



UNIVERSIDAD DE MÁLAGA



Graduado en Ingeniería Informática

Aplicación móvil basada en geofencing para
navegación en interiores

Mobile application based on geofencing
for indoor navigation

Realizado por
Federico González-Vico Berral

Tutorizado por
Vicente Manuel Arévalo Espejo

Departamento
Ingeniería de Sistemas y Automática

MÁLAGA, septiembre de 2024



UNIVERSIDAD
DE MÁLAGA



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADO EN INGENIERÍA INFORMÁTICA

**Aplicación móvil basada en geofencing para
navegación en interiores**

**Mobile application based on geofencing
for indoor navigation**

Realizado por
Federico González-Vico Berral

Tutorizado por
Vicente Manuel Arévalo Espejo

Departamento
Ingeniería de Sistemas y Automática

UNIVERSIDAD DE MÁLAGA
MÁLAGA, SEPTIEMBRE DE 2024

Fecha defensa: septiembre de
2024

Abstract

This Final Degree Project presents a mobile application that employs geofencing technology and beacon-based localization to facilitate indoor navigation and regulate access in closed environments. This tool, which has gained particular relevance during the COVID-19 pandemic, establishes virtual zones on a map using beacons and notifies the user when they enter or leave these areas. The application will offer functionalities for precise navigation within buildings. Expected outcomes include a significant improvement in the accessibility of these environments and an optimization in space management through the monitoring of people flow.

For its development, an agile methodology will be followed to ensure the flexibility and adaptability of the project. The deliverables will include a functional mobile application, detailed documentation on the design and evolution of the system, as well as a user manual for the correct use of the application. In summary, this geofencing application is expected to generate innovative and beneficial results in terms of accessibility, security, and space management in indoor environments.

Keywords: beacons, geofencing, BLE, indoor navigation

Resumen

Este Trabajo de Fin de Grado presenta una aplicación móvil que emplea la tecnología de geofencing y localización mediante balizas para facilitar la navegación en interiores y regular el acceso en entornos cerrados. Esta herramienta, que ha cobrado especial relevancia durante la pandemia del COVID-19, establece zonas virtuales en un mapa gracias al uso de balizas y notifica al usuario cuando entra o sale de estas áreas. La aplicación ofrecerá funcionalidades para la navegación precisa dentro de edificios. Se esperan resultados tales como una mejora significativa en la accesibilidad a dichos entornos y una optimización en la gestión de espacios mediante la monitorización del flujo de personas.

Para su desarrollo, se seguirá una metodología ágil para la implementación del software, garantizando así la flexibilidad y adaptabilidad del proyecto. Los entregables incluirán una aplicación móvil funcional, documentación detallada sobre el diseño y evolución del sistema, así como un manual de usuario para una correcta utilización de la aplicación. En resumen, se espera que esta aplicación de geofencing genere resultados innovadores y beneficiosos en términos de accesibilidad, seguridad y gestión de espacios en entornos interiores.

Palabras clave: balizas, geofencing, BLE, navegación en interiores

Índice

1. Introducción	11
1.1. Conceptos iniciales	11
1.2. Objetivo	13
1.3. Fases del proyecto	14
1.3.1. Investigación y análisis de requisitos	14
1.3.2. Diseño y planificación del sistema	14
1.3.3. Desarrollo de la aplicación móvil e integración con las placas ESP32	15
1.3.4. Pruebas y depuración	16
1.3.5. Documentación y entrega del TFG	16
1.4. Contenido de la memoria	16
2. Beacons: Estructura y aplicaciones	19
2.1. Beacons (balizas) - Definición	19
2.2. Proveedores	19
2.3. Aplicaciones de las beacons	21
2.4. Estructura	22
2.5. Ventajas e inconvenientes	23
3. Flujos de trabajo	27
3.1. LaTeX y Overleaf	27
3.2. Metodologías ágiles	27
3.3. Jira	28
3.4. GitHub	29
3.5. Adaptación del proyecto	31
4. Análisis de requisitos	35
4.1. Identificación de necesidades de localización en interiores	35
4.2. Definición de requisitos del sistema	36
4.2.1. Requisitos funcionales	36

4.2.2. Requisitos no funcionales	37
5. Diseño de la aplicación	39
5.1. Introducción	39
5.2. Interacción entre el móvil y la base de datos	39
5.2.1. Proceso de detección de la posición	40
5.2.2. Transferencia de datos a la base de datos	40
5.2.3. Procesamiento en la base de datos	40
5.2.4. Respuesta del servidor	41
5.3. Configuración de las balizas	41
5.4. Descripción del hardware	42
5.5. Cálculos y aproximaciones	43
6. Implementación	51
6.1. Preparación del entorno	51
6.2. Entidades Beacon	52
6.3. Adaptadores	53
6.4. Servicios	54
6.5. Actividades	55
6.5.1. BeaconActivity	56
6.5.2. BeaconScanActivity	57
6.5.3. FAQActivity	57
6.5.4. MapActivity	58
6.5.5. MainActivity	58
6.6. Base de Datos	59
6.7. Útiles	60
6.8. Pantallas	60
6.8.1. MainActivity	60
6.8.2. BeaconActivity	61
6.8.3. BeaconScanActivity	64
6.8.4. MapActivity	65
6.8.5. FaqActivity	71

7. Pruebas y depuración	73
7.1. Casos de prueba	73
7.2. Pruebas unitarias	74
7.3. Pruebas de integración	75
7.4. Identificación y corrección de errores	79
8. Conclusiones y Líneas Futuras	81
8.1. Conclusiones	81
8.2. Futuras líneas de trabajo	82
Apéndice A. Cálculo de distancias y calibración	89
A.1. Primeras mediciones	89
A.2. Preparación de los datos	90
A.3. Prueba de predicción con A y B	91
A.4. Cálculo del coeficiente C	92
A.5. Prueba de predicción final	93
Apéndice B. Implementación del código de las balizas	95
B.1. Análisis del código	95
B.2. Librerías necesarias	95
B.3. Declaraciones globales	96
B.4. Variables para publicidad BLE y tiempo	96
B.5. UUID del Beacon	96
B.6. Configuración del Beacon	97
B.7. Configuración inicial - setup	97
Apéndice C. Manual de uso	99
C.1. Consultar las preguntas frecuentes	99
C.2. Escaneo de beacons	99
C.3. Beacons	100

C.4. Acceso al mapa	100
-------------------------------	-----

Índice de figuras

1.	Mapa del mundo elaborado en 1467. [28]	12
2.	Algunas beacons según el proveedor [3].	20
3.	Múltiples usos de beacons en una misma habitación. [13]	25
4.	Logo de Overleaf [25].	27
5.	Logo de Jira [5].	28
6.	Logo de Github [38].	29
7.	Vista de Backlog del proyecto[10].	31
8.	Cronograma de las fases del proyecto. Mes: Agosto [11].	32
9.	Vista de ramas en el repositorio GitHub del proyecto [8].	33
10.	Últimos commits de la rama <i>v1deteccionbeacons</i> en GitHub [9].	34
11.	Esquema de flujo de información entre el móvil, las balizas y la base de datos, incluyendo obstáculos.	40
12.	Captura de la base de datos [12].	41
13.	ESP-32-WROOM-32 Module Board. [6]	43
14.	Cálculo de una posición (B) mediante trilateración. [37]	46
15.	Distancia entre dos puntos en una esfera. [35]	49
16.	Esquema de clases del proyecto - Diagrama en LucidChart [23].	52
17.	<i>Permisos</i> - Solicitar acceso a Bluetooth.	55
18.	<i>MainActivity</i> - Pantalla inicial de la aplicación.	61
19.	<i>BeaconActivity 1</i> - Listado de beacons registrados vacío.	62
20.	<i>BeaconActivity 2</i> - Listado de beacons registrados, en este caso, la que coincide con la búsqueda parcial.	63
21.	<i>BeaconScanActivity</i> - Pantalla de escaneo de dispositivos BLE y beacons.	64
22.	<i>MapActivity 1</i> - Vista previa al mapa.	66
23.	<i>MapActivity 2</i> - Mapa con beacons establecidas.	67
24.	<i>MapActivity 3</i> - Mapa con balizas, ubicación del usuario e indicaciones.	68
25.	<i>MapActivity 4</i> - Mensaje lanzado al llegar a una baliza.	69
26.	<i>MapActivity 5</i> - Mensaje con la opción de detener el servicio del mapa.	70

27.	<i>FaqActivity</i> - Pantalla que contiene el listado de preguntas.	71
28.	Captura con los permisos necesarios en nRF Connect para escanear.	76
29.	Captura con la detección de una baliza en nRF Connect	77
30.	Librerías empleadas en el código.	95
31.	Declaraciones globales empleadas en el código.	96
32.	Variables empleadas en el código.	96
33.	Definición del UUID del Beacon empleado en el código.	96
34.	Configuración del Beacon empleado en el código.	97
35.	Configuración inicial (setup) usada en el código.	98

1

Introducción

1.1. Conceptos iniciales

Las Tecnologías de la Información y la Comunicación (TIC) han revolucionado la forma en que nos conectamos y compartimos información. En un mundo hiperconectado, la tecnología desempeña un papel fundamental en todos los aspectos de la vida humana. La evolución tecnológica ha permitido avances significativos en la forma en que interactuamos con nuestro entorno y entre nosotros. Una de las necesidades más esenciales que ha surgido en este contexto es la de conocer nuestra ubicación precisa en cualquier momento y lugar. Esta necesidad es impulsada tanto por razones prácticas como por la evolución de nuestra filosofía y comprensión del mundo.

Desde una perspectiva científica, se puede entender a través de varios campos del conocimiento. Primero, la influencia de la geografía y la cartografía en el desarrollo humano, ya que conocer nuestra posición en el mundo ha sido crucial para la exploración, el comercio o la guerra. La tecnología ha llevado estos campos a nuevos niveles, permitiendo mapas digitales interactivos y sistemas de navegación avanzados. Los sistemas de posicionamiento global y otras tecnologías de localización son componentes esenciales, permitiendo aplicaciones desde la navegación en tiempo real hasta el seguimiento de activos y personas.

Los principios de la física y la ingeniería subyacen a las tecnologías de localización. La trilateración, por ejemplo, se basa en principios geométricos y matemáticos para determinar posiciones exactas utilizando señales de diferentes fuentes. Estas disciplinas han proporcionado el conocimiento técnico necesario para desarrollar y mejorar estas tecnologías. También las ciencias sociales y el urbanismo han sido impactados por la capacidad de ubicarnos. Los datos permiten un urbanismo más eficiente, mejorando el transporte público, la distribución

de servicios y la planificación de infraestructuras. La integración de datos de localización en la planificación urbana ha permitido ciudades más inteligentes y conectadas.

La filosofía de la tecnología invita a reflexionar sobre cómo las herramientas y dispositivos que creamos afectan nuestra existencia y nuestra percepción del mundo. La capacidad de ubicarnos no es solo una cuestión técnica, sino que también tiene profundas implicaciones filosóficas y sociales.

Martin Heidegger argumenta que la tecnología no es solo una herramienta, sino una forma de revelación [22] del mundo. La tecnología de localización revela nuestra posición en el espacio de manera precisa, pero también cambia nuestra relación con el espacio y el tiempo. A través de esta lente, podemos ver cómo los dispositivos de localización mediatizan nuestra percepción del mundo y nos permiten interactuar con él de nuevas maneras.

Desde los primeros mapas rudimentarios hasta los avanzados sistemas de navegación actuales, la tecnología ha evolucionado para proporcionar métodos cada vez más precisos y confiables para la localización.



Figura 1: Mapa del mundo elaborado en 1467. [28]

En este contexto, varias tecnologías han emergido para satisfacer esta necesidad, cada una con sus propias ventajas y limitaciones. El GPS¹ es quizás la tecnología de localización más conocida y utilizada en la actualidad. Desarrollado inicialmente por el Departamento de Defensa de los Estados Unidos [32] en la década de 1970, el GPS se convirtió en un sistema completamente operativo en 1993. Consiste en una constelación de al menos 24 satélites que orbitan la Tierra, proporcionando señales que permiten a los receptores GPS calcular su posición exacta en cualquier lugar del planeta. Revolucionó la navegación y la localización, ofreciendo una

¹Global Positioning System

precisión que varía de unos pocos metros a centímetros, dependiendo del equipo y las condiciones. Sin embargo, la señal GPS puede ser bloqueada o degradada en interiores, áreas urbanas densas o bajo condiciones meteorológicas adversas, lo que limita su efectividad en ciertos entornos.

El Bluetooth es una tecnología que, aunque inicialmente no fue diseñada para la localización, ha encontrado su lugar en este campo debido a su versatilidad y bajo consumo de energía [7]. Introducido en 1994 por Ericsson, el Bluetooth fue concebido para facilitar la comunicación inalámbrica de corto alcance entre dispositivos, creando redes personales (PANs) que permiten la sincronización de datos y la conectividad entre diversos dispositivos electrónicos. Es por ello que se convierte en un potencial candidato cuando las condiciones del lugar en el que se encuentra un usuario no son favorables al uso de GPS. Con la introducción de Bluetooth Low Energy (BLE) y la popularización de los beacons, el Bluetooth se ha convertido en una opción viable para la localización en interiores.

1.2. Objetivo

El objetivo principal de este proyecto es desarrollar una aplicación móvil que, mediante el uso de balizas Bluetooth (Beacons), permita la localización precisa de usuarios en interiores, ofreciendo indicaciones básicas de navegación y proporcionando una solución efectiva para controlar el acceso a espacios cerrados. Esta aplicación tiene como finalidad mejorar la experiencia de localización en entornos donde los sistemas de posicionamiento global no son efectivos, implementando algoritmos avanzados de cálculo de distancias y posicionamiento basados en la tecnología de Beacons.

Los objetivos específicos del proyecto son:

- **Implementar un sistema de detección de Beacons:** Configurar el hardware y desarrollar el software para detectar balizas BLE en tiempo real.
- **Desarrollar algoritmos de cálculo de distancia:** Implementar algoritmos basados en la intensidad de la señal recibida (RSSI) para estimar la distancia entre el dispositivo móvil y las balizas.
- **Incorporar un sistema de posicionamiento mediante multilateración:** Aplicar

técnicas de trilateración/multilateración para calcular la posición precisa del usuario en relación a varias balizas.

- **Diseñar una interfaz gráfica intuitiva:** Crear una interfaz amigable que permita al usuario visualizar su posición y la de las balizas en un mapa.
- **Integrar las funcionalidades en un entorno de trabajo completo:** Asegurar que todas las características de la aplicación funcionen en conjunto, realizando pruebas de integración y depuración.

1.3. Fases del proyecto

El desarrollo del proyecto siguió una serie de fases claramente definidas, según lo especificado en el anteproyecto, que cubren desde la investigación inicial hasta la documentación final. Estas fases permitieron organizar el trabajo de manera eficiente y asegurar la entrega de una aplicación.

1.3.1. Investigación y análisis de requisitos

En esta primera fase, se llevó a cabo una investigación exhaustiva sobre las tecnologías y conceptos clave utilizados en el proyecto, además de la definición de los requisitos funcionales y no funcionales:

1. **Geofencing, ¿qué es y qué se puede hacer con él?:** Se investigó sobre el concepto de geofencing y su aplicabilidad en sistemas de localización, analizando sus posibles usos y cómo podría integrarse con la tecnología de Beacons.
2. **Anteproyecto y definición de requisitos:** Se realizó el anteproyecto, en el cual se definieron los requisitos específicos de la aplicación, tanto desde el punto de vista técnico como funcional, considerando las necesidades de los usuarios y las limitaciones del hardware.

1.3.2. Diseño y planificación del sistema

Durante esta fase, se estructuró y planificó el diseño de la aplicación a nivel de arquitectura y de interfaz de usuario:

1. **Definición de la arquitectura de la aplicación:** Se diseñó la arquitectura de la aplicación móvil, incluyendo la integración de las balizas BLE, la comunicación con las placas ESP32 y el sistema de cálculo de distancias.
2. **Diseño de la interfaz de usuario:** Se diseñaron las pantallas de la aplicación y la experiencia de usuario, asegurando que fueran intuitivas y que facilitaran la navegación y la interacción con los Beacons.

1.3.3. Desarrollo de la aplicación móvil e integración con las placas ESP32

Esta fase fue la más extensa, cubriendo tanto la implementación de la aplicación móvil como la configuración e integración del hardware (balizas ESP32):

1. **Investigación y configuración de balizas en ESP32:** Se investigó la configuración de las balizas utilizando los módulos ESP32, así como las librerías necesarias para convertir estos dispositivos en emisores de señales BLE.
2. **Desarrollo de algoritmos de localización en interiores:** Se implementaron los algoritmos de localización para estimar la distancia del dispositivo móvil a las balizas mediante el uso de RSSI y la fórmula de cálculo de distancias.
3. **Integración con otros BLEs y MACs:** Se integraron otros dispositivos BLE en el sistema, asegurando que la aplicación fuera capaz de interactuar con múltiples dispositivos y que se obtuvieran las direcciones MAC para el cálculo de posiciones.
4. **Diseño e implementación de la interfaz de usuario:** En este apartado, se implementó el diseño previamente definido, desarrollando las pantallas de la aplicación y su lógica de navegación.
5. **Desarrollo del módulo de navegación:** Se creó el módulo de navegación que, utilizando las distancias a las balizas y el mapa, permite guiar al usuario a su destino.
6. **Desarrollo del módulo de cálculo de distancias y multilateración:** Se implementó el sistema de cálculo de distancias, incluyendo el algoritmo de multilateración para estimar la posición precisa del usuario en función de las señales BLE recibidas.

1.3.4. Pruebas y depuración

Esta fase se enfocó en asegurar el correcto funcionamiento de la aplicación mediante una serie de pruebas:

1. **Elaboración de casos de prueba:** Se diseñaron casos de prueba para evaluar cada funcionalidad de la aplicación, desde la detección de balizas hasta la estimación de la posición.
2. **Ejecución de pruebas unitarias:** Se realizaron pruebas unitarias para verificar el correcto funcionamiento de los módulos individuales, como el cálculo de distancias y el escaneo BLE.
3. **Pruebas de integración con placas ESP32:** Se evaluó la integración entre la aplicación móvil y las balizas ESP32, asegurando que la comunicación y los cálculos de distancia se realizaran correctamente.
4. **Identificación y corrección de errores:** A través de las pruebas realizadas, se identificaron errores y se corrigieron para asegurar el funcionamiento estable de la aplicación.

1.3.5. Documentación y entrega del TFG

En esta última fase, se redactaron los documentos asociados al proyecto, incluyendo la memoria y el manual de usuario:

1. **Redacción de la memoria y de la presentación:** Se redactó la memoria del proyecto, donde se documenta todo el proceso de desarrollo, desde la investigación inicial hasta las pruebas y conclusiones finales.
2. **Redacción del manual de usuario:** Se elaboró un manual de usuario detallado para guiar al usuario final en el uso de la aplicación, describiendo cómo navegar por las pantallas y utilizar las funcionalidades clave.

1.4. Contenido de la memoria

La memoria del presente trabajo se estructura en ocho capítulos, que cubren todos los aspectos relacionados con el desarrollo del proyecto, desde la introducción teórica (**Capítulo**

1 - actual) hasta las conclusiones finales. A continuación, se describe el contenido de cada capítulo:

- **Capítulo 2:** Este capítulo profundiza en la tecnología de las balizas Bluetooth (Beacons), describiendo su estructura, funcionamiento y aplicaciones. Se analizan los beneficios de usar Beacons para el posicionamiento interior en comparación con otras tecnologías.
- **Capítulo 3:** Se detallan los diferentes flujos de trabajo involucrados en la aplicación, desde la detección de balizas hasta el cálculo de distancias y la interacción con el usuario. Se explican los casos de uso principales y el ciclo de vida de las operaciones en la aplicación.
- **Capítulo 4:** Aquí se presenta el análisis de requisitos funcionales y no funcionales de la aplicación. Se identifican las necesidades del usuario y se especifican las características que debe cumplir la aplicación para ser efectiva en su contexto de uso.
- **Capítulo 5:** En este capítulo se describe el diseño de la aplicación, incluyendo la arquitectura del sistema y el diseño de la interfaz de usuario. Se explica cómo se estructuran los componentes del software y cómo interactúan entre sí para ofrecer las funcionalidades necesarias.
- **Capítulo 6:** Se detalla el proceso de implementación de la aplicación, desde la configuración del hardware de las balizas hasta la integración del software móvil. Se describen los algoritmos implementados y las librerías utilizadas para el cálculo de distancias y el procesamiento de las señales BLE.
- **Capítulo 7:** Este capítulo se centra en las pruebas realizadas a la aplicación para verificar su correcto funcionamiento. Se describen las pruebas unitarias, de integración y de campo realizadas, así como el proceso de depuración llevado a cabo para corregir los errores detectados.
- **Capítulo 8:** Finalmente, este capítulo presenta las conclusiones derivadas del proyecto, destacando los logros alcanzados y las limitaciones encontradas. También se proponen posibles mejoras y futuras líneas de investigación o desarrollo.

Beacons: Estructura y aplicaciones

2.1. Beacons (balizas) - Definición

Existen unos pequeños dispositivos llamados *beacons*, que son balizas que emiten una señal que se interpreta por otros dispositivos [34]. Operan mediante la emisión periódica de paquetes de datos que contienen un identificador único y otros parámetros relevantes, como los valores *Major* o *Minor* (en la siguiente sección se describe de manera pormenorizada la estructura de la información almacenada típicamente en las beacons). Estos paquetes pueden ser captados por dispositivos móviles que estén equipados con Bluetooth y tengan una aplicación específica para interpretar estas señales. Son utilizados principalmente para transmitir información a dispositivos cercanos (aproximadamente unos 40 metros [15]), como smartphones y tabletas, con el propósito de identificar su proximidad y, en consecuencia, desencadenar acciones específicas basadas en dicha proximidad.

2.2. Proveedores

Hay muchos proveedores de beacons, pero destacan iBeacon (Apple), AltBeacon (Radius Networks) y EddyStone (Alphabet, Google). Uno de los tipos más conocidos es iBeacon, introducido por Apple en 2013 [36]. Estos dispositivos utilizan BLE para transmitir un identificador único (UUID) junto con valores *Major* y *Minor*, permitiendo una identificación más específica dentro de un grupo. Esta tecnología es compatible tanto con dispositivos iOS como Android y se utiliza ampliamente en aplicaciones de marketing de proximidad y navegación en interiores.



Figura 2: Algunas beacons según el proveedor [3].

AltBeacon, desarrollado por Radius Networks [19], es otra tecnología de beacon que destaca por ser un estándar abierto y no propietario. Similar en estructura a iBeacon, utiliza un UUID y los valores *Major* y *Minor*, fomentando la interoperabilidad y la adopción en diversas aplicaciones sin restricciones de patentes.

Por otro lado, Eddystone, introducido por Google en 2015, es un formato de beacon que también es compatible con dispositivos iOS y Android. Eddystone ofrece varias formas de transmisión, incluyendo Eddystone-UID, que transmite un identificador único; Eddystone-URL, que transmite una URL que puede ser interpretada por navegadores y aplicaciones; y Eddystone-TLM [30], que transmite datos de telemetría, como el nivel de batería y la temperatura del beacon. Esta versatilidad hace que Eddystone sea particularmente útil en el contexto

de la web y el Internet de las Cosas (IoT) [27].

2.3. Aplicaciones de las beacons

En esta sección se enumeran algunas de las principales aplicaciones de estas tecnologías. Se pueden clasificar las beacons de distintas formas, siendo su aplicación la más interesante de todas. Actualmente, los más desarrollados son la optimización y desarrollo de procesos, marketing de proximidad, de información y pagos *contact-less*.²

En el ámbito de la **optimización** y desarrollo de procesos, los beacons se utilizan ampliamente en la logística y la gestión de inventarios. Por ejemplo, en almacenes y centros de distribución, los beacons pueden rastrear la ubicación de palés y productos en tiempo real. Esto facilita la gestión del inventario, reduce los tiempos de búsqueda y mejora la precisión de los envíos. Además, en la fabricación, los beacons pueden monitorizar el movimiento de piezas y materiales a lo largo de la cadena de producción, identificando cuellos de botella y optimizando el flujo de trabajo. Esta tecnología permite una mayor eficiencia operativa y una mejor toma de decisiones basada en datos en tiempo real.

El **marketing de proximidad** es otra aplicación clave de los beacons. En tiendas y centros comerciales, los beacons pueden enviar notificaciones personalizadas a los smartphones de los clientes cuando se encuentran cerca de ciertos productos o zonas del establecimiento. Estas notificaciones pueden incluir descuentos, promociones y recomendaciones personalizadas basadas en el historial de compras del cliente o su ubicación dentro de la tienda. Además, los beacons permiten a los minoristas recopilar datos sobre el comportamiento [33] de los clientes, como las rutas de navegación dentro de la tienda y el tiempo pasado en cada sección. Esta información es invaluable para optimizar la disposición de los productos y mejorar la experiencia del cliente.

En museos, galerías de arte, y eventos, los beacons se utilizan para proporcionar **información adicional** sobre exhibiciones y actividades. Por ejemplo, al acercarse a una obra de arte, el visitante puede recibir en su smartphone información detallada [17] sobre el artista, la histo-

²Pagos sin contacto, usualmente con tecnología NFC (Near Field Communication).

ria de la obra y otros datos relevantes. Esto enriquece la experiencia del visitante y proporciona una manera interactiva de explorar el contenido del museo. De manera similar, en conferencias y ferias, los beacons pueden proporcionar información sobre los horarios de las sesiones, la ubicación de los stands y detalles de los expositores, mejorando la navegación y la participación en el evento.

Los **pagos sin contacto**, especialmente en combinación con la tecnología NFC. Los beacons pueden facilitar los pagos móviles sin contacto al proporcionar una autenticación adicional y garantizar que la transacción se realice en una ubicación específica. Por ejemplo, en un restaurante, un cliente puede recibir la cuenta en su smartphone y realizar el pago acercando su dispositivo a un beacon en la mesa. Esto no solo acelera el proceso de pago sino que también mejora la seguridad al reducir la necesidad de manejar tarjetas físicas. Además, en el transporte público y otros servicios, se permiten pagos más rápidos y eficientes, mejorando la conveniencia para los usuarios (ver figura 3).

2.4. Estructura

Es importante conocer el tipo de beacon para que su funcionalidad se adapte al entorno en el que se usará, al igual que el conocimiento de su estructura básica. Por ejemplo, una trama iBeacon está formada principalmente por los siguientes campos:

1. **UUID:** Siglas de *Universally Unique IDentifier*, es un identificador único que permite distinguir un beacon de otro. Es una cadena de 16 bytes que se utiliza para identificar el propietario del beacon y la aplicación asociada. Este identificador es crucial para las aplicaciones que necesitan diferenciar entre múltiples beacons en el mismo entorno.
2. **Major y Minor:** Los campos *Major* y *Minor*, junto con el UUID, proporcionan una mayor especificación dentro del grupo de beacons. El valor *Major* es un número de 2 bytes que generalmente se utiliza para definir y distinguir un grupo grande, como una planta de un edificio. El valor *Minor*, también de 2 bytes, se asigna a un individuo dentro de ese grupo, como una habitación específica dentro de la planta. Este grado de concentración permite una gestión más detallada y precisa de la localización.

3. **Tx PwR:** La potencia de transmisión es un valor crucial que determina el alcance de la señal transmitida por el beacon. Este valor se mide en decibelios milivatios (dBm) y se utiliza para calibrar y estimar la distancia entre el beacon y el dispositivo receptor. Un Tx Power más alto implica un mayor alcance de la señal.
4. **RSSI:** Siglas de *Received Signal Strength Indicator*, mide la potencia de la señal recibida por el dispositivo. Los valores de RSSI también se expresan en dBm y suelen oscilar entre -100 dBm (señal muy débil) y 0 dBm (señal muy fuerte). Este indicador es esencial para estimar la proximidad del dispositivo al beacon.

Obteniendo estos valores, se puede calcular de forma aproximada la distancia a una baliza y, más adelante, triangular la posición del dispositivo que escanea dicha señal. Además, esta distancia define una región o perímetro alrededor de la baliza, lo que ayuda a ofrecer alertas de tiempo real al usuario cuando entra o sale de esta zona definida previamente. Es esta la definición de *geofencing*, la tecnología que se requiere para implementar la aplicación y que consiste en la definición de estas vallas virtuales que permiten registrar objetos dentro del área que se ha definido. La fusión de beacons y geofencing es la encargada de estimar la ubicación de un dispositivo al observar la señal (intensidad medida con el RSSI).

2.5. Ventajas e inconvenientes

Para analizar las ventajas e inconvenientes del uso de la tecnología geofencing o del posicionamiento en interiores con beacons se deben plantear una serie de preguntas concretas:

1. ¿Por qué no usamos GPS para calcular nuestra posición en interiores?

El uso de GPS en interiores presenta varios desafíos. La señal GPS, que se recibe de los satélites, no es capaz de atravesar edificios y estructuras, lo que impide una localización precisa dentro de estos entornos. Además, el GPS no puede determinar en qué planta de un edificio se encuentra el usuario [1]. Una de las ventajas que proporcionan las balizas es la capacidad de implementar un eje vertical, lo que permite determinar con precisión la ubicación del usuario dentro de un edificio, incluyendo la planta en la que se encuentra.

2. ¿Cuál es la relación Coste-Despliegue de un sistema con beacons?

El coste y el despliegue de un sistema basado en beacons dependen del tamaño y la complejidad del área que se desea cubrir. Para un centro comercial o una facultad, se necesitará aproximadamente una beacon por habitación, lo que equivale a una baliza por cada 10-15 metros cuadrados. El coste de cada beacon puede ser relativamente bajo [31], pero cuando se requiere cubrir grandes áreas, el número total de beacons y su mantenimiento pueden representar una inversión significativa. Sin embargo, la facilidad de instalación y la flexibilidad de los beacons compensan en gran medida estos costos, especialmente en comparación con otras tecnologías de localización en interiores.

3. **¿El uso de beacons vulnera nuestra privacidad?**

Es natural preocuparse por la privacidad cuando se utiliza cualquier tecnología de localización. Los beacons requieren acceso a ciertos permisos en el teléfono del usuario, como el Bluetooth y la ubicación aproximada. No obstante, los usuarios tienen control sobre estos permisos y pueden desactivarlos en cualquier momento para dejar de compartir sus datos. A diferencia de otras tecnologías que pueden rastrear ubicaciones más detalladas y personales, los beacons generalmente recopilan datos menos intrusivos y se utilizan principalmente para mejorar la experiencia del usuario mediante servicios personalizados [16].

4. **¿Cómo es la precisión de los beacons en comparación a otras tecnologías de localización?**

La precisión de los beacons puede variar en función de diversos factores como el entorno, la densidad de los beacons o la calibración de la señal [14]. En comparación con otras tecnologías de localización, los beacons proporcionan una precisión suficiente para numerosas aplicaciones, especialmente en interiores donde el GPS no es práctico. La precisión típica de un sistema de beacons oscila entre 1 y 3 metros, lo cual resulta adecuado para la mayoría de las aplicaciones comerciales y de consumo.

Una de las ventajas más destacadas de los beacons es su capacidad para mantener una precisión constante en interiores, a pesar de las interferencias que afectan al GPS y al Wi-Fi. No obstante, esta precisión puede disminuir debido a la presencia de obstáculos físicos, como paredes y muebles, y a la interferencia de otras señales electrónicas.

Para reducir estos efectos, es crucial llevar a cabo una calibración adecuada y una pla-

nificación meticulosa del despliegue de los beacons. En contraste, un inconveniente es que los beacons pueden no ser tan precisos como otras tecnologías avanzadas de localización, como las basadas en cámaras RGB-D [2], sensores de rango o ultrasonidos [29], que pueden alcanzar precisiones en el rango de centímetros.

5. Entonces, ¿cuáles son los usos reales y las ventajas de las beacons?

Los beacons tienen múltiples usos prácticos y ventajas en diversas aplicaciones. Entre los usos más comunes se encuentran los mencionados anteriormente, como la navegación en entornos cerrados, el marketing de proximidad, donde los usuarios reciben anuncios y ofertas personalizadas y los pagos sin contacto. La navegación en interiores permite a los usuarios encontrar su camino en grandes instalaciones como aeropuertos, centros comerciales y museos.

Este mejora la experiencia del cliente mediante la entrega de contenido relevante y ofertas especiales basadas en su ubicación actual. Los pagos sin contacto, facilitados por la combinación de beacons y tecnología NFC, hacen que las transacciones sean más rápidas y seguras. Dado que una imagen vale más que mil palabras, esta representación visual resulta altamente esclarecedora:

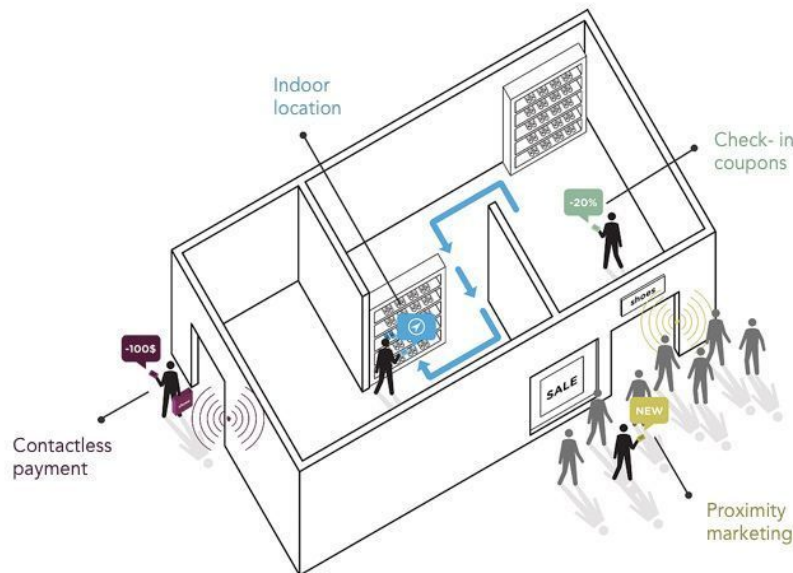


Figura 3: Múltiples usos de beacons en una misma habitación. [13]

Las ventajas de las beacons son su tamaño, que es inferior al de una moneda. La duración puede ser de años si está conectada a una pila, dependiendo del tipo de beacon. Toda la gestión del protocolo, implementaciones y recopilación de datos se realiza por parte del dispositivo (se comporta como receptor de señal), por lo que es muy fácil diseñar nuevas aplicaciones que las detecten. Obtener información de todo el entorno, enviar alertas en tiempo real o información personalizada a los usuarios cercanos.

6. **¿Se están utilizando beacons en la actualidad?**

Sí, los beacons se están utilizando ampliamente en diversas industrias en todo el mundo. Grandes empresas como Walmart [26] han implementado sistemas de localización basados en beacons en todos sus almacenes para mejorar la gestión del inventario y optimizar los procesos logísticos. McDonald's utiliza beacons para liderar el marketing de proximidad [18], enviando ofertas personalizadas, alertas y cupones a sus clientes en sus establecimientos. Otros espacios como hospitales, museos, cines, centros comerciales y grandes almacenes también están aprovechando las múltiples posibilidades que ofrece esta tecnología para mejorar la experiencia del usuario, optimizar operaciones y aumentar la eficiencia (ver figura 3).

Flujos de trabajo

3.1. LaTeX y Overleaf

LaTeX es un sistema de composición de textos utilizado fundamentalmente en la academia y en la publicación de documentos técnicos y científicos. Su capacidad para manejar fórmulas matemáticas complejas y su control preciso sobre la estructura del documento lo convierten en la herramienta preferida para la creación de documentos de alta calidad.

Overleaf (ver figura 4) es un editor en línea [25] que facilita la escritura en LaTeX, proporcionando un entorno colaborativo en la nube. Permite a varios usuarios trabajar simultáneamente en un documento, ver los cambios en tiempo real, y gestionar versiones. Esta herramienta ha sido utilizada en la creación de la memoria de este proyecto, permitiendo una redacción eficiente y organizada del contenido.



Figura 4: Logo de Overleaf [25].

3.2. Metodologías ágiles

Las metodologías ágiles son un enfoque de gestión de proyectos que enfatiza la entrega incremental, la colaboración constante con el cliente y la capacidad de respuesta ante los cambios. Entre los principios fundamentales se encuentran:

1. **Iteración e incremento:** Dividir el proyecto en pequeños ciclos de desarrollo, conocidos como iteraciones, que permiten entregar componentes funcionales del producto de manera continua.
2. **Colaboración continua:** Fomentar una comunicación constante entre todos los miembros del equipo y con los stakeholders para asegurar que el proyecto se alinee con las expectativas y necesidades.
3. **Adaptabilidad:** Permitir la flexibilidad para cambiar la dirección del proyecto según sea necesario, basándose en el feedback recibido durante el desarrollo.

3.3. Jira

Jira (ver figura 5) es una herramienta de gestión de proyectos desarrollada por Atlassian [20], ampliamente utilizada para implementar metodologías ágiles. Permite a los equipos planificar, rastrear y gestionar proyectos de manera eficiente.



Figura 5: Logo de Jira [5].

Dentro de las funcionalidades clave destacan el uso del **Backlog** y **Sprint Planning**, que facilitan la creación y gestión de un conjunto de tareas y la **planificación** en sprints, asegurando que el equipo tenga un flujo de trabajo claro.

- **Backlog:** Es una lista de tareas pendientes o elementos de trabajo que deben completarse para desarrollar un proyecto. Si imaginamos un cuaderno donde se anotan todas las cosas necesarias para completar una tarea grande, cada página del cuaderno es una subtarea más específica que contribuye al objetivo final. En el contexto de desarrollo de software, el backlog incluye características, mejoras y correcciones necesarias para el producto.

- **Sprint Planning:** es una reunión donde el equipo decide qué tareas del backlog van a trabajar en un período de tiempo corto y fijo, llamado **sprint** (usualmente entre una y cuatro semanas). La forma de entenderlo es imaginar que observas el cuaderno de tareas mencionado anteriormente y eliges qué páginas (tareas) puedes completar en la próxima semana o dos. El objetivo es planificar de manera realista lo que se puede lograr, dividir el trabajo entre los miembros del equipo y definir un objetivo claro para el sprint.

Kanban y Scrum Boards proporcionan tableros visuales para gestionar tareas utilizando sus propios marcos de trabajo, permitiendo una visualización clara del progreso del proyecto. Otro aspecto muy destacable es la posibilidad de generar informes y análisis, ya que se ofrecen herramientas que ayudan a evaluar el rendimiento del equipo y la evolución del proyecto.

3.4. GitHub

GitHub (ver figura 6) es una plataforma de desarrollo colaborativo que permite a los equipos gestionar y almacenar su código fuente utilizando el sistema de control de versiones Git. Esta herramienta es esencial para gestionar proyectos de software, ya que facilita la colaboración entre desarrolladores, el seguimiento de cambios, la revisión de código y la gestión de versiones.



Figura 6: Logo de Github [38].

Una de las principales utilidades de GitHub en proyectos de desarrollo de software es el manejo eficiente de **ramas** (branches), que permite a los desarrolladores trabajar en diferentes funcionalidades o correcciones de errores de manera independiente y paralela. Al trabajar en ramas, se evita el riesgo de introducir errores en la versión principal del código (normalmente

la rama *master* o *main*), ya que las nuevas funcionalidades o correcciones pueden ser desarrolladas y probadas antes de ser integradas en la rama principal. Además, GitHub permite la gestión de **versiones** del proyecto, lo que es fundamental para llevar un control riguroso de las distintas fases del desarrollo. Cada versión o lanzamiento puede ser etiquetado (tagged), facilitando el acceso a versiones específicas del código en cualquier momento.

3.5. Adaptación del proyecto

Para aplicar adecuadamente esta metodología a nuestro entorno, se trasladan las fases definidas en el anteproyecto a Jira, de forma que queda reflejado el uso del backlog (ver figura 7).

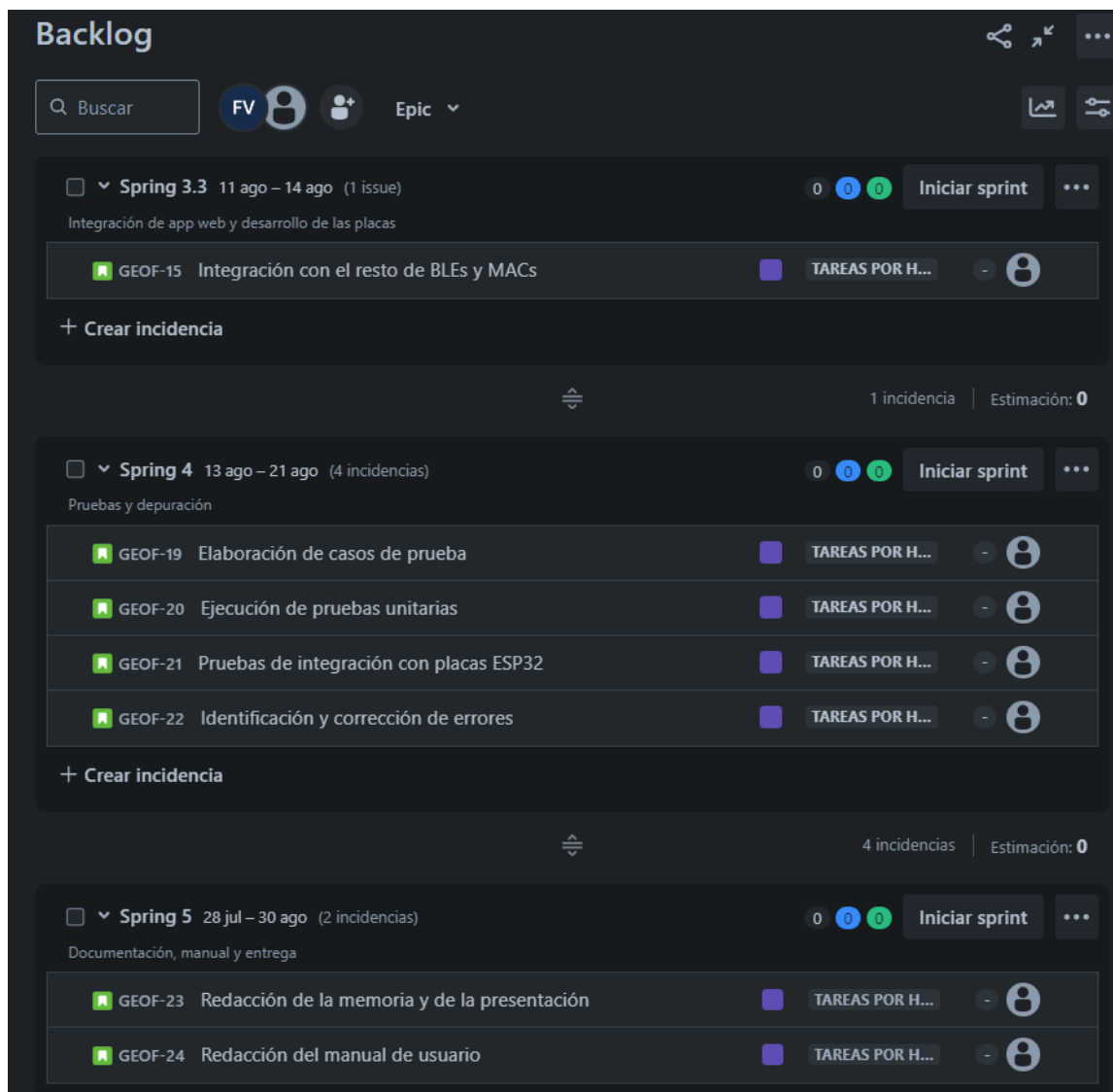


Figura 7: Vista de Backlog del proyecto[10].

Del mismo modo, definimos los sprints siguiendo la metodología mencionada. En este caso, cada uno de los sprints se iguala a una tarea principal y, cada una de las subtarefas se derivan de estas (ver imagen 8).

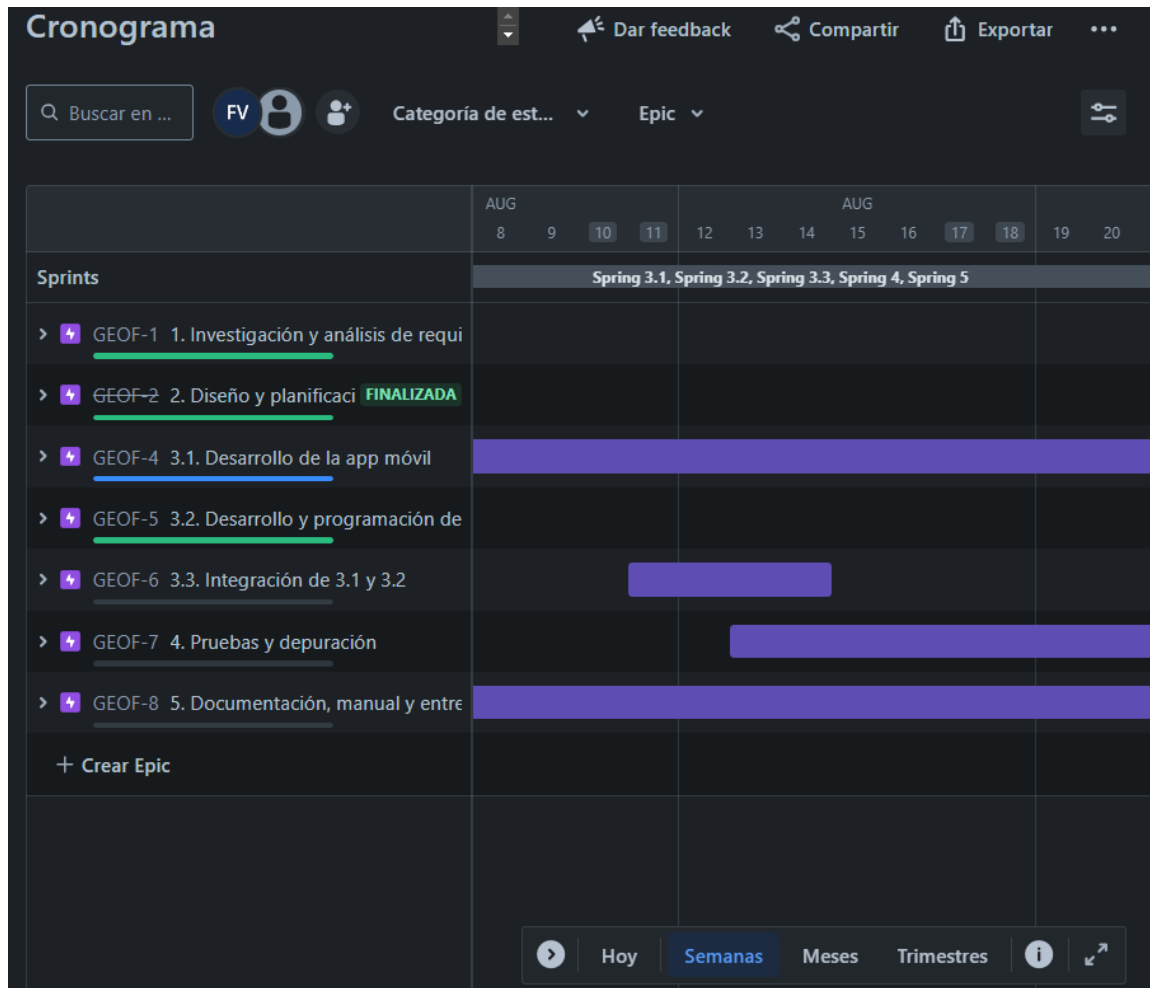


Figura 8: Cronograma de las fases del proyecto. Mes: Agosto [11].

También se ha incorporado el uso de GitHub para gestionar el control de versiones y el desarrollo colaborativo. Se han creado múltiples ramas a raíz de la principal, como la rama *develop*, en la que se efectúan los cambios finales de la app, o *v1deteccionbeacons*, donde se han desarrollado nuevas funcionalidades o pruebas de concepto relacionadas con la detección de beacons (ver figura 9). Esto ha permitido trabajar de manera independiente en nuevas características sin afectar la rama principal ‘master’, que representa la versión estable del proyecto.

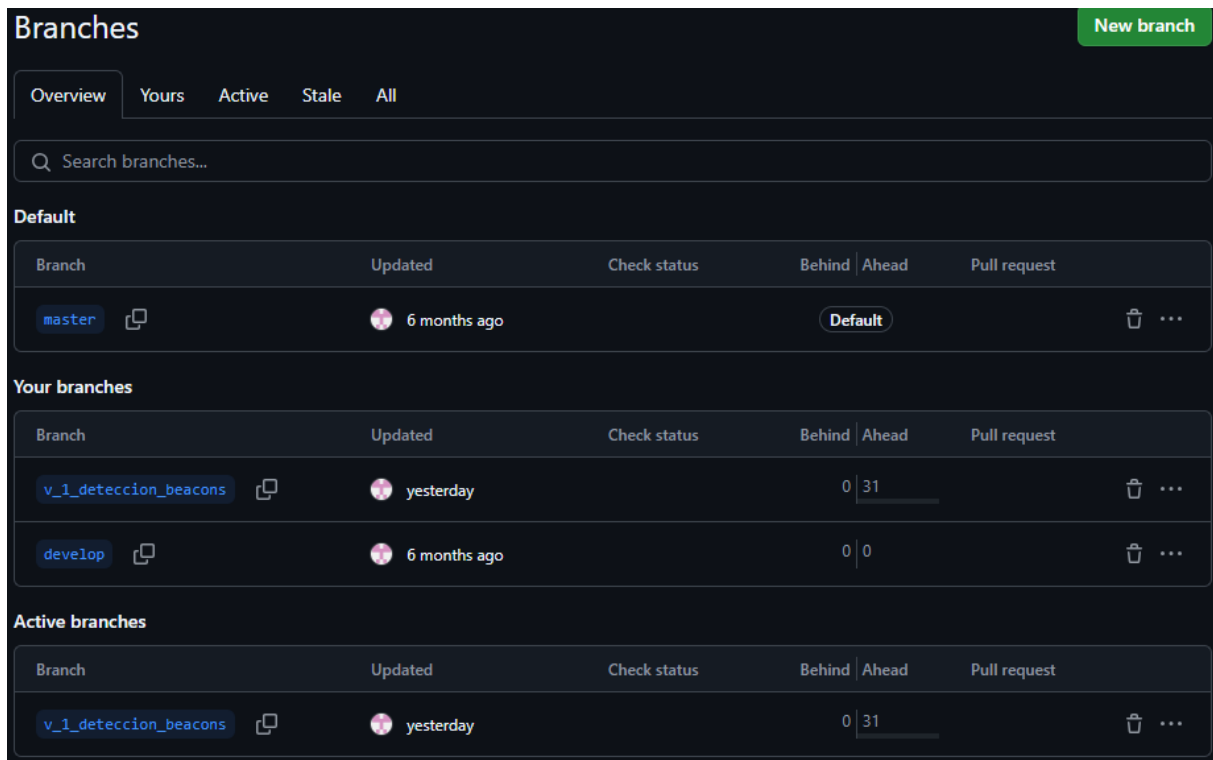


Figura 9: Vista de ramas en el repositorio GitHub del proyecto [8].

Además, el uso de ramas y *commits* ha facilitado la integración continua y el despliegue de nuevas versiones, manteniendo un flujo de trabajo ágil y organizado, acorde con las metodologías ágiles implementadas en este proyecto (ver figura 10).

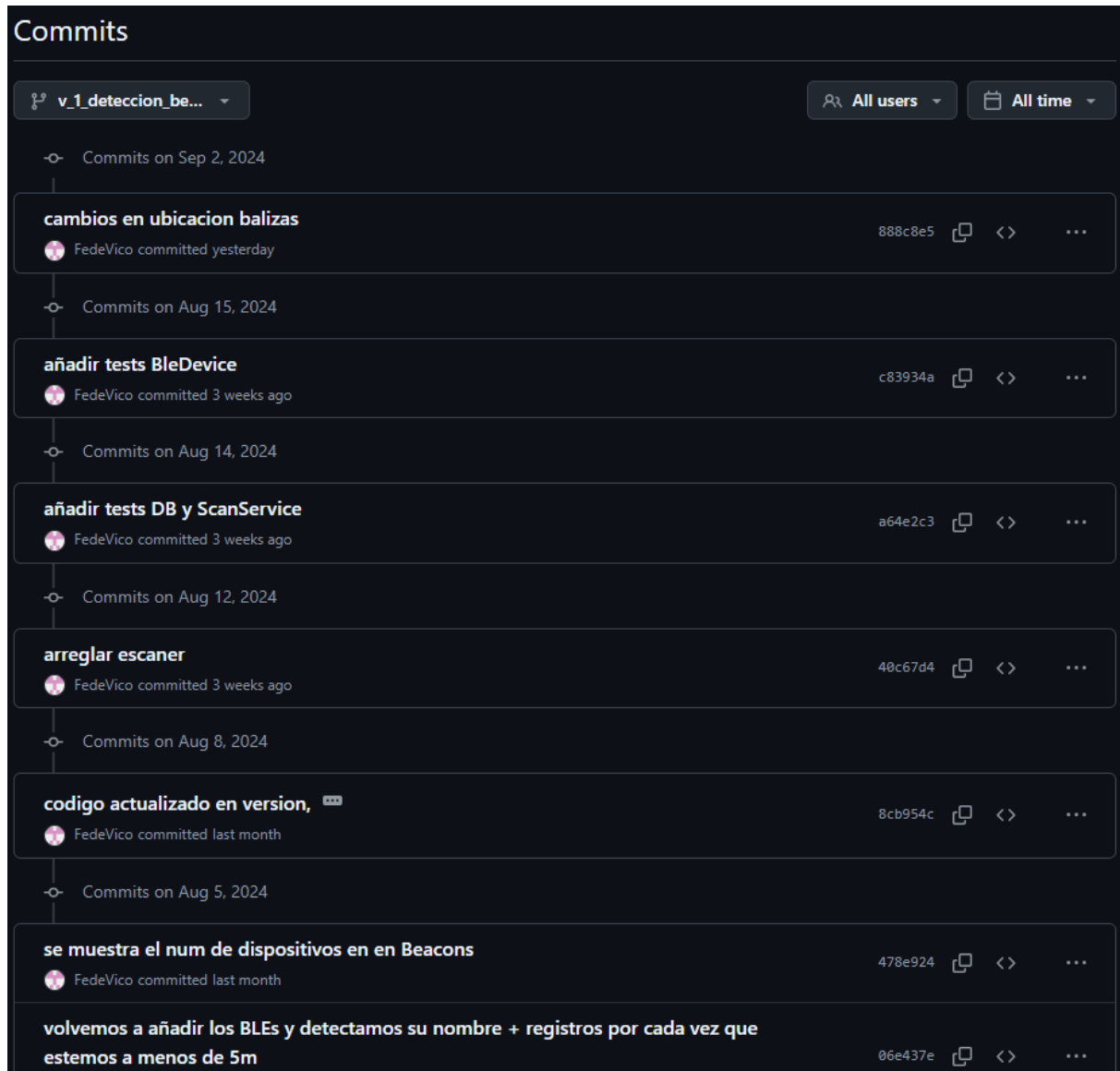


Figura 10: Últimos commits de la rama *v1deteccionbeacons* en GitHub [9].

4

Análisis de requisitos

4.1. Identificación de necesidades de localización en interiores

En el contexto de la implementación de una solución de geolocalización en interiores mediante balizas Bluetooth (Beacons), es fundamental identificar las principales necesidades que impulsan el desarrollo de esta tecnología. Actualmente, los sistemas tradicionales de posicionamiento global tienen un rendimiento limitado en espacios cerrados, como edificios y entornos urbanos densos, donde las señales de satélite son bloqueadas o interferidas. Esto presenta un desafío importante para sectores que dependen de la localización precisa de personas o activos, como hospitales, centros comerciales, aeropuertos o grandes almacenes.

Uno de los principales objetivos de este proyecto es mejorar la accesibilidad en estos entornos, facilitando la navegación y proporcionando indicaciones precisas de ubicación para los usuarios. Para ello, se han identificado las siguientes necesidades:

- **Posicionamiento preciso en interiores:** Garantizar que los usuarios puedan ser localizados con una precisión adecuada, incluso en espacios donde el GPS no es funcional.
- **Navegación en tiempo real:** Proporcionar indicaciones en tiempo real a los usuarios, ayudándolos a moverse dentro de edificios grandes o complejos gracias al envío de notificaciones y alertas.
- **Mejora en la gestión de espacios:** Optimizar la gestión de espacios cerrados a través de la monitorización del flujo de personas y la ubicación precisa de activos.

4.2. Definición de requisitos del sistema

Se han tenido en cuenta las necesidades descritas y, a continuación, se detallan los principales requisitos funcionales y no funcionales que el sistema debe cumplir para satisfacer los objetivos del proyecto.

4.2.1. Requisitos funcionales

Los requisitos funcionales especifican las principales características que el sistema debe implementar para cumplir con sus objetivos. Estas funciones son esenciales para garantizar que el sistema realice correctamente sus tareas principales y ofrezca una experiencia óptima al usuario. En nuestro sistema, estos requisitos son los mencionados a continuación:

- **Detección de balizas BLE:** La aplicación debe ser capaz de detectar y procesar las señales emitidas por las balizas Bluetooth en tiempo real.
- **Cálculo preciso de distancias:** Implementar algoritmos que estimen con precisión la distancia entre las balizas y los dispositivos móviles, basándose en la intensidad de la señal (RSSI).
- **Posicionamiento mediante multilateración:** Calcular la posición precisa del usuario utilizando técnicas de multilateración, combinando las señales de múltiples balizas para obtener coordenadas fiables.
- **Interfaz de usuario intuitiva:** Diseñar una interfaz gráfica sencilla y amigable que permita a los usuarios ver su posición y las balizas en un mapa, con indicaciones claras para la navegación.
- **Integración con dispositivos móviles:** La aplicación debe integrarse de manera eficiente con los sistemas operativos móviles (iOS y Android) para manejar las señales BLE y proporcionar notificaciones precisas.
- **Accesibilidad:** La aplicación debe ser accesible y fácil de usar para personas con diferentes niveles de conocimiento tecnológico.

4.2.2. Requisitos no funcionales

Los requisitos no funcionales se refieren a los atributos de calidad que el sistema debe poseer para garantizar su rendimiento, seguridad, y capacidad de escalabilidad. Aunque no están directamente relacionados con funciones específicas del sistema, estos requisitos son fundamentales para asegurar que el sistema funcione de manera efectiva y satisfactoria en diferentes entornos y condiciones. Se destacan los siguientes:

- **Eficiencia y bajo consumo de energía:** La aplicación y las balizas deben funcionar de manera eficiente, optimizando el consumo energético para no agotar rápidamente la batería de los dispositivos.
- **Escalabilidad:** El sistema debe ser escalable para ser implementado en diferentes tipos de edificios, desde pequeños espacios comerciales hasta grandes complejos industriales.
- **Seguridad y privacidad:** Garantizar que la información personal del usuario, así como sus datos de localización, estén protegidos mediante el uso de protocolos de seguridad robustos.

5

Diseño de la aplicación

5.1. Introducción

Este capítulo se centra en la configuración y el funcionamiento de los componentes clave utilizados en la implementación de nuestro sistema de posicionamiento basado en beacons. Se comenzará describiendo la configuración del hardware, particularmente de las balizas que emiten las señales necesarias para calcular la ubicación del dispositivo móvil. A continuación, se explicarán los métodos utilizados para medir y calcular la distancia entre las balizas y el dispositivo, y cómo esta información se transmite entre cada una de las partes involucradas.

5.2. Interacción entre el móvil y la base de datos

Una vez que el dispositivo móvil determina su posición a través del procesamiento de señales recibidas de las balizas BLE, se produce un flujo de datos entre el dispositivo y la base de datos que facilita el almacenamiento y análisis de la información. A continuación, describimos los pasos clave que intervienen en este proceso, haciendo referencia a la figura

11.

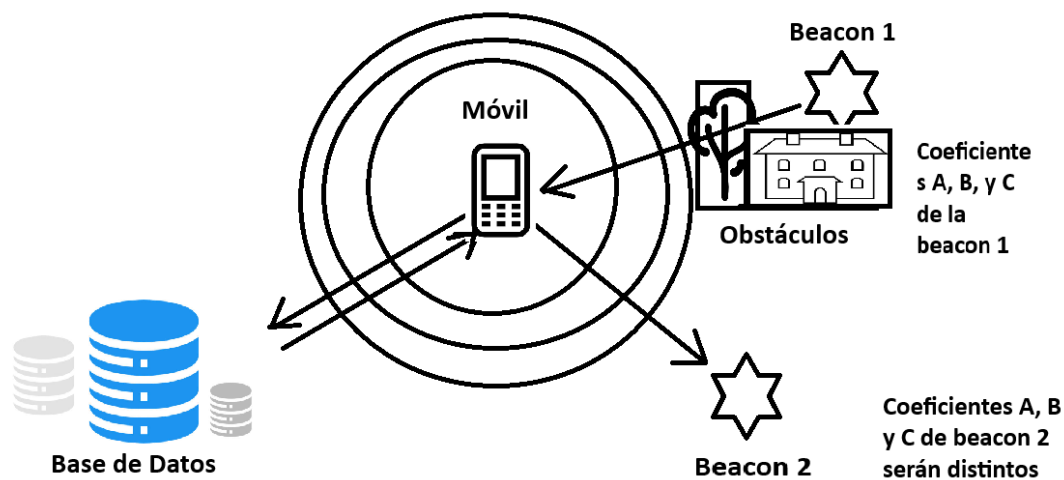


Figura 11: Esquema de flujo de información entre el móvil, las balizas y la base de datos, incluyendo obstáculos.

5.2.1. Proceso de detección de la posición

Cuando el dispositivo móvil detecta las señales de las balizas, utiliza los coeficientes A, B y C para calcular su distancia con base en la fórmula de regresión que relaciona la potencia de la señal RSSI con la distancia. Este cálculo de distancia es ajustado por el dispositivo móvil como se podrá observar en secciones posteriores.

5.2.2. Transferencia de datos a la base de datos

Una vez determinada la posición del móvil, la información recopilada (incluyendo el tiempo de detección y la identificación de las balizas cercanas) se transfiere a la base de datos. Esta comunicación puede realizarse a través de una conexión a internet, en la que el móvil envía los datos a un servidor central que contiene la base de datos en tiempo real.

5.2.3. Procesamiento en la base de datos

La base de datos almacena toda la información necesaria para gestionar la red de balizas (ver figura 12) y realizar análisis posteriores. Al recibir los datos del dispositivo móvil, la base de datos puede llevar a cabo varias funciones, como insertar una nueva baliza, actualizar los datos de las existentes, etc.

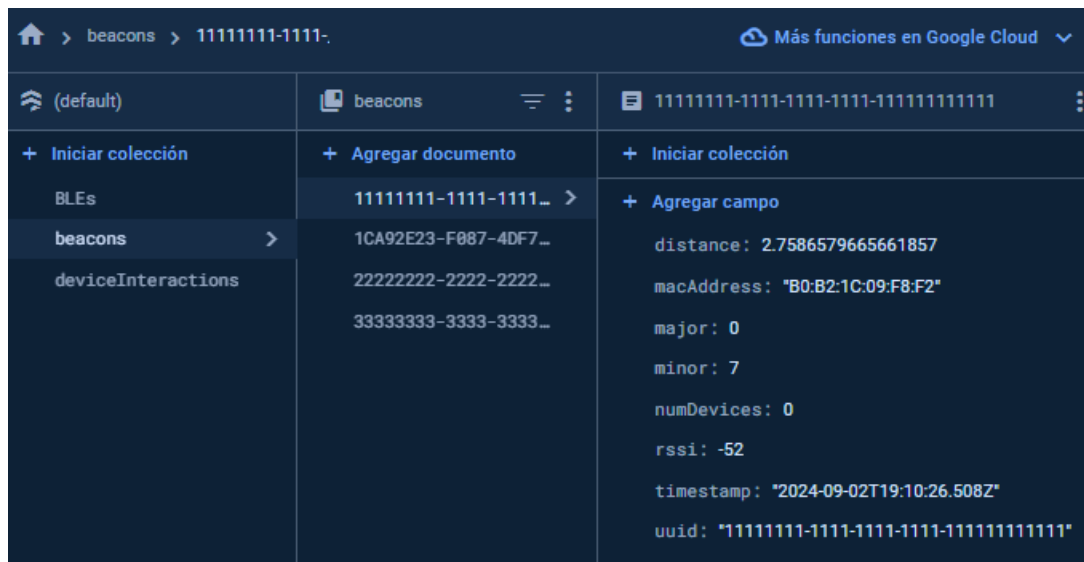


Figura 12: Captura de la base de datos [12].

5.2.4. Respuesta del servidor

El servidor envía una respuesta al dispositivo móvil con los datos de cada baliza en tiempo real. Por ejemplo, si se detecta una nueva baliza se enviarán los datos que previamente ha recibido el dispositivo móvil sobre ella. Si esta baliza ya existía en la base de datos, se actualizarían los datos correspondientes (distancia, RSSI y marca de tiempo).

Este proceso permite una retroalimentación constante entre el dispositivo móvil y la base de datos, asegurando que las posiciones calculadas sean lo más precisas posible.

5.3. Configuración de las balizas

En esta parte se define el comportamiento de las beacons a nivel de hardware. Se define un protocolo de emisión de señal de bajo consumo mediante una adaptación del código desarrollado en Arduino cuyos fragmentos aparecen en el Apéndice B.

Este programa crea un emisor BLE en un dispositivo ESP32, el cual enviará periódicamente tramas iBeacon. La secuencia de acciones para la creación del emisor es la siguiente:

1. Se crea un emisor BLE.
2. Se generan los datos de publicidad.
3. Se inicia la publicidad y después un pequeño periodo de tiempo de espera.

4. Se detiene la publicidad.
5. Se entra en estado de hibernación.

El emisor generado envía tramas iBeacon con una UUID específica y valores de identificación únicos. Este comportamiento se logra mediante la configuración de los datos de publicidad y la generación de tramas iBeacon con la biblioteca BLEBeacon [21].

Cada vez que el dispositivo se despierta, incrementa un contador de arranques para llevar un seguimiento del número de veces que ha sido reiniciado.

Una vez que se inicia la publicidad y se envían las tramas iBeacon, el dispositivo entra en un estado de hibernación para conservar la energía, hasta que se despierte nuevamente para repetir el proceso.

El propósito de este programa es permitir que el dispositivo ESP32 funcione como un emisor de tramas iBeacon, que más tarde serán recibidas por nuestro dispositivo móvil para obtener la localización y el seguimiento de este.

5.4. Descripción del hardware

Como se menciona anteriormente, el módulo que se usa para el desarrollo de la aplicación es el ESP32-WROOM-32 (ver figura 13). Éste se basa en el microcontrolador ESP32, desarrollado por Espressif Systems. Destaca por sus capacidades de conectividad inalámbrica y eficiencia energética, lo que lo hace ideal para proyectos como este, que implican la transmisión de datos mediante Bluetooth Low Energy (BLE) y la gestión del consumo de energía. Algunas de las especificaciones técnicas son:

1. **Voltaje de alimentación:** 5V, a través de una conexión USB Micro B.
2. **Voltaje de Entrada/Salida:** 3.3V.
3. **Tx PwR:** Potencia de transmisión, de esta depende el alcance de la señal transmitida.
4. **Rango de Frecuencia del Reloj:** 80 MHz / 240 MHz .
5. **SoC (System on Chip):** ESP32-WROOM-32, integra microprocesador, Bluetooth y Wi-Fi.

6. **Wi-Fi:** Compatible con los estándares Wi-Fi 802.11. y frecuencias de 2.4 GHz o 2.5 GHz.
7. **Bluetooth:** Soporta Bluetooth versión 4.2, incluyendo Bluetooth de baja energía y Bluetooth.
8. **Dimensiones:** Las dimensiones del módulo son 56 mm de largo, 28 mm de ancho y 13 mm de alto, lo que lo hace compacto y fácil de integrar.

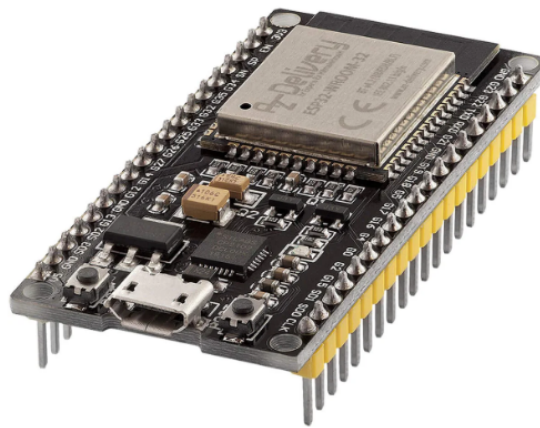


Figura 13: ESP-32-WROOM-32 Module Board. [6]

5.5. Cálculos y aproximaciones

Se han aplicado una serie de factores que se deben considerar para el cálculo de la distancia de un dispositivo a la baliza. En este caso, las mediciones se han realizado gracias a las aproximaciones del paquete Android Beacon Library de Altbeacon [39].

Algorithm 1 Cálculo de Distancia

procedure GETDISTANCE

$measuredPower \leftarrow -59$ ▷ Potencia medida en dBm a 1 metro de distancia

$N \leftarrow 2,0$ ▷ Factor de atenuación de la señal

return $10,0^{\frac{measuredPower - rssi}{10 \times N}}$

end procedure

Para calcular la distancia entre el dispositivo móvil y una baliza utilizando la señal RSSI, es esencial realizar un calibrado preciso de esta señal. Este proceso de calibrado se puede desglosar en varios pasos que aseguran que las mediciones reflejen con exactitud la distancia real. Se comienza estableciendo una potencia de referencia medida (`measuredPower`) a una distancia fija de un metro. Por ejemplo, se puede usar un valor de -53 dBm, que es una medida común para muchos beacons. El ratio se calcula dividiendo la potencia medida a una distancia específica por la potencia de referencia a un metro. Este ratio ayuda a normalizar las variaciones en la señal debidas a factores ambientales.

Utilizando los datos de RSSI y distancias conocidas, se aplican técnicas de regresión para obtener las variables A y B de la fórmula de cálculo de distancia. Estas variables ajustan el modelo a las condiciones específicas del entorno donde se encuentran los beacons. Para mejorar la precisión, se introduce una constante C que optimiza las predicciones a la distancia de referencia de un metro. Esta constante se ajusta mediante pruebas y optimizaciones iterativas. Por último, se realizan pruebas de predicción utilizando los valores ajustados de A, B y C para verificar la exactitud del modelo. La predicción se compara con las distancias reales para evaluar y mejorar la precisión. Para ello, se realizan todos los cálculos necesarios en función a distancias predeterminadas y, en condiciones que se ajusten al entorno real, como pueden ser obstáculos físicos (paredes, personas, etc.) o interferencias de otras señales.

Tras la realización de las mediciones pertinentes y calcular los valores mediante la tabla nombrada (ver Apéndice A), se seleccionan los valores $A = 1.203420305$, $B = 6.170094565$ y $C = -0.203420305$ como los coeficientes calibrados del dispositivo en función a la señal recibida. Por lo tanto, la nueva función que realiza el cálculo de la distancia se transforma en:

En lo referente a la posición de las balizas, es fundamental entender que su ubicación precisa (latitud y longitud) ya es conocida y está predeterminada. Sin embargo, el cálculo de la distancia entre el dispositivo móvil y las balizas se obtiene mediante la señal que estas emiten, la cual solo proporciona una distancia relativa, no las coordenadas exactas. Para determinar la ubicación exacta del dispositivo móvil (latitud y longitud), es necesario utilizar un algoritmo de **multilateración**, que es una extensión del algoritmo de trilateración. La trilateración utiliza tres puntos de referencia para calcular la posición (ver figura 13), mientras que la multilateración emplea múltiples puntos, lo que permite una mayor precisión al determinar la ubicación

Algorithm 2 Cálculo de distancia con coeficientes

procedure GETDISTANCE

$A \leftarrow 1,203420305$

$B \leftarrow 6,170094565$

$C \leftarrow -0,203420305$

$measuredPower \leftarrow -53$ $\triangleright$ Potencia medida en dBm a 1 metro de distancia

$ratio \leftarrow \frac{average(rssiList)}{measuredPower}$

$predictedDistance \leftarrow e^A \times ratio^B + C$

return $predictedDistance$

end procedure

del dispositivo en relación con las balizas. Es crucial no confundir la trilateración con la triangulación, ya que esta última se basa en medir ángulos absolutos entre los nodos para estimar una ubicación aproximada.

Algunos de los datos que requieren un tratamiento especial para su uso en la aplicación son:

1. **Precisión en el cálculo de la posición usando trilateración:** La trilateración es un método fundamental en nuestra aplicación de posicionamiento, ya que permite determinar la ubicación del dispositivo móvil utilizando la distancia a al menos tres balizas (beacons) con coordenadas conocidas. Este método se basa en principios geométricos y matemáticos para calcular la posición exacta del dispositivo, midiendo las distancias desde el dispositivo hasta estas balizas.

Para comprender cómo funciona la trilateración, es importante visualizar el proceso. Cada baliza emite una señal que es captada por el dispositivo móvil. Al medir la intensidad de la señal recibida (RSSI), se puede estimar la distancia del dispositivo a cada baliza. Esta distancia se representa como el radio de un círculo con el centro en la posición de la baliza. Cuando se utilizan al menos tres balizas, los círculos dibujados a partir de estas distancias intersecarán en uno o más puntos, y la intersección de estos círculos permite calcular la posición del dispositivo.

El cálculo de la trilateración en dos dimensiones implica resolver un sistema de ecuaciones basado en las distancias a tres puntos conocidos, como se describe en el siguiente

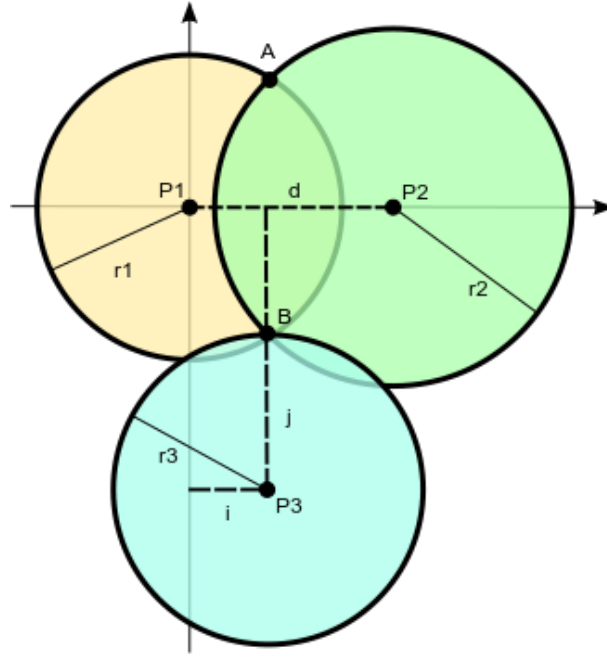


Figura 14: Cálculo de una posición (B) mediante trilateración. [37]

ejemplo.

Para tres balizas ubicadas en las coordenadas (x_1, y_1) , (x_2, y_2) , y (x_3, y_3) y las distancias del dispositivo móvil a estas balizas son d_1 , d_2 , y d_3 respectivamente, las ecuaciones que describen los círculos son:

$$(x - x_1)^2 + (y - y_1)^2 = d_1^2 \quad (1)$$

$$(x - x_2)^2 + (y - y_2)^2 = d_2^2 \quad (2)$$

$$(x - x_3)^2 + (y - y_3)^2 = d_3^2 \quad (3)$$

Supongamos que tenemos tres balizas ubicadas en las siguientes coordenadas: Baliza A en $(2, 3)$, Baliza B en $(5, 7)$ y Baliza C en $(1, 9)$.

Y las distancias medidas del dispositivo móvil a estas balizas son: $d_1 = 3,2$ metros, $d_2 = 4,1$ metros y $d_3 = 2,8$ metros.

Las ecuaciones del círculo para cada baliza serían:

$$(x - 2)^2 + (y - 3)^2 = 3,2^2 \quad (4)$$

$$(x - 5)^2 + (y - 7)^2 = 4,1^2 \quad (5)$$

$$(x - 1)^2 + (y - 9)^2 = 2,8^2 \quad (6)$$

Estas ecuaciones permiten calcular la posición aproximada del dispositivo móvil resolviendo el sistema de ecuaciones resultante, en este caso propuesto como ejemplo: $(x, y) = (0,93, 6,105)$.

La trilateración tiene varias ventajas, entre ellas, la simplicidad matemática y la capacidad de proporcionar una estimación razonable de la posición utilizando únicamente la distancia a tres puntos conocidos. No obstante, también tiene sus limitaciones, ya que la precisión de la trilateración depende en gran medida de la exactitud de las mediciones de distancia, las cuales pueden estar sujetas a errores debido a la interferencia de señales y obstáculos físicos. Además, en entornos muy dinámicos o con muchas reflexiones de señal, puede producir resultados menos precisos.

2. **Suavizado de la distancia en metros:** El suavizado de datos es una técnica fundamental en el procesamiento de señales y en el análisis de datos, especialmente en aplicaciones donde se necesita reducir el ruido y obtener mediciones más precisas y estables. En el contexto de la estimación de distancias utilizando beacons, el suavizado se aplica para minimizar los errores en las mediciones de RSSI (Received Signal Strength Indicator), proporcionando así una estimación más precisa de la distancia.

El filtro de media móvil es una de las técnicas más simples y efectivas para el suavizado de datos. Esta técnica implica el cálculo de un promedio dentro de una ventana móvil que se desplaza a lo largo de la serie de datos. La idea principal es que, al promediar múltiples mediciones, se reduce el impacto de los valores atípicos y del ruido aleatorio. Se define una ventana de tamaño fijo, que determina cuántas mediciones se incluirán

en el cálculo del promedio. El tamaño asignado a esta ventana es clave, ya que afecta directamente al grado de suavizado. Una ventana más grande proporcionará un mayor nivel de suavizado, pero a su vez, puede disminuir la sensibilidad a cambios rápidos en los datos.

Para cada posición en la serie de datos, se calcula el promedio de los valores que se encuentran dentro de la ventana. Este promedio se utiliza como el valor suavizado para la posición central de la ventana. Con esto se asegura que cada punto de la serie de datos se basa en un conjunto de mediciones, lo que ayuda a reducir el impacto de valores atípicos y el ruido aleatorio.

La ventana se desplaza un paso adelante a lo largo de la serie de datos, y se repite el cálculo del promedio para la nueva posición de la ventana. Este procedimiento continúa hasta que todos los valores de la serie de datos hayan sido procesados. Al desplazar la ventana y recalcular el promedio en cada paso, el filtro de media móvil genera una serie suavizada que refleja mejor la tendencia subyacente de los datos, minimizando las variaciones abruptas. Este es el pseudocódigo que se genera más adelante para la implementación del suavizado:

Algorithm 3 Aplicar filtro de media móvil

procedure APPLYMOVINGAVERAGEFILTER(distances : List)

$windowSize \leftarrow 3$ ▷ Tamaño de la ventana del filtro

$filteredDistances \leftarrow \text{List<Double>}()$ ▷ Lista para las distancias filtradas

for $i \leftarrow 0$ to $distances.length - 1$ **do**

$startIndex \leftarrow \max(0, i - windowSize + 1)$ ▷ Índice de inicio de la ventana

$endIndex \leftarrow i + 1$ ▷ Índice de fin de la ventana

$values \leftarrow distances[startIndex]$ ▷ Valores dentro de la ventana

$average \leftarrow \text{sum}(values)/values.length$ ▷ Promedio de los valores

$filteredDistances.append(average)$ ▷ Agregar promedio a la lista filtrada

end for

return $filteredDistances$ ▷ Devolver distancias filtradas

end procedure

3. **Conversión en coordenadas geodésicas:** Debido a que la información de la ubicación de las balizas se encuentra en coordenadas (latitud, longitud), es necesario transformar estas distancias para obtener una representación precisa en un plano esférico. La fórmula de Haversine es una herramienta matemática utilizada para este propósito. Esta es un caso especial dentro de la trigonometría esférica que establece una relación entre las coordenadas de 2 puntos dados en una esfera y la distancia que los separa. Para entender mejor esta fórmula, se plantea un ejemplo de cálculo a gran escala. Supongamos que tenemos dos balizas con las siguientes coordenadas:

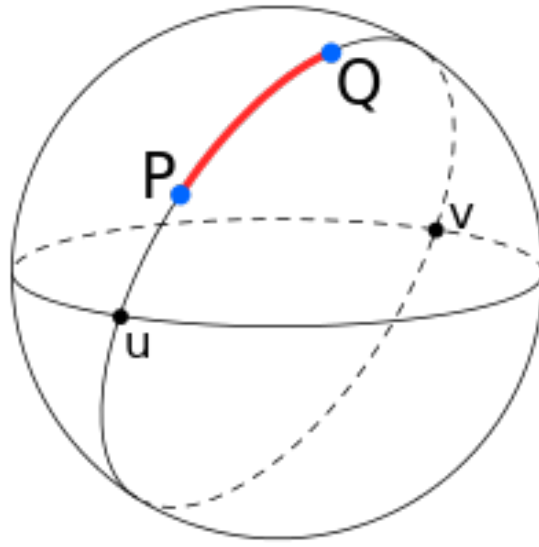


Figura 15: Distancia entre dos puntos en una esfera. [35]

- **Baliza A (San Francisco, USA):** $(\varphi_1, \lambda_1) = (37,7749^\circ N, 122,4194^\circ W)$
- **Baliza B (Los Ángeles, USA):** $(\varphi_2, \lambda_2) = (34,0522^\circ N, 118,2437^\circ W)$

Primero, convertimos las coordenadas de grados a radianes:

$$\varphi_1 = 37,7749^\circ \times \frac{\pi}{180} = 0,6593 \text{ rad} \quad (7)$$

$$\lambda_1 = -122,4194^\circ \times \frac{\pi}{180} = -2,1366 \text{ rad} \quad (8)$$

$$\varphi_2 = 34,0522^\circ \times \frac{\pi}{180} = 0,5943 \text{ rad} \quad (9)$$

$$\lambda_2 = -118,2437^\circ \times \frac{\pi}{180} = -2,0637 \text{ rad} \quad (10)$$

Luego, calculamos $\Delta\varphi$ y $\Delta\lambda$:

$$\Delta\varphi = \varphi_2 - \varphi_1 = 0,5943 - 0,6593 = -0,065 \quad (11)$$

$$\Delta\lambda = \lambda_2 - \lambda_1 = -2,0637 - (-2,1366) = 0,0729 \quad (12)$$

Aplicamos la fórmula de Haversine:

$$a = \sin^2\left(\frac{\Delta\varphi}{2}\right) + \cos(\varphi_1) \cdot \cos(\varphi_2) \cdot \sin^2\left(\frac{\Delta\lambda}{2}\right) \quad (13)$$

$$a \approx 0,0011 \quad (14)$$

$$c = 2 \cdot 2 \left(\sqrt{a}, \sqrt{1-a} \right) \quad (15)$$

$$c \approx 0,066 \quad (16)$$

Finalmente, multiplicamos por el radio de la Tierra:

$$d = 6371 \cdot c \approx 6371 \cdot 0,066 \approx 420 \text{ km} \quad (17)$$

Este resultado indica que la distancia entre las dos balizas es de aproximadamente 420 kilómetros. El uso más extendido de este principio matemático se observa en las rutas que trazan los aviones al volar, ya que la ruta más corta no siempre es una línea recta (debido a la propia geometría de la Tierra).

6

Implementación

La aplicación móvil ha sido desarrollada íntegramente en **Android Studio** utilizando **Kotlin** como lenguaje principal, en combinación con Java. Requiere Android 7.0 (API nivel 24) o superior para su correcta ejecución.

El proyecto integra múltiples bibliotecas gestionadas con Gradle, lo que facilita el desarrollo, pruebas y mejora la funcionalidad general. Algunas de las dependencias más relevantes son:

- **com.google.firebase:** Permite la integración de la base de datos en tiempo real Firestore de Firebase, lo que facilita el manejo de datos sin necesidad de un servidor propio.
- **org.osmdroid:** Se utiliza para integrar mapas dentro de la aplicación, con soporte para el servicio de OpenStreetMap.
- **junit** y **androidx.test.ext:** Bibliotecas utilizadas para realizar pruebas unitarias.
- **org.mockito:** Facilitan la creación de pruebas mediante la simulación de objetos y dependencias dentro de la aplicación.
- **robolectric:** Herramienta que permite realizar pruebas unitarias fuera del entorno Android real, simulando el comportamiento de la plataforma.

6.1. Preparación del entorno

En cuanto a la estructura del proyecto, se definen una serie de apartados en los cuales se agrupan las clases principales:

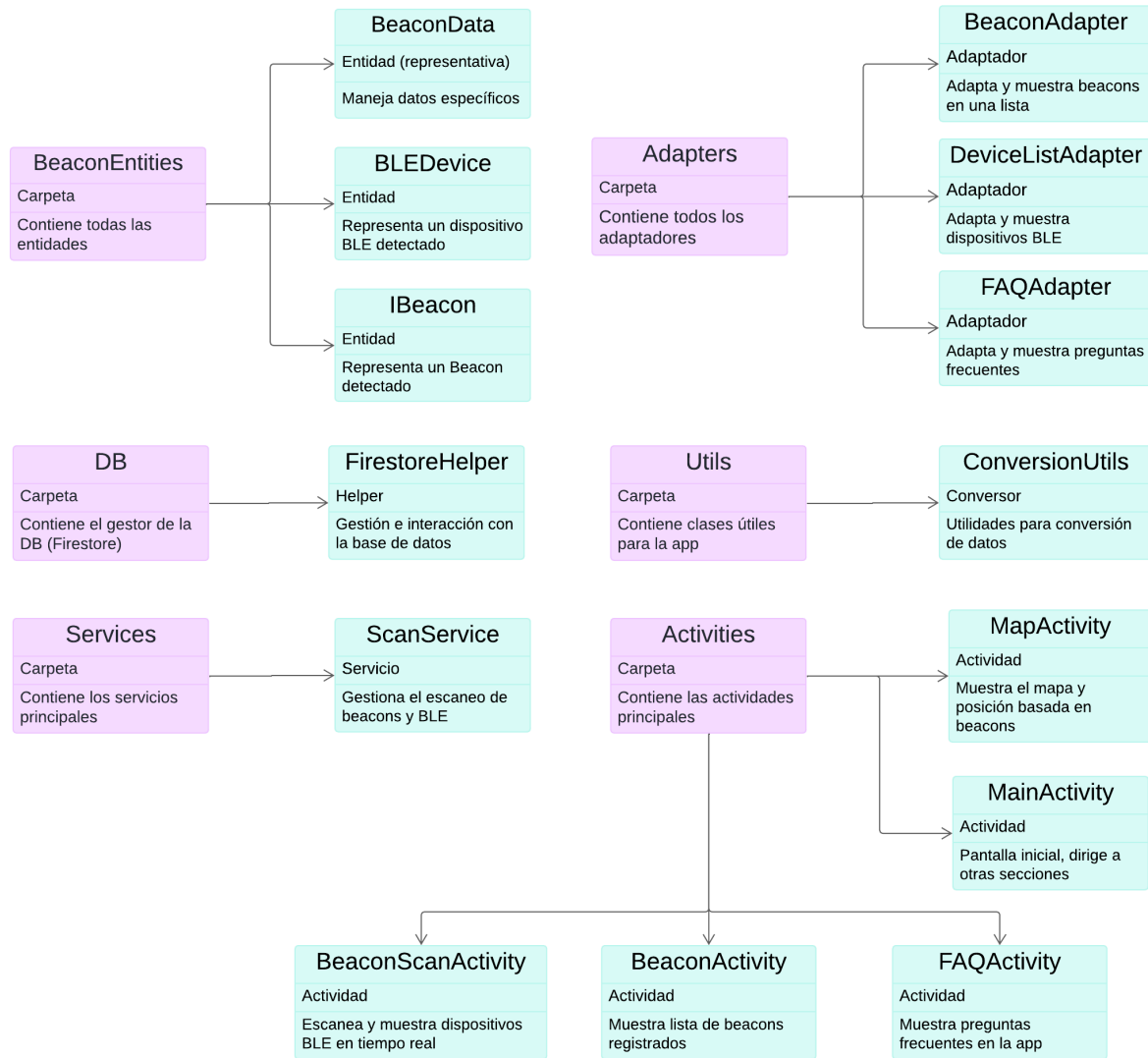


Figura 16: Esquema de clases del proyecto - Diagrama en LucidChart [23].

6.2. Entidades Beacon

Dentro de la carpeta *BeaconEntities* se encuentran 3 definiciones importantes que forman la base de la app. La primera es *BLEDevice*, que recibe el resultado del escaneo y está formada por los atributos *RSSI*, *Address* y *Name* (opcional). Este objeto es el que proporciona el cálculo de la distancia en función a la intensidad y forma el cuerpo de la siguiente clase.

La segunda es *IBeacon*, encargada de procesar el paquete de datos enviado por las balizas. Esta clase incluye la lógica para extraer y formatear el UUID del paquete recibido, como se muestra

a continuación:

Algorithm 4 Parseo del UUID

procedure PARSEUUID

$startByte \leftarrow 2$

while $startByte \leq 5$ **do**

if $(rawByteData[startByte + 2].toInt() \& 0xff == 0x02) \&\& (rawByteData[startByte + 3].toInt() \& 0xff == 0x15)$ **then**

$uuidBytes \leftarrow \text{ByteArray}(16)$

$\text{System.arraycopy}(rawByteData, startByte + 4, uuidBytes, 0, 16)$

$hexString \leftarrow \text{ConversionUtils.bytesToHex}(uuidBytes)$

if $!hexString.isNullOrEmpty()$ **then**

$uuid \leftarrow hexString.substring(0, 8) + "-" +$

$hexString.substring(8, 12) + "-" +$

$hexString.substring(12, 16) + "-" +$

$hexString.substring(16, 20) + "-" +$

$hexString.substring(20, 32)$

return

end if

end if

$startByte \leftarrow startByte + 1$

end while

end procedure

Por último, se define una clase adicional (*BeaconData*) con información del Beacon, la que se usa más adelante para mostrar la información en la sección de Beacons.

6.3. Adaptadores

Los adaptadores responsables de gestionar las listas de dispositivos y preguntas frecuentes están organizados en la carpeta *Adapters*. El *BeaconAdapter* maneja la visualización de la lista de beacons detectados en procesos previos, utilizando la clase *BeaconData*. Similarmente, el *DeviceListAdapter* sigue una lógica parecida pero se actualiza en tiempo real durante la detec-

ción de nuevos beacons. Finalmente, *FAQAdapter* organiza y presenta las preguntas frecuentes en un formato fácil de navegar.

6.4. Servicios

En este módulo destaca el servicio *ScanService*, clave en el proyecto por encargarse del escaneo de dispositivos Beacon y otros BLE. Su funcionalidad se organiza en los siguientes apartados:

1. **Declaración y Variables:** Se declaran las variables necesarias para manejar el escaneo de Bluetooth, la lista de dispositivos y la base de datos.
2. **Constructor:** Inicializa el *BluetoothManager*, el *BluetoothAdapter*, el *ScanSettings* y otros componentes necesarios.
3. **Inicializar escáner:** Distinguimos entre *initScanner* (inicializa la instancia del *BluetoothLeScanner*, que es el adaptador BLE del paquete *android.bluetooth*) y *startBLEScan*, donde el elemento más destacable es el modo de escaneo que se fija en *Low Latency*, con el que se obtiene el mayor ciclo de escaneo y es apto para trabajar en segundo plano.³
4. **Detener el escáner:** Con la función *stopBLEScan*, se detiene el escaneo y el envío de notificaciones, que informan al usuario cuando entra o sale del perímetro de una baliza.
5. **Callback del escáner:** Procesa los resultados del escaneo BLE y actualiza la lista de dispositivos. Este componente se encarga de procesar los resultados del escaneo de dispositivos Bluetooth Low Energy (BLE) y actualizar la lista de dispositivos detectados. Durante el escaneo, se analiza cada resultado recibido, y en función de ciertos criterios, se determina si corresponde a un beacon de interés. Si es así, se procesa la información relevante, como la distancia estimada, y se almacena o actualiza en la base de datos. Además, se monitorizan los cambios en el estado del beacon, como su proximidad, para notificar al usuario si se produce algún cambio significativo, como la entrada o salida de un dispositivo dentro de un rango determinado.

³Otros modos de escaneo pueden ser *Balanced* o *Low Power*

6. **Permisos:** Una de las partes cruciales de la app es la gestión de permisos. Para poder ejecutar correctamente el escaneo, se requieren permisos de BLUETOOTH (Admin, Scan y Connect), LOCATION (Fine y Coarse), INTERNET y POST-NOTIFICATIONS. Todos estos se declaran en el AndroidManifest y, en tiempo de ejecución, se solicitan los de Bluetooth (ver figura 17) y ubicación. De esta manera, se consigue que, al pulsar el botón de *Escanear*, se soliciten todos estos permisos si no están activados previamente:

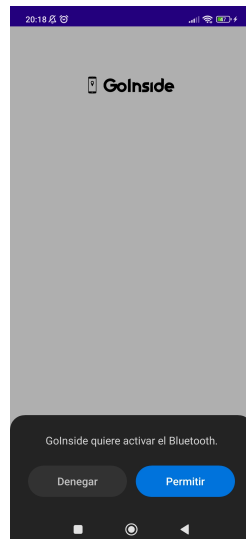


Figura 17: *Permisos* - Solicitar acceso a Bluetooth.

7. **Métodos Auxiliares:** Por último, una lista de funciones auxiliares indispensables para el correcto funcionamiento. *insertBeacon*: Inserta un beacon en la base de datos. *showNotification*: Muestra una notificación al usuario. *isIBeacon*: Verifica si un paquete es de un iBeacon. *isScanning*: Devuelve el estado del escaneo.

6.5. Actividades

El desarrollo de las Activities de la aplicación fue uno de los elementos fundamentales para su funcionamiento general. Durante el proceso, se enfrentaron múltiples retos, desde la integración de la detección de dispositivos BLE hasta la correcta visualización en un mapa interactivo. A continuación, se detallan las 5 principales actividades desarrolladas en el proyecto, los problemas encontrados y cómo fueron solucionados. Las actividades clave de la aplicación incluyen la **BeaconActivity**, que lista los beacons registrados en la base de datos, y la **Bea-**

conScanActivity, responsable del manejo de permisos, el escaneo de dispositivos BLE en tiempo real y su almacenamiento. Además, la **FAQActivity** ofrece una sección de preguntas frecuentes para asistir al usuario en la comprensión del funcionamiento de la app.

La **MapActivity**, fundamental en la aplicación, procesa las distancias a las balizas detectadas durante el escaneo. Utiliza la entidad *BeaconWithPosition*, compuesta por:

- **Nombre**
- **UUID**
- **Coordinate(latitud, longitud)**
- **Distance**

Además, se calcula la posición del usuario en relación a cada baliza con *calculatePosition*. Este método usa los cálculos mencionados con anterioridad en el apartado de Cálculo y aproximaciones y envía y actualiza las coordenadas del dispositivo al mapa. En este caso, se utiliza el mapa de *OpenStreetMap* [24] por ser colaborativo, de uso libre, con una cobertura prácticamente global, compatible con varios formatos y fácilmente de integrar con otros sistemas. Finalmente, la **MainActivity** contiene el comportamiento de cada uno de los elementos (botones) de la pantalla inicial. Éstos son los que derivan al usuario hacia cada una de las secciones de la app.

6.5.1. BeaconActivity

La **BeaconActivity** se encarga de mostrar una lista de beacons registrados en la base de datos de Firebase Firestore. Permite buscar beacons por su UUID o listar todos los registrados. Para la implementación de esta funcionalidad se utilizó un *RecyclerView* con un *Adapter* personalizado que muestra los datos de cada beacon.

- **Problema encontrado:** Inicialmente, surgieron problemas al obtener y actualizar la lista de beacons de Firestore en tiempo real, ya que la respuesta del servidor podía tardar y la interfaz no se actualizaba de inmediato.
- **Solución:** Se optó por utilizar *listeners* de Firestore y la función *notifyDataSetChanged()* en el *Adapter*, lo que permitió que la lista se actualizara dinámicamente a medida que se

obtenían los resultados. Además, se añadió un filtro para buscar por UUID y se mejoró el rendimiento al usar *orderBy* en la consulta de Firestore, lo que agilizó la búsqueda.

6.5.2. BeaconScanActivity

Esta actividad fue una de las más complejas de implementar, ya que maneja la detección en tiempo real de dispositivos BLE, el manejo de permisos, y la actualización de la interfaz mientras se escanean y registran los dispositivos. Algunos de los problemas encontrados se comentan a continuación:

- **Permisos en Android 12+:** Los permisos BLE cambiaron a partir de Android 12, lo que requería manejar los permisos de manera diferente para versiones más recientes del sistema operativo.
- **Escaneo en tiempo real:** Escanear dispositivos BLE de manera eficiente y sin consumir demasiados recursos fue un reto significativo.

Las soluciones propuestas se resumen en un manejo condicional de permisos, diferenciando entre versiones anteriores y posteriores a Android 12, permitiendo una compatibilidad amplia sin errores en la solicitud de permisos. Para optimizar el escaneo, se implementó un servicio en segundo plano (*ScanService*) que maneja el ciclo de vida del escaneo BLE. Esto permitió pausar o detener el escaneo cuando fuera necesario, optimizando el consumo de batería.

6.5.3. FAQActivity

La FAQActivity es una pantalla simple que muestra una lista de preguntas frecuentes para ayudar al usuario a entender mejor la aplicación. Fue fácil de implementar, ya que utiliza un diseño de preguntas con respuestas desplegadas.

- **Problema encontrado:** Un desafío menor fue hacer la lista accesible y clara para todo tipo de usuarios, asegurando que el texto fuese lo suficientemente grande y legible.
- **Solución:** Se utilizó un diseño accesible, con texto claro y botones grandes, además de asegurarse de que la pantalla cumpliera con los estándares de accesibilidad de Android.

6.5.4. MapActivity

La MapActivity es una de las funcionalidades más importantes de la aplicación. Esta pantalla permite visualizar la posición del usuario y de las balizas detectadas en un mapa interactivo utilizando la librería OpenStreetMap. Los principales problemas de este proceso y, sus soluciones, fueron los siguientes:

1. **Representación precisa en el mapa:** Mostrar las balizas y la posición del usuario en tiempo real requirió cálculos precisos de posicionamiento y actualización constante de la interfaz. Se desarrolló una función para calcular la dirección hacia una baliza específica, guiando al usuario con instrucciones claras (p.e., “Camina hacia la derecha”). Esta dirección se calculó basándose en la diferencia angular entre la posición del usuario y la baliza seleccionada.
2. **Trilateración:** Calcular la posición del usuario en función de las distancias a varias balizas fue un reto técnico importante, especialmente al trabajar con múltiples dispositivos simultáneamente. Para solventarlo, se implementó un sistema que permite mostrar las balizas en el mapa mediante marcadores, y se calculó la posición relativa del usuario utilizando la trilateración. Los cálculos de distancias se realizaron a través de la intensidad de la señal RSSI, y para evitar variaciones bruscas, se aplicó un filtro de media móvil para suavizar las distancias calculadas.

6.5.5. MainActivity

La implementación de la MainActivity fue sencilla en cuanto a la lógica de navegación entre pantallas. La pantalla principal tiene 4 botones que redirigen al usuario a las diferentes funciones: escaneo de beacons, mapa, listado de beacons y preguntas frecuentes. Cada botón ejecuta un *Intent* para abrir la actividad correspondiente.

- **Problema encontrado:** Un desafío inicial fue la gestión del botón de detener todos los procesos, ya que la aplicación cuenta con subprocesos ejecutándose en segundo plano (como el escaneo de BLE).
- **Solución:** Para detener todos los procesos, se implementó una función `stopAllProcesses` que finaliza los servicios activos y cierra la aplicación de forma controlada. Esto se hace

mediante el uso de *finishAffinity()*, que asegura que todos los procesos relacionados con la app finalicen correctamente.

En general, la implementación de las *Activities* del proyecto requirió resolver múltiples problemas relacionados con permisos, escaneo en tiempo real, y cálculos de posicionamiento. Se utilizaron varias librerías y técnicas de optimización para garantizar que la app funcionara eficientemente en diferentes dispositivos y versiones de Android.

6.6. Base de Datos

En una primera iteración del desarrollo de la aplicación, se implementó una clase llamada *SQLiteHelper*, la cual gestionaba la base de datos local utilizando SQLite. Esta clase permitía inicializar una base de datos local para almacenar los atributos de los beacons, realizar operaciones de inserción y actualización de registros, y obtener la lista de dispositivos detectados junto con sus distancias estimadas. Para la manipulación de los datos, se empleaba una estructura de datos como *ArrayList(Pair(String Uuid, Double distance))*, que facilitaba el cálculo de posiciones relativas. También se desarrollaron métodos para iterar sobre los registros y devolver listas de beacons en formato *BeaconData*, mostrando esta información en las distintas pantallas de la app. Además, se incluyó la funcionalidad para reiniciar la base de datos mediante el método *deleteDatabase*.

Sin embargo, en una segunda iteración, se decidió migrar la gestión de la base de datos a Firebase Firestore [12], lo que permitió aprovechar sus características avanzadas, como la sincronización en tiempo real, la escalabilidad y el manejo de datos offline. Esta transición mejoró la eficiencia de la aplicación y simplificó la gestión de datos, tanto localmente como en la nube. Las principales ventajas de Firestore sobre SQLite incluyen:

1. **Sincronización en tiempo real:** Firestore actualiza automáticamente los datos entre clientes y servidores, manteniendo la información siempre al día.
2. **Escalabilidad:** Se adapta al crecimiento de la base de datos y al aumento de la carga sin necesidad de intervención adicional.
3. **Fácil integración con Firebase:** Integra de manera sencilla otros servicios de Firebase, como la autenticación y las funciones en la nube.

4. **Consultas flexibles:** Permite realizar consultas complejas sin necesidad de escribir SQL, simplificando el acceso a los datos.
5. **Manejo de datos Offline:** Los datos están disponibles incluso sin conexión y se sincronizan automáticamente cuando el dispositivo vuelve a estar en línea.
6. **Seguridad:** Firestore proporciona reglas de seguridad basadas en el servidor, lo que permite un control preciso sobre el acceso y la manipulación de los datos.

En esta nueva implementación, la clase *FirestoreHelper* se encarga de gestionar todas las operaciones de base de datos, como la inserción y actualización de beacons, la obtención de dispositivos y sus distancias, la conversión de documentos de Firestore a objetos *BeaconData*, y el manejo de las interacciones con los beacons en tiempo real. Esta transición a Firestore ha mejorado considerablemente tanto la funcionalidad como la eficiencia de la aplicación, proporcionando una solución más robusta y escalable para el manejo de datos.

6.7. Útiles

Dentro de esta carpeta, la clase *ConversionUtils* contiene un método de conversión de bits a hexadecimal que se usa en el parseo de UUID anterior [4]. Éste es muy importante ya que ayuda a descomponer la trama obtenida de cada beacon.

6.8. Pantallas

Los *layouts* de la aplicación incluyen pantallas clave que permiten interactuar con las funciones principales. Cada una está diseñada para ser intuitiva y accesible, facilitando la navegación para todo tipo de usuarios.

6.8.1. MainActivity

La pantalla principal de la aplicación muestra las cuatro funcionalidades principales: escaneo de beacons, mapa, listado de beacons y preguntas frecuentes. Desde esta vista, el usuario puede acceder fácilmente a cada sección de la app según lo que necesite realizar. En la parte central, se observan cuatro botones grandes y accesibles que guían al usuario hacia las diferentes opciones (ver figura 35).

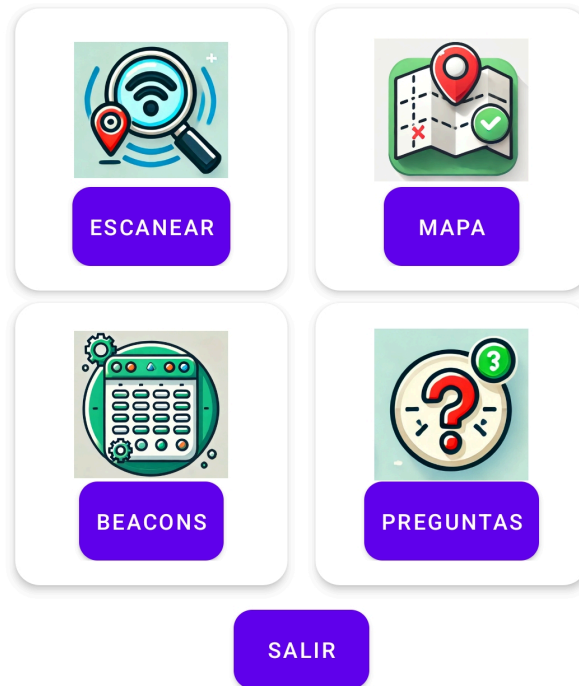
 Golnside

Figura 18: *MainActivity* - Pantalla inicial de la aplicación.

6.8.2. BeaconActivity

En esta pantalla, se lista todos los beacons registrados en la base de datos. Inicialmente, la lista se muestra vacía (ver figura 19), a la espera de ingresar algún UUID en el buscador superior. También se puede dejar vacío para mostrar todas las beacons. Se puede observar una lista con los datos relevantes de cada beacon, como su dirección, UUID y el número de

veces que ha sido detectado. Esto permite al usuario tener una vista completa de todos los dispositivos capturados por la aplicación (ver figura 20).



Figura 19: *BeaconActivity 1* - Listado de beacons registrados vacío.

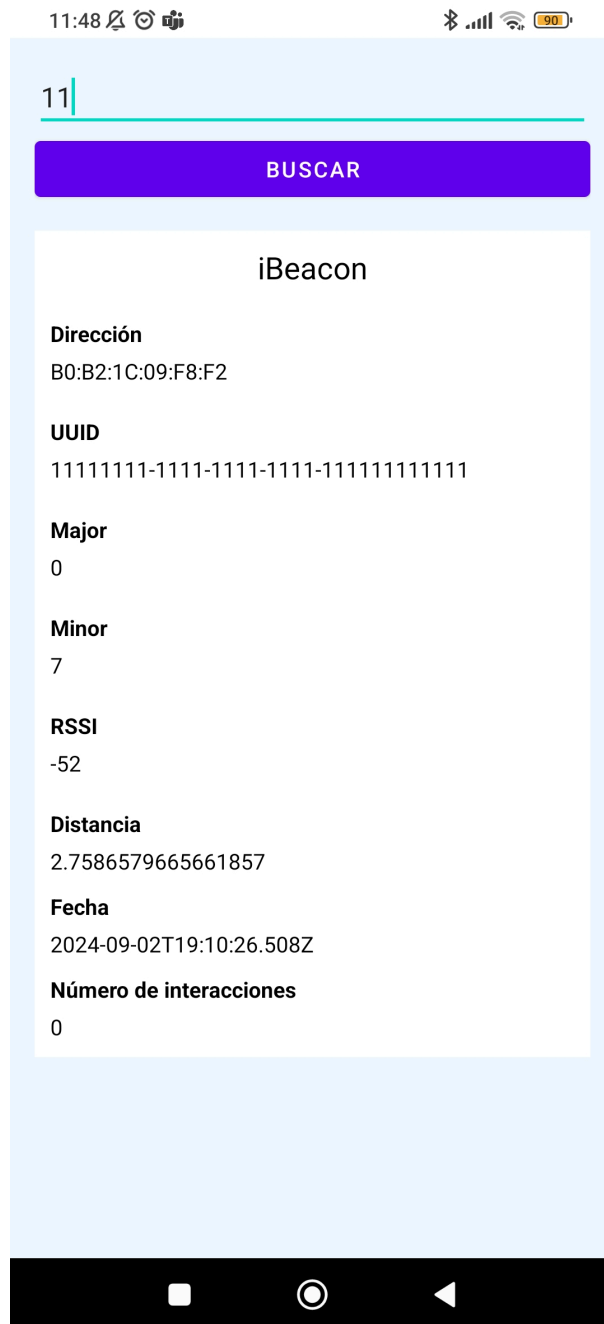


Figura 20: *BeaconActivity 2* - Listado de beacons registrados, en este caso, la que coincide con la búsqueda parcial.

6.8.3. BeaconScanActivity

La pantalla de escaneo muestra un botón para iniciar o detener el escaneo de dispositivos BLE y beacons. Mientras el escaneo está activo, la aplicación actualiza una lista en tiempo real con los dispositivos detectados, proporcionando información como la dirección del dispositivo, RSSI y la distancia aproximada (ver figura 21).



Figura 21: *BeaconScanActivity* - Pantalla de escaneo de dispositivos BLE y beacons.

6.8.4. MapActivity

Esta pantalla presenta un mapa que muestra tanto la posición actual del usuario como las posiciones de las beacons detectadas (marcadas con puntos rojos). También se incluye un buscador que permite al usuario introducir un destino. En el mapa, los usuarios pueden ver su ubicación relativa respecto a las beacons y cómo interactúan con ellas a medida que se mueven en el espacio.

- **Figura 22:** Muestra las opciones de visualización del mapa.

- **Figura 23:** Muestra el mapa con las beacons establecidas.

- **Figura 24:** Interacción con la baliza escrita en el buscador superior, al pulsar el botón *Ir* se resalta la ubicación de dicha baliza y se comienzan a dar indicaciones para llegar hasta ella.

- **Figura 25:** Mensaje que indica la llegada a la baliza seleccionada previamente.

- **Figura 26:** Mensaje que se muestra al pulsar el botón *Atrás* del dispositivo. Se da la opción de detener el servicio del mapa.

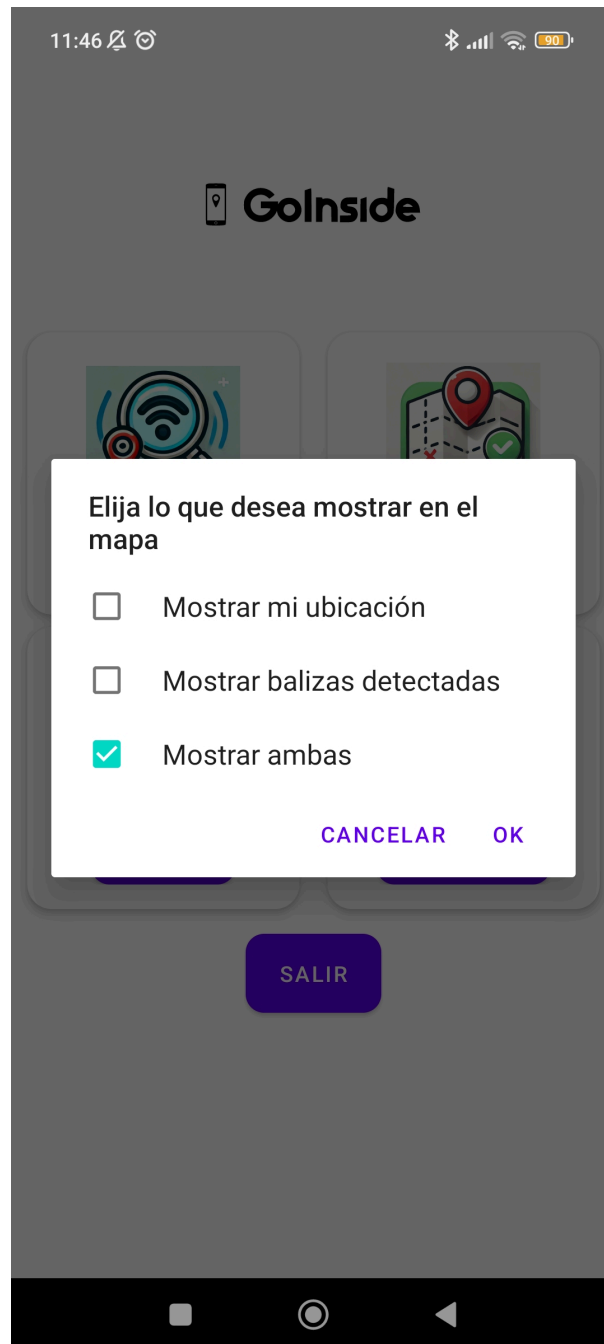


Figura 22: *MapActivity 1* - Vista previa al mapa.

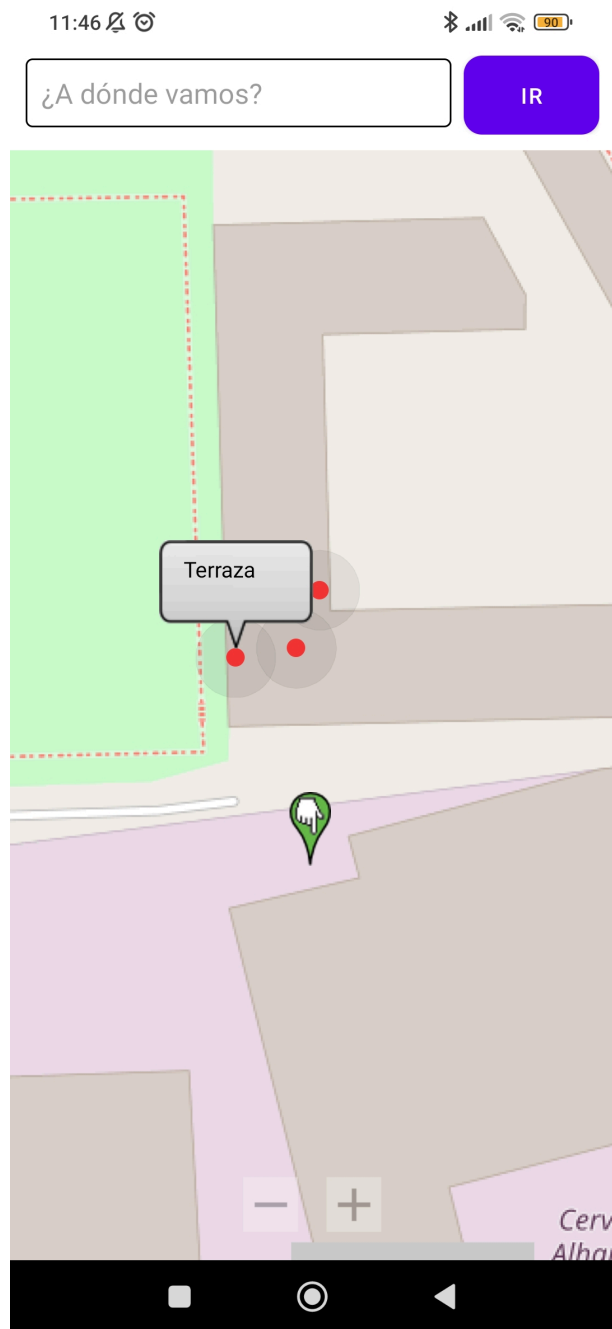


Figura 23: *MapActivity 2* - Mapa con beacons establecidas.

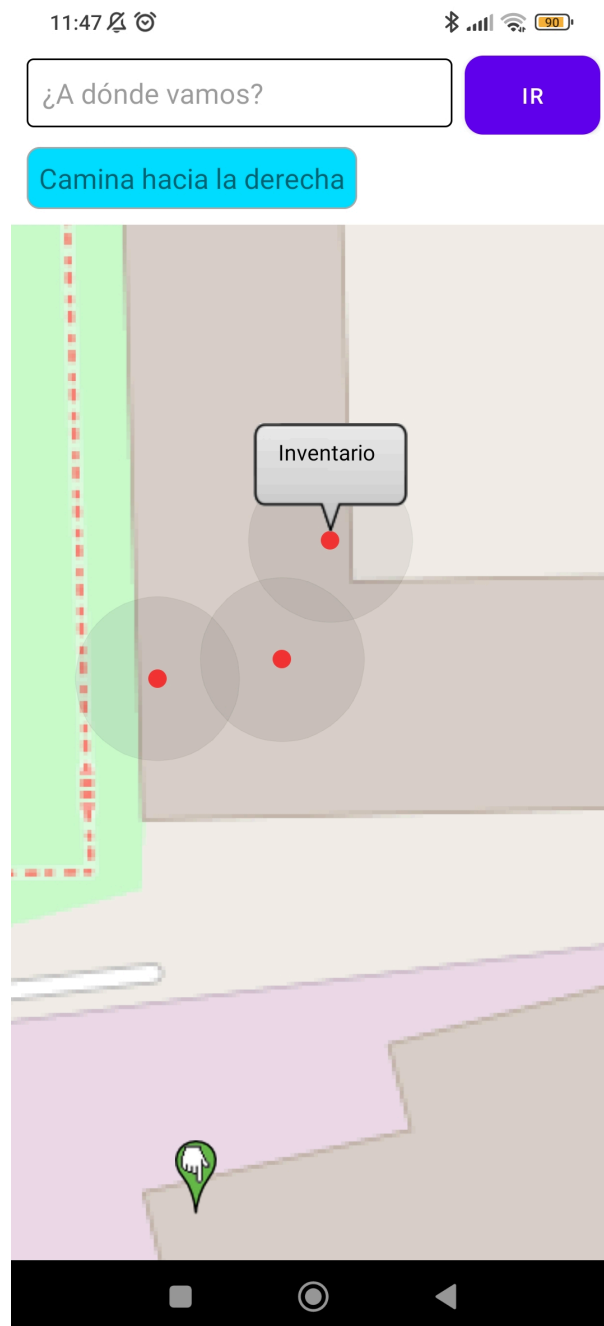


Figura 24: *MapActivity 3* - Mapa con balizas, ubicación del usuario e indicaciones.

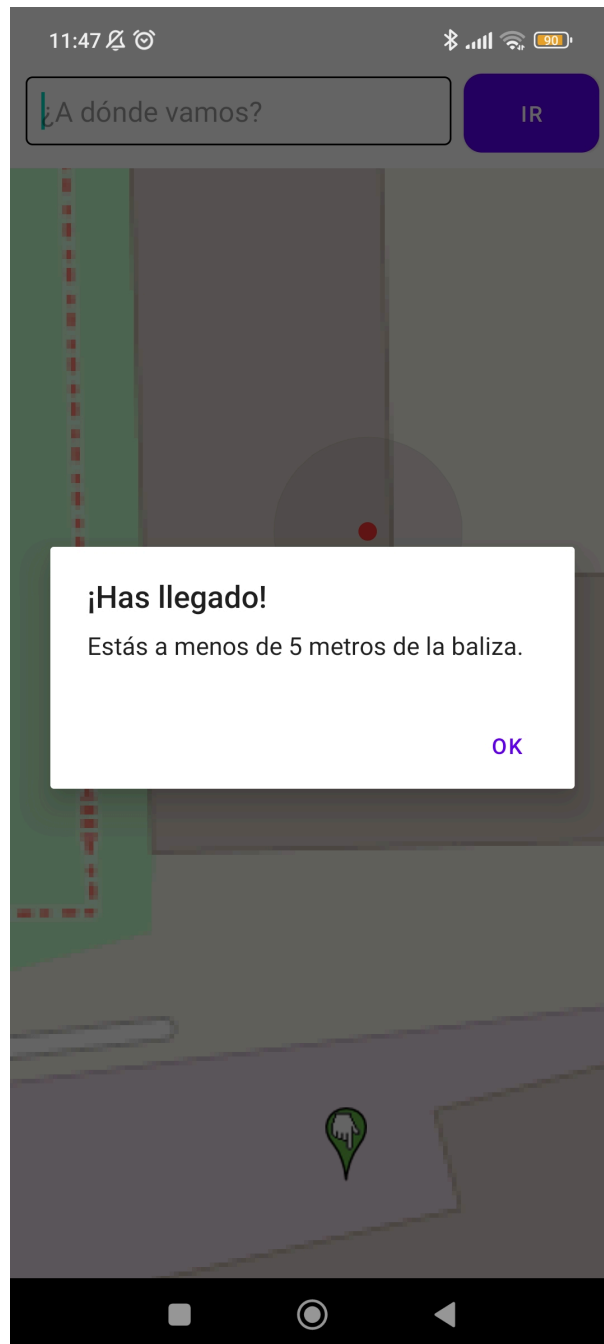


Figura 25: *MapActivity 4* - Mensaje lanzado al llegar a una baliza.

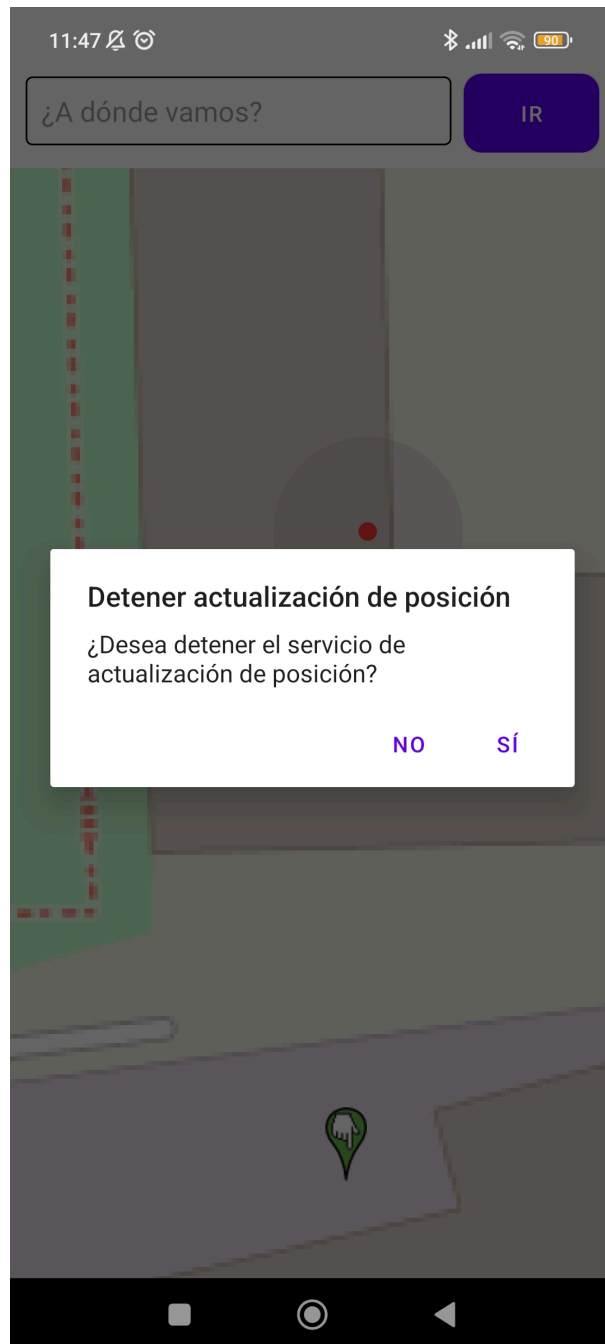


Figura 26: *MapActivity 5* - Mensaje con la opción de detener el servicio del mapa.

6.8.5. FaqActivity

En la vista de preguntas frecuentes (FAQ), se proporciona un listado de las consultas comunes del usuario, organizadas en elementos desplegables. Al hacer clic en una pregunta, se despliega la respuesta correspondiente, ayudando al usuario a resolver dudas frecuentes sobre el uso de la aplicación (ver figura 27).

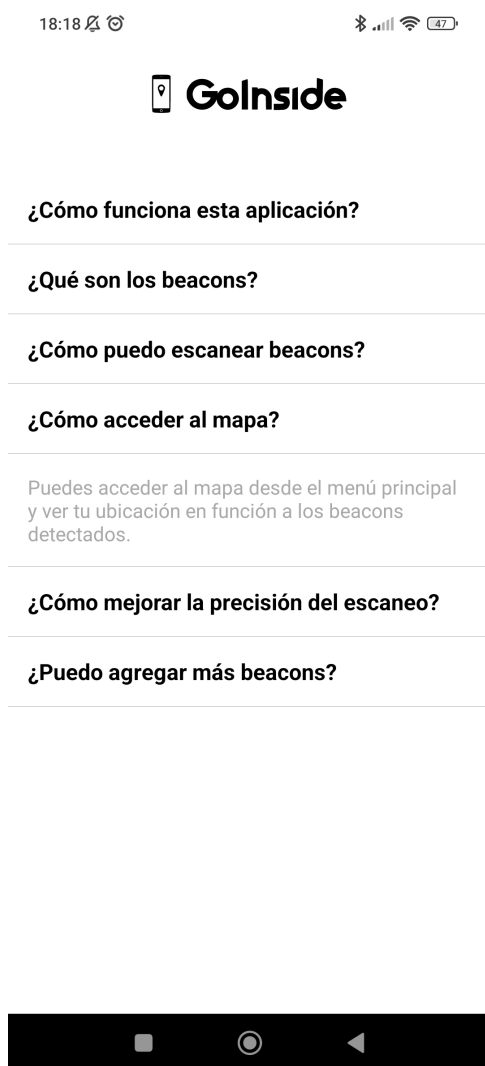


Figura 27: *FaqActivity* - Pantalla que contiene el listado de preguntas.

Para entender mejor el comportamiento de cada pantalla y su funcionalidad, se incluye en el Apéndice C un manual de usuario que facilita el uso de la aplicación.

Pruebas y depuración

7.1. Casos de prueba

En esta sección se describen los escenarios utilizados para validar las funcionalidades clave de la aplicación. Cada caso fue diseñado para asegurar que todos los componentes del sistema operan de manera correcta y que las integraciones entre ellos son funcionales. Se cubrieron los siguientes casos de prueba:

1. **Detección de balizas:** Se verifica que la aplicación es capaz de detectar todas las balizas dentro del alcance y registrar su información correctamente en la base de datos.
2. **Cálculo de distancia:** Se prueba que el cálculo de la distancia basado en la intensidad de la señal RSSI es preciso y se ajusta según los coeficientes definidos.
3. **Notificaciones de geofencing:** Se asegura que la aplicación envía las notificaciones correspondientes al usuario cuando entra o sale de una zona geográfica definida.
4. **Escaneo de múltiples dispositivos:** Se prueba que la aplicación puede manejar múltiples balizas simultáneamente sin pérdida de datos o fallas en el sistema.

7.2. Pruebas unitarias

Las pruebas unitarias son fundamentales para asegurar que cada componente de la aplicación funcione correctamente de manera aislada. Se implementan tests automatizados en las siguientes áreas clave:

1. **Cálculo de distancia:** Se desarrollan pruebas para validar las funciones matemáticas que calculan la distancia entre el dispositivo y las balizas basadas en RSSI. Encontramos las pruebas en el archivo *BLEDeviceTests*:
 - **testGetAddress():** Verifica que la dirección MAC del dispositivo es la esperada.
 - **testAddRssi():** Se afirma que los valores de RSSI se agregan correctamente y que el promedio se calcula adecuadamente.
 - **testCalculateRssi():** Calcula el valor promedio de RSSI de la lista.
 - **testGetDistance1() y testGetDistance():** Verifican el cálculo de la distancia en base a RSSI utilizando dos métodos diferentes.
 - **testRssiListSizeLimit():** Verifica que la lista de RSSI respete el tamaño límite definido y que el cálculo del promedio se realice correctamente después de eliminar valores antiguos.
2. **Almacenamiento en la base de datos:** Se verificó que la información recibida de las balizas se almacena correctamente en Firestore, garantizando que los datos son consistentes y se pueden recuperar sin errores. Dentro del archivo *DBUnitTests* destacamos *testInsertBLEDevice*, que verifica la correcta inserción y *testInsertDeviceInteraction* que gestiona las interacciones con el dispositivo
3. **Notificaciones:** Se probaron las funciones responsables de generar y mostrar notificaciones para asegurar que se activan bajo las condiciones correctas.
4. **Manejo de permisos:** Se realizaron pruebas para asegurarse de que la aplicación solicita y maneja adecuadamente los permisos necesarios en Android para acceder a Bluetooth y ubicación.

5. **Interacción con la interfaz de usuario:** Se validó que la información mostrada en la interfaz es precisa y se actualiza en tiempo real conforme los datos de las balizas cambian.
6. **Manejo de interrupciones:** Se probó cómo el sistema maneja interrupciones como la pérdida de conexión o la inactividad de las balizas, asegurando que la aplicación se recupera correctamente sin pérdida de datos.

7.3. Pruebas de integración

Las pruebas de integración se enfocan en analizar cómo interactúan las diferentes partes del sistema, con el objetivo de asegurar que todos los componentes trabajen correctamente en conjunto. Estas pruebas son cruciales para detectar problemas que podrían no ser evidentes en pruebas individuales de los módulos.

En nuestra aplicación, las pruebas de integración se centran principalmente en el análisis del comportamiento de las placas ESP32 dentro del sistema completo. Se busca evaluar cómo se comunican con el dispositivo móvil, específicamente en la recepción de la señal BLE y su uso para calcular distancias.

La principal prueba realizada, descrita en el Anexo A (Cálculo de distancias y calibración), permitió mejorar de manera significativa la precisión en el cálculo de distancias en función de la intensidad de la señal recibida desde las placas. Este proceso de calibración es fundamental para asegurar que la aplicación ofrezca resultados más fiables y precisos.

En una primera fase del proceso, se utilizó la aplicación *nRF Connect* para Android/iOS con el fin de verificar que la señal BLE se estaba emitiendo correctamente. Esta herramienta facilitó la detección de la señal iBeacon emitida por la ESP32, permitiendo visualizar los campos esenciales para la aplicación (como el UUID y la intensidad de la señal) y otros parámetros de configuración relevantes. En las pruebas, se activó el Bluetooth y la localización del dispositivo, como se puede ver en las imágenes 28 y 29:

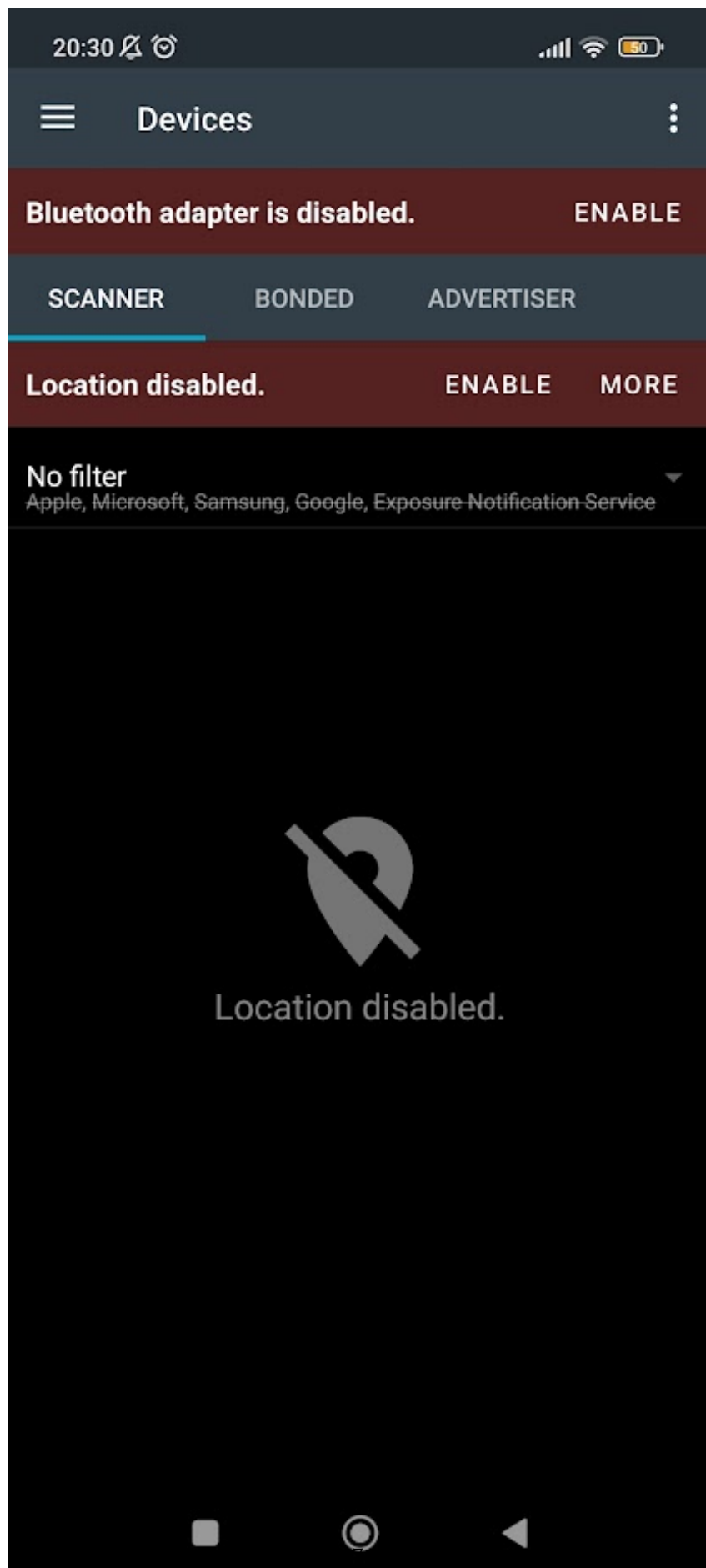


Figura 28: Captura con los permisos necesarios en **nRF Connect** para escanear.

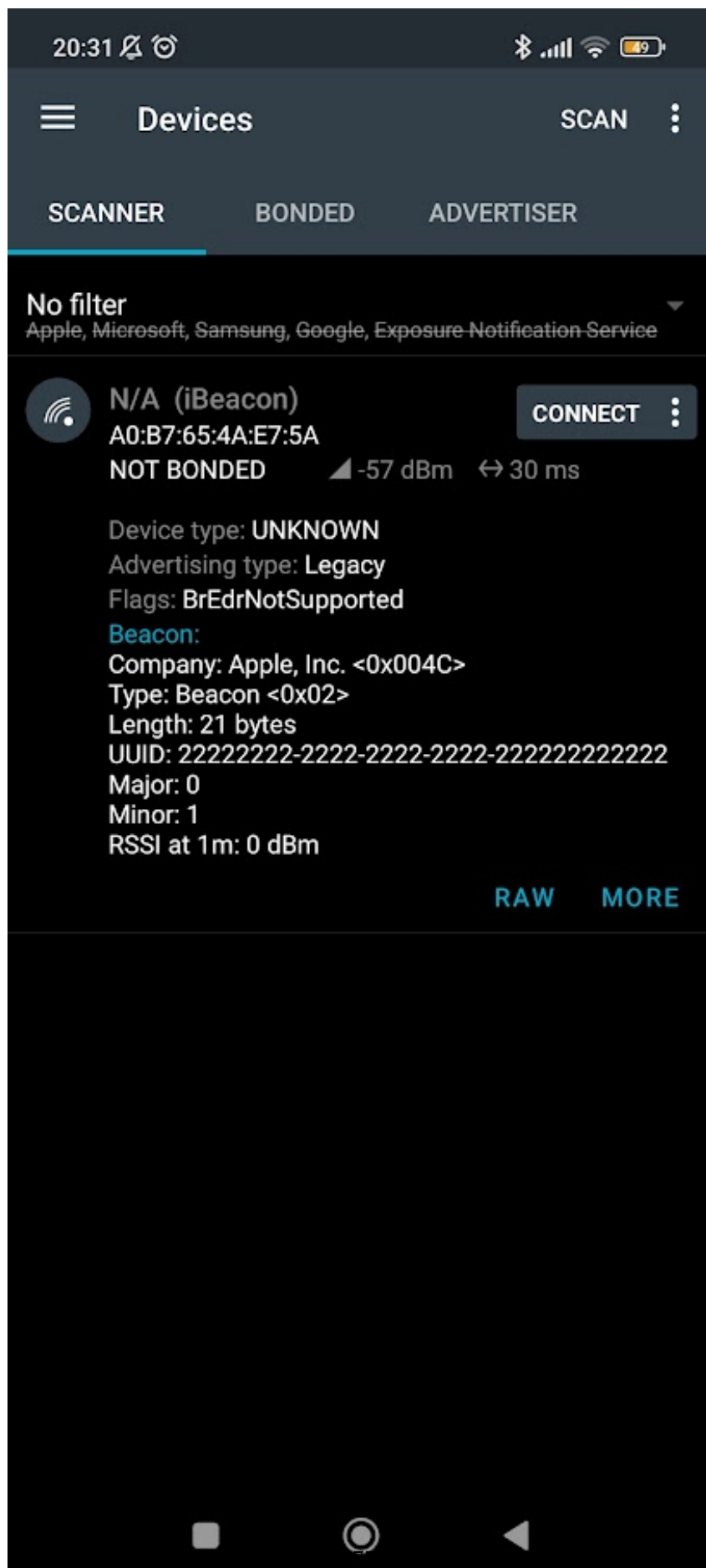


Figura 29: Captura con la detección de una baliza en **nRF Connect**.

Confirmar que los datos de publicidad se estaban transmitiendo correctamente y con la frecuencia deseada aseguró que la placa ESP32 está emitiendo la señal BLE de manera efectiva, y que los dispositivos móviles pueden detectar y procesar esta señal para la posterior utilización en la aplicación. Con el uso de esta aplicación gratuita también se verificó la detección de varias placas ESP32 simultáneamente para asegurar que la aplicación puede manejar múltiples emisores BLE sin problemas.

Otra de las pruebas del proyecto consistió en evaluar cómo los obstáculos e interferencias afectan a la señal recibida en diversos entornos. Para ello, se consideró la posibilidad de recalcular los coeficientes de atenuación de la señal utilizados en la calibración, dado que cada dispositivo capta la señal en condiciones específicas de tiempo, lugar y forma. El registro continuo de estos valores permitiría calcular la distancia de manera más precisa.

Sin embargo, esto no se implementó debido a la dependencia de la base de datos, la cual ocasionaba una ralentización significativa en el proceso, lo que representa una desventaja en aplicaciones de localización en tiempo real.

1. **Dispositivo móvil:** Recibe señales de múltiples balizas en diferentes entornos y calibra la distancia utilizando coeficientes A, B, y C específicos para cada baliza.
2. **Beacons:**
 - **Beacon 1:** Posicionada detrás de obstáculos, con coeficientes ajustados para reflejar la atenuación de la señal.
 - **Beacon 2:** Sin obstáculos inmediatos, con coeficientes que reflejan una propagación de señal directa.
3. **Interferencias y obstáculos:** Impactan los coeficientes de atenuación, requiriendo ajustes continuos para mantener la precisión.
4. **Base de datos:** Registra continuamente RSSI y cálculos de distancia, pero enfrenta problemas de rendimiento debido a consultas frecuentes para recalibración en tiempo real.

7.4. Identificación y corrección de errores

Durante el desarrollo y pruebas, se identificaron diversos errores, los cuales se documentaron y corrigieron siguiendo un proceso iterativo. Algunos de los principales errores encontrados incluyen:

1. **Problemas de sincronización con Firestore:** Se identificó un retraso en la sincronización de datos, que fue corregido mediante la optimización de las consultas.
2. **Inexactitud en el cálculo de distancias:** Como ya se ha mencionado en varias ocasiones, se observó que las distancias calculadas no eran precisas en ciertos entornos, lo que llevó a una recalibración de los coeficientes utilizados en el modelo de trilateración.
3. **Interrupciones en el escaneo BLE:** Se detectaron fallos en el escaneo continuo de balizas, especialmente en dispositivos con recursos limitados. Este problema fue resuelto optimizando el uso de memoria y ajustando los tiempos de escaneo.
4. **Manejo incorrecto de permisos en Android:** En algunos casos, la aplicación no solicitaba adecuadamente los permisos necesarios, lo que se solucionó mejorando la gestión de permisos en el flujo de la aplicación.

Conclusiones y Líneas Futuras

8.1. Conclusiones

En esta sección, se analizan los resultados obtenidos a partir del desarrollo de la aplicación, así como los posibles usos futuros, modificaciones y mejoras. Durante el desarrollo del proyecto se ha explorado y demostrado la viabilidad de utilizar balizas Bluetooth (beacons) para la localización en interiores, proporcionando una alternativa robusta cuando el GPS no es una opción viable. A través de la implementación de una aplicación móvil basada en Android, se ha logrado la **eficiencia** en el uso de energía: el módulo ESP32-WROOM-32 para la transmisión de señales BLE ha demostrado ser eficiente energéticamente, manteniendo un equilibrio entre el rendimiento y el consumo de batería.

En cuanto a la **precisión** en la localización: mediante técnicas de trilateración y el uso de algoritmos de procesamiento de señales, se alcanza una precisión aceptable en la estimación de la posición de los dispositivos móviles en entornos cerrados, contando con interferencias y otros impedimentos técnicos.

La **interfaz** de usuario se puede catalogar como intuitiva: la aplicación desarrollada ofrece una interfaz de usuario clara y funcional, que facilita la interacción y comprensión de las funcionalidades de localización.

A pesar de los logros alcanzados, también se identifican fácilmente ciertas limitaciones y áreas de mejora. La dependencia de la cantidad y disposición de las balizas afecta directamente a la precisión y fiabilidad del sistema y, el coste de despliegue puede ser considerable en grandes instalaciones.

8.2. Futuras líneas de trabajo

El desarrollo de este prototipo funcional ha demostrado la viabilidad de utilizar balizas BLE para la localización en interiores, resaltando su potencial para aplicaciones en una variedad de sectores. Aunque el sistema actual ofrece una funcionalidad básica sólida, existen varias áreas clave donde se podrían realizar mejoras para optimizar su precisión, eficiencia y escalabilidad. Con ciertas mejoras, este prototipo puede servir como base para futuras implementaciones más avanzadas. A continuación, se proponen las siguientes líneas de trabajo para su evolución:

1. **Guiado en espacios cerrados:** Una de las funcionalidades más innovadoras y útiles en el contexto de la localización en interiores es el guiado en espacios cerrados, similar a cómo Google Maps proporciona navegación en exteriores. Esta funcionalidad permite a los usuarios encontrar rutas óptimas dentro de grandes edificios como centros comerciales, hospitales, aeropuertos y oficinas.

Este guiado utilizaría una combinación de balizas BLE y mapas digitales de interiores para proporcionar indicaciones precisas para llegar a su destino. Esta tecnología es particularmente útil en lugares complejos y extensos, donde la orientación puede ser un desafío.

El sistema ofrecería una accesibilidad mejorada facilitando la movilidad de personas con discapacidades, proporcionando rutas accesibles o la optimización del tráfico peatonal, que ayuda a gestionar el flujo de personas, evitando aglomeraciones y mejorando la experiencia general.

2. **Optimización del coste de despliegue:** Investigar y desarrollar balizas más económicas sin comprometer la calidad de la señal, con el fin de reducir el coste total de implementación y algoritmos que optimicen la ubicación de las balizas para minimizar el número necesario sin perder precisión.
3. **Mejora en la precisión de localización:** Considerar una fusión de BLE con otras tecnologías, como Wi-Fi, UWB (Ultra-Wideband) o sensores inerciales (acelerómetro triaxial, giroscopio o magnetómetro) para una mejora en la precisión. La implementación de algoritmos de filtrado más sofisticados, como el filtro de Kalman, para suavizar las mediciones y reducir el ruido en los datos. En el apartado de la integración con otros

dispositivos BLE y sus respectivas direcciones MAC, se barajó la posibilidad de fijar la ubicación de dichos dispositivos para mejorar la precisión en el cálculo de distancias. Sin embargo, esto no se ha podido implementar debido a que fijar la ubicación de dispositivos que pueden estar en movimiento o que no tienen una posición definida impactaría negativamente en la precisión de las mediciones de distancia. Por tanto, se ha decidido no proceder con la fijación de ubicaciones y simplemente se menciona en la memoria como una consideración que no se ha llevado a cabo por las razones expuestas.

En el futuro, podrían desarrollarse nuevos algoritmos que permitan calcular las ubicaciones de manera más precisa, incluso en condiciones donde los dispositivos están en movimiento. Esto abriría la posibilidad de mejorar la integración y precisión en el cálculo de distancias sin los problemas actuales asociados a la fijación de ubicaciones.

4. **Seguridad y privacidad:** Desarrollar y aplicar técnicas de encriptación y anonimización de los datos de ubicación para garantizar la privacidad de los usuarios. También, se considera refinar la gestión de permisos y accesos para asegurar que los datos de los usuarios sean utilizados de manera responsable y con su consentimiento explícito.
5. **Expansión de funcionalidades:** Extender la compatibilidad de la aplicación a otras plataformas móviles, como iOS. El desarrollo de módulos específicos para diferentes industrias (salud, logística, etc) que aprovechen las capacidades de localización en interiores de manera más especializada.
6. **Pruebas y validación en entornos reales:** Realizar pruebas piloto en distintos entornos, como hospitales, centros comerciales y oficinas, para validar la eficacia del sistema en condiciones variadas. Obviamente, recoger y analizar el feedback de los usuarios finales para realizar mejoras continuas en la experiencia y funcionalidad de la aplicación.

Referencias

- [1] F. Javier Alejandro. *Errores en longitudes y desniveles medidos con GPS*. 2014. URL: <https://fjalejandro.blogspot.com/2014/03/erroresmedicionesGPS.html>.
- [2] Amazon. *Cámara de profundidad RGB-D*. 2024. URL: <https://www.amazon.com/-/es/profundidad-incluyendo-seguimiento-esquel%C3%A9tico-alternativa/dp/B08HKLKPHK>.
- [3] Agustin Bassi. *Intro a la tecnología Beacons*. 2021. URL: https://www.gotoiot.com/pages/articles/beacons_intro/content.html.
- [4] Evgeniy Berezovsky. *Java: convert a byte array to a hex string*. 2020. URL: <https://stackoverflow.com/a/9855338>.
- [5] Wikimedia Commons. *File:Jira Logo.svg*. 2024. URL: https://commons.wikimedia.org/wiki/File:Jira_Logo.svg.
- [6] Az-Delivery. *ESP32 NodeMCU Module*. 2021. URL: <https://www.az-delivery.de/en/products/esp32-developmentboard?variant=36542176914>.
- [7] DIGI. *Casos de uso y ejemplos de aplicación de Bluetooth Low Energy (BLE)*. 2024. URL: <https://es.digi.com/solutions/by-technology/bluetooth-standards#bluetooth-low-energy>.
- [8] Federico. *Branches*. 2024. URL: <https://github.com/FedeVico/BeaconDetection/branches>.
- [9] Federico. *Commits*. 2024. URL: https://github.com/FedeVico/BeaconDetection/commits/v_1_deteccion_beacons.
- [10] Federico. *Jira Backlog*. 2024. URL: <https://vicofede99.atlassian.net/jira/software/projects/GE0F/boards/1/backlog>.
- [11] Federico. *Jira Timeline*. 2024. URL: <https://vicofede99.atlassian.net/jira/software/projects/GE0F/boards/1/timeline>.
- [12] Firestore. *Firebase*. 2024. URL: <https://console.firebase.google.com/u/0/?hl=es>.

- [13] Grupo Fractalía. *Beacons, modo de funcionamiento, ventajas y potenciales aplicaciones*. 2014. URL: <https://fractaliasystems.com/beacons-modo-de-funcionamiento-ventajas-y-potenciales-aplicaciones/>.
- [14] Luis G. *Rango y precisión de los Beacons BLE*. 2023. URL: <https://www.linkedin.com/pulse/rango-y-precisi%C3%B3n-de-los-beacons-ble-luis-gonzalez/>.
- [15] Grdar. *Qué son los Beacons*. 2024. URL: <https://www.grdar.com/es/blog/que-son-los-beacons>.
- [16] EVE Museos + Innovación. *Como Funcionan los Beacons*. 2018. URL: <https://evemuseografia.com/2018/05/01/como-funcionan-los-beacons/>.
- [17] EVE Museos + Innovación. *Museos y Experiencia con iBeacons*. 2018. URL: <https://evemuseografia.com/2018/01/01/museos-y-experiencia-con-ibeacons/>.
- [18] IPMARK. *McDonald's elige el marketing de proximidad para promocionar el Mcflurry Conguitos*. 2021. URL: <https://ipmark.com/mcdonalds-elige-el-marketing-de-proximidad-para-promocionar-el-mcflurry-conguitos/>.
- [19] Izertis. *Beacons en Android*. 2017. URL: <https://www.izertis.com/es/-/blog/beacons-en-android>.
- [20] Jira. *Jira*. 2024. URL: <https://www.atlassian.com/software/jira>.
- [21] Neil Kolban. *BLE iBeacon*. 2018. URL: https://github.com/nkolban/ESP32_BLE_Arduino/blob/master/examples/BLE_iBeacon/BLE_iBeacon.ino.
- [22] Rafael Letts. *¿Cuál es la esencia de la tecnología?* 2023. URL: <https://sudaca.pe/noticia/opinion/rafael-letts-cual-es-la-esencia-de-la-tecnologia/>.
- [23] Lucid. *LucidChart*. 2024. URL: <https://www.lucidchart.com/pages/es>.
- [24] OpenStreetMap. *OpenStreetMap*. 2024. URL: <https://www.openstreetmap.org/#map=6/40.01/-2.49>.
- [25] Overleaf. *Overleaf*. 2024. URL: <https://www.overleaf.com/>.
- [26] PEUMALAB. *¿Qué es el Marketing de Proximidad (Proximity Marketing)?* 2018. URL: <https://peumalab.cl/blog-peumalab/50-que-es-el-marketing-de-proximidad-proximity-marketing.html>.

- [27] RedHat. *¿Qué es el Internet de las cosas (IoT)?* 2023. URL: <https://www.redhat.com/es/topics/internet-of-things/what-is-iot>.
- [28] José Segovia. *¿Quién inventó los mapas?* 2021. URL: <https://www.elcorreo.com/xlsemanal/historia/inventor-mapas-historia-griegos-ptolomeo-geografia.html?ref=https%3A%2F%2Fwww.elcorreo.com%2Fxlsemanal%2Fhistoria%2Finventor-mapas-historia-griegos-ptolomeo-geografia.html>.
- [29] Sensing. *Sensor de distancia por ultrasonidos rango ajustable*. 2024. URL: <https://sensores-de-medida.es/catalogo/sensor-de-distancia-por-ultrasonidos-rango-ajustable/>.
- [30] Moko Smart. *¿Qué es el protocolo y las especificaciones de Eddystone?* 2019. URL: <https://www.mokosmart.com/es/eddystone-protocol-and-specifications/>.
- [31] Moko Smart. *Balizas BLE frente a otras balizas RF: ¿Por qué elegir balizas Bluetooth de bajo consumo?* 2023. URL: <https://www.mokosmart.com/es/low-energy-bluetooth-beacons/>.
- [32] Sarah Stodola. *El primer país del mundo en usar GPS y cómo cambió la aviación mundial*. 2019. URL: <https://www.bbc.com/mundo/vert-tra-47831766#:~:text=Invento%20estadounidense,y%20de%20la%20ruta%20visual..>
- [33] Zap Tech. *Balizas: el secreto del marketing de proximidad*. 2024. URL: <https://zapt.tech/es/blog/venta-al-por-menor/baliza-el-secreto-del-marketing-de-proximidad/>.
- [34] The Valley. *Qué son los Beacons y cuál es su potencial*. 2024. URL: <https://thevalley.es/blog/que-son-los-beacons-y-cual-es-su-potencial/#:~:text=Los%20Beacons%20son%20peque%C3%B1os%20dispositivos,a%20la%20que%20se%20encuentran..>
- [35] Wikipedia. *Haversine formula*. 2024. URL: https://en.wikipedia.org/wiki/Haversine_formula#The_law_of_haversines.
- [36] Wikipedia. *iBeacon*. 2024. URL: <https://es.wikipedia.org/wiki/IBeacon>.
- [37] Wikipedia. *Trilateración*. 2022. URL: <https://es.wikipedia.org/wiki/Trilateraci%C3%B3n>.

- [38] Logos World. *GitHub Logo*. 2024. URL: <https://logos-world.net/github-logo/>.
- [39] David G. Young. *Calculating Formula Constants*. 2014-2023. URL: <https://altbeacon.github.io/android-beacon-library/distance-calculations2.html>.

Apéndice A

Cálculo de distancias y calibración

A.1. Primeras mediciones

Este primer paso consiste en registrar el valor de RSSI a una distancia fija de 1 metro entre el móvil y las balizas BLE. Este valor se toma como referencia para el resto de los cálculos y se utiliza para normalizar las mediciones a diferentes distancias. En este caso, se registró un valor de -53 dBm, que se utilizará como el valor base para calcular el ratio en el siguiente paso.

Móvil a 1m RSSI		
-53		

El ratio se calcula dividiendo el RSSI medido a diferentes distancias por el valor de referencia de -53 dBm (el RSSI a 1 metro). Este ratio nos indica cuántas veces cambia la potencia de la señal en relación con la distancia de referencia, permitiéndonos normalizar los valores y compararlos de manera más efectiva.

1. Distancia: Se registraron los valores de RSSI a diferentes distancias (desde 0.25 metros hasta 40 metros).
2. RSSI: Los valores de RSSI a cada distancia.
3. Ratio: Es el cociente entre el RSSI medido y el valor de referencia (RSSI a 1 metro).

El cálculo del ratio nos permite cuantificar cómo se degrada la señal a medida que aumentamos la distancia.

Distancia (m)	RSSI	Ratio
0.25	-46	0.867924528301887
0.5	-50	0.943396226415094

1	-53	1
2	-61	1.150943396226415
3	-62	1.169811320754717
4	-70	1.320754716981132
5	-75	1.415094339622642
6	-76	1.433962264150943
7	-77	1.452830188679245
8	-78	1.471698113207547
9	-80	1.509433962264151
10	-78	1.471698113207547
12	-80	1.509433962264151
14	-81	1.528301886792453
16	-82	1.547169811320755
18	-83	1.566037735849057
20	-84	1.584905660377358
25	-86	1.622641509433962
30	-86	1.622641509433962
40	-87	1.641509433962264

A.2. Preparación de los datos

Para obtener una relación matemática entre el ratio y la distancia, preparamos los datos para una regresión. En esta tabla, los valores de ratio (independientes) y las distancias medidas (dependientes) se organizan para ser utilizados en el modelo de regresión.

La regresión ayudará a encontrar una relación matemática entre el ratio y la distancia, permitiendo hacer predicciones sobre distancias no incluidas en los datos de entrenamiento.

Independent	Dependent
0.8679245283	0.25
0.9433962264	0.5
1	1
1.150943396	2

1.169811321	3
1.320754717	4
1.358490566	5
1.396226415	6
1.41509434	7
1.452830189	8
1.471698113	9
1.471698113	10
1.509433962	12
1.528301887	14
1.547169811	16
1.566037736	18
1.58490566	20
1.622641509	25
1.622641509	30
1.641509434	40

Para proceder con la obtención de A y B, se realiza una regresión lineal sobre los datos. Estos coeficientes se utilizan para predecir la distancia en función del ratio obtenido.

A	B
1.203420305	6.170094565

A.3. Prueba de predicción con A y B

En esta tabla, se evalúa la precisión del modelo de regresión utilizando los valores de A y B. Se comparan las distancias reales con las distancias predichas por el modelo (ver fórmula 18).

$$\text{Distancia predicha} = A \times x^B \quad (18)$$

Se puede observar que en algunos casos la predicción es cercana, mientras que en otros, especialmente a mayores distancias, hay desviaciones.

RSSI	Ratio	Actual Distance	Predicted Distance
-46	0.8679245283	0.25	0.2987432886
-50	0.9433962264	0.5	0.6365767302
-53	1	1	1
-61	1.150943396	2	2.661589253
-62	1.169811321	3	2.963946677
-70	1.320754717	4	6.493907433
-72	1.358490566	5	7.765313234
-74	1.396226415	6	9.233029585
-75	1.41509434	7	10.04784501
-77	1.452830189	8	11.8552198
-78	1.471698113	9	12.85452328
-78	1.471698113	10	12.85452328
-80	1.509433962	12	15.06230168
-81	1.528301887	14	16.27840117
-82	1.547169811	16	17.57464972
-83	1.566037736	18	18.95525472
-84	1.58490566	20	20.42458944
-86	1.622641509	25	23.64779668
-86	1.622641509	30	23.64779668
-87	1.641509434	40	25.41128494

A.4. Cálculo del coeficiente C

El valor C se introduce como una constante adicional para refinar la precisión de la fórmula obtenida a través de la regresión lineal, ajustando el modelo para mejorar las predicciones.

Se calcula la constante C que se utilizará en el siguiente paso para mejorar la precisión de las predicciones. El valor de C obtenido es -0.2034, y este valor se utiliza para ajustar la ecuación de predicción.

C	
---	--

-0.203420305	
--------------	--

A.5. Prueba de predicción final

Finalmente, se introduce el valor de C en la ecuación para mejorar las predicciones (ver fórmula 19).

$$\text{Distancia predicha} = A \times x^B + C \quad (19)$$

Al ajustar la predicción con C, se observa que la precisión mejora considerablemente para algunas distancias. Por ejemplo, para un RSSI de -46 dBm (Ratio: 0.8679) y una distancia real de 0.25 metros, la distancia predicha ahora es (ver fórmula 20):

$$\text{Distancia predicha} = 1,2034 \times 0,8679^{6,1701} - 0,2034 = 0,2987 \text{ metros} \quad (20)$$

Este ajuste proporciona una estimación más precisa, mejorando las predicciones en comparación con el modelo que solo utilizaba A y B.

RSSI	Ratio	Distancia real (m)	Distancia predicha (m)
-46	0.867924528301887	0.25	0.2987432886
-50	0.943396226415094	0.5	0.6365767302
-53	1	1	1
-61	1.150943396226415	2	2.661589253
-62	1.169811320754717	3	2.963946677
-70	1.320754716981132	4	6.493907433
-75	1.415094339622642	5	10.04784501
-76	1.433962264150943	6	10.92080783
-77	1.452830188679245	7	11.8552198
-78	1.471698113207547	8	12.85452328
-80	1.509433962264151	9	15.06230168
-83	1.566037735849057	18	18.95525472
-84	1.584905660377358	20	20.42458944
-86	1.622641509433962	25	23.64779668

-87	1.641509433962264	40	25.41128494
-----	-------------------	----	-------------

Apéndice B

Implementación del código de las balizas

B.1. Análisis del código

Este código en C++ sobre Arduino IDE 2.2.1 configura un dispositivo BLE (ESP32) para funcionar como un beacon, utiliza funciones de sueño profundo para ahorrar energía, y cuenta con algunas características adicionales. A continuación, se presenta una explicación detallada de cada parte del código.

B.2. Librerías necesarias

Se utilizan varias librerías para manejar el tiempo, las funcionalidades BLE, y el modo de sueño profundo del ESP32. Estas librerías proporcionan las funciones necesarias para configurar y manejar el beacon BLE, así como para gestionar el ahorro de energía del dispositivo.

```
18  #include "sys/time.h"
19
20  #include "BLEDevice.h"
21  #include "BLEServer.h"
22  #include "BLEUtils.h"
23  #include "BLEBeacon.h"
24  #include "esp_sleep.h"
```

Figura 30: Librerías empleadas en el código.

B.3. Declaraciones globales

Se define una duración para el sueño profundo de 10 segundos. Las variables `last` y `bootcount` almacenan el tiempo del último arranque y el número de veces que el dispositivo ha arrancado, respectivamente, en la memoria RTC, que persiste a través de los reinicios.

```
26 #define GPIO_DEEP_SLEEP_DURATION 10 // sleep x seconds and then wake up
27 RTC_DATA_ATTR static time_t last; // remember last boot in RTC Memory
28 RTC_DATA_ATTR static uint32_t bootcount; // remember number of boots in RTC Memory
```

Figura 31: Declaraciones globales empleadas en el código.

B.4. Variables para publicidad BLE y tiempo

Se declara un puntero `pAdvertising` para manejar las operaciones de publicidad BLE y una estructura `now` para gestionar el tiempo actual.

```
44 BLEAdvertising *pAdvertising;
45 struct timeval now;
```

Figura 32: Variables empleadas en el código.

B.5. UUID del Beacon

Se define un UUID estático para el beacon, que será utilizado para identificarlo de manera única. En nuestro caso, se han empleado los UUID 1, 2 y 3 por simplicidad (los 16 dígitos iguales).

```
47 #define BEACON_UUID "33333333-3333-3333-3333-333333333333"
```

Figura 33: Definición del UUID del Beacon empleado en el código.

B.6. Configuración del Beacon

La función `setBeacon` configura los parámetros del beacon, incluyendo el ID del fabricante, el UUID, y los valores mayor y menor. Luego, prepara los datos de publicidad y respuesta de escaneo para el beacon.

```
73 void setup() {  
74     Serial.begin(115200);  
75     gettimeofday(&now, NULL);  
76  
77     Serial.printf("start ESP32 %d\n", bootcount++);  
78  
79     Serial.printf("deep sleep (%lds since last reset, %lds since last boot)\n", now.tv_sec, now.tv_sec - last);  
80  
81     last = now.tv_sec;  
82  
83     // Create the BLE Device  
84     BLEDevice::init("");  
85  
86     pAdvertising = BLEDevice::getAdvertising();  
87  
88     setBeacon();  
89     // Start advertising  
90     pAdvertising->start();  
91     Serial.println("Advertising started...");  
92     delay(100);  
93     pAdvertising->stop();  
94     Serial.printf("enter deep sleep\n");  
95     esp_deep_sleep(1000000LL * GPIO_DEEP_SLEEP_DURATION);  
96     Serial.printf("in deep sleep\n");  
97 }
```

Figura 34: Configuración del Beacon empleado en el código.

B.7. Configuración inicial - setup

La función `setup` realiza las siguientes acciones:

1. Inicializa la comunicación serie.
2. Obtiene el tiempo actual y lo imprime junto con el número de arranques.
3. Inicializa el dispositivo BLE.
4. Configura y comienza la publicidad del beacon.
5. Después de un breve retraso, detiene la publicidad y entra en modo de sueño profundo por la duración especificada.

```

49 void setBeacon() {
50
51     BLEBeacon oBeacon = BLEBeacon();
52     oBeacon.setManufacturerId(0x4C00); // fake Apple 0x004C LSB (ENDIAN_CHANGE_U16!)
53     oBeacon.setProximityUUID(BLEUUID(BEACON_UUID));
54     oBeacon.setMajor((bootcount & 0xFFFF0000) >> 16);
55     oBeacon.setMinor(bootcount&0xFFFF);
56     BLEAdvertisementData oAdvertisementData = BLEAdvertisementData();
57     BLEAdvertisementData oScanResponseData = BLEAdvertisementData();
58
59     oAdvertisementData.setFlags(0x04); // BR_EDR_NOT_SUPPORTED 0x04
60
61     std::string strServiceData = "";
62
63     strServiceData += (char)26; // Len
64     strServiceData += (char)0xFF; // Type
65     strServiceData += oBeacon.getData();
66     oAdvertisementData.addData(strServiceData);
67
68     pAdvertising->setAdvertisementData(oAdvertisementData);
69     pAdvertising->setScanResponseData(oScanResponseData);
70
71 }

```

Figura 35: Configuración inicial (setup) usada en el código.

Apéndice C

Manual de uso

Este manual proporciona una guía detallada sobre el funcionamiento y uso de la aplicación de detección de beacons. Asegúrese de seguir cada paso cuidadosamente para maximizar el rendimiento y precisión del sistema.

C.1. Consultar las preguntas frecuentes

Antes de comenzar a utilizar la aplicación, se recomienda consultar la sección de *Preguntas*. Esta sección contiene respuestas a las dudas más comunes relacionadas con el uso de beacons y la aplicación. Acceda a esta sección desde el menú principal de la aplicación para obtener información esencial y solucionar problemas básicos (ver figura 27).

C.2. Escaneo de beacons

Para empezar a utilizar la aplicación, es necesario realizar un escaneo de beacons. Este proceso es crucial para detectar y triangular la posición de los dispositivos móviles con precisión. Siga estos pasos:

1. **Iniciar el escaneo:** Desde el menú principal, seleccione la opción de *Escanear*. La aplicación solicitará los permisos necesarios para acceder al Bluetooth y la ubicación del dispositivo. Asegúrese de conceder estos permisos para permitir un escaneo efectivo.
2. **Requerimientos:** La trilateración requiere la detección de al menos tres beacons para calcular una posición precisa. Asegúrese de que hay suficientes beacons activos y correctamente posicionados en el área de uso.
3. **Ayudas:** Se puede conocer el estado del dispositivo respecto a las balizas ya que se envían notificaciones *push*⁴ cuando estamos dentro del rango de acción de la baliza (5

⁴Mensajes que se envían de forma inmediata desde un servidor remoto hasta una aplicación, p.e. WhatsApp

metros).

4. **Finalización del escaneo:** Una vez que se hayan detectado los beacons necesarios, la aplicación mostrará un mensaje de confirmación. Regrese al menú principal para continuar con la próxima etapa.

C.3. Beacons

Entender el funcionamiento y los datos que se obtienen de las beacons es otro de los puntos principales de esta app. Por ello, esta sección es muy importante y ofrece la siguiente información:

1. **Datos propios de la beacon:** Como ya se comenta en anteriores secciones, toda la información de las balizas detectadas está dispuesta en esta funcionalidad. Si se desean obtener los datos de una baliza en concreto, se usará la barra de búsqueda en la parte superior. Si no se escribe nada y pulsamos el botón *Buscar*, nos devuelve todas las balizas encontradas.
2. **Datos externos de la beacon:** Adicionalmente, se muestra el campo *Fecha y Número de interacciones*, que reflejan la fecha y hora en la que se detectó la baliza y el número de dispositivos que ha interactuado con ella desde hace 24 horas.

Después de completar el escaneo de beacons y ver la información de las beacons, es hora de acceder al mapa y seleccionar cómo desea visualizar la información.

C.4. Acceso al mapa

Desde el menú principal, seleccione la opción *Mapa*. Se le presentarán tres opciones:

1. **Mostrar baliza detectadas:** Visualiza solo las posiciones de los beacons en el mapa.
2. **Mostrar mi ubicación:** Muestra solo la ubicación del dispositivo móvil.
3. **Mostrar ambas:** Muestra tanto las posiciones de los beacons como la ubicación del dispositivo móvil.

Una vez dentro del mapa, puede utilizar las funciones de navegación para dirigirse a su destino. Siga estos pasos para utilizar la funcionalidad de navegación:

1. **Ingresar destino:** En la parte superior de la pantalla del mapa, encontrará un cuadro de texto para ingresar el destino deseado. Escriba el nombre o la descripción del destino al que desea ir. La aplicación buscará coincidencias en los nombres de las balizas previamente escaneadas.
2. **Iniciar navegación:** Pulse el botón *Ir* junto al cuadro de texto. La aplicación calculará la ruta más adecuada hacia el destino y se mostrará en el mapa, proporcionando indicaciones para guiarlo hasta su destino.
3. **Seguimiento de la ruta:** Siga las indicaciones proporcionadas por la aplicación para llegar a su destino. La ubicación en tiempo real se actualizará continuamente en el mapa para asegurarse de que esté en la ruta correcta.
4. **Ayudas a la navegación:** Se incluyen pequeñas indicaciones en el seguimiento de la ruta para ayudar a encontrar el destino seleccionado. Al indicar un destino y pulsar *Ir*, aparecerá debajo de la barra de búsqueda un cuadro con indicaciones simples como, por ejemplo, *Camina hacia delante*. Si nos acercamos a menos de 5 metros de la baliza se lanza un pequeño pop-up en el que se indica que hemos llegado al destino.



UNIVERSIDAD
DE MÁLAGA

| **uma.es**

E.T.S. DE INGENIERÍA INFORMÁTICA

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