Img 50.Tabla FAQ

Consejos generales para resolver problemas

Si experimentas cualquier tipo de error con la aplicación, estos pasos generales pueden ayudarte a resolverlo antes de contactar al soporte técnico:

1. **Reinicia la aplicación**: Ciérrala completamente desde el administrador de tareas del dispositivo y vuelve a abrirla.
2. **Verifica los permisos**: La app necesita permisos de ubicación y acceso a archivos para funcionar correctamente. Puedes gestionarlos desde la configuración del sistema.
3. **Comprueba tu conexión a internet**: La mayoría de funciones requieren conexión activa, especialmente el inicio de sesión, la subida de imágenes y el acceso al historial.
4. **Reinicia tu teléfono**: Esto puede resolver conflictos temporales con sensores como el GPS o errores de red.
5. **Reinstala la aplicación**: Si nada más funciona, desinstala la APK y vuelve a instalarla. Asegúrate de tener una copia actual de la ruta de instalación (Dropbox, Google Drive, etc.).
6. **Consulta las actualizaciones**: Puede que estés usando una versión antigua. Siempre descarga la versión más actualizada desde el enlace proporcionado.

Si después de seguir estas soluciones sigues teniendo problemas con la instalación o el acceso a **TrackMyRide**, puedes ponerte en contacto directamente con el desarrollador para recibir ayuda personalizada.

Correo de soporte técnico: trackmyride.support@gmail.com

Explica brevemente cuál es el problema, qué dispositivo estás utilizando (marca, modelo y versión de Android) y, si es posible, incluye una captura de pantalla del error que estás viendo. El equipo de soporte te responderá lo antes posible para ayudarte a resolverlo.

9.8 Mantenimiento y actualizaciones

Mantener tu aplicación **TrackMyRide** actualizada y en buenas condiciones es esencial para disfrutar de una experiencia fluida, segura y completa. En este apartado te explicaremos cómo gestionar las actualizaciones, qué aspectos tener en cuenta para un rendimiento óptimo y algunos consejos básicos de seguridad para proteger tus datos y tu cuenta.

¿Por qué es importante actualizar TrackMyRide?

Las actualizaciones son una parte fundamental del ciclo de vida de cualquier aplicación. En el caso de TrackMyRide, las nuevas versiones pueden incluir:

- Corrección de errores o fallos detectados en versiones anteriores.
- Mejoras en la precisión del tracking de rutas mediante GPS.
- Optimización del rendimiento general de la app.
- Nuevas funcionalidades o ampliación de las existentes (por ejemplo, mejoras en el historial, estadísticas más detalladas, nuevas opciones premium, etc.).
- Aumento de la seguridad y protección de los datos del usuario.
- Compatibilidad con nuevas versiones del sistema operativo Android.

Por eso, siempre que haya una nueva versión disponible, es recomendable actualizarla cuanto antes.

¿Cómo actualizar la aplicación?

Actualmente, **TrackMyRide** se distribuye en formato APK, por lo que las actualizaciones no se gestionan automáticamente desde la Play Store. Aquí tienes los pasos para actualizarla correctamente:

1. **Consulta el canal de descarga**: Habitualmente, el desarrollador proporcionará un enlace a la nueva versión, que puede estar en una plataforma como Dropbox, Google Drive o similar. Este enlace será actualizado cada vez que haya una nueva versión disponible.
2. **Descarga la nueva APK**: Desde el enlace proporcionado, descarga la nueva versión del archivo APK en tu dispositivo móvil.
3. **Desinstala la versión anterior (opcional)**: Aunque no siempre es necesario, se recomienda desinstalar la versión antigua para evitar conflictos, a menos que el desarrollador indique lo contrario.
4. **Instala la nueva versión**:
    - Localiza el archivo APK en tu carpeta de descargas.
    - Haz clic sobre él para iniciar la instalación.
    - Acepta los permisos que te solicite el sistema.

    - Si es la primera vez que instalas una APK manualmente, puede que debas habilitar la opción de "Instalar desde fuentes desconocidas" en los ajustes de tu teléfono, tal y como se detalla en este apartado de Instalación y configuración inicial

5. **Inicia la aplicación** y comprueba que todo funcione correctamente. Tus rutas e información no se perderán, ya que están vinculadas a tu cuenta de usuario.

Nota: Asegúrate siempre de que el enlace de descarga sea confiable y oficial para evitar instalar versiones no autorizadas o manipuladas.

Recomendaciones para optimizar el rendimiento

Para que la aplicación funcione de forma estable, rápida y sin errores, puedes seguir estos consejos generales:

- **Cierra la app cuando no la estés usando**: Esto ahorra batería y recursos del sistema.
- **Evita tener muchas aplicaciones abiertas** en segundo plano mientras grabas una ruta, especialmente si tu dispositivo es de gama media o baja.
- **Activa el GPS solo cuando vayas a grabar una ruta**, así conservarás batería.
- **Concede los permisos necesarios** (ubicación, almacenamiento) para evitar bloqueos o mal funcionamiento.
- **Revisa la configuración de ahorro de energía** de tu móvil: algunas marcas limitan el acceso al GPS en segundo plano para ahorrar batería, lo cual puede afectar el seguimiento de la ruta.

Consejos de seguridad

Tu cuenta y tus datos personales son importantes. Sigue estas recomendaciones para mantenerlos protegidos:

- **No compartas tus credenciales** con nadie.
- **Cambia tu contraseña regularmente**.
- **Evita conectarte a redes Wi-Fi públicas** al iniciar sesión o al compartir rutas.
- **Verifica siempre que estás descargando la app desde el canal oficial de distribución**.
- **Si pierdes acceso a tu cuenta**, contacta cuanto antes con el soporte técnico para tomar medidas.

¿Cómo saber si hay una nueva versión disponible?

El desarrollador puede informarte sobre nuevas versiones a través de:

- Una **notificación** dentro de la App.
- Un **correo electrónico**.
- El mismo enlace de descarga, donde se reemplazará el archivo APK por la última versión.
- Comunicaciones por redes sociales o canales de contacto habilitados.

Te recomendamos revisar periódicamente el canal de descarga para asegurarte de tener siempre la última versión.

9.9 Soporte y Contacto

Aunque **TrackMyRide** ha sido diseñada para ser intuitiva, sencilla de usar y fiable en su funcionamiento, entendemos que pueden surgir dudas, inconvenientes o incluso sugerencias por parte de los usuarios. Por eso, contar con un buen canal de soporte es fundamental para asegurar una experiencia completa y satisfactoria.

En este apartado te explicamos todas las vías disponibles para contactar con el desarrollador, buscar ayuda o acceder a recursos complementarios que pueden resolver tus dudas o problemas rápidamente.

Correo electrónico de contacto

La forma principal de soporte es a través del **correo electrónico oficial del desarrollador**. Si tienes alguna duda, sugerencia, problema técnico o necesitas asistencia personalizada, puedes enviar un mensaje a la siguiente dirección:

trackmyride.support@gmail.com

Al escribir tu correo, intenta incluir los siguientes datos para que el soporte pueda ayudarte lo más rápido posible:

- Nombre y versión de la aplicación (por ejemplo, "TrackMyRide v1.0").
- Marca y modelo de tu dispositivo móvil.
- Versión de Android instalada.
- Breve descripción del problema o consulta.
- Capturas de pantalla si es necesario.

Nota: Cuanta más información proporciones, más fácil será diagnosticar y solucionar el problema.

Horario de atención

Aunque el soporte no se realiza mediante un sistema en tiempo real, los correos son revisados con regularidad y recibirás una respuesta en el menor tiempo posible. El plazo habitual de respuesta es de **24 a 48 horas (días laborables)**.

Ten en cuenta que, durante fines de semana o festivos, la respuesta podría tardar un poco más.

Recursos adicionales

Además del soporte directo por correo electrónico, te recomendamos seguir estas otras opciones si necesitas ayuda o quieres aprender más sobre el uso de la aplicación:

- **Este manual de usuario**: Estás leyendo el documento más completo para resolver dudas sobre la instalación, el uso de funcionalidades básicas y avanzadas, configuración inicial, resolución de problemas frecuentes, etc.
- **Sección de preguntas frecuentes (FAQ)**: En el apartado anterior del manual encontrarás respuestas a las dudas más comunes de los usuarios.
- **Canales de actualización**: Si la aplicación en el futuro se distribuye a través de tiendas oficiales como Google Play, también se habilitará una sección de valoraciones y comentarios que podrá servir como punto de contacto e información.
- **Redes sociales o foros (si aplica en el futuro)**: Cualquier canal extra que se cree será anunciado a través de la app o el canal de distribución oficial.

¿Qué tipo de consultas puedes realizar?

Puedes contactar al soporte para cuestiones como:

- Problemas técnicos (error al iniciar sesión, fallos al grabar rutas, estadísticas que no se calculan correctamente, etc.).
- Dudas sobre el uso de alguna funcionalidad específica.
- Sugerencias para mejorar la aplicación.
- Informar de errores en el funcionamiento o en los datos mostrados.
- Consultas sobre el uso del modo Premium o el proceso de simulación de pago.
- Ayuda para recuperar tu cuenta si pierdes acceso.

¿Y si no recibo respuesta?

En el raro caso de que no recibas respuesta tras varios días, asegúrate de revisar tu carpeta de **spam o correo no deseado**, ya que en ocasiones los correos automáticos o respuestas pueden acabar allí por error. También puedes reenviar el mensaje original o contactar desde una cuenta diferente.

10. Conclusiones

10.1 Comparación del resultado con la idea inicial y mejoras futuras

El desarrollo de **TrackMyRide** ha supuesto no solo la culminación de un proyecto funcional y completo, sino también una evolución natural de una idea personal hacia una solución técnica mucho más madura, escalable y profesional. El objetivo inicial del proyecto era sencillo y muy concreto: contar con una herramienta fiable y libre de distracciones para registrar rutas en exteriores, principalmente en el contexto de salidas en moto. Este objetivo partía de una necesidad real detectada por el propio desarrollador: la falta de aplicaciones que ofrecieran una experiencia intuitiva, sin anuncios intrusivos y con un sistema de almacenamiento de rutas que permitiera recuperar recorridos pasados con precisión.

10.2 De una idea personal a una solución técnica escalable

Inicialmente, el enfoque iba a ser completamente local, utilizando bases de datos embebidas como SQLite para guardar los datos directamente en el dispositivo del usuario. Sin embargo, durante la fase de diseño y análisis, surgió la necesidad de mejorar aspectos relacionados con la seguridad, la sincronización entre dispositivos y el control de acceso a funcionalidades premium. Esto supuso un giro importante en la arquitectura del sistema, apostando por una solución mucho más robusta y segura: el desarrollo de una API propia con **Spring Boot**, autenticación basada en **Firebase** y gestión de sesiones mediante **JWT y refresh tokens**.

Este cambio fue crucial, ya que permitió no solo asegurar la integridad de los datos y la protección de las funciones avanzadas, sino también mantener una coherencia en la experiencia del usuario entre dispositivos, abriendo la posibilidad de iniciar sesión desde cualquier móvil y seguir accediendo a su historial personal, sin depender de un almacenamiento local que podría verse alterado o perdido.

10.3 Resultados alcanzados

En términos generales, el resultado final del proyecto cumple —e incluso supera— la visión inicial. Se ha implementado no solo la funcionalidad básica de registrar rutas y consultarlas más tarde, sino que también se han incluido múltiples mejoras complementarias: añadir imágenes a las rutas, colocar pines o marcadores durante el trayecto, cambiar foto de perfil, actualizar contraseñas, y gestionar roles o acceso premium mediante lógica de backend. Todo esto eleva la utilidad práctica de la app y mejora significativamente la experiencia del usuario final.

Una de las mayores satisfacciones personales del proyecto ha sido la capacidad de mantener una arquitectura limpia y coherente tanto en la App Android como en la API. Se ha trabajado con una clara separación de responsabilidades, dividiendo adecuadamente las capas de presentación, dominio y datos, y utilizando buenas prácticas en el diseño del código. También se ha prestado especial atención a la escalabilidad del sistema, de modo que futuras funcionalidades puedan integrarse de manera natural sin necesidad de rehacer la base del proyecto.

El diseño en **Jetpack Compose** también ha permitido una interfaz moderna, reactiva y bien estructurada, lo que aporta frescura y sencillez a la experiencia de uso. En cuanto al backend, destaca el manejo adecuado de los tokens, especialmente el **refresh token**, cuyo correcto funcionamiento fue uno de los retos técnicos más complejos del desarrollo. Hubo dificultades para que funcionara con la lógica deseada, sobre todo cuando se invalidaba el token JWT y, simultáneamente, el refresh token no era aceptado. Esto se solucionó tras revisar el interceptor de red y ajustar la lógica para que el token de refresco se utilizara correctamente de forma automática cuando fuese necesario.

10.4 Retos técnicos superados

Durante el proceso de desarrollo, se han presentado varios desafíos significativos, como:

- La implementación segura del sistema de **autenticación con Firebase** y generación personalizada de JWT.
- La correcta gestión del **refresh token**, asegurando que su uso fuera transparente para el usuario y que permitiera mantener la sesión activa sin necesidad de repetir el inicio de sesión.
- El **despliegue del backend en la nube**, para lo cual se utilizaron herramientas como **Render** para la API y **Railway** para la base de datos. Además, el empaquetado de la aplicación como .war planteó retos adicionales debido al tamaño del archivo y la necesidad de alojarlo temporalmente en plataformas como Dropbox para su posterior descarga automatizada mediante Docker.
- Otro reto fue la correcta gestión del archivo serviceAccount.json de Firebase en producción, evitando subirlo al repositorio por seguridad. Finalmente se optó por incluirlo directamente en el .war compilado, garantizando así su disponibilidad sin comprometer la seguridad.

Estos retos no solo fueron superados, sino que me ha permitido como desarrollador adquirir conocimientos valiosos sobre despliegue en la nube, control de seguridad y distribución automatizada de aplicaciones.

10.5 Mejoras y características futuras

Si bien el proyecto es funcional y completo, existen múltiples áreas de mejora y expansión para futuras versiones:

- **Animaciones interactivas:** al mostrar la ruta en detalle, sería interesante incluir una animación que muestre el recorrido del usuario con un icono móvil representando el tipo de vehículo elegido (por ejemplo, bicicleta o coche). Esto aportaría una capa visual dinámica y atractiva.
- **Estadísticas y análisis de rutas:** añadir métricas avanzadas como calorías estimadas en el caso de usar bicicleta, desniveles acumulados, comparativa de

rutas, informes semanales o mensuales, etc., enriquecería notablemente el núcleo funcional de la aplicación.

- **Seguimiento en segundo plano:** actualmente, la app requiere estar activa en primer plano para registrar el trayecto. Una mejora crítica sería permitir la grabación del recorrido en segundo plano o con la pantalla apagada, lo cual es técnicamente complejo, pero altamente demandado por usuarios reales.
- **Mayor personalización:** la inclusión de opciones de idioma, configuración de unidades (km/millas), entre otros ajustes personalizados, mejorarían la accesibilidad y adaptación del software a distintos perfiles de usuario.
- **Notificaciones push o recordatorios** para salir a realizar una ruta, o para indicar cuándo ha finalizado una, también sería una funcionalidad a considerar en el futuro.
- **Importación de rutas a formatos como GPX o KML**. Ya que actualmente se puede exportar, pero no importar.

10.6 Conclusión final

TrackMyRide ha pasado de ser una necesidad personal a una solución técnica sólida y bien diseñada, que puede ser utilizada por cualquier persona interesada en registrar y consultar sus recorridos. El proceso de desarrollo ha sido un camino de constante aprendizaje, donde los retos se han convertido en oportunidades para profundizar en temas como seguridad, arquitectura de software, despliegue en la nube y distribución móvil.

El resultado no es solo una app funcional, sino también una base extensible sobre la que se pueden construir nuevas funcionalidades y experiencias. El proyecto refleja el compromiso con la calidad, la organización del código y la atención al detalle en cada fase, desde el diseño hasta el despliegue final.
Con tiempo y recursos adicionales, el potencial de crecimiento y mejora de **TrackMyRide** es evidente, abriendo la puerta a futuras versiones más completas, inteligentes y adaptadas a nuevos contextos de uso.

11. Índice de tablas e imágenes

A continuación, se muestra un índice que recopila todas las imágenes, capturas de pantalla, esquemas y tablas utilizadas a lo largo del documento. Cada elemento está referenciado con su número correspondiente, título y página, permitiendo al lector localizar rápidamente el contenido gráfico de apoyo que complementa la explicación del texto.

Este índice se ha generado automáticamente mediante la funcionalidad “Tabla de ilustraciones” de Microsoft Word, utilizando las etiquetas asignadas a cada imagen y tabla en el documento.

Índice

12. Referencias y Bibliografía

Canales de YouTube (Tutoriales y aprendizaje)

- AristiDevs. https://www.youtube.com/@AristiDevs
- Philipp Lackner. https://www.youtube.com/@PhilippLackner
- Himanshu Gaur. https://www.youtube.com/@himanshugaur684
- Stephen Nnamani. https://www.youtube.com/@StephenNnamani-m6l
- DevKiper. https://www.youtube.com/@DevKiper
- DevExpert.io. https://www.youtube.com/@devexpert_io

Servicios en la nube y APIs

- Firebase. *Documentación Firebase*. https://firebase.google.com/docs
- Cloudinary. *Documentación Cloudinary*. https://cloudinary.com/documentation
- PayPal Developers. *PayPal SDK Guía*https://developer.paypal.com/docs
- Render. *Documentación Render*. https://render.com/docs
- Railway. *Host de base de datos*. https://railway.app

Herramientas y utilidades

- PlantUML. *PlantUML Guía*. https://plantuml.com
- Carbon. *Carbon – Imágenes código*. https://carbon.now.sh
- Photopea. *Photopea – Editor de fotos online*. https://www.photopea.com
- OpenAI – ChatGPT. https://chat.openai.com
- Figma. UI/UX Prototipos y diseño. https://www.figma.com
- DeepSeek – https://github.com/deepseek-ai

Desarrollo Android y Jetpack Compose

- Android Developers. *Jetpack Compose*. Google. https://developer.android.com/jetpack/compose
- Medium – Álvaro Rodríguez Díaz. (2023). *Google Maps + Location + Jetpack Compose*. https://medium.com/@alrodiaz15/google-maps-location-jetpack-compose-36cd3fa617a4
- Programmingheadache – https://programmingheadache.com/2024/08/25/how-to-easily-integrate-gps-location-updates-in-jetpack-compose-a-5-step-guide/