UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Desarrollo de una aplicación Android de localización interior y cálculo de rutas.

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Xu, Qiyue

Tutor/a: Albert Albiol, Manuela

CURSO ACADÉMICO: 2023/2024

Resumen

Las herramientas de posicionamiento y localización exterior han progresado considerablemente los últimos años, desde la aparición en los años 70 del sistema de posicionamiento global (GPS). Sin embargo, sistemas y aplicaciones para el posicionamiento y ubicación interior sigue siendo un desafío pendiente. En lugares como hospitales, centros comerciales, estaciones y aeropuertos, es común que las personas se extravíen fácilmente dentro de los edificios. Por ello, la navegación interior se vuelve esencial para mejorar la experiencia del usuario y facilitar la búsqueda de destinos dentro de estos espacios cerrados.

Este proyecto se enfoca en el desarrollo de una aplicación para el posicionamiento y búsqueda de rutas en espacios interiores. El objetivo principal de la aplicación es optimizar la experiencia del usuario y simplificar la orientación dentro de estos complejos entornos, con el fin de reducir el tiempo de búsqueda y mejorar la navegación interna. La aplicación desarrollada es una aplicación Android que emplea un mapa junto con una capa de plano GeoJson. La implementación de Beacons, combinada con el cálculo de la posición utilizando señales Received Signal Strength Indicator (RSSI), ayuda a la mejora significativamente de la precisión de la localización del usuario en interiores.

La aplicación se desarrolla en Kotlin utilizando Android Studio y se ha implementado con la API de Google Maps. Se ha adoptado una estructura de arquitectura Model View Intent (MVI) para gestionar el flujo de datos y los estados de la aplicación, mientras que Jetpack Compose se ha utilizado para diseñar una interfaz de usuario moderna y reactiva.

Para calcular rutas dentro del edificio, se ha empleado el eficiente algoritmo de Dijkstra, aplicándolo a un grafo que representa las posibles rutas interiores. Además, para lograr una localización precisa del usuario en estos espacios, se ha utilizado el método de promedio ponderados, que combina datos de múltiples Beacons para mejorar la exactitud de la posición.

Palabras clave: localización y navegación interior, Model View Intent, Beacons, RSSI, GeoJson, algoritmo de Dijkstra, el método de promedio ponderados

Abstract

Outdoor positioning and location tools have progressed considerably in recent years, since the emergence of the Global Positioning System (GPS) in the 1970s. However, systems and applications for indoor positioning and location remain a challenge. In places such as hospitals, shopping centres, stations and airports, it is common for people to easily get lost inside buildings. Therefore, indoor navigation becomes essential to improve the user experience and make it easier to find destinations within these enclosed spaces.

This project focuses on the development of an application for positioning and wayfinding in indoor spaces. The main objective of the application is to optimise the user experience and simplify orientation within these complex environments, in order to reduce search time and improve internal navigation. The developed application is an Android application that uses a map together with a GeoJson map layer. The implementation of Beacons, combined with position calculation using Received Signal Strength Indicator (RSSI) signals, helps to significantly improve the accuracy of the user's indoor location.

The application is developed in Kotlin using Android Studio and implemented with the Google Maps API. A Model View Intent (MVI) architecture framework has been adopted to manage the data flow and states of the application, while Jetpack Compose has been used to design a modern and responsive user interface.

To calculate routes inside the building, the efficient Dijkstra algorithm has been used, applying it to a graph representing the possible interior routes. In addition, to achieve an accurate location of the user in these spaces, the weighted averaging method has been used, which combines data from multiple Beacons to improve positional accuracy.

Keywords: indoor localization and navigation, Model View Intent, Beacons, RSSI, GeoJson, Dijkstra's algorithm, weighted average method

Resum

Les ferramentes de posicionament i localització exterior han progressat considerablement els últims anys, des de l'aparició en els anys 70 del sistema de posicionament global (GPS). No obstant això, sistemes i aplicacions per al posicionament i ubicació interior continua sent un desafiament pendent. En llocs com a hospitals, centres comercials, estacions i aeroports, és comú que les persones s'extravien fàcilment dins dels edificis. Per això, la navegació interior es torna essencial per a millorar l'experiència de l'usuari i facilitar la cerca de destins dins d'estos espais tancats.

Este projecte s'enfoca en el desenvolupament d'una aplicació per al posicionament i cerca de rutes en espais interiors. L'objectiu principal de l'aplicació és optimitzar l'experiència de l'usuari i simplificar l'orientació dins d'estos complexos entorns, amb la finalitat de reduir el temps de cerca i millorar la navegació interna. L'aplicació desenvolupada és una aplicació Android que empra un mapa juntament amb una capa de pla GeoJson. La implementació de Beacons, combinada amb el càlcul de la posició utilitzant senyals Received Signal Strength Indicator (RSSI), ajuda a la millora significativament de la precisió de la localització de l'usuari en interiors.

L'aplicació es desenvolupa en Kotlin utilitzant Android Studio i s'ha implementat amb la API de Google Maps. S'ha adoptat una estructura d'arquitectura Model View Intent (MVI) per a gestionar el flux de dades i els estats de l'aplicació, mentres que Jetpack Compose s'ha utilitzat per a dissenyar una interfície d'usuari moderna i reactiva.

Per a calcular rutes dins de l'edifici, s'ha emprat l'eficient algorisme de Dijkstra, aplicant-lo a un graf que representa les possibles rutes interiors. A més, per a aconseguir una localització precisa de l'usuari en estos espais, s'ha utilitzat el mètode de mitjana ponderats, que combina dades de múltiples *Beacons per a millorar l'exactitud de la posició.

Paraules clau: localització i navegació interior, Model View Intent, Beacons, RSSI, GeoJson, algorisme de Dijkstra, el mètode de mitjana ponderats

Tabla de contenido

1. Introducción

En este capítulo de la memoria, presentaremos el contexto en el que se ha desarrollado este Trabajo Fin de Grado (TFG), la motivación detrás de su realización, los objetivos que se pretenden alcanzar, el impacto esperado y la estructura de la memoria.

1.1 Contexto

El TFG que se presenta en esta memoria se desarrolla en colaboración con la empresa Inditex. Inditex es una empresa multinacional española de moda y distribución. Es una de las mayores compañías de moda. Bajo el nombre de Inditex están muchas marcas conocidas como Zara, Massimo Dutti, Pull&Bear, Bershka, etc.

Inditex fue fundada en 1985 por Amancio Ortega Gaona en Arteijo, La Coruña, España. Desde su fundación, ha crecido para convertirse en una de las mayores compañías del mundo, con presencia en más de 90 países y miles de tiendas bajo sus distintas marcas.

Su cede central se encuentra en Arteijo, La Coruña, y es conocida como el “Campus de Inditex”. Este campus es un lugar moderno y funcional, diseñado para fomentar la colaboración y la innovación entre los empleados. El campus cuenta con diversas instalaciones que incluyen oficinas administrativas, salas de reuniones, áreas de diseño, laboratorios de investigación y desarrollo e instalaciones de logística y distribución. Además, existen espacios de recreación y bienestar para los empleados, como comedores, áreas verdes y zonas deportivas.

A pesar de ser una empresa textil, Inditex cuenta con un departamento de tecnología compuesto por diferentes sub-departamentos especializados. Este departamento de tecnología propone convenios con distintas escuelas tecnológicas para desarrollar proyectos en el contexto de prácticas en empresa. El presente TFG se enmarca en una de estas prácticas de empresa realizadas por la estudiante en Inditex. Estas prácticas se desarrollan dentro del departamento Producto que a su vez está dentro de la Mobile Unit - Android.

1.2 Motivación

En muchos lugares, especialmente en espacios físicos grandes, la capacidad de moverse eficientemente por el entorno es esencial. Por ejemplo, si un usuario se encuentra en una tienda de Inditex con tres plantas y un área extensa, puede resultar complicado encontrar un artículo de interés. Los métodos tradicionales de señalización y mapas estáticos no son suficientes para resolver estos desafíos de manera efectiva.

Ante esta situación, el departamento de tecnología de Inditex ha propuesto el desarrollo de una aplicación para la localización de usuarios en espacios interiores y cerrados, tales como tiendas, centros de distribución y oficinas corporativas de Inditex. La aplicación proporcionará un servicio de navegación hacia el destino deseado, mejorando la experiencia del usuario y la satisfacción del cliente.

La motivación principal de este proyecto es abordar los problemas actuales relacionados con la orientación y navegación en interiores. En muchos entornos, especialmente aquellos con rutas complicadas o áreas extensas, los usuarios a menudo enfrentan dificultades para encontrar su camino, lo que puede resultar en frustración y pérdida de tiempo.

1.3 Objetivo

El objetivo principal de este proyecto es desarrollar una herramienta tecnológica avanzada que ofrezca a los usuarios orientación precisa y en tiempo real sobre posiciones y rutas dentro de edificios. La herramienta será especialmente diseñada para el ámbito comercial de Inditex.

El proyecto consistirá en el desarrollo de una aplicación que integrarán un mapa interactivo que proporciona una visualización detallada y accesible del entorno interior. Utilizaremos balizas Bluetooth para mejorar el cálculo de la posición del usuario dentro del edificio.

La interfaz de usuario de la aplicación facilitará la orientación y el desplazamiento en el mapa, mejorando significativamente la experiencia de navegación en interiores. Además, se implementarán funciones para mostrar puntos de interés relevantes y trazar rutas personalizadas en el mapa, guiando a los usuarios de manera eficiente hacia sus destinos.

Para mejorar la utilidad de la herramienta, se proporcionarán indicaciones dinámicas y ajustes en ruta en tiempo real según el movimiento del usuario, mejorando la precisión y la efectividad de la navegación interior.

Este proyecto busca transformar la experiencia de los usuarios dentro de los edificios comerciales de Inditex, ofreciendo una navegación intuitiva y precisa que optimice la movilidad y la interacción con el entorno físico.

1.4 Impacto Esperado

El proyecto tiene como objetivo principal mejorar la experiencia del cliente al facilitar la búsqueda de productos dentro de las tiendas de Inditex. Se espera que la implementación de esta aplicación de navegación interior no solo reduzca el tiempo de búsqueda para los clientes, sino que también incremente la satisfacción del cliente y, potencialmente, las ventas. A largo plazo, esta tecnología podría ser escalada y

adaptada a otros entornos dentro de la red de Inditex, mejorando así la eficiencia operativa y proporcionando un valor añadido significativo a la empresa.

Aunque este proyecto se ha desarrollado junto con Inditex, su utilidad no está limitada a la utilización comercial. Las tecnologías y metodologías implementadas pueden aplicarse en una amplia gama de contextos, tales como hospitales, centros comerciales, estaciones de transporte y aeropuertos. En estos entornos, la mejora de la navegación interior puede facilitar la localización de servicios y reducir significativamente el tiempo de búsqueda, mejorando la experiencia del usuario en general. Esto demuestra el potencial de las soluciones desarrolladas para ser adaptadas a diversos escenarios, contribuyendo a la eficiencia y la satisfacción del usuario en múltiples ámbitos.

1.5 Estructura de la memoria

Esta memoria está estructurada en varios capítulos y se dividen de la siguiente manera:

- **Introducción**: Este capítulo proporciona una descripción del contexto en el cual se desarrolla el proyecto, resaltando la motivación inicial, los objetivos principales y el impacto esperado.
- **Estado del arte**: Se analiza el estado actual de la tecnología relacionada con el proyecto. Se incluyen estudios de tres aplicaciones similares (Google Maps Indoor, TomTom AmiGo, e IndoorAtlas), ofreciendo críticas sobre cada una de ellas. También se introduce la solución propuesta.
- **Metodología:** Este capítulo detalla las metodologías de trabajo aplicadas durante el desarrollo del proyecto. Se introduce cómo se han aplicado algunos principios de las metodologías ágiles y cómo se ha aplicado Git-Flow para gestión de proyectos, explicando sus beneficios y cómo contribuyen a una gestión eficaz del proyecto.
- **Análisis del problema:** Se presenta un análisis del problema abordado, incluyendo los requisitos funcionales y no funcionales del proyecto. Además, se describen los casos de uso que se derivan de estos requisitos.
- **Tecnología utilizada.** En este capítulo se examinan las distintas tecnologías de posicionamiento interior disponibles. Se realiza un estudio comparativo para seleccionar la tecnología más adecuada para el proyecto, justificando la elección basada en criterios técnicos y operativos. Este análisis asegura que se utiliza la tecnología más eficiente y viable para alcanzar los objetivos del proyecto.
- **Diseño de la solución:** Se aborda el diseño de la solución desde diferentes perspectivas, incluyendo la arquitectura del sistema. Se realiza un análisis del algoritmo de cálculo de posicionamiento y de técnicas para suavizar las fluctuaciones de las señales. También se describe el diseño del algoritmo de cálculo de rutas por el algoritmo de Dijkstra.
- **Implementación de la solución:** Este capítulo muestra la implementación del proyecto, incluyendo fragmentos de código representativos de las distintas

partes de la aplicación. Se explican las decisiones técnicas y se presentan ejemplos concretos que ilustran el funcionamiento del sistema.

- **Pruebas:** En este capítulo se explican las pruebas realizadas que permiten asegurar la calidad del producto desarrollado.
- **Conclusiones y trabajo futuro:** Se resume el proyecto, evaluando los objetivos alcanzados y reflexionando sobre los resultados obtenidos. Además, se proponen posibles mejoras y trabajos futuros.
- **Biografía:** Registro de las fuentes consultadas para redacción.
- **Objetivos de desarrollo sostenible:** Se realiza una reflexión sobre la relación del proyecto con los objetivos de desarrollo sostenible (ODS).

2. Estado del arte

Este capítulo profundiza en las herramientas actuales en el ámbito del software específicamente diseñadas para abordar los desafíos de la localización y el posicionamiento en entornos interiores. El capítulo finaliza con un análisis de las aplicaciones que han sido estudiadas.

2.1 Google Maps Indoor

Google Maps Indoor ofrece una amplia gama de opciones de navegación en distintos modos. La aplicación está integrada en Google Maps, ampliamente conocida y utilizada por su extensa cobertura global y su interfaz fácil de usar. Esto evita la necesidad de descargar una aplicación adicional. Por tanto, Google Indoor está disponible en múltiples plataformas, incluyendo Android, iOS y web, se destaca por su alta precisión, especialmente en entornos exteriores.

Google Maps Indoor utiliza una combinación de Wi-Fi, sensores del dispositivo, puntos de acceso y análisis de patrones de movimiento para determinar la ubicación en interior. Generalmente puede alcanzar una precisión de 5 a 10 metros. Esto depende en gran medida de la densidad de puntos de acceso Wi-Fi y la configuración del edificio.

Google Maps Indoor está limitada a aquellos edificios cuyos propietarios han proporcionado los planos a Google. Por lo tanto, la disponibilidad de mapas interiores depende de la colaboración con propietarios de edificios. Actualmente cubre principalmente grandes aeropuertos, centros comerciales y algunos museos.

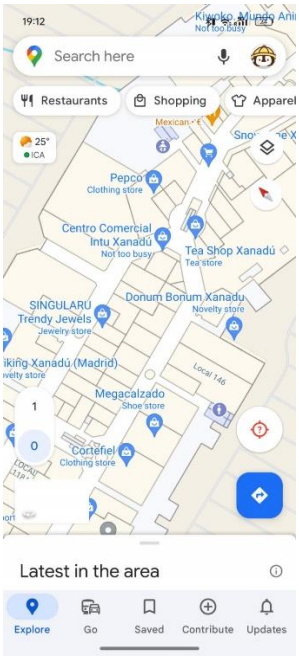

Figura 2. Google Maps Indoor Interfaz de Usuario

2.2 TomTom AmiGo

TomTom Amigo es una aplicación diseñada para la navegación exterior, enfocada principalmente en la navegación automovilística. Ofrece una interfaz altamente personalizable que permite a los usuarios cambiar el icono y el color, y un diseño de botones de servicio que facilita el acceso rápido a puntos de interés cercanos. Esto mejora significativamente la experiencia del usuario al proporcionar herramientas intuitivas para la búsqueda y navegación eficiente.

La precisión de la navegación con TomTom AmiGo está influenciada por diversos factores, como la claridad de las señales GPS y la calidad de los mapas utilizados. Además, ofrece cobertura en numerosas regiones y países alrededor del mundo. Sin embargo, la disponibilidad de características específicas, como la información del tráfico en tiempo real y los puntos de interés, puede variar según la ubicación geográfica.

TomTom AmiGo cuenta con una interfaz intuitiva que ofrece una alta flexibilidad de personalización. Sus funciones son muy amigables y fáciles de usar. Por ejemplo, al tocar el segundo botón con el icono de cubiertos, la aplicación busca automáticamente los restaurantes cercanos. La Figura 3 muestra una captura de pantalla de la aplicación.

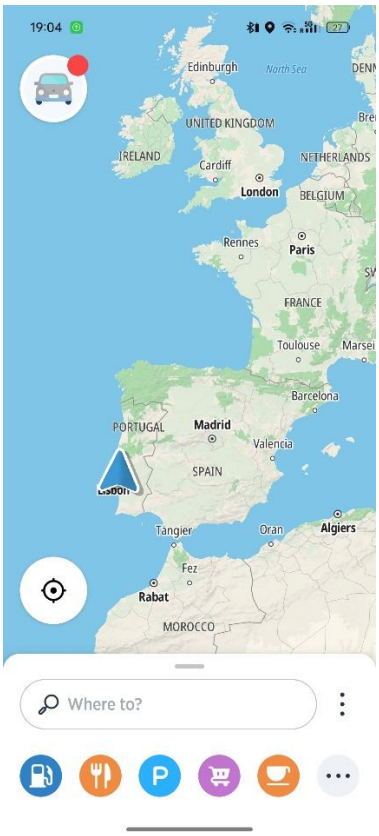

Figura 3. TomTom AmiGo Interfaz de Usuario

2.3 IndoorAtlas

IndoorAtlas es una aplicación de navegación móvil desarrollada por la empresa IndoorAtlas Ltd. Su enfoque principal es ofrecer localización interior precisa utilizando tecnología avanzada. La aplicación es una solución avanzada para la navegación interior que ofrece mapas 2D mejorando la precisión y la utilidad en entornos cerrados.

IndoorAtlas utiliza una tecnología basada en el mapeo de campos magnéticos terrestres y sensores de teléfonos inteligentes para proporcionar localización interior precisa. Esta tecnología aprovecha las variaciones naturales en el campo magnético de la Tierra, así como los sensores magnéticos presentes en los dispositivos móviles, para determinar la ubicación del usuario dentro de edificios con una alta precisión. En condiciones ideales, IndoorAtlas puede ofrecer una precisión de ubicación de hasta varios metros, lo que permite una navegación interior efectiva. La Figura 4 muestra una captura de pantalla de la aplicación IndoorAtlas.

Figura 4. IndoorAtlas Interfaz de Usuario

2.4 Crítica al estado del arte

A continuación, se identifican algunos puntos débiles de las herramientas estudiadas:

- Aunque Google Maps ofrece una amplia cobertura de mapas exteriores, su funcionalidad de navegación en entornos interiores, Google Maps Indoor, es limitada. Esto se debe a que el cálculo de rutas de Google Maps se implementa a través de la API de Google Maps Directions, la cual solo proporciona rutas para exteriores.

- TomTom AmiGo no es válida para la localización en espacios interiores de Inditex, ya que no ofrece un servicio de posicionamiento ni navegación en interiores.
- Aunque IndoorAtlas es una aplicación especialmente dedicada en la localización interior. Sin embargo, no es una solución válida para la localización en espacios interiores de Inditex porque requiere una infraestructura específica y un proceso de calibración detallado del campo magnético en cada ubicación. Este proceso puede ser complejo y costoso de implementar en todas las tiendas de Inditex.

Las soluciones analizadas no son adecuadas debido a su servicio limitado de localización interior o al elevado costo y desafío técnico que implica su implementación en cada tienda de Inditex.

A continuación, se propone una solución que está especialmente adaptada para el entorno comercial de Inditex y no implica un costo tan elevado como la implementación basada en campos magnéticos. Además, ofrece un servicio preciso que permite localizar un artículo en concreto dentro de la tienda y visualizar un plano más detallado del espacio, mejorando significativamente la experiencia de compra.

2.5 Solución propuesta

En este trabajo se desarrollará una aplicación móvil especializada para Inditex enfocada en mejorar la experiencia de navegación interior en sus establecimientos comerciales.

La aplicación permitirá a los usuarios localizar con precisión productos específicos. Utilizando tecnologías avanzadas como Beacons, la aplicación calculará la posición del usuario en tiempo real. Esto permitirá ofrecer una navegación intuitiva a través de mapas detallados basados en Google Maps, complementados con una capa de GeoJson para una representación visual clara del entorno.

Esta solución no solo mejorará la experiencia de compra de los clientes, sino que también optimizará la gestión interna de los establecimientos al facilitar la ubicación de productos y la navegación eficiente del personal en tiempo real.

3. Metodología

En este capítulo se habla de las metodologías y prácticas empleadas durante el desarrollo del proyecto. Se detallan las dinámicas de trabajo y las herramientas de comunicación y organización utilizadas.

3.1 Metodología de trabajo

En el desarrollo del proyecto, se ha aplicado una metodología ágil [1], enfocada en la flexibilidad, la colaboración y la entrega continua de valor. A diferencia de un plan rígido y secuencial, las metodologías ágiles permiten adaptarse rápidamente a los cambios y responder a las necesidades emergentes de manera eficiente.

En concreto, se han seguido los principios de la metodología Kanban. Se utilizó un tablero Kanban [2], una herramienta visual que ayuda a organizar y gestionar el flujo de trabajo. Kanban permite limitar el trabajo en progreso y promover mejoras continuas, proporcionando una visualización clara del progreso. La Figura 5 muestra una captura del tablero de Kanban en un momento concreto del proyecto.

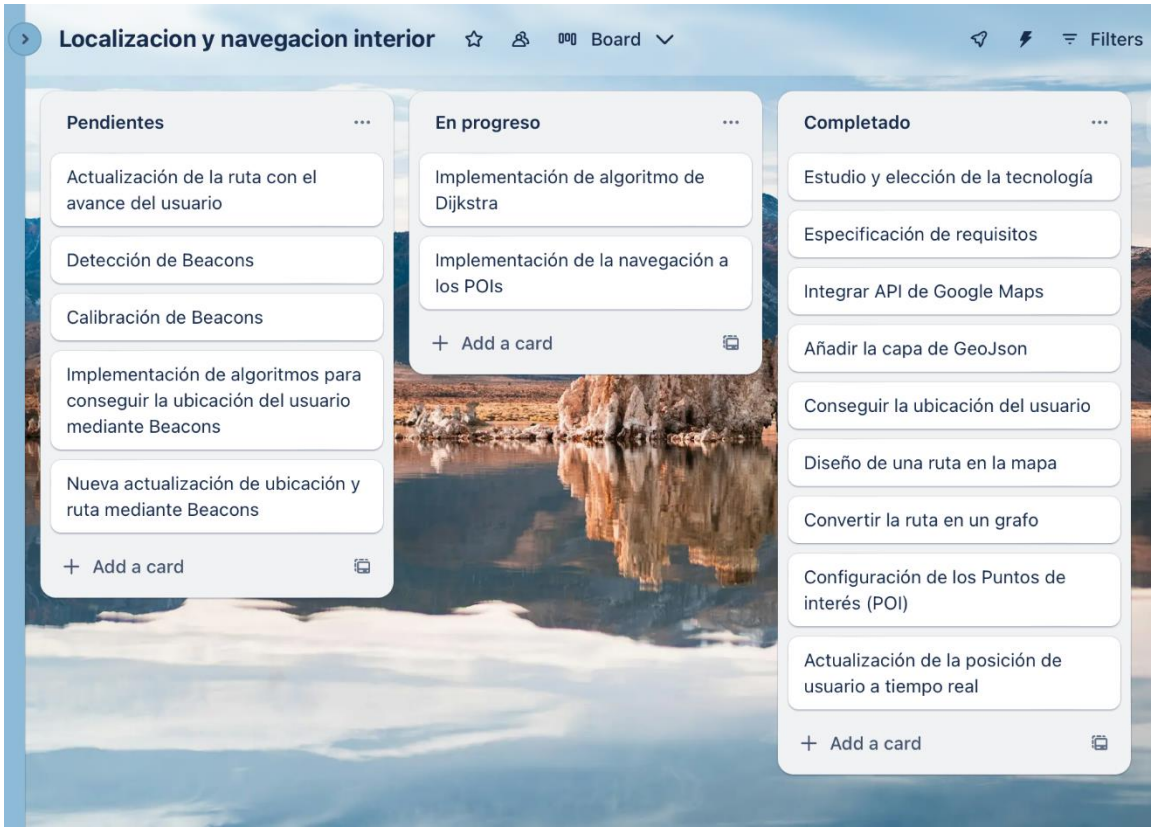

Figura 5: Tablero Kanban

La aplicación de la metodología Kanban requiere de reuniones continuas. Desde el primer día de trabajo, se estableció un régimen de reuniones diarias (Daily) con los tutores del proyecto, Ismael Gonzales Sierra y José Manuel Martin Prado. Estas

reuniones, también conocidas como "stand-ups", no solo han servido para presentar y discutir el progreso del proyecto, sino también para identificar obstáculos y ajustar prioridades. Estas reuniones han permitido determinar los cambios necesarios de manera eficiente para mantener el proyecto alineado con los objetivos y expectativas.

Adicionalmente, cada viernes se organiza una reunión de todo el equipo de Mobile Unit (Weekly), donde todos los miembros se congregan para revisar el estado general del proyecto y coordinar esfuerzos. La integración en el equipo del departamento de Mobile Unit se ha realizado de manera remota, utilizando Microsoft Teams como la herramienta principal para la comunicación y colaboración. En este entorno dinámico, la mayoría de los colaboradores trabajan de forma remota, lo que facilita la flexibilidad y la adaptabilidad en el trabajo.

Además de las reuniones con los tutores de la empresa, se mantienen reuniones periódicas con la tutora de la universidad, Manuela Albert Albiol. Durante estas sesiones, se revisan los hitos alcanzados y se reciben perspectivas académicas y profesionales, que ofrecen orientación y feedback.

En conjunto, el enfoque ágil ha permitido una gestión más eficaz del proyecto, asegurando que se puedan hacer ajustes oportunos y que el trabajo se realice de manera colaborativa y transparente.

3.2 Gestión de proyecto con Git-Flow

Para mantener un control efectivo del desarrollo del software se ha utilizado Git-Flow [3]. Git-Flow es una estrategia de ramificación que facilita la colaboración en equipos y asegura que las diferentes etapas del desarrollo, como nuevas funcionalidades, corrección de errores y lanzamientos, se manejen de manera organizada y eficiente. Fue introducida por Vincent Driessen en 2010 y proporciona una estructura clara y organizada para la gestión de ramas y versiones del código. Git-Flow define un conjunto de prácticas y reglas que ayudan a mantener el proyecto ordenado y facilitar la colaboración entre los desarrolladores. La utilización de Git-Flow ayuda a asegurar un flujo de trabajo organizado y eficiente.

A continuación, se detalla el proceso que se ha seguido en el proyecto aplicando Git-Flow, incluyendo la gestión de Issues, el uso de ramas en Git y la revisión de código.

3.2.1 Creación y gestión de Issues

Una vez se tienen los requisitos especificados, se plantea la creación de Issues en el repositorio de Git. Las Issues son elementos fundamentales en la gestión del proyecto, ya que permiten documentar y rastrear el progreso de las tareas. La Figura 6 muestra una captura de una Issue creada, donde se puede observar que la Issue incluye:

- **Descripción de la Tarea:** Un resumen detallado de lo que se debe realizar.
- **Puntos Que Cumplir:** Objetivos específicos y criterios de finalización.

- **Pruebas de Aceptación:** Pasos para verificar que la tarea cumple con los requisitos definidos.

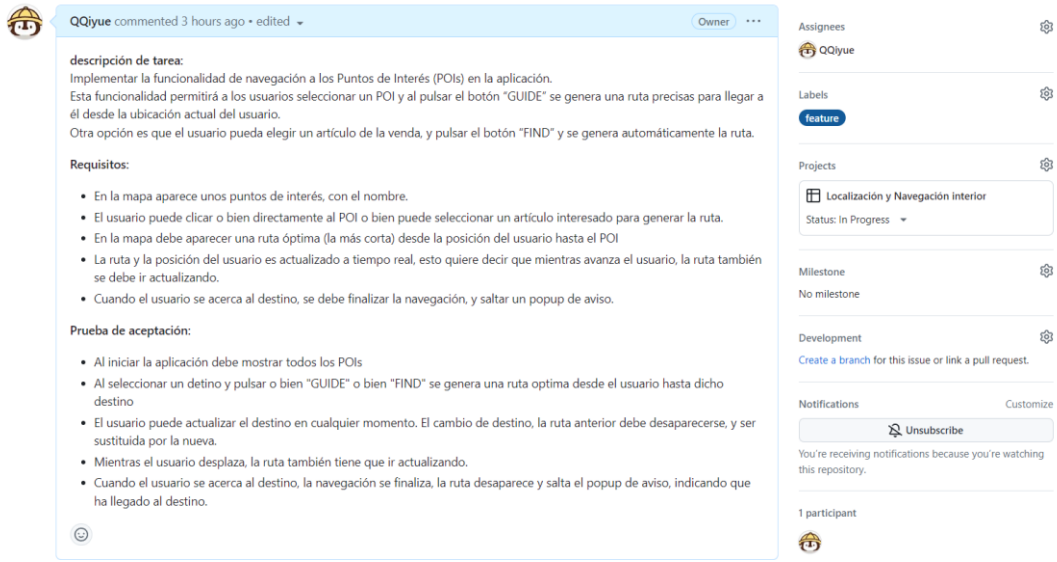

Figura 6: Captura de una Issue

3.2.2 Estados y etiquetas de las Issues

En GitHub, se pueden personalizar los estados de las issues para controlar mejor el progreso de desarrollo y facilitar la gestión dentro de un equipo. Esta personalización se logra principalmente mediante el uso de:

- **Etiquetas (Labels):** Las etiquetas son una herramienta para categorizar y priorizar las issues. Se pueden crear etiquetas personalizadas que reflejen el estado, prioridad, tipo de trabajo o cualquier otra característica relevante para tu proyecto. Por ejemplo, etiquetas como bug, feature, enhancement, urgent, low priority e in progress pueden proporcionar una visión clara del tipo y la urgencia de las tareas. La figura 7 muestra un listado de etiquetas disponibles.
- **Columnas (Status):** Las columnas en los proyectos de GitHub permiten visualizar el flujo de trabajo de una manera más estructurada. Al usar columnas claras en un proyecto, se puede tener una visualización del flujo de trabajo más clara, visualizando el estado de cada tarea, facilitando de esta manera la gestión del progreso del proyecto. Por otra parte, permite que los miembros del equipo puedan ver qué tareas están pendientes, en progreso o completadas, de esta manera mejora la colaboración y la comunicación. También ayuda a identificar rápidamente dónde puede existir cuellos de botella, permitiendo tomar medidas correctivas.

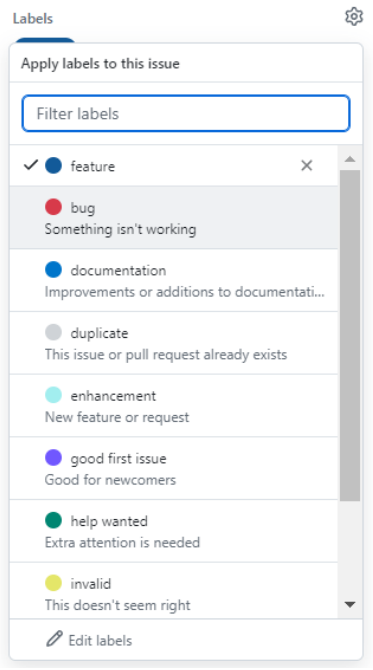

Figura 7: Ejemplos de Labels

Algunos ejemplos de columnas son (ver Figura 8):

- **To Do:** Tareas que han sido creados, pero aún no han comenzado.
- **In Progress:** Tareas en las que está trabajando actualmente.
- **In Review:** Tareas que están completadas, pero necesitan revisión.
- **Done**: Tareas que han sido completadas y revisadas.

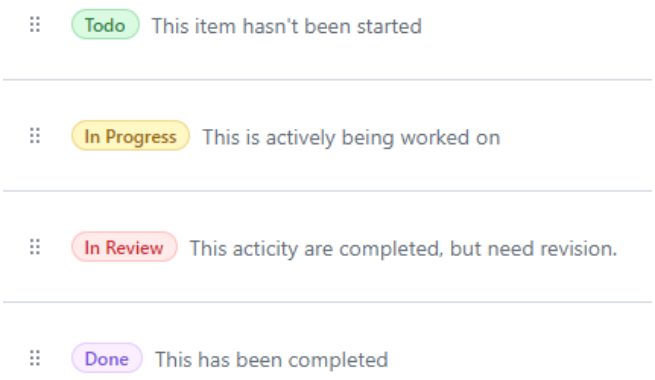

Figura 8: Tipo de columnas

3.2.3 Solicitud de Pull Request y revisión de código

Para cada Issue se crea una nueva rama con el nombre del Issue, y una vez completado el desarrollo de las tareas asociadas, se procede a crear una solicitud de extracción (PR). Esta solicitud es revisada por otro miembro del equipo para garantizar

que el código cumpla con los estándares de calidad establecidos y no introduzca errores en el proyecto.

La revisión del código implica verificar que el código esté conforme con las normas de codificación y que la funcionalidad implementada sea correcta y cumpla con los requisitos especificados en la Issue correspondiente. Además, se realizan pruebas de aceptación para asegurar que la tarea satisfaga los criterios de aceptación definidos inicialmente, asegurando así que la nueva funcionalidad se integre de manera adecuada y sin problemas en el código base del proyecto.

3.2.4 Fusión con la rama principal

La rama principal del proyecto es main, en esta rama no se trabaja directamente, solo se fusionan códigos revisados. Después de finalizar y aprobar un Pull Request de una rama de Issue, el siguiente paso es fusionar esa rama directamente con main. Esto garantiza que solo el código revisado y aprobado se incluya en la versión principal del proyecto.

Este proceso asegura que las nuevas funcionalidades desarrolladas estén disponibles de manera estable y sin errores para todos los usuarios del proyecto en la versión principal del código.

3.2.5 Beneficios del uso de Git-Flow

El uso de Git-Flow en la gestión de proyectos ofrece beneficios significativos para el desarrollo y la gestión eficiente de proyectos de software. Primero, al organizar las tareas en ramas individuales para cada función o característica, Git-Flow proporciona una estructura clara y ordenada que facilita el seguimiento del progreso del proyecto. Esta separación permite que cada tarea se desarrolle de manera independiente, lo cual es crucial para mantener la coherencia y el control sobre las distintas partes del proyecto.

Además, el proceso de revisión de código mediante solicitudes de extracción (PRs) asegura que solo el código validado y de calidad se incorpore a la rama principal del proyecto. Esta práctica no solo mejora la calidad del código base, sino que también minimiza la introducción de errores y asegura que el software final entregado sea robusto y confiable

4. Análisis del problema

En este capítulo, abordaremos la especificación de requisitos del proyecto. La especificación de requisitos es un paso fundamental en el proceso de desarrollo de software, ya que define las funciones, características y comportamientos que el sistema debe cumplir para satisfacer las necesidades del usuario y alcanzar los objetivos del proyecto.

En el siguiente apartado, se detallan los requisitos funcionales del sistema, que abarcan desde la gestión de permisos hasta la búsqueda de productos en la tienda.

4.1 Requisitos funcionales

Para realizar la especificación de requisitos, se ha llevado a cabo un análisis de la documentación proporcionada por la empresa, que incluía sus objetivos y el impacto esperado del proyecto. Utilizando esta información como punto de partida, se han identificado y compilado una lista completa de requisitos del sistema. Esta lista abarcaba funciones, características y comportamientos específicos que el sistema debía cumplir para satisfacer las necesidades del usuario y alcanzar los objetivos del proyecto.

A continuación, se detallan los requisitos de la aplicación agrupados por características.

- **Gestión de Permisos**
  - RF1.1: La aplicación debe solicitar permisos de ubicación, Bluetooth y detección de dispositivos cercanos al iniciar.
  - RF1.2: La aplicación debe mostrar un aviso si algún permiso es denegado, indicando que el permiso es obligatorio para la funcionalidad de la aplicación
- **Visualización del Mapa**
  - RF2.1: La aplicación debe mostrar un mapa
  - RF2.2: La aplicación debe superponer un plano de GeoJson sobre el mapa, mostrando la estructura del edificio.
  - RF2.3: La aplicación debe mostrar puntos de interés en el mapa, que representen la ubicación de los elementos de venta.
- **Interacción con Puntos de Interés**
  - RF3.1: Los puntos de interés deben ser clicables.
  - RF3.2: Al seleccionar un punto de interés y presionar el botón "Guide", la aplicación debe generar una ruta desde la ubicación del usuario hasta el punto de interés seleccionado.
  - RF3.3: La ruta debe actualizarse en tiempo real a medida que el usuario avanza hacia el destino.
  - RF3.4: La aplicación debe mostrar un popup de notificación cuando el usuario llegue al destino y debe desaparecer la ruta.
- **Búsqueda de Producto**

    - RF4.1: La aplicación debe tener una pantalla que muestre una lista de ropas.
    - RF4.2: Cada elemento de la lista debe tener un botón "Find".
    - RF4.3: Al pulsar el botón "Find", la aplicación debe generar automáticamente una ruta hacia el producto seleccionado y cambiar a la pantalla de navegación.

Una vez que los requisitos fueron identificados y aprobados, estos se convirtieron en "Issues" en Git, donde fueron desarrollados y documentados siguiendo un formato acordado. Este enfoque garantizó una gestión eficiente y transparente de los requisitos del proyecto, facilitando su seguimiento y ejecución durante todo el ciclo de desarrollo del software.

4.2 Requisitos no funcionales

A continuación, se muestra los requisitos no funcionales de la aplicación.

- **Usabilidad**
    - RNF1: La aplicación debe ser intuitiva y fácil de usar incluso para usuarios no familiarizados con tecnología.
    - RNF2: El tiempo de respuesta de la aplicación para mostrar mapas y actualizar rutas no deben superar los 3 segundos en condiciones normales de carga.
- **Compatibilidad**
    - RNF3: La aplicación debe ser compatible con una variedad de dispositivos móviles y tablets (siempre y cuando sea de Android).

4.3 Casos de uso

Un caso de uso es una técnica detallada que describe cómo interactúan los usuarios con un sistema para lograr objetivos específicos. En la Figura 9 se muestra el diagrama de casos de uso del proyecto.

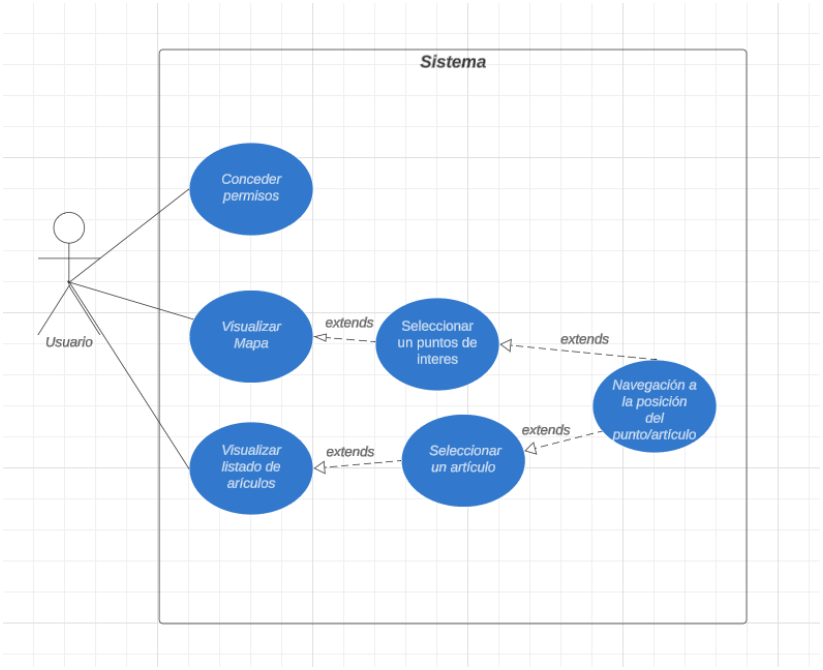

Figura 9: Caso de uso

A continuación, se define cada caso de uso:

Caso de uso	Conceder permisos
Actores	Usuario
Propósito	Conceder los permisos de localización, Bluetooth y detección de dispositivos cercanos
Resumen	Antes de acceso a la aplicación, el usuario debe tener activo los permisos adecuados (localización, Bluetooth y detección de dispositivos cercanos)
Precondiciones	-
Postcondiciones	El usuario se accede correctamente a la aplicación

Tabla 1: Caso de uso “Conceder permisos”

Caso de uso	Visualizar mapa
Actores	Usuario
Propósito	Visualización del mapa
Resumen	Un usuario puede visualizar el mapa junto con la capa de GeoJson para conocer el ámbito interior
Precondiciones	Tener todos los permisos concedidos
Postcondiciones	-

Tabla 2: Caso de uso “Visualizar mapa”

Caso de uso	Visualizar listado de artículos
Actores	Usuario
Propósito	Visualizar la lista de los artículos
Resumen	Un usuario puede visualizar la lista de artículos (prendas)
Precondiciones	Tener todos los permisos concedidos
Postcondiciones	-

Tabla 3: Caso de uso “Visualizar listado de artículos”

Caso de uso	Seleccionar un punto de interés
Actores	Usuario
Propósito	Selección de un punto como destino de la navegación
Resumen	Un usuario puede seleccionar un punto de interés como destino de la navegación
Precondiciones	Tener todos los permisos concedidos
Postcondiciones	Punto seleccionado

Tabla 4: Caso de uso “Seleccionar un punto de interés”

Caso de uso	Seleccionar un artículo
Actores	Usuario
Propósito	Seleccionar un artículo de interés como destino de la ruta
Resumen	Un usuario puede seleccionar un artículo como destino de la navegación
Precondiciones	Tener todos los permisos concedidos
Postcondiciones	Artículo seleccionado

Tabla 5: Caso de uso “Seleccionar un artículo”

Caso de uso	Navegación a la posición del artículo
Actores	Usuario
Propósito	Generar una ruta de navegación al destino
Resumen	Una vez seleccionado o bien un punto o bien un artículo, se puede pulsar el botón para generar una ruta desde la posición del usuario hasta el destino seleccionado y comenzar la navegación.
Precondiciones	Tener un punto/articulo seleccionado
Postcondiciones	Se generará una ruta hacia el destino

5. Tecnologías utilizadas

En este capítulo, se detalla las tecnologías empleadas, tanto las tecnologías empleadas para el desarrollo de software como las tecnologías de posicionamiento en interiores.

5.1 Tecnologías para el desarrollo software

En esta sección se detallan las tecnologías utilizadas en el desarrollo de la aplicación. Como entorno de desarrollo se utilizó Android Studio (sección 5.1.1) y el lenguaje de programación principal es Kotlin (sección 5.1.2).

Para la creación de la IGU, se optó por Jetpack Compose (sección 5.1.3), un framework que simplifica la creación de interfaces, reemplazando el modelo tradicional basado en XML.

Además, se integraron diversas APIs y librerías especializadas, como Google Maps API (sección 5.1.4), GeoJson (sección 5.1.5) para la representación visual de la estructura interior, Accompanist Permissions (sección 5.1.6) para la gestión de permisos en Android, la tecnología de Beacons (sección 5.1.7) para el posicionamiento en interior, JUnit (sección 5.1.8) y Mockk (sección 5.1.9) para la realización de pruebas unitarias.

5.1.1 Android Studio

Android Studio [4] es el entorno de desarrollo integrado (IDE) oficial creado por Google para el desarrollo de aplicaciones Android. Diseñado específicamente para facilitar la creación, depuración y optimización de aplicaciones móviles, Android Studio se ha establecido como la herramienta estándar utilizada por millones de desarrolladores en todo el mundo.

Entre sus características principales se destacan una interfaz de usuario intuitiva que proporciona acceso fácil a todas las herramientas necesarias, una potente herramienta de edición de código con funciones como resaltado de sintaxis y detección de errores en tiempo real, emuladores integrados para probar aplicaciones en una variedad de dispositivos Android virtualmente, y herramientas avanzadas de depuración para identificar y resolver problemas rápidamente.

Además, Android Studio ofrece una integración fluida con los servicios de Google, facilitando la incorporación de funciones como Google Maps, Firebase y Google Analytics en las aplicaciones desarrolladas. Es compatible tanto con Kotlin como con Java, permitiendo a los desarrolladores elegir el lenguaje que mejor se adapte a sus necesidades y preferencias.

5.1.2 Kotlin

Kotlin [5] es un lenguaje de programación que apareció en el año 2011 diseñado por JetBrains. Tiene un carácter multiplataforma, estáticamente tipado, de alto nivel y propósito general con inferencia de tipos.

Una de las características más atractivas de Kotlin es su capacidad para interoperar de manera fluida con Java. Esto significa que los desarrolladores pueden utilizar las bibliotecas y frameworks existentes de Java, haciendo la transición a Kotlin mucho más sencilla para aquellos que ya están familiarizados con el ecosistema de Java. Además, Kotlin se puede compilar a bytecode de Java, lo que permite su ejecución en cualquier máquina virtual Java (JVM), así como a JavaScript o nativo para desarrollos multiplataforma.

El 7 de mayo de 2019, Google anunció que Kotlin se convertiría en el lenguaje preferido para el desarrollo de aplicaciones Android [6], lo que marcó un hito significativo en su adopción. Desde entonces, Kotlin ha ganado una enorme popularidad entre los desarrolladores de Android debido a sus características avanzadas, como la seguridad de nulabilidad (NULL safety), las corrutinas para programación asincrónica y sus expresiones lambda, que simplifican el manejo de colecciones y flujos de datos.

Kotlin es un potente lenguaje de programación multiplataforma, con su sintaxis concisa, su fácil interoperabilidad con Java y su compatibilidad con una amplia gama de plataformas y tecnologías. Kotlin no solo es popular en el desarrollo de aplicaciones Android, sino que también se utiliza en el desarrollo del lado del servidor, aplicaciones web, y en el ámbito de la ciencia de datos.

5.1.3 Jetpack Compose

Jetpack Compose [7] es un kit de herramientas moderno diseñado para simplificar el desarrollo de IU. Combina un modelo de programación reactivo con la concisión y facilidad de uso del lenguaje de programación Kotlin. Es totalmente declarativo, lo que significa que describes tu IU si llamas a una serie de funciones que transforman datos en una jerarquía de IU. lo que permite a los desarrolladores describir cómo debería ser y comportarse la interfaz de usuario en lugar de manipular directamente los elementos de UI, como se hace tradicionalmente con XML en Android.

Jetpack Compose también se integra sin problemas con el sistema de vistas tradicional de Android, lo que facilita la coexistencia de composables y vistas tradicionales dentro de la misma aplicación, permitiendo una migración progresiva hacia esta nueva forma de desarrollo de UI.

En términos de herramientas de desarrollo, Jetpack Compose está completamente integrado con Android Studio, ofreciendo un previsualizador en vivo, edición rápida y herramientas de depuración robustas que mejoran significativamente la experiencia de desarrollo. Además, está diseñado para ser eficiente en términos de rendimiento, minimizando la sobrecarga al actualizar solo las partes necesarias de la interfaz cuando cambian los estados, lo que optimiza la experiencia del usuario final.

5.1.4 Google Maps API

Google Maps API [8] es una colección de servicios y herramientas proporcionados por Google que permiten a los desarrolladores integrar mapas interactivos y funcionalidades relacionadas en sus aplicaciones y sitios web. Estas API ofrecen una amplia gama de capacidades, desde la visualización de mapas hasta la obtención de direcciones, geocodificación, seguimiento de ubicación en tiempo real y mucho más.

Características principales de Google Maps API:

- **Visualización de Mapas:** Google Maps API permite a los desarrolladores integrar mapas interactivos en sus aplicaciones y sitios web, lo que permite a los usuarios explorar ubicaciones, buscar lugares de interés y obtener información geográfica detallada.
- **Servicios de Geocodificación:** La API de Geocodificación de Google Maps permite convertir direcciones de texto en coordenadas geográficas (latitud y longitud) y viceversa, lo que facilita la integración de funciones de búsqueda de ubicación en las aplicaciones.
- **Obtención de Direcciones:** La API de Direcciones de Google Maps permite a los desarrolladores calcular rutas y obtener instrucciones paso a paso para llegar de un lugar a otro, ya sea a pie, en coche, en bicicleta o en transporte público.
- **Seguimiento de Ubicación en Tiempo Real:** Google Maps API ofrece capacidades para el seguimiento de ubicación en tiempo real, lo que permite a los desarrolladores rastrear la ubicación de los usuarios y mostrar su posición en el mapa en tiempo real.
- **Personalización y Estilo:** Los desarrolladores pueden personalizar la apariencia y el estilo de los mapas integrados utilizando la API de Estilos de Google Maps, lo que les permite adaptar el aspecto de los mapas a la estética de sus aplicaciones y sitios web.
- **Integración con otros Servicios de Google:** Google Maps API se integra sin problemas con otros servicios de Google, como Places API, Street View API y Firebase, lo que permite a los desarrolladores agregar funcionalidades adicionales, como búsqueda de lugares, imágenes panorámicas y almacenamiento de datos en la nube.

Es importante tener en cuenta que, aunque Google Maps API ofrece una parte de sus servicios de forma gratuita, el uso intensivo o servicios adicionales pueden generar costos adicionales para los desarrolladores. Sin embargo, Google ofrece un generoso crédito inicial para el desarrollo y una estructura de precios flexible basada en el volumen de consultas.

5.1.5 GeoJson

GeoJson [9] es un formato abierto y ligero diseñado específicamente para representar datos geoespaciales de manera legible para humanos y fácilmente procesable por máquinas. Este formato se basa en JSON (JavaScript Object Notation), un estándar ampliamente reconocido en el mundo de la programación. GeoJson se utiliza principalmente para describir características geográficas como puntos de interés, rutas, límites administrativos y más, utilizando objetos geométricos simples como puntos, líneas y polígonos, junto con sus atributos asociados.

Una de las principales ventajas de GeoJson radica en su estructura basada en JSON, la cual utiliza objetos y arrays que son intuitivos tanto para humanos como para sistemas automatizados. Esta característica facilita su manipulación y entendimiento, lo que lo convierte en una elección ideal para el intercambio de datos entre diferentes sistemas y aplicaciones.

GeoJson también destaca por su amplia compatibilidad con tecnologías web modernas. Es fácilmente integrable y procesable por bibliotecas y herramientas JavaScript, así como por servicios de mapas en línea populares como Google Maps y Leaflet. Esta compatibilidad garantiza que los datos geoespaciales representados en GeoJson puedan ser utilizados de manera efectiva en aplicaciones web y móviles, proporcionando una visualización precisa y dinámica de la información geográfica.

Además, GeoJson es un formato extensible que permite la inclusión de datos adicionales y metadatos dentro de los objetos geoespaciales. Esta capacidad de extensibilidad ofrece flexibilidad para agregar información específica del dominio o referencias a recursos externos, adaptándose así a una amplia gama de necesidades y contextos de aplicación. En resumen, GeoJson no solo simplifica la representación y manipulación de datos geoespaciales, sino que también facilita su integración en sistemas y aplicaciones modernas de manera eficiente y efectiva.

Para editar y visualizar archivos GeoJson de manera conveniente, herramientas como https://geojson.io/ proporcionan una plataforma web accesible donde los usuarios pueden modificar y visualizar ficheros GeoJson de manera interactiva. Esto permite a usuarios explorar y manipular datos geoespaciales con facilidad. En la Figura 10 se presenta una captura de pantalla de la página web https://geojson.io/, donde se muestra el GeoJson correspondiente a Zara en Plaza España. La capa gris representa los polígonos y LineString definidas en dicho archivo.

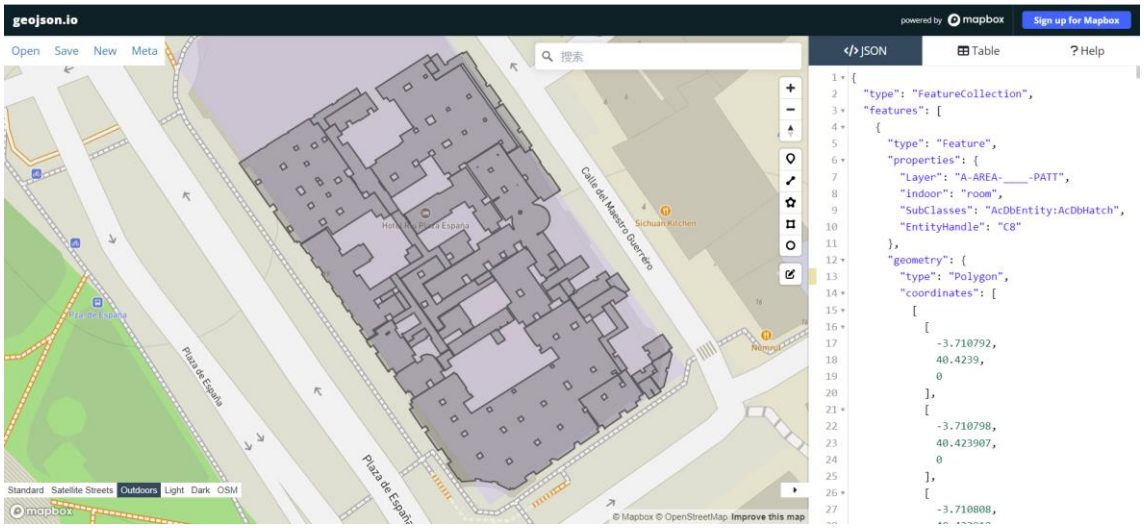

Figura 10: Mostración de GeoJson (Zara, Plaza España)

5.1.6 Accompanist Permissions

Accompanist [10] es una biblioteca Kotlin creada por Chris Banes para simplificar la gestión de permisos en aplicaciones Android con Jetpack Compose. Facilita las solicitudes de permisos con funciones de alto nivel que se integran sin problemas en la interfaz de usuario de Compose. Ofrece manejo asincrónico de permisos, evitando bloqueos en el hilo principal. Su integración fluida con Compose permite gestionar el estado de los permisos de manera clara y eficiente, adaptándose fácilmente a aplicaciones que requieren acceso a recursos protegidos como ubicación, bluetooth o dispositivos cercanos.

5.1.7 Beacons

Los Beacons [11] son dispositivos inalámbricos que utilizan la tecnología Bluetooth de baja energía (BLE) para enviar señales de corto alcance a dispositivos móviles cercanos, transmitiendo información específica de ubicación o contexto. Estos dispositivos pueden ser utilizados para una variedad de aplicaciones, como navegación en interiores, marketing en el lugar, seguimiento de activos y más.

Características de los Beacons:

- Utilizan tecnología Bluetooth de baja energía (BLE).
- Transmiten señales de corto alcance a dispositivos móviles cercanos.
- Permiten la entrega de información específica de ubicación o contexto.
- Pueden ser utilizados para una variedad de aplicaciones.

Parámetros del Beacons:

- **UUID (Universally Unique Identifier):** El UUID es un identificador único global de 128 bits. Su función principal es la identificación del Beacon para otros dispositivos. Cada Beacon tiene un UUID único que lo distingue de otros Beacons en el entorno.
- **Dirección MAC (Media Access Control):** La dirección MAC es un identificador único asignado a la interfaz de red de un dispositivo. En el caso de los Beacons, la dirección MAC se utiliza para identificar de forma única cada Beacons dentro de una red. Esta dirección se utiliza para distinguir los Beacons entre sí y para establecer comunicación con ellos a nivel de hardware. La dirección MAC consta de 6 bytes, organizados en pares hexadecimales y generalmente se muestra en formato hexadecimal, por ejemplo: "AA:BB:CC:DD:EE
- **Mayor (Major):** Se emplea para definir una agrupación lógica o una ubicación más amplia dentro del contexto del UUID. Por ejemplo, un grupo de Beacons en un piso específico de un edificio puede tener el mismo UUID, pero diferentes números de mayor para distinguirlos.
- **Menor (Minor):** El Menor es otro número entero que se utiliza junto con el UUID y el valor Mayor para identificar un Beacon específico dentro de un grupo. Proporciona una identificación adicional y más específica dentro de un conjunto de Beacons que comparten el mismo UUID y Mayor. Por ejemplo, dentro de un grupo de Beacons que representan una tienda, el número Menor podría identificar diferentes secciones o productos específicos.
- **RSSI (Received Signal Strength Indicator):** El RSSI es una medida de la potencia de la señal recibida del Beacon. Indica la intensidad de la señal medida en dBm y se utiliza para estimar la proximidad del dispositivo receptor al Beacon. Una señal RSSI más fuerte indica una mayor proximidad al Beacon.
- **TX Power (Transmit Power):** El TX Power es la potencia de transmisión del Beacon, medida en dBm. Indica la potencia de la señal transmitida por el Beacon a una distancia de 1 metro. Se utiliza junto con el RSSI para estimar la distancia entre el Beacon y el dispositivo receptor mediante técnicas de trilateración o triangulación.
- **Intervalo de transmisión (Broadcasting Interval):** Es el período de tiempo en el que el Beacon transmite sus señales. Se mide en milisegundos y determina la frecuencia con la que el Beacon envía paquetes de señalización. Un intervalo de transmisión más corto resulta en una detección más rápida del Beacon, pero también puede afectar la duración de la batería del dispositivo.
- **Radio Tx Power (Potencia de Transmisión):** Este parámetro determina la potencia a la cual el Beacon transmite sus señales. Se expresa típicamente en decibelios (dBm) y afecta la distancia a la cual el Beacon puede ser detectado por dispositivos receptores. Una potencia de transmisión más alta permite una mayor cobertura y alcance, pero también puede aumentar el consumo de energía y afectar la duración de la batería del Beacon y de los dispositivos receptores.
- **RSSI Calibrado a 1 metro:** El RSSI calibrado a 1 metro es la medida estándar de la potencia de la señal recibida por un Beacon cuando un dispositivo receptor se encuentra a una distancia de 1 metro en condiciones ideales. Esta

medida se utiliza como referencia para estimar la proximidad del dispositivo al Beacon en tiempo real, ayudando a calcular la distancia mediante técnicas de trilateración o triangulación basadas en la diferencia de potencia de la señal recibida (RSSI)

Figura 11: BlueBeacon Manager

En la figura 11 se muestra la detección de los Beacons por la aplicación del BlueBeacon Maneger, donde se puede observar por ejemplo el nombre del Beacon (Mini S/N 035336), dirección Mac (FE: 0E:E6:40:7A:27), RSSI(-58dBm), batería (99%), UUID(ACFD065SE-C3C0-11E3-9BBE-1A514932AC01), Major (1), Minor (35336).

Existen varios tipos de Beacons, pero los dos más comunes son:

1. **iBeacon**: Un estándar desarrollado por Apple, utilizado principalmente en dispositivos iOS, que envía un identificador único a dispositivos cercanos para su detección y seguimiento de ubicación.
2. **Eddystone**: Un protocolo de código abierto desarrollado por Google, que transmite información diversa, como URLs, identificadores únicos, entre otros, y es compatible con una variedad de plataformas.

Los Beacons se utilizan en una variedad de aplicaciones, incluyendo:

1. **Navegación en Interiores**: Ayudan a los usuarios a navegar en entornos interiores, como centros comerciales, aeropuertos y museos.
2. **Marketing en el Lugar**: Permiten a las empresas enviar promociones y anuncios personalizados a clientes cercanos.
3. **Seguimiento de Activos**: Se utilizan para rastrear y gestionar activos, como equipos y productos, dentro de un entorno específico.
4. **Experiencias de Usuario Mejoradas**: Permiten la entrega de experiencias de usuario contextuales y enriquecidas, como guías turísticas interactivas y exhibiciones en museos.

5.1.8 JUnit

JUnit [12] es un framework diseñado para realizar pruebas unitarias en Java y otros lenguajes de programación como Kotlin. Su principal objetivo es ejecutar clases y validar si el comportamiento de cada método cumple con las expectativas definidas. Esto se logra mediante el uso de anotaciones como @Test, que marca un método como prueba unitaria. JUnit proporciona un entorno estructurado para ejecutar pruebas de manera repetible y eficiente, facilitando la detección temprana de errores en el código.

5.1.9 Mockk

Un mock es un objeto simulado que imita el comportamiento de un componente real durante las pruebas unitarias.

Mockk [13]es un framework de simulación diseñado específicamente para Kotlin. Permite crear objetos simulados (mocks) para facilitar las pruebas unitarias e integración en aplicaciones Kotlin. La principal fortaleza de Mockk reside en su sintaxis concisa y en cómo aprovecha las características del lenguaje Kotlin para simplificar la creación y manipulación de estos objetos simulados.

5.2 Tecnologías de Posicionamiento

Hoy en día, la tecnología de posicionamiento se ha convertido en una parte indispensable de nuestra vida y trabajo diario. Según el escenario de aplicación, la tecnología de posicionamiento se puede dividir en tecnología de posicionamiento exterior y tecnología de posicionamiento interior. El posicionamiento exterior depende principalmente de la tecnología de satélites, siendo el Sistema de Posicionamiento Global (GPS) el más utilizado.

Sin embargo, a medida que las actividades humanas se trasladan cada vez más a interiores, como centros comerciales, oficinas, hospitales, etc., la señal de satélite se ve gravemente atenuada por la obstrucción de los edificios, lo que reduce significativamente la precisión del posicionamiento y no puede satisfacer las necesidades de posicionamiento interior.

En esta sección, se ofrece una visión general de varias de las tecnologías más utilizadas en los sistemas inalámbricos de posicionamiento en interiores, como Identificación por Radiofrecuencia (RFID), Espectro Ensanchado por Barrido (CSS), Banda Ultra ancha (UWB), Campo magnético, Wifi y Bluetooth, y se intenta comparar las distintas tecnologías.

La elección de la tecnología de posicionamiento interior debe considerar varios factores, como la precisión del posicionamiento, el coste, el consumo de energía y la adaptabilidad al entorno. Diferentes tecnologías tienen ventajas en diferentes escenarios de aplicación, y mediante una selección y despliegue razonables, se puede mejorar significativamente el efecto del posicionamiento interior y la experiencia del usuario.

5.2.1 Radiofrecuencia (RFID)

La tecnología de posicionamiento RFID [14] es un tipo de tecnología de identificación automática que utiliza ondas electromagnéticas para la comunicación, permitiendo el intercambio de datos de manera bidireccional mediante señales de radiofrecuencia. De esta manera, se logran funciones de identificación y posicionamiento. Los sistemas RFID se componen de varios componentes básicos, incluyendo múltiples lectores RFID y etiquetas RFID. Entre los lectores y las etiquetas se establece una comunicación utilizando señales específicas de radiofrecuencia y protocolos asociados para enviar y recibir datos.

Los lectores RFID pueden leer señales emitidas por etiquetas conocidas, donde la proximidad entre ambos influye directamente en el valor de RSSI. Por lo tanto, el sistema de posicionamiento RFID puede basarse en un modelo de relación RSSI-distancia para detectar los valores de RSSI entre la etiqueta a posicionar y el lector correspondiente, utilizando algoritmos adecuados para determinar la ubicación de la etiqueta.

5.2.2 Espectro Ensanchado por Barrido (CSS)

La tecnología de posicionamiento CSS [15], basada en el estándar IEEE 802.15.4a de redes de área local inalámbrica, utiliza señales de chirp para la comunicación. Estas señales exhiben características de modulación lineal de frecuencia dentro de un solo ciclo y emplean técnicas avanzadas de compresión de pulsos para concentrar la energía de recepción. Esto proporciona a CSS una capacidad excepcional para resolver problemas de multipath y resistir interferencias en entornos complejos.

Nanotron, a través de su desarrollo del chip RF nanoLOC TRX operando en la banda de 2.4 GHz ISM, ha potenciado esta tecnología para realizar mediciones precisas de distancia entre nodos, fundamentales para aplicaciones de posicionamiento en interiores.

5.2.3 Banda Ultra ancha (UWB)

La tecnología de posicionamiento UWB [16] (Ultra Wideband) es una de las más prometedoras para el posicionamiento en interiores. UWB es una técnica de comunicación inalámbrica sin portadora, que utiliza señales de pulso de

nanosegundos para transmitir datos en un ancho de banda extremadamente amplio. Gracias a estos pulsos de corta duración, los sistemas UWB no requieren los dispositivos de conversión de frecuencia típicos ni amplificadores de potencia o mezcladores en su estructura, lo que simplifica considerablemente su diseño y reduce el consumo de energía. Además, la breve duración de los pulsos permite una estimación de tiempo de llegada (TOA) con una precisión de hasta centímetros, proporcionando un rendimiento excepcional en entornos con múltiples trayectorias. No obstante, la baja potencia de transmisión de UWB limita la distancia de transmisión y la velocidad de datos, lo que representa las principales desventajas de los sistemas de posicionamiento UWB [17].

5.2.4 Posicionamiento con Wifi

El posicionamiento interior con Wifi es ampliamente utilizado en puntos de acceso públicos, trabajos y hogares. Esta tecnología puede utilizar las redes Wifi ya existentes, eliminando la necesidad de construir infraestructuras adicionales, lo que la convierte en una opción muy popular.

Durante el proceso de posicionamiento, las señales Wifi envían periódicamente datos que incluyen el nombre del dispositivo, la dirección MAC y la intensidad de la señal. Los métodos comunes de posicionamiento Wifi incluyen algoritmos de localización basados en RSSI (Indicador de Intensidad de Señal Recibida) y algoritmos de huellas dactilares basados en RSSI.

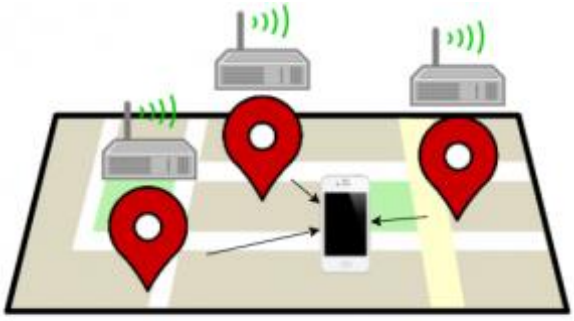

Figura 12: Posicionamiento con Wifi [18]

5.2.5 Campo magnético

Los campos magnéticos están presentes en todos lados. Por ejemplo, la Tierra tiene un campo magnético potente que hace que las brújulas apunten hacia el Norte.

Dentro de los edificios, los campos magnéticos pueden sufrir distorsiones severas causadas por diferentes elementos, tales como los materiales de construcción ferromagnéticos, dispositivos y la instalación eléctrica. Esto es un inconveniente para extraer la orientación del dispositivo, pero ya que la distorsión magnética no es uniforme en todo el espacio, las distorsiones se pueden utilizar para computar la geolocalización del dispositivo. Por suerte, la mayoría de los smartphones modernos

tienen un sensor llamado “magnetómetro”, capaz de medir los 3 componentes del campo magnético.

Las soluciones basadas en campos magnéticos requieren en primer lugar registrar un mapa magnético muy detallado del edificio y después comparar los distintivos magnéticos medidos por el smartphone con los esperados en cada zona del edificio. De este modo, estiman la geolocalización [19].

5.2.6 Posicionamiento con Bluetooth

El posicionamiento por Bluetooth ha sido seleccionado como la tecnología principal para el desarrollo de este trabajo. Este método se fundamenta en la emisión de señales por parte de balizas Bluetooth, como iBeacon o Eddystone, las cuales permiten determinar la ubicación de dispositivos móviles dentro de edificaciones.

Una de las principales ventajas del posicionamiento por Bluetooth radica en su bajo consumo de energía, su costo reducido y su alta precisión. Con la introducción de Bluetooth 4.0 y sus versiones posteriores, se ha implementado Bluetooth de Bajo Consumo (BLE), lo cual facilita que los dispositivos Bluetooth operen durante períodos prolongados sin necesidad de frecuentes recargas de batería.

No obstante, la precisión del posicionamiento Bluetooth puede verse afectada por diversas condiciones ambientales. Las señales Bluetooth pueden sufrir interferencias en entornos con alta actividad electromagnética o con obstáculos físicos, como paredes y grandes objetos metálicos, lo que provoca variaciones en la intensidad de la señal (RSSI). Estas interferencias pueden influir negativamente en el cálculo preciso de la ubicación. Es crucial considerar estos factores y realizar una calibración adecuada de las balizas para obtener resultados más precisos.

Android proporciona soporte nativo para el escaneo de Bluetooth a través de la API BluetoothAdapter, lo cual simplifica el proceso de escaneo. Además, prácticamente todos los dispositivos móviles inteligentes modernos están equipados con tecnología Bluetooth, lo que amplía significativamente la utilidad de esta tecnología en diversos contextos.

Figura 13: Beacons Utilizado durante el proyecto

5.2.7 Comparación de tecnologías de posicionamiento

La tabla 7 muestra una comparación de las tecnologías de posicionamiento presentadas, donde se señalan las ventajas, las desventajas y la precisión de la tecnología.

Tecnología	Ventajas	Desventajas	Precisión
RFID	Sin necesidad de contacto físico. Amplia gama de aplicaciones	Costo inicial	Varía según tipo de RFID
CSS	Alta precisión en interiores	Requiere instalación y mantenimiento costoso	Centímetros a metros
UWB	Alta precisión. Adecuado para entornos críticos	Costoso, consumo de energía elevada	Centímetros
Wifi	Utilización de la infraestructura existentes	Inestable, No funciona en iOS. En Android > 9	5 – 20 metros
Bluetooth	Bajo consumo de energía. Fácil de configuración. Costes relativamente bajo en comparación con otras tecnologías avanzadas	Menor precisión, requiere muchos Beacons en áreas extensas	1 – 10 metros

Tabla 7: Comparación de las tecnologías de posicionamiento interior

Después de estudiar estas tecnologías de posicionamiento interior, se puede concluir que no hay una tecnología que sea mejor que otra, sino que su aplicación varía según las necesidades específicas.

Para el desarrollo de este TFG, se optará por la localización mediante Bluetooth, utilizando balizas de Beacons. Esta elección se fundamenta en su bajo costo de infraestructura, consumo reducido de energía y una precisión aceptablemente buena. Además, su capacidad de personalización y su tamaño físico pequeño hacen que la implementación sea más sencilla en comparación con otras tecnologías.

6. Diseño de la solución

En este capítulo, se presenta en primer lugar la arquitectura de la aplicación desarrollada. La arquitectura muestra los componentes principales en los que se estructura la aplicación. A continuación, se presenta el diseño del algoritmo de posicionamiento interior. Por último, se presenta el diseño del algoritmo de cálculo de ruta.

6.1 Arquitectura de la solución propuesta

En esta sección, se describe la arquitectura de la solución desarrollada. La arquitectura aplica el patrón arquitectónico Model View Intent (MVI). Model-View-Intent (MVI) es un patrón arquitectónico para el desarrollo de aplicaciones que facilita la gestión del estado y las interacciones de la interfaz de usuario (UI). MVI se basa en la idea de un flujo unidireccional de datos, lo que significa que los datos fluyen en una sola dirección desde los usuarios hasta la UI, lo que mejora la previsibilidad y la depuración del código.

Los tres componentes principales de MVI son:

- **Model:** El 'Model' representa la lógica d negocio y el estado de la aplicación, es responsable de manejar los datos.
- **View:** La 'View' es responsable de la presentación de la UI, se suscribe al estado proporcionado por el Model y renderiza la UI en consecuencia. Se encarga de mostrar los datos manera reactiva, actualizando la interfaz cuando cambia el estado.
- **Intent:** Los intent representa las intenciones del usuario, como por ejemplo clics, selecciones o entradas, se traducen en eventos que será procesados por el Model, que puede desencadenar cambios que luego son reflejados en la View.

La Figura 14 muestra una representación visual de estos componentes y sus relaciones. Las características fundamentales del patrón MVI son:

- **Flujo Unidireccional de Datos:** Los datos fluyen en una sola dirección, lo que simplifica la gestión del estado y reduce el riesgo de inconsistencias.
- **Estado Inmutable:** El estado de la UI se representa como un objeto inmutable, lo que facilita la detección de cambios y la depuración.
- **Desacoplamiento:** La separación clara entre la UI y la lógica de negocio permite un código más modular y fácil de mantener.
- **Reactividad:** La UI reacciona automáticamente a los cambios en el estado, proporcionando una experiencia de usuario fluida y dinámica.

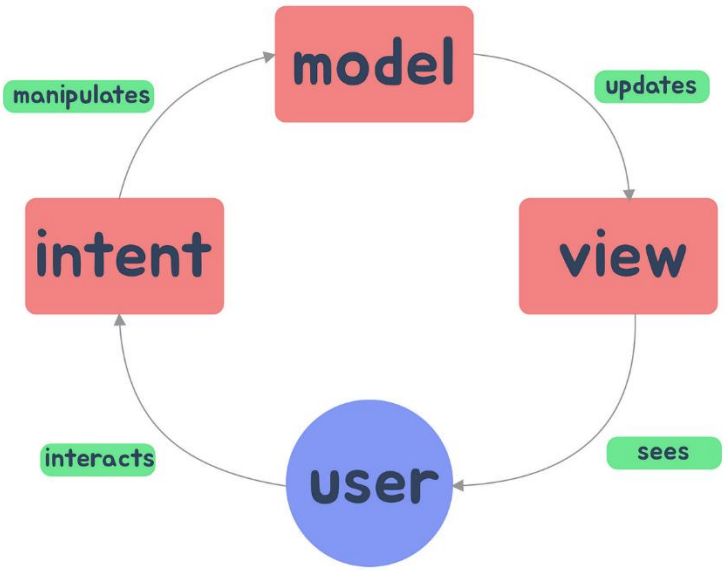

Figura 14: Diagrama de MVI [20]

6.2 Algoritmos para calcular posicionamiento interior

En este apartado, se explican distintos algoritmos para calcular la ubicación del usuario en interiores. Los algoritmos han sido clasificados según su principio de funcionamiento. Esta taxonomía divide los algoritmos principalmente en tres categorías: (1) método de proximidad, (2) método de características geométricas y (3) método de análisis de escenarios. El método de proximidad solo ofrece una estimación de la posición, no tiene una precisión muy alta y su utilización es más limitada que los otros dos métodos, por lo que no se incluyen algoritmos que sigan este método.

6.2.1 Métodos de características geométricas

Los algoritmos que entran dentro de esta categoría aprovechan propiedades como ángulos de llegada, diferencias de tiempo de llegada u otras características de las señales recibidas para calcular la ubicación del usuario.

6.2.1.1 Algoritmo de trilateración

La trilateración [21] es una técnica matemática que se basa en principios geométricos para determinar la ubicación de un punto desconocido, en este caso, la posición de un usuario. El método puede determinar la posición del usuario con al menos tres puntos de acceso preestablecidos.

Dando un ejemplo: Si establecemos 3 puntos A (a1, a2), B (b1, b2), C (c1, c2) y la ratio de cada uno de ellos corresponde la distancia hasta el punto a determinar, podemos imaginar un círculo con este radio para cada uno de los puntos. Y podemos concluir que el punto solución se encuentra en la intersección entre todos los círculos.

En este contexto de localización interior, los puntos fijos son Beacons, sabiendo la posición de estos Beacons, para determinar la distancia entre el usuario y los Beacons se puede aplicar la técnica de trilateración.

$$d = 10\frac{TxPower - RSSI}{10n}$$

Donde:

- d es la distancia
- TxPower es el valor de RSSI esperado a 1 metro de distancia (Es un valor que necesita calibración, mencionado en 5.1.7 Beacons)
- RSSI es la intensidad de señal recibida en el momento
- n es el factor de propagación del entorno, normalmente entre 2 – 4 en interiores. Un valor de n más bajo indica un entorno con pocas obstrucciones y un valor de n más alto indica un entorno más obstruido.

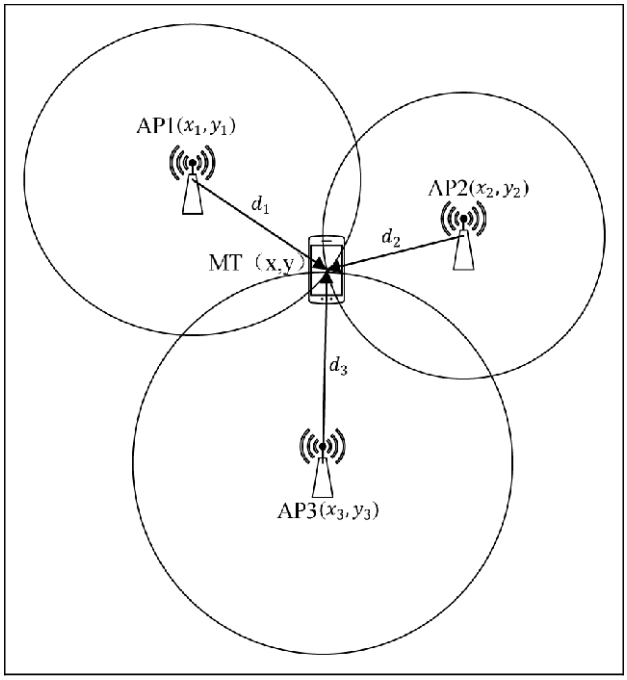

Figura: 15 mostración de trilateración [22]

6.2.1.2 Algoritmo de diferencia de tiempo de llegada (TDOA)

TDOA [23] es un algoritmo basado en comparar la diferencia de tiempo entre las señales y cada ancla y esta técnica requiere una función de sincronización de tiempo precisa. Este método es utilizado en aplicaciones que demanda alta precisión, como sistemas de banda ultra ancha (UWB) y sistemas de posicionamiento acústico.

Dando un ejemplo:

Queremos saber la posición de un barco, el barco emite una señal, tenemos tres puntos de referencia A, B, C (se tiene que saber la ubicación del estos tres puntos). Estos tres puntos juegan el papel de receptor de la señal.

Datos:

- A, B, C: Puntos
- TA, TB, TC: Tiempo de llegada de la señal de diferentes puntos correspondientes
- C: Velocidad de propagación de la señal

Pasos:

1. **Emisión de la señal:**

   El barco emite la señal, pero no sabemos el tiempo exacto

2. **Registro de la Tiempo:**

   Punto A recibe la señal y lo registra como TA

   Punto B recibe la señal y lo registra como TB

   Punto C recibe la señal y lo registra como TC

3. **Cálculo de la diferencia del tiempo (DT)**
   Diferencia de tiempo entre A y B: DT1 =TA - TB
   Diferencia de tiempo entre A y C: DT2 = TA - TC
4. **Diferencia de Distancia (DR)**
   Distancia entre A y B: DR1 = C x (TA - TB)
   Distancia entre A y C: DR2 = C x (TA - TC)
5. **Dibujo de Hipérbolas**

   Usando las diferencias de distancias DR1 y DR2, podemos dibujar hipérbolas en las cuales el punto que queremos localizar se encuentra en algún punto a lo largo de estas curvas.

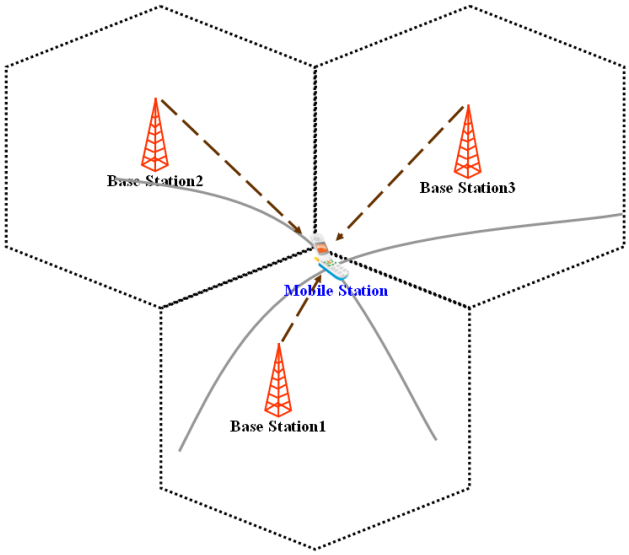

Figura 16: Presentación de hipérbola de TDOA [24]

6.2.1.3 Algoritmo de ángulo de llegada (AOA)

El algoritmo de Ángulo de Llegada (AOA) [25] se utiliza para localizar un dispositivo midiendo el ángulo de llegada de su señal en varias estaciones base. Para que esto funcione, se deben instalar varios receptores de ultrasonido en los puntos receptores, lo que les permite detectar la dirección de la señal.

Cada punto receptor mide el ángulo desde el cual recibe la señal del dispositivo, y con estos ángulos, se pueden crear ecuaciones para determinar la posición del dispositivo. Las ecuaciones son:

$$\tan(\alpha 1) = \frac{y - y1}{x - x1}$$

$$\tan(\alpha 2) = \frac{y - y2}{x - x2}$$

- (x, y) = son las coordenadas del dispositivo
- (x1, y1) y (x2, y2) son las coordenadas de los puntos receptor
- $\alpha 1\ y\ \alpha 2$ son los ángulos medidos

AOA a menudo se combina con el algoritmo TDOA para conseguir una precisión más alta.

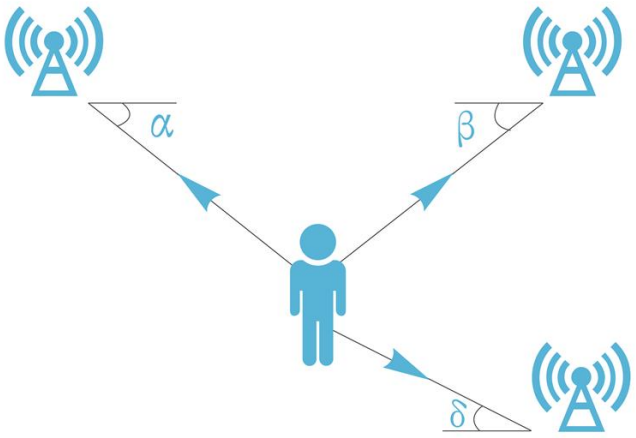

Figura 17: Presentación del algoritmo AOA [26]

6.2.2 Métodos de análisis de escenarios (Fingerprinting)

El método de análisis de escenarios, también conocido como método de coincidencia de huellas (Fingerprinting). Esta técnica se caracteriza por tener un base de datos donde almacena las huellas dactilares de las ubicaciones.

Durante el proceso de localización, se utilizan bases de datos y algoritmos de coincidencia para encontrar la ubicación del dispositivo comparando las características de la señal en tiempo real con las huellas almacenadas.

Los algoritmos de coincidencia comunes incluyen el algoritmo del vecino más cercano (Nearest Neighborhood, NN), el algoritmo de los K vecinos más cercanos (K Nearest Neighborhood, KNN), el algoritmo de los K vecinos más cercanos ponderados (Weighted K Nearest Neighborhood, WKNN). A continuación, se introduce una breve explicación a estos algoritmos.

6.2.2.1 Algoritmo del vecino más cercano (Nearest Neighborhood NN)

Es el método más básico en los algoritmos de coincidencia. Su implementación implica comparar las señales recibidas por un dispositivo desde n puntos (por ejemplo, Beacons) con la información de huellas digitales almacenada en la base de datos. El objetivo es encontrar la huella digital en base de datos que tenga la menor diferencia con las señales recibidas, lo cual determina las coordenadas del punto más cercano.

La fórmula es:

$$d_i = \sqrt{\sum_{i=1}^{n}(RSSI_i - RSSI_{j,i})^2}$$

Donde:

- $RSSI_i$ = La intensidad de la señal recibida del i-ésimo ancla en la posición actual
- $RSSI_{j,i}$ = La intensidad de la señal almacenada en la base de datos para i-ésimo ancla en la huella digital j
- n = Número de anclas desplegadas en el algoritmo (número de Beacons)

Este método determina las coordenadas del punto de huella digital más cercano en la base de datos al usuario, cuya precisión depende de la exhaustividad de la base de datos, por lo que limita la precisión y estabilidad, no puede ofrecer una precisión alta de localización.

6.2.2.2 Algoritmo de los K vecinos más cercanos (K Nearest Neighborhood, KNN)

El algoritmo de k vecinos más cercanos (KNN), es una mejora sobre el algoritmo del vecino más cercano (NN). Al igual que el NN calcula inicialmente la distancia entre la información RSSI recibida y las huellas digitales almacenadas en la base de datos. La diferencia principal está como se determina la ubicación final.

En KNN, en lugar de seleccionar solo el punto de huella digital más cercano, se calcula el promedio de las coordenadas de los K puntos de huella digital más cercanos.

La fórmula [27] es:

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$$

Donde:

- d (x, y) representa la distancia euclidiana entre los puntos x, y
- n es el número de puntos de Beacons considerados

- y_i es el valor de la i-ésima característica del vector y
- x_i es el valor de la i-ésima característica del vector x

6.2.2.3 Algoritmo de los K vecinos más cercanos ponderados (Weighted K Nearest Neighborhood, WKNN)

El método de vecino más cercano ponderado (WKNN) [28] es una mejora de vecino más cercano (KNN). En entornos reales, la información obtenida de Beacons más cercanos son más precisas que las obtenidas de Beacons más lejanos. Por lo tanto, en el método de vecino más cercanos no es preciso simplemente promediar las coordenadas de estos puntos.

En este algoritmo de los K vecinos más cercanos ponderados, se asigna pesos más altos a los puntos más cercanos al punto objetivo y luego calcular un promedio ponderado de las coordenadas de los K puntos seleccionados como la predicción final de las coordenadas.

La fórmula [25] es:

$$(x, y) = \sum_{i=i}^{K} w_i(xi.yi)$$

Donde:

- (x, y) son las coordenadas estimadas
- K es el número de Beacons más cercanos considerados
- w_i es el pero asignado al i-ésimo Beacon
- $(xi.yi)$ son las coordenadas del i-ésimo Beacon

Cálculo del peso w_i se calcula mediante la fórmula:

$$w_i = \frac{1}{d_i + \epsilon}$$

Donde:

- d_i es la distancia del i-ésimo Beacon al usuario
- ϵ es un valor pequeño positivo para evitar la división por cero

6.2.3 Resultado de la comparación

Tras realizar el estudio sobre los distintos algoritmos de cálculo del posicionamiento interior, se optó por la trilateración por varias razones. Este algoritmo no requiere una sincronización de tiempo muy precisa, como el algoritmo de diferencia de tiempo de llegada (TDOA), ni la instalación de varios receptores de ultrasonido, como el algoritmo de ángulo de llegada (AOA). Además, tanto algoritmo de diferencia de tiempo de llegada (TDOA) como Algoritmo de ángulo de llegada (AOA) no son adecuados para la infraestructura basada en Beacons, que es la tecnología utilizada en este proyecto.

Por otro lado, los métodos de Fingerprinting se basan en la construcción de mapas de huellas digitales que relacionan la intensidad de la señal con ubicaciones conocidas. Si bien estos métodos pueden ser robustos, requieren una fase inicial intensiva de mapeo y pueden ser complejos de implementar y mantener. En un entorno como el de Inditex, donde las tiendas pueden variar y requieren actualizaciones regulares, la complejidad de mantener estos mapas de huellas digitales no es ideal.

Por lo tanto, la decisión de optar por la trilateración como método de cálculo del posicionamiento interior es adecuada para este caso. Es compatible con la tecnología de Beacons, ofrece facilidad en la configuración y calibración, y su mantenimiento es sencillo. Esto asegura una solución eficiente y adaptable para aplicar en distintos entornos de tiendas, cumpliendo con los requisitos específicos de este proyecto.

6.2.4 Suavización de la fluctuación de señal

La suavización de las fluctuaciones de señales es un proceso muy necesario en cuando realiza un cálculo de la posición en interior. Este proceso evita saltos de señal que influye en resultados. Consiste en aplicar técnicas y métodos para reducir las variaciones abruptas de las señales, mejorando así la estabilidad de las señales.

Existen diversas técnicas para suavizar señales. Entre ellas, se destacan la Media Móvil Simple (SMA), la Media Móvil Exponencialmente Ponderada (EMA) y el Filtrado de Kalman. A continuación, se presenta un breve estudio de cada una:

- **Media Móvil Simple (SMA)** [29]: La SMA asigna el mismo peso a todos los valores dentro del periodo de cálculo, lo que puede resultar en un retraso significativo en la detección de cambios. Este método es sencillo de implementar, pero su falta de sensibilidad a los cambios recientes lo hace menos ideal para sistemas dinámicos.
- **Media Móvil Exponencialmente Ponderada (EMA)** [30]: A diferencia de la SMA, la EMA asigna mayor peso a los datos más recientes, lo que permite que el promedio responda más rápidamente a los cambios en la señal. Esta característica hace que la EMA sea más adecuada para capturar cambios rápidos y proporcionar una salida suavizada y estable.

  $$EMA = \alpha \cdot Valor_{nuevo} + (1 - \alpha) \cdot EMA_{anterior}$$

  Donde:

  - α es el factor de suavización ($0 < \alpha < 1$)
  - $Valor_{nuevo}$ es el Valor de RSSI nuevo en nuestro caso
  - $EMA_{anterior}$ es el valor de EMA calculado anteriormente

- **Filtrado de Kalman** [31]: Este es un algoritmo de estimación óptima utilizado para inferir el estado de un sistema dinámico a lo largo del tiempo, especialmente cuando el sistema está afectado por ruido y otras incertidumbres. Y es más complejo en su implementación.

Como resultado de la comparación, se elige la EMA debido a su capacidad para reflejar rápidamente las nuevas tendencias, ofreciendo una solución eficiente para el procesamiento de señales en un entorno dinámico.

6.3 Algoritmo para calcular la ruta (Algoritmo de Dijkstra)

Para el cálculo de la ruta se ha utilizado el algoritmo de Dijkstra [32] . Este algoritmo es un algoritmo clásico de búsqueda de caminos en grafos, utilizado para encontrar el camino más corto entre un nodo de origen y todos los demás nodos en un grafo ponderado y dirigido. Desarrollado por Edsger W. Dijkstra en 1956, es uno de los algoritmos más utilizados en aplicaciones de redes, como la optimización de rutas en sistemas de navegación.

Las características principales del algoritmo de Dijkstra son las siguientes:

1. **Eficiencia en la Búsqueda del Camino Más Corto:** El algoritmo de Dijkstra encuentra el camino más corto desde un nodo de origen hacia todos los demás nodos en un grafo ponderado y dirigido. Utiliza una estrategia de búsqueda voraz (greedy), expandiendo gradualmente el conjunto de nodos visitados y seleccionando la ruta más corta en cada paso.
2. **Grafos Ponderados:** El algoritmo de Dijkstra trabaja con grafos que tienen aristas con pesos no negativos, es decir, cada arista tiene asociado un valor numérico que representa la distancia o el costo entre dos nodos. Puede ser aplicado tanto a grafos dirigidos como no dirigidos.
3. **Estructuras de Datos Auxiliares:** Para llevar a cabo la búsqueda eficiente del camino más corto, el algoritmo de Dijkstra utiliza estructuras de datos como colas de prioridad (priority queues) o montículos (heaps) para seleccionar el siguiente nodo a explorar de manera eficiente, priorizando los nodos con las distancias más cortas conocidas.
4. La complejidad temporal del algoritmo de Dijkstra depende del número de nodos (|V|) y el número de aristas (|E|) en el grafo. En el peor caso, su complejidad es O ((|V| + |E|) * log |V|) utilizando un montículo binario como estructura de datos auxiliar.

7. Implementación de la solución

En este capítulo, se explicará la implementación del proyecto atendiendo a varios puntos claves.

En primer lugar, se explica la estructura Model View Intent (MVI), donde se detalla cómo se organiza el código para mantener una separación entre la lógica de negocio y la interfaz de usuario. A continuación, se aborda el manejo de los controles de permisos necesarios, tales como bluetooth, localización y los dispositivos cercanos. Se continúa con la integración de servicios de mapas y manejo de archivos GeoJson para la representación de datos geoespaciales. Luego, se describe la implementación de la funcionalidad de cálculo de rutas, utilizando el algoritmo de Dijkstra. Finalmente, se aborda la implementación de Beacons, explicando cómo se utilizan estos dispositivos para mejorar el posicionamiento interior.

7.1 Implementación de Model View Intent

La implementación de la arquitectura Model View Intent (MVI) se llevó a cabo de manera que dentro de un archivo “viewmodel” se definen los tres componentes principales: Event, State y Effect.

- **Event:** Los eventos representan las intenciones y describen las acciones del usuario o los eventos del sistema. Actúan como señales que informan al modelo (Model) de que algo ha ocurrido y necesita ser manejado.
- **State:** El estado refleja los datos actuales y la condición de la interfaz de usuario. Es un modelo que contiene toda la información necesaria para representar la vista en cualquier momento dado. Mantiene la interfaz de usuario sincronizada con los datos subyacentes.
- **Effect:** Los efectos son acciones que afectan la interfaz de usuario, pero no modifican directamente el estado del modelo. Se utilizan para manejar tareas que deben ser ejecutadas por la vista.

Otra función muy importante que destaca en esta arquitectura es ‘handleEvent’. Esta función actúa como el listener de los eventos que ocurre en la aplicación. Dependiendo del tipo de evento capturado, ‘handleEvent’ ejecuta diferentes bloques de código. Esta función permite gestionar las respuestas de las acciones del usuario de manera centralizada.

Lo anteriormente descrito forma parte del archivo "viewmodel". En cuanto a la parte del "screen", su función principal es definir las interfaces utilizando Jetpack Compose. Además, esta sección se encarga de capturar las acciones del usuario y comunicarlas al "viewmodel".

La interfaz gráfica de usuario de la aplicación desarrollada tiene dos pantallas principales, “Home” y “Other.” “Home” es la pantalla principal donde se muestra la interfaz del mapa, mientras que “Other” es la pantalla donde se muestra la lista de los artículos (prendas). Las Figuras 18 y 19 muestran estas dos pantallas.

La organización de fichero sigue una estructura modular, donde cada pantalla tiene su propio conjunto de archivos. Esto significa que “Home” cuenta con su propio archivo “HomeViewModel” para manejar la lógica y otro “HomeScreen” para la interfaz de usuario. De igual manera la pantalla “Other” tiene sus dos archivos, “OtherViewModel” y “OtherScreen”

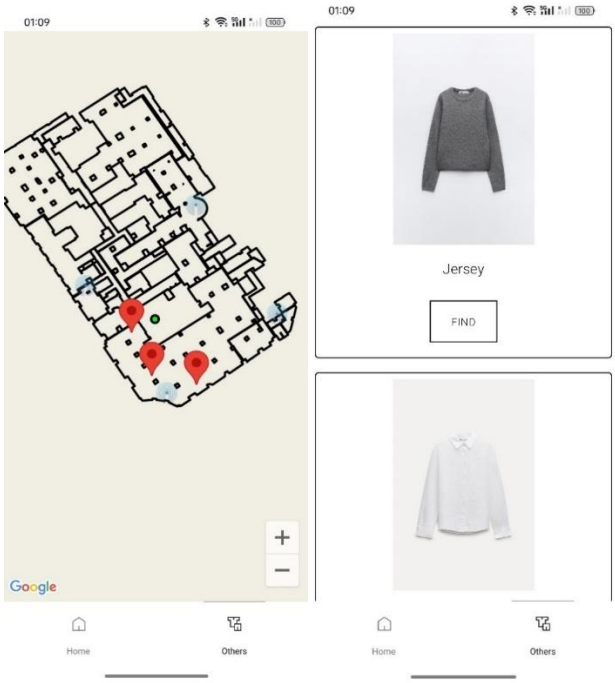

Figura 18, 19: (izquierda 18, derecha 19) Interfaces de la aplicación

7.2 Manejo de Permisos

Al tratarse de un proyecto de localización, es crucial gestionar los permisos dentro del archivo “AndroidManifest.xml” para un correcto funcionamiento. Los permisos necesarios son: acceso a la ubicación, Bluetooth y dispositivos cercanos, como se ilustra en la Figura 20.

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN"/>
<uses-permission android:name="android.permission.BLUETOOTH_SCAN"/>
<uses-permission android:name="android.permission.BLUETOOTH_CONNECT"/>
<uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION"/>
```

Figura 20: Listado de permisos en “AndroidManifest.xml”

Una vez definido, hay que gestionar correctamente el ciclo de permisos. Este ciclo consta de los siguientes pasos:

1. Verificacion de permisos: La aplicación verifica si ya tiene los permisos necesarios antes de realizar una operación que los requiera
2. Solicitud de permisos: Si la aplicación no tiene los permisos necesarios, solicita al usuario que los conceda.
3. Respuesta del usuario: El usuario decide si concede o deniega los permisos solicitandos.
4. Manejo de respuesta: La aplicación debe manejar la respuesta del usuario. Si el usuario concede los permisos, la aplicación puede proceder con una funcionalidad correcta. En el caso de denegación, la aplicación explica la necesidad de estos permisos.

7.3 Integración del Mapa y GeoJson

Para integrar la API de Google Maps en un proyecto, el primer paso es crear una cuenta en Google Cloud Console y configurar un nuevo proyecto. Dentro de este proyecto, será necesario activar las API de Google Maps relevantes. Posteriormente, se procede a generar una clave de API en la sección de credenciales de Google Cloud Console. Esta clave de API debe ser registrada en el archivo "AndroidManifest.xml". Además, es esencial agregar las dependencias necesarias en el archivo "build.gradle.kts". En la Figura 21 muesta el listado de las dependencias utilizadas. Entre ellas también esta la de GeoJson.

```
implementation(libs.play.services.maps)
implementation(libs.android.maps.utils)
implementation(libs.accompanist.permissions)
implementation(libs.play.services.location)
implementation(libs.kotlinx.coroutines.android)
implementation(libs.kotlinx.coroutines.play.services)
implementation(libs.maps.compose)
implementation(libs.okhttp)
implementation(libs.gson)
implementation(libs.jgrapht.core)
```

Figura 21: Listado de dependencias

Una vez integrada la API de Google Maps, es necesario realizar una configuración adecuada del mapa. Esta API ofrece diversas opciones de configuración que permiten ajustar el mapa según las necesidades específicas del proyecto. En este caso, se ha optado por configurar el mapa en un tipo de mapa normal, así como activar funciones como los botones de zoom, la brújula y la barra de herramientas (mapToolbar).

Aunque la API de Google Maps ofrece la capacidad de obtener la posición del usuario mediante GPS, esta solución no es adecuada para la localización en interiores. Además, Google Maps no proporciona la opción de recalcular la posición obtenida por su API, lo que limita su utilidad en este contexto. Por estas razones, no se ha optado

por utilizar esta solución ni se ha aprovechado su funcionalidad para la localización interior.

No obstante, Google Maps ofrece la posibilidad de integrar una capa de GeoJson de manera sencilla, lo que facilitó la incorporación de dicha capa en el proyecto. Esta funcionalidad simplifica el proceso de visualización de datos geoespaciales. A continuación, en la Implementación de rutas (sección 7.4) se detallará el uso de archivos GeoJson para el desarrollo de rutas.

7.4 Implementación de rutas

En la implementación de rutas, se ha llevado a cabo una transformación de una capa de GeoJson a un grafo. Inicialmente, se define una capa de GeoJson donde se especifican las rutas posibles. En la Figura 22 se muestra la capa de GeoJson que representa estas rutas.

Figura 22: GeoJson de rutas

Gracias a que GeoJson tiene un comportamiento similar al de un grafo, donde los “Feature” de un GeoJson son puntos con coordenadas o conexiones de diferentes puntos. Comparando con un grafo, que son conexión de vértices y aristas.

La transformación de GeoJson a grafo presentó ciertas dificultades. Aunque visualmente una ruta puede parecer una línea continua en el mapa, en realidad está compuesta por múltiples vértices conectados entre sí. Esto complicó la transformación, ya que, al ser un archivo creado manualmente, no siempre tenía líneas perfectamente conectadas. Durante la conversión, los vértices no se conectan automáticamente, sino que se reconocen como vértices individuales o líneas no conectadas. Por ello, fue necesario implementar una función para la detección de vértices muy cercanos y tratarlos como vértices conectados.

Pintar con numerosos vértices para formalizar la ruta es más complicado de implementar y requiere más trabajo a la hora de pintar, pero permite que la ruta sea

más precisa y que el movimiento del usuario entre vértices sea más suave, evitando cambios bruscos.

Otro punto interesante que hay que destacar es la integración de la posición del usuario dentro del grafo de rutas. Dado que la posición del usuario es dinámica y no estática, el proceso implica primero añadir la posición del usuario como un vértice normal al grafo y conectarlo al vértice más cercano del grafo. Después, cada vez que se actualiza la posición del usuario, se elimina el vértice y la conexión previamente creados, para luego agregar la posición actualizada como un nuevo vértice en el grafo.

Finalmente, para calcular la ruta más corta utilizando el algoritmo de Dijkstra, se utilizó la API de jGraph, que incluye un método `DijkstraShortestPath` con los parámetros de entrada (grafo, nodo de inicio, nodo de destino). Esta función devuelve la ruta más corta entre los nodos especificados.

7.5 Implementación de Beacons

A continuación, se presenta una explicación sobre Beacons utilizados y se detalla cómo se implementó los Beacons dentro del desarrollo del proyecto. Los Beacons utilizados son fabricados por la empresa BlueUp, el modelo es BlueBeacon Mini.

BlueUp ofrece la aplicación BlueBeacon Manager, una herramienta poderosa y fácil de usar diseñada para configurar y administrar los dispositivos BlueBeacon, incluido el BlueBeacon Mini. Esta aplicación proporciona a los usuarios un acceso intuitivo a una amplia variedad de opciones de configuración y personalización, lo que les permite adaptar los dispositivos según sus necesidades específicas.

Una característica destacada es la capacidad de simular el funcionamiento tanto como iBeacon, Edystone o Quuppa, lo que brinda una flexibilidad adicional en la implementación de diferentes soluciones.

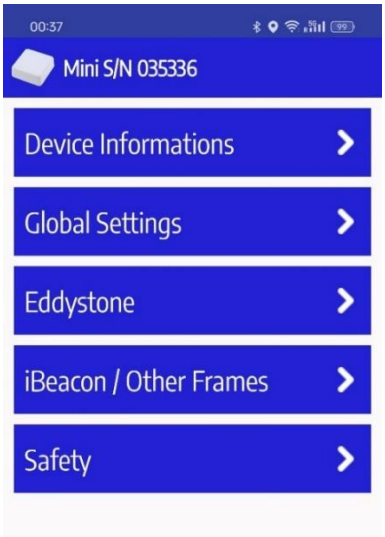

Figura 23: Página de configuración 1 de Beacon

En la figura 23, se puede observar que una vez conectado al Beacon, se proporciona una amplia flexibilidad en términos de configuración. Además, ofrece información detallada sobre dicho Beacon.

Entre los parámetros del Beacons que se pueden modificar se incluyen:

- UUID
- Major y Menor
- Intervalo de transmisión
- Radio Tx Power
- Calibrated RSSI at 1m

Figura 24: Página de configuración 2 del Beacon

En la figura 24 se pueden observar los parámetros configurables, cada uno con su influencia particular al ser ajustado. Es crucial destacar que no se recomienda modificar el UUID debido al riesgo de causar confusión.

Como se explicó en la sesión Posicionamiento con Beacons (sección 5.2.6), tanto el Mayor como el Menor son números enteros utilizados junto con el UUID para identificar grupos extensos de Beacons. Esta funcionalidad resulta especialmente útil en

entornos con numerosos Beacons cercanos, ya que permite filtrar estos dispositivos al ser detectados, mejorando así la precisión y eficiencia del sistema al reducir datos no relevantes. En este caso no se realizó ningún ajuste a estos dos números ya que no resulta necesario.

Los parámetros ajustados son:

- El intervalo de transmisión está configurado en 1000 ms para evitar un consumo excesivo de recursos. Actualizar con mayor frecuencia los valores del Beacon incrementaría los costes y la carga en el dispositivo móvil. Por otro lado, establecer un intervalo demasiado largo podría retrasar la actualización de la posición del usuario, afectando negativamente la precisión y la experiencia del usuario.
- El Radio Tx Power puede ajustarse en un rango que va desde 4 dBm hasta -40 dBm. Un valor más negativo indica una potencia de transmisión más baja del Beacon, lo que reduce su alcance. Es importante ajustar este valor según el entorno específico para optimizar el consumo de energía y la cobertura necesaria. En este caso particular, dado que el lugar de experimento no es una zona ruidosa ni muy extenso está configurada a -12dBm para optimizar el consumo de energía.
- El RSSI calibrado a 1 metro es un valor obtenido mediante la calibración manual del Beacon, tomando múltiples mediciones de RSSI a una distancia de un metro y calculando un promedio. En este caso se ha llegado con un resultado de -75dBm

Una vez configurados los Beacons, se inicia la implementación y escaneo dentro de la aplicación. Android ofrece un BluetoothAdapter nativo para realizar esta tarea, permitiendo establecer un filtro basado en las direcciones MAC de los Beacons de interés para excluir información de dispositivos Bluetooth irrelevantes.

Los resultados de escaneo son gestionados por un método llamado ‘scanCallback’, el cual maneja la información obtenida, incluyendo el RSSI de cada Beacon. Esta información es crucial para calcular la distancia del usuario a los diferentes Beacons. Se ha observado una fluctuación en las señales durante el proceso de escaneo, lo cual puede causar picos de cambios bruscos no deseados. Como respuesta a este problema, se implementó una función diseñada para mejorar la estabilidad de las señales, tal como se explicó en la sección suavización de la fluctuación de señal (sección 6.2.4).

Antes de implementar el cálculo de posicionamiento interior, identificamos varios puntos críticos. Uno de ellos fue la necesidad de obtener las coordenadas exactas de los Beacons para realizar cálculos precisos. Sin embargo, en ese momento no encontramos herramientas disponibles para obtener estas coordenadas de manera efectiva.

Además, Google Maps tiene un límite en el nivel de zoom, lo que impide alcanzar el tamaño de zoom deseado.

También se identificó como problema la transición desde el escenario inicial de experimentación en el interior de una casa hacia la simulación de un entorno similar al de una tienda de Zara. Esta solicitud representó un desafío significativo debido a la falta de acceso a muchas medidas y detalles específicos necesarios para una simulación precisa.

Como resultado, para llevar a cabo este cálculo se utilizó el método Algoritmo de los K vecinos más cercanos (WKNN) mencionado en la sección Algoritmo de los K vecinos más cercanos ponderados (sección 6.2.2.3). Aunque su precisión no es muy alta por falta de una base digital, los resultados fueron mejores que los obtenidos con GPS.

8. Pruebas

Existen numerosas pruebas en un proyecto de software que permiten evaluar la calidad, el rendimiento y la funcionalidad del sistema desarrollado. Sin embargo, en este proyecto solo se realizaron pruebas unitarias por falta de tiempo.

En el contexto del desarrollo de software, las pruebas unitarias desempeñan un papel fundamental al validar el comportamiento de métodos específicos dentro de componentes individuales. En este caso, se han realizado pruebas unitarias para verificar la funcionalidad de los métodos en la clase “HomeViewModel”.

Estas pruebas se llevaron a cabo utilizando JUnit en conjunto con Mockk, una biblioteca especializada para Kotlin que permite la creación de mocks que simulan el comportamiento de objetos reales durante el proceso de testing (sección 5.1.9). Los mocks son esenciales porque permiten probar el funcionamiento de componentes de manera controlada, sin depender de recursos externos o dependencias complejas.

A continuación, en la Figura 25 se muestra el inicio de las pruebas. El método setup(), anotado con “@Before”, se ejecuta antes de cada prueba. El método tearDown(), anotado con “@After”, se ejecuta después de cada prueba. Se utiliza el método clearAllMocks() para realizar una limpieza de los mocks creados, asegurando un estado limpio entre pruebas.

Además, los nombres de las pruebas están encapsulados entre comillas simples (‘’) para mejorar la legibilidad y comprensión de cada escenario probado. Los nombres descriptivos son cruciales, ya que facilitan la comprensión inmediata del propósito y el resultado esperado de cada prueba.

La estructura de una prueba generalmente se divide en tres partes y sigue el modelo GIVEN-WHEN-THEN. En este modelo, tal como muestra la Figura 25:

- GIVEN representa la condición previa
- WHEN describe la acción realizada
- THEN indica el resultado esperado

```kotlin
@Before
fun setup() {
    mockMap = mockk(relaxed = true)
    viewModel = HomeViewModel()
}

@After
fun tearDown() {
    clearAllMocks()
}

@Test
fun `test OnMyLocationClicked event`() {
    // GIVEN
    val location = LatLng( latitude: 40.423652,  longitude: -3.710821)
    val event = HomeViewModel.Event.OnMyLocationClicked(location)
    // WHEN
    viewModel.handleEvent(event)
    // THEN
    assertEquals(location, viewModel.currentState.userLocation)
}
```

Figura 25: Pruebas Unitarias de HomeViewModel

9. Conclusión

En este capítulo se realiza un análisis sobre si se han cumplido los objetivos propuestos al inicio del trabajo. También, se describen los problemas que se han encontrado y cómo se han solucionado. Además, se resume qué se ha aprendido con este TFG. Por último, se indica la relación del trabajo realizado con los conceptos aprendidos durante la carrera de grado de Ingeniería Informática.

9.1 Objetivos cumplidos

Los objetivos planteados en el apartado de objetivos (sección 1.3) se cumplieron de manera casi perfecta, como se muestra en el apartado de implementación (sección 7). La mayoría de los objetivos se lograron satisfactoriamente, incluyendo la integración del mapa, la actuación de posición en tiempo real, la navegación con rutas, el uso de Beacons, la visualización de puntos de interés, etc.

El único objetivo que no se cumplió de manera perfecta fue el cálculo de la posición interior del usuario. Inicialmente, se planeó usar trilateración, pero la dificultad para obtener las coordenadas exactas de los Beacons y la necesidad de simular un escenario real de ZARA complicaron la implementación.

El cálculo de la distancia de usuario a un solo Beacon fue sencillo aplicando la fórmula mencionada en el algoritmo de trilateración (sección 6.2.1.1). Sin embargo, combinar los resultados de varios Beacons resultó más complejo de lo anticipado. Finalmente, se optó por el algoritmo de los K vecinos más cercanos (WKNN), que, aunque es un tipo de Fingerprinting y requiere una base de huellas digitales para una precisión máxima, en este caso proporcionó una solución con precisión media.

9.2 Aprendizaje del proyecto

Durante el desarrollo de este proyecto, tuve que familiarizarme con tecnologías completamente nuevas para mí. La mayoría de las tecnologías empleadas, desde el lenguaje de programación hasta los Beacons y el modelo de arquitectura Model View Intent (MVI), eran conceptos desconocidos previamente.

Al inicio del desarrollo, dediqué entre dos y tres semanas completas a familiarizarme con estas tecnologías, especialmente con Kotlin y MVI. Considero crucial mantener un código limpio y bien estructurado desde el principio, a pesar de que es inevitable realizar modificaciones en el futuro. Es preferible planificar bien desde el inicio que comenzar a desarrollar sin una estructura clara.

Aprender Kotlin no fue tan complicado como familiarizarme con MVI. En mi opinión, introducirme en Kotlin no fue una tarea ardua, ya que este lenguaje es bastante similar a Java. Sin embargo, la estructura de MVI resultó ser más desafiante.

La implementación de la ruta también presentó su propio desafío. En un entorno interior, no existen rutas predefinidas, por lo que tuve que diseñar una solución que involucrara la conversión de geojson a grafos para su implementación.

Asimismo, la implementación de los Beacons también fue un gran desafío, no por su estructura, que es relativamente fácil de entender, sino por el cálculo necesario para obtener resultados estables y precisos.

En resumen, este proyecto me permitió dominar tecnologías nuevas como Kotlin y el modelo MVI, enfrentar desafíos como la implementación de Beacons y la generación de rutas en entornos interiores, tener contactos con las nuevas tecnologías, estudiar diferentes tecnologías y algoritmos de cálculo interior. Estoy satisfecha con los resultados y segura de que las habilidades adquiridas serán invaluables en futuros proyectos técnicos.

9.3 Relación con los estudios cursados

La formación en la rama de ingeniería de informática y especializada en la rama de ingeniería de Software, ha sido fundamental para el desarrollo de este proyecto.

Es una rama que enseña a los estudiantes a cómo desarrollar un proyecto respetando el ciclo de desarrollo. Enseña desde la planificación hasta el mantenimiento. Además, la forma de enseñanza es muy práctica, gracias a este modelo de enseñanza nos ha proporcionado capacidad de desarrollar y enfrentar problemas con más confianza y tranquilidad.

Durante la carrera, he encontrado que los conceptos aprendidos emergen de manera sorprendente en situaciones específicas. Aunque inicialmente podría no haber comprendido completamente su utilidad, cada área de conocimiento adquirida se ha revelado muy valiosa en proyectos como este.

En este desarrollo en concreto:

- La metodología de trabajo ágil enseñada y practicada durante los dos últimos cursos con la asignatura Proyecto de Software (4A) y Proceso de Software (3B).
- Análisis del problema, análisis de requisitos funcionales, requisitos no funcionales y caso de uso. Son conceptos introducidos desde la asignatura Ingeniería de Software (2B) y posteriormente se profundizan en cursos de tercero y cuarto en la rama de Software, como en la asignatura Análisis y especificación de Requisitos (4A).
- La importancia de mantener un código limpio y bien estructurado se ha enfatizado en la asignatura de Diseño de Software (3B).
- El algoritmo de Dijkstra se estudió en primer curso con la asignatura Matemática discreta (1A).
- El conocimiento de Git se ha adquirido en la asignatura Mantenimiento y evolución de software (4A).
- La sólida base en Java, adquirida desde Introducción a la Informática y Programación (1AB), me ha permitido aprender otros lenguajes como Kotlin con facilidad.

En conclusión, pienso que el grado de Ingeniería Informática en la Universidad Politécnica de Valencia proporciona un amplio conocimiento en informática, capacita

a los estudiantes para estar preparados y les permite elegir entre diversos caminos según sus preferencias personales.

9.4 Trabajos Futuros

Como trabajo futuro, existe un gran potencial para mejorar la precisión de la localización en interiores utilizando Beacons, dado que aún no se ha alcanzado el nivel de precisión deseado. Además, una posible implementación futura incluiría integrar la aplicación con una base de datos para adquirir datos sobre la ubicación de los artículos en venta.

Una opción que se consideraría cambiar es la exclusión del uso de la API de Google Maps, que, aunque ofrece una gran capacidad y una integración sencilla con capas GeoJson, está principalmente diseñada para la localización y navegación exterior.

No resultó especialmente útil y no alcanzó un resultado ideal por su limitación en el nivel de Zoom, lo que impide ampliar el mapa hasta un nivel satisfactorio. Además, carece de la capacidad de combinar cálculo de posición con GPS junto con los de Beacons.

Es importante destacar que el posicionamiento interior es una tecnología muy útil que tiene aplicaciones potenciales en diversas áreas, más allá de los entornos comerciales de venta

Bibliografía

[1] Atlassian, «Gestión ágil de proyectos: qué es y como empezar,» [En línea]. Available: https://www.atlassian.com/es/agile/project-management.

[2] Atlassian, «¿Qué es un tablero de kanban?,» [En línea]. Available: https://www.atlassian.com/es/agile/kanban/boards.

[3] Atlassian, «Flujo de trabajo de Git-Flow,» [En línea]. Available: https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow.

[4] «Android Developers,» [En línea]. Available: https://developer.android.com/studio/intro?hl=es-419.

[5] «Kotlin,» JetBrains, [En línea]. Available: https://kotlinlang.org/.

[6] mattwojo, B.-M. v.-m. y D. , «Microsft Learn,» [En línea]. Available: https://learn.microsoft.com/es-es/windows/android/native-android.

[7] A. Developer, Google, [En línea]. Available: https://developer.android.com/codelabs/jetpack-compose-basics?hl=es-419#0.

[8] «Google Maps Platform,» [En línea]. Available: https://developers.google.com/maps/documentation/android-sdk?hl=es-419.

[9] F. P. Serrano, «MappingGIS,» 17 Julio 2018. [En línea]. Available: https://mappinggis.com/2018/03/json-y-geojson-en-el-mundo-gis/.

[10] Accompanist, [En línea]. Available: https://google.github.io/accompanist/permissions/.

[11] B. P. G. a. J. Cordero, «Indoor positioning system using Beacon technology,» de *15th Iberian Conference on Information Systems and Technologies (CISTI)*, Seville, Spain, 2020.

[12] «JUnit,» [En línea]. Available: https://junit.org/junit5/.

[13] Mockk, [En línea]. Available: https://mockk.io/.

[14] Y. L. Y. C. L. a. A. P. P. L. M. Ni, «LANDMARC: indoor location sensing using active RFID,» de *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*, Fort Worth, TX, USA, 26-26 March 2003.

[15] X. ChunLu, *Research and Implementation of Bluetooth Indoor Positioning System,* Hangzhou: Hangzhou Dianzi University, 2018.

[16] A. Alarifi, A. A.-S. M. A. y A. A. , *Ultra Wideband Indoor Positioning Technologies: Analysis and Recent Advances,* Saudi Arabia, 2016.

[17] M. E. A. H.-S. a. A. V. J. Choliz, «Comparison of Algorithms for UWB Indoor Location and Tracking Systems,» de *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)*, Budapest, Hungary, 2011.

[18] M. Compte, *Posicionamiento Indoor,* 2018 Jul 09.

[19] SITUM, 24 Febrero 2022. [En línea]. Available: https://situm.com/es/blog/posicionamiento-en-interiores/sistemas-de-localizacion-en-interiores/.

[20] M. Y. Satriawan, «Medium,» 3 Aug 2023. [En línea]. Available: https://medium.com/@satriawaaan/mastering-the-mvi-architecture-a-comprehensive-guide-28b2cafee1c9.

[21] A. Cejas y A. C. , *Algotirmos avanzados de posicionamiento en interiores utiliza ndo la combinacion de distintos tipos de sensores,* Buenos Aires: Univer sidad Nacional del centro de la Provincia de Buenos Aires, 2019, Abril.

[22] L. Yong, J. Y. y W. C. , «Illustration of trilateration,» [En línea]. Available: https:// www.researchgate.net/publication/324061572_An_improved_geometric_ algorithm_for_indoor_localization.

[23] Nanjing Woxu Wireless, 23 Feb 2018. [En línea]. Available: https://www.uwbaid e.com/news/uwb-localization-techniques-tof-and-tdoa-53317533.html.

[24] G.-I. Jee, J. L. k. y C. G. P. , 2008. [En línea]. Available: https://www.researchgat e.net/figure/TDOATime-Difference-of-Arrival-Method_fig1_224354814.

[25] Y. Zhang, «Research on improvement of indoor locationalgorithm based on iBea con,» Master Thesis Susmitted to University of Electronic Science and Te chnology of China, 2019.

[26] Minas, «Ilustración de algoritmo AOA,» [En línea]. Available: https://www.minew. com/es/angle-of-arrival-minew-stretches-out-its-antennae-to-higher-preci sion-positioning/.

[27] «IBM,» [En línea]. Available: https://www.ibm.com/es-es/topics/knn.

[28] Z. J. a. W. Honglan, «"An Improved WKNN Algorithm Using in Indoor Positionin g,» de *6th International Conference on Information Science and Control Engineering (ICISCE)*, Shanghai, China, 2019.

[29] MathWorks, [En línea]. Available: https://es.mathworks.com/help/signal/ug/signa l-smoothing.html.

[30] L. Llamas, 24 Mar 2017. [En línea]. Available: https://www.luisllamas.es/arduino-paso-bajo-exponencial/.

[31] KalmanFilter.NET, [En línea]. Available: https://www.kalmanfilter.net/.

[32] A.Rodríguez, «Grafos - Software para la construcción, edición y análisis de graf os,» [En línea]. Available: https://arodrigu.webs.upv.es/grafos/doku.php?i d=algoritmo_dijkstra.

Objetivos de desarrollo sostenible

En la tabla 8 se presenta la relación entre el proyecto realizado y los 17 objetivos de desarrollo sostenible (ODS).

Objetivos de Desarrollo Sostenibles	**Alto**	**Medio**	**Bajo**	**No Procede**
ODS 1. **Fin de la pobreza.**				**X**
ODS 2. **Hambre cero.**				**X**
ODS 3. **Salud y bienestar.**				**X**
ODS 4. **Educación de calidad.**				**X**
ODS 5. **Igualdad de género.**				**X**
ODS 6. **Agua limpia y saneamiento.**				**X**
ODS 7. **Energía asequible y no contaminante.**				**X**
ODS 8. **Trabajo decente y crecimiento económico.**	**X**			
ODS 9. **Industria, innovación e infraestructuras.**	**X**			
ODS 10. **Reducción de las desigualdades**.				**X**
ODS 11. **Ciudades y comunidades sostenibles**.				**X**
ODS 12. **Producción y consumo responsables.**		**X**		
ODS 13. **Acción por el clima.**				**X**
ODS 14. **Vida submarina.**				**X**
ODS 15. **Vida de ecosistemas terrestres.**				**X**
ODS 16. **Paz, justicia e instituciones sólidas.**				**X**
ODS 17. **Alianzas para lograr objetivos.**				**X**

Tabla 8: Objetivos de desarrollo sostenible (ODS)

Los objetivos de desarrollo sostenible (ODS) son una colección de 17 metas globales establecidas por las Naciones Unidas para abordar los desafíos más urgentes del mundo. El proyecto desarrollado está alineado con los siguientes objetivos:

- **ODS 8 – Trabajo docente y crecimiento económico:** Facilitando la navegación interior, el proyecto permite guiar a los clientes de manera más rápida y efectiva, liberando recursos humanos para tareas estratégicas y creando oportunidades de empleo más productivas.
- **ODS 9 – Industria, innovación e infraestructura**: Mediante el uso de tecnología innovadora, la aplicación mejora la infraestructura digital en espacios interiores. Esto no solo facilita la navegación para los usuarios, sino que

también fomenta la innovación en la gestión de espacios comerciales y otros entornos.

- **ODS 12 – Producción y consumo responsable:** Contribuye a optimizar el uso de recursos dentro de espacios interiores al mejorar la gestión de inventarios y reducir el tiempo que los usuarios dedican a buscar productos. Esto reduce el desperdicio de recursos y promueve prácticas de consumo más responsables.