UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

GRADUADO EN INGENIERÍA DEL SOFTWARE

Mapa de concentración de población a partir de datos de aplicaciones móviles

Population concentration map from mobile application data

Realizado por
Luis Ramos Matas

Tutorizado por
Gabriel Jesús Luque Polo

Departamento
Lenguajes y Ciencias de la Computación
UNIVERSIDAD DE MÁLAGA

MÁLAGA, JUNIO DE 2023

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA DEL SOFTWARE

Mapa de concentración de población a partir de datos de aplicaciones móviles

Population concentration map from mobile application data

Realizado por
Luis Ramos Matas

Tutorizado por
Gabriel Jesús Luque Polo

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, JUNIO DE 2023

Fecha defensa: junio de 2023

Resumen

En la actualidad, la gran mayoría de los ciudadanos tienen acceso a un dispositivo móvil que cuenta con múltiples sensores y otros elementos para recopilar información sobre su entorno y ubicación. A través de una aplicación móvil, es posible recopilar y analizar esta información para obtener datos valiosos que pueden ser utilizados por las autoridades de una ciudad para la gestión urbana.

En este TFG, hemos desarrollado una aplicación móvil distribuida que permita la obtención de datos sobre la localización de los usuarios a través de los sensores de los dispositivos de forma automática. Tras ser procesados, nos servirán para visualizarlos en un mapa, añadir filtros y generar gráficas y estadísticas que posibiliten a los ciudadanos y las autoridades analizar el grado de congestión de cada lugar del mapa.

Los usuarios también podrán crear y gestionar puntos favoritos, los cuales, serán actualizados de forma diaria y sin la intervención de este.

Esta aplicación móvil puede ser una herramienta valiosa para la gestión urbana. Las autoridades serán capaces de observar patrones repetidos con los que tomar decisiones informadas y mejorar la calidad de vida de los ciudadanos.

Palabras clave: aplicación móvil, densidad, mapa, sensores

Abstract

Nowadays, the vast majority of citizens have access to a mobile device that has multiple sensors and other elements to collect information about their environment and location. Through a mobile application, it is possible to collect and analyse this information to obtain valuable data that can be used by city authorities for urban management.

In this TFG, we have developed a distributed mobile application that allows obtaining data about the location of users through the sensors of the devices automatically. After being processed, we will visualise them on a map, add filters and generate graphs and statistics that enable citizens and authorities to analyse the degree of congestion at each location on the map.

Users will also be able to create and manage favourite spots, which will be updated on a daily basis and without the user's intervention.

This mobile application can be a valuable tool for urban management. Authorities will be able to observe repeated patterns in order to make informed decisions and improve the quality of life of citizens.

Keywords: mobile application, density, map, sensors

Índice

1

Introducción

En este capítulo se presentarán los motivos por los cuales se lleva a cabo el desarrollo de este proyecto, los objetivos que se pretenden cumplir al terminar este y la descripción de la estructura de este documento.

1.1 Motivación

Hoy en día la mayoría de los ciudadanos están en disposición de un dispositivo móvil, de hecho, según el último estudio del INE, el 99,5% de los hogares en España dispone de teléfono móvil [1]. Estos dispositivos disponen de una gran cantidad de sensores y otro tipo de elementos que pueden captar información de su alrededor, además de la capacidad de informar de la ubicación del usuario.

Está información, una vez anonimizada y tratada de forma adecuada, puede resultar muy útil para las autoridades de una ciudad ya que se podrían analizar estos datos con el fin de encontrar zonas o franjas horarias en las cuales hay una mayor concentración de personas. Esto podría ayudar a la gestión de la ciudad, ya que podría usarse para implantar nuevas estaciones de transporte público o trazar rutas óptimas sin pasar por vías muy congestionadas.

1.2 Objetivos

El objetivo de este TFG es el desarrollo de una aplicación distribuida con dos componentes principales: una componente móvil que se ejecuta en un dispositivo móvil de un ciudadano y una componente formada por un servidor que recolecta la información proporcionada.

La aplicación móvil tendrá dos propósitos. Por un lado, recopilará de forma automática y sin necesidad de intervención del usuario la información de los sensores, que será enviada al servidor junto a su ubicación (contando con el consentimiento del usuario). Por otro lado, contará con un interfaz donde podrá consultar un mapa en el cual se podrá visualizar la densidad de población en cada instante y cada lugar de la ciudad, visualización datos estadísticos y otros potenciales datos derivados como puede ser una predicción de la densidad en una zona o franja horaria.

El otro componente como hemos dicho será un servidor que debe estar en funcionamiento de forma continua y que recibirá los datos aportados por las aplicaciones móviles. Estos datos serán tratados por el servidor para obtener una estimación de la concentración de personas cercana a la ubicación donde se recolectaron los datos. También obtendrá diferentes estadísticas de ellos y otros datos derivados.

1.3 Estructura de la memoria

En esta memoria está toda la información necesaria para entender la finalidad y las herramientas y métodos empleados para llevar a cabo el desarrollo de la aplicación. Por ello, este documento se divide de la siguiente forma:

- **Capítulo 1 - Introducción:** en este apartado hemos explicado la motivación que hace llevar a cabo el desarrollo de este proyecto y el objetivo que se quiere obtener como resultado de este.
- **Capítulo 2 - Tecnologías y herramientas:** definiremos que estructuras, tecnologías y herramientas se han empleado durante las fases de producción.
- **Capítulo 3 - Metodología:** capítulo en el que hablaremos de la metodología utilizada, la estrategia y conjunto de operaciones de gestión del proyecto para generar resultados periódicamente.

- **Capítulo 4 - Análisis:** continuaremos indicando el conjunto de funcionalidades y flujo de acciones que debe cumplir la aplicación al final de su desarrollo.
- **Capítulo 5 - Aplicación:** en este capítulo se describen los elementos que conforman tanto el cliente como el servidor.
- **Capítulo 6 - Conclusiones y líneas futuras:** para finalizar, comentaremos las conclusiones finales a las que se han llegado tras finalizar el desarrollo de este proyecto y mencionaremos las potencialidades de los sistemas producidos.

También se han añadido 3 apéndices:

- **Apéndice A - Manual de usuario:** documentación que permite obtener los conocimientos necesarios para utilizar la aplicación.
- **Apéndice B - Manual de despliegue del servidor:** pasos a seguir para poner en funcionamiento el servidor de forma local.
- **Apéndice C - Manual de despliegue del cliente:** pasos a seguir para poner en marcha el cliente en un dispositivo móvil real.

2

Tecnologías y herramientas

En este capítulo se comentarán las tecnologías y herramientas utilizadas, indicando tanto los lenguajes de programación y bibliotecas aplicadas como las aplicaciones web y de escritorio para llevar a cabo la implementación del código y la gestión del proyecto.

2.1 Servidor backend

2.1.1 Python

Python [2] es un lenguaje de programación muy popular que se caracteriza por su sintaxis clara y legible. Se usa en muchas aplicaciones, desde la ciencia de datos y el aprendizaje automático hasta la creación de juegos y la automatización de tareas.

Una de las ventajas de Python es su facilidad de aprendizaje, lo que hace que el proceso de codificación sea más rápido y eficiente. Además, tiene muchas bibliotecas y módulos estándar que permiten realizar tareas complejas de forma sencilla.

Python es multiplataforma, es decir, se puede utilizar en sistemas operativos. También, cuenta con una gran comunidad de desarrolladores que contribuyen a su desarrollo y mejora continua, y que proporcionan recursos como documentación, tutoriales y bibliotecas que facilitan la programación en Python.

Figura 2.1: Logo de Python

Hemos elegido Python como lenguaje para desarrollar la parte del servidor por varias razones:

- No se necesitan conocimientos muy extendidos para producir código.
- Cuenta con bibliotecas que permiten la automatización de tareas de forma periódica.
- Sencillez a la hora de crear aplicaciones REST.

2.1.2 FastAPI

FastAPI [3] es un *framework* de desarrollo web para Python que se ha vuelto muy popular gracias a su velocidad, facilidad de uso y eficiencia. Permite crear aplicaciones web de alto rendimiento que utilizan la validación de datos y el asincronismo para una gestión eficiente de solicitudes.

Una de las características que hacen interesante el uso de FastAPI es su capacidad para generar automáticamente una documentación interactiva de la API, lo que simplifica enormemente su comprensión y uso. Además, es fácil de aprender y utilizar, gracias a su documentación detallada y una curva de aprendizaje baja.

FastAPI es una opción ideal para aplicaciones de alta concurrencia, ya que permite una gestión eficiente de múltiples solicitudes simultáneas. También cuenta con tipado de datos, lo que permite una validación más estricta de los datos y una mejor documentación de la API.

En resumen, FastAPI es un *framework* web que permite crear aplicaciones web rápidas y eficientes con una documentación interactiva de la API y una fácil gestión de solicitudes

simultáneas. Es una herramienta valiosa para desarrolladores que buscan un *framework* web de alto rendimiento y fácil de aprender.

Figura 2.2: Logo de FastAPI

Hemos decidido utilizar esta tecnología por las siguientes razones:

- Genera documentación de la API de forma rápida y sencilla.
- Tiene una curva de aprendizaje baja.
- Gestión óptima de llamadas múltiples al servidor.
- Permite el control de los tipos de datos utilizados.

2.1.3 MongoDB

MongoDB [4] es un sistema de gestión de bases de datos NoSQL que ha ganado popularidad en los últimos años gracias a su arquitectura de almacenamiento de documentos. A diferencia de las bases de datos relacionales que utilizan tablas y filas para almacenar datos, MongoDB guarda los datos en documentos JSON (BSON) que se pueden organizar en colecciones.

Lo que hace que MongoDB sea una solución de bases de datos única son sus características distintivas. Por ejemplo, el esquema flexible permite que MongoDB agregue nuevos campos y estructuras de datos sin tener que modificar el esquema existente. MongoDB también ofrece alta disponibilidad y tolerancia a fallos gracias a su arquitectura de replicación y fragmentación.

MongoDB puede distribuir los datos a través de varios servidores, lo que permite una escalabilidad horizontal fácil y sin interrupciones. Además, MongoDB proporciona consultas avanzadas, incluyendo consultas ad hoc, indexación y agregación.

MongoDB es fácil de usar y cuenta con una documentación completa que facilita su adopción. Además, la comunidad de desarrolladores detrás de MongoDB es muy activa, lo que significa que hay mucho soporte y colaboración disponible para los desarrolladores.

Figura 2.3: Logo de MongoDB

Las razones por las cuales hemos optado por utilizar este tipo de almacenamiento son las siguientes:

- La escalabilidad, flexibilidad y rapidez que proporciona hace que los tipos de datos y su volumen se adapten perfectamente a esta tecnología.
- Podemos realizar consultas muy avanzadas y potentes relacionadas con geolocalización.
- Contábamos con experiencia previa en esta herramienta.

2.2 Cliente frontend

2.2.1 JavaScript

JavaScript [5] es un lenguaje de programación utilizado para crear interactividad en sitios web y aplicaciones web. JavaScript se utiliza para crear efectos visuales dinámicos, validación de formularios, intercambio de datos asíncronos y otros comportamientos interactivos en las páginas web.

JavaScript se ejecuta en el lado del cliente, lo que significa que se ejecuta en el navegador web del usuario en lugar de en un servidor web. Como resultado, JavaScript es un lenguaje de programación esencial para la creación de aplicaciones web modernas. Además, se utiliza ampliamente en tecnologías web como Node.js, que permite a los desarrolladores utilizar JavaScript en el lado del servidor.

JavaScript es un lenguaje de programación interpretado, lo que significa que no se compila antes de la ejecución. En cambio, el código JavaScript se ejecuta directamente en el navegador web del usuario. JavaScript también es un lenguaje de programación orientado a objetos, lo que significa que se pueden crear objetos con propiedades y métodos. Esto permite a los desarrolladores crear código JavaScript modular y reutilizable.

Figura 2.4: Logo de JavaScript

Hemos seleccionado este lenguaje por las siguientes razones:

- Existe un gran soporte por parte de la comunidad y gran cantidad de bibliotecas con las que desarrollar las funcionalidades deseadas.
- Contábamos con experiencia previa en este lenguaje.
- Es compatible con Expo, una plataforma de desarrollo de aplicaciones móviles que se describirá más adelante.

2.2.2 React Native

React Native [6] es un *framework* de código abierto para el desarrollo de aplicaciones móviles. Utiliza la biblioteca de React, que es una biblioteca de JavaScript de código abierto, para desarrollar aplicaciones móviles para iOS y Android con un enfoque en el rendimiento y la velocidad.

La principal ventaja de React Native es que permite a los desarrolladores crear aplicaciones móviles utilizando un único código base de JavaScript, lo que les permite desarrollar aplicaciones móviles para múltiples plataformas de forma eficiente y con menos esfuerzo. Esto significa que los desarrolladores pueden crear aplicaciones móviles para iOS y Android sin tener que aprender diferentes lenguajes de programación o utilizar diferentes herramientas de desarrollo.

React Native también ofrece una serie de componentes y bibliotecas preconstruidos, lo que facilita a los desarrolladores la creación de interfaces de usuario sofisticadas y la integración de características complejas en sus aplicaciones móviles.

Figura 2.5: Logo de React Native

El motivo por el que hemos elegido este *framework* para la implementación del cliente es la gran cantidad de bibliotecas y recursos que presenta para crear interfaces, gráficas y otros elementos visuales.

2.2.3 Expo

Expo [7] es una plataforma que ofrece una serie de herramientas y servicios para el desarrollo de aplicaciones móviles nativas. Utilizando Expo, los desarrolladores pueden crear aplicaciones para iOS, Android y la web, utilizando tecnologías como JavaScript y React Native.

Lo que hace a Expo especialmente interesante es que ofrece una variedad de herramientas y servicios que permiten a los desarrolladores centrarse en la creación de aplicaciones de alta calidad, en lugar de preocuparse por la configuración de herramientas y servicios. Expo cuenta con una interfaz de línea de comandos para el desarrollo de aplicaciones, un cliente de aplicaciones móviles para previsualizar y probar aplicaciones en tiempo real, y una amplia gama de bibliotecas y componentes preconstruidos que permiten a los desarrolladores crear aplicaciones móviles rápidamente y sin complicaciones.

Expo también proporciona una serie de herramientas para el desarrollo de aplicaciones móviles, como herramientas de depuración, pruebas y análisis de rendimiento. Además, cuenta con una comunidad activa de desarrolladores que proporciona soporte, tutoriales y

recursos adicionales para ayudar a los desarrolladores a crear aplicaciones móviles de alta calidad.

Figura 2.6: Logo de Expo

Las ventajas que nos proporciona esta plataforma que han hecho que nos decantemos por esta opción son las siguientes:

- Expo nos permite centrarnos en el desarrollo de la aplicación ya que simplifica el proceso de configuración del entorno.
- Cuenta con multitud de bibliotecas y componentes preconstruidos, los cuales hemos utilizado para llevar a cabo funcionalidades como la geolocalización y las tareas en segundo plano.

2.3 Bibliotecas de desarrollo

2.3.1 Bibliotecas usadas en el backend

Debemos resaltar la importancia que han tenido las bibliotecas utilizadas en la parte del servidor. Estas herramientas han permitido que el *backend* pudiera efectuar su tarea como aplicación REST API y encargarse de llevar a cabo las tareas automáticas diarias.

- Como primer punto, tenemos en cuenta la bibliotecas del *framework* FastAPI. Gracias al enfoque que tiene, hemos creado de forma rápida y sencilla toda la estructura base del servidor, nos permite evitar muchos errores debido al robusto control de los tipos que se utilizan tanto en los modelos como en las rutas y nos ha facilitado su documentación mediante el *swagger* que se genera a la par que creábamos las funcionalidades.

- PyMongo [8] es la siguiente biblioteca de la que vamos a hablar. Esta biblioteca nos permite trabajar con la base de datos MongoDB de manera fácil y eficiente mediante la utilización de consultas que pueden ir desde una simple obtención de todos los documentos de una colección hasta el filtrado avanzado de puntos de un mapa que se encuentran dentro de una circunferencia dado un radio y su punto central.
- Scheduler [9] es una biblioteca que permite al desarrollador crear programas que puedan ejecutar tareas periódicas o en momentos específicos. Es ideal para programar tareas automáticas. Para que funcionara correctamente, la hemos combinado con la función que nos permite crear hilos de Python.

2.3.2 Bibliotecas usadas en el frontend

Al igual que en el *backend*, el uso de bibliotecas en el cliente también ha tenido una gran relevancia. Hemos utilizado elementos creados por la comunidad para la creación de las interfaces, listados y formularios, además de componentes como el mapa de Google o las gráficas para representar datos.

- Axios [10] es una bibliotecas que nos permite realizar solicitudes HTTP a un servidor. Es popular debido a su facilidad de uso y por su capacidad para trabajar con diversos tipos de datos de manera efectiva. También, permite la cancelación de estas y la gestión de errores.
- Async Storage [11] es una herramienta que se utiliza en aplicaciones móvil desarrolladas con React Native. Nos permite guardar y recuperar datos mediante funciones asíncronas en el dispositivo del usuario. En nuestro caso, la información que almacenamos es el id de usuario mediante el cual se obtiene la mayoría de los datos del resto de funcionalidades.
- React Native Material [12] y React Native Elements [13] son dos bibliotecas de componentes de interfaz de usuario que se utilizan en aplicaciones móviles desarrolladas en React Native. Desde botones o elementos con animaciones hasta contenedores más complejos que son conjuntos de los anteriores con la finalidad de crear interfaces atractivas y eficientes.

- React Navigation [14] es una biblioteca de enrutamiento y navegación para aplicaciones desarrolladas en React Native. Nos ha permitido la navegación fácil y su gestión eficiente a la hora de cambiar de vistas y pasar datos entre estas.
- React Hook Form [15] es una biblioteca de formularios que utiliza los hooks de React para facilitar su creación. La hemos utilizado ya que proporciona una forma sencilla de manejar la validación y el envío de datos de estos formularios. Gracias a su sintaxis clara y simple hemos podido crear un formulario de creación y modificación de los puntos favoritos del usuario.
- Expo Location [16] es una biblioteca de Expo que proporciona una forma fácil de acceder a la información de la ubicación en aplicaciones móviles. Esta biblioteca permite a los desarrolladores de aplicaciones móviles de React Native acceder a la ubicación del dispositivo y utilizar esta información para crear funciones relacionadas con la ubicación, como la navegación, la búsqueda y el seguimiento de la ubicación.
- React Native Maps [17] es una biblioteca para React Native que proporciona una interfaz para interactuar con mapas en aplicaciones móviles. Permite a los desarrolladores agregar mapas interactivos y personalizados. Algunas de las características que ofrece incluyen la definición de marcadores, la visualización de información de ubicación, la búsqueda de lugares, la creación de rutas, la integración con geolocalización y la compatibilidad con diferentes tipos de mapas (como Google Maps y Apple Maps). Hemos utilizado esta biblioteca para la implementación del mapa con los puntos de calor y la visualización de puntos favoritos personalizados con los marcadores y su respectivo radio de medición.
- React Native Chart Kit [18] es una biblioteca de gráficos para React Native que permite a los desarrolladores crear gráficos y visualizaciones de datos personalizados en sus aplicaciones móviles. Esta biblioteca ofrece una amplia gama de tipos de gráficos, como gráficos de barras, líneas, áreas, pastel, radar y más, así como diferentes estilos y opciones de personalización. Gracias a su facilidad de uso y personalización, React Native Chart Kit se ha convertido en una de las bibliotecas de gráficos más populares para aplicaciones móviles de React Native. Con ella, hemos conseguido implementar varios gráficos para mostrar datos referentes a los puntos favoritos del usuario.

2.4 Herramientas de desarrollo

A continuación, procederemos a explicar cuáles han sido las herramientas que se han utilizado para la implementación del código y el diseño de algunos elementos gráficos de la aplicación.

2.4.1 Visual Studio Code

Visual Studio Code [19] es un editor de código fuente gratuito y de código abierto desarrollado por Microsoft. Es una herramienta muy popular utilizada por desarrolladores de todo el mundo para crear y editar programas de software, aplicaciones web y sitios web.

Cuenta con una comunidad activa de desarrolladores que crean y mantienen extensiones para ampliar la funcionalidad del editor. Estas extensiones pueden añadir soporte para diferentes lenguajes de programación, integración con diferentes herramientas y servicios, y otras características adicionales como, por ejemplo, subir o descargarnos el contenido de un repositorio de GitHub o hacer pruebas de la API mediante la extensión Thunder Client [20].

Las razones por las cuales hemos optado por utilizar este editor es su fácil manejo, la experiencia previa que teníamos con él, la fácil gestión de los cambios del repositorio y la agilidad en las pruebas de la API.

Figura 2.7: Logo de Visual Studio Code

2.4.2 Figma

Figma [21] es una herramienta de diseño de interfaces de usuario y colaboración en línea que permite a los usuarios crear, compartir y colaborar en diseños de alta calidad en tiempo real. Es una herramienta popular utilizada por diseñadores, desarrolladores y equipos de proyectos para crear y compartir diseños de interfaces de usuario para aplicaciones web y

móviles. Ofrece una amplia variedad de características, incluyendo herramientas de dibujo, iconos, componentes, fuentes, animaciones, prototipado y más.

Figura 2.8: Logo de Figma

Hemos recurrido a esta aplicación web para crear los diseños del icono de la aplicación y los marcadores favoritos.

2.5 Herramientas auxiliares

2.5.1 GitHub

GitHub [22] es una plataforma de alojamiento y colaboración de código fuente para proyectos de software. Es un lugar donde los desarrolladores pueden alojar y gestionar sus proyectos de código fuente, así como colaborar con otros desarrolladores en proyectos de código abierto y privados.

GitHub se basa en el sistema de control de versiones Git, que permite a los desarrolladores llevar un registro de los cambios realizados en su código fuente y trabajar en diferentes versiones de un proyecto de forma eficiente.

Figura 2.9: Logo de GitHub

En nuestro caso, la colaboración no es la ventaja que nos aporta. Hemos utilizado GitHub por la experiencia que tenemos con esta herramienta, por su fácil manejo y la gestión de versiones del código en combinación con Visual Studio Code.

2.5.2 Trello

Trello [23] es una herramienta de gestión de proyectos en línea que permite a los usuarios organizar y visualizar sus proyectos en un tablero virtual. Es una forma fácil y visual de llevar un registro de tareas, proyectos y actividades en equipo, y está diseñada para mejorar la colaboración y la productividad.

La utilización de esta herramienta nos ha beneficiado a la hora de organizar las tareas a lo largo del desarrollo del proyecto y para anotar avances, cambios y errores de las funcionalidades producidas.

Figura 2.10: Logo de Trello

2.5.3 Microsoft Word

Microsoft Word [24] es un procesador de texto de los más utilizados en el mundo para la creación, edición y formateo de documentos de texto, como cartas, informes, currículums, trabajos escolares y otros tipos de documentos. Ofrece una amplia variedad de herramientas para dar formato a estos, como estilos de fuente, tamaños, colores, márgenes, interlineado, viñetas y numeración, entre otras.

Esta aplicación ha sido la que hemos utilizado para el desarrollo de este documento.

Figura 2.11: Logo de Microsoft Word

2.5.4 Mockaroo

Mockaroo [25] es una herramienta en línea que permite generar datos de prueba de manera rápida y sencilla. Con Mockaroo, hemos podido crear conjuntos de datos personalizados con una gran cantidad de opciones de configuración, como el tipo de datos que se generarán, el número de filas de datos y las restricciones de formato.

Además, Mockaroo también permite exportar los datos generados en diferentes formatos, como CSV, JSON o SQL. Es útil para probar aplicaciones, bases de datos y para hacer demostraciones de cómo se verían los datos en una aplicación sin tener que ingresar manualmente cada uno de ellos.

Figura 2.12: Logo de Mockaroo

3

Análisis

Una parte importante del proceso de desarrollo de cualquier tipo de software es la definición de las funcionalidades, es decir, qué es lo que debe hacer nuestra aplicación para alcanzar los objetivos marcados al inicio del proyecto. También definiremos los casos de uso los cuales nos ayudarán a entender cómo reaccionará el sistema al interactuar con él.

3.1 Requisitos

Los requisitos se clasifican en dos tipos. Por un lado, tenemos los requisitos funcionales aquellos que nos sirven para definir lo que debe hacer un sistema. Por otro lado, los requisitos no funcionales los encargados de definir las características de este.

3.1.1 Requisitos funcionales

Los requisitos funcionales que hemos definido para esta aplicación son los siguientes:

- RF1: Un usuario podrá ver localizaciones de otros usuarios en forma de puntos de calor sobre un mapa.
- RF2: Un usuario podrá filtrar por tiempo los puntos de calor sobre el mapa.
- RF3: Un usuario podrá visualizar información de un punto cualquiera del mapa.
- RF4: Un usuario podrá guardar cualquier punto del mapa como localización favorita.
- RF5: Un usuario podrá editar cualquier punto favorito.
- RF6: Un usuario podrá visualizar todos los puntos favoritos en forma de marcadores sobre el mapa.
- RF7: Un usuario podrá limpiar todos los marcadores existentes sobre el mapa.
- RF8: Un usuario podrá acceder a la información de los puntos favoritos en el mapa.

- RF9: Un usuario podrá visualizar un listado de sus puntos favoritos.
- RF10: Un usuario podrá eliminar sus puntos favoritos.
- RF11: Un usuario podrá ver datos históricos de sus puntos favoritos representados en gráficos.

3.1.2 Requisitos no funcionales

Los requisitos no funcionales que hemos definido para este sistema son los siguientes:

- RNF1: La aplicación funcionará en dispositivos móviles que dispongan de sensores GPS activos.
- RNF2: La aplicación funcionará en dispositivos móviles que tengan acceso a Internet.
- RNF3: El registro y actualización de sesión del usuario se realizará de forma automática sin intervención del usuario al ejecutar la aplicación.
- RNF4: El sistema usará el almacenamiento local para guardar y recuperar el identificador del usuario.
- RNF5: El guardado de datos históricos de los puntos se realizará de forma automática a las 6 de la tarde de cada día.
- RNF6: Los datos se almacenarán en una base de datos NoSQL como MongoDB.
- RNF7: El servidor se desarrollará en Python usando FastAPI.
- RNF8: El cliente se desarrollará en JavaScript usando React Native y Expo.

3.2 Casos de uso

A continuación, mostraremos una serie de tablas que describirán las posibles interacciones del usuario con el sistema y las diferentes casuísticas que se podrán dar.

Tabla 3.1: Filtrar puntos de calor del mapa

Título	Filtrar puntos de calor del mapa
Descripción	El usuario filtra los puntos de calor que aparecen en el mapa pulsando una de las opciones del selector.
Requisitos	RF1, RF2
Precondición	Iniciar la aplicación
Postcondición	La cantidad de puntos de calor mostrados en el mapa cambia

Escenario principal
1. El usuario pulsa en “Mapa”. 2. La aplicación muestra un mapa con los puntos de calor guardados las últimas 24 horas. 3. El usuario pulsa en “Filtro”. 4. La aplicación despliega un listado de opciones con intervalos de tiempo. 5. El usuario pulsa en una de las opciones. 6. La aplicación muestra el mapa con los nuevos puntos de calor guardados en ese intervalo de tiempo.

Tabla 3.2: Guardar punto del mapa como favorito

Título	Guardar punto del mapa como favorito
Descripción	El usuario selecciona una posición del mapa y la guarda como punto favorito
Requisitos	RF3, RF4
Precondición	Iniciar la aplicación Entrar en la vista “Mapa”
Postcondición	Aparece un modal dando información al usuario del resultado
Escenario principal	
1. El usuario pulsa en una posición del mapa 2. El sistema muestra un marcador en la posición seleccionada 3. El usuario pulsa en el marcador 4. El sistema muestra un botón de opciones sobre el marcador 5. El usuario pulsa en el botón “Opciones” 6. El sistema muestra un modal con información del punto y un botón “Añadir a favoritos” 7. El usuario pulsa el botón “Añadir a favoritos” 8. El sistema muestra un formulario con dos campos 9. El usuario introduce un nombre para el punto válido	

10. El usuario introduce un radio de medición válido 11. El usuario pulsa “Guardar” 12. El sistema muestra la vista de “Mapa” 13. El sistema muestra un modal de éxito
Escenario alternativo
Campo deja un campo vacío 9. El usuario deja uno de los campos vacío 10. El usuario pulsa el botón “Guardar” 11. El sistema muestra un mensaje de error debajo del campo vacío 12. El usuario continua por el paso 9 del escenario principal **Campo “Radio de medición” inválido** 10. El usuario introduce un número entero inferior a 1 11. El usuario pulsa el botón “Guardar” 12. El sistema muestra un mensaje de error debajo del campo “Radio de medición” 13. El usuario continua por el paso 9 del escenario principal

Tabla 3.3: Mostrar marcadores favoritos en el mapa

Título	Mostrar marcadores favoritos en el mapa
Descripción	El usuario pulsa en “Mostrar favoritos” y aparecen marcadores con las localizaciones favoritas en el mapa
Requisitos	RF6
Precondición	Iniciar la aplicación Entrar en la vista “Mapa”
Postcondición	El mapa muestra marcadores de localizaciones favoritas
Escenario principal	
1. El usuario pulsa el botón “Acciones” 2. El sistema muestra un listado de botones con acciones de marcadores 3. El usuario pulsa en “Mostrar favoritos” 4. El sistema muestra marcadores personalizados en el mapa	

Tabla 3.4: Eliminar marcadores del mapa

Título	Eliminar marcadores del mapa
Descripción	El usuario pulsa en “Limpiar marcadores” para hacer desaparecer todos los marcadores del mapa
Requisitos	RF6, RF7
Precondición	Iniciar la aplicación Entrar en la vista “Mapa” Mostrar los puntos favoritos del usuario Seleccionar una ubicación del mapa
Postcondición	Los marcadores son eliminados del mapa
Escenario principal	
1. El usuario pulsa en “Acciones” 2. El sistema muestra un listado de botones con acciones de marcadores 3. El usuario pulsa en “Limpiar marcadores” 4. El sistema esconde el listado de acciones 5. El sistema elimina todos los marcadores del mapa	

Tabla 3.5: Visualizar lista de favoritos

Título	Visualizar lista de favoritos
Descripción	El usuario visualiza una lista de sus puntos favoritos al entrar en la vista “Favoritos”
Requisitos	RF4, RF9
Precondición	Iniciar la aplicación Haber creado un punto favorito
Postcondición	Mostrar un listado de puntos favoritos
Escenario principal	
1. El usuario pulsa en el botón “Favoritos” en el menú principal 2. El sistema muestra la vista “Favoritos”	

Tabla 3.6: Eliminar puntos favoritos

Título	Eliminar puntos favoritos
Descripción	El usuario elimina puntos favoritos desde la vista “Favoritos”

Requisitos	RF4, RF9, RF10
Precondición	Iniciar la aplicación Haber creado un punto favorito Entrar en la vista “Favoritos”
Postcondición	Aparece un modal dando información al usuario del resultado
Escenario principal	
1. El usuario pulsa en el checkbox “+” de un punto favorito del listado 2. El sistema cambia el signo “+” por un signo “-” 3. El usuario pulsa en el botón “Eliminar” 4. El sistema muestra un modal de éxito	
Escenario alternativo	
Eliminar más de 1 punto favorito 1. El usuario pulsa en el checkbox “+” de varios puntos favoritos del listado 2. El usuario continua por el paso 2 del escenario principal	

Tabla 3.7: Mostrar gráfica de tendencias

Título	Mostrar gráfica de tendencias
Descripción	El usuario visualiza los datos históricos de los puntos favoritos representados en una gráfica de líneas
Requisitos	RF4, RF11
Precondición	Iniciar la aplicación Haber creado un punto favorito Haber generado datos históricos
Postcondición	Visualizar una gráfica de líneas representando los datos históricos desde 7 días atrás hasta el día actual
Escenario principal	
1. El usuario pulsa en el botón “Gráficas” del menú principal 2. El sistema muestra la vista de “Gráficas”	

Tabla 3.8: Mostrar gráfica de densidades

Título	Mostrar gráfica de densidades

Descripción	El usuario visualiza la densidad de los puntos favoritos representados en una gráfica de pastel
Requisitos	RF4, RF11
Precondición	Iniciar la aplicación Haber creado un punto favorito Entrar en la vista “Gráficas”
Postcondición	Visualizar una gráfica de pastel representando las densidades actuales de los puntos favoritos
Escenario principal	
1. El usuario pulsa en el botón “Densidades” de la barra de navegación de la parte superior de la pantalla 2. El sistema muestra la gráfica de densidades	

Tabla 3.9: Mostrar gráfica de usuarios

Título	Mostrar gráfica de usuarios
Descripción	El usuario visualiza los usuarios que han pasado por los puntos favoritos representados en una gráfica de contribución
Requisitos	RF4, RF11
Precondición	Iniciar la aplicación Haber creado un punto favorito Entrar en la vista “Gráficas”
Postcondición	Visualizar una gráfica de contribución representando los usuarios que han pasado por los puntos favoritos clasificado por días
Escenario principal	
1. El usuario pulsa en el botón “Usuarios” de la barra de navegación de la parte superior de la pantalla 2. El sistema muestra la gráfica de usuarios 3. El usuario pulsa en el selector de puntos favoritos 4. El sistema despliega una lista con los puntos favoritos 5. El usuario pulsa en una de las opciones	

6. El sistema muestra los usuarios que han pasado por ese punto favorito

4

Metodología

En este capítulo hablaremos sobre la metodología de trabajo que se ha seguido a lo largo del ciclo de vida del proyecto. La metodología que hemos usado nos ha facilitado la organización del desarrollo de funcionalidades y su distribución en el tiempo, haciendo un análisis y priorización de tareas según las necesidades que nos han ido aconteciendo.

4.1 Metodología ágil

Lo primero que vamos a explicar en este capítulo es el concepto de metodología ágil. Una metodología ágil [26] es un enfoque de gestión de proyectos que se centra en la entrega rápida y continua de productos o servicios de alta calidad a través de la colaboración y el trabajo en equipo.

Scrum [27] es un marco de trabajo ágil para la gestión de proyectos de software que se basa en la entrega iterativa e incremental de un producto. El equipo Scrum trabaja en *sprints*, períodos de tiempo cortos (usualmente de 1 a 4 semanas) en los cuales se planifica, se desarrolla, se prueba y se entrega un conjunto de funcionalidades del producto. Durante cada *sprint*, el equipo Scrum se reúne diariamente para revisar el progreso y planificar el trabajo para el día siguiente.

En nuestro caso, hemos aplicado Scrum adaptado a una persona que cubre todos los roles excepto el de *Product Owner* que ha sido asignado al tutor. Las reuniones fijadas por Scrum,

también se han modificado atendiendo a este reparto de roles, y se han realizado reuniones periódicas con el tutor con el fin de mostrar y validar las funcionalidades acordadas.

4.2 Gestión del proyecto

Para llevar un seguimiento de las tareas que estaban en proceso y las que terminábamos, decidimos apoyarnos en la aplicación Trello. Hemos utilizado un tablero cuyas columnas principales son: *To Do*, *Doing* y *Done.* El resto de las columnas han servido para señalar cambios en funcionalidades ya implementadas, revisiones o fases de testeo de estas.

Las funcionalidades las hemos representado como tarjetas, anotando en ellas que esperábamos implementar, como se ha llevado a cabo y que hemos obtenido como resultado final.

4.3 Iteraciones

A continuación, describiremos las fases o iteraciones del proyecto, también conocidas como *sprints* en el marco de trabajo Scrum. Con la siguiente estructura, se puede destacar de forma más clara y organizada las partes del proyecto, así como las actividades realizadas en cada una de ellas.

4.3.1 Iteración 0

Esta iteración ha tenido como objetivo el análisis de la meta que teníamos que alcanzar, entendiendo por esto, la definición de las principales funcionalidades que debía cumplir la aplicación.

Una vez terminado este análisis, pasamos a reflexionar sobre si las herramientas y tecnologías que se habían pensado de antemano, al realizar el anteproyecto, eran las adecuadas para conseguir nuestro objetivo.

Tras un trabajo de investigación de diversas opciones, llegamos a la conclusión de que las herramientas descritas en este documento, las cuales han sido finalmente utilizadas, eran las

correctas.

4.3.2 Iteración 1

El siguiente paso que dimos fue establecer las entidades y datos que se iban a manejar en un principio junto con la configuración de los proyectos *backend* y *frontend*.

Una vez inicializado el servidor, nos dispusimos a crear una primera versión de la API que iba a almacenar los datos básicos de la aplicación. Hicimos una primera prueba de API sin utilizar *frameworks.* Tras estudiar la complejidad de implementación y la poca claridad que añadía este hecho, decidimos desarrollar una aplicación API REST con la ayuda de FastAPI.

A la par que esto, tras terminar la configuración del *frontend,* lo primero que hicimos fue implementar un menú principal básico y el módulo de operaciones básicas de gestión de datos (también denominado CRUD) mediante la biblioteca Axios.

4.3.3 Iteración 2

Durante esta iteración empezamos a asentar lo que serían las bases de la vista del mapa y las tareas en segundo plano que se encargarían de recoger la información de las localizaciones del usuario.

Tras implementar estas funcionalidades, empezamos las primeras pruebas para cargar el mapa de calor con las localizaciones recogidas con la biblioteca Expo Location.

4.3.4 Iteración 3

El objetivo de esta fase fue la implementación de un sistema de creación y actualización del usuario al entrar en la aplicación. En esta tarea utilizamos al almacenamiento interno del dispositivo para guardar el identificador del usuario.

Preparamos esta funcionalidad de forma que fuera automática nada más acceder a la aplicación sin necesidad de registro o de guardar datos relevantes.

4.3.5 Iteración 4

En esta iteración tuvimos que añadir un nuevo tipo de documento a la base de datos, los puntos favoritos. Para ello, añadimos el CRUD necesario en la parte del servidor y las modificaciones necesarias en el módulo de llamadas a la API en el cliente.

En el cliente, añadimos varias funcionalidades:

- En primer lugar, implementamos que el usuario, pulsando en cualquier parte del mapa, podría añadir esa localización como punto favorito mediante un formulario simple.
- Por otro lado, tuvimos que crear una vista para gestionar estos puntos. Para ello, agregamos un listado de puntos favoritos en el que se muestran los datos correspondientes de cada uno de ellos y la capacidad de eliminar estos puntos.
- Por último, diseñamos un marcador personalizado para el punto favorito, los cuales se muestran en el mapa si el usuario pulsa en el botón correspondiente.

4.3.6 Iteración 5

En esta penúltima iteración, añadimos un nuevo tipo de entidad a la base de datos, los checkpoints. Estos documentos se encargan de guardar la información de cada uno de los puntos favoritos de forma diaria.

Para llevar a cabo esta funcionalidad, utilizamos la biblioteca *scheduler* de Python para crear una tarea periódica diaria que ejecuta una actualización masiva de todos los puntos favoritos cuando llegan las 6 de la tarde.

Gracias a este nuevo tipo de datos, pudimos implementar una nueva vista en el cliente, la vista de Gráficas en las cuales se pueden visualizar los datos recogidos en esas actualizaciones de forma rápida e interactiva.

4.3.6 Iteración 5

La última iteración nos sirvió para llevar a cabo pruebas y correcciones menores a nivel de diseño y estética.

5

Aplicación

Hemos explicado las funcionalidades y el comportamiento que debe tener el sistema. En este capítulo hablaremos del diseño de la aplicación. La estructura está dividida en dos partes: una parte corresponde al servidor o *backend,* y la otra corresponde al cliente o *frontend*. Comentaremos las entidades almacenadas en la base de datos y sus relaciones, la organización que siguen las estructuras de ambas partes y las razones de esta.

5.1 Diseño base de datos

Lo primero que definiremos será el modelo de datos que hemos utilizado en nuestra aplicación. Como se puede ver en la Figura 5.1, la base de datos está formada por 4 colecciones: *Usuario, Log Usuario, Favorito* y *Checkpoint.*

Al ser un base de datos NoSQL, las claves foráneas de las relaciones se representan como atributos de las colecciones.

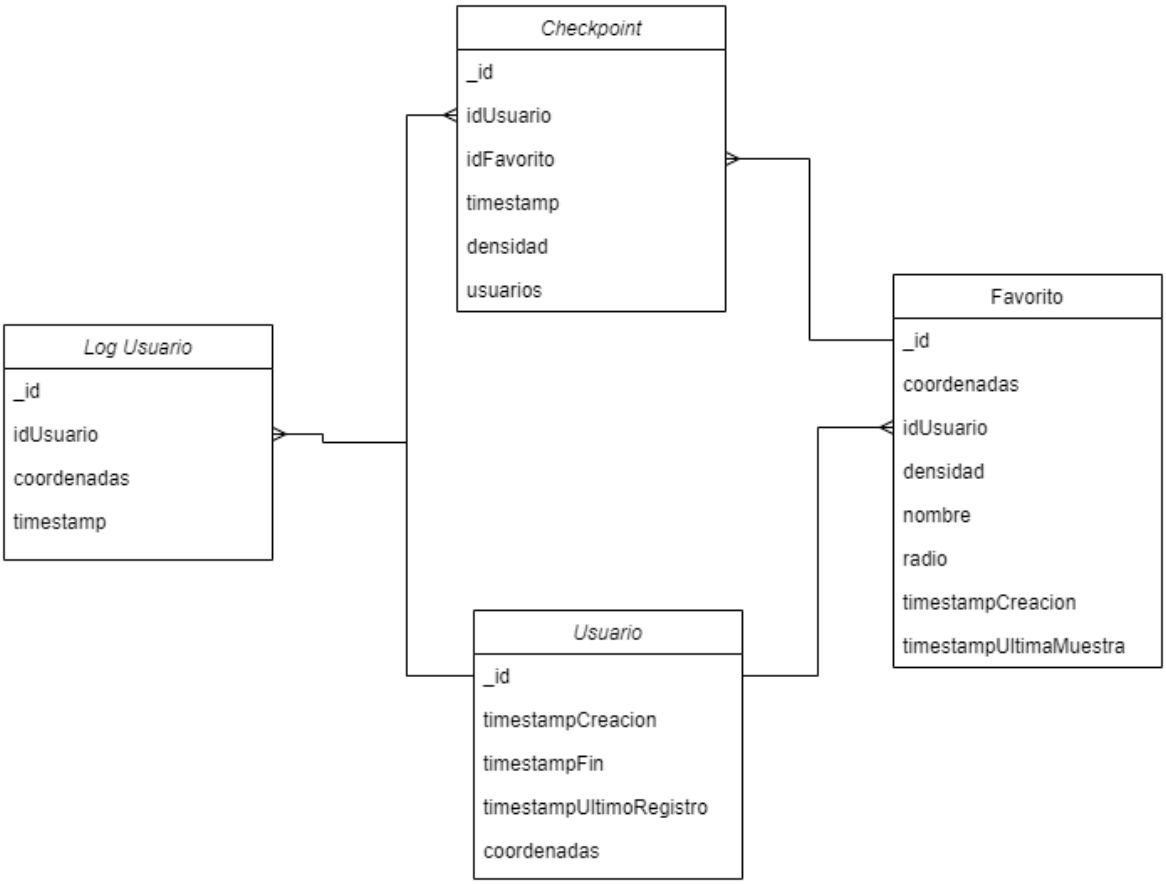

Figura 5.1: Diagrama Entidad Relación

5.1.1 Colección *Usuario*

La colección *Usuario* es la encargada de guardar la información básica de cada usuario de la aplicación. Tenemos que recalcar que en ningún caso guardamos información personal. El usuario es totalmente anónimo y el uso de su identificador es para la recuperación de datos del resto de entidades relacionadas con él.

- _id: identificador del usuario.
- timestampCreacion: fecha de creación del usuario, cuando inicia por primera vez la aplicación.
- timestampUltimoRegistro: representa la última vez que el usuario inicio la aplicación.

5.1.2 Colección *Log Usuario*

La colección *Log Usuario* es la encargada de guardar la ubicación del usuario y la fecha de creación del punto.

- _id: identificador del registro de ubicación del usuario.
- idUsuario: identificador del usuario que ha generado el registro.
- coordenadas: ubicación en la que se ha tomado el registro.
- timestamp: fecha en la cual se genera el registro.

5.1.3 Colección *Favorito*

La colección *Favorito* es la encargada de guardar la información de las ubicaciones personalizadas del usuario.

- _id: identificador del punto favorito del usuario.
- coordenadas: ubicación en la que se encuentra el punto favorito del usuario.
- idUsuario: identificador del usuario que registra el punto favorito.
- nombre: nombre que el usuario otorga al punto favorito.
- radio: radio de medición del punto favorito en metros.
- densidad: cantidad de usuarios por kilómetro cuadrado que se han encontrado dentro del radio de medición del punto favorito.
- timestampCreacion: fecha de creación del punto favorito.
- timestampUltimaMuestra: fecha de actualización de la densidad del punto favorito.

5.1.4 Colección *Checkpoint*

La colección *Checkpoint* es la encargada de guardar los datos de los puntos favoritos en forma de histórico.

- _id: identificador del checkpoint.
- idUsuario: identificador del usuario que ha creado el punto favorito.
- idFavorito: identificador del punto favorito al que está asociado el checkpoint.
- timestamp: fecha de creación del checkpoint.
- densidad: última densidad registrada del punto favorito.
- usuarios: número de usuarios que han pasado por el área de medición del punto favorito.

5.2 Diseño servidor

A la hora de desarrollar el *backend*, la idea desde el principio fue implementar una API REST que nos permitiera gestionar las comunicaciones con la base de datos para guardar y recuperar la información de las localizaciones e incorporar una función que creara un hilo paralelo para la actualización automática diaria de los puntos favoritos.

Con la ayuda de FastAPI, hemos podido crear una aplicación API REST de forma rápida, escalable, flexible y que permite la rápida comprensión del funcionamiento de cada módulo. FastAPI recomienda la división del servidor en tres módulos bien diferenciados: *models*, *schemas* y *routers*.

5.2.1 Models

Este módulo está compuesto por los modelos de datos que utilizamos en la aplicación. Estos modelos representan las entidades del dominio y están relacionados con los esquemas. La estructura de los documentos de nuestra base de datos serán iguales o similares, según el tipo de dato, como podemos observar en el ejemplo de la Figura 5.2.

```python
from pydantic import BaseModel
from bson import ObjectId
from models.coordenadas import Coordenadas

class LogUsuario(BaseModel):
    _id: ObjectId
    idUsuario: str
    coordenadas: Coordenadas
    timestamp: float
```

Figura 5.2: Ejemplo de modelo - LogUsuario

5.2.2 Schemas

Los esquemas describen la estructura y validaciones de los datos que se envían y reciben a través de la API. Estos esquemas los hemos implementado como funciones que nos sirven para controlar lo que queremos mandar al cliente como podemos ver en la Figura 5.3. Es la última capa por la que pasan los datos tras realizarse una llamada al servidor.

```python
def logUserEntity(item) -> dict:
    return {
        "_id":str(item["_id"]),
        "idUsuario":item["idUsuario"],
        "coordenadas":{
            "longitud":item["coordenadas"]["coordinates"][0],
            "latitud":item["coordenadas"]["coordinates"][1],
        },
        "timestamp": item["timestamp"],
    }

def logUsersEntity(entity) -> list:
    return [logUserEntity(item) for item in entity]
```

Figura 5.3: Ejemplo de esquema - LogUsuario

5.2.3 Routers

Los enrutadores son responsables de definir las rutas y los controladores asociados para cada *endpoint* de la aplicación. Hemos organizado los enrutadores según las entidades descritas por los modelos de datos.

Como podemos observar en la Figura 5.3, en los enrutadores hacemos uso de los modelos y esquemas para controlar la respuesta que devolvemos.

```python
logUser = APIRouter()

@logUser.get('/logUsers', response_model=list[LogUsuario], tags=["Log Usuarios"])
def get_all_log_users():
    return logUsersEntity(connection.PCM.LogUsuario.find())

@logUser.get('/logUsers/{timestamp}', response_model=list[LogUsuario], tags=["Log Usuarios"])
def get_filtered_log_users(timestamp: float):


    fecha_actual = datetime.datetime.now()
    timestamp_actual = datetime.datetime.timestamp(fecha_actual)* 1000
    timestamp_limite = timestamp_actual - (timestamp * 1000)

    filtro = {
        "timestamp": {
            "$gte": timestamp_limite,
            "$lt": timestamp_actual
        }
    }

    result = logUsersEntity(connection.PCM.LogUsuario.find(filtro))
    return result
```

Figura 5.3: Ejemplo de enrutador - LogUsuario

5.2.4 Scheduler

La biblioteca *Scheduler* de Python es la que hemos utilizado para implementar un tarea de actualización diaria y automática de los datos de los puntos favoritos y el almacenamiento de los datos actuales del punto en forma de histórico.

El *scheduler* lo hemos programado para que se ejecute todos los días a las 18:00 en la zona horaria de Europa y Ámsterdam.

```
def job():
    favourites = get_all_favourites()

    for favourite in favourites:
        print("**** Updating "+ favourite['_id'] +" ****")
        favouriteData = get_logs_in_favourite_radius(favourite['coordenadas']['latitud'],
                                                     favourite['coordenadas']['longitud'],
                                                     favourite['radio'])

        usuarios = favouriteData['usuarios']
        densidad = favouriteData['densidad']
        timestamp = datetime.datetime.timestamp(datetime.datetime.now()) * 1000

        checkpoint = {
            "idUsuario": favourite['idUsuario'],
            "idFavorito":  favourite['_id'],
            "timestamp": timestamp,
            "densidad": densidad,
            "usuarios": usuarios,
        }

        coordenadas = Coordenadas(latitud=favourite['coordenadas']['latitud'],
                                  ongitud=favourite['coordenadas']['longitud']
                                 )

        favorito = Favorito(idUsuario=favourite['idUsuario'],
                            nombre=favourite['nombre'],
                            coordenadas=coordenadas,
                            timestampCreacion=favourite['timestampCreacion'],
                            timestampUltimaMuestra=timestamp,densidad=densidad,
                            radio= favourite['radio']
                           )

        create_checkpoint(checkpoint)
        upsert_favourite(favourite['idUsuario'],
                         favourite['coordenadas']['latitud'],
                         favourite['coordenadas']['longitud'],
                         favourite['timestampCreacion'],favorito
                        )
        print("**** Location updated "+ favourite['_id'] +" ****")

    print("------ Daily update done ------")
```

Figura 5.4: Tarea encargada de la actualización diaria

Para poder llevar a cabo esta tarea sin interferir con la función de API REST que tiene el servidor, hemos tenido que combinar esta biblioteca con *Threading* para que un hilo paralelo al principal la ejecute.

5.3 Diseño cliente

En este apartado, explicaremos cómo se ha desarrollado la parte *frontend* o cliente, la organización de los diferente módulos, el diseño de las vistas y el funcionamiento de estas.

El cliente lo hemos desarrollado usando React Native y Expo, ambos *framework*s de la bibloiteca React que se utilizan principalmente para el desarrollo de aplicaciones móviles. Expo nos permite transformar nuestro código JavaScript para poder desplegarlo en Android, iOS o incluso en la web.

Hemos divido el *frontend* en los siguientes módulos:

- assets: carpeta contenedora de imágenes que hemos empleado para fondos, marcadores personalizados y el icono de la aplicación (véase Figura 5.5).

Figura 5.5: Logo de la aplicación

- components: componentes personalizados utilizados por toda la aplicación. La idea de estos elementos es proporcionar coherencia y consistencia visual y funcional. Esto nos permite evitar la repetición de código.
- config: archivos cuya finalidad es establecer los estilos de los componentes y constantes que se repiten por todo el código.
- screens: en esta carpeta almacenamos las diferentes vistas que conforman la aplicación.

- services: módulo de llamadas al servidor. Esta implementado de forma que sea código flexible, escalable y, sobretodo, reusable.
- tasks: define las tareas utilizadas para la creación o modificación del usuario y la tarea en segundo plano para pedir el consentimiento del usuario de utilizar el sensor de geolocalización y guardar los datos pertinentes como registros de ubicación.
- tools: en esta carpeta se encuentra los archivos encargados de hacer operaciones auxiliares para tratar datos y ajustarlos a los parámetros de los componentes que dibujan las gráficas.

Para tener una primera idea de cómo es el flujo de navegación de la aplicación, adjuntamos un diagrama en el que se ve de forma clara la sencillez de las transiciones entre las diferentes vistas y las acciones que las llevan a cabo tal y como podemos observar en la Figura 5.6.

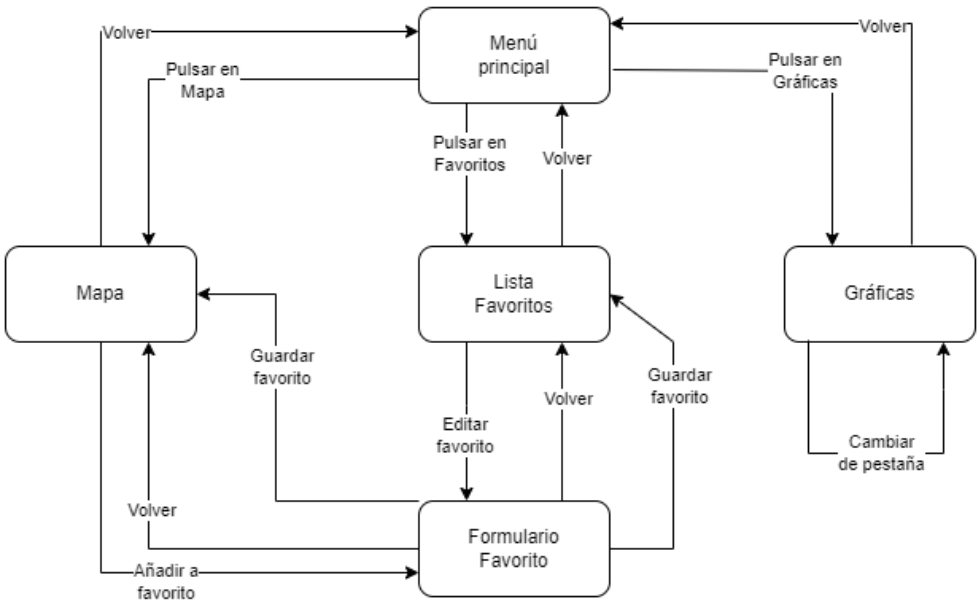

Figura 5.6: Diagrama de navegación

5.3.1 Vista Menú principal

En primer lugar, nos encontramos con el menú principal, una interfaz sencilla que tiene como finalidad separar las 3 funcionalidades principales de nuestra aplicación: el mapa de

calor, la interfaz de gestión de puntos favoritos del usuario y las gráficas para representar de forma más analítica los datos de estos puntos y sus registros históricos.

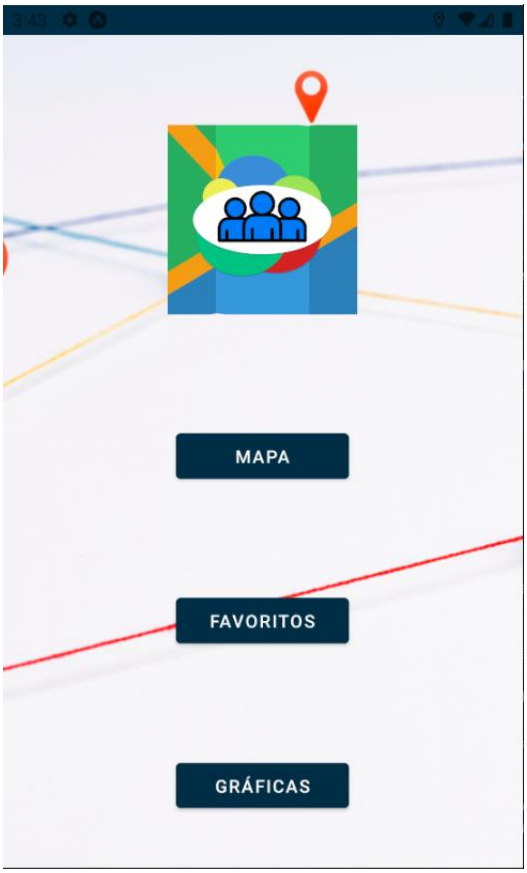

Figura 5.7: Vista Menú principal

5.3.2 Vista Mapa

Una vez pulsamos en el botón “Mapa” del menú, navegamos a la siguiente vista en la que podemos visualizar un mapa en la que se aplica una capa con las diferentes ubicaciones registradas de los usuarios representadas como puntos de calor. La intensidad de la coloración de las zonas de calor dependerá de la cantidad de puntos sobre el mapa y su acumulación.

Figura 5.8: Vista Mapa

Si pulsamos en el marcador del punto seleccionado por el usuario, aparecerá un botón “Opciones” encima del marcador. Si pulsamos en este, se abrirá un modal con información del punto y la opción de añadirlo a favoritos. Al pulsar en “Añadir a favoritos” navegaremos a la vista de un formulario la cual explicaremos más adelante.

Figura 5.9: Modal de ubicación seleccionada

En la parte inferior de la pantalla podemos encontrar 2 botones, uno para el filtrado de puntos de calor y otro para mostrar acciones sobre los marcadores.

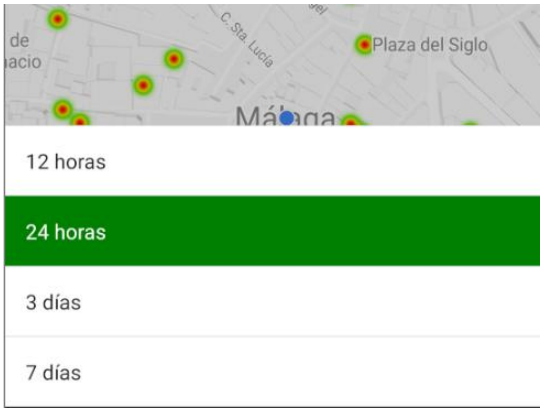

Figura 5.10: Filtro de puntos de calor por tiempo

Al pulsar el botón desplegado de “Mostrar favoritos” en el mapa aparecerán los marcadores personalizados favoritos del usuario como podemos visualizar en la Figura 5.8.

Figura 5.11: Acciones sobre marcadores

5.3.3 Vista Lista de puntos favoritos

La siguiente vista que vamos a describir es el listado de favoritos del usuario. En esta vista podemos ver un listado tarjetas con los datos de localizaciones favoritas. Si pulsamos en una de estas tarjetas, se desplegará información adicional del punto. Cada fila cuenta con 2 elementos adicionales: un checkbox para seleccionar uno o varios puntos para ser eliminados y un botón de editar que nos lleva a la vista de formulario de punto favorito.

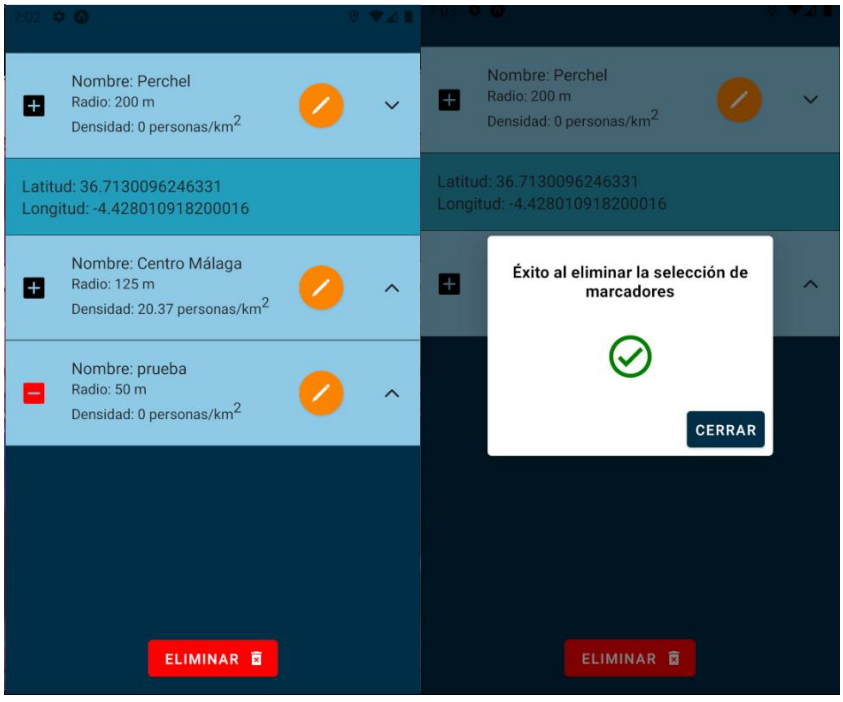

Figura 5.12: Vista Favoritos

Si decidimos eliminar alguna localización, al pulsar el botón, aparecerá un modal con información del estado de la acción. Si todo ha ido correctamente, el listado se actualizará de forma automática.

5.3.4 Vista Formulario creación/modificación punto favorito

A esta vista podremos llegar desde el modal de una ubicación seleccionada en el mapa o desde el listado de puntos favoritos. En este formulario tendremos que rellenar dos campos: nombre que queremos asignarle al punto, el cual debe ser una cadena alfanumérica no vacía, y el radio de medición, número que debe ser un entero positivo mayor que 0.

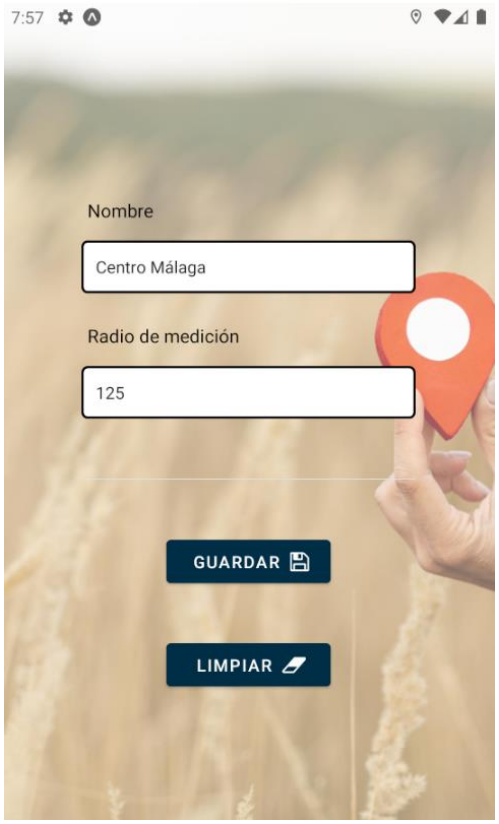

Figura 5.13: Vista Formulario favorito

Si alguno de los 2 campos está vacío o el radio de medición no cumple con la condición establecida, podremos visualizar mensajes de error justo debajo de los campos impidiendo la creación o modificación del punto.

5.3.5 Vista Gráficas

La última vista que explicaremos será la de gráficas. Esta vista cuenta con una barra de navegación con 3 pestañas: Tendencia, Densidades y Usuarios. Pulsar en ellas hará que nos movamos entre las distintas gráficas.

La gráfica de tendencias muestra una representación linear de como las densidades han ido cambiando en los últimos 7 días. Pulsando en los puntos distribuidos por el gráfico podremos cambiar el valor de la tarjeta que se encuentra justo encima como podemos ver en la Figura 5.14.

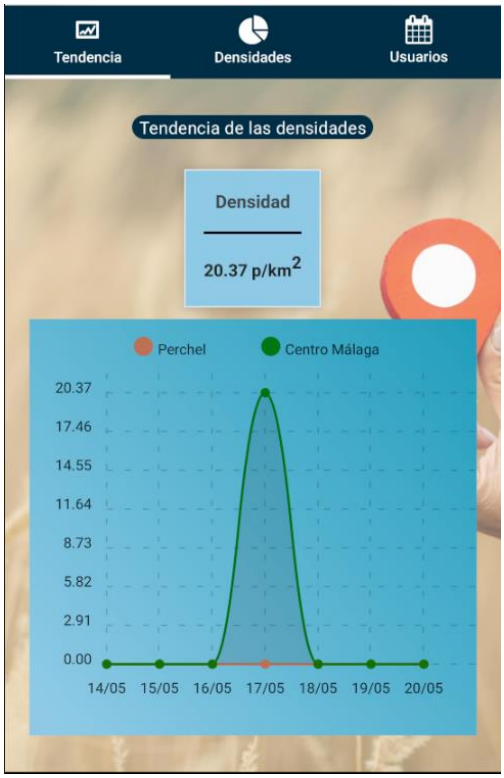

Figura 5.14: Gráfica de tendencia de las densidades

Si cambiamos a la gráfica de densidades, mostraremos un gráfico de pastel indicando las densidades actuales de los diferentes puntos favoritos del usuario.

Figura 5.15: Gráfica de densidades

La última representación que podemos encontrar en esta vista es la gráfica de contribución o mapa de calor de los usuarios encontrados dentro del radio de medición de los puntos favoritos a lo largo de los 98 días atrás desde la fecha actual siendo esta el cuadro encontrado en la esquina inferior derecha del gráfico situado en la Figura 5.16.

En esta pestaña también encontraremos un selector que tiene como opciones los diferentes puntos favoritos del usuario.

Figura 5.17: Gráfica de usuarios filtrado por punto favorito

Si pulsamos en uno de los cuadros del gráfico, podremos ver la fecha y la cantidad de usuarios en las tarjetas situadas justo encima de este.

5.3.6 Módulo de llamadas Axios al servidor

Para terminar este capítulo, explicaremos el módulo de llamadas al servidor mediante la biblioteca Axios.

En esta carpeta encontraremos un primer archivo que se encarga de estandarizar los métodos HTTP utilizando una instancia personalizada de Axios llamada “axiosClient”. Mediante esta instancia creamos una configuración específica para indicar la URL base y las cabeceras que van a tener las llamadas.

Las funciones exportadas (“getRequest”, “postRequest”, “putRequest” y “deleteRequest”) envuelven las solicitudes HTTP correspondientes utilizando la instancia “axiosClient”

previamente creada. Cada una de estas funciones toma una URL como parámetro, y en el caso de las funciones de solicitud “postRequest” y “putRequest”, también se proporciona una carga útil (*payload*) en forma de objeto.

Una vez definido este archivo, implementamos un servicio por cada una de los modelos de datos que se manejan en la aplicación. De esta forma, en cada uno de los servicios encontraremos las funciones para almacenar u obtener los datos necesarios para que la aplicación funcione correctamente.

6

Conclusiones y Líneas Futuras

Tras finalizar el proyecto, toca hacer un trabajo de retrospectiva y analizar todo el proceso para obtener conclusiones acerca de este, además de concretar líneas de trabajo futuras que se podrían realizar.

6.1 Conclusiones

En el presente proyecto, hemos logrado alcanzar los objetivos establecidos de manera satisfactoria. Durante el desarrollo del proyecto, hemos enfrentado diversos desafíos y obstáculos, especialmente debido a nuestro desconocimiento inicial de las tecnologías empleadas. Sin embargo, hemos logrado superar estas dificultades y utilizar los conocimientos adquiridos durante nuestro grado para obtener resultados exitosos.

En primer lugar, hemos logrado cumplir con los objetivos propuestos al diseñar y desarrollar una aplicación móvil funcional. Nuestro proyecto se ha enfocado en ofrecer una solución útil para abordar una necesidad específica en el campo de la gestión urbana. A través de un proceso de análisis, diseño y programación, hemos conseguido implementar las características y funcionalidades deseadas, brindando una experiencia óptima para los usuarios.

Es importante destacar que, al iniciar este proyecto, nos enfrentamos a un desconocimiento inicial de las tecnologías empleadas. Sin embargo, hemos demostrado una gran capacidad de aprendizaje y adaptación, aprovechando los recursos disponibles y adquiriendo nuevos conocimientos a lo largo del desarrollo del proyecto.

Nuestro grado académico nos ha brindado los fundamentos necesarios para abordar este proyecto de ingeniería. Durante nuestra formación, hemos adquirido conocimientos sólidos en áreas como la gestión de proyectos, modelado de datos y diseño de aplicaciones e interfaces. Además, hemos aplicado metodologías y técnicas aprendidas en el grado, como Scrum o la toma de requisitos de un sistema, para gestionar y ejecutar el proyecto de manera eficiente.

6.2 Líneas futuras

Actualmente, el uso de sistemas inteligentes o aplicar conocimientos de *Machine Learning* en aplicaciones se está volviendo una práctica muy popular.

Una propuesta de desarrollo para ampliar las funcionalidades de esta aplicación sería la implementación de modelos de regresión o redes neuronales para la predicción de ubicaciones futuras de los usuarios. No obstante, para poder aplicar estas tecnologías necesitaríamos datos más avanzados o detallados como pueden ser la velocidad y la dirección a la que transita el usuario. Añadiendo estos datos a los registros de ubicación podríamos identificar patrones de movimiento o comportamientos del usuario.

Referencias

[1] Instituto Nacional de Estadística. “Encuesta sobre Equipamiento y Uso de Tecnologías de Información y Comunicación en los Hogares”. Año 2020. URL: https://www.ine.es/prensa/tich_2020.pdf **(Accedido el 8/5/2023)**

[2] Python Software Foundation. *El tutorial de python*. Python documentation. URL: https://docs.python.org/es/3/tutorial/ **(Accedido el 12/5/2023)**

[3] FASTAPI. URL: https://fastapi.tiangolo.com/ **(Accedido el 12/5/2023)**

[4] MongoDB. *MongoDB: The Developer Data Platform*. URL: https://www.mongodb.com/ **(Accedido el 12/5/2023)**

[5] *JavaScript | MDN*. (2023, 9 febrero). URL: https://developer.mozilla.org/es/docs/Web/JavaScript **(Accedido el 12/5/2023)**

[6] *Setting up the development environment · React Native*. (2023, 17 marzo). URL: https://reactnative.dev/docs/environment-setup **(Accedido el 12/5/2023)**

[7] *Expo Documentation*. Expo Documentation. URL: https://docs.expo.dev/ **(Accedido el 12/5/2023)**

[8] *PyMongo 4.3.3 Documentation — PyMongo 4.3.3 documentation*. URL: https://pymongo.readthedocs.io/en/stable/ **(Accedido el 13/5/2023)**

[9] *schedule — schedule 1.2.0 documentation*. URL: https://schedule.readthedocs.io/en/stable/ **(Accedido el 13/5/2023)**

[10] *Empezando*. Empezando | Axios Docs. URL: https://axios-http.com/es/docs/intro **(Accedido el 13/5/2023)**

[11] *Usage*. Async Storage. URL: https://react-native-async-storage.github.io/async-storage/docs/usage/ **(Accedido el 13/5/2023)**

[12] *Hello from React Native Material | React Native Material*. URL: https://www.react-native-material.com/ **(Accedido el 13/5/2023)**

[13] *React Native Elements*. URL: https://reactnativeelements.com/ **(Accedido el 13/5/2023)**

[14]*React Navigation | React Navigation*. URL: https://reactnavigation.org/ **(Accedido el 14/5/2023)**

[15]*Home*. URL: https://react-hook-form.com/ **(Accedido el 14/5/2023)**

[16]*Location*. Expo Documentation. URL: https://docs.expo.dev/versions/latest/sdk/location/ **(Accedido el 14/5/2023)**

[17]React-Native-Maps. *GitHub - react-native-maps/react-native-maps: React Native Mapview component for iOS + Android*. GitHub. URL: https://github.com/react-native-maps/react-native-maps **(Accedido el 14/5/2023)**

[18]Indiespirit. *GitHub - indiespirit/react-native-chart-kit: React Native Chart Kit: Line Chart, Bezier Line Chart, Progress Ring, Bar chart, Pie chart, Contribution graph (heatmap)*. GitHub. URL: https://github.com/indiespirit/react-native-chart-kit **(Accedido el 14/5/2023)**

[19]*Visual Studio Code - Code Editing. Redefined*. (2021, 3 noviembre). URL: https://code.visualstudio.com/ **(Accedido el 14/5/2023)**

[20]*Thunder Client - Rest API Client Extension for VS Code*. URL: https://www.thunderclient.com/ **(Accedido el 14/5/2023)**

[21]*Figma: the collaborative interface design tool.* Figma. URL: https://www.figma.com/ **(Accedido el 14/5/2023)**

[22]*Hello World - GitHub Docs*. GitHub Docs. URL: https://docs.github.com/en/get-started/quickstart/hello-world **(Accedido el 14/5/2023)**

[23]*Qué es Trello: descubre sus funciones, usos y todo lo que ofrece | Trello*. URL: https://trello.com/es/tour **(Accedido el 14/5/2023)**

[24]Admin, & Admin. (2021). A veces llamado Winword, MS Word, o Word, Microsoft Word es un Read more. *Administra Proyectos*. URL: https://administraproyectos.info/caracteristicas-de-microsoft-word/ **(Accedido el 14/5/2023)**

[25]Testeando Software. (2014). Mockaroo. Generador de datos para pruebas. *Testeando Software*. URL: https://testeandosoftware.com/mockaroo-generador-datos-prueba/ **(Accedido el 14/5/2023)**

[26]*¿Qué es la metodología ágil?.* URL: https://www.redhat.com/es/devops/what-is-agile-methodology **(Accedido el 15/5/2023)**

[27]Atlassian. *Scrum: qué es, cómo funciona y por qué es excelente*. URL: https://www.atlassian.com/es/agile/scrum **(Accedido el 15/5/2023)**

Apéndice A

Manual de usuario

En este manual explicaremos el funcionamiento de la aplicación de forma que cualquier persona, independientemente del perfil que tenga, sea capaz de entender y utilizarla sin ningún tipo de problema.

Menú principal

Lo primero que nos encontraremos será un menú sencillo con tres botones. Estos botones nos llevaran a las siguientes vistas: mapa, listado de favoritos y gráficas.

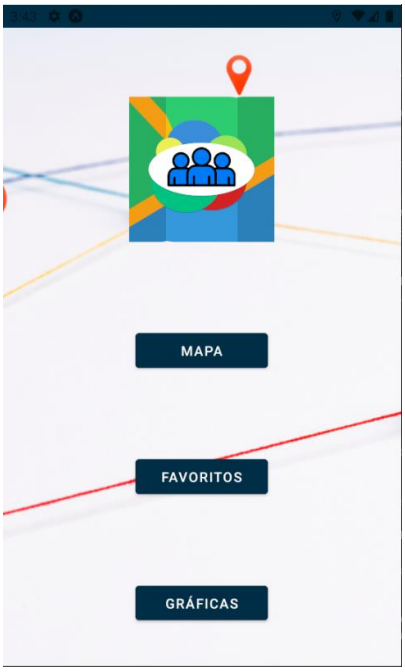

Figura A.1: Vista Menú principal

Mapa

Una vez se ha cargado el mapa, lo primero que visualizamos son 3 elementos: en la parte inferior de la pantalla 2 botones, de los cuales uno es un filtro por tiempo de los puntos de

calor que aparecen en el mapa y un botón que despliega un listado de acciones sobre los marcadores dibujados en el mapa; y los puntos de calor.

Figura A.2: Mapa y marcadores

Pulsar en el botón “Filtro” desplegará una lista de opciones con diferentes cantidades de tiempo para filtrar la cantidad de puntos de calor que podemos visualizar en el mapa.

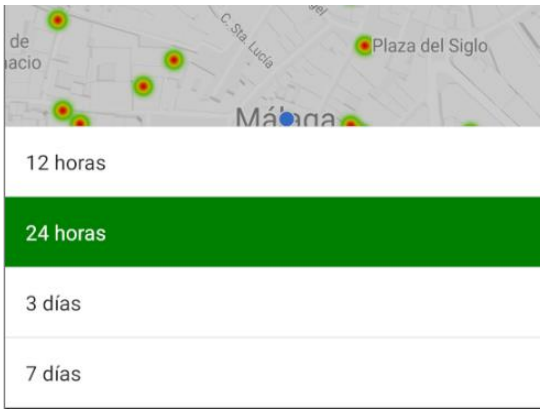

Figura A.3: Filtro de puntos de calor por tiempo

Pulsar en el botón “Acciones” desplegará una lista de botones. Por un lado está el botón “Mostrar favoritos” que mostrará en el mapa unos marcadores personalizados para las

ubicaciones guardadas por el usuario. Por otro lado, tenemos el botón “Limpiar marcadores” que eliminará todo tipo de marcador que haya en el mapa.

Figura A.4: Acciones sobre marcadores

Seleccionando una ubicación del mapa, aparecerá un marcador. Si pulsamos en el marcador, aparecerá un botón justo encima con el texto “Opciones”. Pulsar en este elemento, mostrará una ventana con información sobre la ubicación seleccionada y un botón “Añadir a favoritos”.

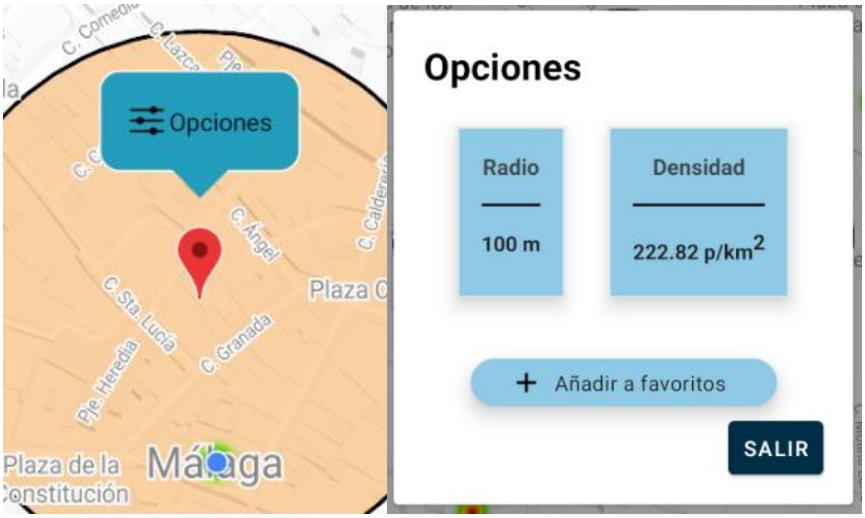

Figura A.5: Botón y pestaña “Opciones”

Seleccionando una ubicación del mapa, aparecerá un marcador. Si pulsamos en el marcador, aparecerá un botón justo encima con el texto “Opciones”. Pulsar en este elemento, mostrará una ventana con información sobre la ubicación seleccionada y un botón “Añadir a favoritos”.

Figura A.6: Botón y pestaña “Opciones”

De forma similar a la anterior, siguiendo los mismo pasos pero con los marcadores favoritos, visualizaremos una pestaña con unos datos adicionales y la posibilidad de editar el punto creado.

Pulsando en “Añadir a favoritos” o “Editar marcador” accederemos a una nueva vista, el formulario de favoritos. En este aparecerán dos campos: nombre y radio de medición. Estos no pueden ser vacíos y además el radio de medición debe ser un número entero positivo.

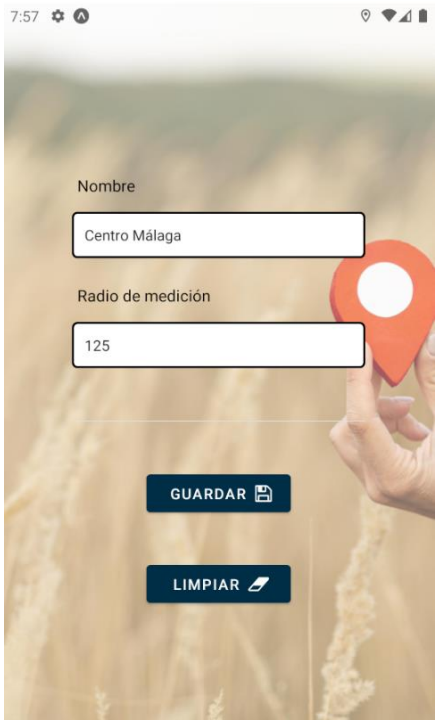

Figura A.7: Vista Formulario favorito

Listado de favoritos

Volviendo al menú principal, tenemos el botón “Favoritos” el cual nos llevará a un listado de puntos creados por el usuario. En esta vista el usuario podrá visualizar los datos de los puntos, modificarlos o eliminarlos.

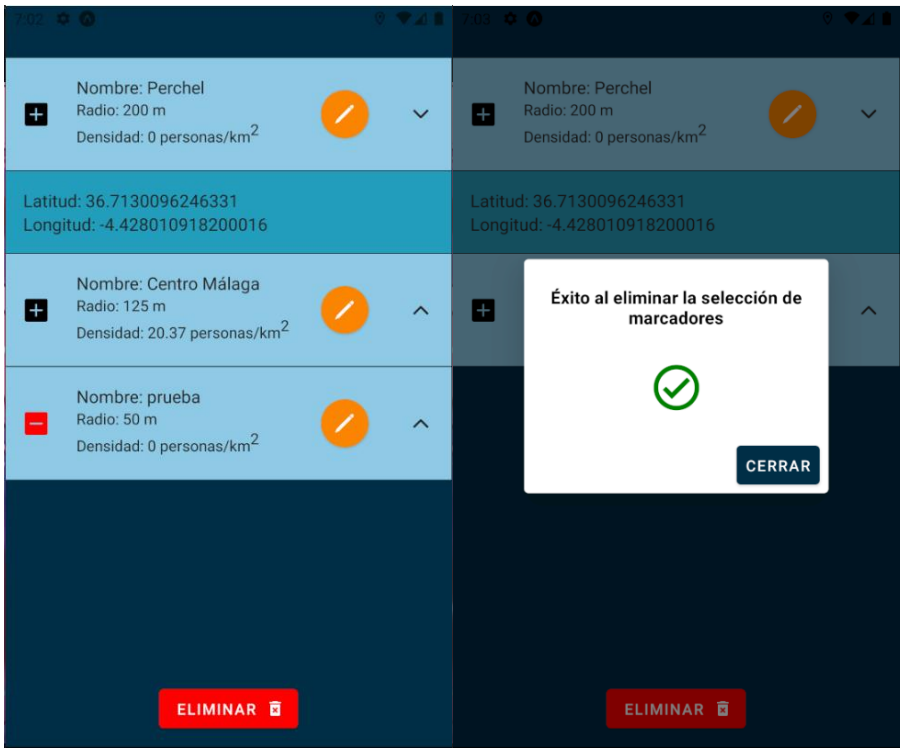

Figura A.8: Lista de favoritos y mensaje de éxito al eliminar marcadores

Si pulsamos en las filas, desplegaremos la información adicional del punto. Pulsando en el botón del lápiz podremos editar el punto desde el formulario de marcadores. Además, cada fila cuenta con un cuadrado con un signo "+" en el lado izquierdo. Pulsando en él, indicaremos que hemos seleccionado el punto para el eliminarlo y aparecerá un signo "-" en rojo. Esto permite hacer una eliminación de uno o varios puntos al mismo tiempo. Al pulsar el botón de "Eliminar" aparecerá un mensaje indicando si se ha ejecutado la operación con éxito.

Gráficas

Por último, si pulsamos en el botón "Gráficas" del menú principal, navegaremos a la vista en la cual se mostrarán los datos de los puntos favoritos de forma más visual.

En la parte superior encontramos un navegador con 3 pestañas: Tendencia, Densidades y Usuarios. Pulsar en ellas hará que nos movamos entre las distintas gráficas.

La gráfica de tendencias muestra una representación linear de como las densidades han ido cambiando en los últimos 7 días. Pulsando en los puntos distribuidos por el gráfico podremos cambiar el valor de la tarjeta que se encuentra justo encima como podemos ver en la Figura A.9.

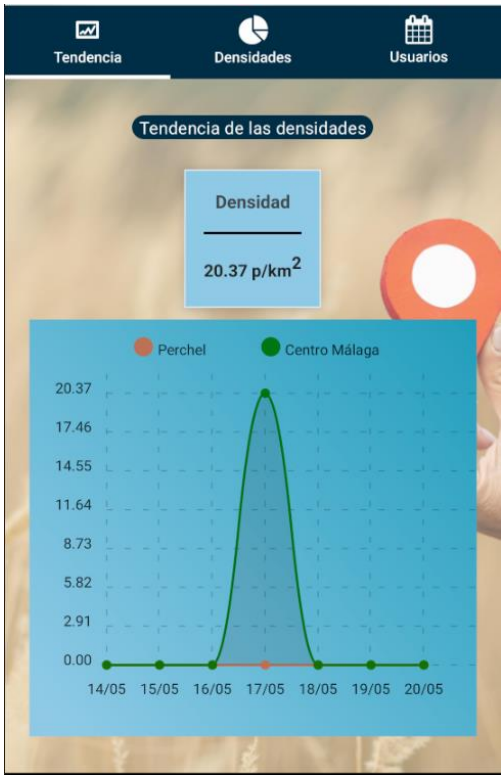

Figura A.9: Gráfica de tendencia de las densidades

Si cambiamos a la gráfica de densidades, mostraremos un gráfico de pastel indicando las densidades actuales de los diferentes puntos favoritos del usuario.

Figura A.10: Gráfica de densidades

La última representación que podemos encontrar en esta vista es la gráfica de contribución o mapa de calor de los usuarios encontrados dentro del radio de medición de los puntos favoritos a lo largo de los 98 días atrás desde la fecha actual siendo esta el cuadro encontrado en la esquina inferior derecha del gráfico situado en la Figura A.11.

En esta pestaña también encontraremos un selector que tiene como opciones los diferentes puntos favoritos del usuario.

Figura A.11: Gráfica de usuarios filtrado por punto favorito

Si pulsamos en uno de los cuadros del gráfico, podremos ver la fecha y la cantidad de usuarios en las tarjetas situadas justo encima de este.

Apéndice B

Manual de despliegue del servidor

En este apéndice explicaremos como hacer funcionar el servidor de forma local en nuestro equipo.

Como requisitos previos, necesitaremos un entorno de desarrollo en el que podamos abrir el proyecto y que tenga acceso a la terminal del sistema además de tener instalado Python 3.10.4.

Lo primero que debemos hacer es ejecutar un entorno virtual de Windows en el que instalar las dependencias del proyecto. Para ello ejecutaremos el siguiente comando:

```
& <path equipo>/pcm-fastapi/venv/Scripts/Activate.ps1
```

Siendo <path equipo> la ruta donde se encuentra la carpeta del proyecto.

Seguido de esto, para instalar las dependencias, ejecute el comando:

```
pip install -r requirements.txt
```

Y por último, para ejecutar definitivamente el servidor y que empiece a funcionar correctamente, introducir el siguiente comando:

```
uvicorn app:app
```

Si todo ha ido correctamente, tendría que salirnos por consola unas líneas parecidas a las de la Figura B.1.

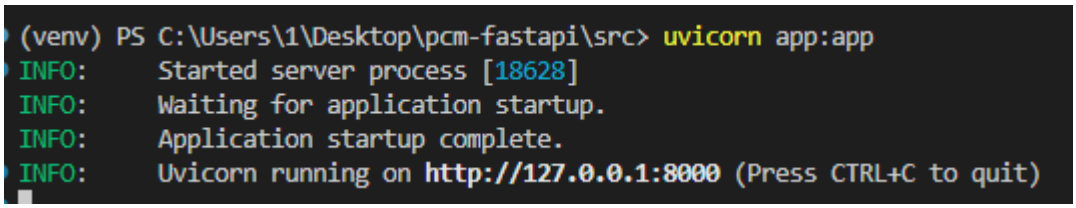

Figura B.1: Salida por consola tras ejecutar comando de inicio del servidor

Apéndice C

Manual de despliegue del cliente

En este apéndice explicaremos como hacer funcionar el cliente de forma local en nuestro dispositivo móvil.

Como requisitos previos necesitaremos instalar la aplicación de móvil Expo Go y en el equipo personal la versión 46 del SDK de Expo.

Desde el directorio del proyecto ejecutaremos el comando:

```
npm install
```

Esto instalará las dependencias incluidas dentro del archivo package.json.

Una vez instaladas las dependencias, debemos asegurarnos que el equipo y el dispositivo móvil están utilizando la misma red WiFi. Ejecutaremos el comando:

```
expo start
```

Aparecerá por pantalla un código QR que debe ser leído mediante la aplicación móvil de Android Expo Go.

Figura C.1: Salida por consola tras ejecutar expo start

UNIVERSIDAD DE MÁLAGA | uma.es

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

E.T.S. DE INGENIERÍA INFORMÁTICA