Universidad de Valladolid

Escuela de Ingeniería Informática

Trabajo Fin de Grado

Grado en Ingeniería Informática

UrbanAIR: Sistema de Gestión de préstamo de bicicletas para la Universidad de Valladolid

Autor:
D. Jorge Sanzo Hernando

Universidad de Valladolid

Escuela de Ingeniería Informática

Trabajo Fin de Grado

Grado en Ingeniería Informática

UrbanAIR: Sistema de Gestión de préstamo de bicicletas para la Universidad de Valladolid

Autor:
D. Jorge Sanzo Hernando

Tutores:
Dr. Joaquín Nicolás Adiego Rodríguez

Resumen

Este Trabajo de Fin de Grado consiste en el desarrollo de una aplicación móvil para gestionar un sistema de préstamo de bicicletas por parte de la Universidad de Valladolid. Esta aplicación está destinada a los alumnos de la Universidad de Valladolid que residan en Valladolid.

Este sistema surge con la idea de reducir la contaminación excesiva generada por los automóviles en la ciudad, el objetivo es proporcionar a los alumnos la posibilidad de evitar coger el coche para asistir diariamente a clase.

La idea principal es tratar de reducir el uso de vehículos particulares fomentando el uso de bicicletas ya que todo el centro de la ciudad dispone de buena comunicación con el carril bici.

Este problema de contaminación está producido por el enorme crecimiento de la cantidad y uso de automóviles en los últimos años, se genera una gran contaminación que se puede evitar o al menos reducir.

La propuesta es una aplicación móvil con la que el usuario podrá visualizar la localización de su bicicleta en el mapa de la ciudad, también será capaz de abrir y cerrar un candado bluetooth integrado en la bicicleta para evitar robos o usos indebidos. Se registrará la ruta recorrida por el usuario, la duración, las emisiones que ha evitado producir en ese trayecto y las calorías que ha quemado para fomentar su uso. Cada usuario dispondrá de una única bici para él.

El sistema mostrará un calendario con todos los viajes del usuario para que pueda consultar en todo momento su progreso.

También se utilizarán unos sensores para medir la calidad del aire en algunas bicicletas, recogerán datos medioambientales durante los trayectos realizados por los usuarios. Este sensor enviará los datos medidos a la aplicación móvil por bluetooth durante los trayectos realizados.

Por último se mostrara en el mapa el estado del aire de la ciudad en función de la contaminación, coloreando de verde las zonas con menor grado de emisiones en el aire y de rojo las zonas con mayor grado de emisiones en el aire.

Tabla de Contenidos

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1. Contexto y motivación

Actualmente, el tráfico de vehículos está continuamente en aumento alrededor del mundo, lo cual provoca una elevada contaminación que puede ser fácilmente reducible o evitable.

De forma genérica, las emisiones contaminantes de los automóviles se pueden dividir en dos tipos:

- Las que afectan a la salud de las personas, como el monóxido de carbono (CO), los óxidos de nitrógeno (NOx), el ozono (O3) y los óxidos de azufre (SO).
- Las que contribuyen al cambio climático, como el dióxido de carbono (CO2), que es el principal culpable del efecto invernadero.

La calidad del aire se caracteriza por la cantidad de sustancias contaminantes presentes en la atmósfera, que pueden ser perjudiciales para la salud. Viene determinada principalmente por:

- La distribución geográfica de las fuentes de emisión de contaminantes
- Las cantidades de contaminantes emitidas.

La Agencia Europea de Medio Ambiente estimó que, en España, durante el año 2014, la exposición a PM2,5 (Partículas en suspensión en la atmósfera de menos de 2,5 micras) , NOx y O3 produjo 23.000, 7.000 y 1.600 muertes prematuras, respectivamente.

Se considera que más del 30 % de la población respira aire por encima de los niveles de contaminación fijados por la UE.

Los efectos que tienen en la salud son:

- Dolor de cabeza y ansiedad:
  - SOx (Óxidos de azufre)
- Efectos en el sistema nervioso, efectos en las vías respiratorias, irritaciones, inflamaciones, infecciones, asma y enfermedades pulmonares:
  - PM (Partículas en suspensión)
- Enfermedades cardiovasculares:
  - PM (Partículas en suspensión)
  - O3 (Ozono)
  - SOx (Óxidos de azufre)

- Enfermedades en el hígado, bazo y en la sangre:
  - NOx (Óxidos de nitrógeno)
- Problemas respiratorios e irritación de nariz, ojos y garganta
  - O3 (Ozono)
  - PM (Partículas en suspensión)
  - NOx (Óxidos de nitrógeno)
  - SOx (Óxidos de azufre)
  - BaP (Creosota)
- Cáncer de pulmón
  - BaP (Creosota)

Con toda esta información podemos observar que es un problema serio de la sociedad actual y que además va en constante aumento, en este TFG vamos a centrarnos en el análisis de NOx (Óxidos de nitrógeno), PM (Partículas en suspensión), COV (Compuestos orgánicos volátiles), O3 (Ozono) y CO (óxidos de carbono) ya que son los que vamos a poder medir con los sensores medioambientales y obtener del servidor.

El desarrollo de este TFG se ve motivado por el deseo de la disminución de la concentración de dichos elementos en el aire, mejorando así la calidad de vida en la ciudad.

En los diferentes capítulos de este documento se detallarán todas las fases que este trabajo, como cualquier otro proyecto software debiera tener, desde su nacimiento como un listado de requisitos, análisis, diseño, implementación y su finalización en la fase de pruebas y posterior despliegue.

1.2. Objetivos

El objetivo de este Trabajo de Fin de Grado es proporcionar una alternativa al transporte público para los alumnos de la UVA que residan en la ciudad. También se quiere concienciar sobre la contaminación del aire en la ciudad y promover la reducción de gases contaminantes por parte de los vehículos particulares. El sistema podrá medir y analizar la contaminación de las rutas que sigan los usuarios con su bicicleta, también la duración de los viajes, las emisiones evitadas y las calorías quemadas fomentando su uso.

Todas estas funcionalidades estarán integradas en una aplicación móvil. El software de los sensores medioambientales implementado en raspberrys, el servidor que utiliza la aplicación como API REST y el software de los candados bluetooth no han sido desarrollados en este Trabajo de Fin de Grado.

1.3. Punto de partida

Para comenzar con este proyecto, desarrollar la aplicación móvil y ponerla en funcionamiento se ha necesitado:

- **Candados bluetooth**
  Estos dispositivos son los que van a permitir al usuario bloquear su bicicleta de forma que nadie más pueda utilizarla, se han elegido los candados bluetooth de la marca iLockIt [1] por su funcionalidad simple y bajo coste. El vendedor de los mismos es el que ha proveído la documentación de la API para poder conectar con ellos, abrirles y cerrarles. Los candados también poseen una alarma para evitar robos.

- **Sensores medioambientales**
  Estos dispositivos son los que van a medir las partículas que hay flotando en el aire de la ciudad, cuando el usuario viaje con su bicicleta por las calles se irán registrando los valores de la concentración de las mismas. Después se enviaran al servidor para poder evaluar el estado de la contaminación en las calles de la ciudad. Han sido desarrollados por la empresa CARTIF [2], especializada en estos proyectos. Se conectaran vía Bluetooth con la aplicación y el software necesario de cada uno de ellos estará instalado en una Raspberry.

- **Dispositivos BLE**
  Tanto los sensores como los candados funcionan como dispositivos BLE (Bluetooth low energy) [33], este tipo de dispositivos proporcionan un bajo consumo de energía y un coste considerablemente reducido, manteniendo un rango de alcance de comunicación similar a los dispositivos bluetooth habituales.

- **Servidor**: El servidor proporciona una API REST y tiene implementado un sistema de Login que permite tanto la autenticación mediante el uso de sesiones como autenticación mediante token para las consultas a la API REST. Es el encargado de dar soporte a la aplicación para guardar los datos del usuario y los datos medioambientales. Ha sido desarrollado por la empresa GMV [5].

- **Aplicación móvil**: La aplicación móvil es el objetivo de este TFG, su función es la de coordinar todos los elementos mencionados anteriormente y lograr la funcionalidad deseada. Ha sido desarrollada para el sistema operativos Android, con el IDE Android Studio y como lenguaje de programación principal se ha usado Kotlin.

1.3.1. Funcionalidades abarcadas

El trabajo desarrollado en este TFG abarca los siguientes aspectos:

- **Desarrollo de la aplicación móvil**
  Se partirá de cero en el desarrollo de la aplicación móvil que tendrá las siguientes vistas:

  - **Login**
    El usuario podrá iniciar sesión y recuperar la contraseña en caso de haberla olvidado, para crear una cuenta deberá ponerse en contacto con la Universidad de Valladolid ya que no es una funcionalidad proporcionada por la aplicación.
  - **Información medioambiental**
    El usuario podrá desplegar un menú donde podrá observar los valores de las emisiones medidas (NOx, PM, COV, O3, étc), también se mostrara un mapa donde, en función de los valores medidos, se colorearan las calles en color verde, amarillo o rojo, indicando una baja, media o alta concentración de emisiones. El usuario podrá elegir que filtro utilizar para mostrar únicamente un tipo de partículas u otro.
  - **Pantalla principal**
    El usuario podrá observar su localización y la de su bicicleta en el mapa, también podrá abrir y cerrar el candado bluetooth para iniciar o finalizar un viaje. Además se mostrara la distancia recorrida acumulada en el día, las emisiones evitadas durante el día y el estado y batería del candado de su bicicleta.
  - **Pantalla de viaje**
    El usuario podrá observar la ruta recorrida, las calorías quemadas, las emisiones evitadas y el tiempo de viaje actual, todo en tiempo real. También se mostraran los valores que mida el sensor medioambiental durante el viaje.

- **Historial de viajes**
  El usuario podrá ver todos sus viajes ordenados por días, tendrá un calendario para poder seleccionar la fecha deseada y cada viaje tendrá asociado un mapa con la ruta recorrida, además del tiempo de viaje, calorías y emisiones.
- **Perfil de usuario**
  El usuario podrá consultar su información personal (nombre, apellidos, nombre de usuario, fecha de nacimiento, género, altura y peso) y editar algunos campos como la altura y el peso. También podrá leer y consultar las políticas de privacidad y atribuciones de la aplicación, cambiar su contraseña y cerrar su sesión.

- **Conexion con el servidor**
  Para realizar la comunicación con la API REST del servidor se usara la librería Android Retrofit2 [27] la cual permite construir las peticiones Https de forma sencilla y segura, cada petición (excepto el login) incluirá los tokens de autenticación del usuario en la parte de autenticación del header de la aplicación. Cuando el usuario hace login se crean dos tokens, el de autenticación y el de refresco, ambos tienen un tiempo de caducidad. Cuando el token de autenticación caduca, se intenta conseguir uno nuevo utilizando el de refresco, si el de refresco esta caducado también, se pedirá al usuario que inicie sesión de nuevo, si no, se mantendrá abierta su sesión con los nuevos tokens hasta que uno o ambos caduquen de nuevo.

- **Conexión con los dispositivos BLE**
  Para realizar la conexión con los dispositivos BLE se ha necesitado conocer el protocolo de ambas APIs (candado y sensores), ha sido proporcionado por los fabricantes [1] y [2].

  - En el caso de los candados, cada uno de ellos posee un identificador único y tiene asociada una clave de conexión. Es necesario intercambiar la clave con el candado para recibir un código de validez. Este código de validez habrá que incluirlo en cada una de las operaciones que se quieran realizar.
  - En el caso de los sensores la conexión es más simple ya que no se pueden realizar operaciones potencialmente peligrosas con ellos, solo se puede realizar un seguimiento de los datos que miden en tiempo real, estos son parseados y formateados por la aplicación para mostrárselos al usuario de forma legible

Después de este capítulo de introducción, el siguiente capítulo se ocupará del estado del arte, en relación a la presentación del tema del TFG.

Capítulo 2

Estado del Arte

El estado del arte (State of Art) proviene originalmente del campo de la investigación técnica, científica e industrial y significa, en pocas palabras, la situación de una determinada tecnología, lo más innovador o reciente con respecto a un arte, ciencia o tecnología o específica [7].

2.1. Aplicaciones relacionadas con temas medioambientales

Actualmente cada vez son mas las empresas y a veces grupos de personas que se preocupan por el estado del medioambiente, las aplicaciones móviles hoy en día son uno de los mejores medios para poder crear conciencia sobre ello o fomentar las buenas prácticas que nos ayuden a mejorarlo o al menos a no empeorarlo más, a continuación se van a mencionar algunas aplicaciones móviles novedosas que cumplen con este propósito, todas se pueden descargar desde Google Play Store:

- **Plume Labs: Contaminación del aire**
  Esta aplicación es muy sencilla de usar, simplemente hay que registrarse o entrar como invitado y buscar la ciudad deseada, inmediatamente se muestran los niveles de contaminación de la ciudad. Estos niveles vienen divididos por franjas horarias y también se indica el índice AQI [10] que es un índice utilizado por diversas agencias gubernamentales para cuantificar y comunicar al público la calidad del aire. Indica el grado de pureza o contaminación atmosférica y los efectos para la salud humana asociados con dicha contaminación.

  Esta aplicación reúne datos de los propios usuarios mediante unos sensores de contaminación personales que ellos mismos comercian, de las ONGs relacionadas con el medioambiente, y de agencias gubernamentales.

- **Clean Spot**
  Esta aplicación también tiene un uso muy sencillo, hay que introducir la ciudad deseada, seleccionar el tipo de residuo que queremos reciclar y la aplicación nos mostrará un mapa en donde podemos localizar los puntos limpios o puntos de reciclaje más cercanos donde depositar correctamente los residuos [8].

- **Perdida de la noche**
  Esta aplicación trata de concienciar sobre el impacto que tenemos en la naturaleza con nuestras ciudades y su contaminación lumínica. La aplicación nos muestra un mapa del cielo con todas las estrellas y constelaciones que deberíamos poder ver a simple vista desde nuestra posición para que podamos comprobar con nuestros propios ojos que somos incapaces de ver la gran mayoría [21]. Esto contaminación puede alterar los ciclos biológicos de algunos animales y plantas, sobre todo las

aves, generando desorientación y cambios en sus ciclos biológicos. Incluso también se alteran los ciclos de sueño de las personas, al filtrarse la luz artificial en las viviendas.

Todas estas aplicaciones son solo una pequeña muestra de la gran cantidad de aplicaciones similares que existen hoy en día, también existen aplicaciones que fomentan el uso de bicicletas como Vallabici [11] para evitar utilizar otros medios de transporte y otras que fomentan las visitas en puntos de interés medioambientales creados en la ciudad como UrbanGreenUp [31].

2.2. Tecnologías utilizadas en este proyecto

Las tecnologías actuales de diseño de aplicaciones móviles ofrecen una gran cantidad de herramientas, con sus ventajas y desventajas. Existen desde Frameworks que realizan ellos automáticamente casi todo el trabajo y cualquiera puede realizar una aplicación sencilla, hasta Frameworks que permiten realizar aplicaciones híbridas entre Android s iOS. En este caso se ha optado por utilizar únicamente el IDE Android Studio y un desarrollo nativo en Kotlin para realizar el proyecto. Se ha tomado esta decisión por la simpleza y sencillez de uso de Android Studio, la experiencia previa en el IDE y en el lenguaje de programación Kotlin y la facilidad de personalización de cualquier detalle de la aplicación, ya sea visual o de programación.

Por otro lado existen numerosas herramientas que pueden facilitar el desarrollo, los test en fase de pruebas o el despliegue de las aplicaciones, en este caso se ha utilizado Firebase [16]. Esta herramienta de Google cuenta con muchos módulos de gran utilidad como:

- **Consola Firebase** [17]
  La consola de Firebase es un panel de control para nuestras aplicaciones, nos permite crear un sistema de autenticación, almacenar datos en la nube, realizar tareas de machine learning, realizar test, medir el rendimiento de nuestra aplicación y mucho mas. En este proyecto se ha utilizado la funcionalidad de distribución de aplicaciones.

  Esta funcionalidad que nos ofrece Firebase, nos permite distribuir nuestras aplicaciones a los usuarios que queramos sin necesidad de subirla a Google Play Store, nos permite definir grupos de usuarios que realizaran pruebas reales para distribuirles la aplicación. También nos permite llevar un registro de versiones de aplicación con las notas de versión de cada fichero ".apk"que decidamos distribuir.

- **Google Analytics** [18]
  Google Analytics es un módulo que nos permite medir estadísticas de nuestra aplicación sobre el uso de las mismas y de la participación de los usuarios. Por ejemplo podemos ver que porcentaje de usuarios nunca utiliza un botón o que porcentaje siempre viaja a la misma vista después de realizar una función concreta, de esta forma podemos optimizar y mejorar la usabilidad de la aplicación. También puede ser útil para mejorar el marketing y el rendimiento de la aplicación.

- **Firebase Crashlytics** [20]
  Firebase Crashlytics nos permite obtener estadísticas prácticas sobre los problemas de la aplicación relacionados con errores y bugs de programación. Nos informa de fallos en tiempo real y hace un seguimiento de los problemas de estabilidad que afectan a la calidad de la aplicación. Se genera un informe con toda la información relacionada con el problema surgido y sus posibles causas, de esta forma se puede aislar el problema y además comprobar si es algo común a muchos a usuarios. Puede llegar incluso a determinar qué líneas de código están provocando los fallos.

A continuación, en el siguiente capitulo se explicará y realizará el análisis y elicitación de requisitos.

Capítulo 3

Análisis de requisitos

El siguiente capítulo está dedicado al análisis de requisitos. En él se ofrece una descripción de los diferentes actores que interactúan con el sistema, así como una lista numerada de los diferentes tipos de requisitos (funcionales, no funcionales, de información y reglas de negocio). El objetivo de este capítulo es contar con una base que permita realizar la planificación del proyecto, y posteriormente la fase de análisis de la aplicación.

3.1. Definición de los actores

3.1.1. Usuarios de la aplicación

El Usuario de la aplicación será el propietario de la bicicleta, podrá iniciar sesión, ver en el mapa su vehículo, abrir y cerrarlo y ver los datos medioambientales medidos por los sensores, también podrá acceder a su historial de viajes y a su perfil de usuario.

3.1.2. Administrador

El administrador se encargara de crear las cuentas para los usuarios y de asignarles una bicicleta a cada uno, no interactuará directamente con la aplicación móvil si no que trabajara desde una página web administrativa creada por la empresa GMV [5], esta pagina web administrativa está fuera de las funcionalidades abarcadas en este Trabajo Fin de Grado.

3.2. Requisitos funcionales

Lista de las funcionalidades que deberá proporcionar la aplicación móvil:

- RF-1: El sistema deberá permitir a los usuarios iniciar sesión con su nombre de usuario y contraseña.
- RF-2: El sistema deberá permitir a los usuarios recuperar su contraseña en caso de que la hayan olvidado
- RF-3: El sistema deberá permitir a los usuarios ver la suma de la distancia total recorrida en todos los viajes para la fecha actual.
- RF-4: El sistema deberá permitir a los usuarios ver la suma de las emisiones totales evitadas en todos los viajes para la fecha actual.
- RF-5: El sistema deberá permitir a los usuarios observar si la conexión con el candado de su bicicleta esta disponible.

- RF-6: El sistema deberá permitir a los usuarios observar el nivel de batería del candado de su bicicleta.
- RF-7: El sistema deberá permitir a los usuarios observar la localización de su bicicleta en el mapa.
- RF-8: El sistema deberá permitir a los usuarios observar su localización en el mapa.
- RF-9: El sistema deberá permitir a los usuarios abrir el candado de su bicicleta e iniciar un nuevo viaje con ella.
- RF-10: El sistema deberá permitir a los usuarios cerrar el candado de su bicicleta y finalizar el viaje con ella.
- RF-11: El sistema deberá permitir a los usuarios conectarse al sensor medioambiental asociado para que puedan observar los valores de emisiones medidos en tiempo real durante un viaje.
- RF-12: El sistema deberá permitir a los usuarios observar el tiempo total transcurrido mientras realizan un viaje con su bicicleta.
- RF-13: El sistema deberá permitir a los usuarios observar la distancia total recorrida mientras realizan un viaje con su bicicleta.
- RF-14: El sistema deberá permitir a los usuarios observar el ID de la bicicleta con la que esta viajando.
- RF-15: El sistema deberá permitir a los usuarios observar los valores medidos por el sensor medioambiental en tiempo real mientras realizan un viaje con su bicicleta.
- RF-16: Mientras los usuarios estén viajando, el sistema solo permitirá a los usuarios cerrar su bicicleta y finalizar su viaje, por motivos de seguridad y para evitar distracciones no se permitirán mas interacciones entre el usuario y el dispositivo móvil.
- RF-17: El sistema deberá permitir a los usuarios observar los viajes realizados en una fecha concreta.
- RF-18: El sistema deberá permitir a los usuarios cambiar la fecha seleccionada para observar los viajes realizados.
- RF-19: El sistema deberá permitir a los usuarios observar en que fechas tienen viajes registrados.
- RF-20: El sistema deberá permitir a los usuarios observar las calorías quemadas durante un viaje en una fecha concreta.
- RF-21: El sistema deberá permitir a los usuarios observar las emisiones evitadas durante un viaje en una fecha concreta.
- RF-22: El sistema deberá permitir a los usuarios observar la distancia recorrida durante un viaje en una fecha concreta.
- RF-23: El sistema deberá permitir a los usuarios observar la hora de inicio de un viaje en una fecha concreta.
- RF-24: El sistema deberá permitir a los usuarios observar la hora de fin de un viaje en una fecha concreta.
- RF-25: El sistema deberá permitir a los usuarios observar las ruta recorrida en un mapa durante un viaje en una fecha concreta.

- RF-26: El sistema deberá permitir a los usuarios observar los niveles de contaminación en las calles de la ciudad en un mapa.
- RF-27: El sistema deberá permitir distinguir entre 3 niveles de contaminación, baja, media y alta.
- RF-28: El sistema deberá permitir a los usuarios cambiar el tipo de emisiones que desean observar en el mapa, deberá poder elegir entre CO, CO2, COV, NOx y PM
- RF-29: El sistema deberá permitir a los usuarios observar los valores medios, máximos y mínimos de todos los tipos de emisiones entre dos fechas.
- RF-30: El sistema deberá permitir a los usuarios modificar las fechas entre las que se muestran los datos medios, máximos y mínimos de todos los tipos de emisiones
- RF-31: El sistema deberá permitir a los usuarios observar su perfil con sus datos personales como nombre, apellidos, nombre de usuario, correo electrónico, fecha de nacimiento, género, peso y estatura.
- RF-32: El sistema deberá permitir a los usuarios editar los campos de altura y peso en su perfil.
- RF-33: El sistema deberá permitir a los usuarios acceder y leer la política de privacidad de la aplicación.
- RF-34: El sistema deberá permitir a los usuarios acceder y leer las atribuciones de la aplicación.
- RF-35: El sistema deberá permitir a los usuarios cambiar su contraseña de acceso.
- RF-36: El sistema deberá permitir a los usuarios cerrar su sesión.

3.3. Requisitos no funcionales

Esta sección muestra las restricciones que se aplican a las funcionalidades que debe proporcionar el sistema.

- RNF-01: La localización de las bicicletas se definirá con dos coordenadas, latitud y longitud.
- RNF-02: El sistema usará una base de datos SQLite para el almacenamiento de la información necesaria.
- RNF-03: El sistema usará la API REST creada por GMV [5] para la obtención de datos y autenticación de los usuarios.
- RNF-04: Los datos de la API REST se recibirán en formato JSON.
- RNF-05: La versión de Android mas baja con la que la aplicación debe ser compatible sera Android Lollipop (5.0) SDK Version 21.
- RNF-06: La estética de la aplicación seguirá las Material Android Design Guidelines. [14]
- RNF-07: El sistema de control de versiones se implementará con Git usando como almacenamiento en la nube un repositorio privado en el GitLab de GMV [4].
- RNF-08: Se utilizará la API de Google Maps para el funcionamiento de los mapas. [15]
- RNF-09: Se utilizara la API de Firebase Crashlytics para tener un historial de fallos y errores de los usuarios. [20]

- RNF-10: Se utilizará la librería Koin para gestionar la inyección de dependencias. [24]
- RNF-11: Se utilizará la librería Room para gestionar las operaciones CRUD de la base de datos. [13]
- RNF-12: Se utilizará la librería Retrofit2 para gestionar las peticiones y respuestas a la API REST. [27]

3.4. Reglas de negocio

Listado de las restricciones del dominio de la aplicación.

- RN-01: Un usuario solo puede tener una bicicleta a la vez.
- RN-02: Una bicicleta solo puede estar asignada a un único usuario a la vez.
- RN-03: Un usuario puede tener asignado un sensor medioambiental o no, en caso de tenerlo debe ser un único sensor al mismo tiempo.
- RN-04: Un sensor medioambiental puede estar asignado a un usuario o no, en caso de estar asignado deber ser a un único usuario.
- RN-05: Un usuario solo puede tener un único viaje activo a la vez.
- RN-06: Una bicicleta puede estar en estado Abierta, Cerrada o En mantenimiento.
- RN-07: El nivel de contaminación en las calles se representara con un gradiente de colores HSL [3] siendo el color verde el mínimo valor y el color rojo el máximo valor
- RN-08: La conexión con la bicicleta pasara a estado "Disponible" si el usuario tiene activados el bluetooth y la ubicación de su dispositivo móvil, si no, pasara a estado "No disponible".
- RN-09: El nivel de la batería del candado se actualizara después de cada una de las operaciones realizadas.

3.5. Requisitos de información

Enumeración de los datos necesarios para la aplicación.

- RI-01: Para cada bicicleta se almacenará:
    1. Identificador de la bicicleta.
    2. Identificador del candado.
    3. Estado del vehículo (Abierto, Cerrado o En mantenimiento).
    4. Ultimo valor de la latitud obtenida.
    5. Ultimo valor de la longitud obtenida.
    6. Ultimo instante de tiempo en el que se interactuó con la bicicleta.
- RI-02: Para cada viaje se almacenará:
    1. Identificador del viaje.
    2. Identificador de la bicicleta con la que se realizo el viaje.

3. Calorías quemadas durante el viaje.
4. Distancia recorrida durante el viaje.
5. Emisiones evitadas durante el viaje.
6. Instante de tiempo en que comenzó el viaje.
7. Instante de tiempo en que finalizó el viaje.
8. Posiciones registradas durante el viaje.

- RI-03: Para cada posición de un viaje se almacenará:

1. Bearing.
2. Latitud.
3. Longitud.
4. Instante de tiempo en que fue registrada.
5. Velocidad del usuario.

- RI-04: Para cada usuario se almacenará:

1. Username.
2. Nombre.
3. Primer apellido.
4. Segundo apellido.
5. Email.
6. Genero.
7. Edad.
8. Peso.
9. Altura.

- RI-05: Para los datos medioambientales se almacenará:

1. Instante de tiempo inicial de muestra de datos.
2. Instante de tiempo final de muestra de datos.
3. Lista de posiciones medidas.
4. Valor medio de CO
5. Valor medio de CO2
6. Valor medio cd COV
7. Valor medio de NOx
8. Valor medio de Pm
9. Valor mínimo de CO
10. Valor mínimo de CO2
11. Valor mínimo cd COV
12. Valor mínimo de NOx
13. Valor mínimo de Pm

14. Valor máximo de CO
15. Valor máximo de CO2
16. Valor máximo cd COV
17. Valor máximo de NOx
18. Valor máximo de Pm

- RI-06: Para cada posición registrada por los sensores medioambientales se almacenará:

1. Latitud
2. Longitud
3. Instante de tiempo en que fue medida
4. Valor del CO
5. Valor del CO2
6. Valor del COV
7. Valor del NOx
8. Valor del Pm

Tras este capítulo de análisis de requisitos, el siguiente capítulo se encargará de detallar el plan de proyecto seguido en todo el desarrollo.

Capítulo 4

Modelo de análisis

Este capítulo está dedicado al análisis de la aplicación, utilizando como punto de partida los requisitos previamente elicitados.

4.1. Casos de uso del sistema

En esta sección se dedica al estudio de los casos de uso, los cuales se definen como una secuencia de acciones realizadas por el sistema, que producen un resultado valioso para un actor en particular [**?**]. Los casos de uso permiten describir el sistema, el entorno del mismo y la interacción entre el sistema y su entorno; y cuentan con tres objetivos fundamentales: capturar el detalle de los requisitos funcionales del sistema, guiar la construcción de los modelos de diseño y servir de base para las pruebas del sistema.

4.1.1. Casos de uso

Este primer diagrama sirve para obtener una idea aproximada de las distintas interacciones de los actores con el sistema.

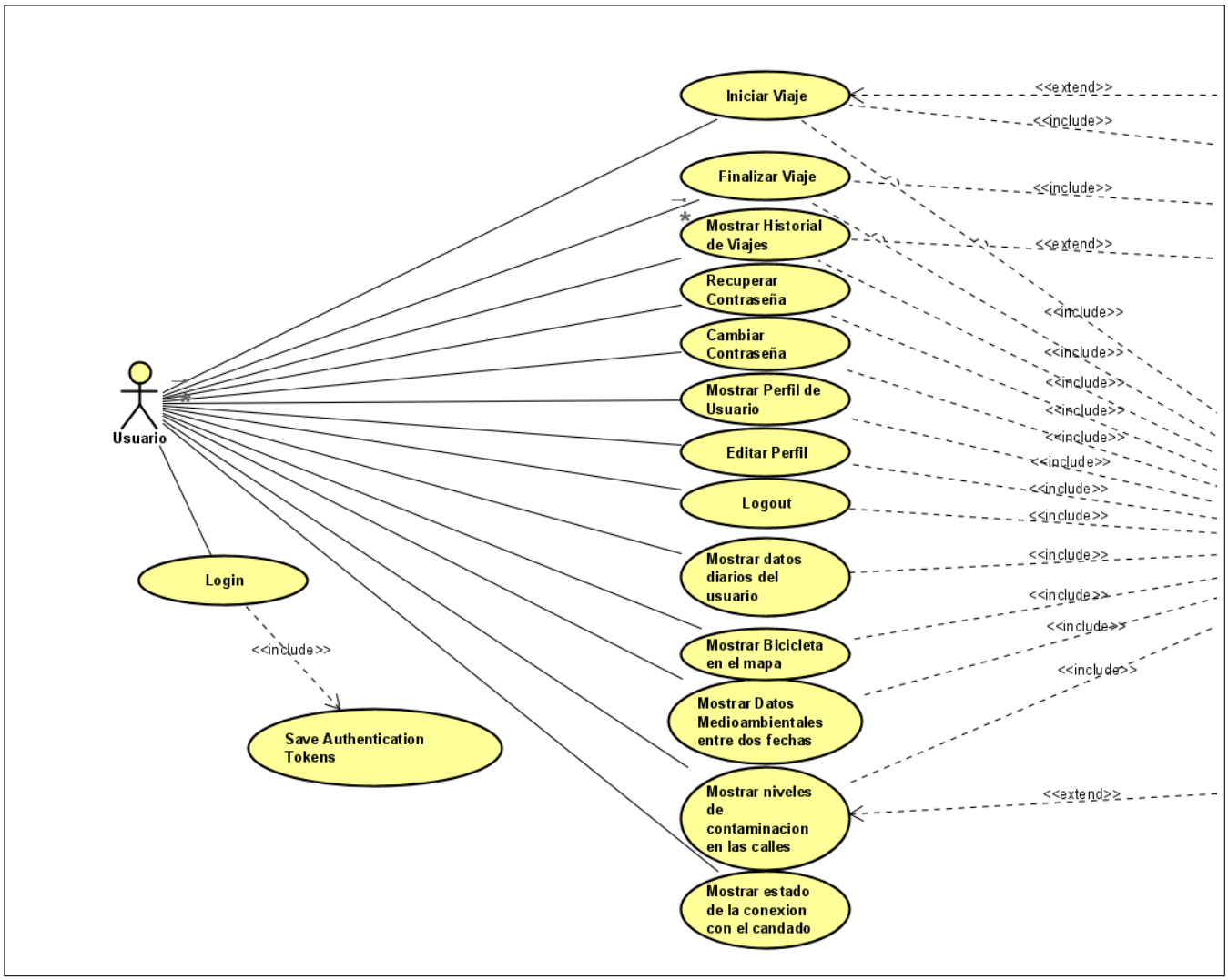

Figura 4.1: Diagrama de los caso de uso parte 1.

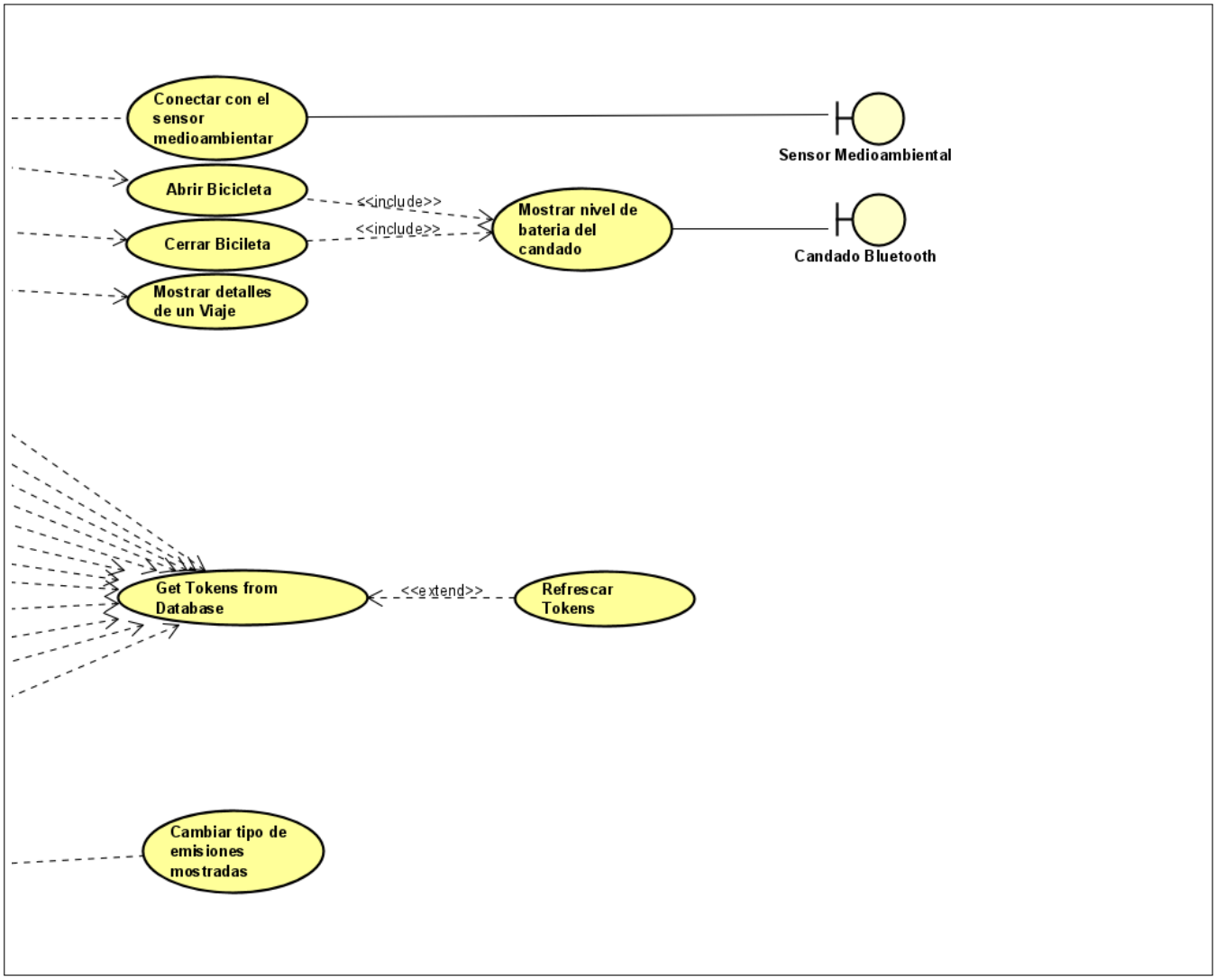

Figura 4.2: Diagrama de los caso de uso parte 2.

A continuación se va a proceder a explicar en detalle cada uno de los casos de uso.

CU-1	**Iniciar Sesión**
Actor	**Usuario**
Descripción	El sistema deberá permitir al actor iniciar sesión usando su nombre de usuario y contraseña
Precondición	El actor ha sido dado de alta en el sistema
Secuencia Normal	
Paso	**Acción**
1	El actor introduce su nombre de usuario y contraseña y presiona el botón "Acceder".
2	El sistema envía las credenciales a la API REST que gestiona las peticiones de la aplicación.
3	El sistema recibe los tokens de autenticación (Acceso y Refresco) de la API REST y ejecuta el caso de uso CU-1.2 (4.2).
4	El sistema muestra la pantalla principal de la aplicación al usuario.
Postcondición	Se muestra la pantalla inicial al usuario con su sesión activa.
Excepciones	
Variación	**Acción**
2b	Si el nombre de usuario o la contraseña están vacíos se mostrará un mensaje de error al actor.
2c	Si el nombre de usuario o la contraseña son incorrectos se mostrará un mensaje de error al actor.
2d	Si el sistema no logra conectar con la API REST, se mostrará un mensaje de error al actor.

Tabla 4.1: Caso de Uso 1 - Iniciar Sesión

CU-1.2	Save Authentication Tokens
Actor	**Usuario**
Descripción	El sistema deberá guardar en la base de datos los tokens recibidos de la API REST
Precondición	El actor ha introducido su nombre de usuario y contraseña y ha intentado iniciar sesión.
Secuencia Normal	
Paso	**Acción**
1	El sistema recibe los tokens de autenticación (Acceso y Refresco) de la API REST.
2	El sistema guarda en la base de datos local los valores de ambos tokens.
Postcondición	Los tokens de autenticación han sido guardados en la base de datos.
Excepciones	
Variación	**Acción**
1b	Si el sistema no logra conectarse con la API REST se mostrará un mensaje de error al usuario.
2b	Si el sistema no puede guardar los tokens en la base de datos se mostrará un mensaje de error al usuario.

Tabla 4.2: Caso de Uso 1.2 - Save Authentication Tokens

CU-2	Iniciar Viaje
Actor	**Usuario**
Descripción	El sistema deberá permitir al actor iniciar un nuevo viaje
Precondición	El actor ha iniciado sesión y tiene una bicicleta asignada correctamente.
Secuencia Normal	
Paso	**Acción**
1	El actor presiona el botón de "Iniciar viaje".
2	El sistema ejecuta el caso de uso "Abrir Bicicleta"
3	El sistema envía a la API REST el mensaje de que la bicicleta ha sido abierta.
4	El sistema elimina todos los elementos de la pantalla excepto el mapa, la pantalla con la información del viaje y los botones de "Abrir", "Cerrar" y "Centrar Mapa" para cumplir con el RF-18.
5	El sistema comienza el viaje del actor y registra la distancia, duración y ruta.
6	El sistema ejecuta el caso de uso "Conectar con el sensor medioambiental".
Postcondición	El usuario tiene un viaje activo y se están registrando la duración, emisiones, calorías y ruta que está siguiendo.
Excepciones	
Variación	**Acción**
4b	Si el sistema no logra conectar con la API REST, se mostrará un mensaje de error al usuario.

Tabla 4.3: Caso de Uso 2 - Iniciar viaje

CU-2.1	Abrir Bicicleta
Actor	**Usuario**
Descripción	El sistema abre el candado bluetooth instalado en la bicicleta
Precondición	El actor ha intentado iniciar un viaje.
Secuencia Normal	
Paso	**Acción**
1	El sistema comienza a escanear en busca del candado bluetooth asociado a la bicicleta asignada al usuario.
2	El sistema encuentra el dispositivo, genera la clave de acceso cifrada a partir de la clave de acceso del candado y la envía al candado.
3	El candado bluetooth acepta la conexión y envían un contador de autenticación de operación al sistema.
4	El sistema aumenta en uno el contador recibido y lo envía junto con el código de operación de abrir al candado.
5	El candado bluetooth aumenta en uno el contador recibido, abre el candado y envía de nuevo el contador de operación al sistema.
6	El sistema recibe el contador de operación y ejecuta el caso de uso "Mostrar batería del candado"
Postcondición	El candado bluetooth está abierto y el usuario puede mover su bicicleta.
Excepciones	
Variación	**Acción**
2b	Si el sistema no logra encontrar el candado bluetooth, se mostrará un mensaje de error al usuario y no se comenzara a registrar su viaje.
3b	Si el proceso de autenticación falla, se mostrará un mensaje de error al usuario y no se comenzara a registrar su viaje.
4b	Si la conexión con el candado falla, se muestra un mensaje de error al usuario y el candado no se abre.
6b	Si el candado ya estaba abierto, se mostrará un mensaje de error al usuario.

Tabla 4.4: Caso de Uso 2.1 - Abrir bicicleta

CU-2.2	Conectar con el sensor medioambiental
Actor	**Usuario**
Descripción	El sistema deberá permitir al actor finalizar un viaje activo.
Precondición	El actor ha iniciado sesión y tiene una bicicleta asignada correctamente.
Secuencia Normal	
Paso	**Acción**
1	El sistema comienza a escanear en busca del sensor medioambiental asignado al usuario.
2	El sistema encuentra el dispositivo y solicita la conexión con el sensor medioambiental.
3	El sensor medioambiental acepta la conexión y empieza a enviar los valores de las emisiones medidos en tiempo real, también enviará todos los datos medidos de forma acumulada a la API REST de Cartif [2] (desarrolladores del sensor medioambiental) cada vez que se conecte a una red Wifi abierta o previamente configurada por ellos, como la de la Universidad de Valladolid por ejemplo.
4	El sistema actualiza la pantalla de información del viaje añadiendo los campos necesarios para mostrar el valor de las emisiones recibidas.
Postcondición	El usuario puede observar en la pantalla de información de viaje los valores que esta midiendo el sensor medioambiental.
Excepciones	
Variación	**Acción**
2b	Si el sistema no logra encontrar el sensor medioambiental bluetooth, se continuará registrando el viaje con normalidad pero no se mostrará información acerca de las emisiones medidas.
4b	Si el sistema pierde la conexión con el sensor medioambiental bluetooth, se continuará registrando el viaje con normalidad pero no se mostrará información acerca de las emisiones medidas.

Tabla 4.5: Caso de Uso 2.3 - Conectar con el sensor medioambiental

CU-3	Finalizar Viaje
Actor	**Usuario**
Descripción	El sistema deberá permitir al actor finalizar un viaje activo.
Precondición	El actor tiene un viaje activo.
Secuencia Normal	
Paso	**Acción**
1	El actor presiona el botón de "Finalizar Viaje".
2	El sistema sistema ejecuta el caso de uso "Cerrar bicicleta".
3	El sistema envía a la API REST el mensaje de que la bicicleta ha sido cerrada y las coordenadas de la posición del usuario donde ha realizado la operación
4	El sistema vuelve a mostrar todos los elementos iniciales de la pantalla que se habían ocultado previamente en el caso de uso 2 (4.3)
Postcondición	El usuario ha finalizado su viaje.
Excepciones	
Variación	**Acción**
3b	Si el sistema no logra conectar con la API REST, se mostrará un mensaje de error y el viaje no habrá finalizado.

Tabla 4.6: Caso de Uso 3 - Finalizar viaje

CU-3.1	Cerrar Bicicleta
Actor	**Usuario**
Descripción	El sistema cierra el candado bluetooth instalado en la bicicleta
Precondición	El actor ha intentado finalizar un viaje.
Secuencia Normal	
Paso	**Acción**
1	El sistema comienza a escanear en busca del candado bluetooth asociado a la bicicleta asignada al usuario.
2	El sistema encuentra el dispositivo, genera la clave de acceso cifrada a partir de la clave de acceso del candado y la envía al candado.
3	El candado bluetooth acepta la conexión y envían un contador de autenticación de operación al sistema.
4	El sistema aumenta en uno el contador recibido y lo envía junto con el código de operación de cerrar al candado.
5	El candado bluetooth aumenta en uno el contador recibido, cierra el candado y envía de nuevo el contador de operación al sistema.
6	El sistema recibe el contador de operación y ejecuta el caso de uso "Mostrar batería del candado"
6	El sistema recibe el contador de operación y ejecuta el caso de uso "Mostrar batería del candado"
Postcondición	El candado bluetooth está cerrado y el usuario no puede mover su bicicleta.
Excepciones	
Variación	**Acción**
2b	Si el sistema no logra encontrar el candado bluetooth, se mostrará un mensaje de error al usuario y no se comenzara a registrar su viaje.
3b	Si el proceso de autenticación falla, se mostrará un mensaje de error al usuario y no se comenzara a registrar su viaje.
4b	Si la conexión con el candado falla, se muestra un mensaje de error al usuario y el candado no se cierra.
6b	Si el candado ya estaba cerrado, se mostrará un mensaje de error al usuario.

Tabla 4.7: Caso de Uso 3.1 - Cerrar Bicicleta

CU-4	Mostrar nivel de batería del candado
Actor	**Usuario**
Descripción	El sistema solicita el nivel de batería actual al candado bluetooth
Precondición	El usuario ha realizado alguna operación con el candado, independientemente haya sido abrir o cerrar, con éxito.
Secuencia Normal	
Paso	**Acción**
1	El sistema aumenta el contador de operación en uno y lo envía con el código de operación de obtener el nivel de batería actual al candado.
2	El candado bluetooth aumenta en uno el contador de operación y lo envía de vuelta junto con el nivel actual de su batería.
3	El sistema actualiza y muestra el nivel de batería actual del candado al usuario.
Postcondición	El sistema muestra el nivel actual de la batería del candado.
Excepciones	
Variación	**Acción**
1b	Si el sistema no logra conectar con el candado bluetooth, se mostrará un mensaje de error al usuario y no se actualizara el nivel de batería del candado en el sistema.

Tabla 4.8: Caso de Uso 4 - Mostrar nivel de batería del candado

CU-5	Mostrar Historial de viajes
Actor	**Usuario**
Descripción	El sistema mostrará todos los viajes realizados en el día seleccionado por el actor, este caso de uso se ejecuta cada vez que el usuario cambia la fecha seleccionada, para poder cambiar de fecha el usuario tendrá un calendario desplegable desde donde podrá hacerlo.
Precondición	El actor ha iniciado sesión con su nombre de usuario y contraseña.
Secuencia Normal	
Paso	**Acción**
1	El actor presiona el botón "Mis viajes".
2	El sistema solicita a la API REST la lista de viajes del actor para el día seleccionado.
3	El sistema recibe de la API REST todos los viajes del actor para el día seleccionado y los muestra en pantalla.
Postcondición	El sistema muestra los viajes registrados por el actor en la fecha seleccionada.
Excepciones	
Variación	**Acción**
2b	Si el sistema no logra conectar con la API REST, se mostrará un mensaje de error al usuario y no se mostrarán los viajes registrados para la fecha seleccionada.
3b	Si el sistema recibe una lista vacía de viajes de la API REST porque el actor no tiene ninguno registrado en la fecha seleccionada, se mostrará un mensaje informativo de aviso al usuario.

Tabla 4.9: Caso de Uso 5 - Mostrar el historial de viajes

CU-5.1	Mostrar Detalles de un Viaje
Actor	**Usuario**
Descripción	El sistema mostrará los detalles del viaje seleccionado por el actor
Precondición	El actor ha seleccionado un viaje de su historial para consultar sus detalles
Secuencia Normal	
Paso	**Acción**
1	El actor selecciona un viaje de la lista de viajes registrados en el día seleccionado.
2	El sistema muestra las calorías quemadas, las emisiones evitadas, la hora de inicio, la hora de fin y la distancia total recorrida durante el viaje.
3	El actor presiona el botón "Consultar Ruta".
4	El sistema despliega un mapa con la ruta recorrida por el actor durante el viaje.
Postcondición	El sistema muestra los viajes registrados por el actor en la fecha seleccionada.
Excepciones	
Variación	**Acción**
4b	El sistema no puede desplegar el mapa debido a un error de conexión o carga, se mostrará un mensaje de error al usuario y no se mostrará la ruta recorrida.

Tabla 4.10: Caso de Uso 5.1 - Mostrar Detalles de un Viaje

CU-6	Recuperar Contraseña
Actor	**Usuario**
Descripción	El sistema permitirá al usuario recuperar su contraseña en caso de que no la recuerde.
Precondición	El actor ha sido dado de alta en el sistema
Secuencia Normal	
Paso	**Acción**
1	El actor presiona el botón "Contraseña olvidada".
2	El sistema solicita el nombre de usuario del actor.
3	El actor introduce su nombre de usuario y presiona el botón "Recuperar contraseña".
4	El sistema solicita a la API REST que envíe un correo electrónico al usuario con los datos necesarios para recuperar su contraseña y muestra un aviso al usuario para que revise la bandeja de entrada de su correo electrónico
Postcondición	El actor ha recibido un correo electrónico con las instrucciones de recuperación de contraseña.
Excepciones	
Variación	**Acción**
3b	El usuario deja vacío el nombre de usuario y el sistema muestra un mensaje de error al actor.
3c	El usuario introduce un nombre de usuario que no es válido y el sistema muestra un mensaje de error al actor.

Tabla 4.11: Caso de Uso 6 - Recuperar Contraseña

CU-7	Cambiar Contraseña
Actor	**Usuario**
Descripción	El sistema permitirá al actor cambiar su contraseña.
Precondición	El actor ha iniciado sesión.
Secuencia Normal	
Paso	**Acción**
1	El actor presiona el botón "Cambiar contraseña".
2	El sistema envía a la API REST la petición de cambio de contraseña y muestra un mensaje de información al usuario confirmando su cambio de contraseña.
Postcondición	El actor ha cambiado su contraseña de acceso al sistema.
Excepciones	
Variación	**Acción**
2b	Si la conexión con la API REST falla, el sistema muestra un mensaje de error al actor.

Tabla 4.12: Caso de Uso 7 - Cambiar Contraseña

CU-8	Mostrar Perfil de Usuario
Actor	**Usuario**
Descripción	El sistema permitirá al actor consultar los datos de su perfil de usuario.
Precondición	El actor ha iniciado sesión.
Secuencia Normal	
Paso	**Acción**
1	El actor presiona el botón "Perfil de usuario".
2	El sistema solicita los datos del perfil del actor a la API REST.
3	El sistema muestra al usuario su nombre, apellidos, correo electrónico, nombre de usuario, peso, altura, genero y fecha de nacimiento.
Postcondición	El sistema muestra los datos del perfil del usuario.
Excepciones	
Variación	**Acción**
3b	Si el sistema no consigue conectar con la API REST, se mostrará un mensaje de error al actor.

Tabla 4.13: Caso de Uso 8 - Mostrar perfil de usuario

CU-9	Editar Perfil de Usuario
Actor	**Usuario**
Descripción	El sistema permitirá al actor editar los datos de su perfil de usuario.
Precondición	El actor ha entrado en su perfil de usuario.
Secuencia Normal	
Paso	**Acción**
1	El actor presiona el botón "Editar Perfil de usuario".
2	El sistema permite editar al usuario la altura y el peso.
3	El actor presiona el botón "Guardar datos" después de haber modificado los valores deseados.
4	El sistema comunica a la API REST el cambio de datos en el perfil de usuario.
5	El sistema muestra un mensaje de información confirmando la actualización de los datos del perfil del usuario.
Postcondición	El actor ha actualizado los datos de sus perfil.
Excepciones	
Variación	**Acción**
4b	Si el sistema no consigue conectar con la API REST, se mostrará un mensaje de error al actor.

Tabla 4.14: Caso de Uso 9 - Editar perfil de usuario

CU-10	Logout
Actor	**Usuario**
Descripción	El sistema permitirá al actor cerrar su sesión
Precondición	El actor ha iniciado sesión.
Secuencia Normal	
Paso	**Acción**
1	El actor presiona el botón "Cerrar sesión".
2	El sistema comunica a la API REST que el actor ha solicitado cerrar su sesión.
3	El sistema muestra la pantalla de login y muestra un mensaje de información confirmando el cierre de sesión del actor.
Postcondición	El sistema ha cerrado la sesión y muestra la pantalla de login.
Excepciones	
Variación	**Acción**
4b	Si el sistema no consigue conectar con la API REST, se volverá al la pantalla de login igualmente.

Tabla 4.15: Caso de Uso 10 - Logout

CU-11	Mostrar datos diarios del usuario
Actor	**Usuario**
Descripción	El sistema mostrara la distancia recorrida, calorías quemadas y emisiones evitadas durante el día actual
Precondición	El actor ha iniciado sesión.
Secuencia Normal	
Paso	**Acción**
1	El actor presiona el botón "Pantalla principal".
2	El sistema solicita a la API REST los viajes realizados por el usuario en la fecha actual
3	El sistema muestra calcula el total de distancia recorrida, emisiones evitadas y calorías quemadas de todos los viajes recibidos y los muestra en la pantalla principal.
Postcondición	El sistema muestra los el total de distancia recorrida, emisiones evitadas y calorías quemadas de todos los viajes de la fecha actual.
Excepciones	
Variación	**Acción**
2b	Si el sistema no consigue conectar con la API REST, se mostrará un mensaje de error al actor
2b	Si el sistema recibe de la API REST una lista vacía de viajes para la fecha actual se mostrará cero en todos los valores

Tabla 4.16: Caso de Uso 11 - Mostrar datos diarios del usuario

CU-12	Mostrar bicicleta en el mapa
Actor	**Usuario**
Descripción	El sistema mostrara la localización de la bicicleta en el mapa.
Precondición	El actor ha iniciado sesión y tiene una bicicleta asignada.
Secuencia Normal	
Paso	**Acción**
1	El actor presiona el botón "Pantalla principal".
2	El sistema solicita a la API REST la última localización registrada de la bicicleta del usuario.
3	El sistema muestra en el mapa un dibujo de una bicicleta en las coordenadas recibidas de la API REST.
Postcondición	El sistema muestra los el total de distancia recorrida, emisiones evitadas y calorías quemadas de todos los viajes de la fecha actual.
Excepciones	
Variación	**Acción**
2b	Si el sistema no consigue conectar con la API REST, se mostrará un mensaje de error al actor

Tabla 4.17: Caso de Uso 12 - Mostrar bicicleta en el mapa

CU-13	Mostrar datos medioambientales entre dos fechas
Actor	**Usuario**
Descripción	El sistema mostrara los valores máximos, mínimos y medios de las emisiones de CO, CO2, COV, NOx y PM. El usuario tendrá un dialogo desplegable donde podrá seleccionar las fechas que desee tanto de inicio de recogida de datos como de finalización.
Precondición	El actor ha iniciado sesión.
Secuencia Normal	
Paso	**Acción**
1	El actor presiona el botón "Datos Medioambientales".
2	El sistema solicita a la API REST los valores máximos, mínimos y medios de las emisiones de CO, CO2, COV, NOx y PM entre la fecha de inicio y la fecha de fin seleccionadas por el usuario.
3	El sistema muestra los valores máximos, mínimos y medios de las emisiones de NoX, Pm, COV y O3.
Postcondición	El sistema muestra los valores máximos, mínimos y medios de las emisiones de CO, CO2, COV, NOx y PM.
Excepciones	
Variación	**Acción**
2b	Si el sistema no consigue conectar con la API REST, se mostrará un mensaje de error al actor y se mostrará cero en todos los valores
2c	Si el actor no ha seleccionado ninguna fecha se seleccionará por defecto el día actual como fecha de inicio y el día siguiente como fecha de fin.

Tabla 4.18: Caso de Uso 13 - Mostrar datos medioambientales entre dos fechas

CU-14	Mostrar niveles de contaminación en las calles
Actor	**Usuario**
Descripción	El sistema coloreará las calles donde hayan sido recogidos los datos en un gradiente de color que va desde el verde (nivel bajo) hasta el rojo (nivel alto) pasando por el amarillo (nivel medio).
Precondición	El actor ha iniciado sesión.
Secuencia Normal	
Paso	**Acción**
1	El actor presiona el botón "Datos Medioambientales".
2	El sistema solicita a la API REST los valores máximos, mínimos y medios de las emisiones de CO, CO2, COV, NOx y PM entre la fecha de inicio y la fecha de fin seleccionadas por el usuario.
3	El sistema analiza la posición de cada una de las emisiones medidas y calcula el color correspondiente de cada punto en función del valor de emisiones obtenido, pintando las calles con el color correspondiente.
Postcondición	El sistema muestra las calles con el color correspondiente al nivel de contaminación medido.
Excepciones	
Variación	**Acción**
2b	Si el sistema no consigue conectar con la API REST, se mostrará un mensaje de error al actor y no se pintará el mapa.
2c	Si el actor no ha seleccionado ninguna fecha se seleccionará por defecto el día actual como fecha de inicio y el día siguiente como fecha de fin.

Tabla 4.19: Caso de Uso 14 - Mostrar niveles de contaminación en las calles

CU-14.1	Cambiar tipo de emisiones mostradas
Actor	**Usuario**
Descripción	El sistema mostrará al usuario un filtro para cada una de las emisiones medidas CO, CO2, COV, NOx y PM, permitiendo al usuario observar el nivel de contaminación de cada una de ellas por separado.
Precondición	El actor ha iniciado sesión.
Secuencia Normal	
Paso	**Acción**
1	El actor presiona presiona el botón "Cambiar filtro de emisiones".
2	El sistema muestra un botón para cada una de las emisiones medidas (CO, CO2, COV, NOx y PM).
3	El actor selecciona uno de los filtros disponibles.
4	El sistema borra los valores coloreados en el mapa anteriormente y selecciona solo los de las emisiones correspondientes a las seleccionas por el usuario, después colorea el mapa solamente con estos valores.
Postcondición	El sistema muestra las calles con el color correspondiente al nivel de contaminación medido y al tipo de emisiones seleccionado.
Excepciones	
Variación	**Acción**
3b	Si el actor no selecciona ningún filtro, se seleccionará por defecto el filtro de emisiones de tipo COV.

Tabla 4.20: Caso de Uso 14.1 - Cambiar tipo de emisiones mostradas

CU-15	Mostrar el estado de conexión con el candado
Actor	**Usuario**
Descripción	El sistema mostrará el estado de la conexión con el candado bluetooth.
Precondición	El actor ha iniciado sesión.
Secuencia Normal	
Paso	**Acción**
1	El actor presiona presiona el botón "Pantalla principal".
2	El sistema comprueba que el usuario tiene activados el bluetooth y la ubicación en el dispositivo móvil y que ha concedido los permisos de uso a ambos.
3	El sistema muestra el estado de la conexión con el candado como "Disponible".
Postcondición	El sistema muestra el estado de la conexión con el candado.
Excepciones	
Variación	**Acción**
2b	Si el actor no tiene activado el bluetooth, la ubicación o no ha concedido los permisos de uso, el estado de la conexión con el candado pasara a estar "No disponible".

Tabla 4.21: Caso de Uso 15 - Mostrar el estado de conexión con el candado

CU-16	Get Tokens from Database
Actor	**Usuario**
Descripción	Cada vez que se realiza una petición a la API REST del tipo que sea, excepto el Login, se ejecutara este caso de uso para poder autenticar las peticiones y mantener una sesión de usuario activa. Esta basado en el uso de un token de acceso y otro de refresco de sesión, ambos con un tiempo de caducidad.
Precondición	El actor ha iniciado sesión.
Secuencia Normal	
Paso	**Acción**
1	El actor realiza cualquier petición a la API REST.
2	El sistema recoge de la base de datos el token de acceso y de refresco.
3	El sistema incluye en la cabecera de la petición web, en el campo "Authorization" el valor del token de acceso.
3	El sistema recibe el código `OK_CODE 200` de la API REST y continua con el caso de uso que estuviese ejecutando.
Postcondición	El sistema resuelve con éxito la petición a la API REST.
Excepciones	
Variación	**Acción**
2b	Si el sistema recibe un `UNATHORIZED_CODE 401`, quiere decir que el token de acceso del usuario ha caducado y ejecuta el caso de uso 17 (4.23).

Tabla 4.22: Caso de Uso 16 - Get Tokens from Database

CU-17	Refrescar tokens
Actor	**Usuario**
Descripción	Cada vez que el token de acceso caduca, se realiza una petición a la API REST para obtener uno nuevo pero esta vez usando el token de refresco como autentificación de la petición, ambos tokens tienen distinto tiempo de caducidad, siendo el de refresco el que mas aguanta sin expirar.
Precondición	El actor ha iniciado sesión.
Secuencia Normal	
Paso	**Acción**
1	El sistema solicita a la API REST un nuevo token de acceso usando el de refresco como autentificación.
2	El sistema recibe de la API REST un nuevo token de acceso y de refresco y ejecuta el caso de uso 1.2 (4.2).
3	El sistema relanza la ultima petición pendiente a la API REST pendiente con el nuevo token de acceso como autentificación.
Postcondición	El sistema ha actualizado el token de acceso y de refresco.
Excepciones	
Variación	**Acción**
2b	Si el sistema recibe un `UNATHORIZED_CODE 401`, quiere decir que el token de refresco del usuario ha caducado y lleva al usuario de nuevo al login para que inicie sesión.

Tabla 4.23: Caso de Uso 17 - Refrescar tokens

4.2. Diagramas de secuencia

Los diagramas de secuencia son un tipo de diagramas de acción, modelos que describen cómo colaboran grupos de objetos mediante mensajes que se pasan entre ellos, que muestran las interacciones expresadas en función de secuencias temporales . En el siguiente apartado se muestran los diagramas de secuencia de aquellos casos de uso que se han considerado más representativos para el sistema. Al ser una fase de análisis, dichos diagramas de secuencia no explicarán como realizar cada una de las operaciones en profundidad, será una visión genérica de como deben desarrollarse los casos de uso. Más adelante, en la fase de diseño entraremos en detalle en cada uno de los casos de uso.

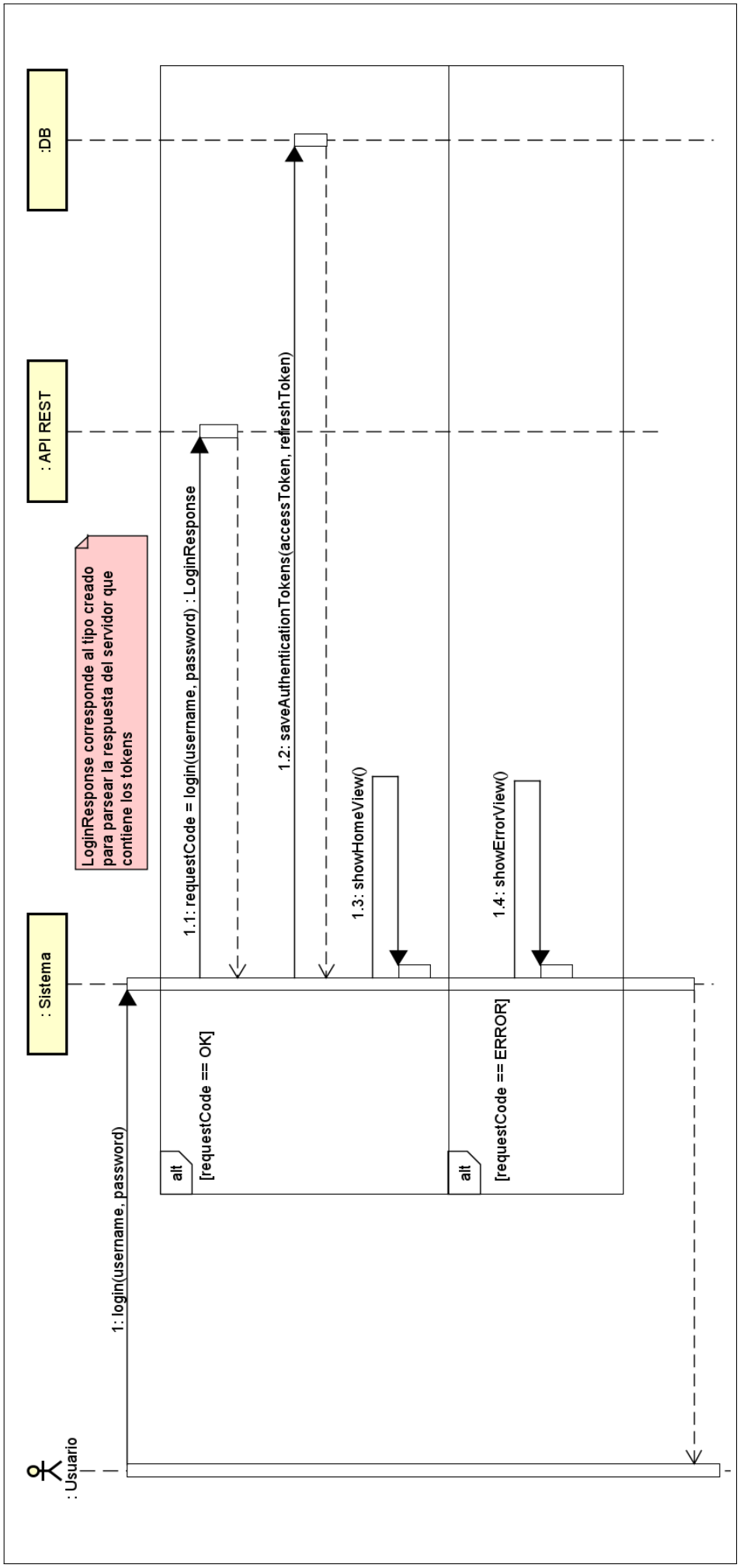

Figura 4.3: Diagrama de secuencia del caso de uso Iniciar Sesión.

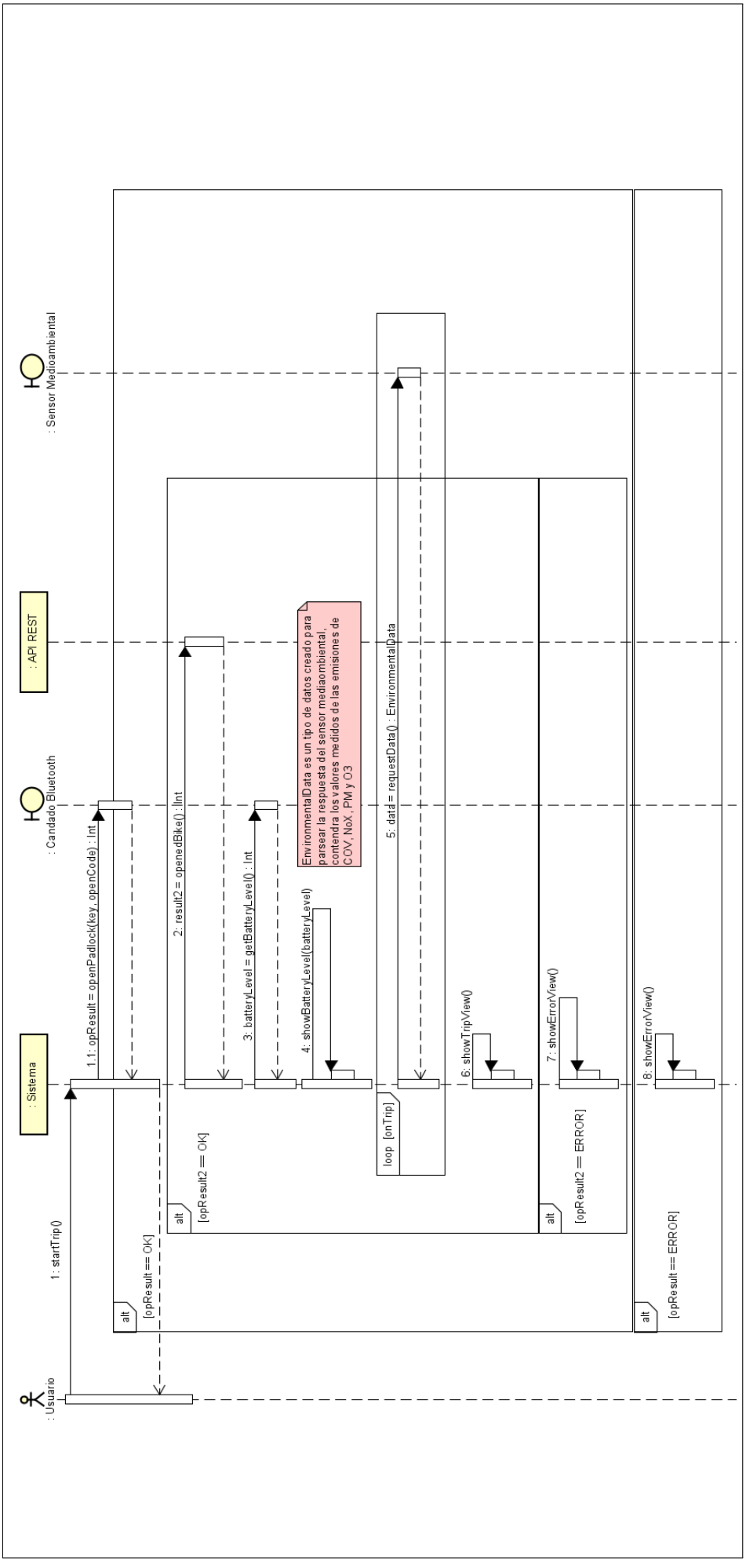

Figura 4.4: Diagrama de secuencia del caso de uso Iniciar Viaje.

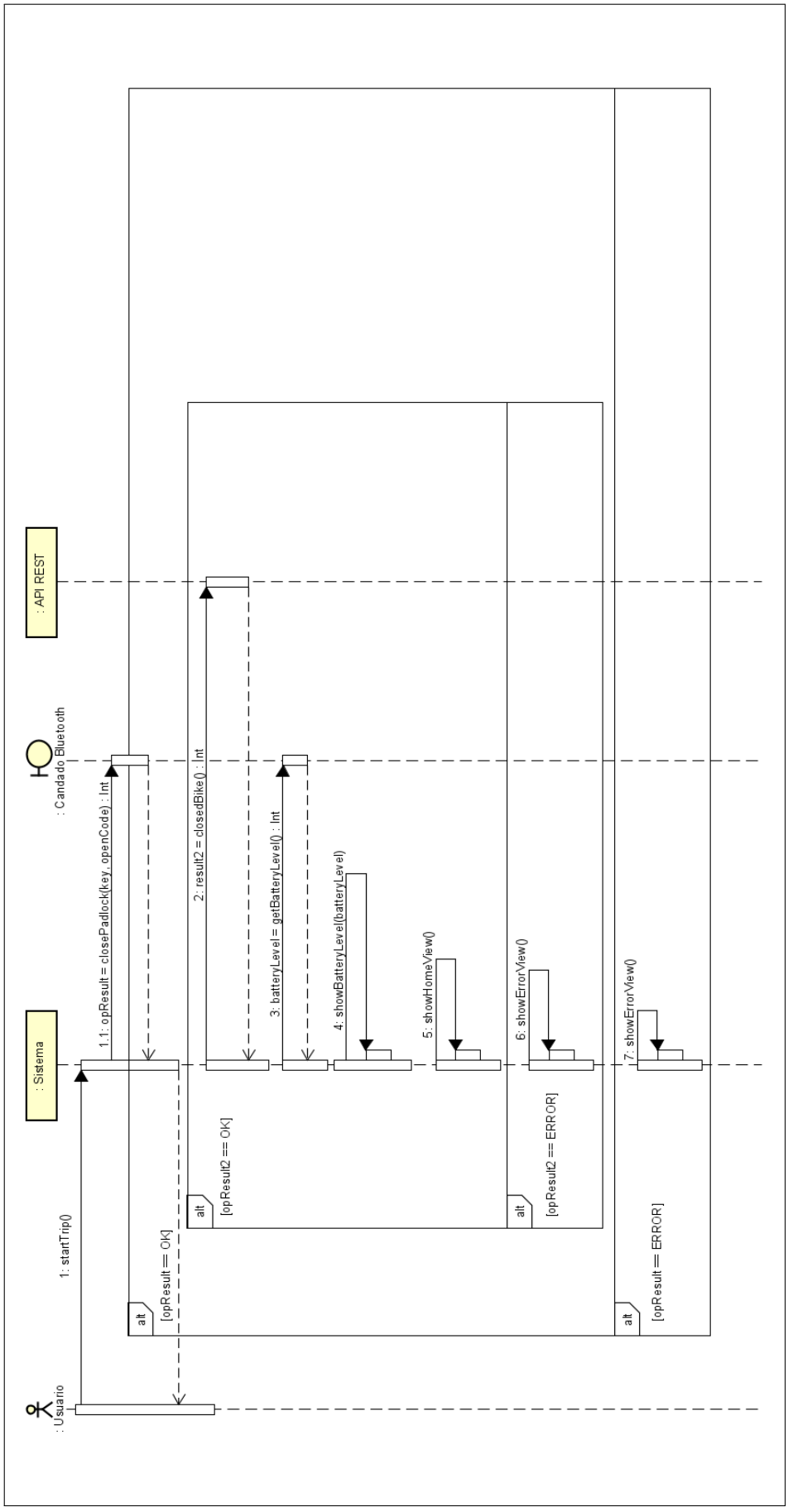

Figura 4.5: Diagrama de secuencia del caso de uso Finalizar Viaje.

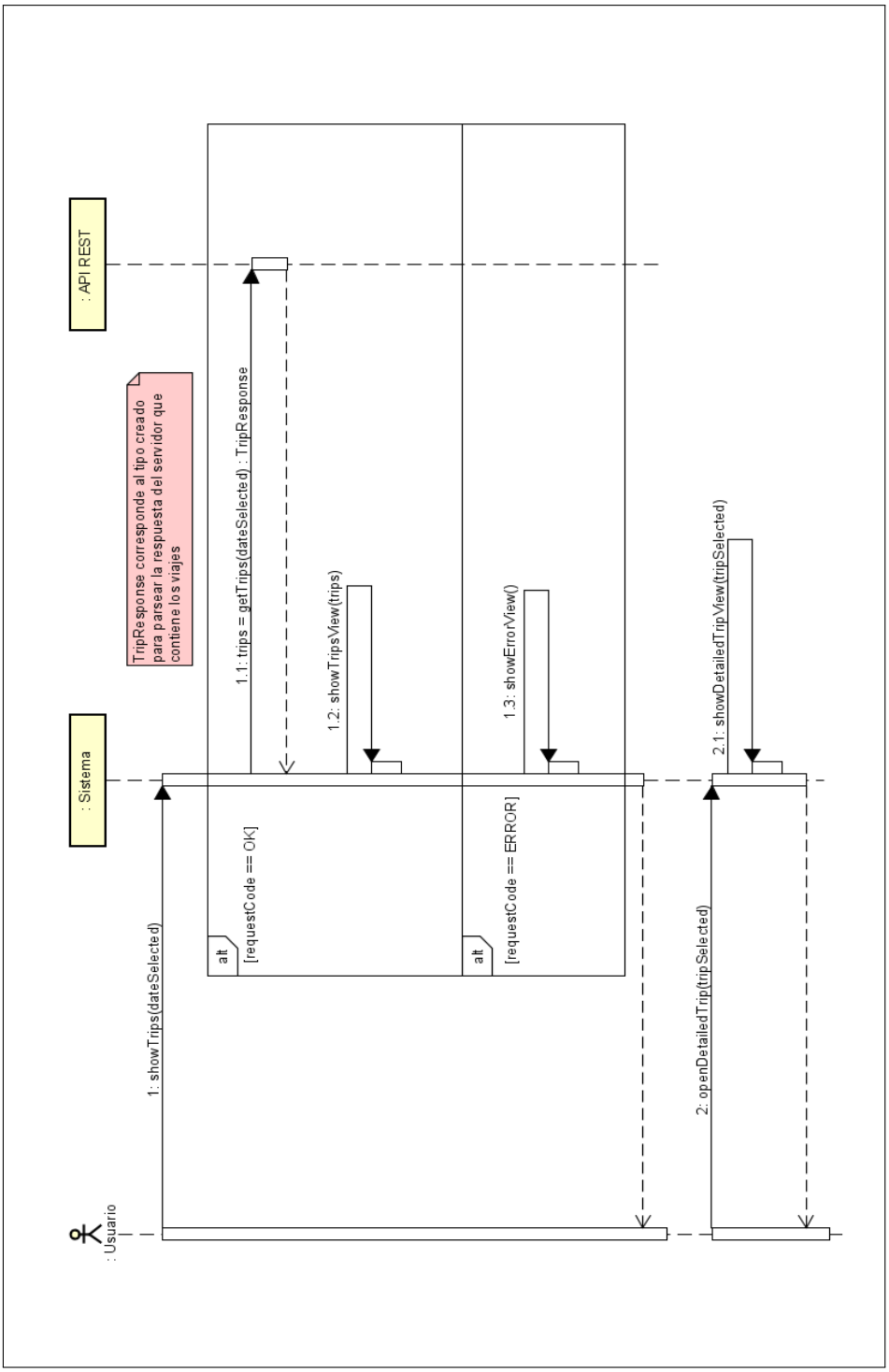

Figura 4.6: Diagrama de secuencia del caso de uso Mostrar Viajes.

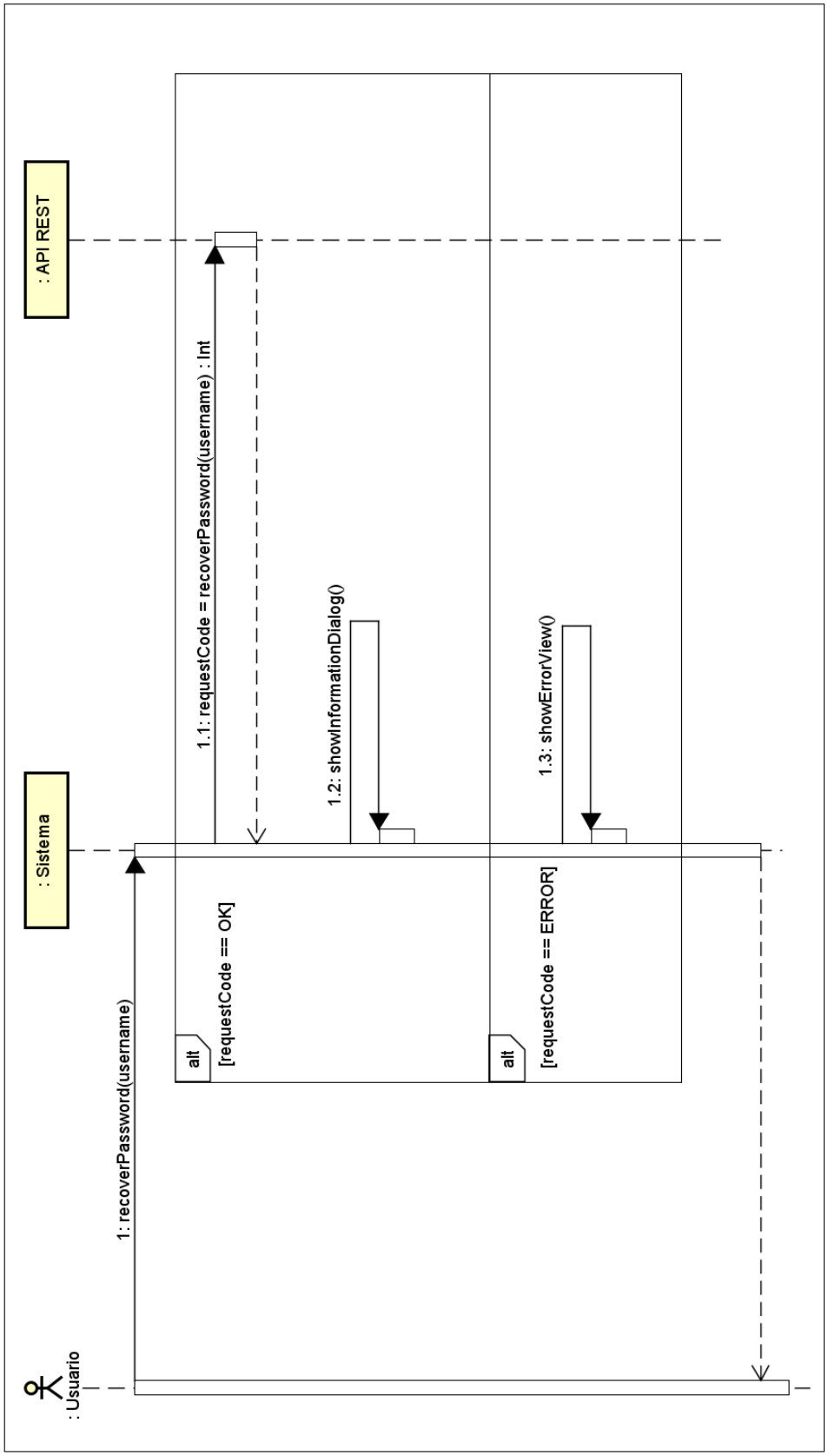

Figura 4.7: Diagrama de secuencia del caso de uso Recuperar Contraseña.

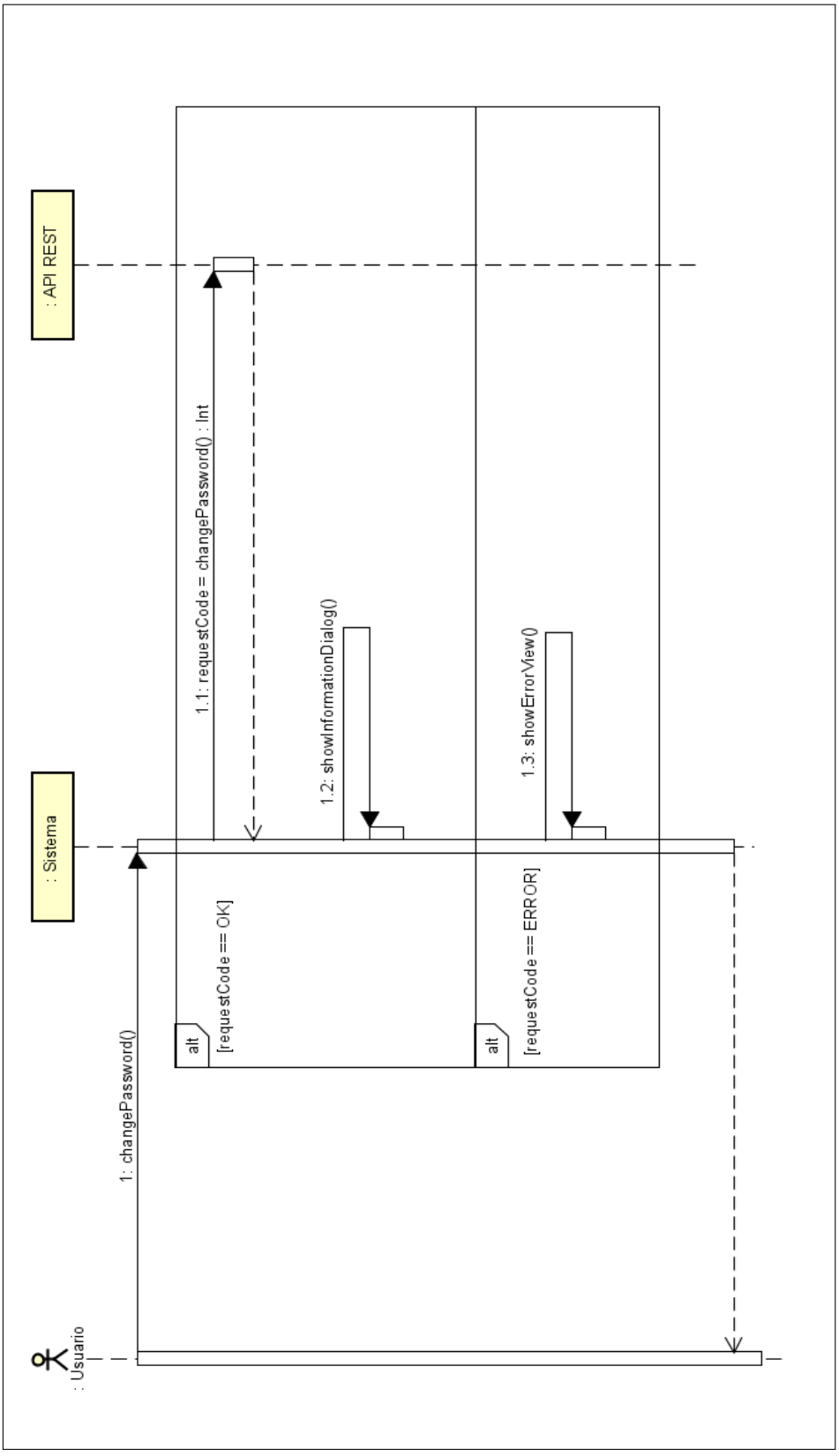

Figura 4.8: Diagrama de secuencia del caso de uso Cambiar Contraseña.

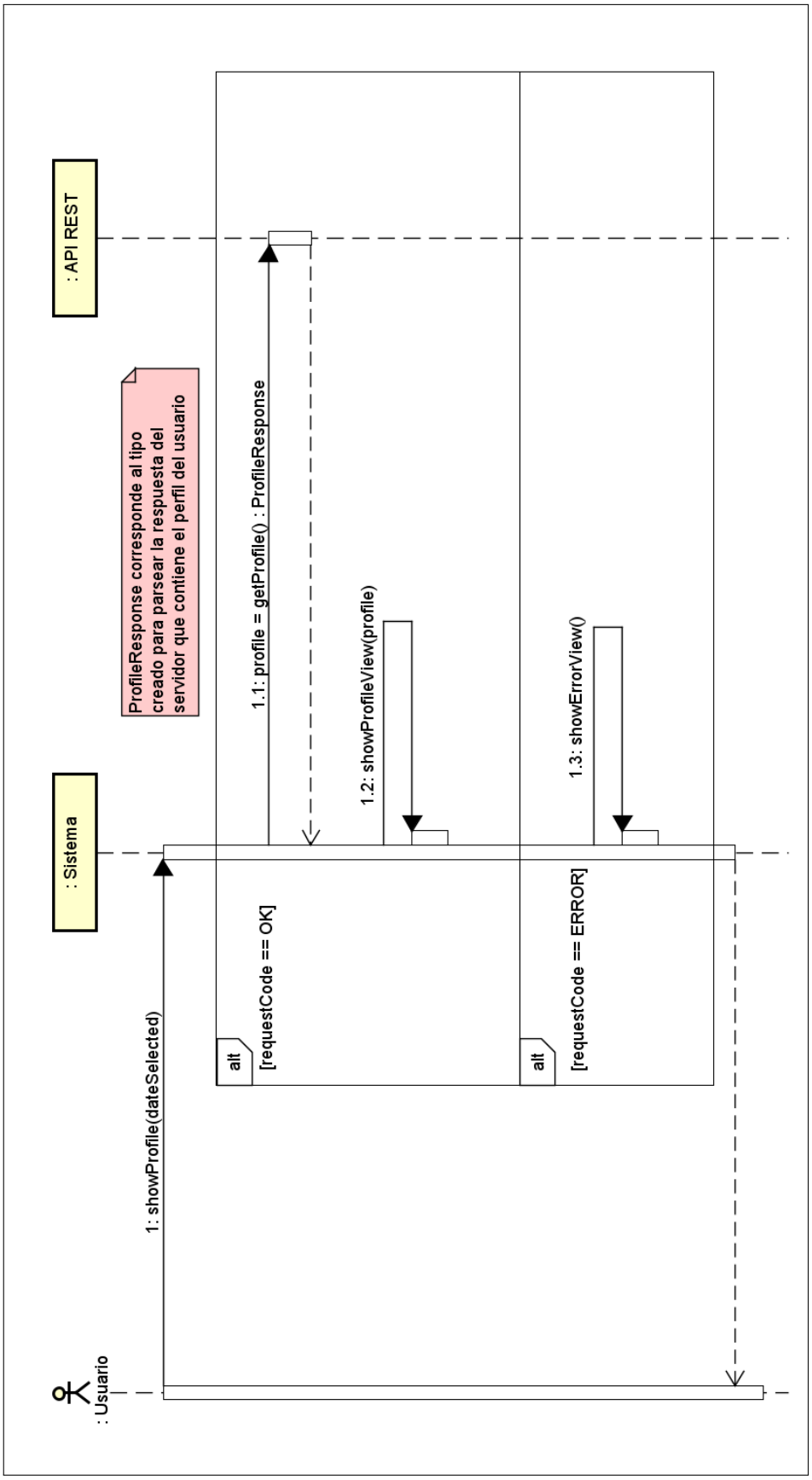

Figura 4.9: Diagrama de secuencia del caso de uso Mostrar Perfil de Usuario.

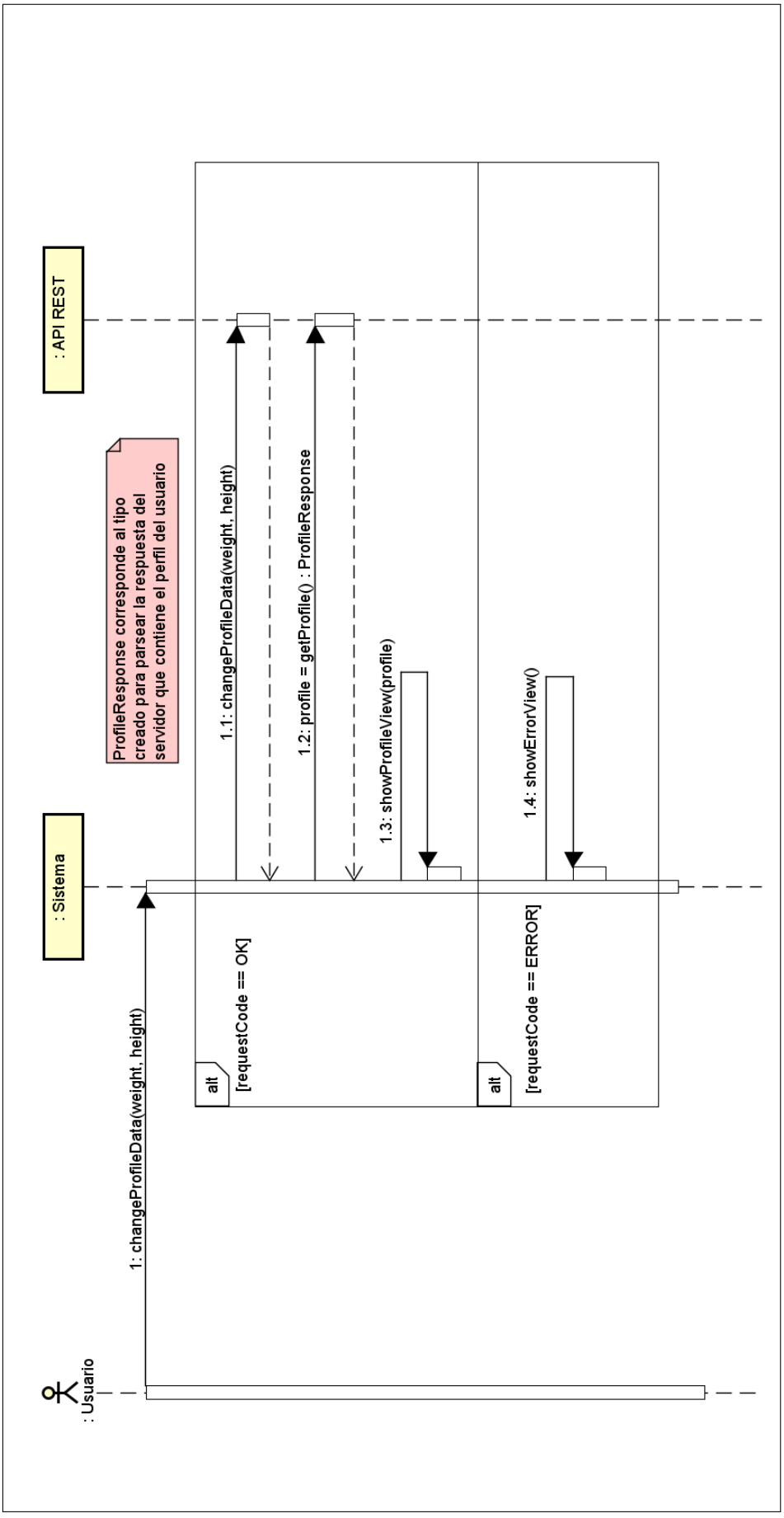

Figura 4.10: Diagrama de secuencia del caso de uso Editar Perfil de Usuario.

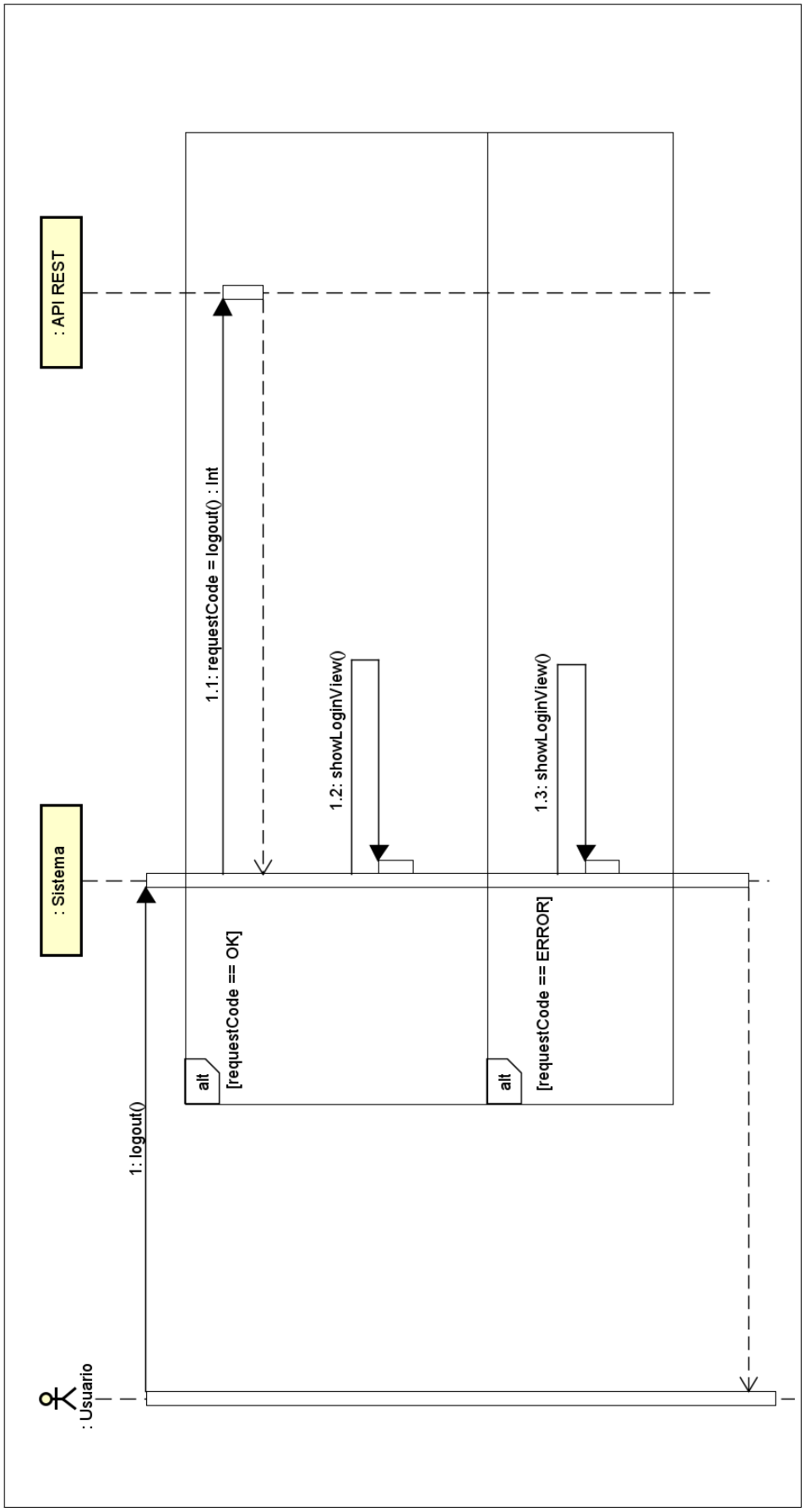

Figura 4.11: Diagrama de secuencia del caso de uso Logout.

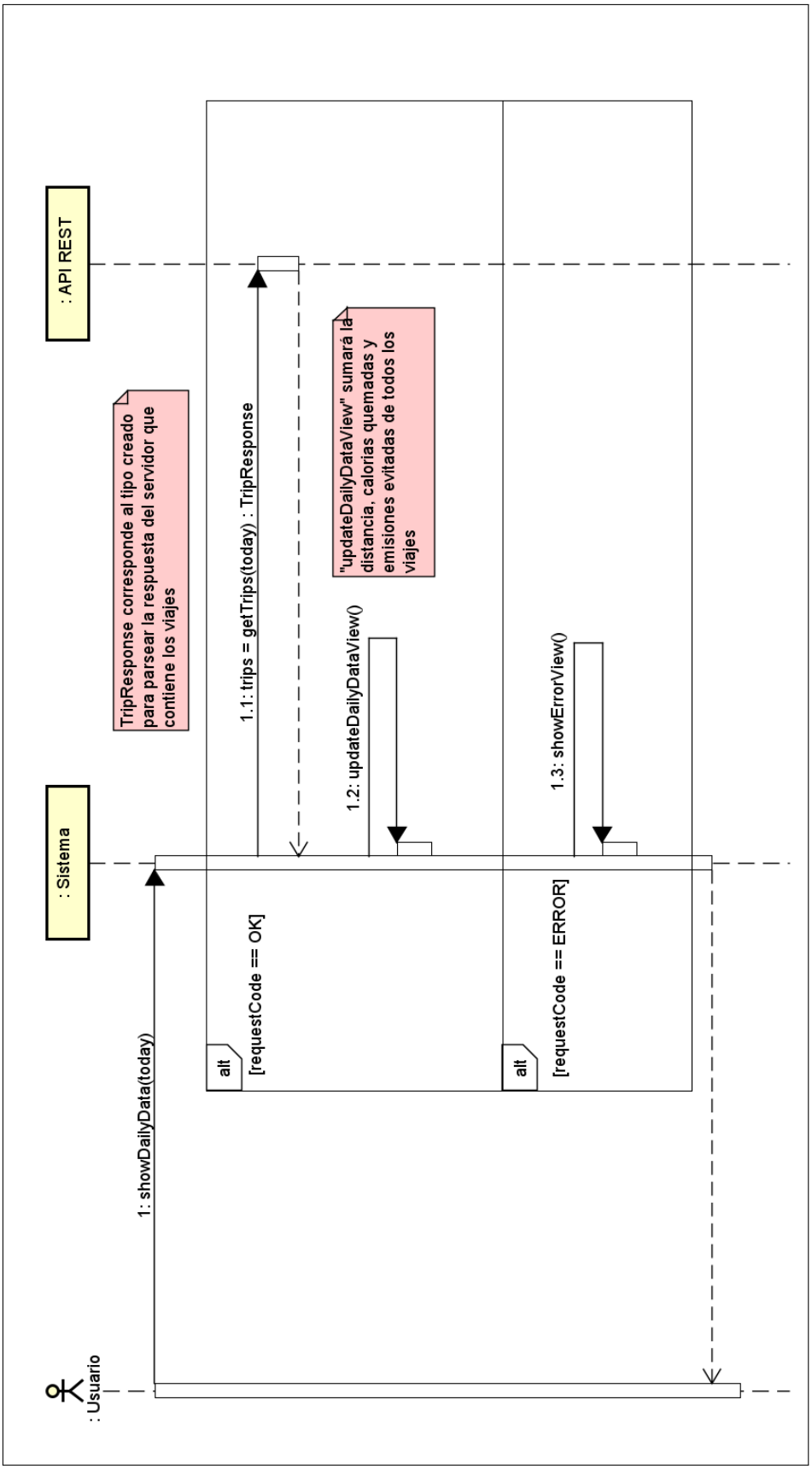

Figura 4.12: Diagrama de secuencia del caso de uso Mostrar Datos Diarios del Usuario.

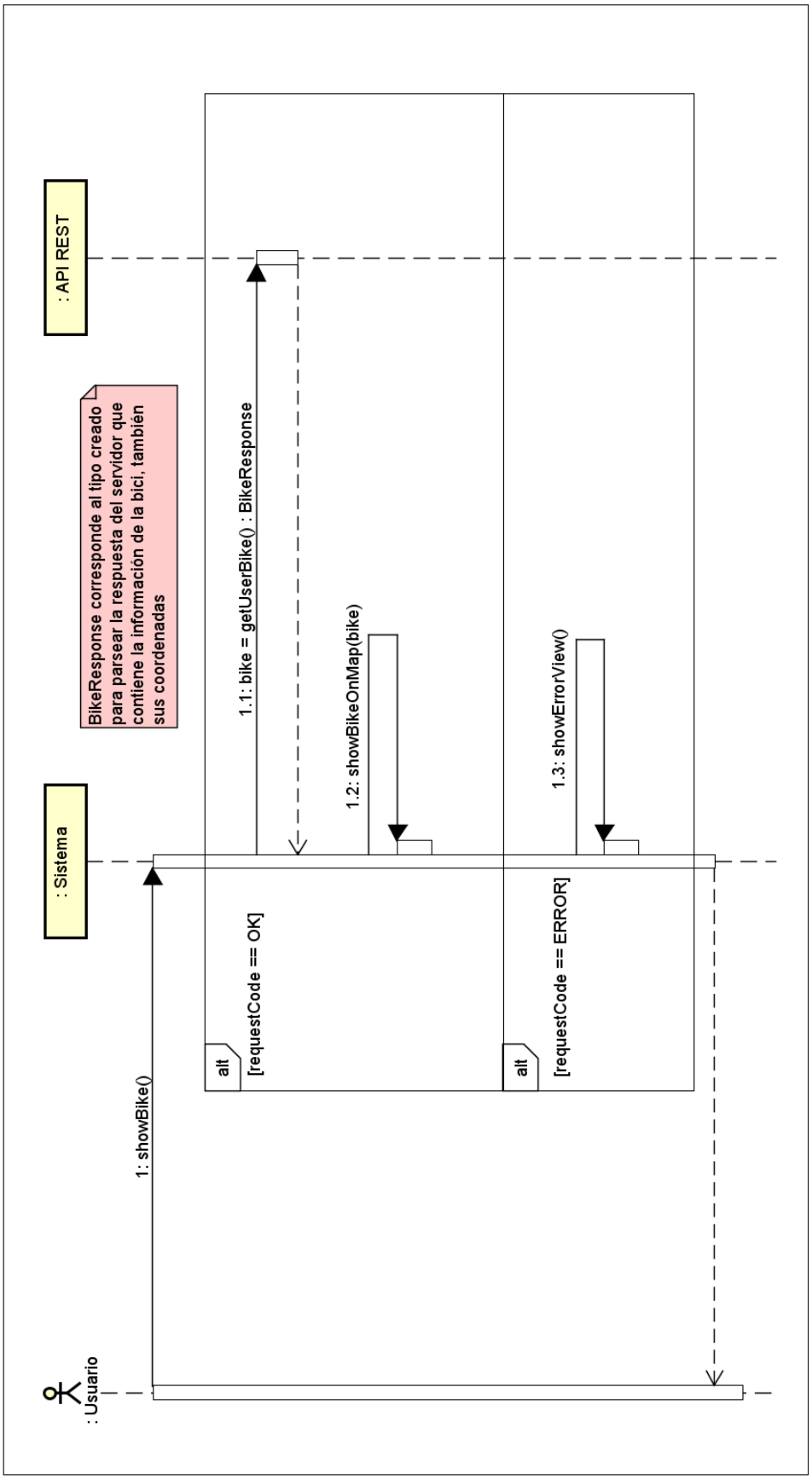

Figura 4.13: Diagrama de secuencia del caso de uso Mostrar Bicicleta en el Mapa.

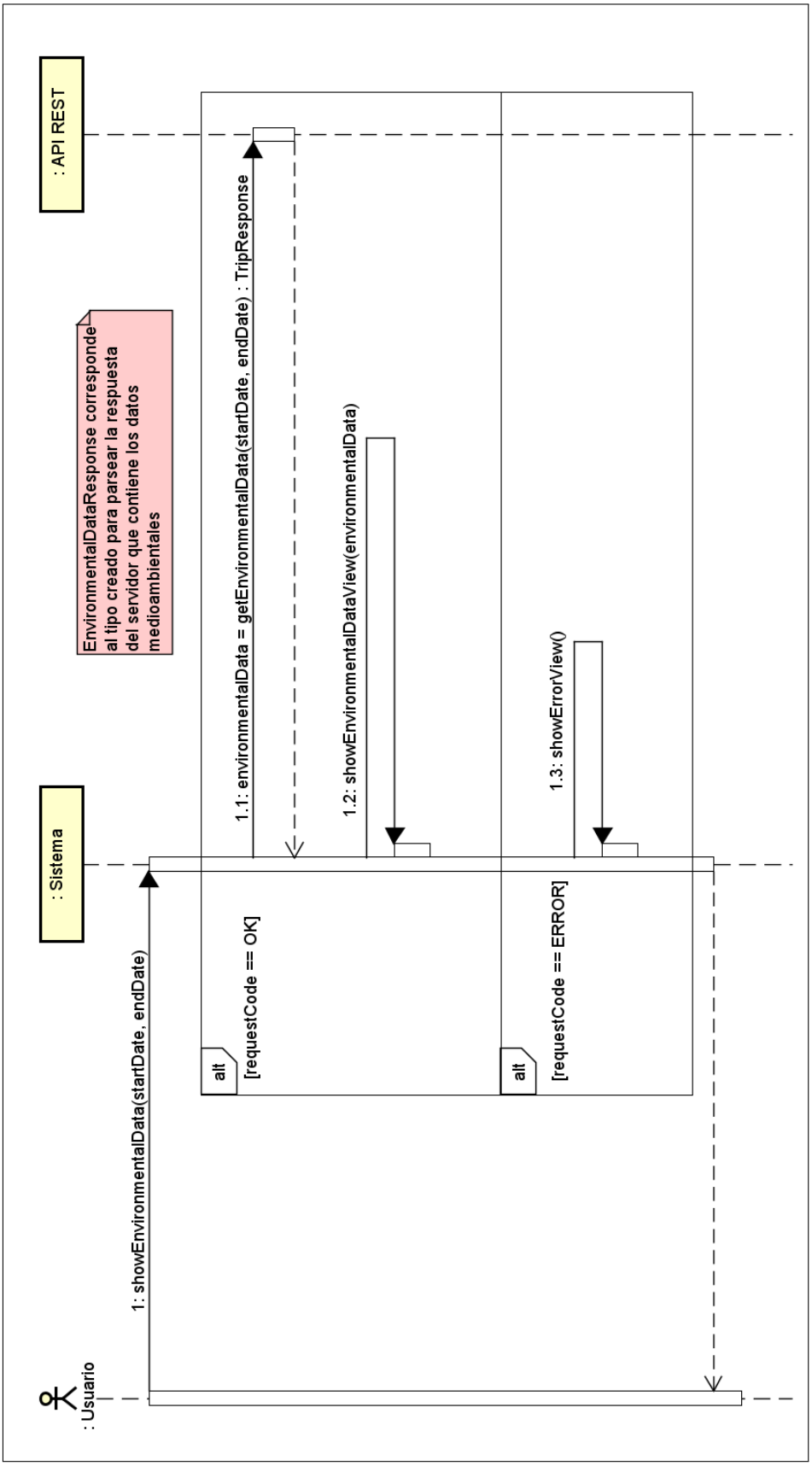

Figura 4.14: Diagrama de secuencia del caso de uso Mostrar Datos Medioambientales entre dos fechas.

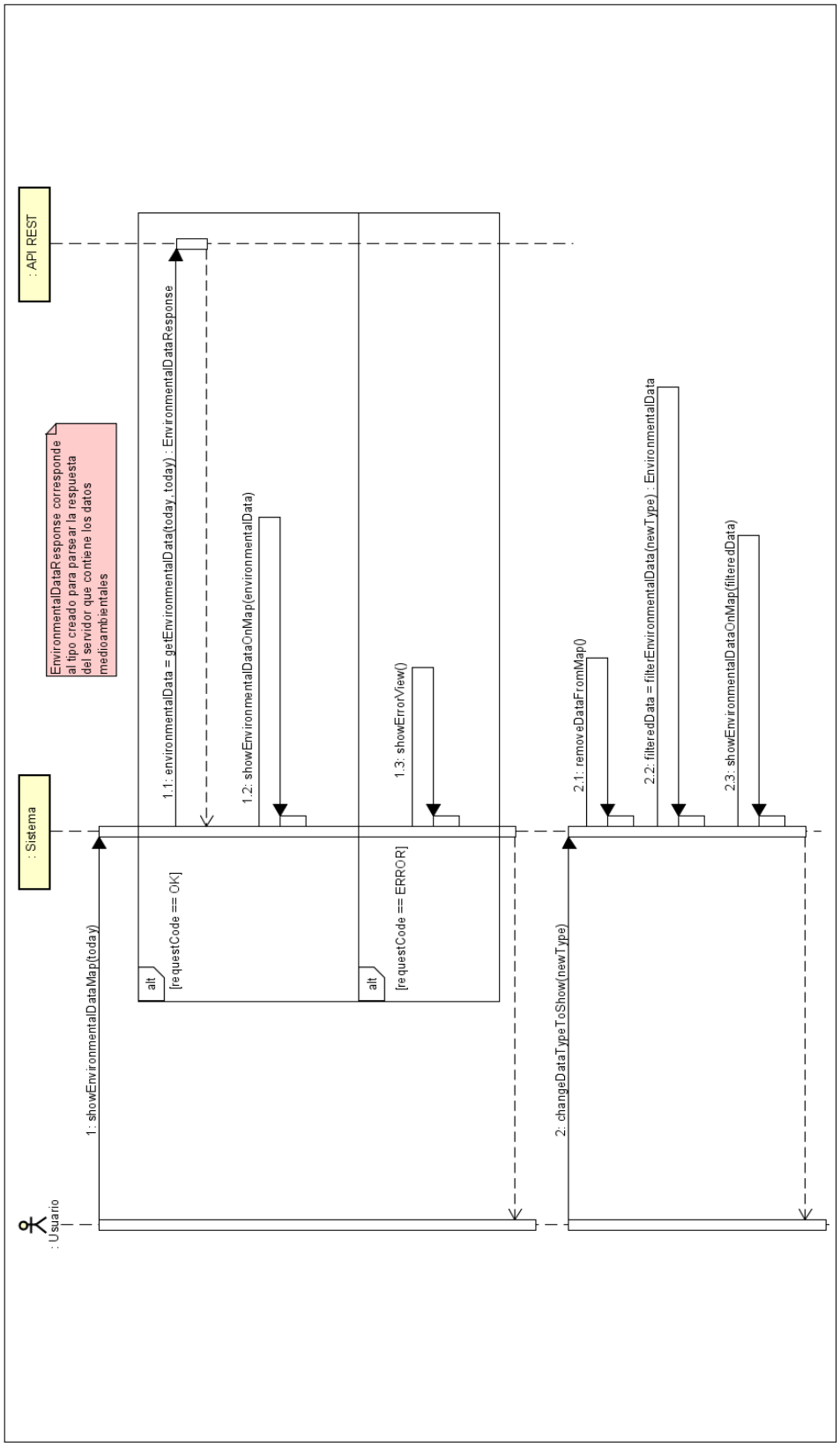

Figura 4.15: Diagrama de secuencia del caso de uso Mostrar Niveles de Contaminación en las calles.

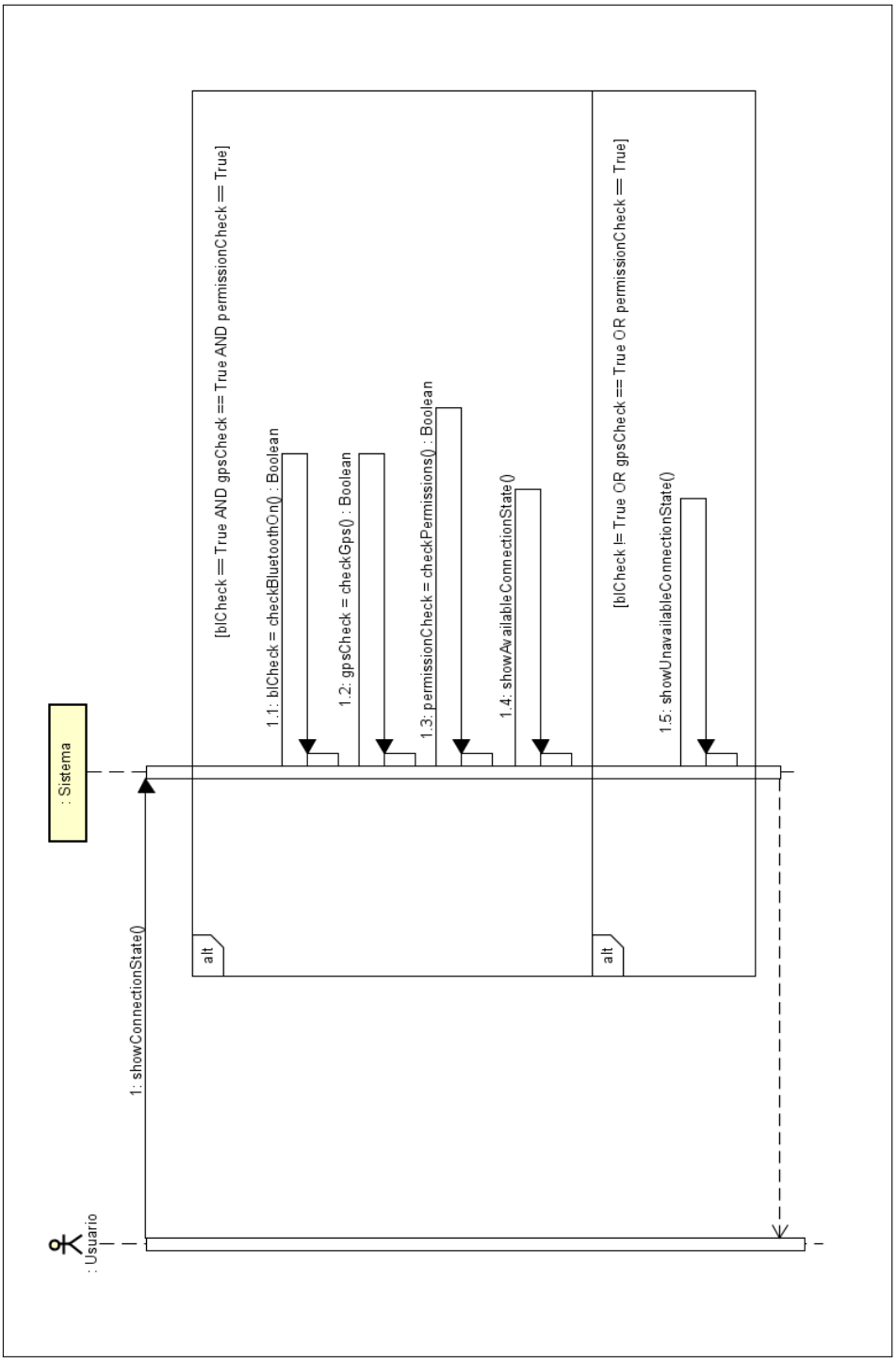

Figura 4.16: Diagrama de secuencia del caso de uso Mostrar el Estado de Conexión con el Candado.

El siguiente caso de uso 4.17 se ejecutará cada vez que la aplicación realice una petición a la API REST, recogerá el token de acceso de la base de datos y lo incluirá en la cabecera de la petición Http. Ambos tokens tienen un tiempo de expiración, el de refresco es mayor que el de acceso. Si el token de acceso esta caducado (la AIP REST devuelve `UNAUTHORIZED`) se usara el de refresco para conseguir uno nuevo de acceso, en caso de que falle también, se llevará al usuario de nuevo al Login ya que su sesión habrá caducado. Si se obtiene un token de acceso nuevo correctamente, se guarda en la base de datos y la sesión del usuario seguirá activa.

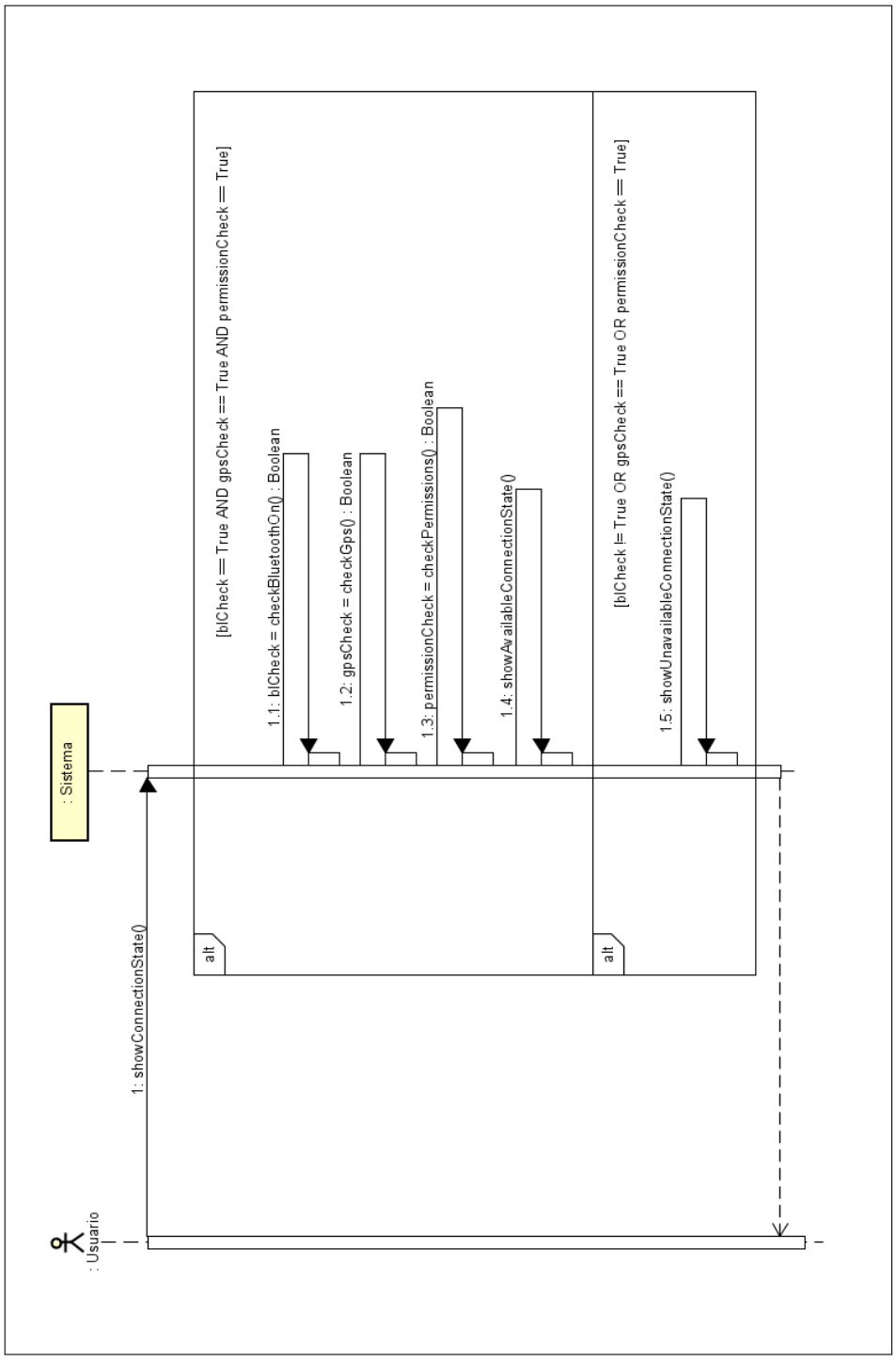

Figura 4.17: Diagrama de secuencia de la realización de cualquier petición web a la API REST exceptuando la de Login.

4.3. Modelo de dominio

La siguiente figura 4.18 muestra las clases del proyecto detectadas a nivel de análisis, puede que después surjan modificaciones en las fases de diseño e implementación.

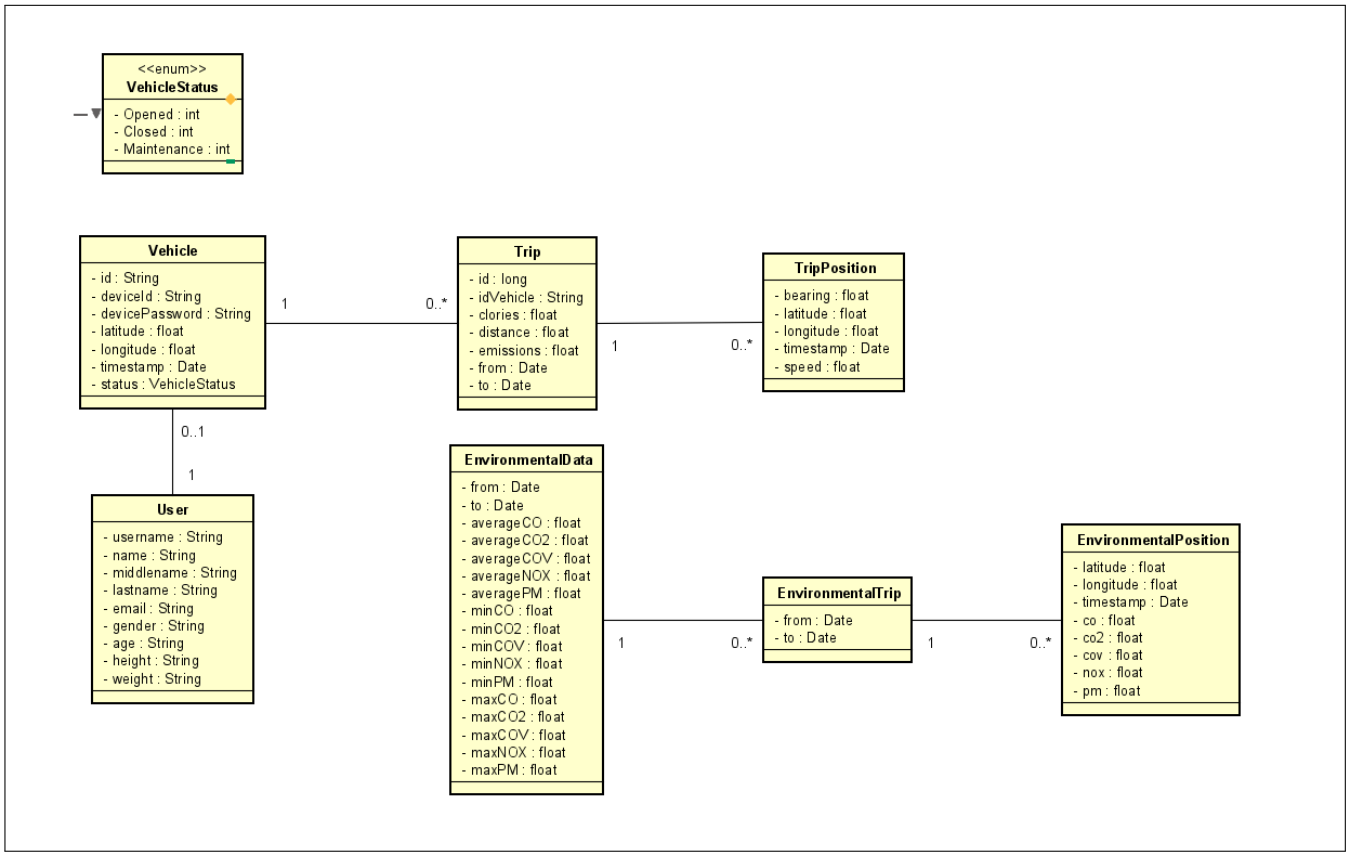

Figura 4.18: Modelo de dominio de la fase de análisis.

4.4. Modelo de datos

En el siguiente apartado se muestra el modelo de datos que describe la base de datos, junto con las entidades que lo forman.

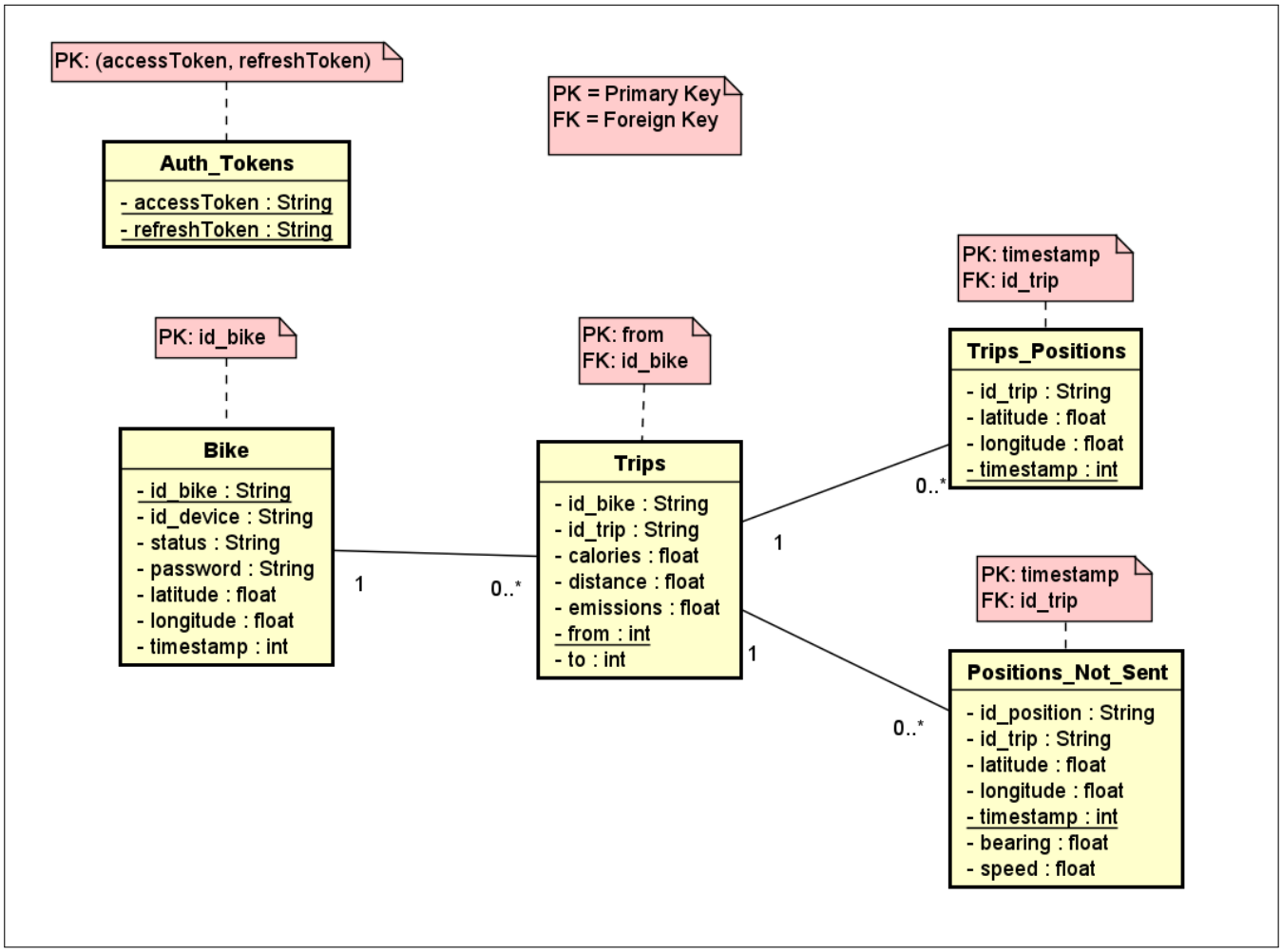

Figura 4.19: Modelo de la base de datos en la fase de análisis.

Después de haber visto este capítulo sobre el análisis de los casos de uso, el siguiente capítulo se ocupará del diseño del sistema.

Capítulo 5

Diseño

En el siguiente capítulo se detalla el modelo de diseño desarrollado para la aplicación, que, partiendo del modelo de análisis, muestra la arquitectura final de la aplicación.

5.1. Arquitectura del sistema

La arquitectura elegida para es "Clean Architecture" [26], este tipo de arquitectura permite separar el código de la aplicación en diferentes módulos o capas. Estos capas tienen restricciones para comunicarse con las demás capas, creando así una única dirección en el flujo del programa.

5.1.1. Capas

- **Presentación:**
  Esta capa incluye todo lo relacionado con las vistas y elementos gráficos de la aplicación y las acciones que el usuario puede realizadas sobre ellas. Para implementar esta capa se ha utilizado un patrón MVVM (modelo, vista, modelo de vista), es muy similar al MVC (modelo, vista, controlador), está formado por:

  - El modelo, al igual que en MVC, representa la capa de datos y la lógica de negocio.
  - La vista presenta la información al usuario y reacciona a a cambios en el modelo, de forma similar a un patrón MVC activo.
  - El modelo de vista es un actor intermediario entre el modelo y la vista y contiene toda la lógica de presentación.

  La vista en MVC depende del modelo y del controlador para obtener datos o modificarlos. Si hay un cambio en la vista que implique un cambio en el modelo necesita acceder al controlador y si hay datos nuevos, depende del modelo para mostrarlos. En el caso del MVVM, la vista solo depende del modelo de la vista para obtener datos o modificarlos, además nadie tiene que decirle que actualice los datos como haría el controlador en MVC pasivo, ya que está observando cambios en el modelo directamente.

  El modelo en MVC puede tener demasiadas responsabilidades como obtener los datos de fuentes de datos, informar al controlador sobre cambios en esos datos y prepararlos para que se puedan mostrar en la vista. En MVVM el modelo queda totalmente desacoplado de la vista y solo contiene información, nunca acciones para manipularla.

  En MVC no queda muy claro quién se debería encargar de la lógica de presentación ya que el modelo solo debería enviar datos a la vista y por otro lado, llevar esta lógica a la vista complicaría algunos test. En MVVM, esta responsabilidad es muy clara y cae en manos del modelo de la vista.

También hay que aclarar que en la programación de esta aplicación móvil se utilizará un único "Activity" como coordinador de vistas y su función será únicamente configurarlas y mostrarlas, es decir, cada vista será un "Fragment" y serán gestionados por el "MainActivity". Cada "Fragment" tendrá su propio modelo de vista con su lógica encapsulada en el.

Esta capa solo se comunicará con la capa de Casos de Uso.

- **Casos de Uso:**
  Incluirá toda la lógica necesaria para ejecutar los casos de uso previamente definidos, esta capa recibe las acciones que el usuario puede desencadenar. Estas pueden ser acciones activas (el usuario hace clic en un botón) o acciones implícitas (la aplicación navega a una pantalla). A partir de este punto, se pueden ejecutar todas las demás acciones necesarias un hilo secundario lo que permite evitar el bloqueo de la interfaz.

  Esta capa solo se relaciona con la capa de Datos.

- **Dominio:**
  Contiene la definición de las reglas de negocio en forma de interfaces para poder ejecutar las acciones de los casos de uso recibidas desde la capa de Casos de Uso.

- **Datos:**
  Corresponde a la implementación de las reglas de negocio previamente definidas en la capa de Dominio. La mayor parte de la lógica va a consistir simplemente en la solicitud y la persistencia de datos.

  Esta capa solo se relaciona con la capa de Repositorio.

- **Repositorio:**
  Define la comunicación de la aplicación con las fuentes de datos mediante la implementación de un patrón repositorio. Para una determinada solicitud, es capaz de decidir dónde encontrar la información, ya se de forma local, como la base de datos del móvil o remota como el servicio de la API REST.

  Esta capa solo se relaciona con las capas Local y Remoto.

- **Local:**
  Implementa toda la gestión de comunicación con la base de datos (usaremos Room [13] en Android) y las operaciones CRUD para poder guardar y recuperar datos desde la base de datos del dispositivo móvil.

- **Remoto:**
  Implementa toda la gestión de la conexión con la API REST para obtener y enviar los datos necesarios.

  Tanto la capa Local como la capa Remoto implementan el patrón DAO para acceder a los datos, la capa Local accede a la base de datos y la capa Remoto a la API REST, el patrón DAO propone encapsular completamente la lógica para acceder a los datos, de esta forma, el DAO proporcionará los métodos necesarios para insertar, actualizar, borrar y consultar la información.

5.2. Diagramas de paquetes en diseño

Hay que aclarar que el lenguaje de programación utilizado sera Kotlin por lo que aparecen nuevos tipos de datos:

- **Instant**
  Representa un punto instantáneo en el tiempo. Esta clase modela un único punto instantáneo en la línea de tiempo. Esto podría usarse para registrar marcas de tiempo de eventos en la aplicación. El rango de un instante requiere el almacenamiento de un número mayor que un long. Para lograr esto, la clase almacena una representación de segundos de época y una representación de nanosegundos que siempre estará entre 0 y 999,999,999. Los segundos de época se miden a partir de la época estándar de Java 1970-01-01T00:00:00Z en la que los instantes posteriores a la época tienen valores positivos y los instantes anteriores tienen valores negativos. Tanto para la parte de época-segundo como para la de nanosegundos, un valor mayor siempre es posterior en la línea de tiempo que un valor menor. [12]

- **Unit**
  Este tipo tiene un solo valor: el objeto Unidad. Corresponde al tipo void en Java. [22]

- **Either**
  Este tipo se define como "Either<A, B>" donde A y B son dos tipos que se pasan como parámetros, es decir, se crea un objeto que puede ser de tipo A o de tipo B. Es muy util en las funciones que pueden fallar o producir un resultado. Por ejemplo si tenemos una funcion que divide dos números, podemos obtener un resultado o un error si dividimos por cero, podríamos definir el tipo de retorno de la función como "Either<Exception, Float>". Después analizando el tipo del valor de retorno podríamos mostrar el resultado o un mensaje de error. [28]

- **Option**
  Este tipo se define como "Option<A>", se crea un objeto que puede ser tipo A o "None". Es muy útil en funciones que pueden fallar pero que su tipo de retorno es void, de esta manera se devolvería un Option<Exception> si falla y "None" si la operación ha tenido éxito. Es equivalente a definir "Either<Exception, Unit>" [29]

- **Generic**
  Se ha utilizado en los diagramas el tipo "Generic", esto no es más que una forma de representar un parámetro de cualquier tipo soportado por el sistema. No es ninguna clase ni es un tipo de objetos con atributos ni operaciones propias.

- **Response**
  Es el tipo utilizado para gestionar las respuestas recibidas de la API REST. [34]

Una vez explicados estos tipos de datos nuevos, podemos observar los diagramas de paquetes que organizan las clases, capas y módulos de la aplicación. Algunas capas se han reorganizado en otros paquetes respecto a la fase de análisis para facilitar la localización del código en la aplicación.

Después de observar el diagrama general de la aplicación vamos a profundizar en cada uno de los paquetes y subpaquetes, comenzaremos por el paquete "App".

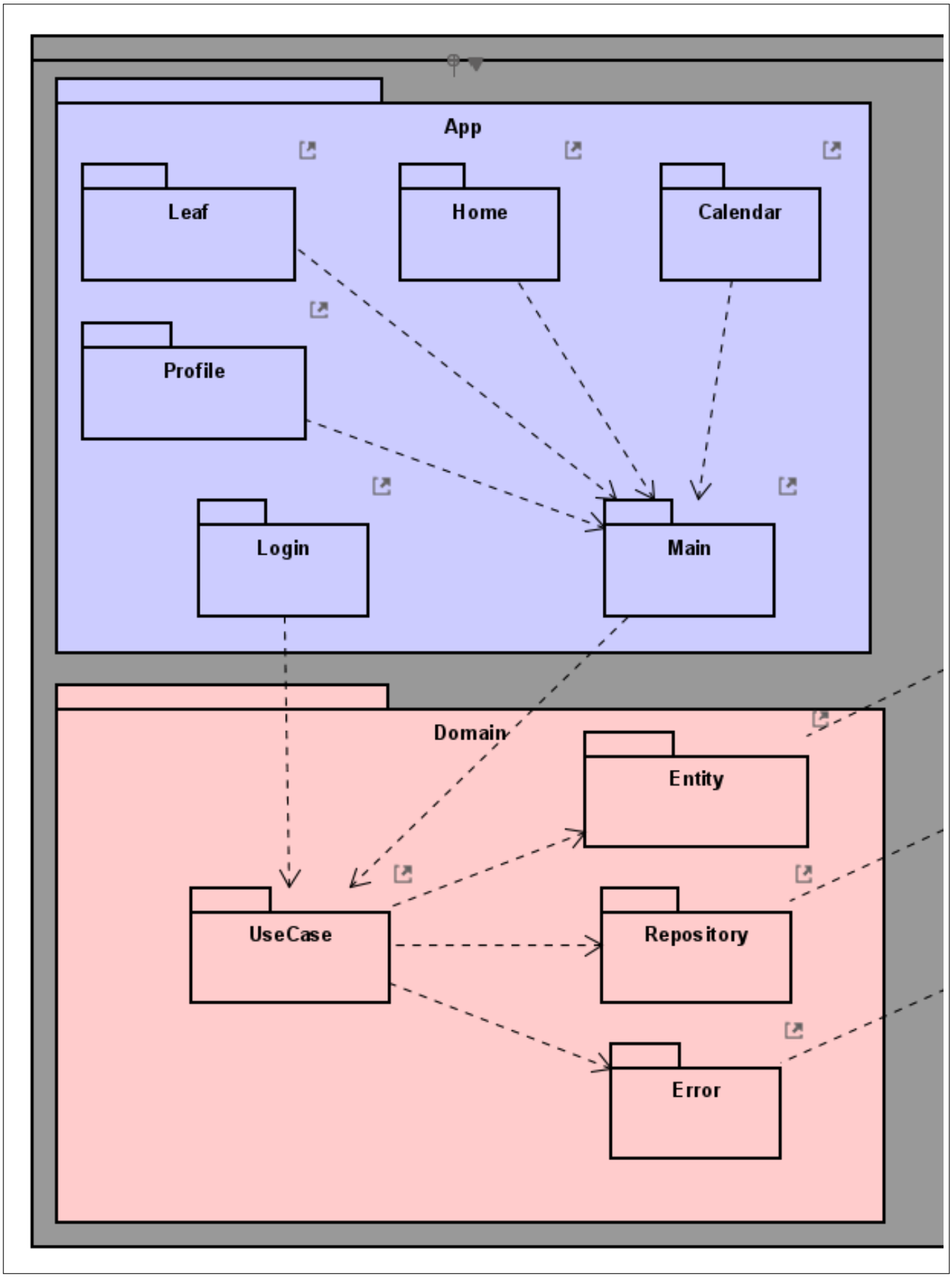

Figura 5.1: Modelo de paquetes general de la fase de diseño parte 1.

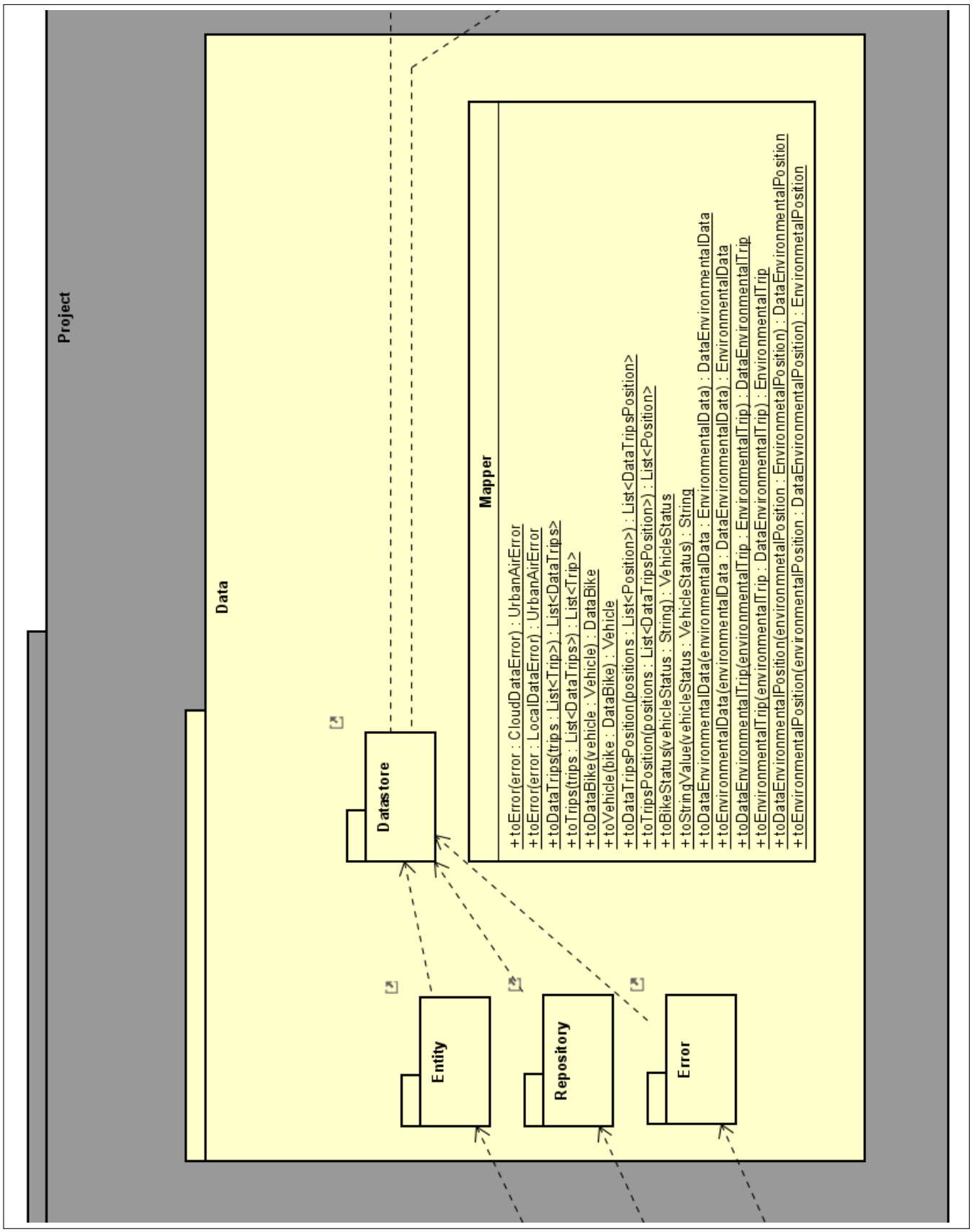

Figura 5.2: Modelo de paquetes general de la fase de diseño parte 2.

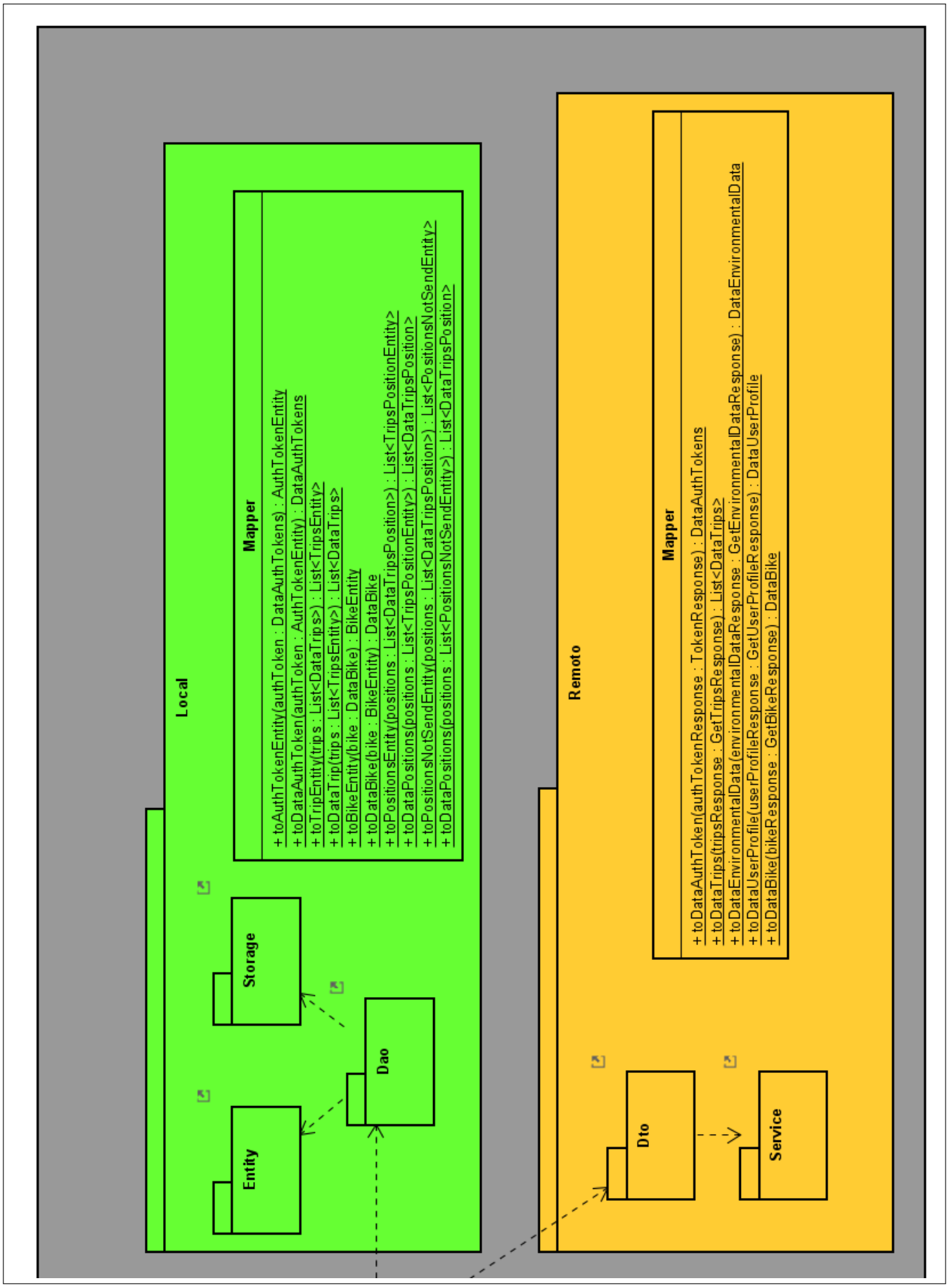

Figura 5.3: Modelo de paquetes general de la fase de diseño parte 3.

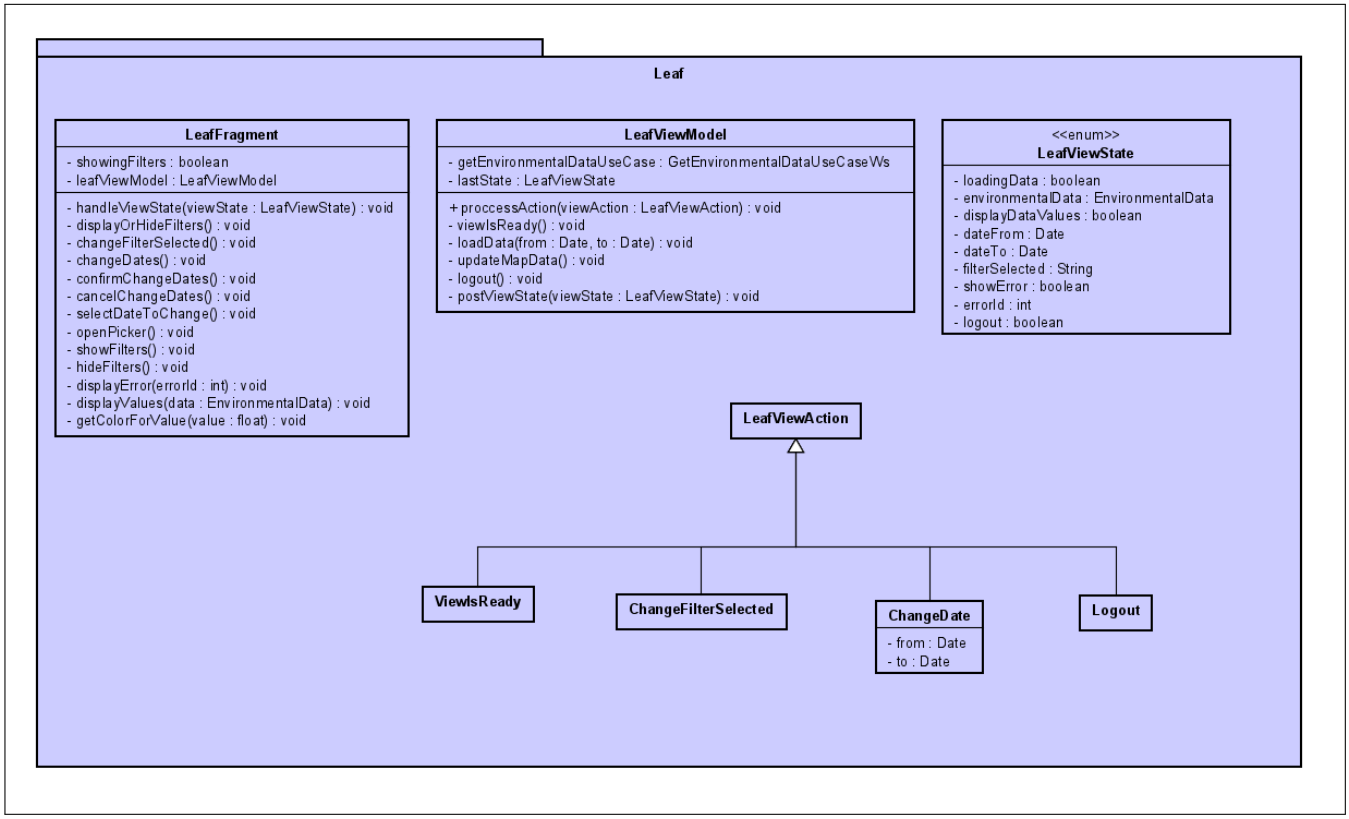

Figura 5.4: Modelo de paquetes del paquete "Leaf" en la fase de diseño.

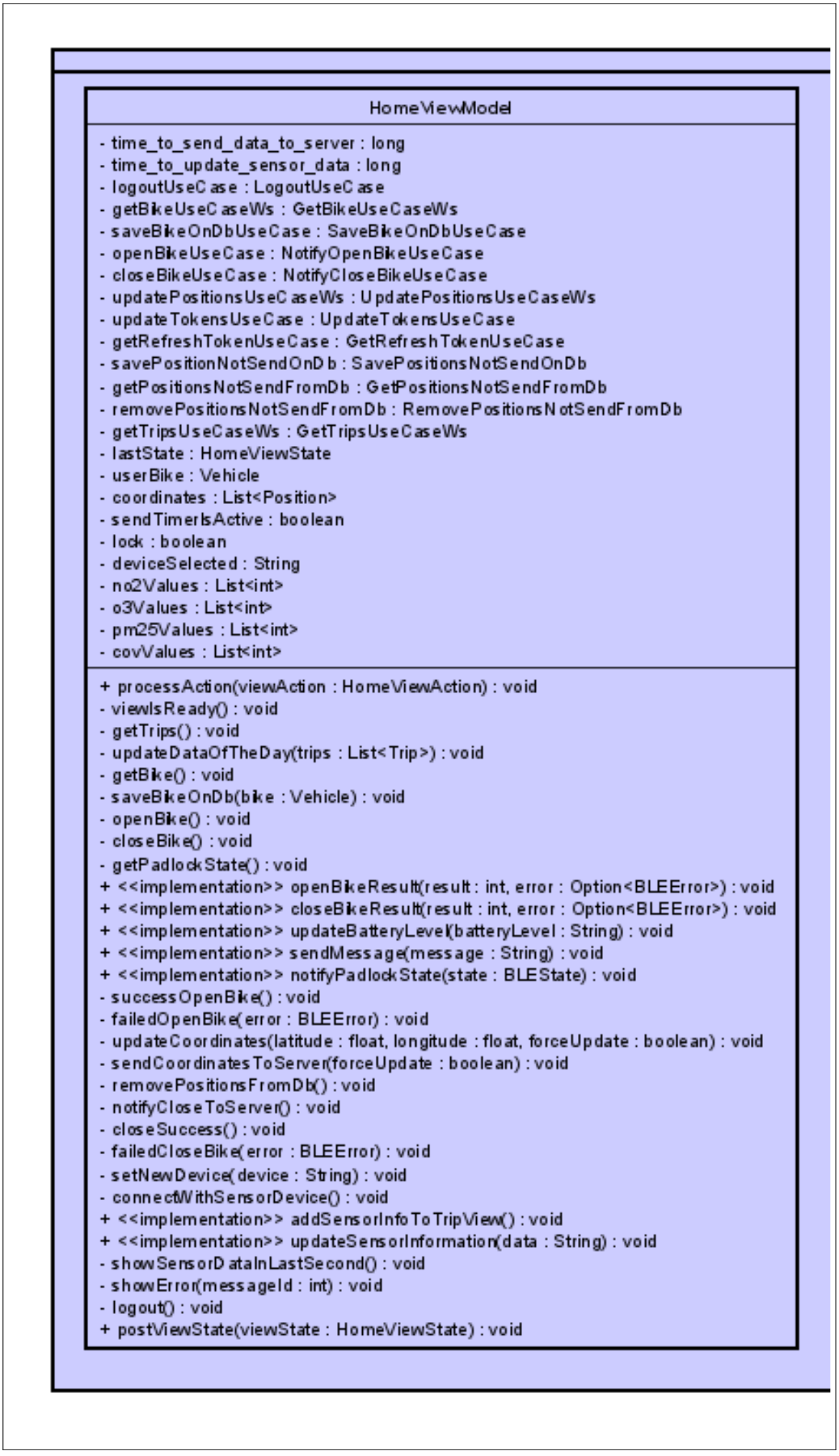

Figura 5.5: Modelo de paquetes del paquete "Home" en la fase de diseño parte 1.

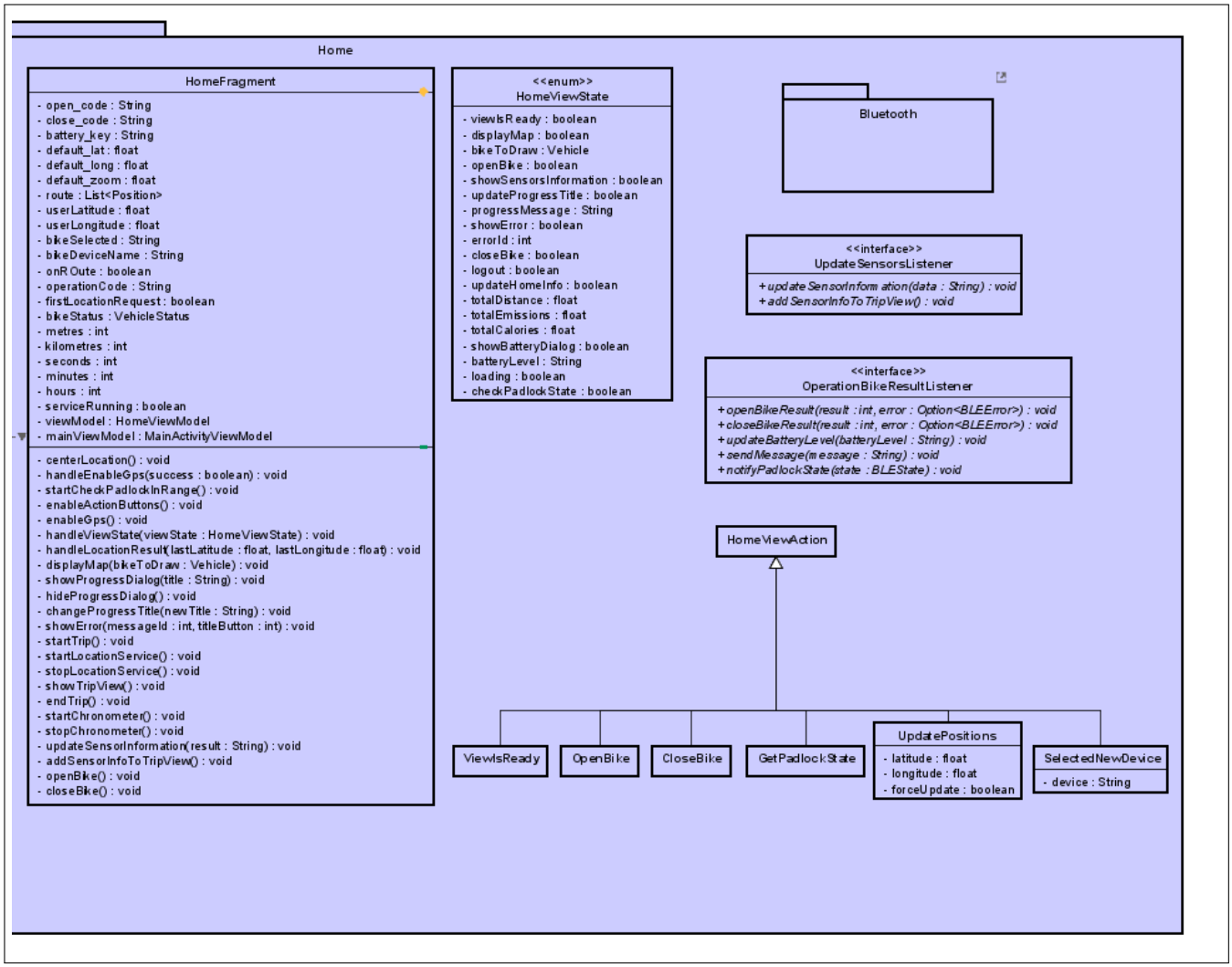

Figura 5.6: Modelo de paquetes del paquete "Home" en la fase de diseño parte 2.

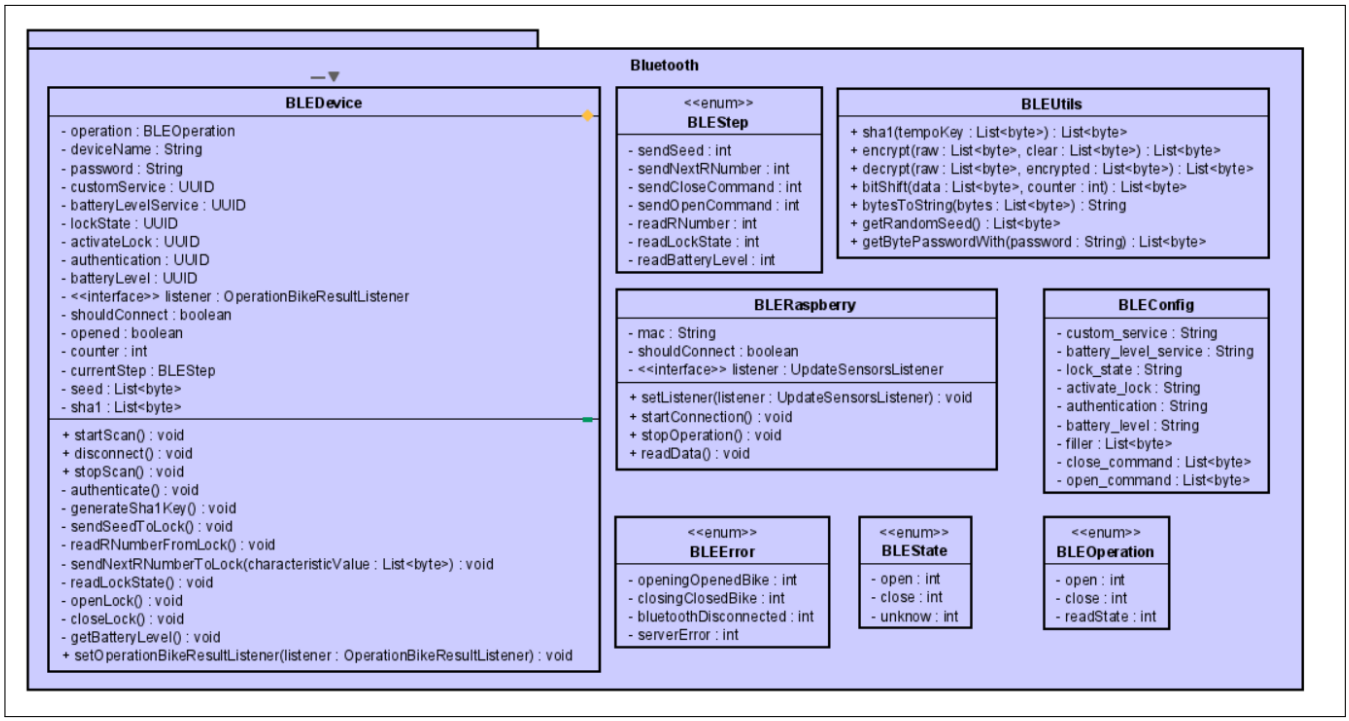

Figura 5.7: Modelo de paquetes del paquete "Bluetooth" en la fase de diseño.

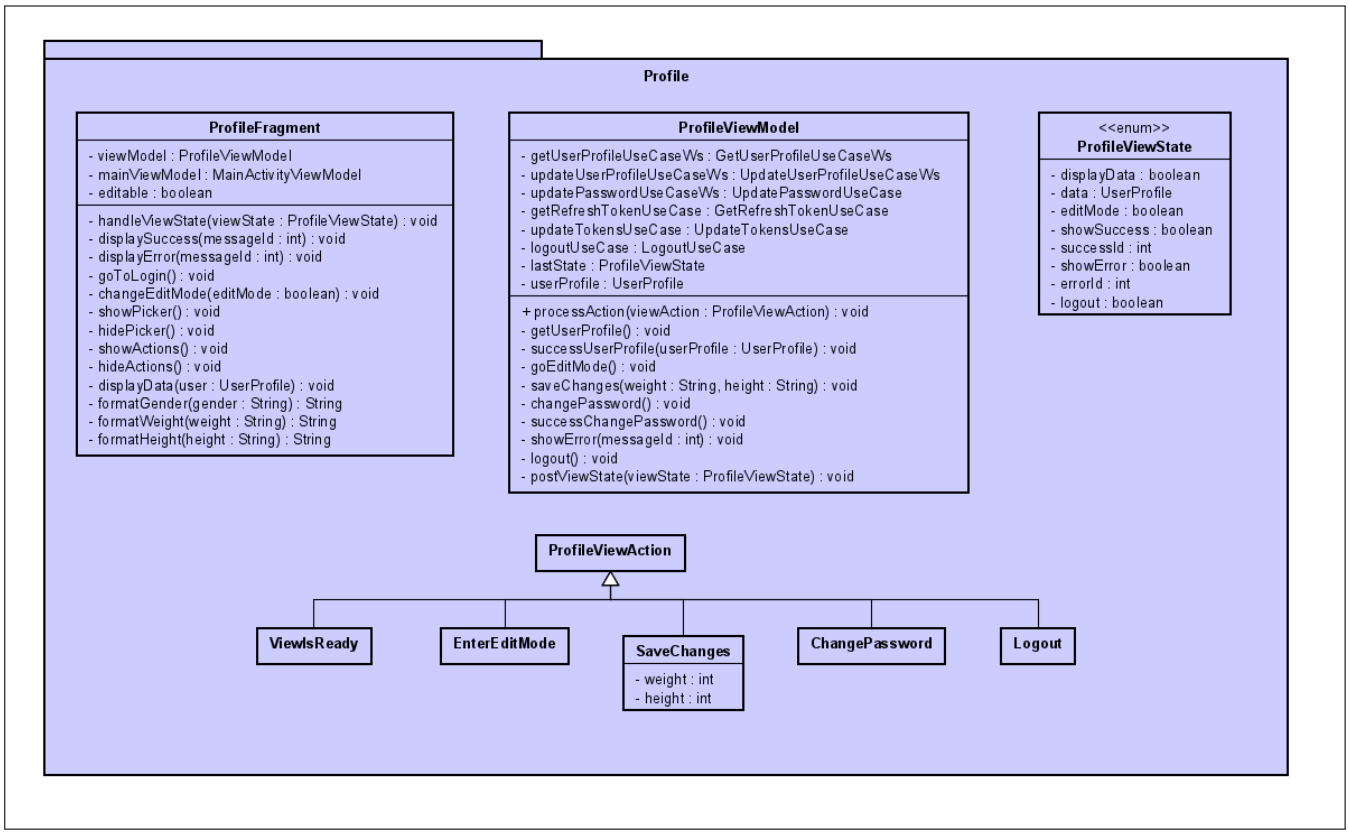

Figura 5.8: Modelo de paquetes del paquete "Profile" en la fase de diseño.

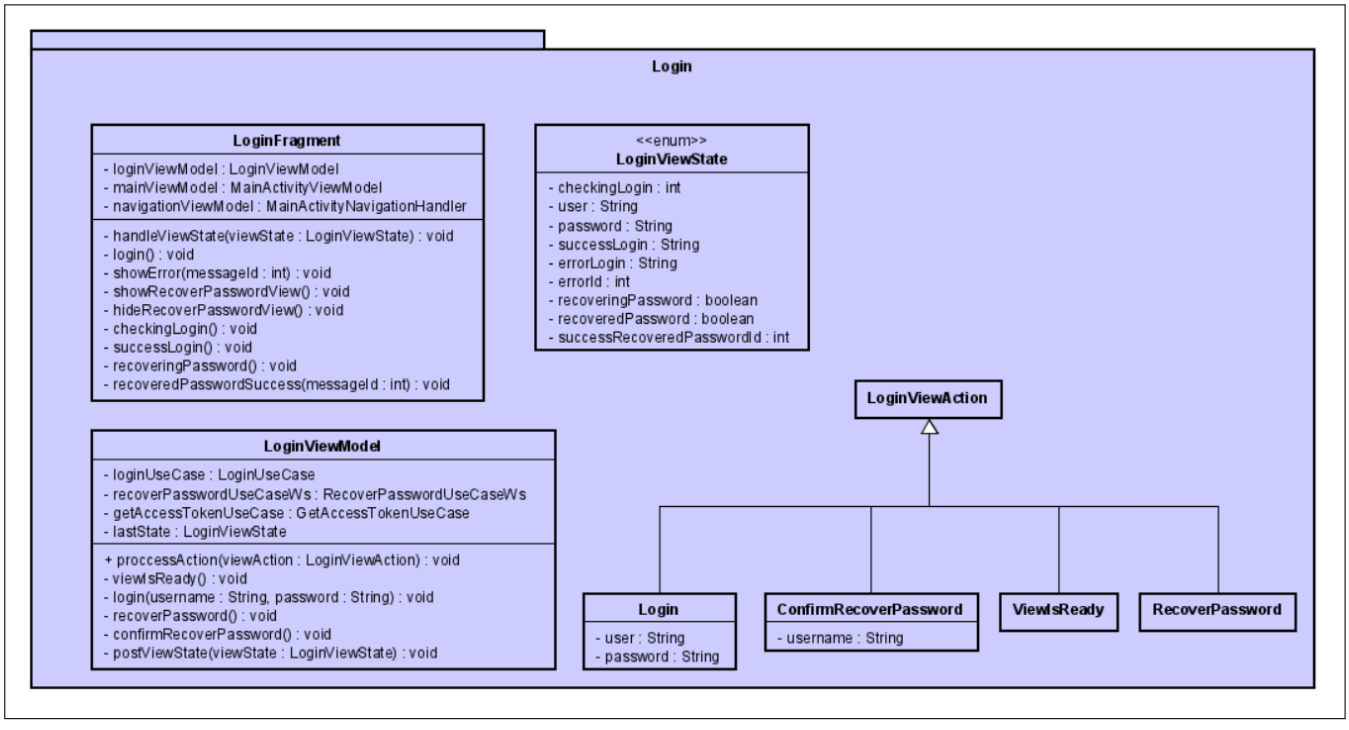

Figura 5.9: Modelo de paquetes del paquete "Login" en la fase de diseño.

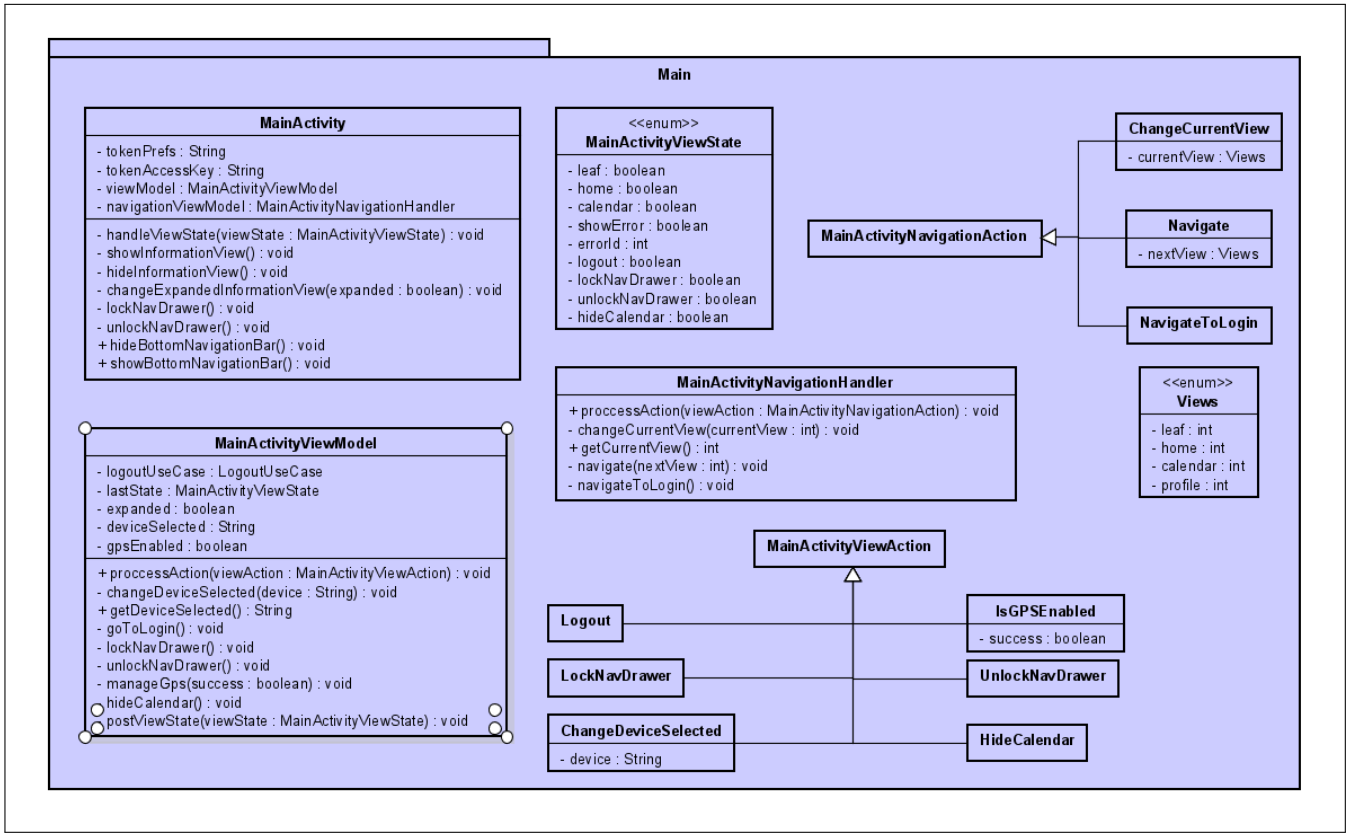

Figura 5.10: Modelo de paquetes del paquete "Main" en la fase de diseño.

A continuación se detallan los paquetes del paquete "Domain"

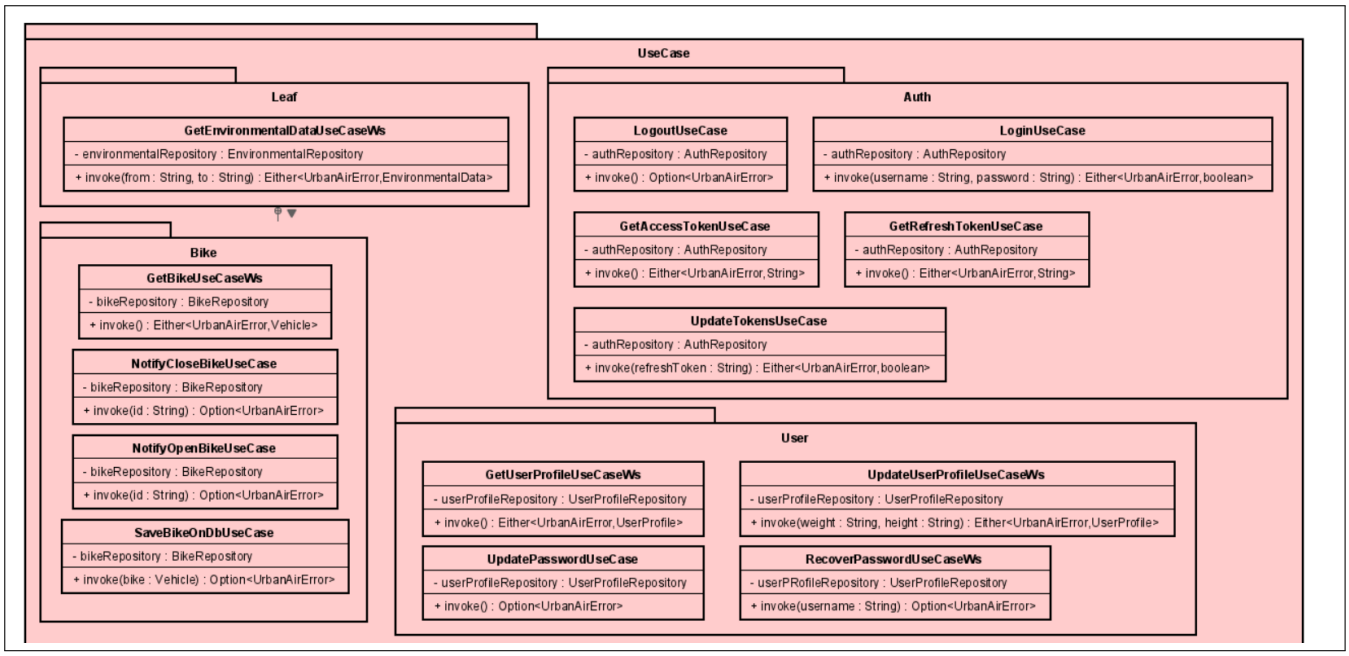

Figura 5.11: Modelo de paquetes del paquete "UseCase" en la fase de diseño parte 1.

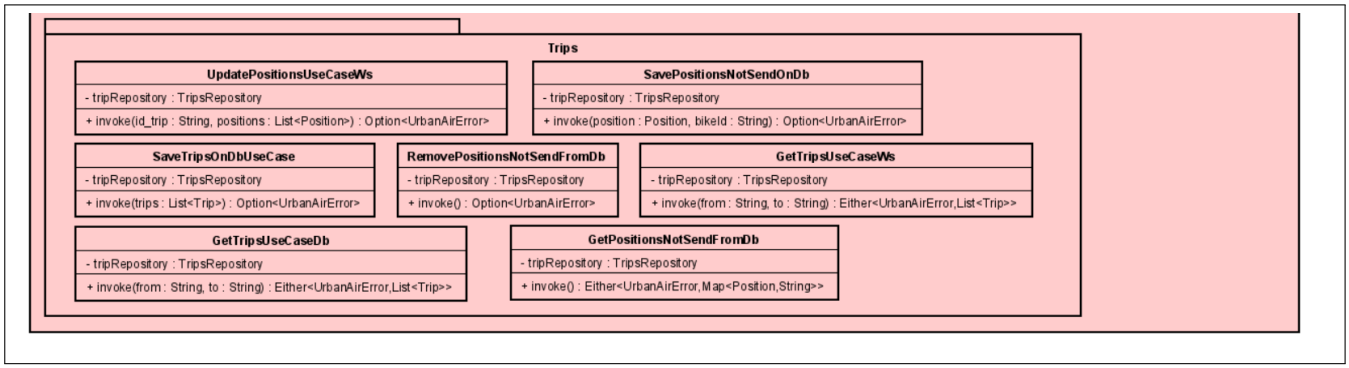

Figura 5.12: Modelo de paquetes del paquete "UseCase" en la fase de diseño parte 2.

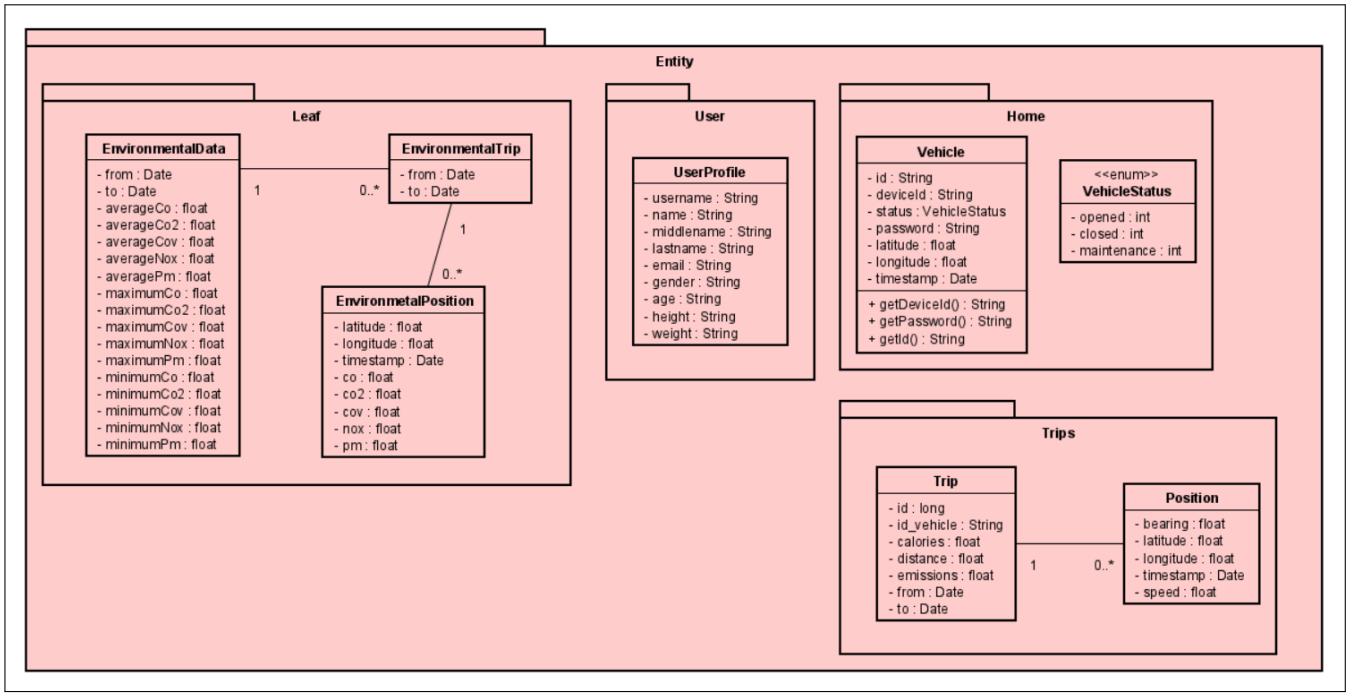

Figura 5.13: Modelo de paquetes del paquete "Entity" de la capa "Domain" en la fase de diseño.

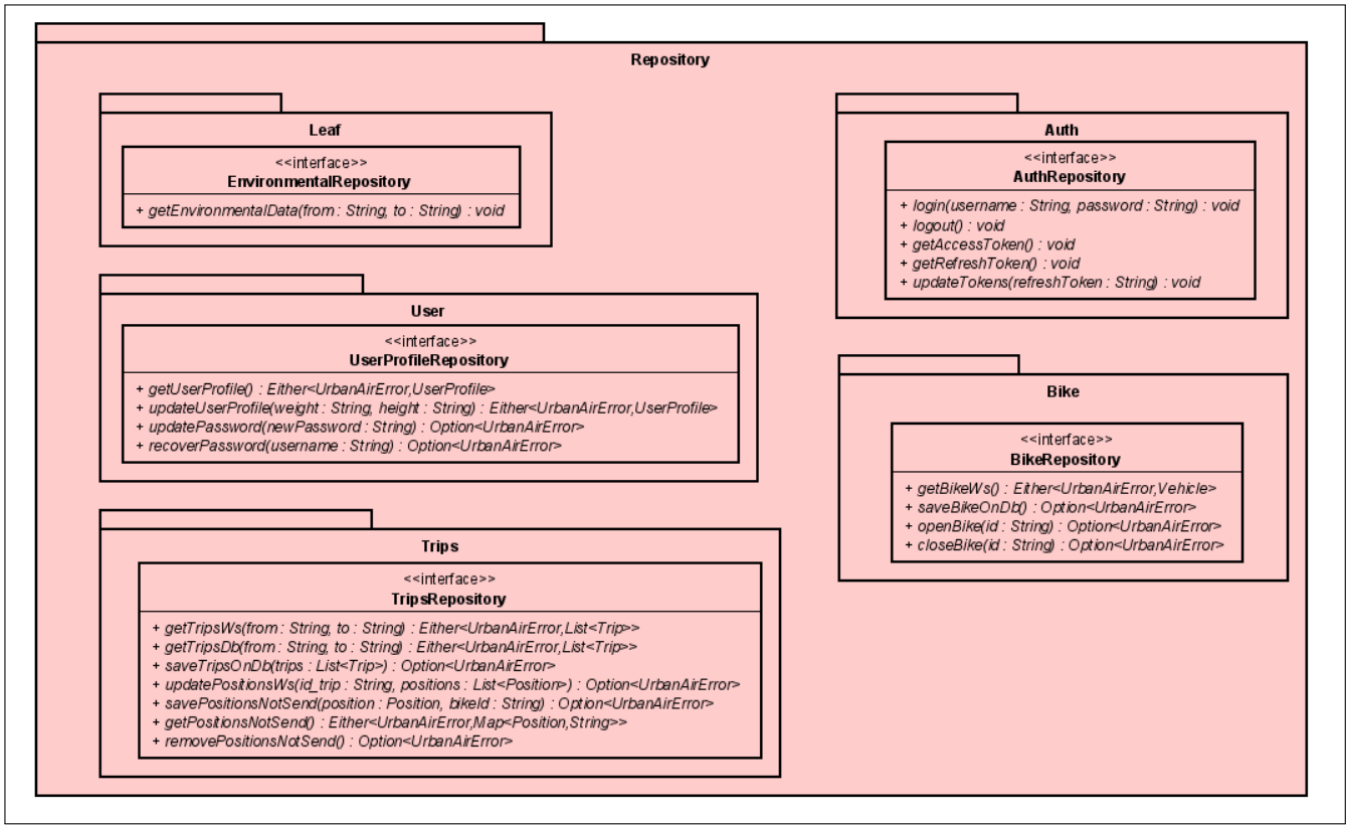

Figura 5.14: Modelo de paquetes del paquete "Repository" de la capa "Domain" en la fase de diseño.

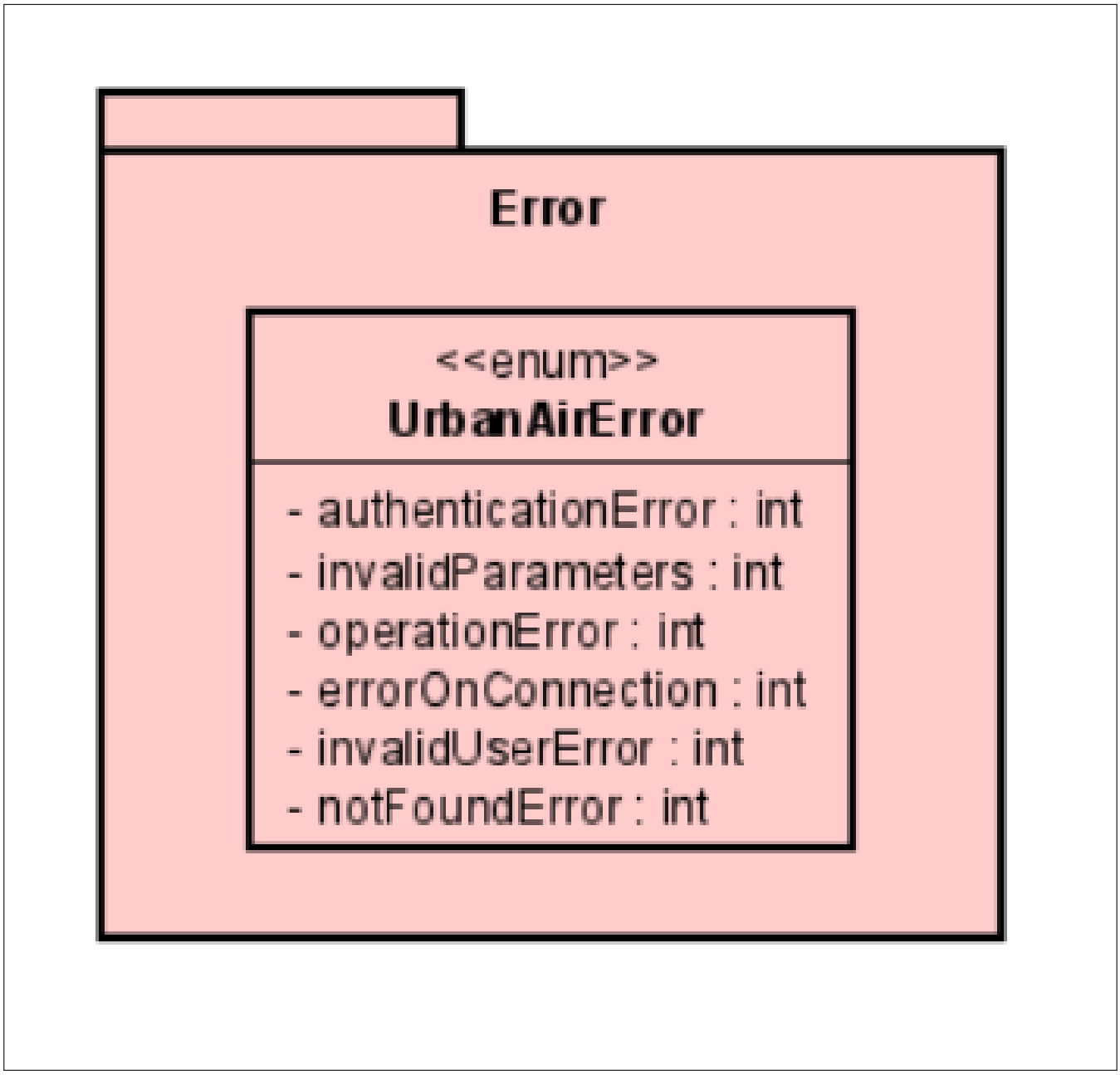

Figura 5.15: Modelo de paquetes del paquete "Error" de la capa "Domain" en la fase de diseño.

A continuación se detallan los paquetes del paquete "Data"

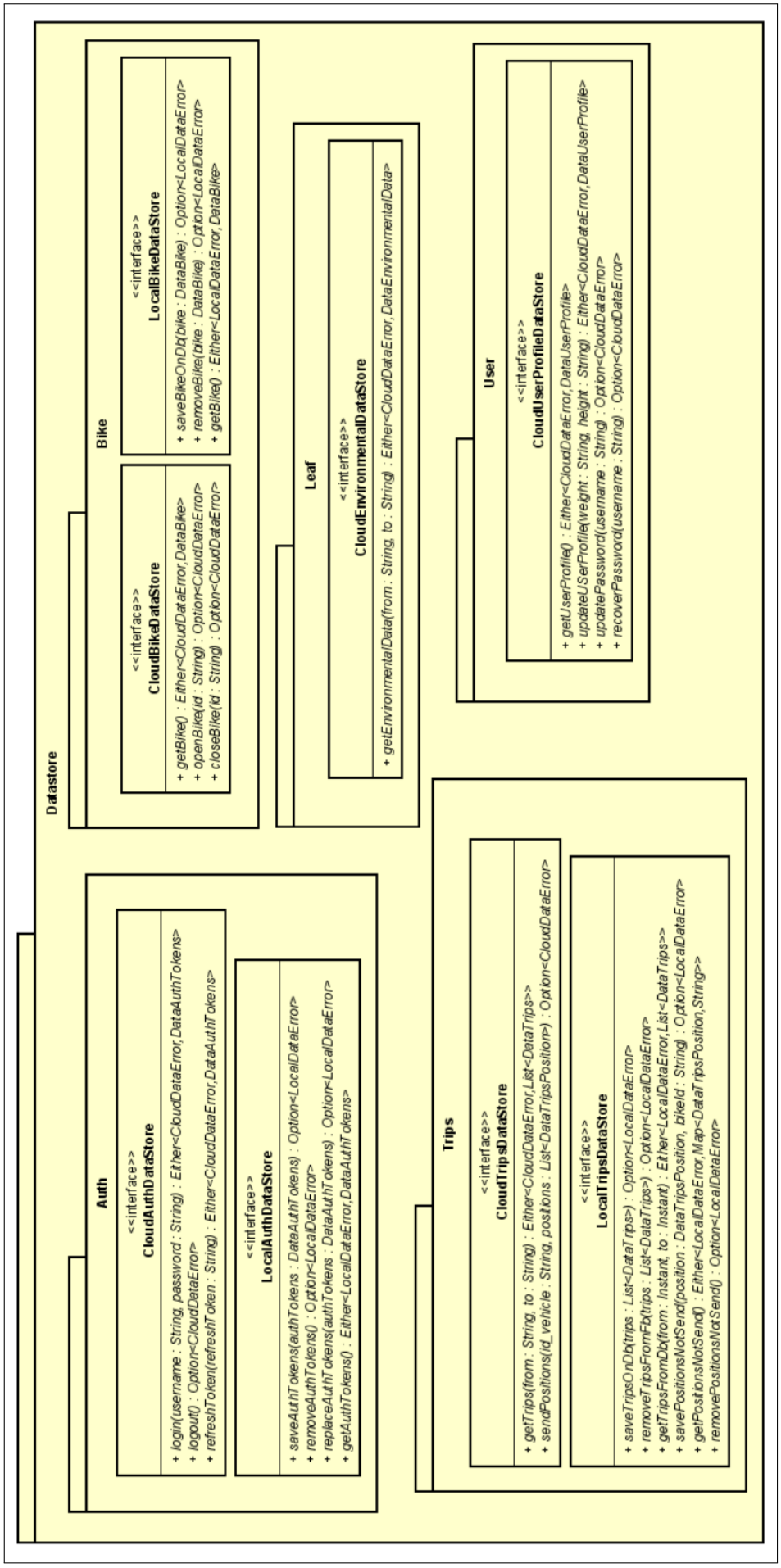

Figura 5.16: Modelo de paquetes del paquete "Datastore" en la fase de diseño.

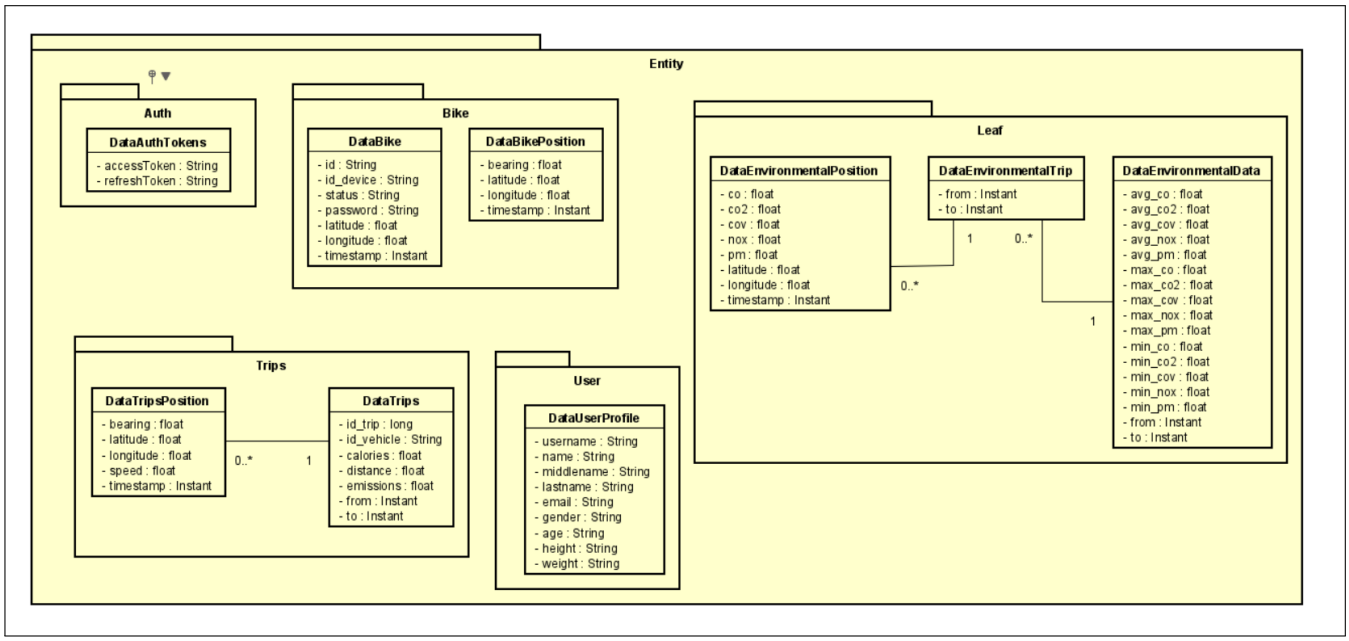

Figura 5.17: Modelo de paquetes del paquete "Entity" de la capa "Data" en la fase de diseño.

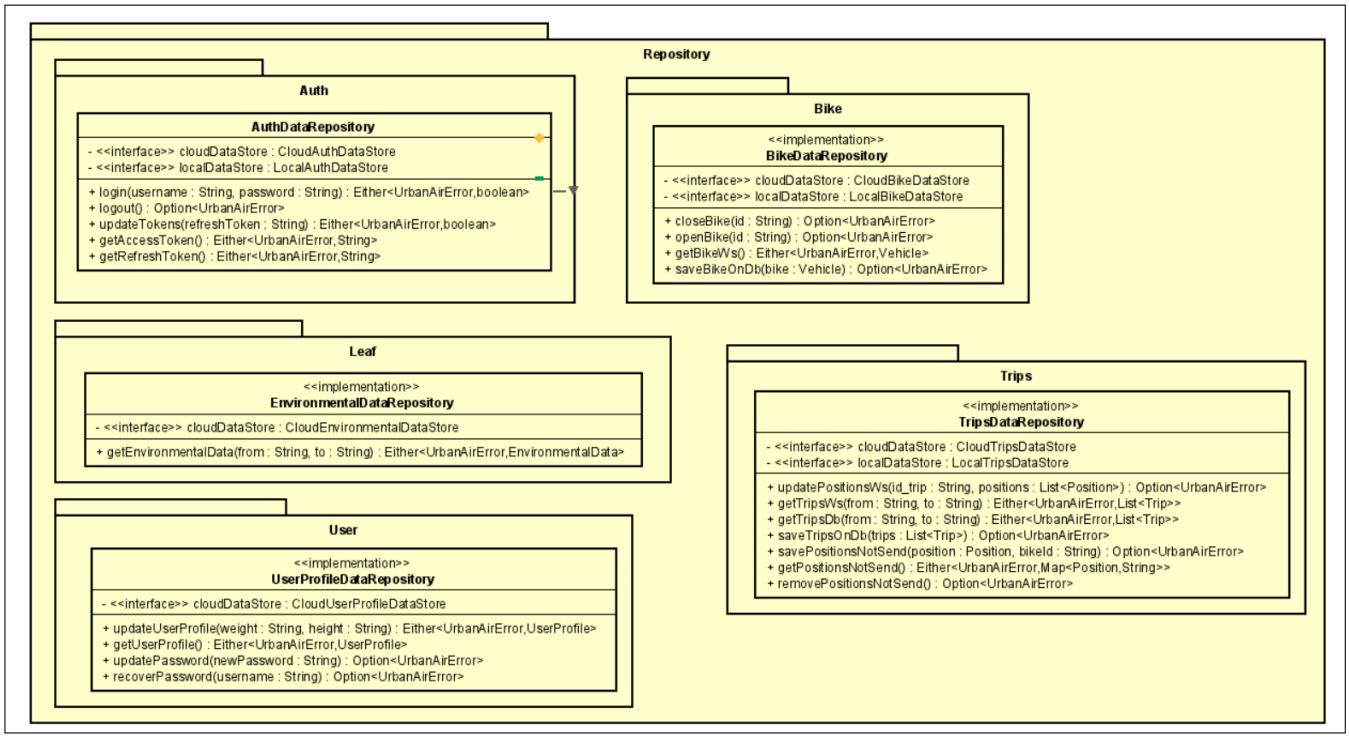

Figura 5.18: Modelo de paquetes del paquete "Repository" de la capa "Data" en la fase de diseño.

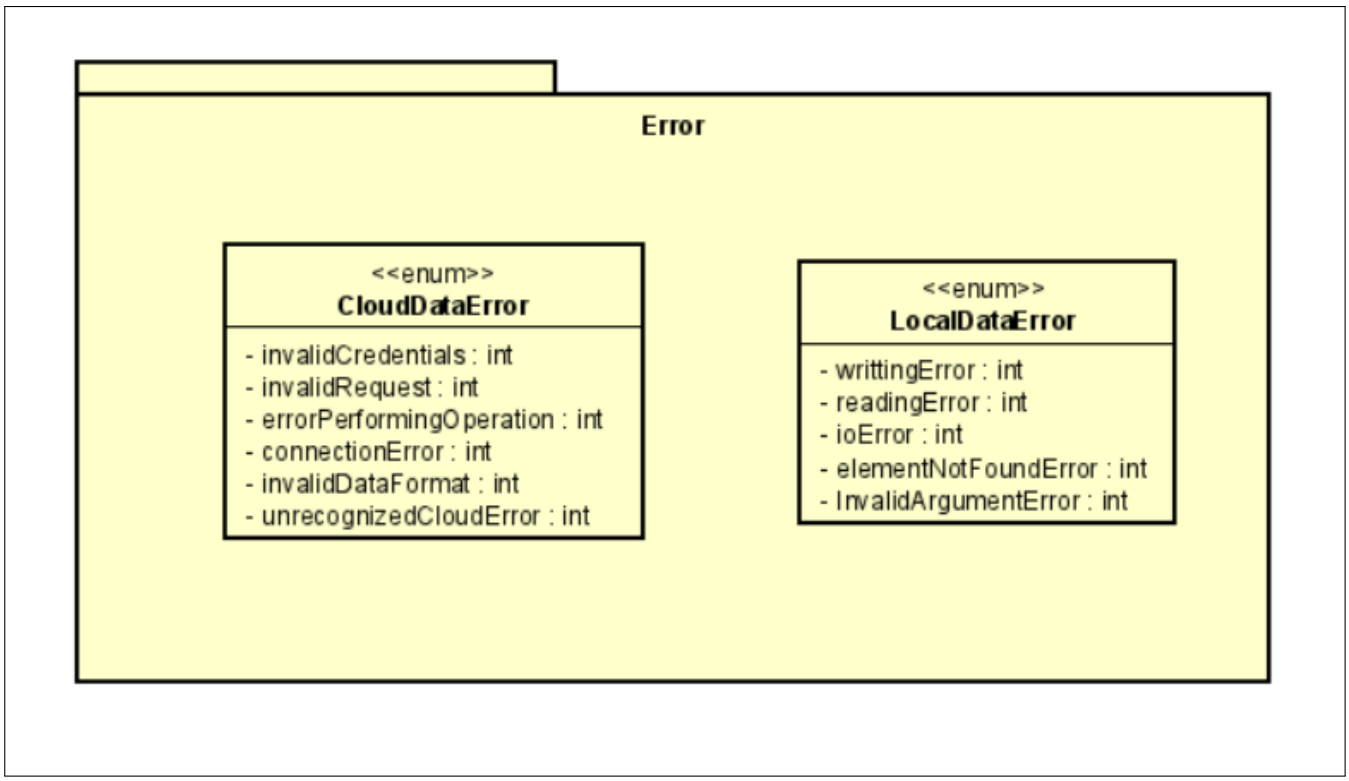

Figura 5.19: Modelo de paquetes del paquete "Error" de la capa "Data" en la fase de diseño.

A continuación se detallan los paquetes del paquete "Local"

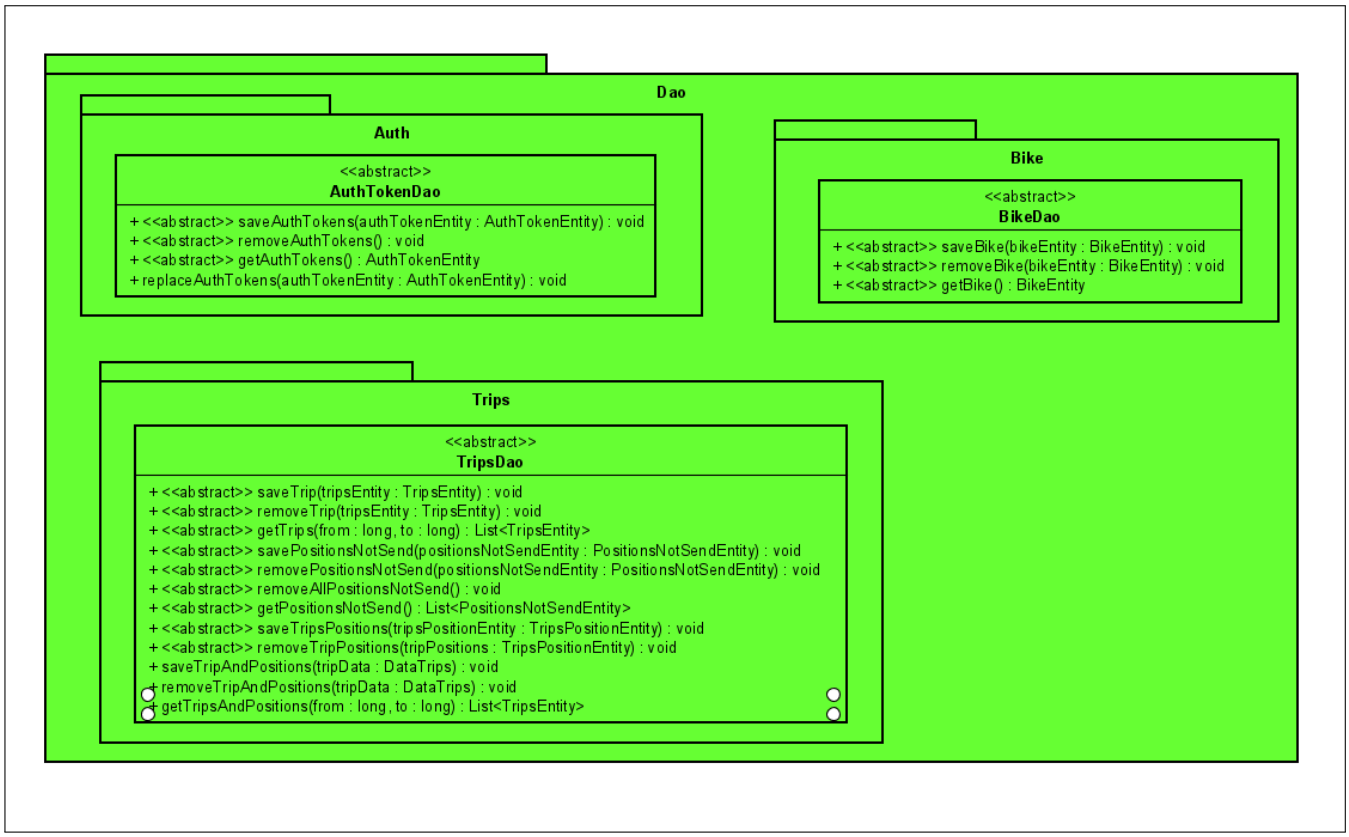

Figura 5.20: Modelo de paquetes del paquete "Dao" en la fase de diseño.

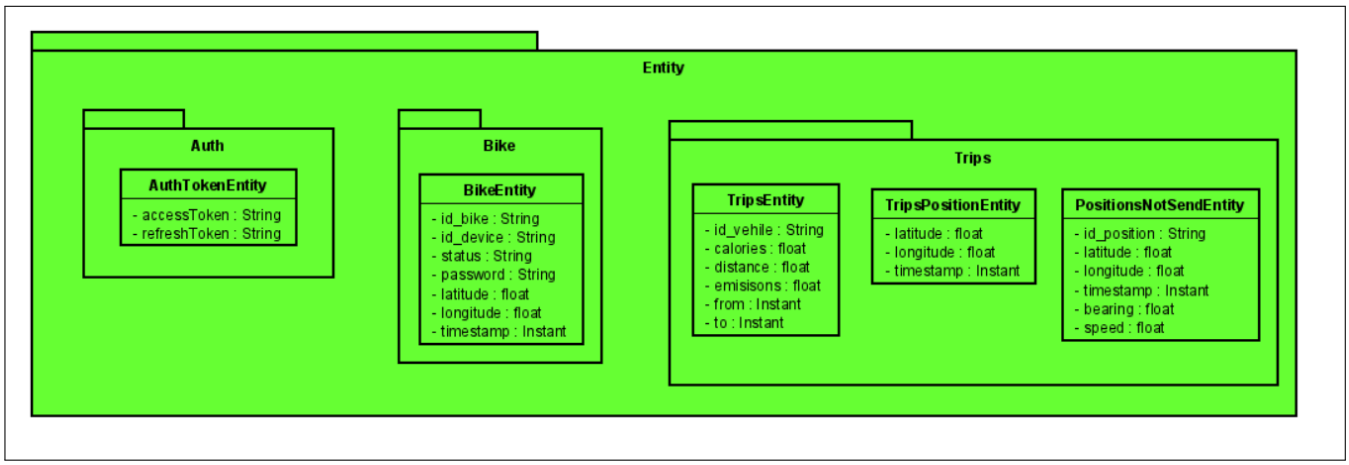

Figura 5.21: Modelo de paquetes del paquete "Entity" de la capa "Local" en la fase de diseño.

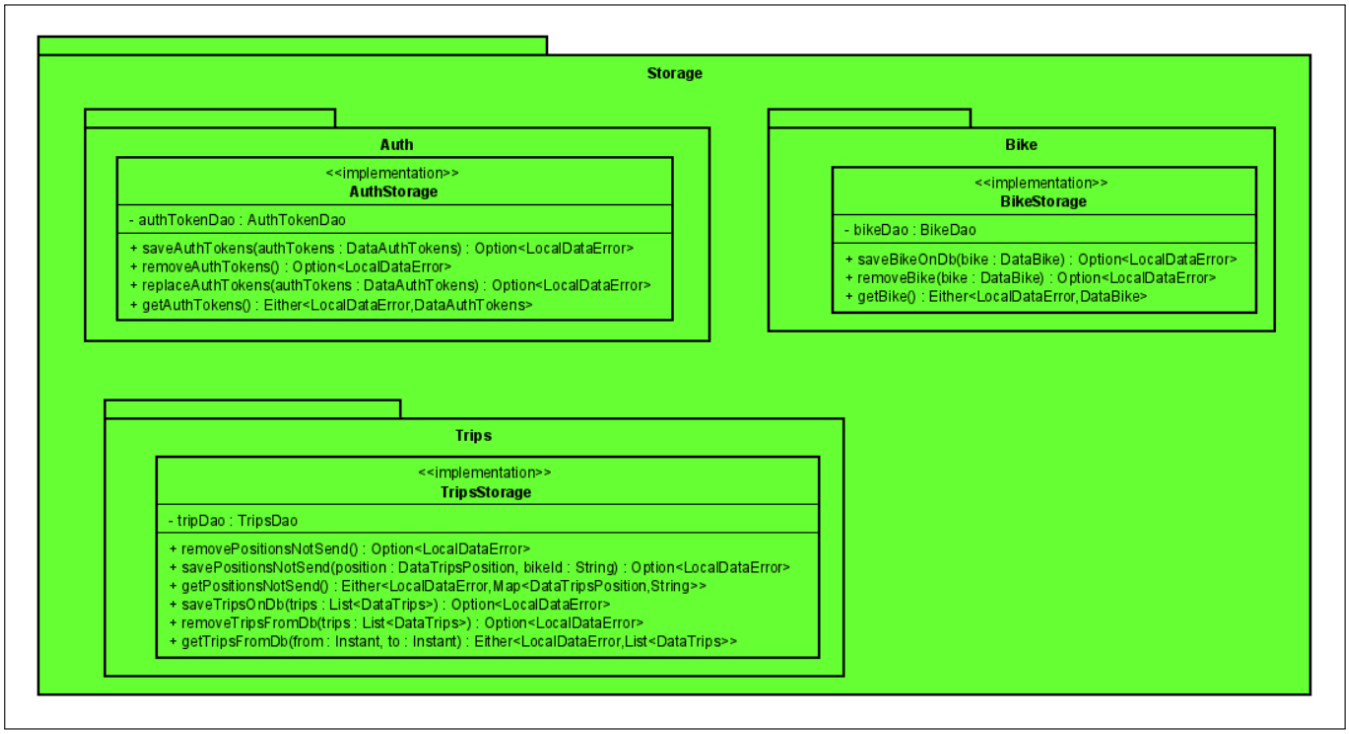

Figura 5.22: Modelo de paquetes del paquete "Storage" en la fase de diseño.

A continuación se detallan los paquetes del paquete "Remote".

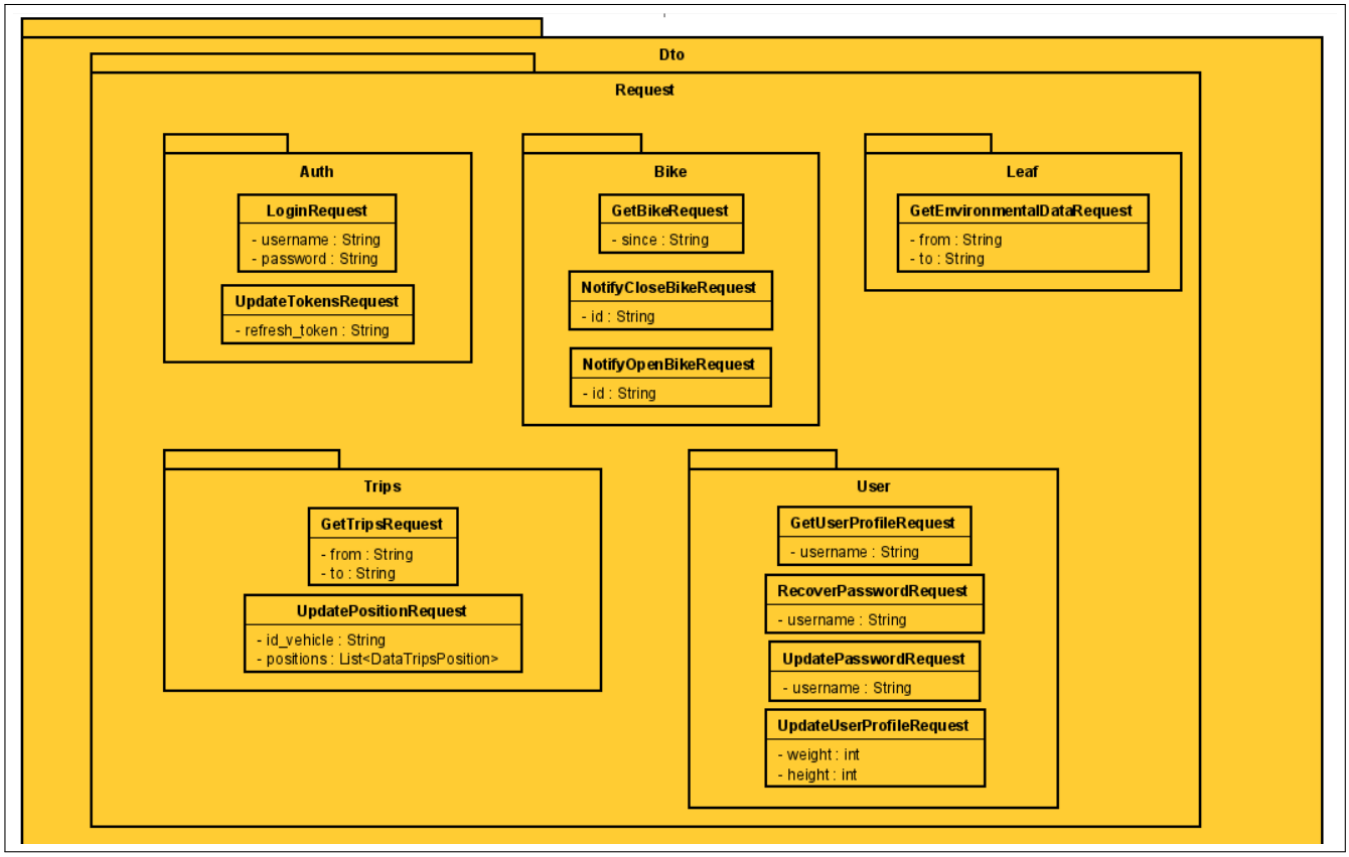

Figura 5.23: Modelo de paquetes del paquete "Dto" correspondientes a las peticiones a la API REST modeladas en la fase de diseño.

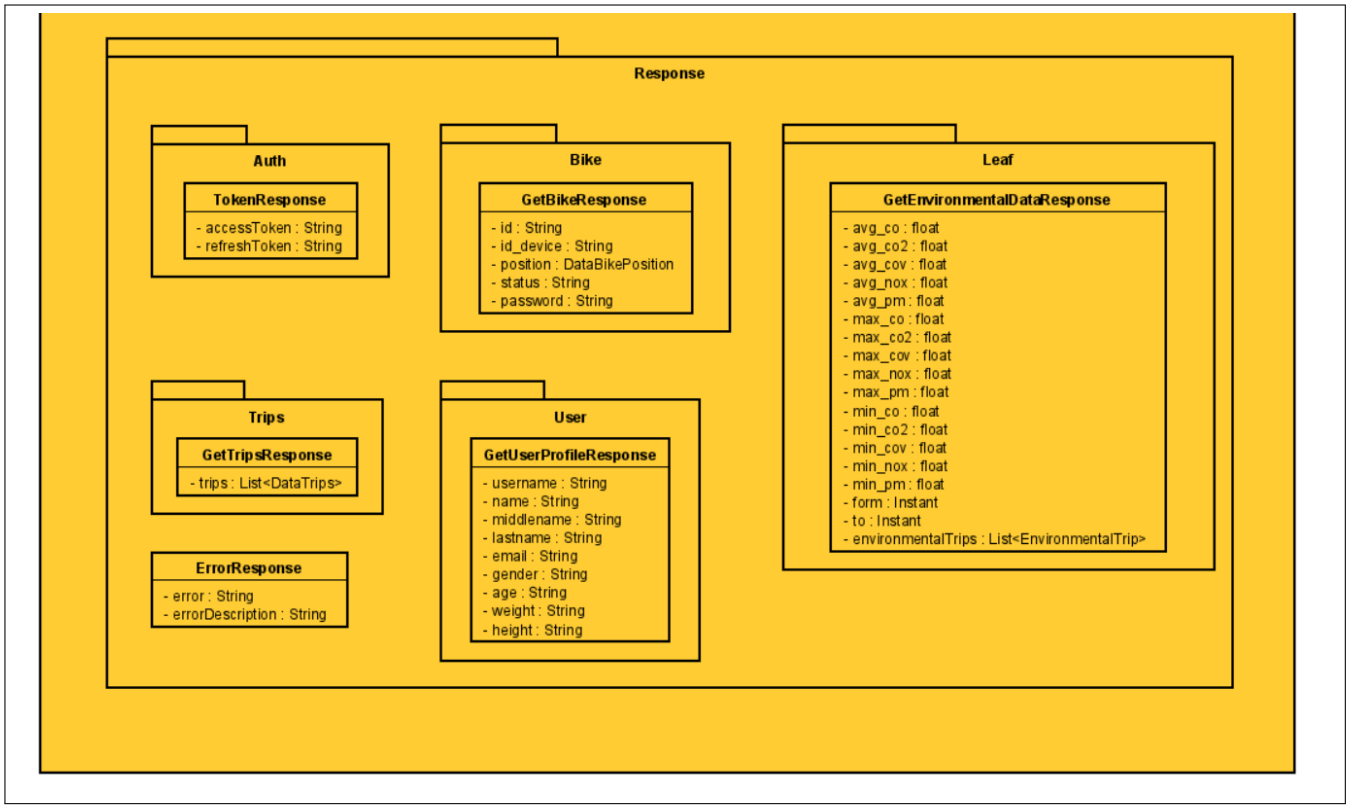

Figura 5.24: Modelo de paquetes del paquete "Dto" correspondientes a las respuestas de la API REST modeladas en la fase de diseño.

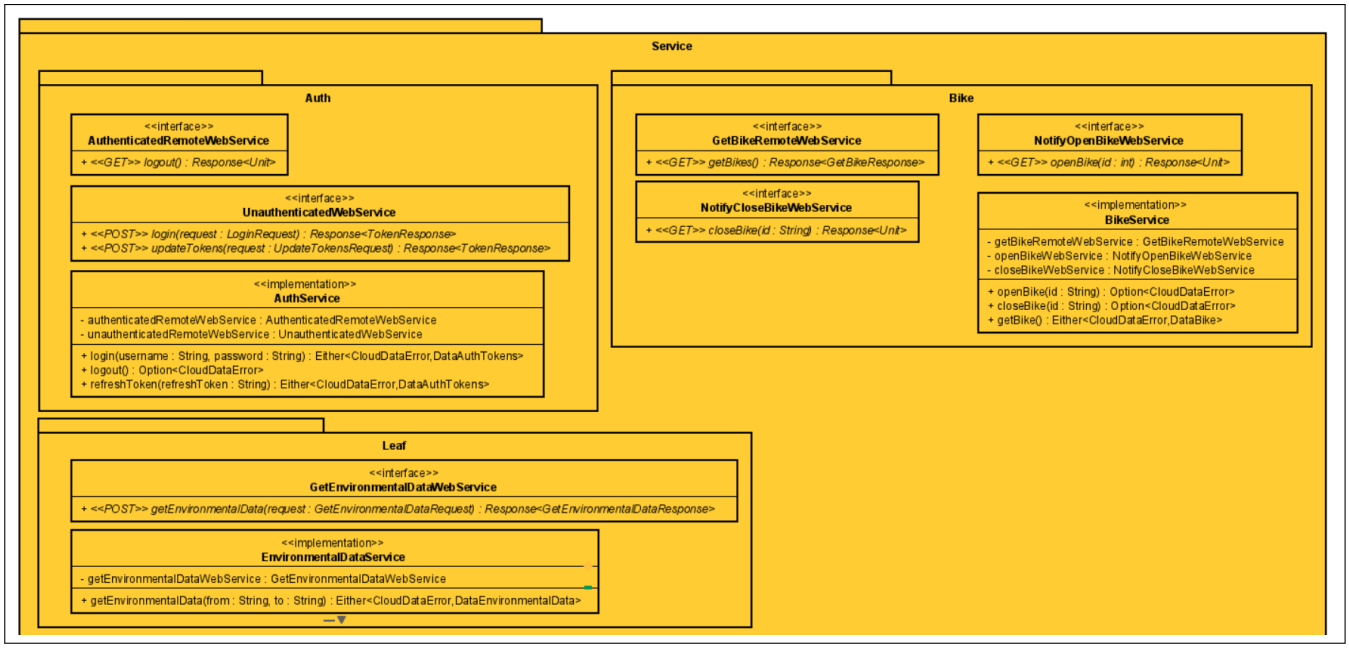

Figura 5.25: Modelo de paquetes del paquete "Dto" correspondientes a los servicios de la API REST modelados en la fase de diseño parte 1.

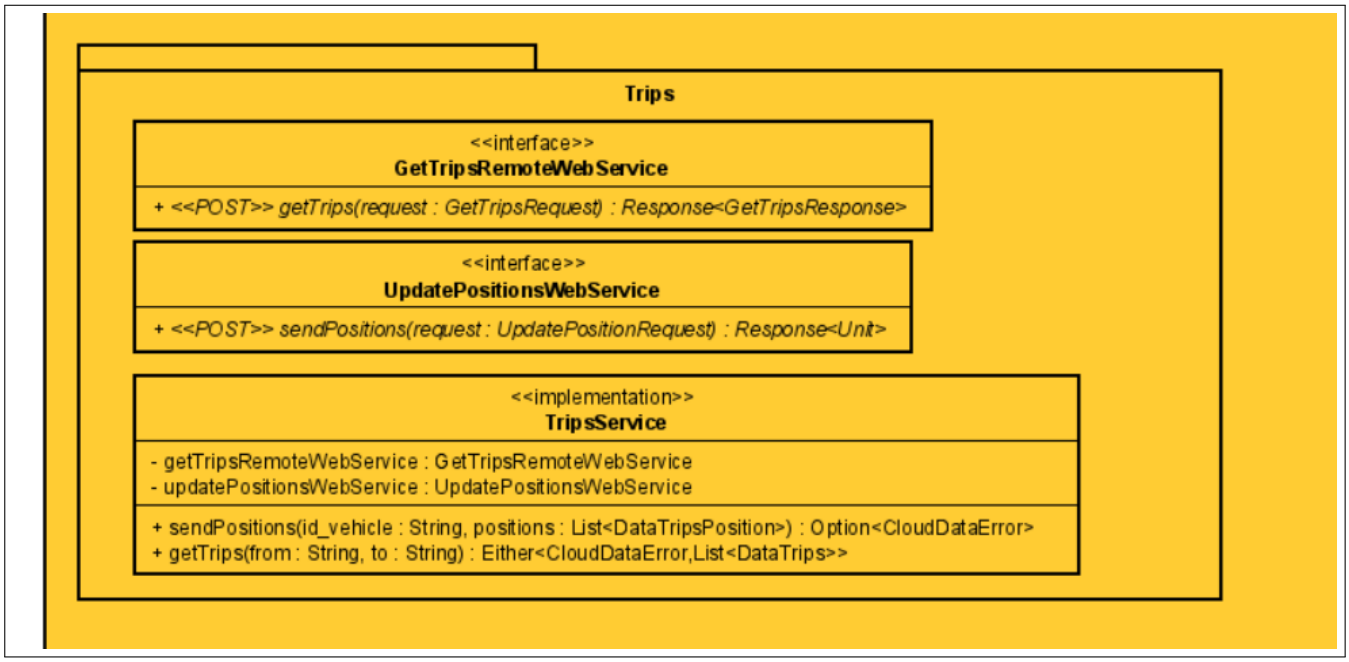

Figura 5.26: Modelo de paquetes del paquete "Dto" correspondientes a los servicios de la API REST modelados en la fase de diseño parte 2.

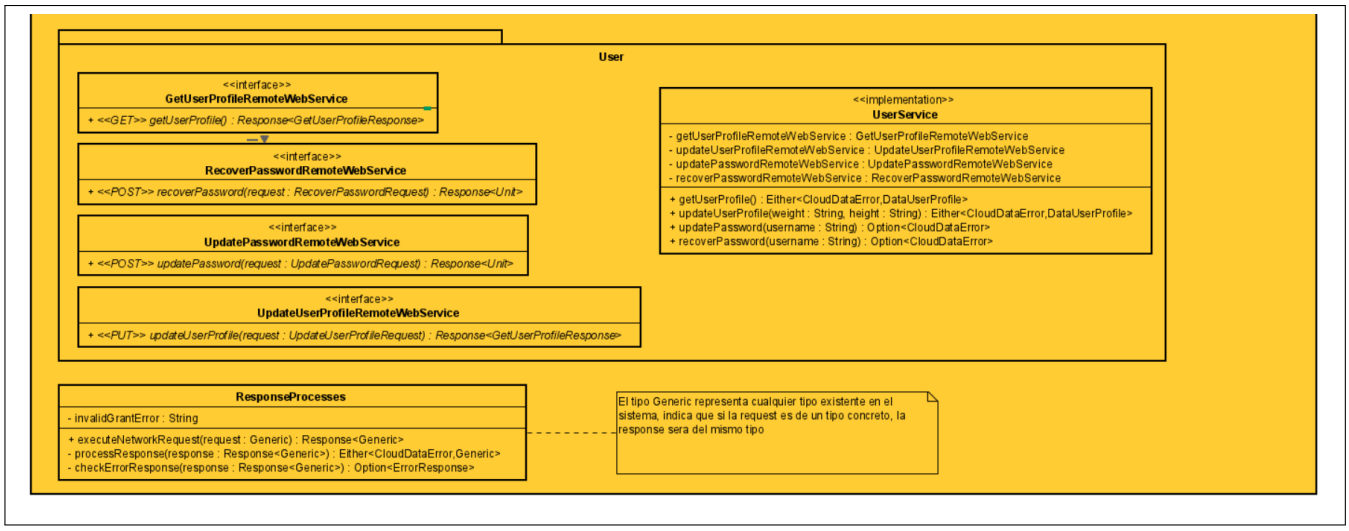

Figura 5.27: Modelo de paquetes del paquete "Dto" correspondientes a los servicios de la API REST modelados en la fase de diseño parte 3.

5.3. Diagramas de secuencia en diseño

En esta sección se muestran los diagramas de secuencia de los casos de uso mas significativos. En este caso se van a mostrar los diagramas que corresponden al caso de uso de Login 4.3, al caso de uso de Iniciar Viaje 4.4 y al caso de uso Mostrar Viajes 4.6 los demás caso de uso son muy similares y se realizan de manera análoga a los representados en este capítulo. Además se representará únicamente la secuencia principal sin considerar los posibles errores o alternativas para mejorar la visibilidad y legibilidad de los diagramas, también se hará un diagrama aparte mostrando como sería una secuencia de error genérica ya que todas se van a tratar de la misma forma.

A continuación se detallan los diagramas de secuencia del caso de uso "Login". 4.3

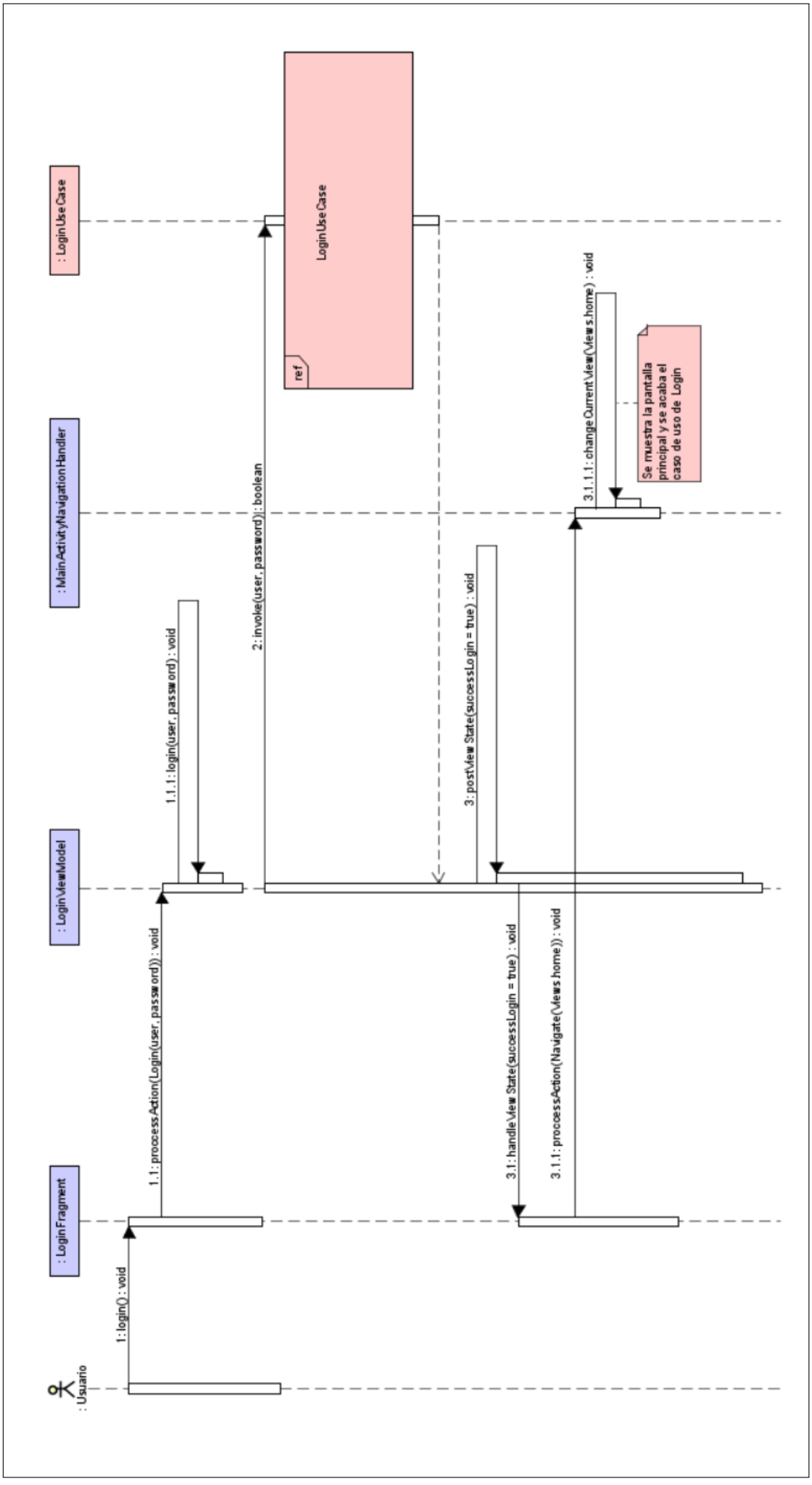

Figura 5.28: Diagrama de secuencia de la capa "App" del caso de uso "Login".

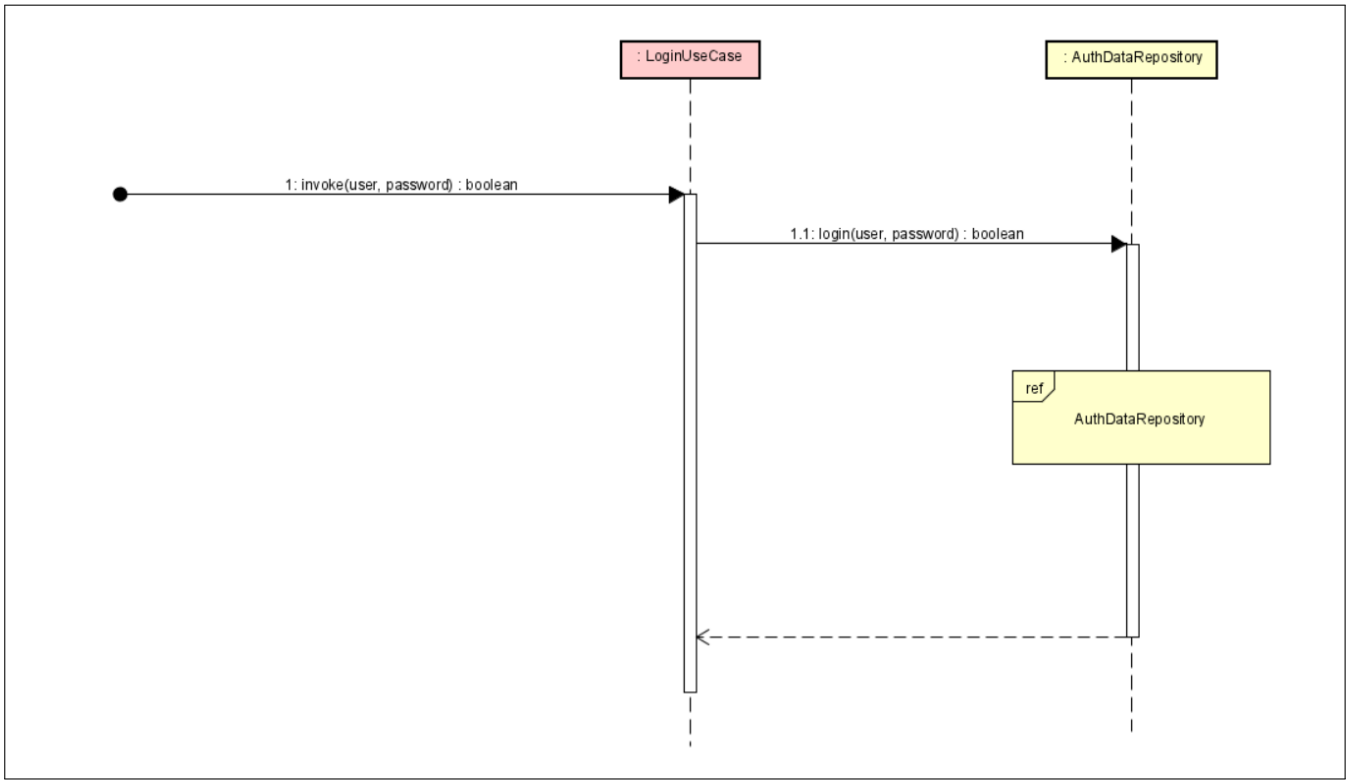

Figura 5.29: Diagrama de secuencia de la capa "Domain" del caso de uso "Login".

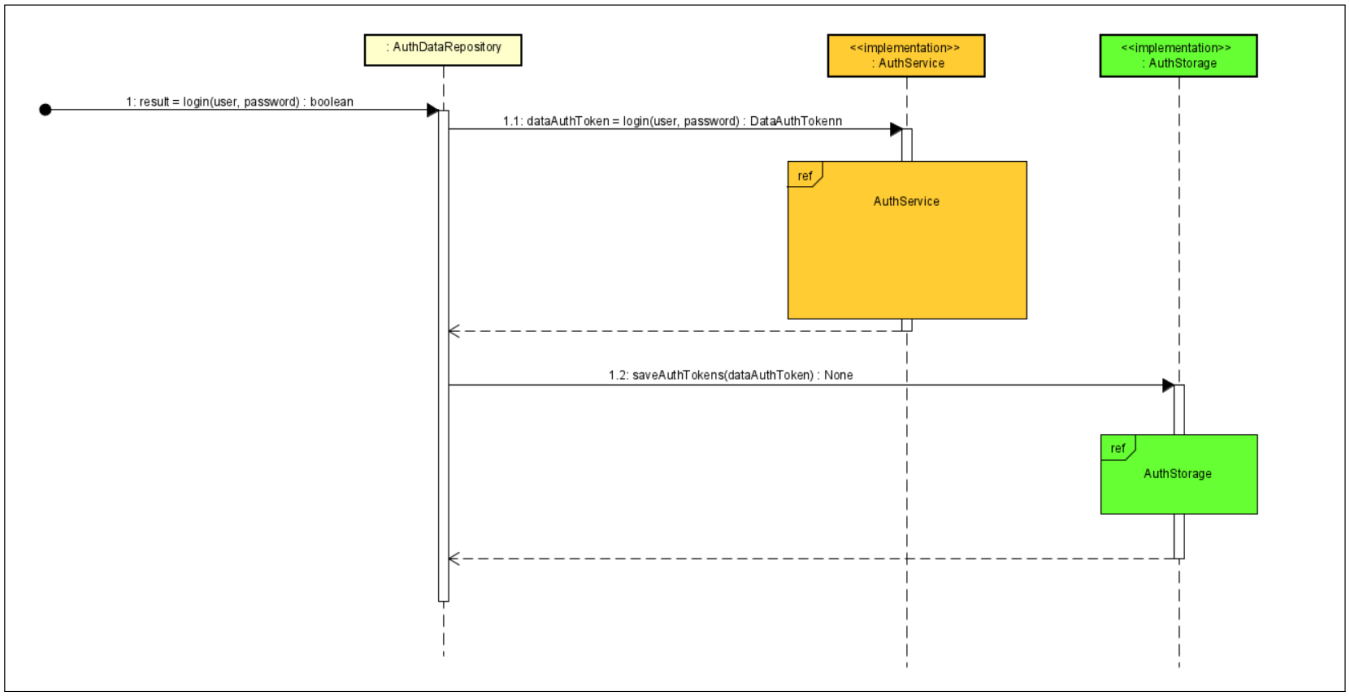

Figura 5.30: Diagrama de secuencia de la capa "Data" del caso de uso "Login".

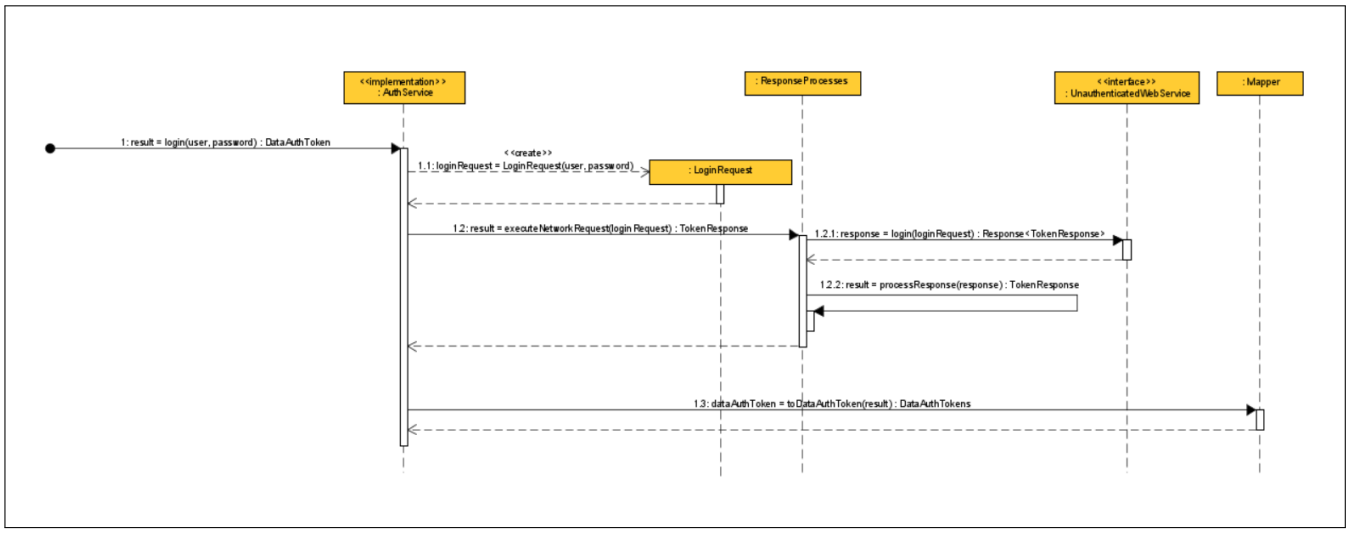

Figura 5.31: Diagrama de secuencia de la capa "Remote" del caso de uso "Login".

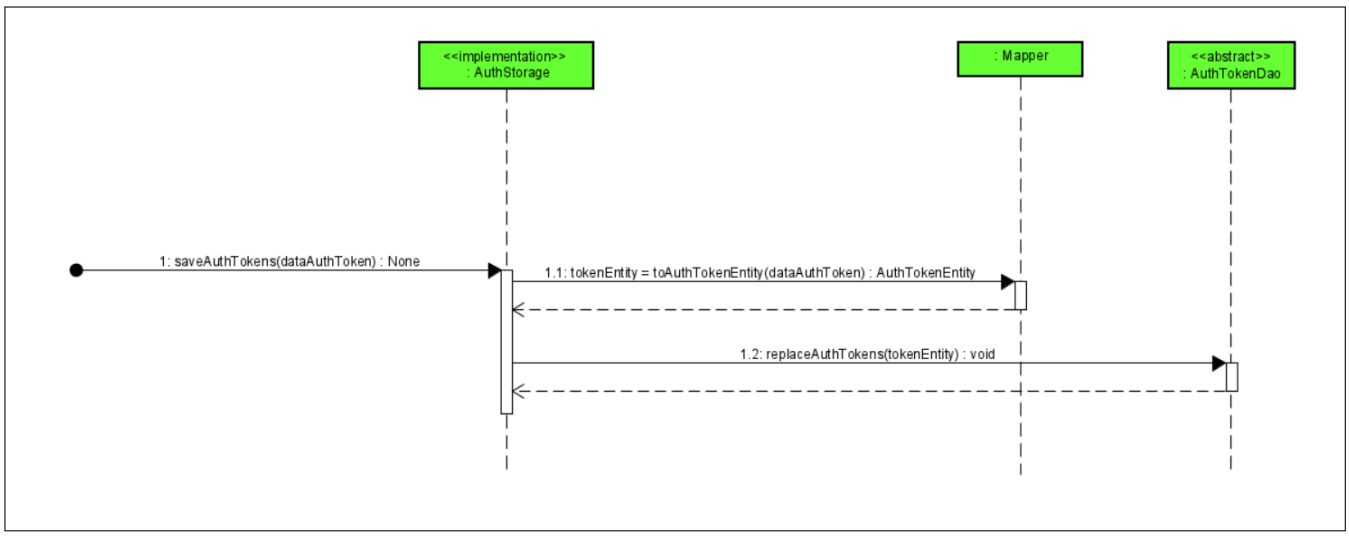

Figura 5.32: Diagrama de secuencia de la capa "Local" del caso de uso "Login".

A continuación se detallan los diagramas de secuencia del caso de uso "Iniciar viaje" 4.4

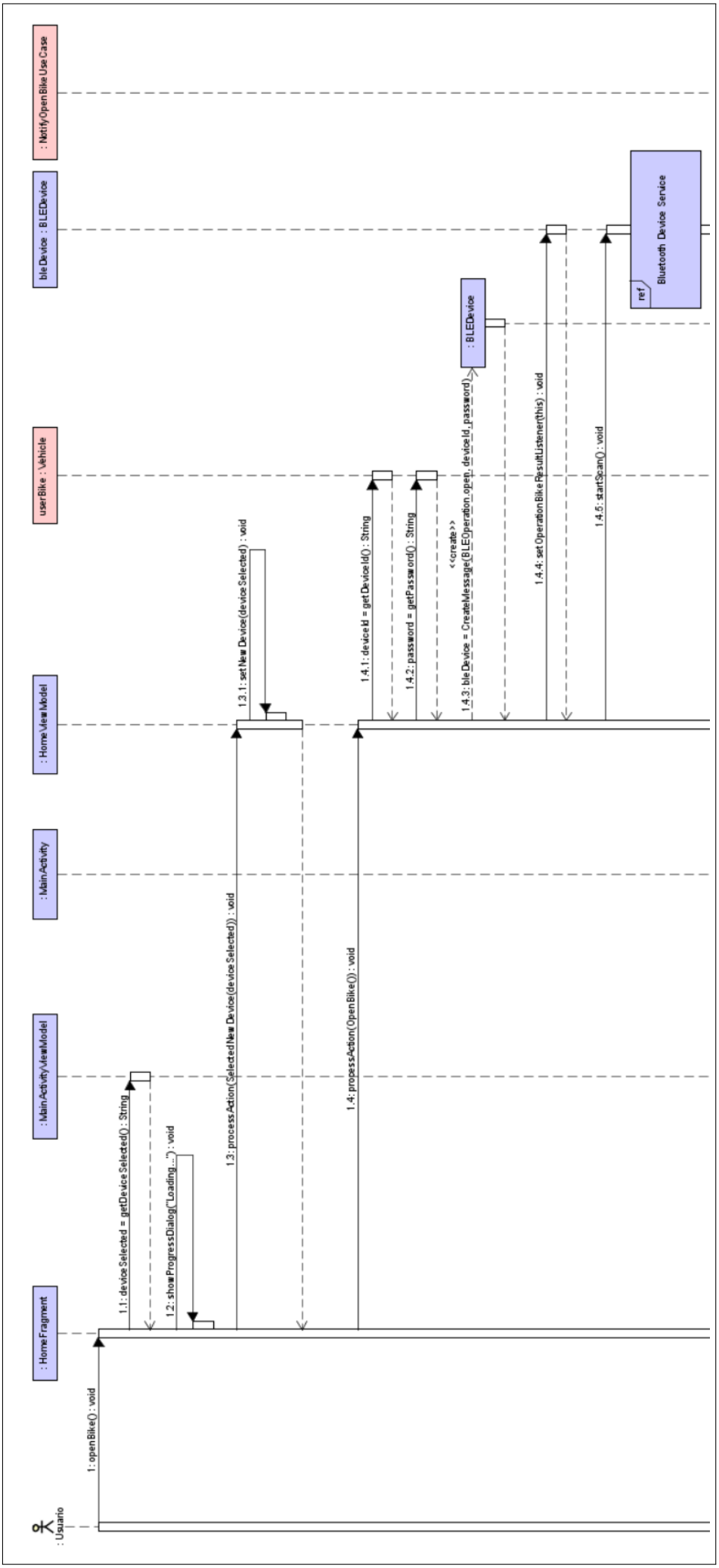

Figura 5.33: Diagrama de secuencia de la capa "App" del caso de uso "Iniciar viaje" parte 1.

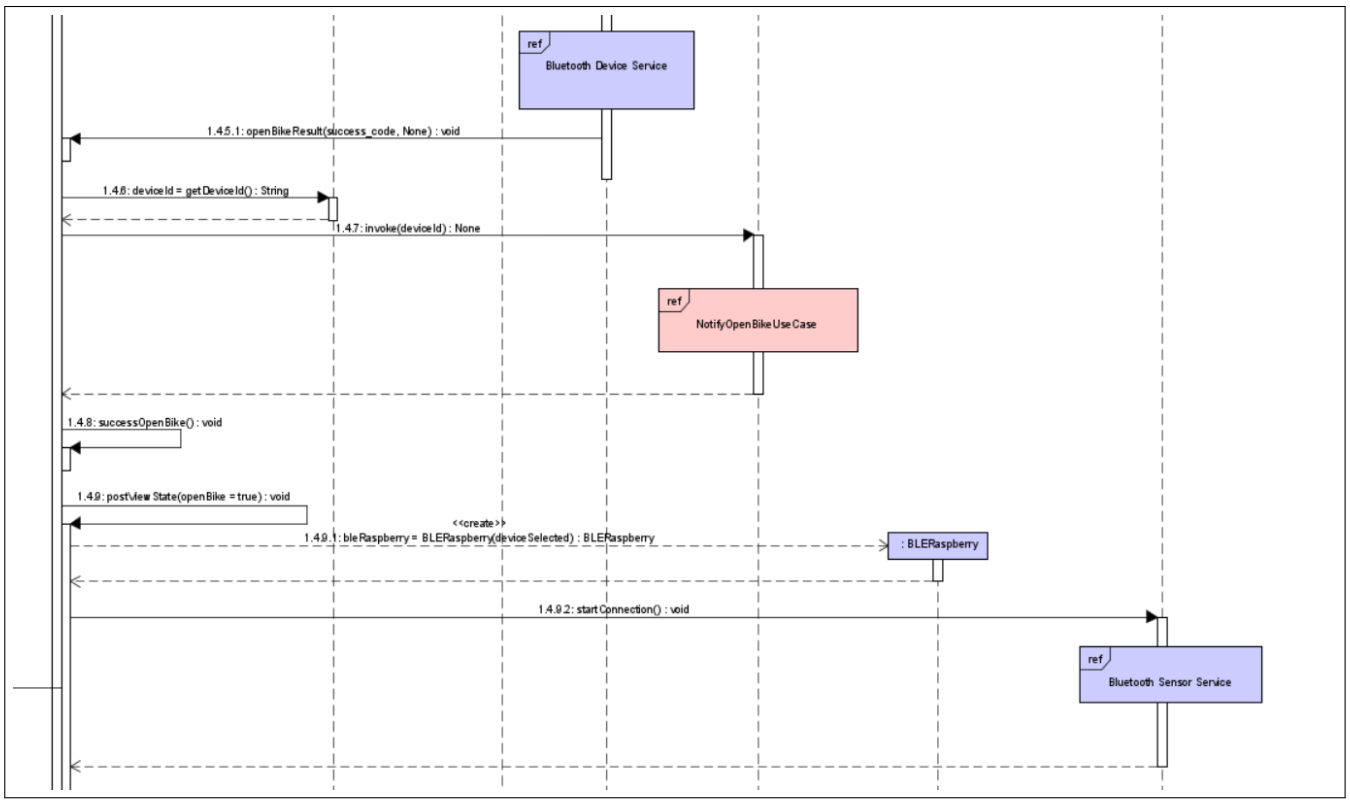

Figura 5.34: Diagrama de secuencia de la capa "App" del caso de uso "Iniciar viaje" parte 2.

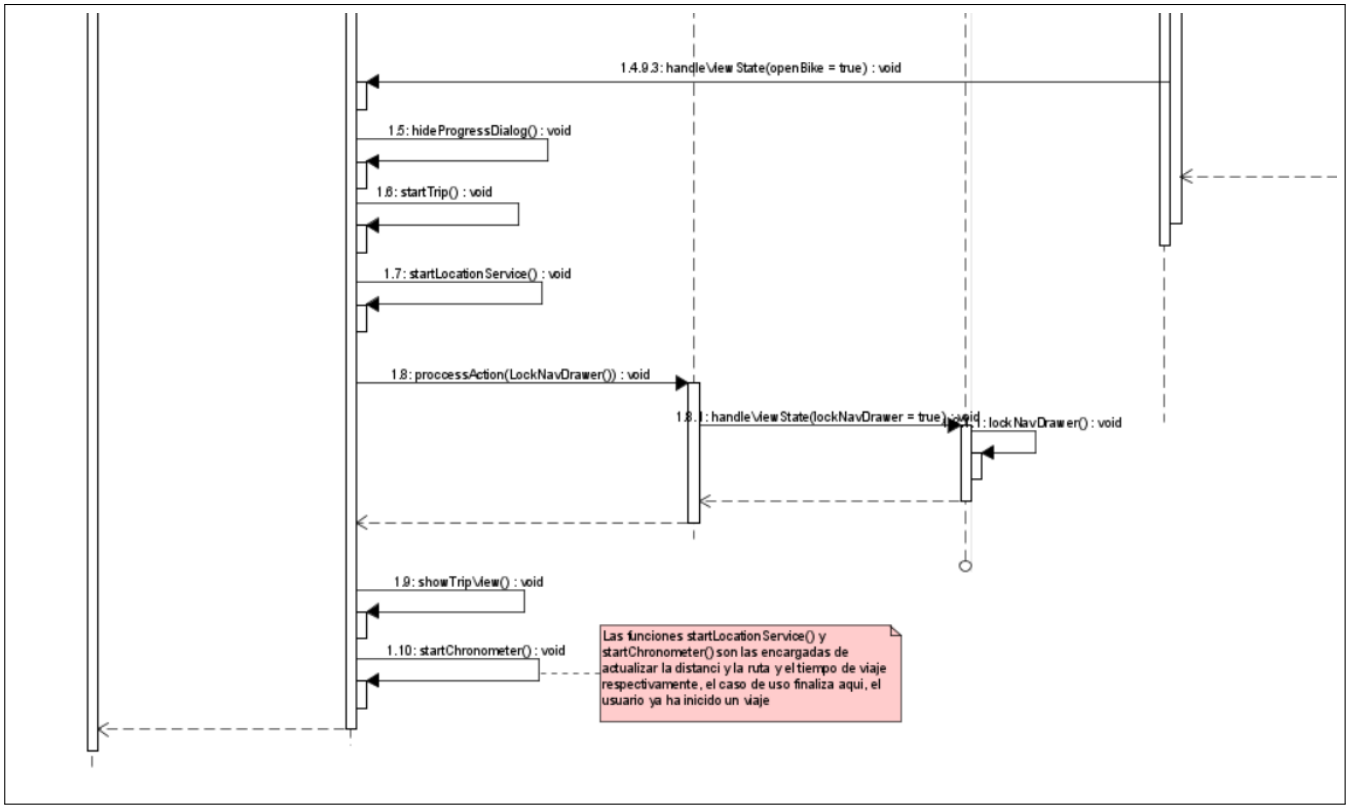

Figura 5.35: Diagrama de secuencia de la capa "App" del caso de uso "Iniciar viaje" parte 3.

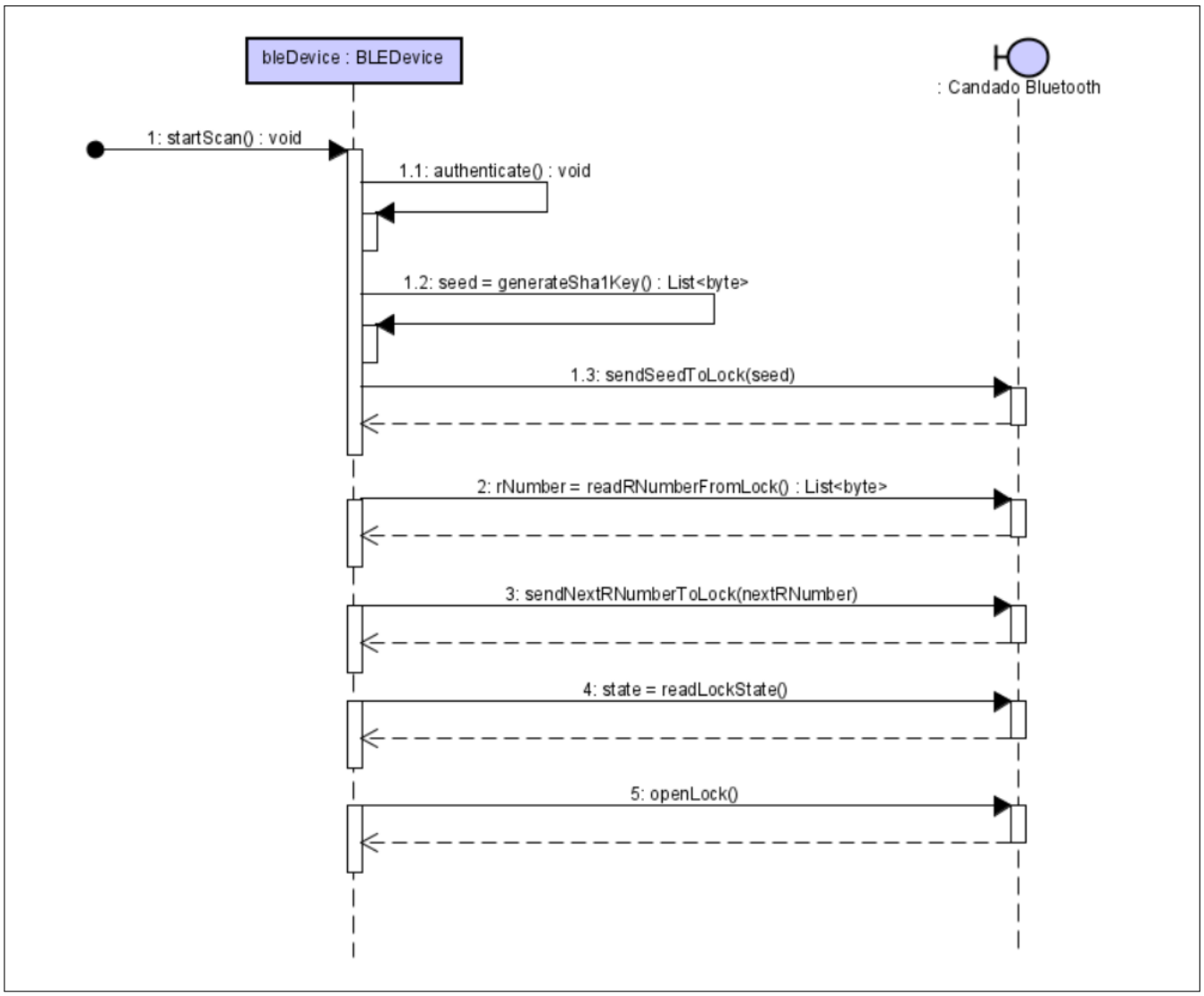

Figura 5.36: Diagrama de secuencia de la capa "App" paquete "Bluetooth" para la comunicación con el candado del caso de uso "Iniciar viaje".

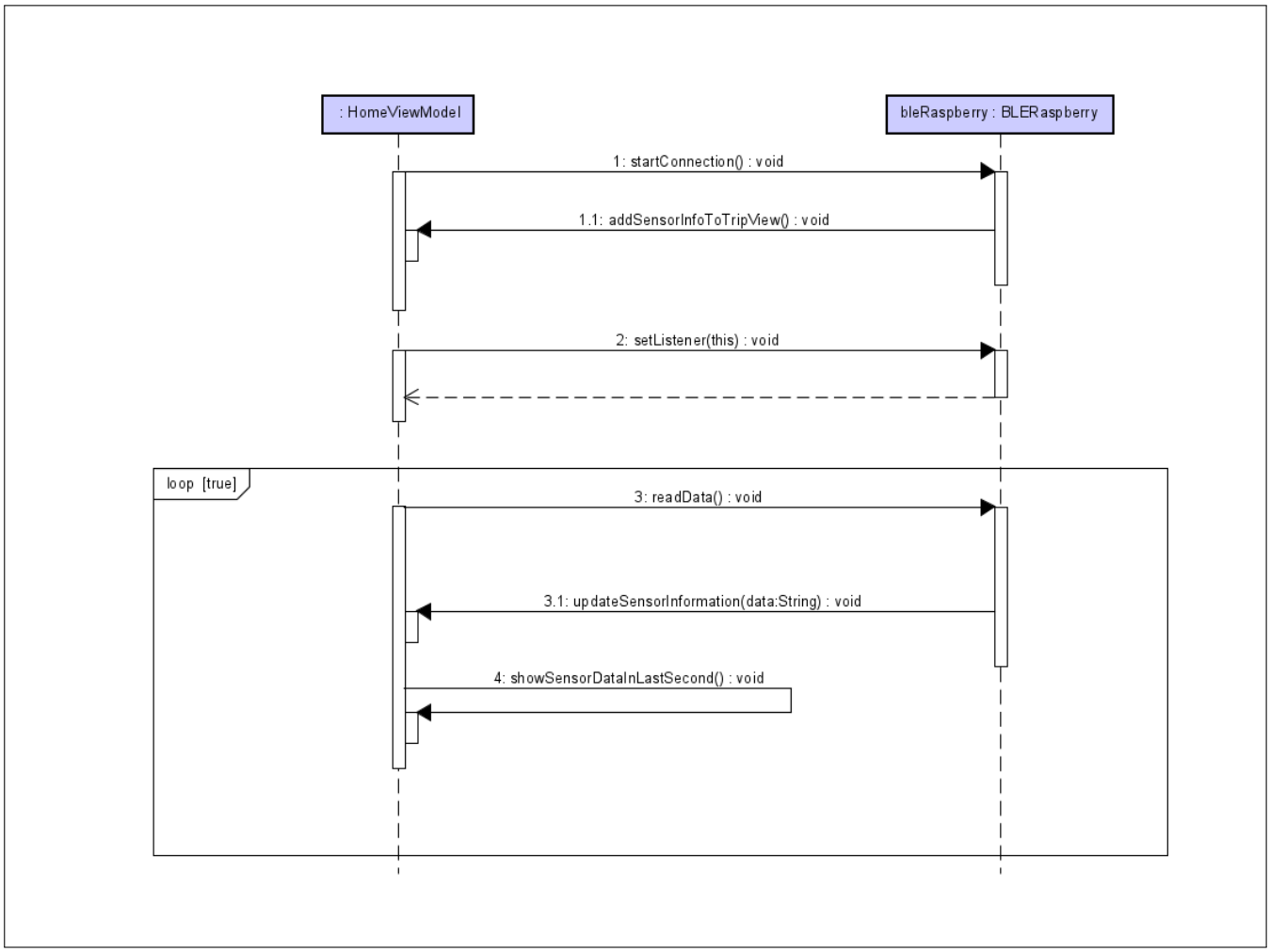

Figura 5.37: Diagrama de secuencia de la capa "App" paquete "Bluetooth" para la comunicación con el sensor del caso de uso "Iniciar viaje".

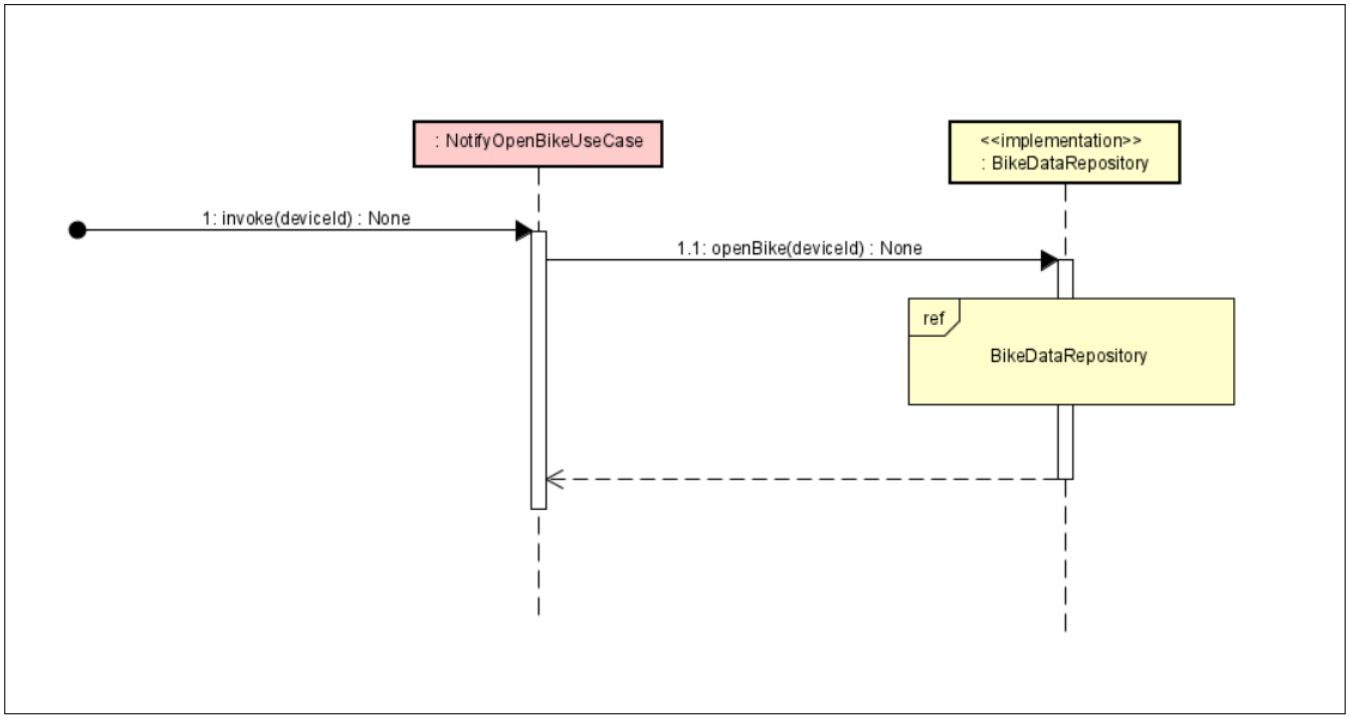

Figura 5.38: Diagrama de secuencia de la capa "Domain" del caso de uso "Iniciar viaje".

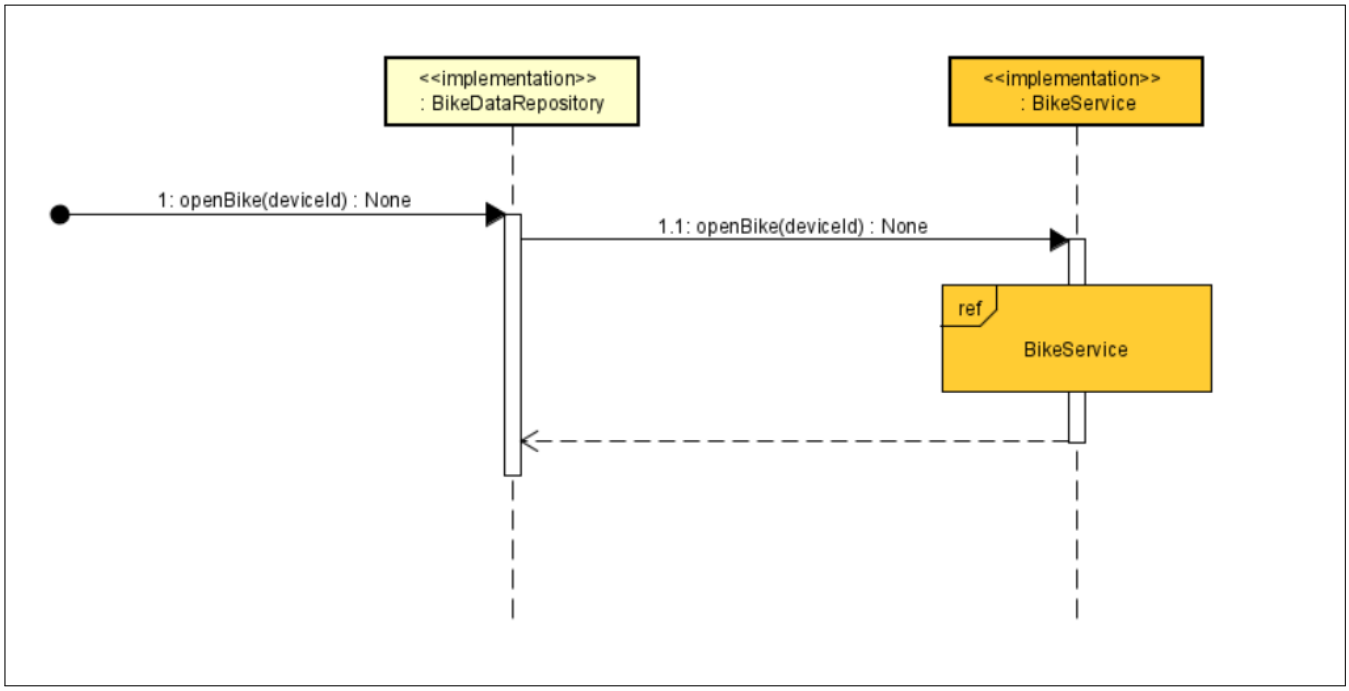

Figura 5.39: Diagrama de secuencia de la capa "Data" del caso de uso "Iniciar viaje".

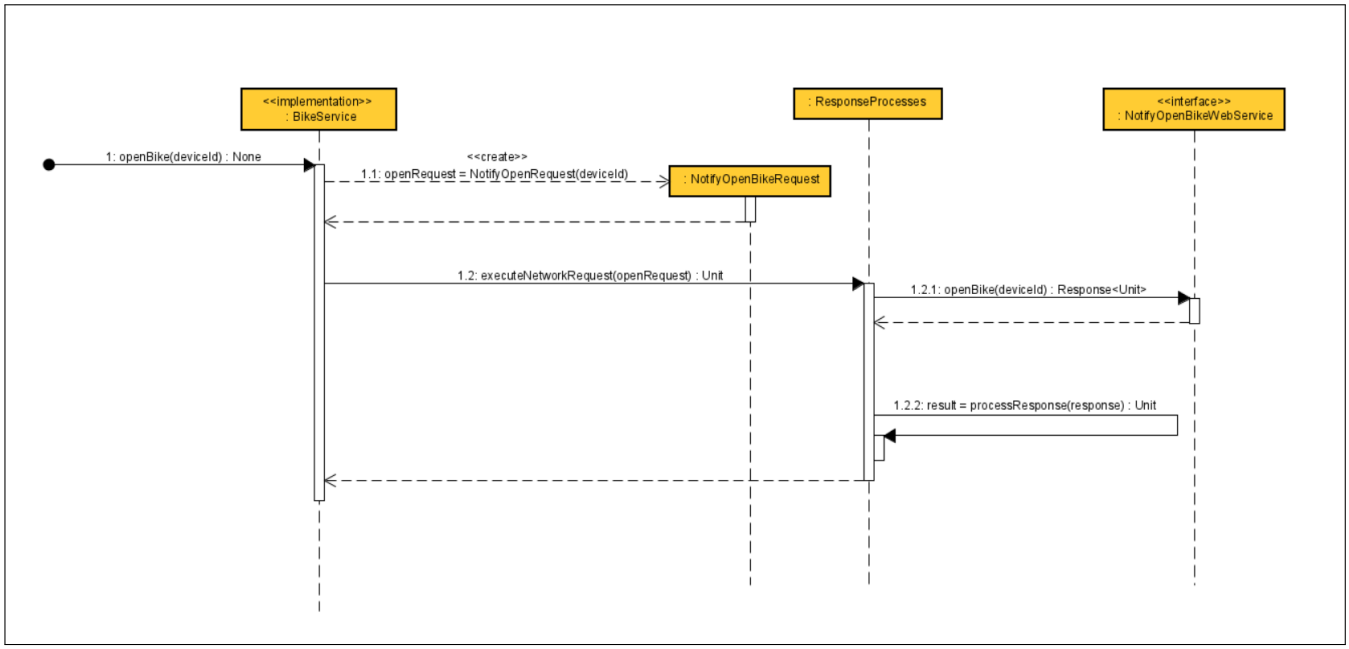

Figura 5.40: Diagrama de secuencia de la capa "Remote" del caso de uso "Iniciar viaje".

A continuación se detallan los diagramas de secuencia del caso de uso "Mostrar Viajes" 4.6

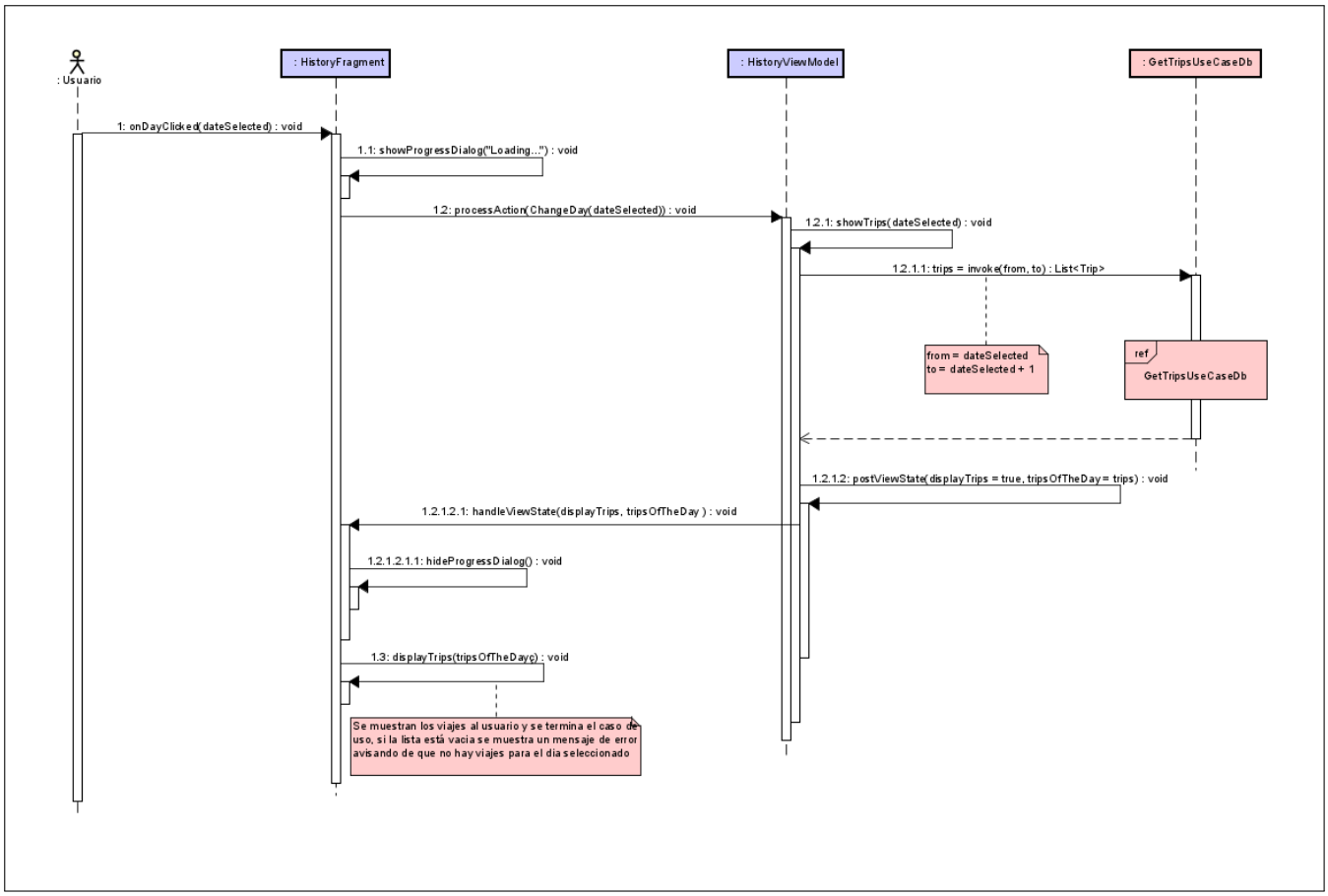

Figura 5.41: Diagrama de secuencia de la capa "App" del caso de uso "Mostrar viajes".

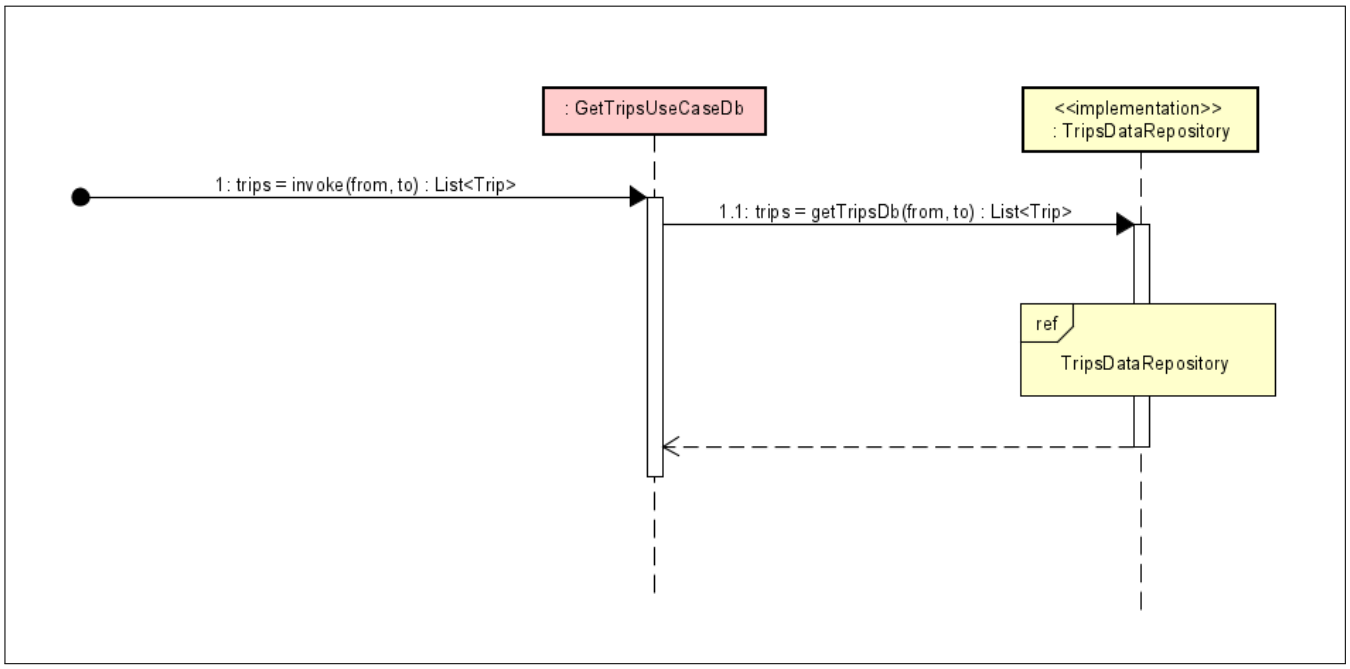

Figura 5.42: Diagrama de secuencia de la capa "Domain" del caso de uso "Mostrar viajes".

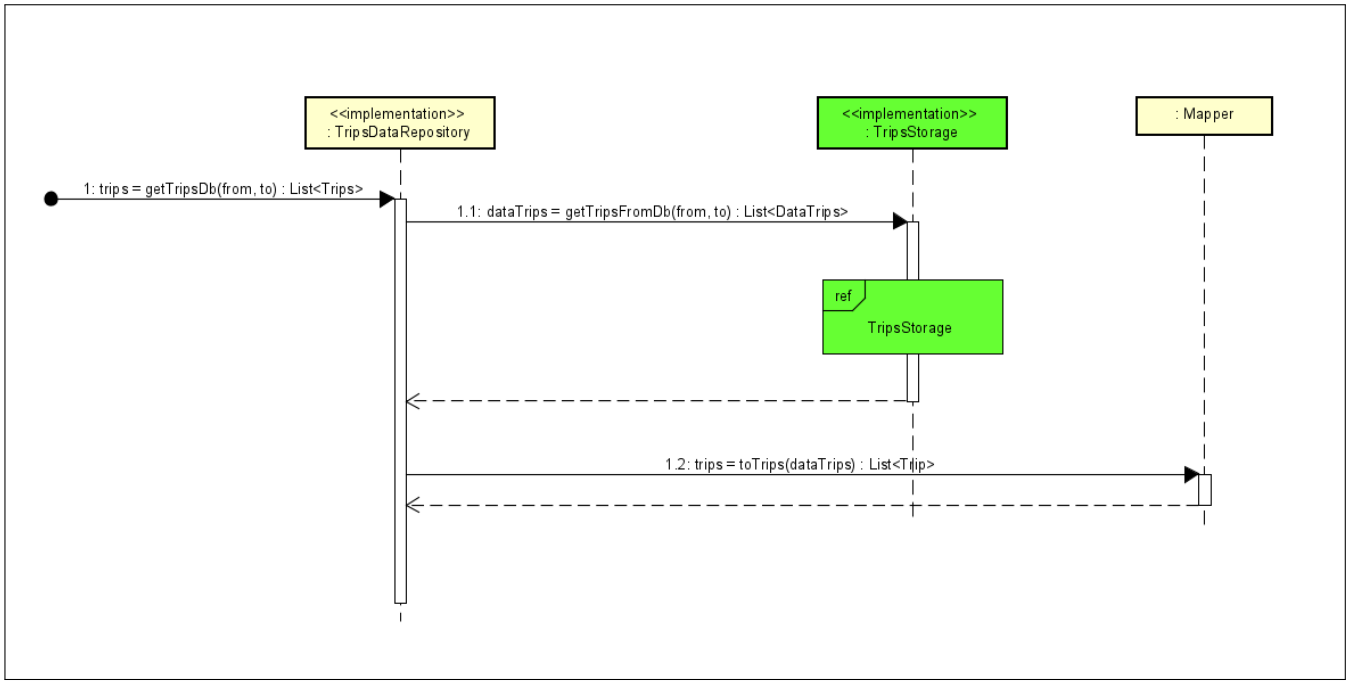

Figura 5.43: Diagrama de secuencia de la capa "Data" del caso de uso "Mostrar viajes".

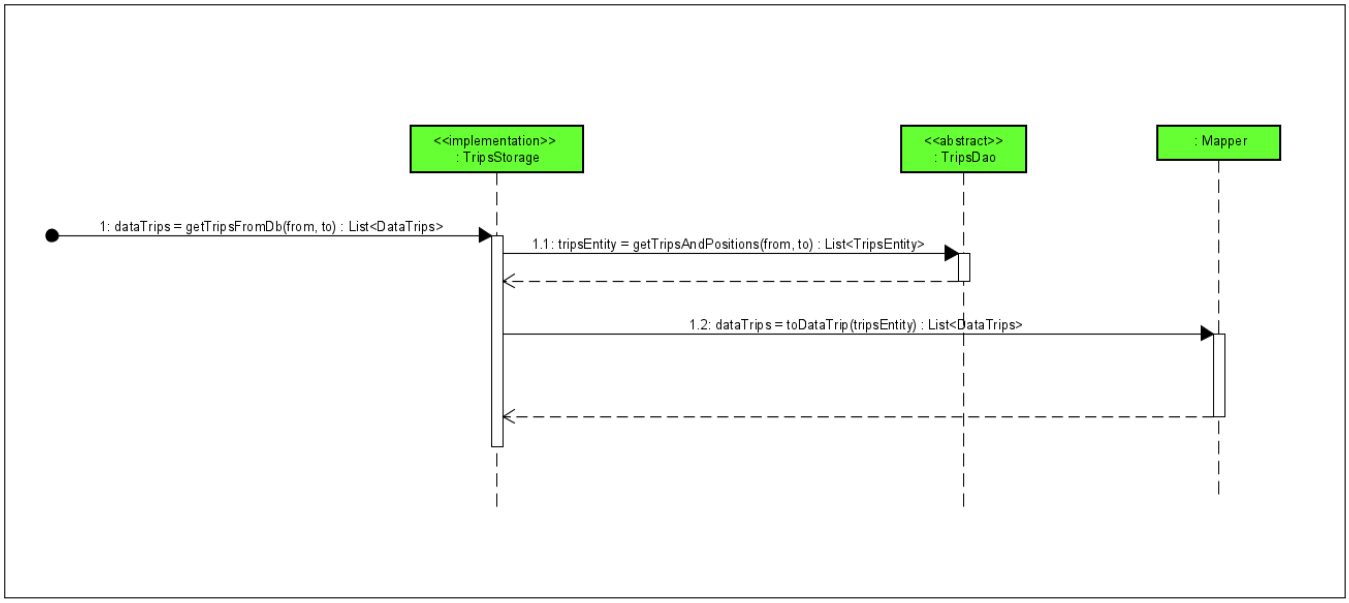

Figura 5.44: Diagrama de secuencia de la capa "Local" del caso de uso "Mostrar viajes".

A continuación se detallan los diagramas de secuencia que representan un caso de uso genérico en el que se ha producido un error, ya sea a nivel local al guardar o recuperar datos de la Base de datos o remoto, producido en un petición a la API REST, ambos errores son llevados sin tratar hasta el "ViewModel" que ejecuto el caso de uso y será él el encargado de gestionarlo. Cualquier error producido en el "ViewModel" sera gestionado por él mismo de la misma forma.

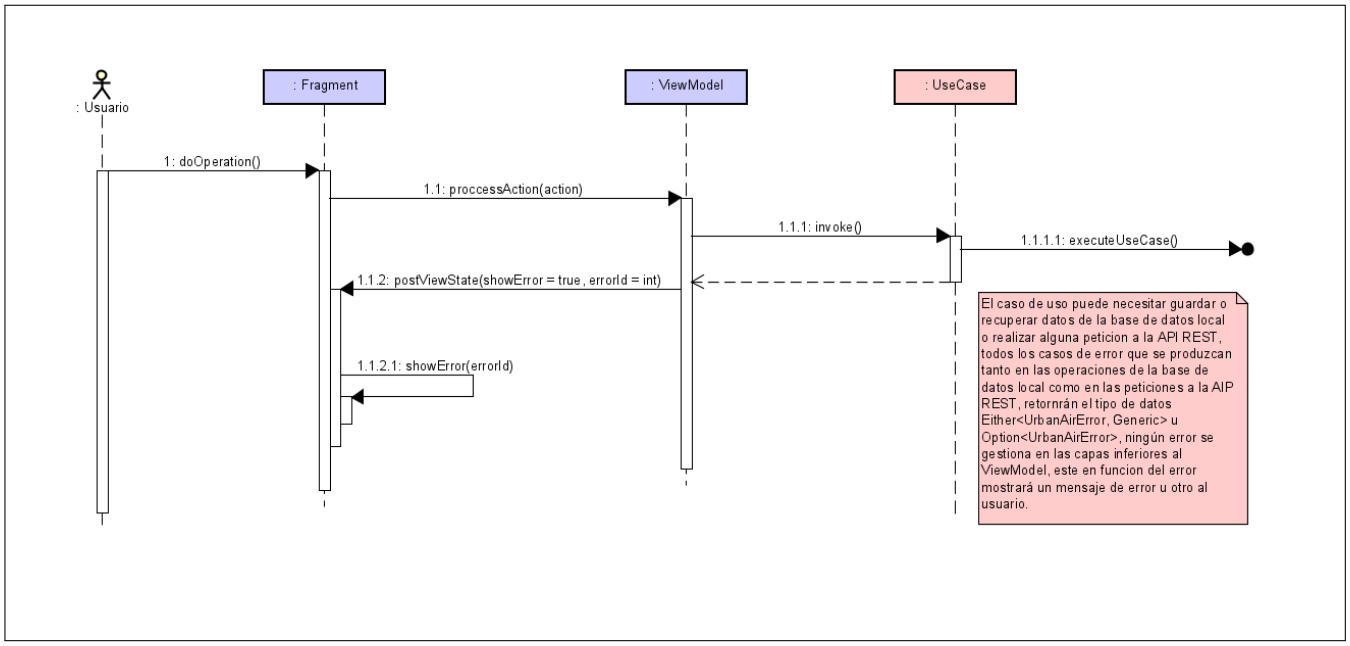

Figura 5.45: Diagrama de secuencia genérico de la gestión de errores producidos en las capas "Local" o "Remote".

5.4. Modelo lógico de datos en diseño

A continuación, se muestra el esquema de la base de datos finalizada en la fase de diseño.

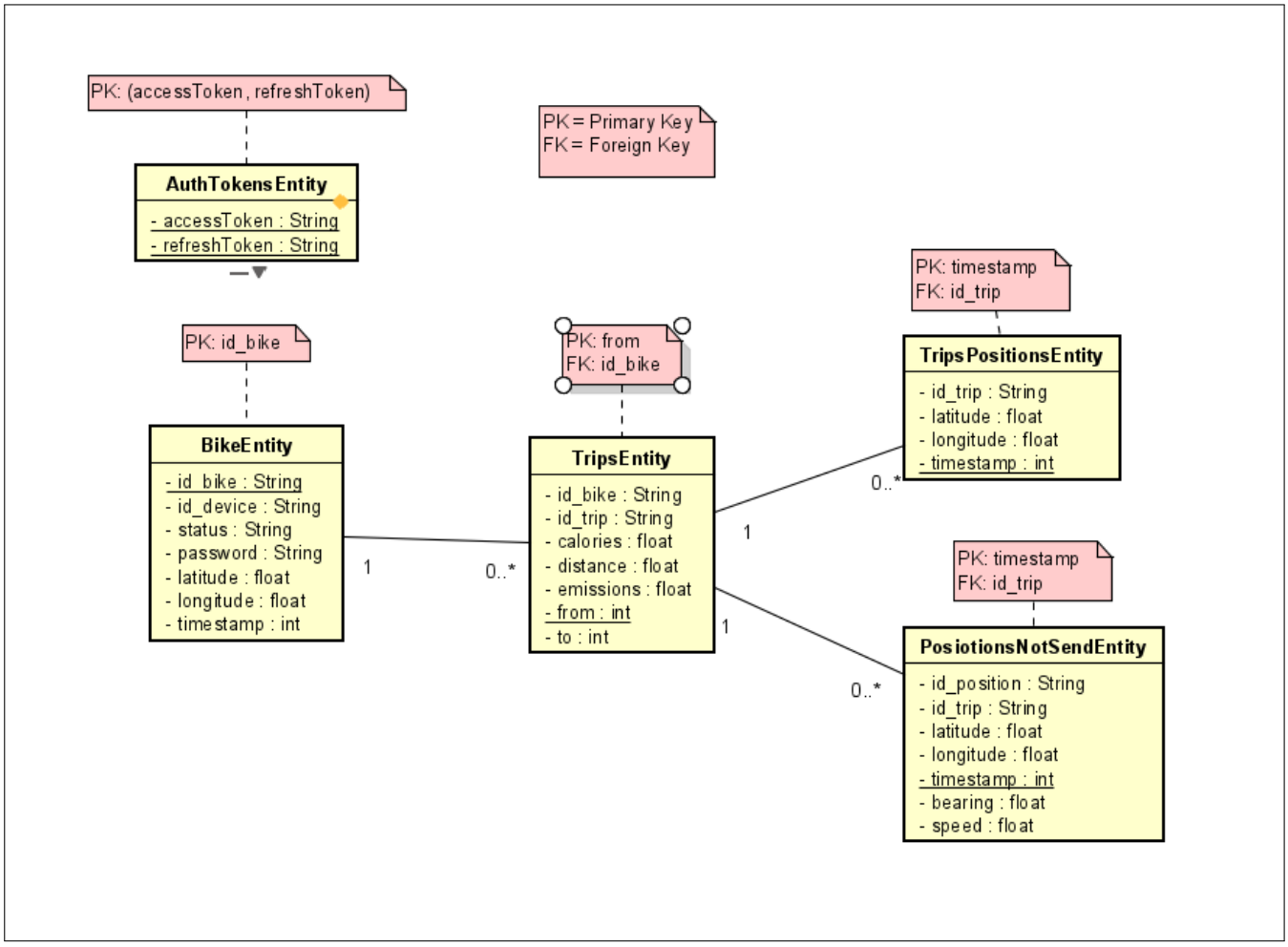

Figura 5.46: Modelo de la base de datos en la fase de diseño.

Capítulo 6

Implementación

En el siguiente capítulo se detallan cuestiones prácticas que se consideran relevantes sobre la implementación. En el siguiente diagrama, figura 6.1 se detalla el diagrama de despliegue del sistema, usando un color **granate** para destacar los paquetes en los cuales no se ha trabajado en este proyecto y un color **gris** para el paquete desarrollado en este TFG.

6.1. Diagrama de despliegue

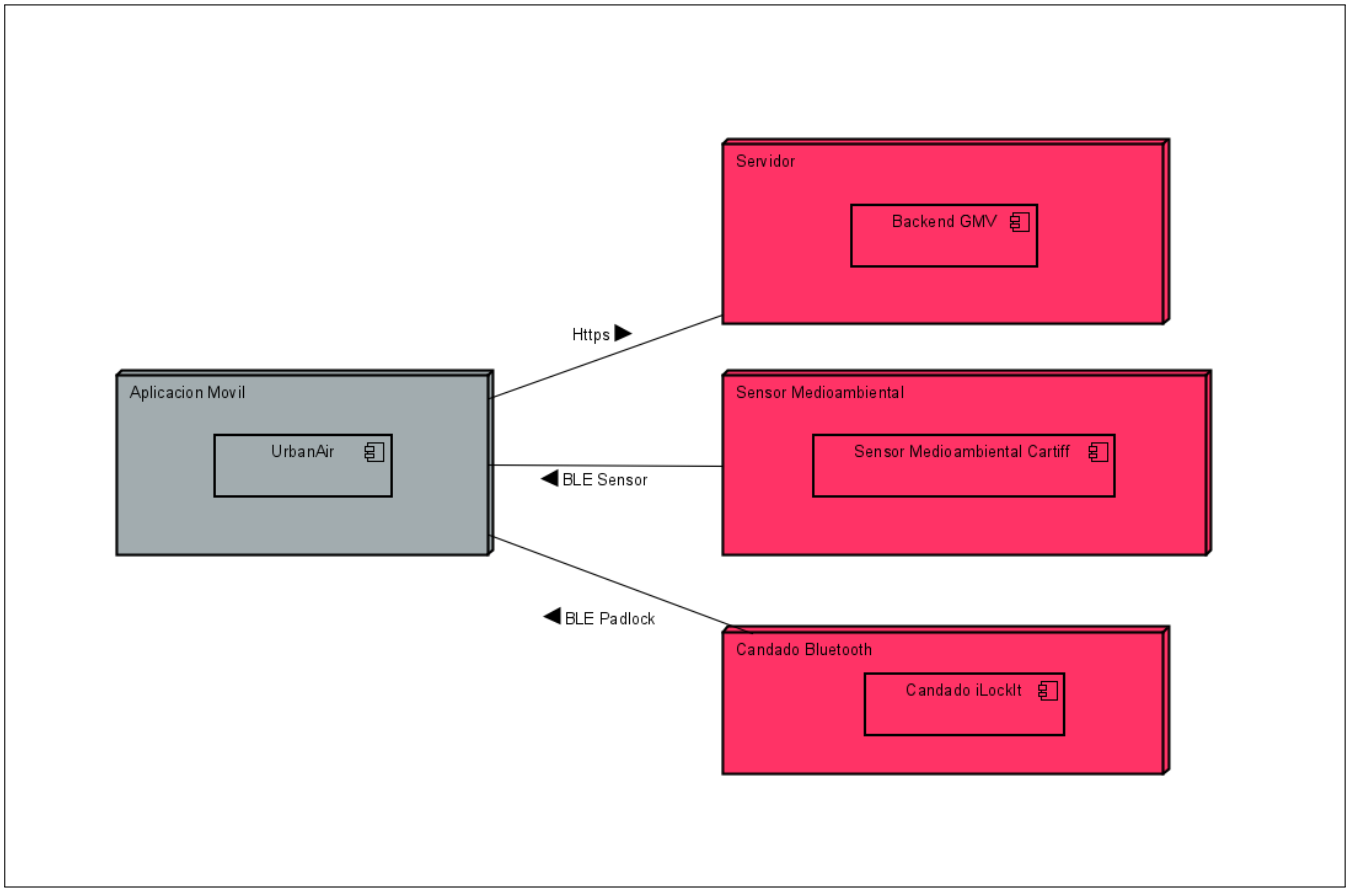

Figura 6.1: Diagrama de despliegue de la aplicación.

6.2. Entorno de desarrollo

Para el desarrollo del la aplicación móvil se ha utilizado el IDE Android Studio [19], basado en el software IntelliJ IDEA de JetBrains [35] y ha sido publicado de forma gratuita a través de la Licencia Apache 2.0. Está disponible para las plataformas Microsoft Windows, macOS y GNU/Linux. Ha sido diseñado específicamente para el desarrollo de Android. En concreto se ha utilizado la versión 4.1 Rc 2, que es la última release disponible a fecha 12/09/2020. [19].

Como sistema operativo se comenzó el desarrollo en macOS Catalina Versión 10.15.3 y después se continuó y finalizó en Windows 10.

6.3. Versiones necesarias de software

Para hacer uso de la aplicación, solo es necesario un smartphone que posea las siguientes características mínimas:

- Sistema Operativo Android
- Versión mínima de Android 5.0 Lollipop
- Conexión a Internet
- Bluetooth
- Ubicación GPS

6.4. Herramientas utilizadas

Durante el desarrollo se han utilizado las siguientes herramientas extras:

- **Postman**: Postman [30] es una herramienta que permite configurar y realizar peticiones a un servidor mediante una URL, en este caso se ha utilizado para poder probar y depurar las llamadas de red realizadas a la API REST, también permite ver que devuelven y en que formato. Una vez testeadas se han modelado y programado en la aplicación, al haberlo hecho de esta manera ha sido relativamente sencillo realizar toda la conexión con la API REST y apenas han surgido errores.
- **Inkscape**: Inkscape [6] es un editor de gráficos vectoriales libre y de código abierto. Inkscape puede crear y editar diagramas, líneas, gráficos, logotipos, e ilustraciones complejas. Se ha utilizado para modificar algunos iconos e imágenes obtenidos de internet ([25] y [32]) ajustándolos a las necesidades de la aplicación, ya fuese modificando el color, el tamaño o la forma. Otros recursos gráficos han sido diseñados directamente por GMV [5].

6.5. Control de versiones

Como herramienta de control de versiones se ha utilizado Git, haciendo uso de un repositorio basado en GitLab [23] perteneciente a GMV [4]. Para hacer más fácil el uso del repositorio se ha utilizado también la herramienta Sourcetree de Atlassian [9], esta herramienta otorga una sencilla interfaz gráfica para gestionar el repositorio y realizar cualquier operación en el como un commit, un push, un merge, étc... de forma sencilla e intuitiva.

6.6. Implementación de la base de datos

Para la implementación de la base de datos se ha utilizado la librería Room [13] de Android, esta librería facilita mucho la implementación de las operaciones CRUD ya que se encarga de generar las clases necesarias a partir de las anotaciones establecidas en cada clase.

Hay 3 componentes principales en Room [13]:

- **Database**

  Sirve como el punto de acceso principal para la conexión entre la aplicación y los datos persistentes de la base de datos. La clase anotada con @Database debe cumplir las siguientes condiciones:

  - Ser una clase abstracta que extienda "RoomDatabase".
  - Incluir la lista de entidades, clases anotadas con @Entity, asociadas con la base de datos.
  - Implementar un método abstracto con 0 argumentos y que devuelve las clase anotadas con @Dao.

- **Entidad**: Representa una tabla perteneciente a la base de datos
- **Dao**: Contiene los métodos que se usan para acceder a los datos de la base de datos.

La aplicación utiliza la base de datos de Room para obtener los objetos de acceso a datos, o DAOs, asociados con la base de datos. Luego, la aplicación usa cada DAO para obtener las entidades de la base de datos y guardar cualquier cambio en esas entidades en la base de datos. Finalmente, la aplicación usa las entidades para obtener y establecer los valores que corresponden a las columnas de la tabla dentro de la base de datos.

Capítulo 7

Plan de pruebas y evaluación

En este capítulo se detallan las pruebas a realizar para comprobar el correcto funcionamiento de las funcionalidades implementadas, comprobando que cumpla los requisitos establecidos.

Por un lado, se realizarán pruebas unitarias de caja negra sobre la API REST; y posteriormente se realizarán pruebas unitarias y de integración sobre la aplicación. Finalmente se realizarán pruebas de sistema para comprobar el correcto funcionamiento en el entorno de producción.

7.0.1. Pruebas sobre el paquete Login

En este caso, se ha de probar el correcto funcionamiento de todas las funcionalidades de la vista de "Login". En este caso, se tratarán los requisitos RF-1 y RF-2.

Prueba 1.1	
Descripción	Esta prueba está destinada a probar el funcionamiento del inicio de sesión
Acción	Introducir el nombre de usuario y contraseña y pulsar el botón de "Iniciar sesión".
Resultado Esperado	Se inicia la sesión y se navega a la vista principal de la aplicación.

Prueba 1.2	
Descripción	Esta prueba está destinada a probar el funcionamiento de la recuperación de la contraseña del usuario.
Acción	Pulsar el botón de recuperar contraseña, introducir el nombre de usuario y pulsar el botón de confirmar.
Resultado Esperado	Se recibe un correo electrónico con las instrucciones para recuperar la contraseña.

7.0.2. Pruebas sobre el paquete Home

En este caso, se ha de probar el correcto funcionamiento de todas las funcionalidades de la vista de "Home". En este caso, se tratarán los requisitos RF-3, RF-4, RF-5, RF-6, RF-7, RF-8, RF-9, RF-10, RF-11, RF-12, RF-13, RF-14, RF-15 y RF-16.

Prueba 2.1	
Descripción	Esta prueba está destinada a probar el funcionamiento de la obtención de la información sobre la distancia recorrida en el día actual
Acción	Completar uno o varios viajes.
Resultado Esperado	Se obtiene la suma total de las distancias de todos los viajes realizados en el día actual.

Prueba 2.2	
Descripción	Esta prueba está destinada a probar el funcionamiento de la obtención de la información sobre las emisiones evitadas en el día actual
Acción	Completar uno o varios viajes.
Resultado Esperado	Se obtiene la suma total de las emisiones evitadas de todos los viajes realizados en el día actual.

Prueba 2.3	
Descripción	Esta prueba está destinada a probar el funcionamiento de la comprobación de la disponibilidad de la conexión con el candado bluetooth.
Acción	Activar el bluetooth y la ubicación del dispositivo móvil y navegar a la vista principal.
Resultado Esperado	El icono de la bicicleta cambia a color verde y se muestra el texto de "Conexión disponible".

Prueba 2.4	
Descripción	Esta prueba está destinada a probar el funcionamiento de la comprobación del nivel de batería del candado bluetooth.
Acción	Abrir o cerrar la bicicleta.
Resultado Esperado	El nivel de batería del candado se muestra actualizado.

Prueba 2.5	
Descripción	Esta prueba está destinada a probar el funcionamiento de la localización de la bicicleta.
Acción	Navegar a la vista principal.
Resultado Esperado	El vehículo se muestra en el mapa en la última posición recibida por el servidor.

Prueba 2.6	
Descripción	Esta prueba está destinada a probar el funcionamiento de la localización del usuario en el mapa.
Acción	Navegar a la vista principal.
Resultado Esperado	Se muestra y actualiza la posición actual del usuario en el mapa.

Prueba 2.7	
Descripción	Esta prueba está destinada a probar el funcionamiento de la apertura del candado y el comienzo de un nuevo viaje.
Acción	Presionar el botón de "Iniciar Viaje".
Resultado Esperado	Se abre el candado, se cambia la vista a la vista de viaje y se comienzan a guardar los datos del viaje.

Prueba 2.8	
Descripción	Esta prueba está destinada a probar el funcionamiento del cierre del candado y la finalización del viaje actual.
Acción	Presionar el botón de "Iniciar Viaje".
Resultado Esperado	Se cierra el candado, se cambia la vista a la vista principal y el viaje queda registrado.

Prueba 2.9	
Descripción	Esta prueba está destinada a probar el funcionamiento de la conexión con el sensor medioambiental.
Acción	Presionar el botón de "Iniciar Viaje".
Resultado Esperado	Una vez iniciado el viaje se muestra la vista de "Datos medioambientales" que muestra los valores del sensor en tiempo real.

Prueba 2.10	
Descripción	Esta prueba está destinada a probar el funcionamiento del cálculo del tiempo de viaje.
Acción	Presionar el botón de "Iniciar Viaje".
Resultado Esperado	Una vez iniciado el viaje se actualiza el contador de tiempo de viaje cada segundo.

Prueba 2.11	
Descripción	Esta prueba está destinada a probar el funcionamiento del cálculo de la distancia recorrida durante el viaje.
Acción	Presionar el botón de "Iniciar Viaje".
Resultado Esperado	Una vez iniciado el viaje se actualiza el contador de distancia recorrida.

Prueba 2.12	
Descripción	Esta prueba está destinada a asegurar que el usuario esta utilizando su bicicleta y no otra.
Acción	Presionar el botón de "Iniciar Viaje".
Resultado Esperado	Una vez iniciado el viaje se muestra el ID de la bicicleta con la que se está viajando.

Prueba 2.12	
Descripción	Esta prueba está destinada a probar la restricción de las funcionalidades de la aplicación mientras el usuario viaja.
Acción	Presionar el botón de "Iniciar Viaje".
Resultado Esperado	Se ocultan todos los elementos de la pantalla excepto la vista de información y los botones de abrir y cerrar. Se desactivan todas las funcionalidades excepto las de abrir y cerrar la bicicleta.

7.0.3. Pruebas sobre el paquete "Calendar"

En este caso, se ha de probar el correcto funcionamiento de todas las funcionalidades de la vista de "Calendar". En este caso, se tratarán los requisitos RF-17, RF-18, RF-19, RF-20, RF-21, RF-22, RF-23, RF-24 y RF-25.

Prueba 3.1	
Descripción	Esta prueba está destinada a probar el funcionamiento de la obtención de los viajes registrados en una fecha concreta.
Acción	Seleccionar una fecha del calendario desplegable.
Resultado Esperado	Se obtiene una lista con los viajes registrados en la fecha seleccionada.

Prueba 3.2	
Descripción	Esta prueba está destinada a probar el funcionamiento del cambio de fecha seleccionada
Acción	Cambiar la fecha seleccionada en el calendario desplegable.
Resultado Esperado	Se obtienen los viajes registrados en la nueva fecha seleccionada.

Prueba 3.3	
Descripción	Esta prueba está destinada a probar el funcionamiento de la muestra de viajes en el calendario.
Acción	Desplegar el calendario.
Resultado Esperado	Se marcan los días que tengan viajes registrados con un punto verde debajo de la fecha.

Prueba 3.4	
Descripción	Esta prueba está destinada a probar el funcionamiento de la obtención de las calorías quemadas durante cada viaje.
Acción	Seleccionar una fecha con viajes registrados.
Resultado Esperado	Se muestra la cantidad de calorías quemadas en cada viaje de la fecha seleccionada.

Prueba 3.5	
Descripción	Esta prueba está destinada a probar el funcionamiento de la obtención de las emisiones evitadas durante cada viaje.
Acción	Seleccionar una fecha con viajes registrados.
Resultado Esperado	Se muestra la cantidad de emisiones evitadas en cada viaje de la fecha seleccionada.

Prueba 3.6	
Descripción	Esta prueba está destinada a probar el funcionamiento de la obtención de la distancia recorrida durante cada viaje.
Acción	Seleccionar una fecha con viajes registrados.
Resultado Esperado	Se muestra la distancia recorrida en cada viaje de la fecha seleccionada.

Prueba 3.7	
Descripción	Esta prueba está destinada a probar el funcionamiento de la obtención de la hora de inicio de cada viaje.
Acción	Seleccionar una fecha con viajes registrados.
Resultado Esperado	Se muestra la hora de inicio de cada viaje de la fecha seleccionada.

Prueba 3.8	
Descripción	Esta prueba está destinada a probar el funcionamiento de la obtención de la hora de fin de cada viaje.
Acción	Seleccionar una fecha con viajes registrados.
Resultado Esperado	Se muestra la hora de fin de cada viaje de la fecha seleccionada.

Prueba 3.8	
Descripción	Esta prueba está destinada a probar el funcionamiento de la obtención de la ruta recorrida de cada viaje.
Acción	Seleccionar un viaje de la lista de viajes.
Resultado Esperado	Se dibuja en el mapa el punto de inicio de la ruta, la ruta recorrida y el punto de fin de la ruta.

7.0.4. Pruebas sobre el paquete "Leaf"

En este caso, se ha de probar el correcto funcionamiento de todas las funcionalidades de la vista de "Leaf". En este caso, se tratarán los requisitos RF-26, RF-27, RF-28, RF-29 y RF-30.

Prueba 4.1	
Descripción	Esta prueba está destinada a probar el funcionamiento de la obtención de los datos medioambientales y su posterior dibujo en el mapa.
Acción	Navegar a la vista "Leaf"
Resultado Esperado	Se obtienen los datos medioambientales y se pintan en el mapa.

Prueba 4.2	
Descripción	Esta prueba está destinada a probar el funcionamiento de los códigos de color en función del valor de los datos medioambientales.
Acción	Navegar a la vista "Leaf"
Resultado Esperado	Se pintan los valores medioambientales en color verde, amarillo o rojo en función de su valor.

Prueba 4.3	
Descripción	Esta prueba está destinada a probar el funcionamiento del cambio del tipo de partículas que se muestran en el mapa.
Acción	Cambiar el tipo de partículas seleccionado.
Resultado Esperado	Se actualizan los colores del mapa con los nuevos valores de las partículas elegidas.

Prueba 4.4	
Descripción	Esta prueba está destinada a probar el funcionamiento de la muestra de los valores mínimos, máximos y medios de las partículas medidas.
Acción	Desplegar la vista de información en la vista "Leaf".
Resultado Esperado	Se muestran los valores mínimos de las partículas CO, CO2, NOx y Pm.

Prueba 4.5	
Descripción	Esta prueba está destinada a probar el funcionamiento del cambio de fechas para la muestra de datos medioambientales mínimos, máximos y medios.
Acción	Cambiar las fechas seleccionadas de inicio y fin.
Resultado Esperado	Se actualizan los valores medios, mínimos y máximos de las partículas.

7.0.5. Pruebas sobre el paquete "Profile"

En este caso, se ha de probar el correcto funcionamiento de todas las funcionalidades de la vista de "Profile". En este caso, se tratarán los requisitos RF-31, RF-32, RF-33, RF-34, RF-35, RF-36 y RF-37.

Prueba 5.1	
Descripción	Esta prueba está destinada a probar el funcionamiento de la obtención de datos del perfil de usuario.
Acción	Navegar a la vista "Profile".
Resultado Esperado	Se muestran los datos del perfil del usuario.

Prueba 5.2	
Descripción	Esta prueba está destinada a probar el funcionamiento de la modificación de datos del perfil de usuario.
Acción	Presionar el botón de "Editar perfil".
Resultado Esperado	Se permite editar la altura y el peso y guardar los cambios.

Prueba 5.3	
Descripción	Esta prueba está destinada a probar el funcionamiento de la muestra de la política de privacidad de la aplicación.
Acción	Presionar el botón "Política de privacidad".
Resultado Esperado	Se muestra el texto que contiene la política de privacidad.

Prueba 5.3	
Descripción	Esta prueba está destinada a probar el funcionamiento de la muestra de las atribuciones de la aplicación.
Acción	Presionar el botón "Atribuciones".
Resultado Esperado	Se muestra el texto que contiene la política de privacidad.

Prueba 5.4	
Descripción	Esta prueba está destinada a probar el funcionamiento del cambio de contraseña.
Acción	Presionar el botón "Cambiar contraseña".
Resultado Esperado	Se envía un correo electrónico al usuario con las instrucciones para cambiar su contraseña.

Prueba 5.4	
Descripción	Esta prueba está destinada a probar el funcionamiento del cierre de sesión.
Acción	Presionar el botón "Cerrar sesión".
Resultado Esperado	Se cierra la sesión del usuario.

Después haber visto el plan de pruebas, el siguiente capítulo detallará el manual de instalación.

Capítulo 8

Manual de instalación

La aplicación móvil *UrbanAir* se puede instalar de varias formas distintas, desde Google Play Store, desde Android Studio si se tiene acceso al proyecto y al código fuente o directamente se puede instalar el archivo ".apk" si se tiene acceso a él.

8.1. Google Play Store

La instalación desde Google Play Store es muy sencilla, es exactamente igual que la instalación de cualquier otra aplicación. Se puede encontrar buscando "UrbanAIR" en el cuadro de búsqueda, o por el fabricante "GMV Sistemas"

El único inconveniente es que la aplicación se continuará actualizando con mejoras y corrección de posibles errores, esto puede provocar que en un futuro la aplicación sea ligeramente diferente a la aplicación descrita en este TFG, la versión actual de este TFG es la versión 2.0.8

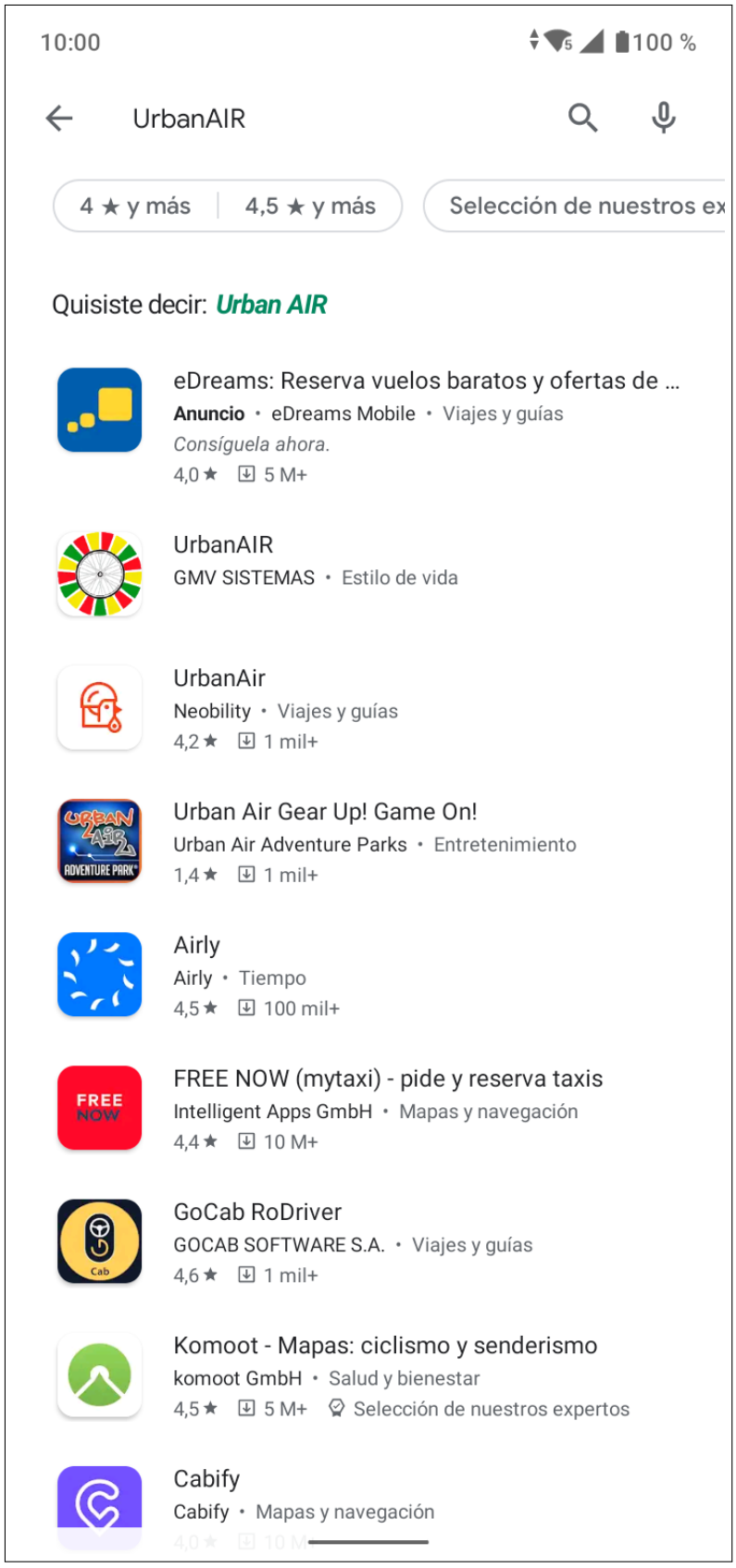

Figura 8.1: Búsqueda de la aplicación en Google Play Store por su nombre.

Figura 8.2: Búsqueda de la aplicación en Google Play Store por su desarrollador.

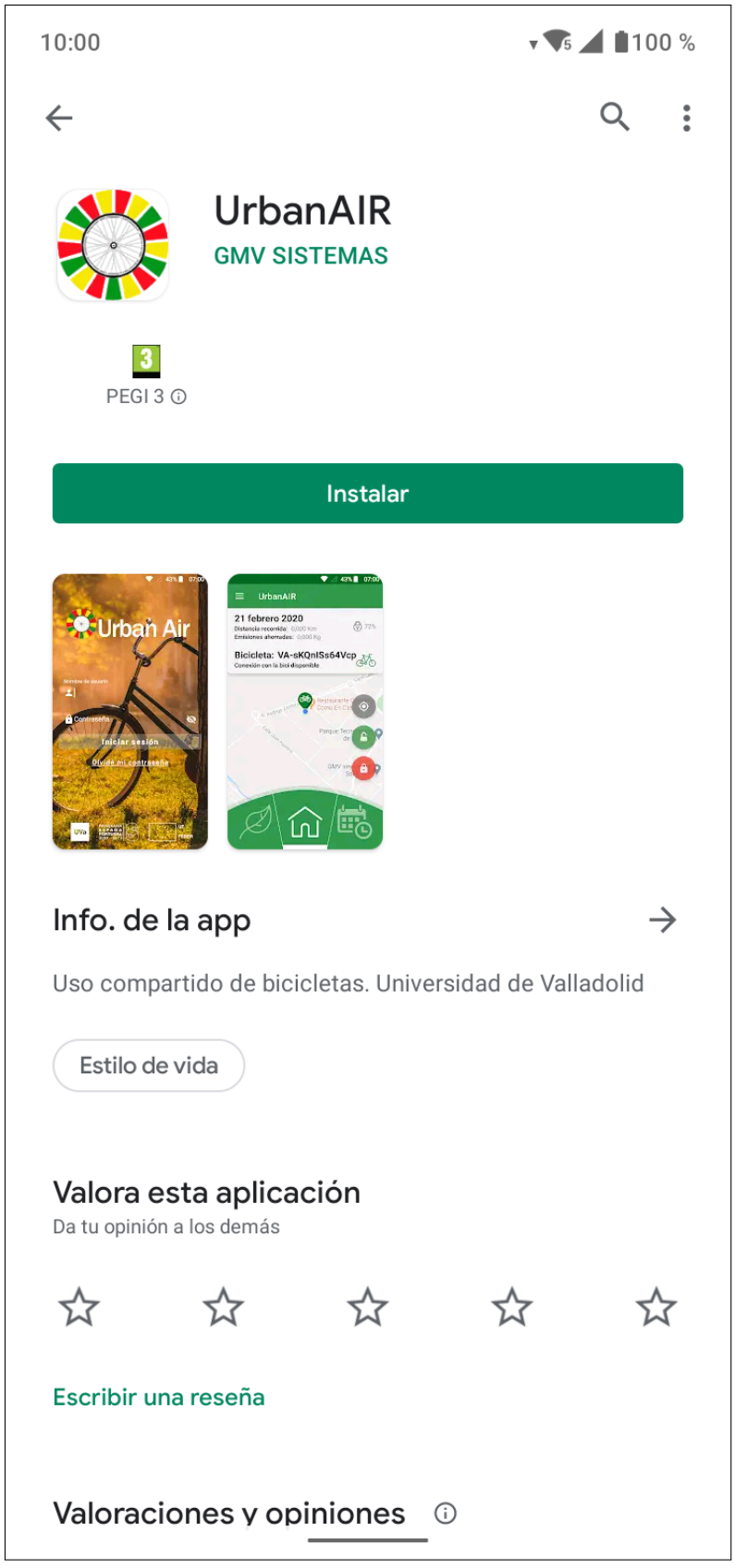

Figura 8.3: Ficha de la aplicación en Google Play Store.

8.2. Android Studio

Si se tiene acceso al código fuente del proyecto se puede instalar y ejecutar desde Android Studio, primero hay que conectar el dispositivo móvil en donde se quiere ejecutar la aplicación al ordenador.

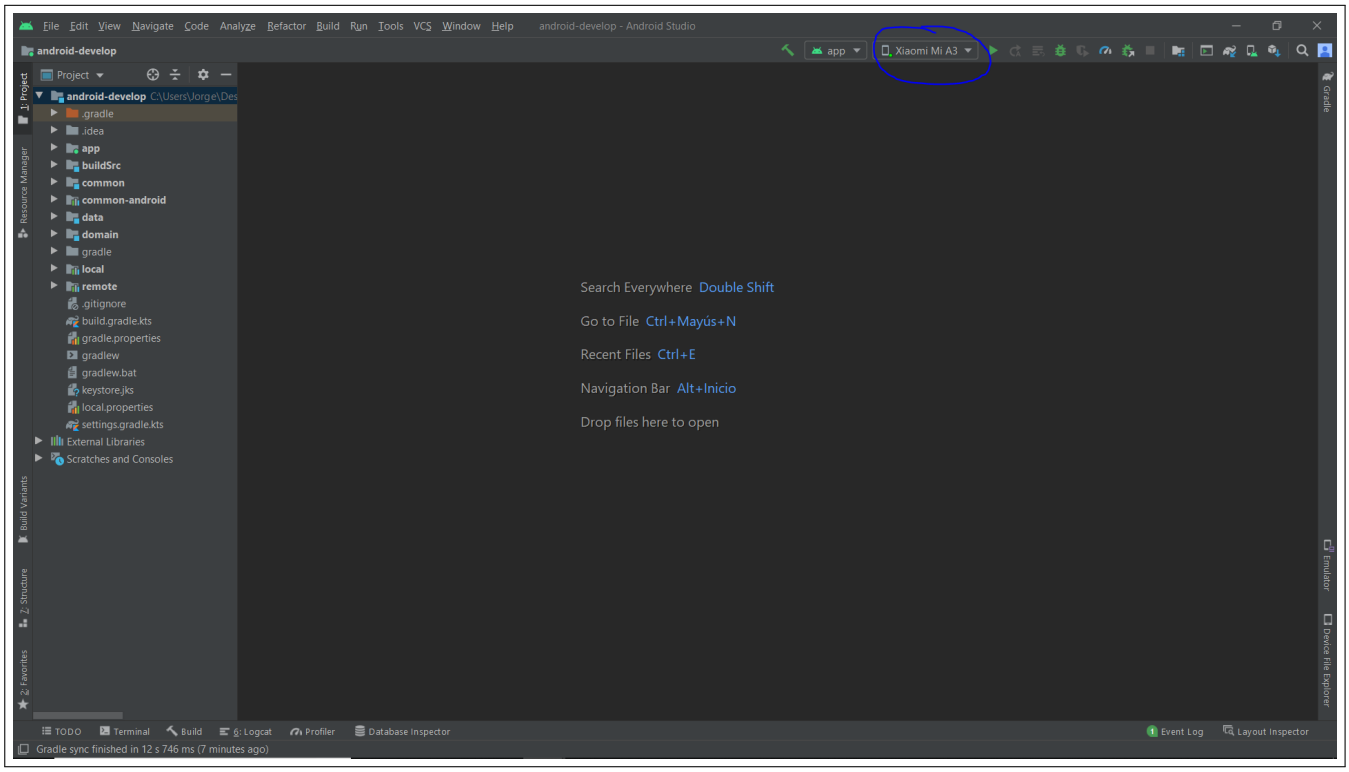

Figura 8.4: Dispositivo móvil conectado al ordenador.

Una vez conectado el dispositivo hay que pulsar el icono de "play" de color verde, el IDE instalará y ejecutará la aplicación en el dispositivo seleccionado.

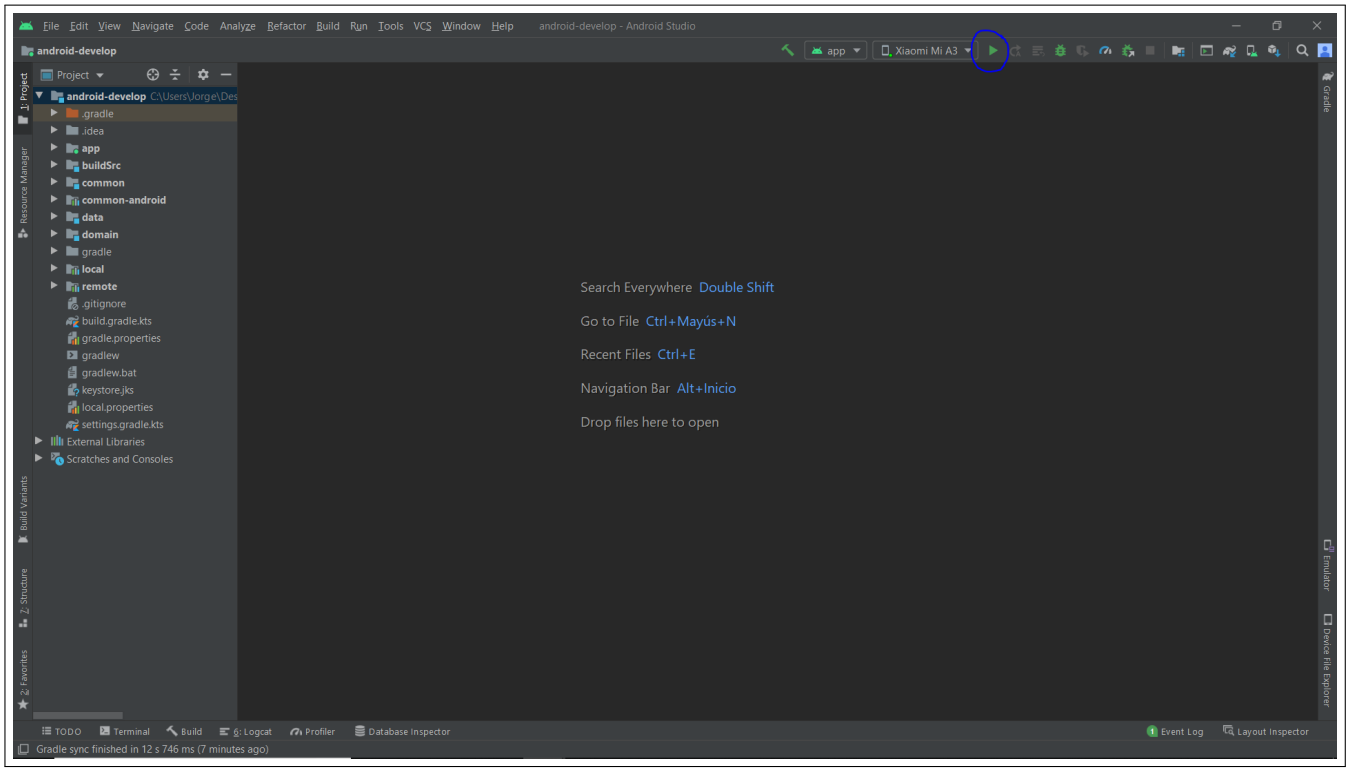

Figura 8.5: Dispositivo móvil conectado al ordenador.

8.3. Archivo ".apk"

Si se posee el archivo ".apk" que corresponde a la aplicación compilada en el dispositivo móvil sólo hay que seguir las instrucciones de instalación que nos indique nuestro teléfono.

Después de detallar en este capítulo el manual de instalación, el siguiente capítulo detallará el manual del programador.

Capítulo 9

Manual de programador

En este capítulo se detallan aquellas cuestiones necesarias para un programador que quisiera extender o modificar las funcionalidades implementadas.

9.1. Organización general de la aplicación

La aplicación es un proyecto Android divido en varios módulos para reorganizar mejor el código. Como se puede observar a continuación, tenemos los módulos de app, buildSrc, common, common-android, data, domain, local y remote.

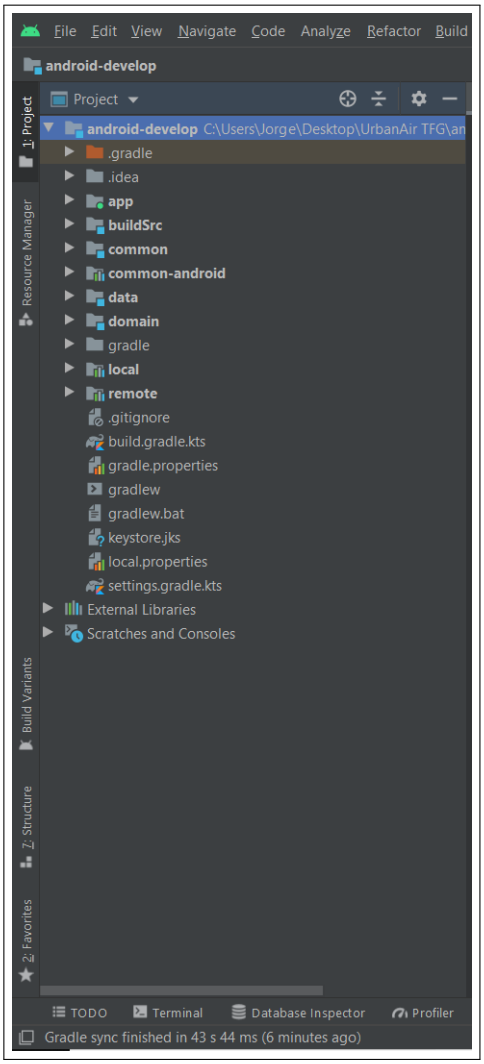

Figura 9.1: Vista de directorios del proyecto UrbanAir.

9.2. Módulo App

En este módulo podemos observar varios subdirectorios, a continuación se explican los más relevantes:

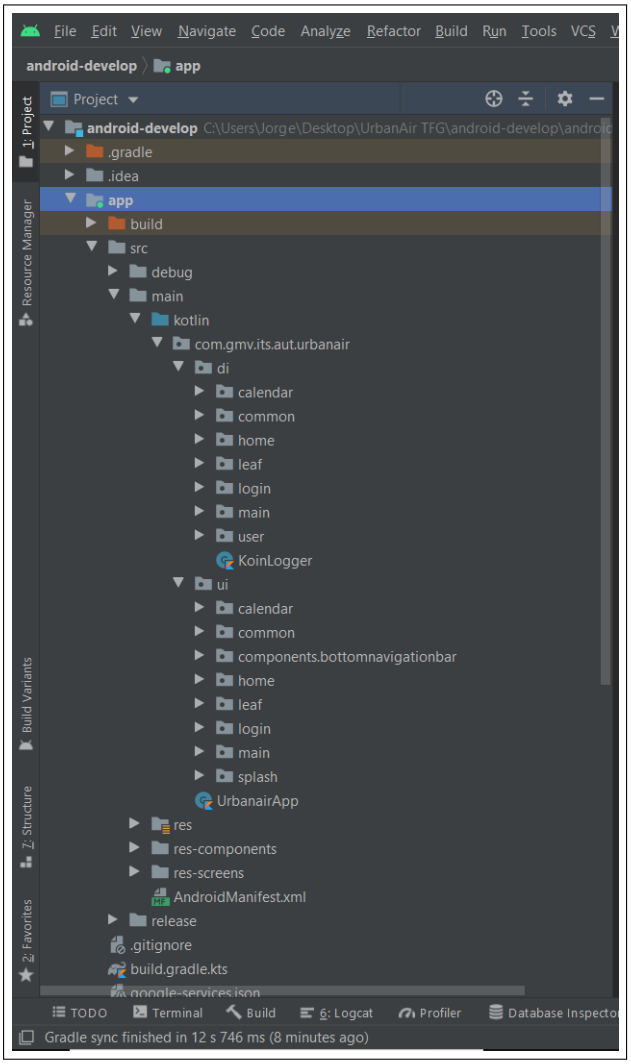

Figura 9.2: Vista de directorios del módulo App parte 1.

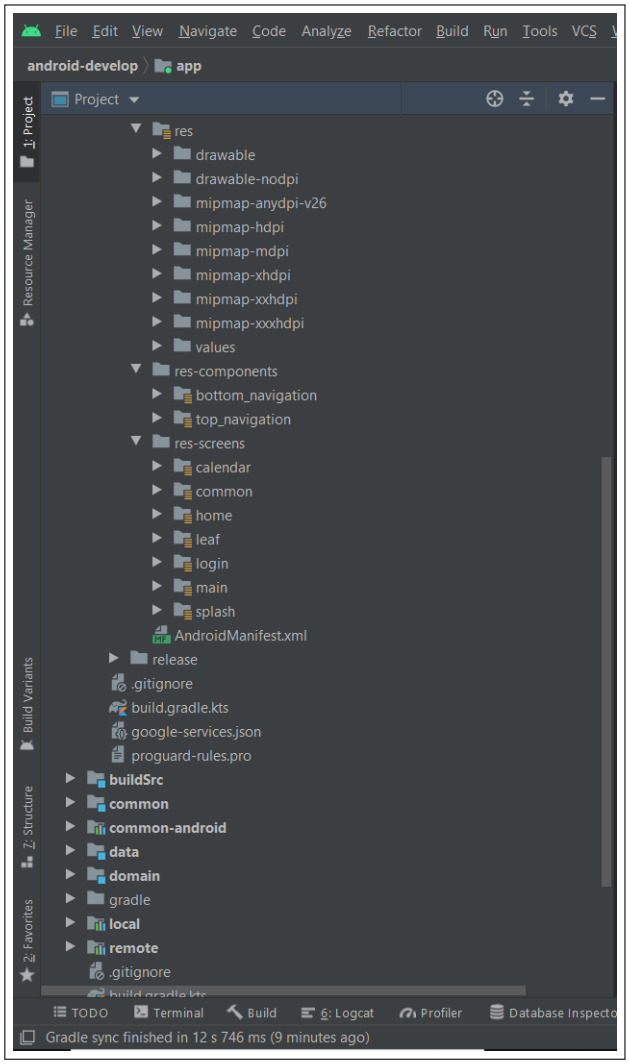

Figura 9.3: Vista de directorios del módulo App parte 2.

9.2.1. Di

Este directorio contiene todo lo necesario para gestionar la inyección de dependencias, cada paquete tiene un fichero con las que necesita y como obtenerlas. La gran mayoría de estas dependencias son los ViewModel y los UseCase de cada una de las vistas, también están todas las dependencias necesarias para los repositorios, la base de datos y la conexión con la API REST.

El fichero "KoinLogger.kt" es para poder usar el sistema de logs que aporta la librería "Koin" encargada de gestionar todas las inyecciones de dependencias.

9.2.2. Ui

Este directorio contiene toda las vistas y su lógica, cada vista esta representada por un Fragment, tiene su ViewModel asociado y sus clases que definen los ViewStates y las ViewActions que se pueden efectuar. El Fragment recoge las acciones ejecutadas por el usuario como la entrada de texto, la pulsación de un botón, étc..., y se las envía al ViewModel en forma de ViewActions. El ViewModel ejecuta la lógica necesaria o el caso de uso correspondiente y cambia el estado de la vista mediante los ViewStates, la vista reacciona y ejecuta la lógica necesaria en función del estado en que se encuentre. Las vistas están agrupadas por paquetes y el fichero "UrbanAirApp.kt" es el que se encarga de cargar todas las dependencias previamente definidas en el paquete anterior.

9.2.3. Res

Este directorio contiene los elementos gráficos principales de la aplicación como el icono de la misma, el archivo de colores, el nombre de la aplicación y el fichero principal de temas y estilos.

9.2.4. Res-components

En este directorio tenemos los dos componentes comunes en todas las vistas de la app, la barra inferior de navegación y la barra superior de navegación:

- **Bottom_Navigation**
  En este subdirectorio se encuentra el layout, los iconos y los tamaños definidos para la barra inferior de navegación de la aplicación.

- **Top_Navigation**
  En este subdirectorio se encuentran el panel desplegable, los iconos y los tamaños definidos para la barra superior de navegación de la aplicación.

9.2.5. Res-screens

En este directorio tenemos todos los iconos de cada vista, los layout, los tamaños, los valores de las cadenas de texto que aparecen en cada vista tanto en Español como en Inglés y los elementos extras que tenga cada vista como por ejemplo animaciones o menús. Todos estos elementos están organizados por paquetes según cada vista.

9.2.6. Módulo BuildSrc

Este directorio tiene un único fichero, "Dependencies.kt", que contiene todas las librerías utilizadas por la aplicación y la versión de cada una que se está utilizando y los plugins necesarios para poder compilar y ejecutar la aplicación. No implementa ninguna funcionalidad más allá de ser un punto común donde enumerar y localizar todas las librerías externas necesarias de la aplicación.

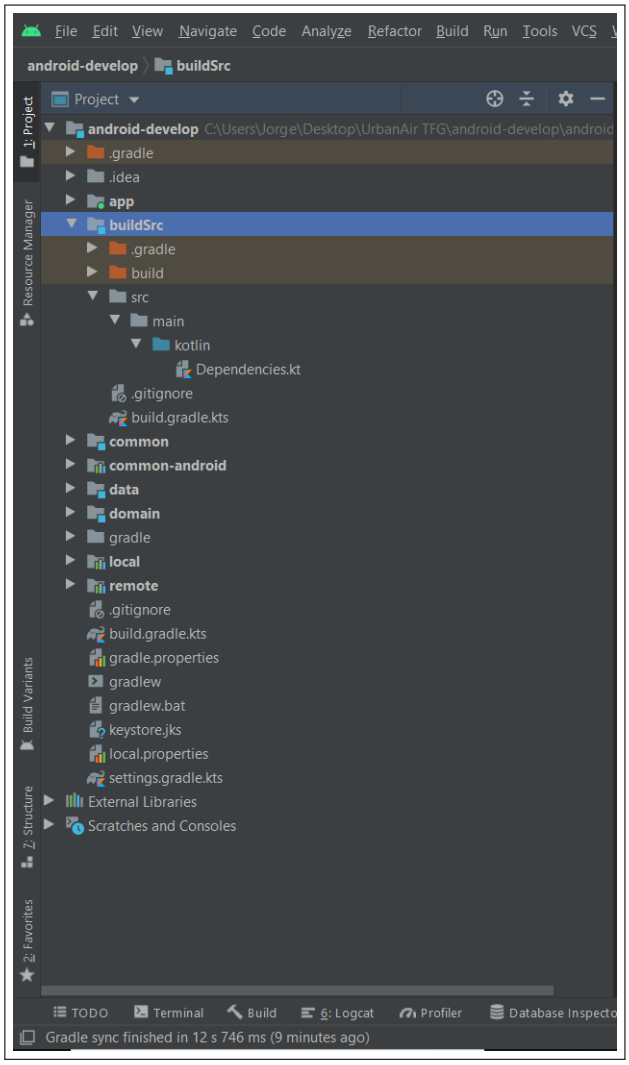

Figura 9.4: Vista de directorios del módulo BuildSrc.

9.2.7. Módulos Common y Common-Android

Estos directorios contienen código que es utilizado por los demás módulos, este código son funciones de utilidad, extensiones de tipos y constantes, la diferencia entre ambos es que las funciones definidas en "Common-Android" son las que utilizan el Activity y los Fragments del módulo "App" mientras que las funciones de "Common" son utilizadas por varios de los módulos restantes.

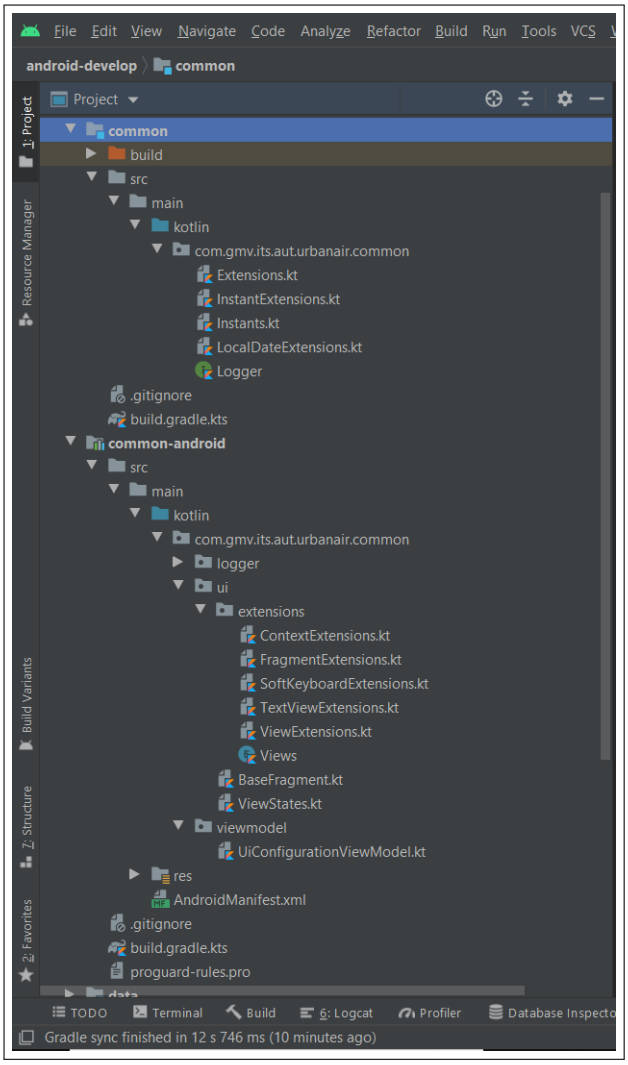

Figura 9.5: Vista de directorios delos módulos Common y Common-Android.

9.2.8. Módulos Domain y Data

Estos directorios contienen el código que corresponde a las reglas de negocio, implementación de casos de uso, representación del modelo de dominio, los errores que maneja el sistema y el acceso a los repositorios ya sean locales o remotos.

- **Domain Entity**
  Contiene las clases que representan las entidades de los objetos que son utilizadas por la capa superior de las vistas, puede que en algunos casos estos objetos no sean iguales a los del paquete "Data Entity". Hay que aclarar que en el paquete "Data" se manejan las entidades tal y como son obtenidas de la base de datos o de la API REST, es en este punto donde quizá algún campo de los objetos se elimine, se cambie el tipo de datos o incluso varios campos se combinen en uno solo, por ejemplo al obtener el perfil del usuario hay un campo que es primer apellido y otro que es segundo apellido, aquí se combinan en uno solo que es apellidos para que sea más sencillo de manejar desde las vistas a la hora de mostrar información.

- **Domain Error**
  Contiene los errores que gestionará la aplicación, se diferencia del paquete "Data Error" en que los errores obtenidos de la base de datos o de la API REST son unificados bajo un mismo tipo "UrbanAirError" para que sea mas sencillo manejarles en la capa superior donde están las vistas.

- **Domain Repository**
  Contiene las interfaces que van a definir como se hará el acceso a la capa de datos, que funciones existirán, con que parámetros y que objetos devolverá cada una de ellas, la implementación se hará en el paquete "Data".

- **Domain UseCase**
  Contiene las clases que representan los casos de uso de la aplicación que permiten satisfacer los requisitos previamente enumerados. Es el punto de unión entre la capa que contiene las vistas y la capa de datos.

- **Data DataStore**
  Contiene las interfaces que deberán implementar las clases cuya función sea la de la conexión directa con el acceso a datos, ya sean remotos o locales, de esta forma queda definido claramente que métodos deben implementar, que parámetros recibir y que objetos devolver.

- **Data Entity**
  Contiene las clases que representan las entidades de los objetos que se devuelven en las clases que conectan con con el acceso a datos, ya sean remotos o locales.

- **Data Error**
  Contiene los posibles errores que se pueden recibir de las llamadas la base de datos o a la API REST, hay dos ficheros, uno enumera los errores que se pueden producir al realizar operaciones sobre la base de datos y otro que enumera los errores que se pueden producir al realizar operaciones relacionadas con la API REST.

- **Data Repository**
  Contiene la implementación de las interfaces definidas en el paquete "Domain Repository", son las clases encargadas de comunicarse con las capas inferiores donde están las clases que hacen las llamadas para obtener o enviar datos a la API REST o a la base de datos.

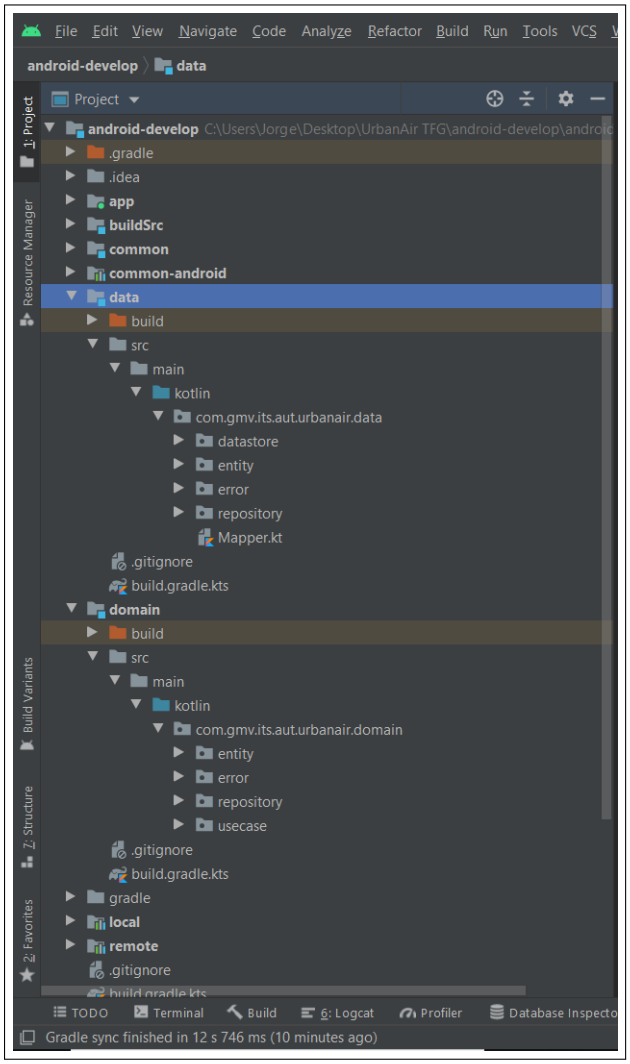

Figura 9.6: Vista de directorios del módulo Domain y Data.

9.2.9. Módulos Local y Remote

Estos directorios contienen el código que realiza toda la conexión con las fuentes de datos, el paquete "Local" se encargará de la base de datos y el paquete "Remote" se encargará de la API REST.

En el paquete "Local" el fichero "Localdatabase.kt" es el fichero de configuración de la base de datos que necesita Room [13] para funcionar correctamente. En ambos paquetes hay un fichero "Mapper.kt" encargado de cambiar y ajustar los tipos de las respuestas de las solicitudes a la base de datos y la API REST para que coincidan con los definidos en las capas superior de "Data".

- **Local Dao**
  Contiene las clases que definen las funciones necesarias para realizar las operaciones CRUD sobre la base de datos. Todas llevan la anotación "@Dao" de Room [13].

- **Local** Contiene las clases que representan las tablas de la base de datos y sus campos y tipos. Todas llevan la anotación "@Entity" de Room [13], también definen las claves primarias y foráneas.

- **Local Storage** Implementa las funciones de las interfaces definidas en el paquete "Data DataStore" pero solo aquellas que necesiten de acceso a la base de datos, utilizan las clases del paquete "Local Dao" para insertar, obtener y modificar los datos de la base de datos.

- **Remote Dto** Modela las peticiones y respuestas de la API REST en forma de objetos con sus

campos, tipos y valores. Deben coincidir con los requisitos de la API REST de cada petición para que no haya errores.

- **Remote Service** Define las interfaces con las funciones necesarias para realizar la conexión con la API REST, que parámetros necesita cada una y que va a devolver, también implementa dichas interfaces dando la funcionalidad completa de acceso a los datos del servidor.

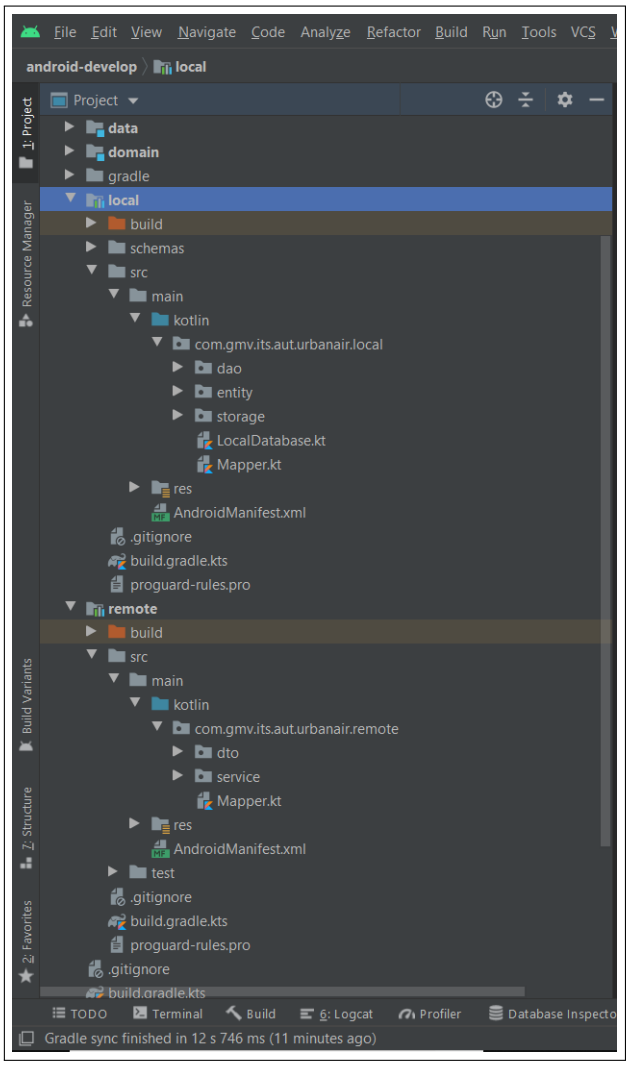

Figura 9.7: Vista de directorios del módulo Local y Remote.

Después de detallar en este capítulo el manual del programador, el siguiente capítulo abordará el manual del usuario.

Capítulo 10

Manual de usuario

En este capítulo se explica cómo hacer uso de las funcionalidades de la aplicación *UrbanAir*. Se incluirán capturas de pantalla de cada una de las vistas en donde se explicará la funcionalidad de cada una y los pasos a realizar.

10.1. Vista de Inicio de Sesión

Esta es primera vista de la aplicación, la que se encontrarán todos los nuevos usuarios, podemos observar que hay dos campos de textos, uno para introducir el nombre de usuario y otro para introducir la contraseña. Como se ha mencionado anteriormente, los usuarios deben haber sido creados previamente por la parte administrativa.

Si el usuario introduce correctamente su nombre de usuario y contraseña o si ya tenía una sesión iniciada, accederá directamente a la vista principal de la aplicación 10.3

Figura 10.1: Vista de Inicio de Sesión.

Si el usuario presiona el botón "Olvidé mi contraseña" se mostrará la siguiente vista:

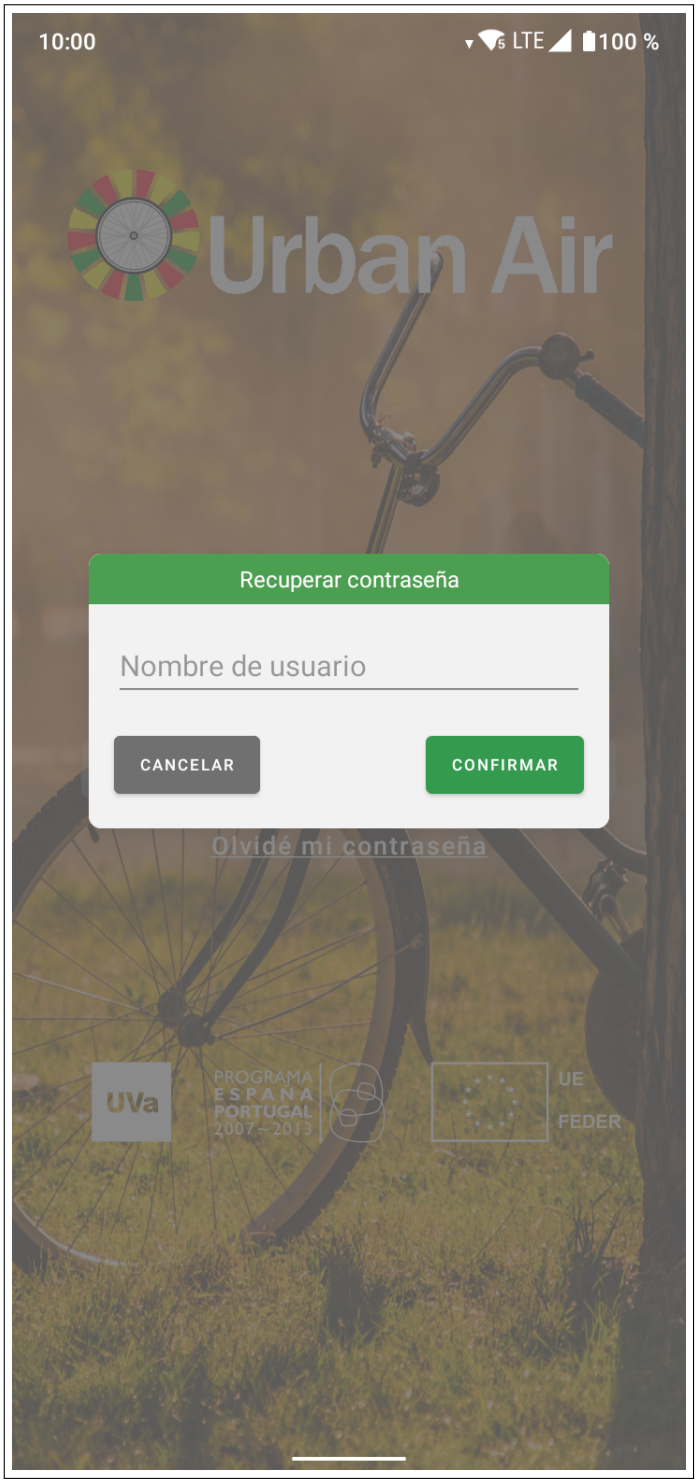

Figura 10.2: Vista de recuperación de contraseña.

Si el usuario introduce su nombre de usuario y presiona confirmar, se le enviará un correo electrónico con las instrucciones necesarias para recuperar su contraseña, si presiona el botón cancelar volverá a la vista de Inicio de Sesión 10.1

10.2. Vista Principal

Esta es la vista principal de la aplicación donde se encuentran las funciones principales como abrir y cerrar la bicicleta.

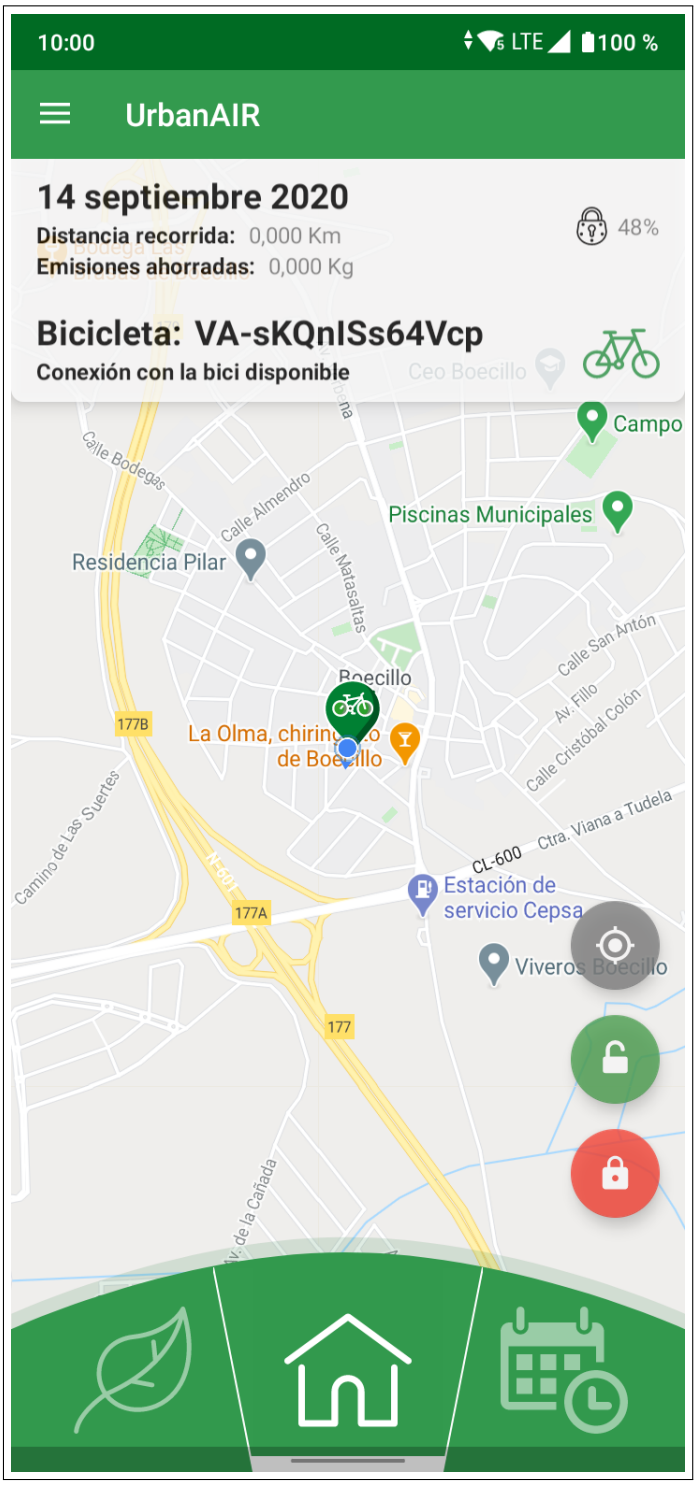

Figura 10.3: Vista principal Home.

Podemos observar que en la parte superior de la pantalla hay un panel con la información diaria del usuario, se muestra la distancia total recorrida en el día actual y la emisiones evitadas al viajar en bicicleta. También se puede observar el nivel de la batería del candado, el ID de la bicicleta y el estado de la conexión con el candado bluetooth.

En el mapa se muestra un punto azul que es la ubicación actual del usuario y un icono verde que

representa la bicicleta y su localización, en la parte derecha están los botones de centra el mapa, abrir la bicicleta y cerrar la bicicleta, en orden de arriba hacia abajo.

Si el usuario presiona el botón verde se abrirá el candado, se iniciará el viaje y se mostrará la siguiente vista:

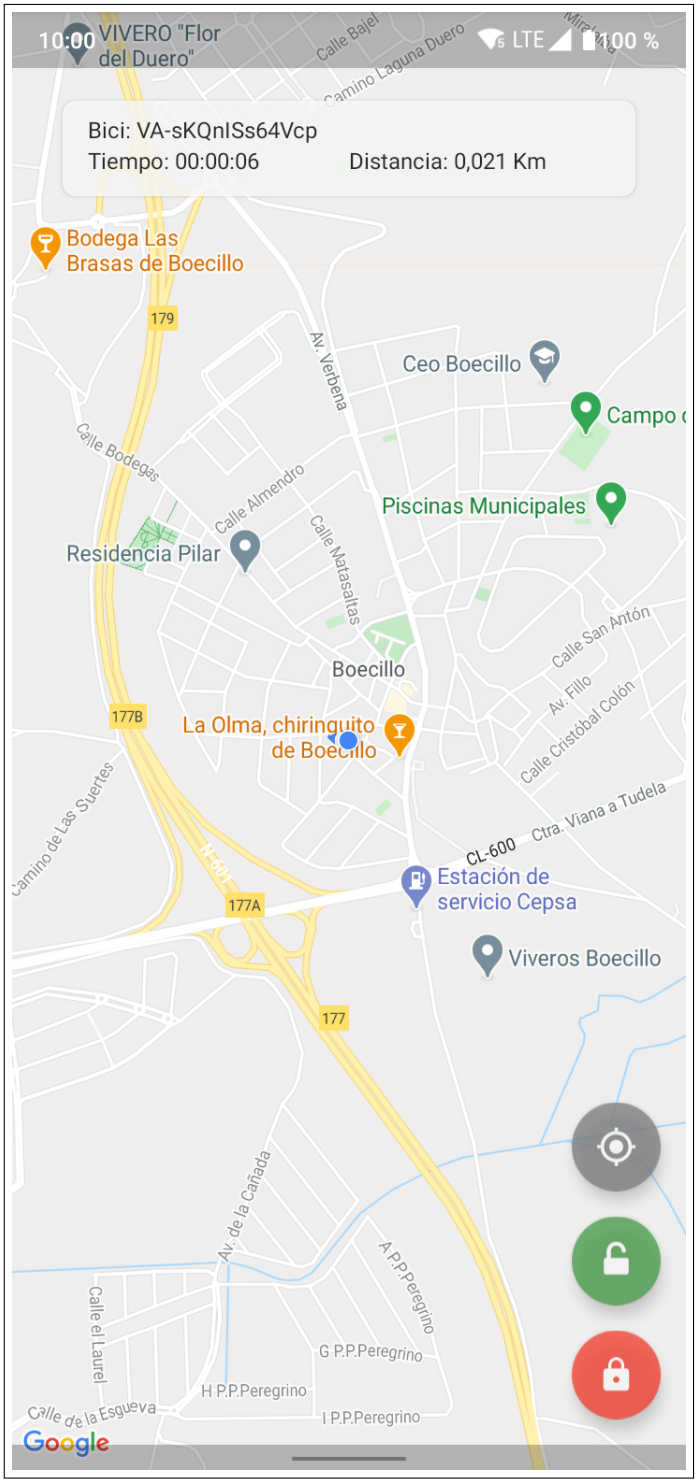

Figura 10.4: Vista durante un viaje.

En esta vista la funcionalidad esta limitada y solo se permite centrar el mapa, abrir la bicicleta y cerrar la bicicleta. En la parte superior se muestra la bicicleta con la que se esta viajando, el tiempo transcurrido y la distancia recorrida en el viaje actual, también se dibujará en color verde la ruta recorrida por el usuario.

Si el usuario tiene asignado un sensor medioambiental y se ha conectado con el se mostrará esta otra

vista similar:

Figura 10.5: Vista durante un viaje con un sensor asignado y conectado.

En esta vista se puede observar lo mismo que en la anterior 10.4 pero con el valor de los datos que está midiendo el sensor durante el viaje en tiempo real.

Si el usuario presiona el botón rojo, se finalizará el viaje, se cerrará el candado y se volverá a la pantalla de inicio 10.3

10.3. Vista Medioambiental

Esta es la vista de la aplicación donde se encuentran los datos medioambientales sobre las emisiones y su concentración.

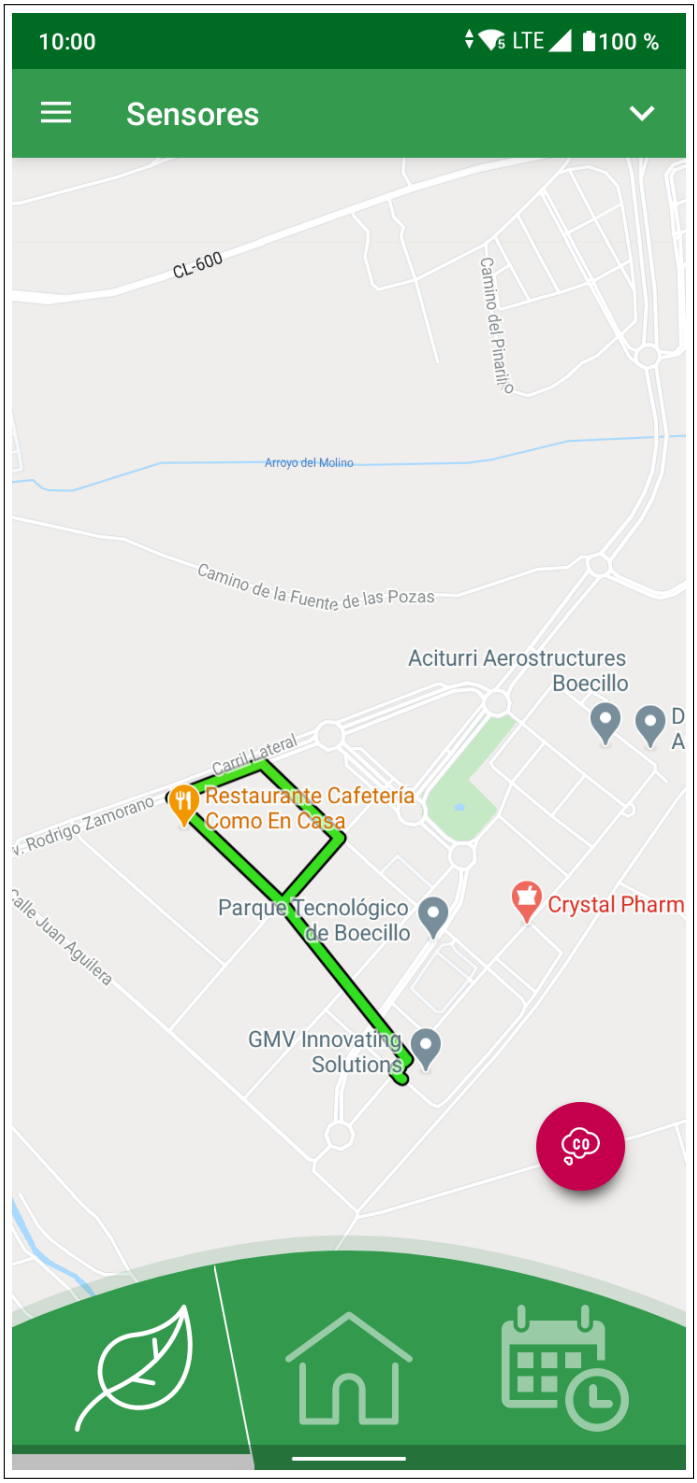

Figura 10.6: Vista de los datos medioambientales.

Se puede observar como se muestra un mapa con los datos medioambientales recogidos en esa zona, en la parte superior derecha hay un icono que despliega un panel con mas información 10.9 y en la parte inferior derecha un botón para cambiar el tipo de partículas seleccionado 10.7 y 10.8.

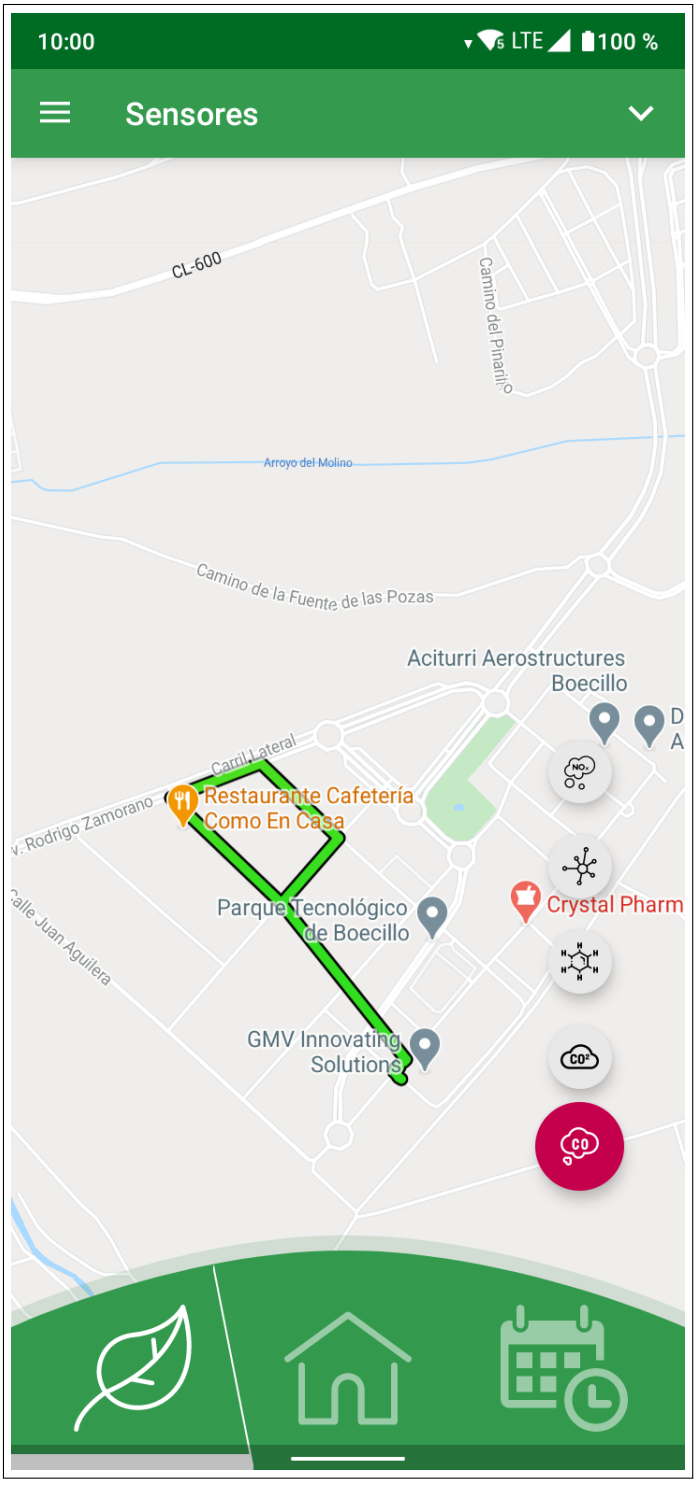

Figura 10.7: Vista de los datos medioambientales con la selección de partículas.

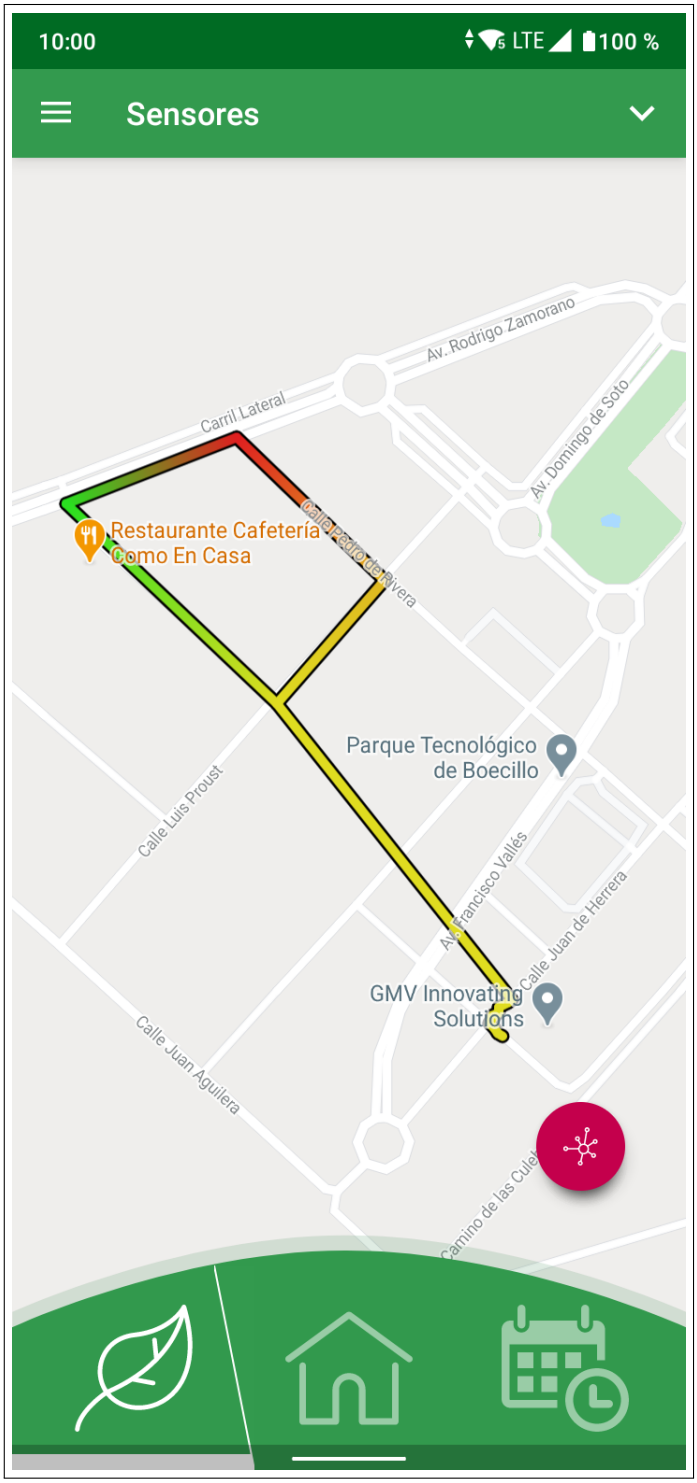

Figura 10.8: Vista de los datos medioambientales después de cambiar el tipo de partículas.

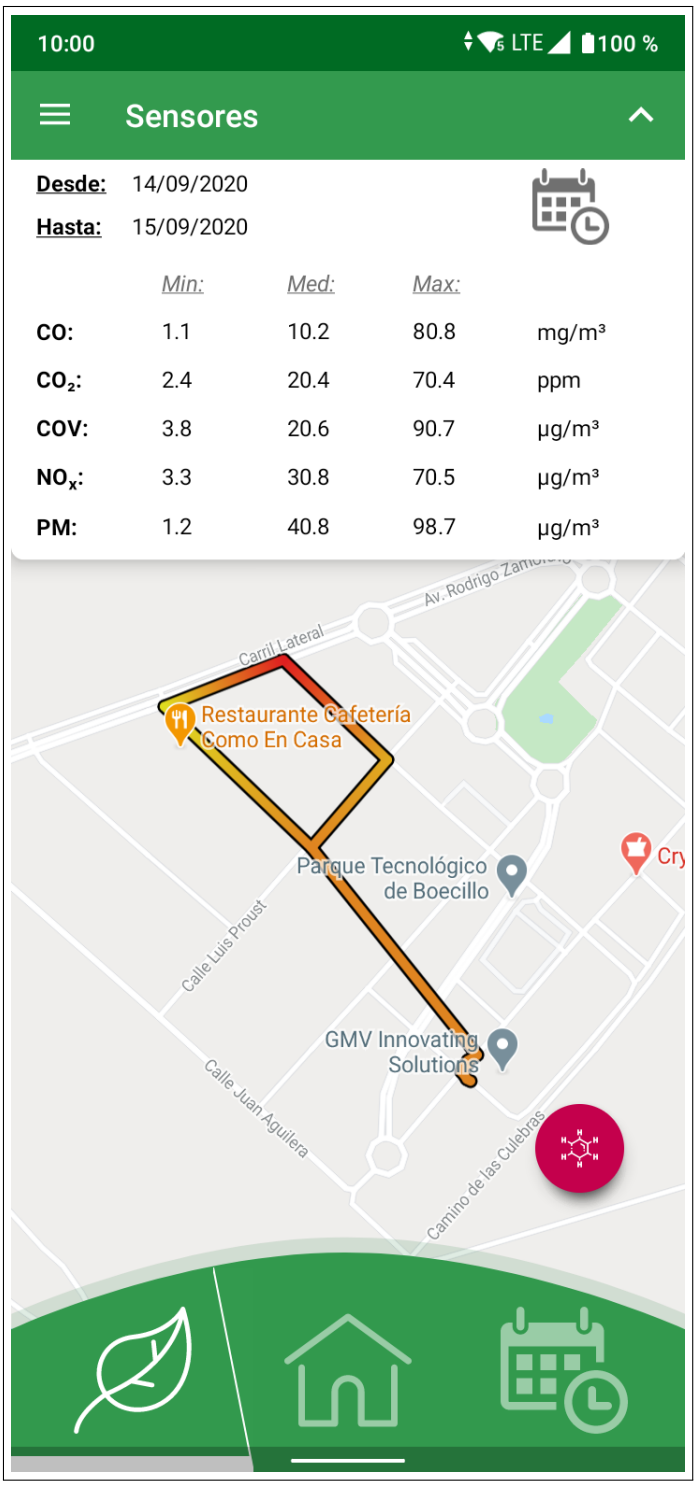

Figura 10.9: Vista de los datos medioambientales con el panel informativo desplegado.

En la figura de la vista anterior 10.9 el usuario podrá observar los valores máximos, mínimos y medios de los distintos tipos de emisiones entre las dos fechas seleccionadas, si presiona una de las fechas o el icono del calendario, podrá cambiarlas y así seleccionar el periodo que deseé como se muestra en la siguiente vista.

Figura 10.10: Vista de los datos medioambientales cambiando las fechas seleccionadas.

10.4. Vista Calendario

Esta es la vista de la aplicación donde el usuario puede consultar todos los viajes que ha realizado en una fecha concreta

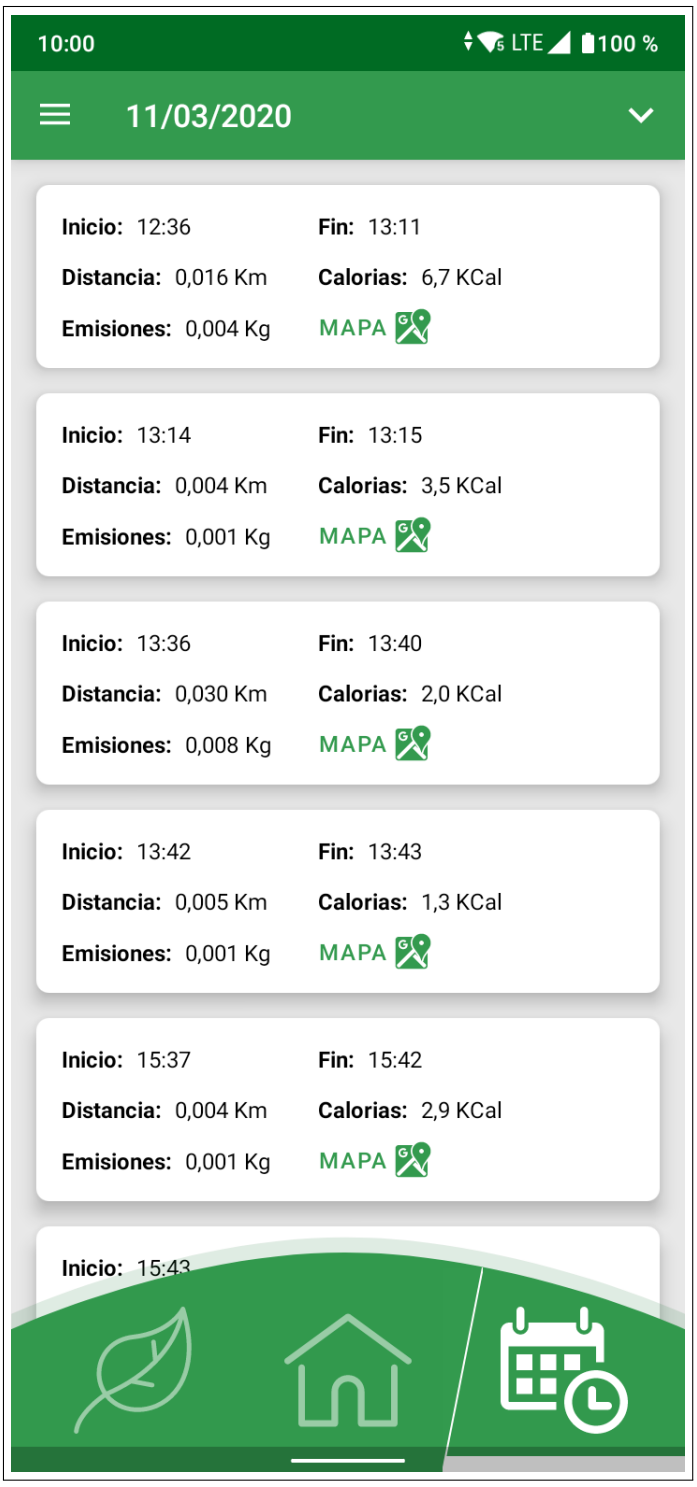

Figura 10.11: Vista del historial de viajes.

Para cada viaje puede consultar la hora de inicio, la hora de finalización, la distancia recorrida, las calorías quemadas, las emisiones evitadas y si presiona el botón "Mapa" podrá observar un mapa con la ruta recorrida como en la siguiente figura 10.12. Si el usuario presiona el botón "Cerrar Mapa" regresará a la vista anterior 10.11.

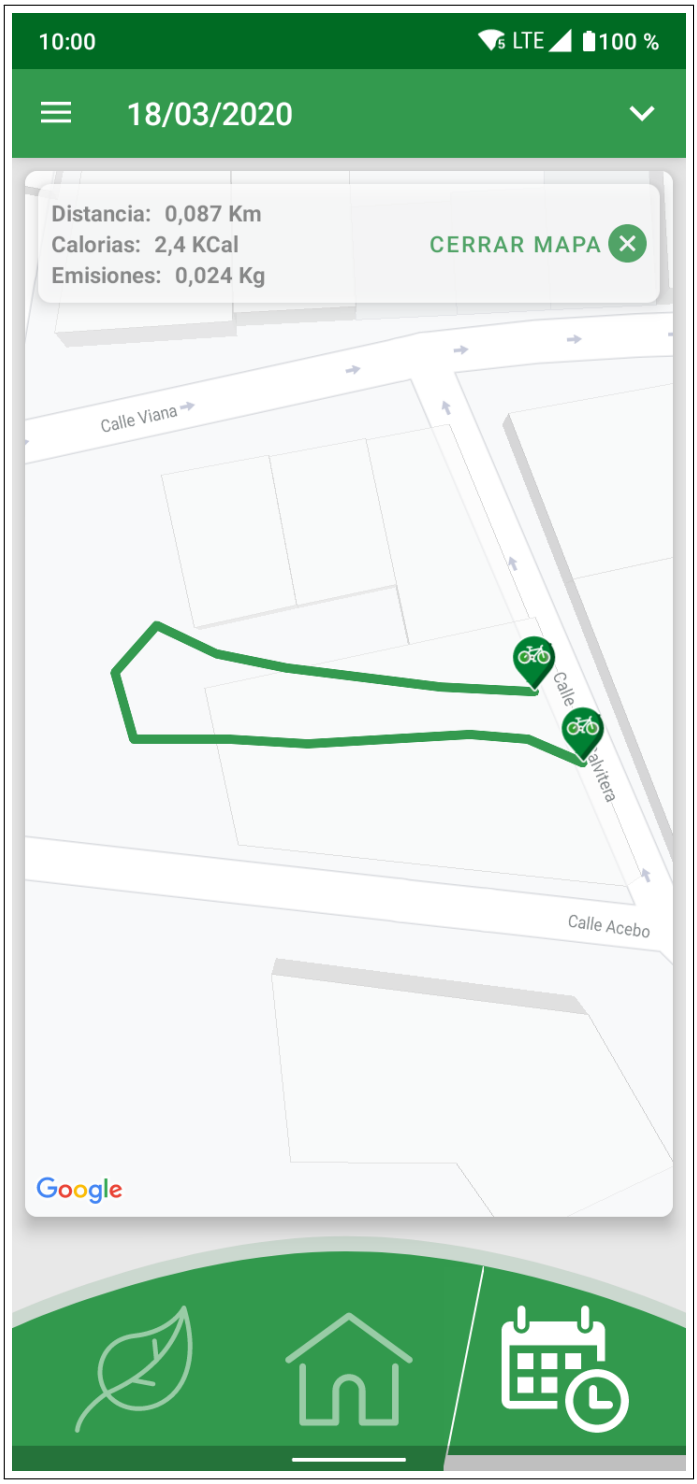

Figura 10.12: Vista del mapa de un viaje.

En la parte superior derecha se puede observar un icono con una flecha hacia abajo, si se presiona se despliega un calendario donde el usuario puede cambiar la fecha en la que quiere consultar sus viajes, también se marcan los días en los que hay viajes registrados.

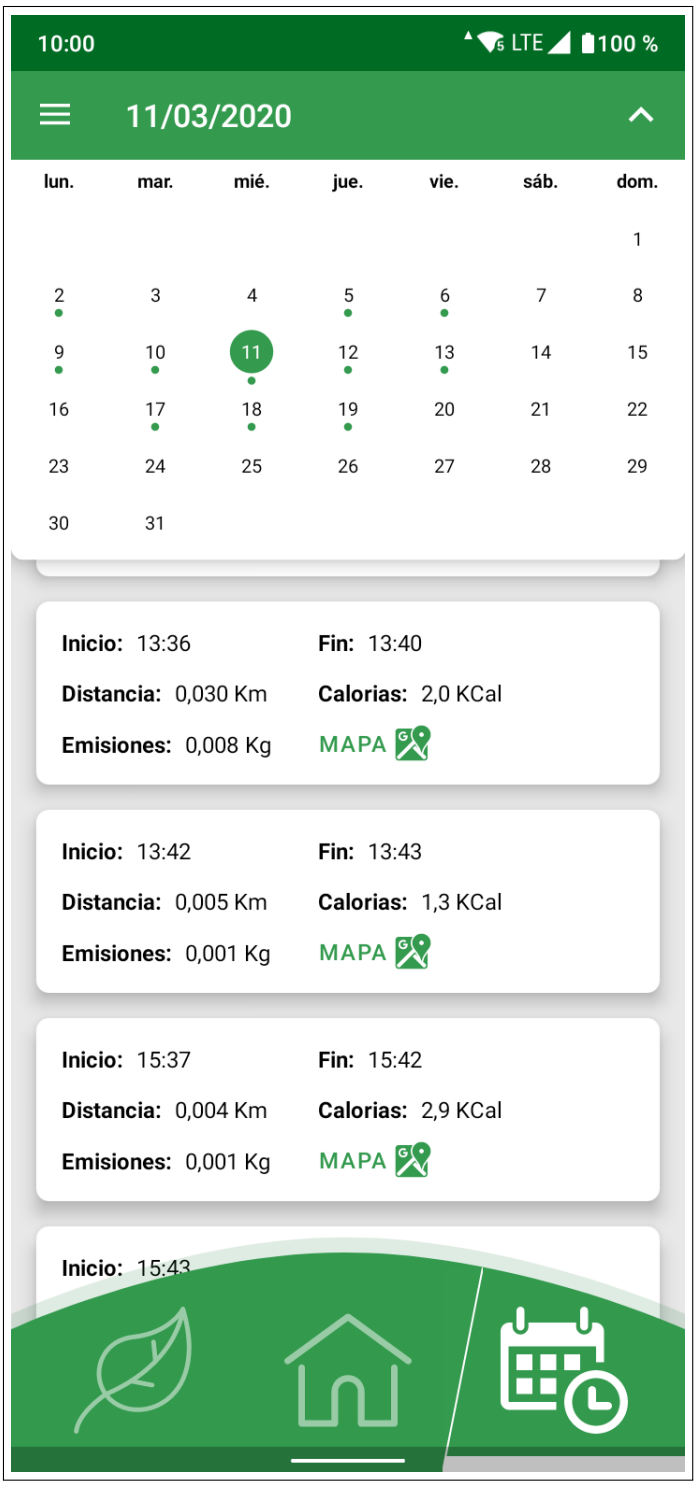

Figura 10.13: Vista del calendario de la vista del historial de viajes.

10.5. Vista Perfil de Usuario

Esta es la vista de la aplicación donde el usuario puede consultar su información personal 10.14, las atribuciones 10.15, consultar la política de privacidad 10.16, cambiar su contraseña 10.17, editar su perfil 10.18 y cerrar su sesión volviendo a la vista inicial de "Inicio de Sesión" 10.1.

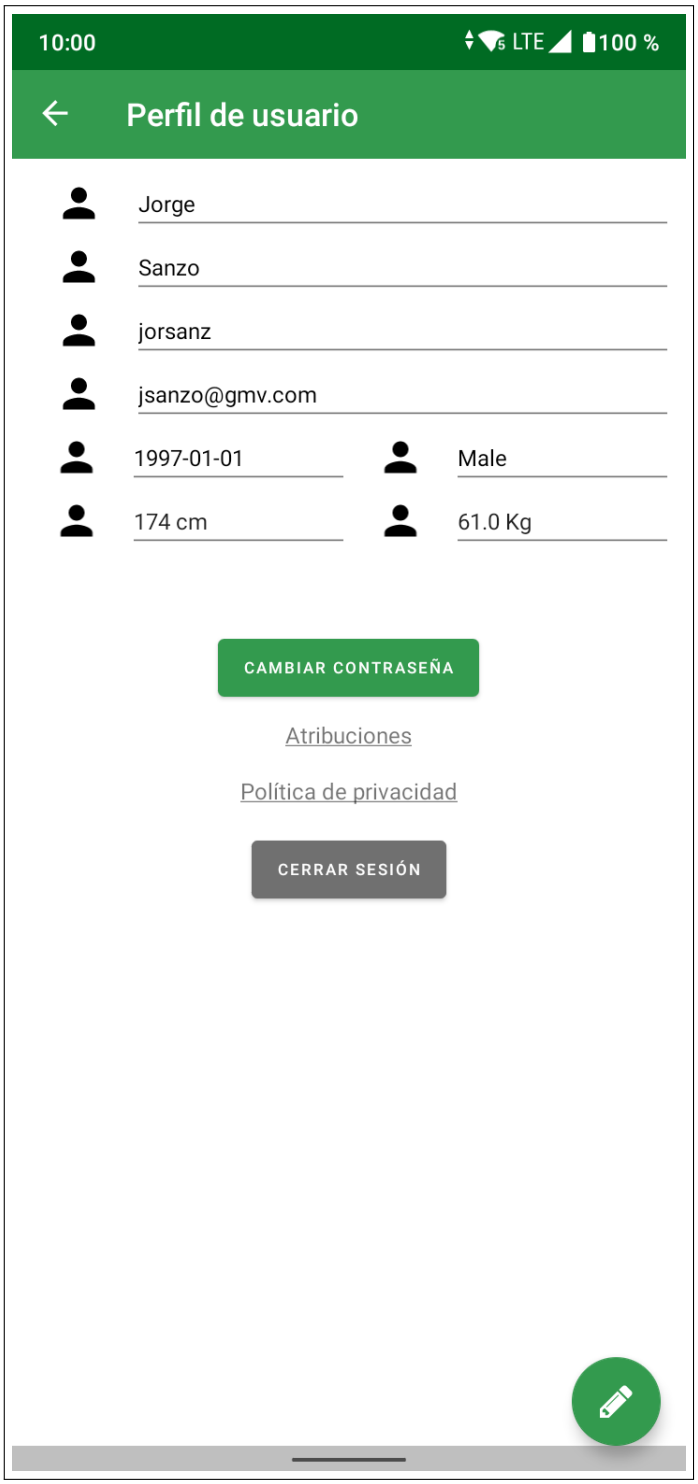

Figura 10.14: Vista del perfil de usuario.

Figura 10.15: Vista de las atribuciones.

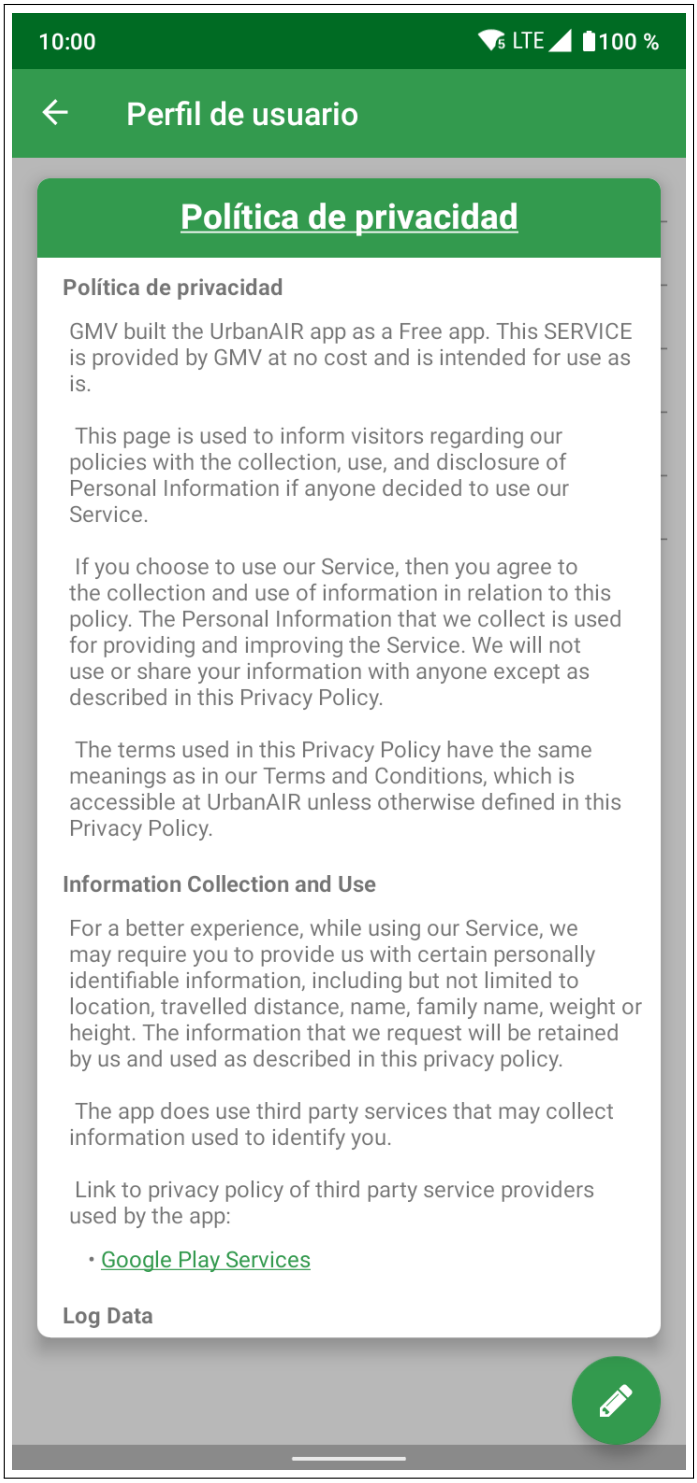

Figura 10.16: Vista de la política de privacidad.

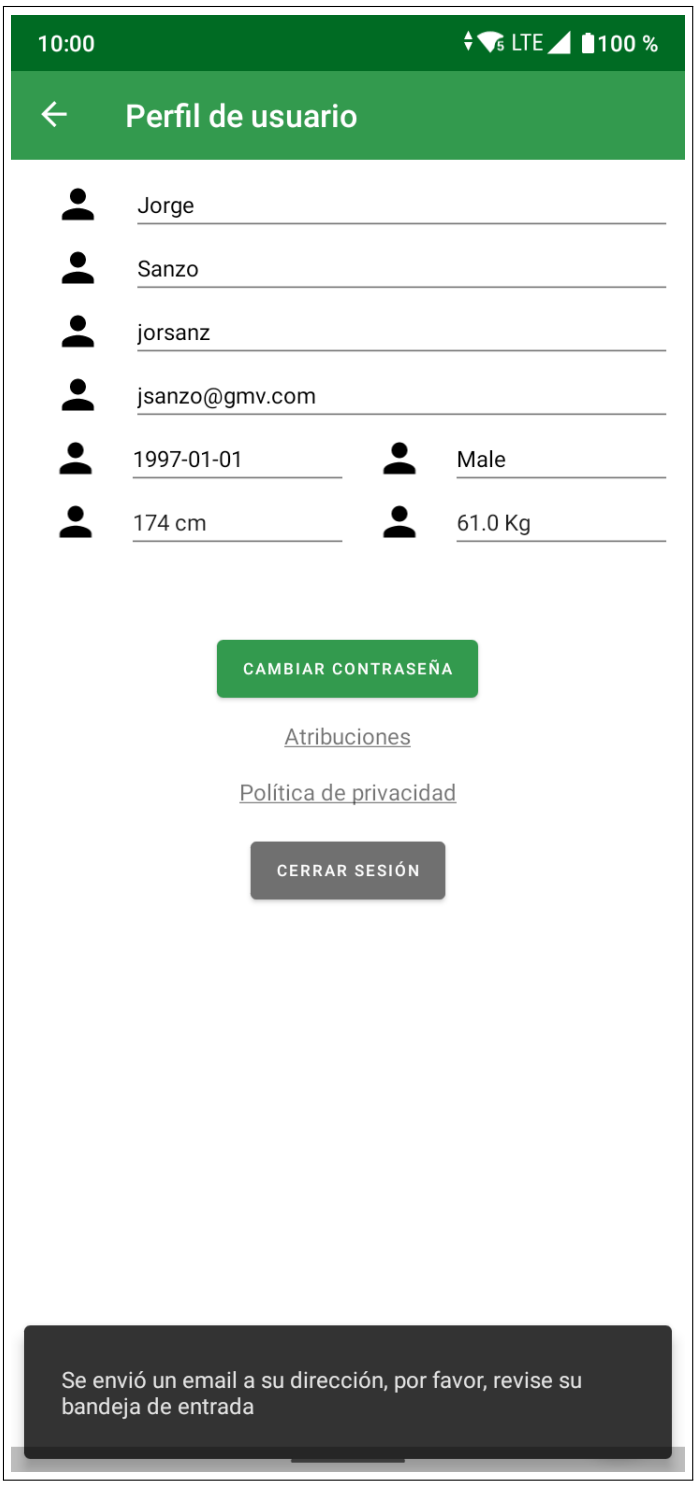

Figura 10.17: Vista del cambio de contraseña.

Si el usuario presiona el botón de la parte inferior derecha entrará en el modo de edición del perfil donde se le permitirá modificar su pero y su altura 10.19 y 10.20

Figura 10.18: Vista del perfil de usuario en el modo de edición.

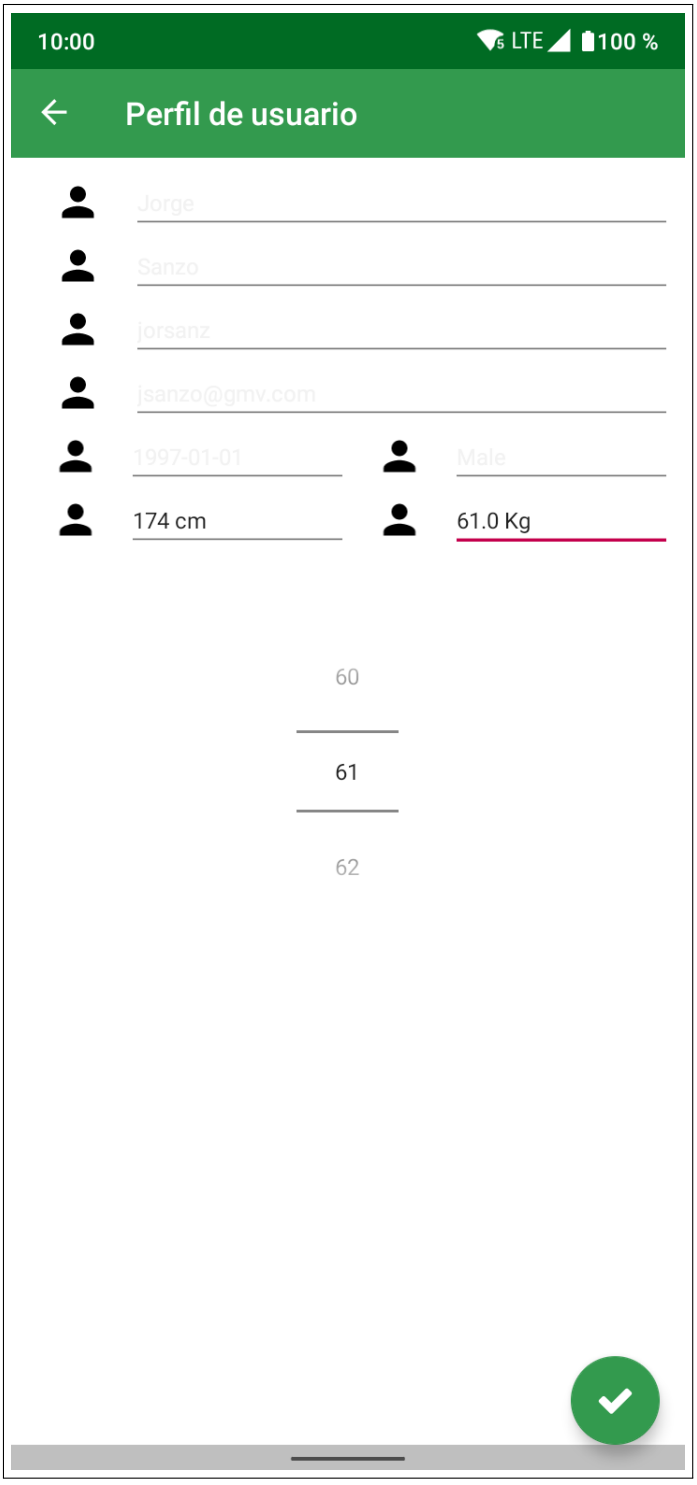

Figura 10.19: Vista del perfil de usuario editando el peso.

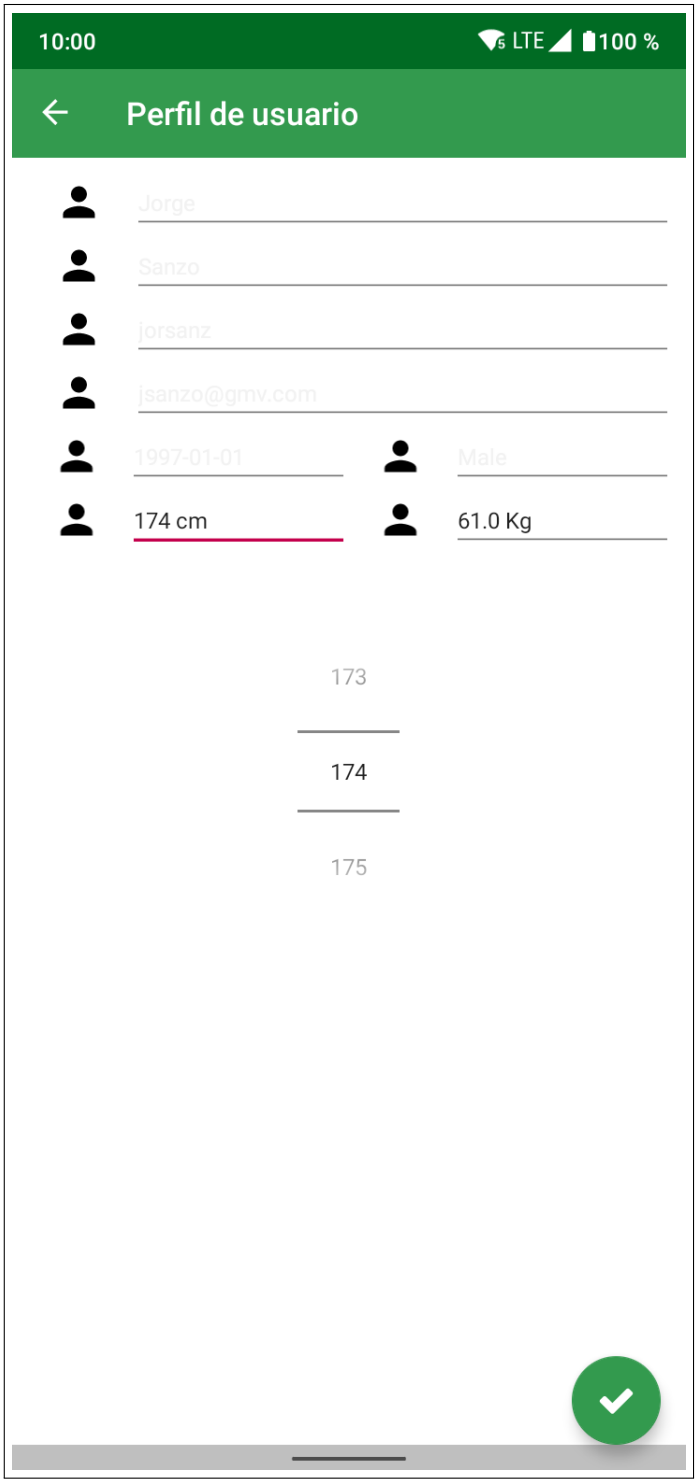

Figura 10.20: Vista del perfil de usuario editando la altura.

10.6. Vista de Sensores

Esta es la vista donde el usuario puede observar todos los sensores medioambientales cercanos y debe seleccionar el suyo antes de iniciar un viaje si quiere usarlo en ese momento. Una vez seleccionado se le mostrará un mensaje de confirmación y podrá iniciar su viaje, esto no es obligatorio para viajar, puede iniciar un viaje sin tener un sensor asignado, seleccionado o conectado.

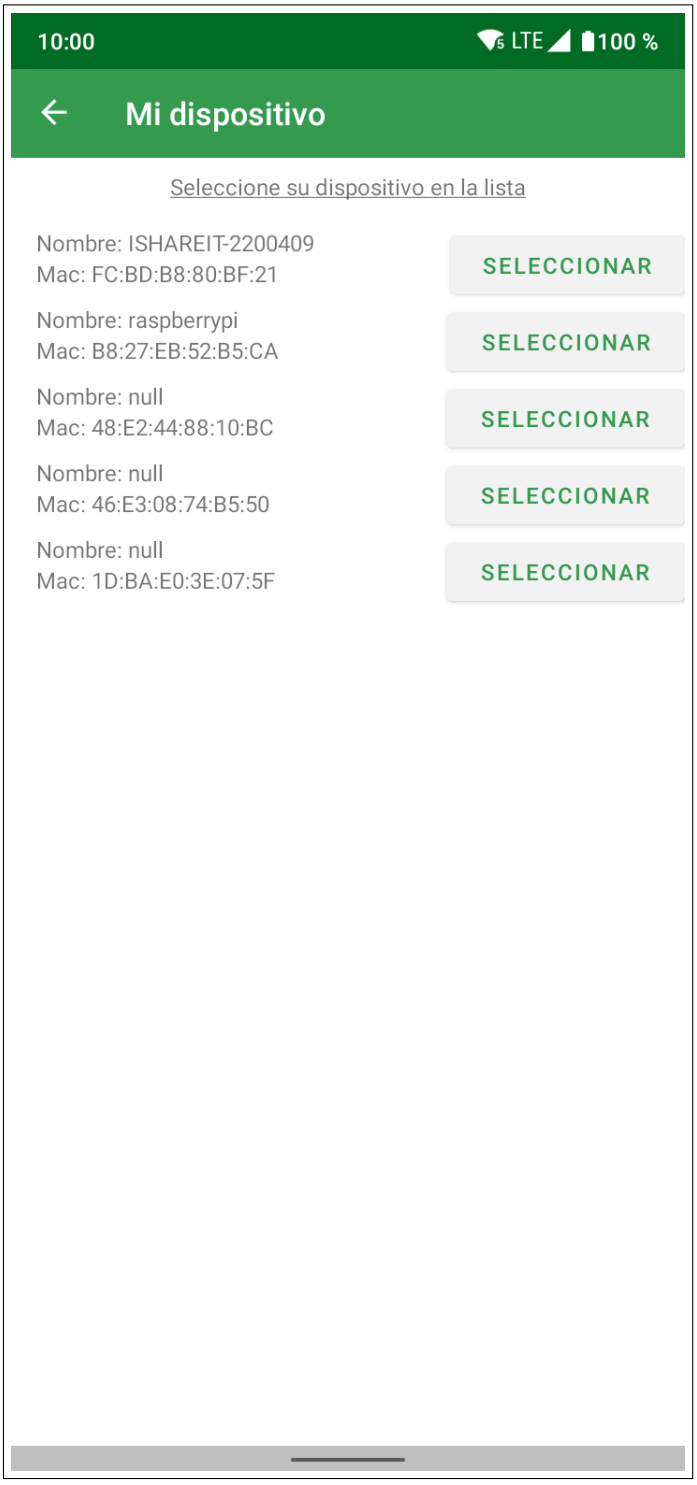

Figura 10.21: Vista de los sensores cercanos disponibles.

10.7. Vista Menú lateral

Para poder navegar a las vistas del perfil de usuario 10.14 y de los sensores disponibles 10.21 el usuario debe desplegar el menú lateral disponible en todas las vistas, excepto en la de inicio de sesión, para hacerlo debe pulsar el icono con tres líneas de la parte superior izquierda, se desplegará un panel como se muestra a continuación. Desde este menú también puede cerrar su sesión.

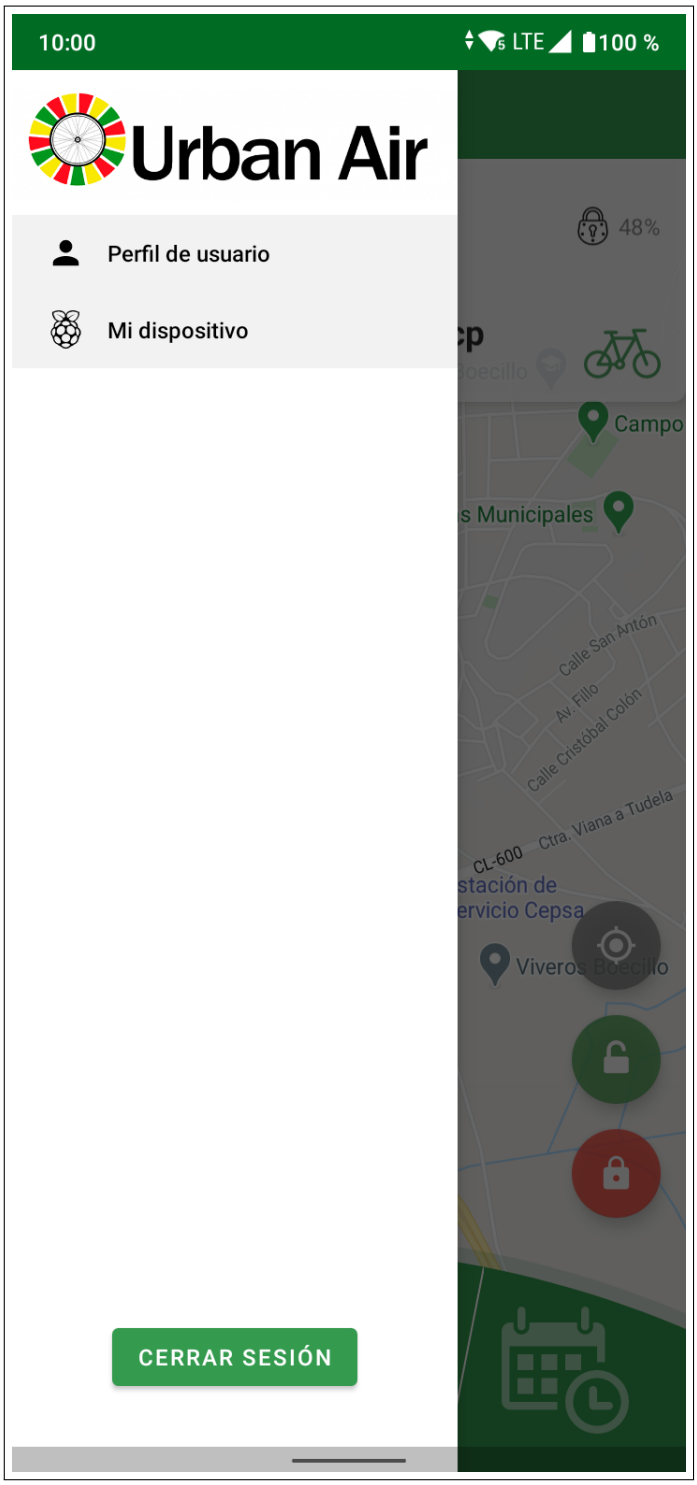

Figura 10.22: Vista del menú desplegado desde la vista Home.

Después de abordar en este capítulo el manual del usuario, el siguiente capítulo se ocupará de las conclusiones y el trabajo futuro.

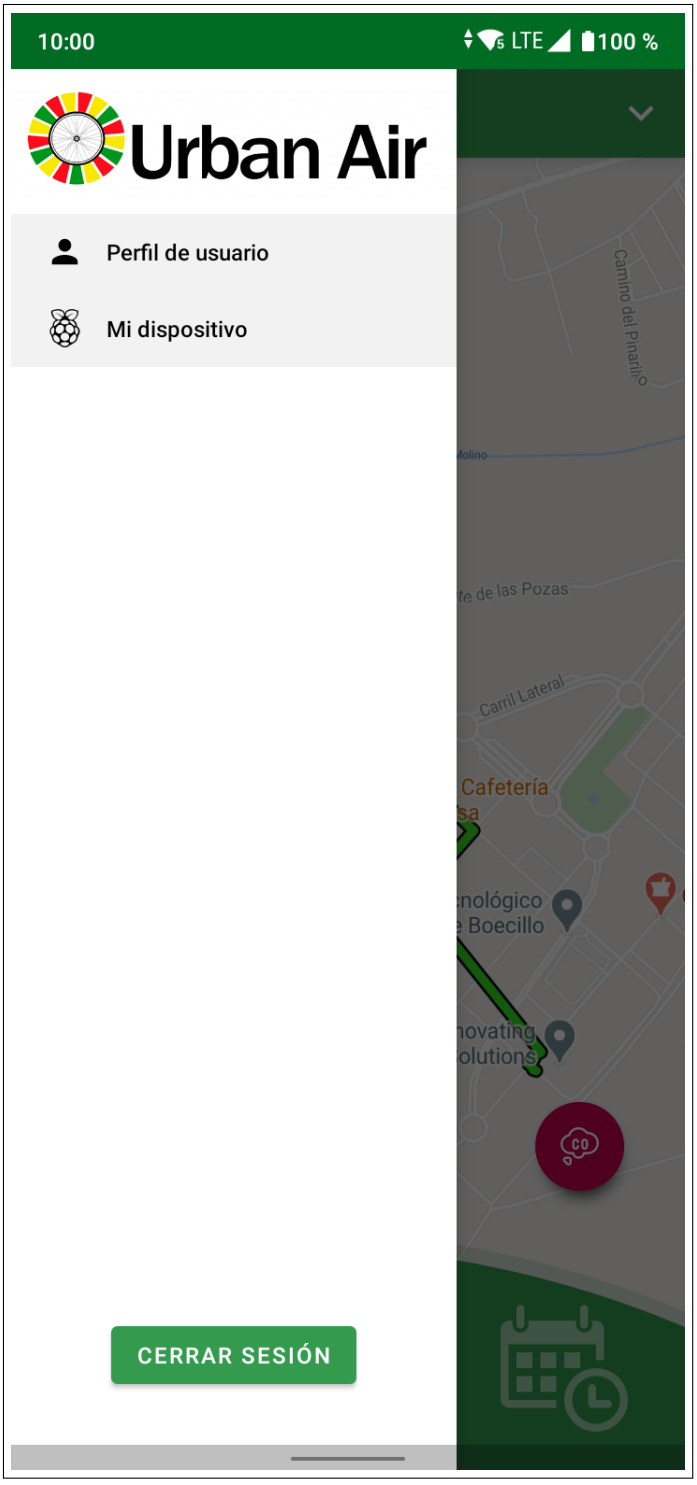

Figura 10.23: Vista del menú desplegado desde la vista Leaf.

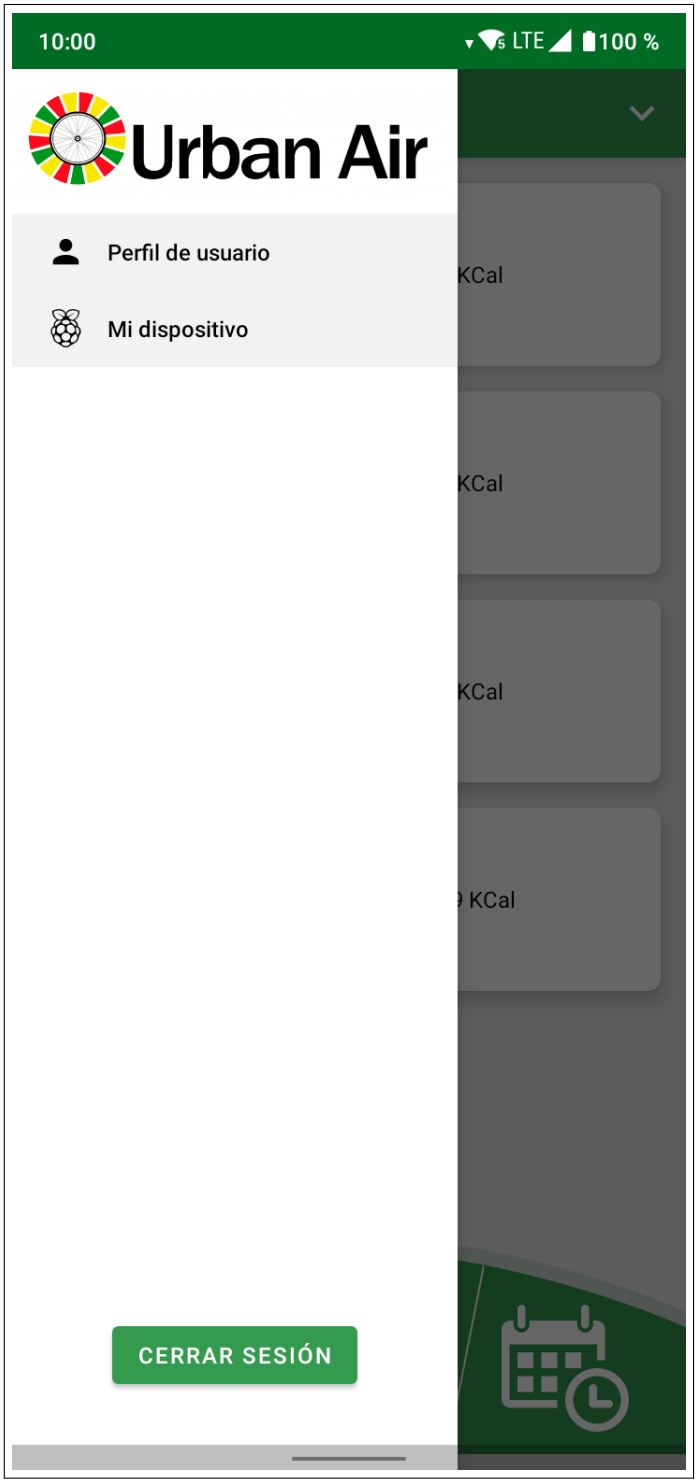

Figura 10.24: Vista del menú desplegado desde la vista Calendar.

Capítulo 11

Conclusiones y trabajo futuro

11.1. Conclusiones

Este TFG ha sido realizado por el alumno Jorge Sanzo Hernando en la empresa GMV [5] donde realizó sus prácticas curriculares y donde actualmente se encuentra trabajando desde hace más de un año. El proyecto ha sido aprobado y supervisado por su responsable en la empresa Pablo Rivas Salmón. Se ha realizado también en colaboración con la empresa CARTIF [2] que ha proveído los sensores medioambientales, la marca iLockIt [1] que ha proveído los candados bluetooth, GMV [5] realizó el backend correspondiente a la API REST. La Universidad de Valladolid ha sido la impulsora de este proyecto además de proveer las bicicletas y los usuarios finales de pruebas, se prevé que entre en funcionamiento con el comienzo del curso 2020/2021.

En la realización de este TFG se ha creado una aplicación que permite a los usuarios utilizar un sistema de préstamos de bicicletas, gestionado por la Universidad de Valladolid. Este proyecto también tiene como finalidad la concienciación de los usuarios sobre el medioambiente y la reducción del uso de los vehículos de motor para ir a la universidad a diario para tratar de reducir la contaminación y el impacto medioambiental.

En los siguientes puntos se resume brevemente el trabajo desarrollado por el alumno:

- Se ha realizado el análisis y diseño, el plan de pruebas, y los manuales del trabajo implementado.
- Se ha programado la aplicación móvil siguiendo los patrones de diseño y arquitecturas descritas en esta memoria.
- Se ha adaptado la aplicación para permitir la conexión con diversos dispositivos externos como el candado bluetooth de las bicicletas o los sensores medioambientales, además de la conexión con la API REST.
- Se ha asignado un código de colores a los valores de la contaminación para que los usuarios puedan ser conscientes visualmente del problema que representa hoy en día.
- Se ha implementado un sistema que permite la apertura y cerradura de los candados de las bicicletas para poder utilizarlas, este sistema permite registrar los viajes realizados por el usuario. Lo más importante de estos viajes es el cálculo de las emisiones evitadas durante el mismo, de esta forma el usuario puede ser consciente del impacto del uso de las bicicletas frente a los vehículos de motor.

11.2. Trabajo futuro

Existen varias posibilidades de ampliación de funcionalidades para un sistema de gestión de préstamo de bicicletas, en consonancia con el trabajo desarrollado en este TFG. A continuación, se destacan aquellas

que se consideran más interesantes:

- Mejora en los candados utilizados, algunos candados llevan incorporado un chip GPS, esto mejoraría la localización de la bicicleta en todo momento, por ahora se guarda la ubicación donde se abrió o cerró por ultima vez, a veces esto puede no ser preciso del todo.
- Mejora en los candados utilizados, algunos candados llevan incorporado placas solares pequeñas que recargan la batería evitando que esta se pueda llegar a agotar, actualmente se cargan por USB.
- Implementación de un sistema de notificaciones cuando el candado se está quedando sin batería para que el usuario lo cargue, ya que si no no va a poder abrir su bicicleta.
- Mejora del sistema de sensores, los usuarios tendrán el suyo propio asignado y la información de conexión se recibirá desde el servidor, esto hará que no sea necesario seleccionar el sensor antes de cada viaje si se quiere utilizar, además no sería opcional utilizarlo si se tiene asignado lo que generaría muchos mas datos mejorando el funcionamiento de la aplicación.
- Implementación de un sistema que permita obtener gráficas de la contaminación a lo largo del tiempo, ya sea un día o un periodo de días, de esta forma se podrían observar los días o las horas con mayores niveles de contaminación.
- Implementación del servicio que devuelve los valores mínimos, medios y máximos de los valores medioambientales ya que aun no esta terminado en el lado del servidor, por ahora se muestran datos que han sido "mockeados" para el desarrollo del proyecto.
- Cambiar el sistema de cambio de contraseña para que el usuario pueda realizarlo desde la propia aplicación sin tener que recibir un correo electrónico con las instrucciones necesarias.

11.3. Archivos entregados

Junto a esta memoria del Trabajo de Fin de Grado se adjuntarán tambien los siguientes archivos, estos pueden encontrarse en el siguiente enlace de Google Drive `https://drive.google.com/drive/folders/1U7Mfiiv8uhAqW4w8db-XRrkO8oW__xt3?usp=sharing`:

- app-urbanair-release-v2.0.8.apk, este archivo corresponde a la versión compilada y ejecutable de la aplicación, es una versión "Release" por lo que es la misma que utilizarán los usuarios finales
- app-urbanair-debug-v2.0.8.apk, este archivo corresponde a la versión compilada y ejecutable de la aplicación, es una versión "Debug", no es la que utilizarán los usuarios finales, esta versión incluye algunas librerías extra utilizadas durante su desarrollo, se pueden ver también en la barra de notificaciones las llamadas de red que se realizan con toda su información.
- UrbanAIR_Sistema_de_Gestion_de_prestamo_de_bicicletas_para_la_Universidad_de_Valladolid.astah, este archivo contiene todos los diagramas realizados para el proyecto, aunque también han sido incluidos en esta memoria en forma de imágenes.
- UrbanAir_Android.zip, este archivo comprimido contiene todo el proyecto Android y el código fuente del mismo, se puede abrir desde Android Studio seleccionando el archivo "android" que contiene.

Referencias

[1] Candados ilockit. https://ilockit.bike/en/. Ultimo acceso el 12/09/2020.

[2] Cartif. https://www.cartif.es/. Ultimo acceso el 12/09/2020.

[3] Colores hsl. http://w3.unpocodetodo.info/css3/hsl.php. Ultimo acceso el 12/09/2020.

[4] Gitlab gmv. https://spass-git-ext.gmv.com/users/sign_in. Ultimo acceso el 12/09/2020.

[5] Gmv s.a. https://www.gmv.com/es/. Ultimo acceso el 12/09/2020.

[6] Inkscape. https://inkscape.org/es/. Ultimo acceso el 12/09/2020.

[7] Normas APA. https://normasapa.net/que-es-el-estado-del-arte/.

[8] EYM Apps. https://cleanspotapp.com/.

[9] Atlassian. Sourcetree. https://www.sourcetreeapp.com/. Ultimo acceso el 12/09/2020.

[10] Nihon Kasetsu CO. https://nihonkasetsu.com/es/el-indice-de-calidad-del-aire-aqi/.

[11] Ayuntamiento de Valladolid. https://www.valladolid.com/vallabici.

[12] Android Developer. Instant type. https://developer.android.com/reference/kotlin/java/time/Instant. Ultimo acceso el 11/09/2020.

[13] Android Developer. Room library. https://developer.android.com/topic/libraries/architecture/room. Ultimo acceso el 11/09/2020.

[14] Android Developers. Android design guidelines. https://developer.android.com/design. Ultimo acceso el 11/09/2020.

[15] Android Developers. Google maps. https://developers.google.com/maps/documentation/android-sdk/overview. Ultimo acceso el 11/09/2020.

[16] Google Developers. https://firebase.google.com/?hl=es.

[17] Google Developers. https://console.firebase.google.com/.

[18] Google Developers. https://firebase.google.com/docs/analytics.

[19] Google Developers. Android studio. https://developer.android.com/studio/?gclid=CjwKCAjw4_H6BRALEiwAvgfzq1nLwUrRoQL94iSVqswVmqE4Olabh3q--yLdzWscortluvDuluOl5BoCvKIQAvD_BwE&gclsrc=aw.ds. Ultimo acceso el 12/09/2020.

[20] Google Developers. Firebase crashlytics. https://firebase.google.com/docs/crashlytics. Ultimo acceso el 11/09/2020.

[21] Comisión Europea. https://digitalearthlab.jrc.ec.europa.eu/app/loss-night.

[22] Kotlin Foundation. Unit type. https://kotlinlang.org/api/latest/jvm/stdlib/kotlin/-unit/. Ultimo acceso el 11/09/2020.

[23] GitLab. Gitlab — a full devops tool. https://about.gitlab.com/. Ultimo acceso el 12/09/2020.

[24] Arnaud Giuliani. Koin library for dependency injection. https://insert-koin.io/. Ultimo acceso el 11/09/2020.

[25] LLC Icons8. Página web de icons8 donde descargar iconos para aplicaciones de forma gratuita. https://iconos8.es/. Ultimo acceso el 12/09/2020.

[26] Antonio Leiva. https://devexperto.com/clean-architecture-android/.

[27] Paul Odhiambo. Retrofit2 library. https://dev.to/paulodhiambo/kotlin-and-retrofit-network-calls-2353. Ultimo acceso el 11/09/2020.

[28] Arrow Org. Either type. https://arrow-kt.io/docs/apidocs/arrow-core-data/arrow.core/-either/. Ultimo acceso el 11/09/2020.

[29] Arrow Org. Option type. https://arrow-kt.io/docs/apidocs/arrow-core-data/arrow.core/-option/. Ultimo acceso el 11/09/2020.

[30] Inc Postman. Postman. https://www.postman.com/. Ultimo acceso el 12/09/2020.

[31] European Union's Horizon 2020 programme. https://www.urbangreenup.eu/valladolid-da-buttare/urban-greenup-in-valladolid.kl.

[32] Freepik Company S.L. Página web de flaticon donde descargar iconos para aplicaciones de forma gratuita. https://www.flaticon.es/. Ultimo acceso el 12/09/2020.

[33] Solid GEAR Projects S.L. https://solidgeargroup.com/bluetooth-ble-el-conocido-desconocido/.

[34] Inc Square. Response type. https://square.github.io/retrofit/2.x/retrofit/retrofit2/Response.html. Ultimo acceso el 11/09/2020.

[35] JetBrains s.r.o. https://www.jetbrains.com/es-es/idea/.

Anexos

Anexo I

Apéndice A

A continuación, se describen las operaciones facilitadas por la API REST.

La URL raíz de la API es la siguiente: `https://urbanair.etc-gmv.com/services/`

Estructura de la respuesta

En todas las peticiones se retorna una respuesta con la siguiente estructura:

Field	Type	Description
Código	HTTP Code	Código HTTP de la respuesta.
Data	APIResponse	Datos de la petición.

El atributo **Data** contiene una estructura formada por un objeto en formato JSON que será analizado por la aplicación.

Los códigos de error que se gestionarán son:

Type	Code	Description
Unauthorized	401	Denegación de servicio por falta de credenciales.
Not Found	404	Recurso no encontrado.
Internal server error	500	Error interno del servidor.

En las siguientes secciones, se detallará el contenido del atributo **Data** para todas las peticiones.

Auth Service

Login

Obtenemos los tokens de acceso del usuario.

Method	POST
URL	/login

Body params

Field	Type	Description
username	String	Nombre de usuario del usuario.
password	String	Contraseña del usuario.

Success Response: OK 200

Nos retorna un objeto con los dos tokens de acceso del usuario.

Field	Type	Description
access_token	String	Token de acceso del usuario.
refresh_token	String	Token de refresco del usuario.

Logout

Obtenemos los tokens de acceso del usuario.

Method	GET
URL	/logout

Success Response: OK 200

No devuelve nada más que el código de respuesta.

Refresh token

Obtenemos un nuevo token de acceso si el anterior había expirado y el de refresco sigue activo.

Method	POST
URL	/auth/refreshtoken

Body params

Field	Type	Description
refresh_token	String	Token de refresco del usuario.

Success Response: OK 200

Nos retorna un objeto con los dos nuevos tokens de acceso del usuario.

Field	Type	Description
access_token	String	Token de acceso del usuario.
refresh_token	String	Token de refresco del usuario.

Bike Service

Open Bike

Notificamos al servidor de que el usuario ha abierto su bicicleta.

Method	GET
URL	/vehicle/id/open

Success Response: OK 200

No devuelve nada más que el código de respuesta.

Close Bike

Notificamos al servidor de que el usuario ha cerrado su bicicleta.

Method	GET
URL	/vehicle/id/close

Success Response: OK 200

No devuelve nada más que el código de respuesta.

Get Bike

Obtenemos la bicicleta del usuario.

Method	GET
URL	/vehicle

Success Response: OK 200

Nos retorna un objeto con los datos de la bicicleta.

Field	Type	Description
id	String	Identificador de la bicicleta.
id_device	String	Identificador del candado asociado a la bicicleta.
position	DataBikePosition	Última posición registrada de la bicicleta.
status	String	Estado de la bicicleta.
key_device	String	Contraseña para interactuar con el candado.

Leaf Service

Get environmental data

Obtenemos los datos medioambientales.

Method	POST
URL	/history/environmentaldata

Body params

Field	Type	Description
from	String	Fecha de inicio del intervalo de medición de los datos.
to	String	Fecha de finalización del intervalo de medición de los datos.

Success Response: OK 200

Nos retorna un objeto con los datos medioambientales medidos entre las dos fechas seleccionadas. Este servicio no está implementado aún pero si está diseñado de esta forma, puede que en el futuro sufra cambios a la hora de su implementación.

Field	Type	Description
avg_co	Float	Valor medio de CO.
avg_co_2	Float	Valor medio de CO2.
avg_cov	Float	Valor medio de COV.
avg_nox	Float	Valor medio de NOX.
avg_pm	Float	Valor medio de PM.
min_co	Float	Valor mínimo de CO.
min_co_2	Float	Valor mínimo de CO.
min_cov	Float	Valor mínimo de CO2.
min_nox	Float	Valor mínimo de NOX.
min_pm	Float	Valor mínimo de PM.
max_co	Float	Valor máximo de CO.
max_co_2	Float	Valor máximo de CO2.
max_cov	Float	Valor máximo de COV.
max_nox	Float	Valor máximo de NOX.
max_pm	Float	Valor máximo de PM.
from	String	Fecha de inicio del intervalo de medida de los datos.
to	String	Fecha de finalización del intervalo de medida de los datos.
environmentalTrips	List<DataEnvironmentalTrips>	Lista con las posiciones de cada medida de datos para poder dibujar las rutas medioambientales en el mapa, en la vista "Leaf" 10.6.

Trips Service

Get Trips

Obtenemos los viajes del usuario entre dos fechas.

Method	POST
URL	/history/trips

Body params

Field	Type	Description
from	String	Fecha de inicio del intervalo para obtener los viajes.
to	String	Fecha de finalización del intervalo para obtener los viajes.

Success Response: OK 200

Nos retorna una lista con todos los viajes del usuario entre las dos fechas seleccionadas.

Field	Type	Description
trips	List<DataTrip>	Lista de viajes del usuario entre las dos fechas seleccionadas.

Update Positions

Envía las posiciones recogidas durante el viaje del usuario.

Method	POST
URL	/tracking

Body params

Field	Type	Description
id_vehicle	String	Identificador del vehículo con el que se está viajando.
positions	List<DataTripsPosition>	Lista de posiciones que se enviarán al servidor.

Success Response: OK 200

No devuelve nada más que el código de respuesta.

User Service

Get User Profile

Obtenemos el perfil del usuario.

Method	GET
URL	/profile

Success Response: OK 200

Nos retorna una lista con el perfil del usuario.

Field	Type	Description
username	String	Nombre de usuario del usuario.
name	String	Nombre del usuario.
middlename	String	Apellidos del usuario.
email	String	Correo electrónico del usuario.
gender	String	Género del usuario.
age	String	Edad del usuario.
height	String	Altura del usuario.
weight	String	Peso del usuario.

Recover Password

Recuperamos la contraseña del usuario.

Method	POST
URL	/auth/recoverpassword

Body params

Field	Type	Description
username	String	Nombre de usuario del usuario.

Success Response: OK 200

No devuelve nada más que el código de respuesta.

Change Password

Cambiamos la contraseña del usuario, este servicio tiene el mismo diseño que el anterior pero están implementados de forma separada para poder modificar este servicio en un futuro, así no necesitaremos recibir un correo electrónico para cambiar la contraseña.

Method	POST
URL	/auth/recoverpassword

Body params

Field	Type	Description
username	String	Nombre de usuario del usuario.

Success Response: OK 200

No devuelve nada más que el código de respuesta.

Update Profile

Actualizamos el perfil del usuario con los cambios que haya realizado.

Method	POST
URL	/profile

Body params

Field	Type	Description
height	Int	Altura del usuario.
weight	Int	Peso del usuario.

Success Response: OK 200

Nos retorna una lista con el perfil del usuario.

Field	Type	Description
username	String	Nombre de usuario del usuario.
name	String	Nombre del usuario.
middlename	String	Apellidos del usuario.
email	String	Correo electrónico del usuario.
gender	String	Género del usuario.
age	String	Edad del usuario.
height	String	Altura del usuario.
weight	String	Peso del usuario.