

UNMDP-FI

Carrera: Ingeniería en Computación

Sistema de Localización en interiores

Informe final



Camila Baldomá, Constanza Maña

Directores: Rodrigo Russo y Walter Andrés Gemin

Contenido del documento

Resumen.....	3
Capítulo 1.....	4
Introducción.....	4
1.1. Marco contextual.....	4
1.2. Objetivo general y objetivos específicos.....	5
1.3. Alcance y exclusiones.....	5
1.4. Fundamentos necesarios.....	6
1.5. Organización del informe.....	7
Capítulo 2.....	7
Anteproyecto.....	7
2.1. Análisis de soluciones.....	7
2.2. Arquitectura general elegida.....	8
2.3. Requerimientos.....	9
2.3.1. Requerimientos funcionales.....	10
2.3.2. Requerimientos no funcionales.....	11
2.4. Plan de trabajo.....	12
Capítulo 3.....	13
Diseño e implementación del sistema.....	13
3.1. Subsistema físico.....	13
3.1.1. Implementación de una baliza BLE.....	13
3.1.2. Diseño del prototipo.....	15
3.1.3. Limitaciones y problemas detectados.....	17
3.1.4. Costos del prototipo.....	18
3.2. Estimación de la posición.....	19
3.2.1. Métodos basados en propagación.....	20
3.2.2. Otras técnicas analizadas.....	22
3.2.3. Evaluación en el entorno real.....	25
3.3. Persistencia y modelo espacial.....	26
3.3.1. Elección del motor de Base de Datos.....	28
3.4. API.....	28
3.4.1. Arquitectura de software y estilo RESTful.....	29
3.4.2. Despliegue.....	30
3.4.3. Autenticación y seguridad.....	30
3.5. Aplicación móvil.....	32
3.5.1. Navegación.....	32
3.5.2. Cálculo de la ruta más corta.....	33
3.5.3. Interacción con el usuario y experiencia de búsqueda.....	34
3.5.4. Evaluación en entorno real y problemas detectados.....	35
3.6. Aplicación de gestión (administrador).....	36
3.6.1. Arquitectura de software.....	37
3.6.2. Funcionalidades específicas.....	38
Capítulo 4.....	41

Verificación y resultados.....	41
4.1. Estrategia y ambiente de pruebas.....	41
4.2. Tipos de pruebas.....	42
4.3. Resultados obtenidos.....	44
Capítulo 5.....	45
5.1. Conclusiones generales.....	45
5.2. Trabajos futuros.....	46
Bibliografía.....	46
Apéndice.....	49
A1. Documentos anexos.....	49
Plan de proyecto.....	49
Especificación de requerimientos.....	49
Especificaciones funcionales.....	49
Especificaciones técnicas.....	49
Plan de pruebas.....	49
Casos de prueba.....	49
Implementación de los códigos.....	49

Resumen

El proyecto surge ante la necesidad de contar con sistemas de localización en interiores precisos y accesibles, dado que las tecnologías tradicionales, como el GPS, presentan limitaciones en espacios cerrados. Grandes establecimientos han manifestado la problemática de que las personas se pierden o les cuesta orientarse dentro de ellos, ocupando gran parte de su tiempo en llegar a donde deben.

El presente trabajo plantea un Sistema de Localización en Interiores como una solución tecnológica diseñada para proporcionar orientación y navegación en entornos cerrados y de gran extensión, como hospitales, universidades, centros comerciales o museos.

El proyecto consiste en la implementación de balizas Bluetooth Low Energy (BLE), mediante el uso de dispositivos ESP32, que emiten señales cuya intensidad (RSSI) es procesada para estimar la posición del usuario en tiempo real.

El sistema desarrollado está compuesto por un prototipo de baliza BLE con ESP32 que integra un Sistema de Alimentación Ininterrumpida (SAI) con batería de litio (6800 mAh, 3.7 V) [40]; una aplicación móvil para dispositivos Android, "InMap", que permite la visualización de la ubicación actual del usuario sobre un mapa digital del espacio y calcula la ruta más corta hacia el destino seleccionado; una aplicación web de administrador para gestionar y mantener actualizados los datos del espacio; una base de datos para persistir la información y los datos espaciales; y una API, que permite la comunicación entre las aplicaciones y la base de datos.

El método finalmente utilizado para estimar la posición del usuario fue el algoritmo del Centroide Ponderado (WCL), implementando filtros para adaptar el cálculo a la geometría del entorno. Para calcular la ruta más corta al destino, a partir de la posición estimada, se utilizó el algoritmo A*.

Las pruebas realizadas en el segundo piso de la Facultad de Ingeniería permitieron verificar el funcionamiento integrado de los distintos componentes y validar la capacidad del sistema para estimar la posición del usuario, representar su ubicación y guiarlo hacia un destino seleccionado. Los resultados obtenidos demuestran la viabilidad de la solución para el propósito planteado y establecen una base funcional para futuras mejoras y su adaptación a otros espacios interiores.

Capítulo 1

Introducción

1.1. Marco contextual

La idea del proyecto se originó a partir de una necesidad planteada por el Hospital Italiano de Buenos Aires. Debido a la extensión y concurrencia del establecimiento, sus visitantes pueden tener dificultades para orientarse y encontrar el lugar al que deben dirigirse. Esta problemática no es exclusiva del ámbito hospitalario, sino que también se presenta en universidades, museos, centros comerciales y otros espacios culturales o recreativos de grandes dimensiones.

En estos entornos surge la necesidad de disponer de sistemas de localización y navegación interior que sean accesibles y adaptables a las características de cada establecimiento. Las tecnologías tradicionales de posicionamiento satelital, como el GPS, pierden precisión o disponibilidad en espacios cerrados debido a la atenuación y distorsión de las señales producidas por techos, paredes y otras estructuras [2],[10].

Bluetooth Low Energy constituye una de las tecnologías utilizadas para brindar servicios basados en proximidad y ubicación dentro de espacios complejos. Por ejemplo, algunos parques y establecimientos recreativos (Disney World) emplean balizas BLE para que sus aplicaciones y dispositivos portátiles activen contenidos, efectos o experiencias según la ubicación aproximada del visitante [7]. Estos antecedentes no representan necesariamente sistemas de navegación cartográfica equivalentes al propuesto, pero demuestran la posibilidad de utilizar BLE como infraestructura para desarrollar servicios contextuales en interiores.

Si bien existen soluciones comerciales de localización interior, muchas de ellas requieren infraestructura específica, equipamiento propietario o inversiones que pueden dificultar su adopción. El presente proyecto no busca reproducir un producto comercial existente, sino estudiar la adaptación e integración de tecnologías disponibles para construir una solución propia, reproducible y potencialmente más económica. Para ello, se aprovechan las capacidades de comunicación BLE y Wi-Fi incorporadas en los dispositivos ESP32, componentes ampliamente disponibles y de costo relativamente bajo [18].

En este contexto, el trabajo se orienta a evaluar la viabilidad técnica y económica de implementar un sistema de localización y navegación interior basado en ESP32. La evaluación comprende tanto su capacidad para estimar la posición y guiar al usuario dentro del entorno de prueba como los recursos, componentes e infraestructura necesarios para su implementación.

1.2. Objetivo general y objetivos específicos

El objetivo general del proyecto es diseñar, implementar y validar un prototipo de sistema de localización y navegación en interiores basado en balizas Bluetooth Low Energy implementadas con dispositivos ESP32, capaz de estimar la posición de un usuario, representarla sobre un mapa digital y guiarlo hacia un destino seleccionado dentro de un entorno de prueba mediante una aplicación móvil.

Dentro de los objetivos específicos se encuentran:

1. **Diseñar e implementar un prototipo de baliza BLE** mediante dispositivos ESP32, capaz de emitir información identificadora y funcionar de manera autónoma mediante un sistema de alimentación con respaldo de batería.
2. **Desarrollar y evaluar un método de estimación de posición** a partir de las intensidades de las señales recibidas desde las balizas distribuidas en el entorno.
3. **Construir una representación digital navegable del espacio**, contemplando recintos, destinos, caminos transitables, obstáculos y zonas restringidas.
4. **Desarrollar una aplicación móvil para dispositivos Android** que permita obtener las señales BLE, representar la posición estimada del usuario, seleccionar un destino y proporcionar indicaciones para llegar hasta él.
5. **Implementar un método de cálculo y actualización de rutas** que determine un recorrido transitable desde la posición estimada hasta el destino seleccionado y responda ante cambios en el entorno o desvíos del usuario.
6. **Diseñar un mecanismo centralizado de almacenamiento y gestión de la información** necesaria para el funcionamiento del sistema, incluyendo datos espaciales, destinos, personas, horarios, usuarios y dispositivos físicos.
7. **Desarrollar una aplicación de administración** que permita consultar y mantener actualizada la información del espacio, gestionar los datos asociados y monitorear el estado operativo de las balizas.
8. **Integrar los componentes desarrollados y evaluar el funcionamiento del sistema** mediante pruebas técnicas y funcionales en el entorno seleccionado.
9. **Estimar los recursos y costos asociados con la construcción del prototipo**, identificando los principales factores que afectarían su implementación en otros espacios interiores.

1.3. Alcance y exclusiones

El proyecto comprende el desarrollo y la validación de una aplicación de localización y navegación interior para dispositivos Android, evaluada en el segundo piso de la Facultad de Ingeniería. El sistema utiliza diez balizas BLE implementadas con ESP32 y permite estimar y visualizar la posición del usuario, buscar destinos y calcular rutas. Incluye además una aplicación web para actualizar la información del espacio y monitorear las balizas, una base de datos espacial y una API para centralizar el acceso a los datos y la lógica de negocio. Su funcionamiento requiere Bluetooth en el dispositivo móvil y conectividad de red para acceder al servidor; las balizas utilizan Wi-Fi para enviar reportes de estado.

Quedan fuera del alcance el posicionamiento exterior mediante GPS, la estimación de posición mediante Wi-Fi, la compatibilidad con iOS, la actualización remota del firmware de las balizas, el análisis histórico de circulación y el despliegue productivo en un establecimiento completo. Los resultados corresponden a un prototipo validado en el entorno de prueba y no a un producto comercial terminado.

1.4. Fundamentos necesarios

Bluetooth es una tecnología de comunicación inalámbrica de corto alcance que opera en la banda de 2.4 GHz. Mientras que Bluetooth clásico está orientado al intercambio continuo de datos, **Bluetooth Low Energy (BLE)** fue diseñado para transmitir pequeñas cantidades de información mediante comunicaciones breves y periódicas. Su funcionamiento sencillo, su disponibilidad en dispositivos móviles y la posibilidad de detectar transmisiones sin establecer conexiones permanentes lo hacen apropiado para implementar sistemas basados en balizas [17].

Las balizas BLE operan principalmente en modo de **publicidad (advertising)**, mediante el cual transmiten periódicamente paquetes que pueden ser detectados por dispositivos cercanos sin establecer una conexión permanente. En este proyecto se adoptó el formato **iBeacon**, que incorpora un identificador UUID para distinguir el sistema o conjunto de balizas, y los campos Major y Minor para identificar agrupaciones y dispositivos individuales [16]. También incluye el valor Tx Power, que representa una referencia de la potencia recibida a una distancia conocida y puede emplearse para aproximar la distancia entre el emisor y el receptor.

Cuando un teléfono detecta una baliza, registra la intensidad con la que recibe su señal mediante el indicador **RSSI** (Received Signal Strength Indicator), expresado en dBm. En términos generales, cuanto menor es la distancia entre ambos dispositivos, mayor es la intensidad recibida. Sin embargo, esta relación no es directa ni constante, ya que la propagación de las señales se ve afectada por paredes, objetos, personas, reflexiones, interferencias y por la orientación relativa del teléfono y las balizas. Por esta razón, las distancias obtenidas a partir del RSSI constituyen estimaciones sujetas a variaciones y requieren mecanismos de filtrado y estabilización [10], [13].

Existen diferentes técnicas para estimar una posición a partir de estas mediciones. La **trilateración** calcula las coordenadas del receptor utilizando las posiciones conocidas de varias balizas y las distancias estimadas respecto de ellas. Su implementación no requiere construir previamente un mapa de señales, aunque su desempeño depende de la precisión con la que puedan obtenerse las distancias a partir del RSSI. En cambio, el **método de fingerprinting** se basa en registrar previamente las intensidades recibidas en distintos puntos conocidos para construir un mapa de radio. Durante el funcionamiento, las mediciones actuales se comparan con ese conjunto de referencia para determinar la posición más probable [8], [13].

En el desarrollo se tomó como base el estudio de estos métodos clásicos y, tras atravesar etapas de prueba con diferentes modelos, se determinó adoptar el algoritmo de **Centroide Ponderado (WCL)** como método principal. Esta solución, que promedia las ubicaciones de las balizas en función de su potencia de señal, fue complementada con técnicas de filtrado y optimización destinadas a reducir la variabilidad del RSSI en el entorno real [14]. Los fundamentos, ajustes y resultados de esta solución se desarrollan posteriormente en el apartado correspondiente a la estimación de la posición.

1.5. Organización del informe

En cuanto a la organización del presente informe, el mismo se estructura de la siguiente manera:

- Capítulo 1: Presenta la introducción y el marco contextual del proyecto.
- Capítulo 2: Describe el anteproyecto, incluyendo los requerimientos, especificaciones funcionales y restricciones.
- Capítulo 3: Expone el diseño e implementación del sistema en su conjunto.
- Capítulo 4: Explica las pruebas realizadas y expone los resultados obtenidos.
- Capítulo 5: Se exponen las conclusiones del trabajo realizado.

Capítulo 2

Anteproyecto

2.1. Análisis de soluciones

La utilización de balizas BLE implementadas con ESP32 formó parte de la definición inicial del proyecto, por lo que no se evaluaron tecnologías alternativas para la emisión de señales. El análisis de soluciones se concentró en decisiones particulares de implementación: el sistema de alimentación de las balizas, el método de estimación de posición, el algoritmo de cálculo de rutas y la tecnología de persistencia. Esta información se encuentra resumida en la Tabla 2.1

Tabla 2.1. *Decisiones de implementación.*

Aspecto	Alternativas analizadas	Solución seleccionada	Criterio de elección
Gestión de alimentación y batería	Módulo 18650 Battery Shield y circuito basado en el chip HT4928S	18650 Battery Shield	Integra carga, protección, conmutación y elevación de tensión, simplificando la construcción y reproducción del prototipo.
Estimación de posición	Métodos basados en propagación (Trilateración y Centroide Ponderado) y Métodos basados en mapas de radio (Fingerprinting).	Algoritmo de Centroide Ponderado (WCL) restringido por la topología del mapa.	Menor complejidad, ausencia de una etapa extensa de recolección y entrenamiento, y mejor estabilidad en las pruebas realizadas (Bahl & Padmanabhan, 2000; Yu & Yu, 2025).
Cálculo de rutas	Dijkstra y A*	A*	Menor tiempo de ejecución y cantidad de nodos explorados gracias al uso de una heurística orientada al destino (Hart et al., 1968).

Tipo de base de datos	Bases relacionales y alternativas NoSQL	Base de datos relacional	La información del sistema presenta relaciones estructuradas entre recintos, destinos, personas, materias y horarios (Elmasri & Navathe, 2007).
Motor de base de datos	Oracle, MySQL, Microsoft SQL Server y PostgreSQL	PostgreSQL con PostGIS	Permite combinar el modelo relacional con el almacenamiento y las consultas de geometrías espaciales (PostGIS Development Group, s.f.).

Estas decisiones permitieron definir una solución acorde con los requerimientos funcionales y con el carácter experimental del proyecto, priorizando la simplicidad de implementación, la disponibilidad de herramientas y la posibilidad de representar y procesar información espacial.

2.2. Arquitectura general elegida

El Sistema de Localización en Interiores está compuesto por diez balizas BLE implementadas con ESP32, una aplicación móvil Android, una aplicación web de administración, una API RESTful y una base de datos relacional con soporte espacial. Las balizas emiten la información utilizada por la aplicación móvil para estimar la posición del usuario. A partir de esta ubicación, la aplicación representa al usuario sobre el mapa digital, permite seleccionar un destino y calcula la ruta correspondiente. La información del espacio y de los dispositivos se almacena de manera centralizada y puede ser consultada o actualizada por las aplicaciones mediante la API.

La Tabla 2.2 resume la forma de implementación, las tecnologías y las dependencias operativas de cada componente, mientras que la Figura 2.1 presenta su arquitectura general y el flujo de información entre ellos.

Tabla 2.2. Características técnicas y operativas de los componentes del sistema.

Componente	Implementación o despliegue	Tecnología principal	Funcionamiento autónomo	Conexión a internet
Balizas BLE	Dispositivos instalados en posiciones fijas; alimentación principal desde la red y batería de respaldo	ESP32, BLE/iBeacon y Wi-Fi	Emiten señales BLE sin depender del servidor. La batería permite mantener temporalmente su funcionamiento ante un corte eléctrico	No para emitir BLE. Sí para enviar reportes de estado a la API
Aplicación móvil InMap	Aplicación instalada en un dispositivo Android	Flutter/Dart/Unity	Puede recibir señales BLE y realizar localmente la estimación de posición y el cálculo de rutas	Sí para consultar el mapa y la información almacenada en el servidor. En caso de que no haya conexión, se utilizan los datos

				almacenados en caché.
Aplicación web de administración	Aplicación web accesible mediante navegador y desplegada en Render.	React y JavaScript (JSX)	No; depende de la API y de la base de datos	Sí
API RESTful	Servicio web desplegado en Render.	Java 17 y Spring Boot	No; requiere acceso a la base de datos para la mayoría de sus operaciones	Sí
Base de datos relacional y espacial	Servicio de base de datos desplegado en Render.	PostgreSQL y PostGIS	Conserva los datos de manera independiente, pero su utilización depende de las aplicaciones que los consultan	Sí para el acceso remoto desde la API

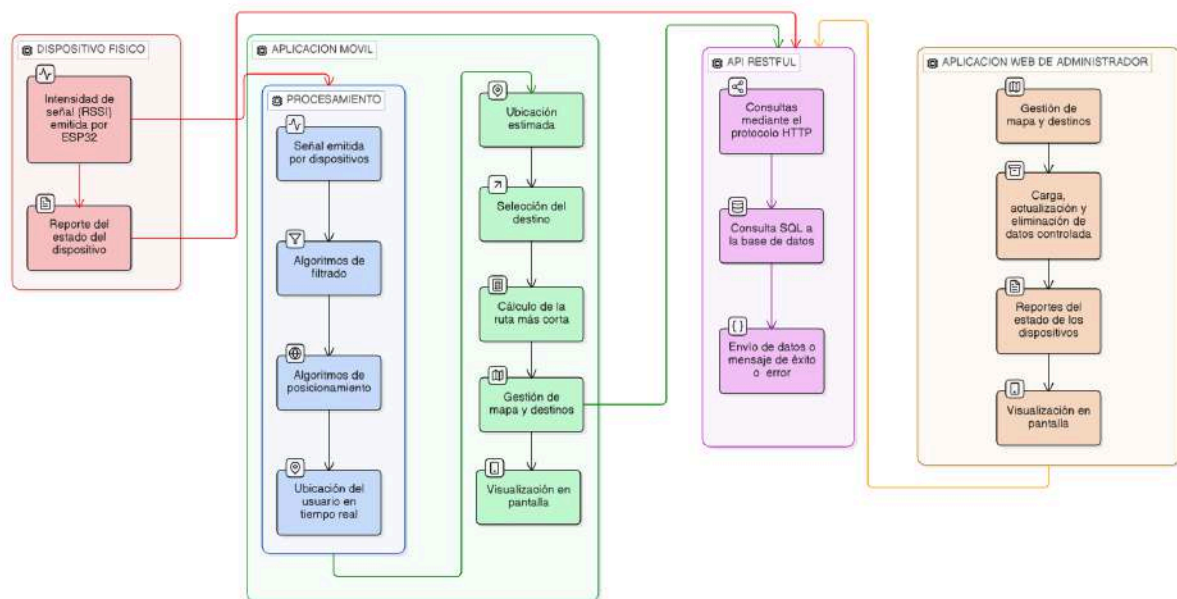


Figura 2.1. Arquitectura general del sistema y flujo de datos inter-componentes.

2.3. Requerimientos

En esta sección se definen y describen los requerimientos funcionales y no funcionales del proyecto “Sistema de Localización en Interiores”. Todos los requerimientos, excepto aquellos señalados expresamente, fueron implementados y verificados mediante las pruebas descritas en el Capítulo 4.

2.3.1. Requerimientos funcionales

RF1: Estimación de la posición del usuario

El sistema debe estimar la posición en tiempo real del usuario con un error promedio ≤ 2.5 metros.

RF2: Visualización en pantalla

La aplicación debe permitir ver en pantalla la ubicación del usuario (RF1) en formato de mapa digital.

RF3: Implementación de la persistencia

Se debe implementar una base de datos que sea consistente con los destinos proporcionados en el mapa. Debe almacenar nombres y personal asociado al destino principalmente.

RF4: Selección del destino por nombre

La aplicación debe permitir seleccionar un destino usando su nombre de la base de datos.

RF5: Selección del destino por criterio asociado

La aplicación debe permitir seleccionar un destino basado en el personal o recurso asociado a ese lugar.

RF6: Cálculo de la ruta más corta

El sistema debe calcular la ruta más corta desde la posición actual (RF1) al destino (RF4, RF5).

RF7: Actualización de la posición del usuario

El sistema debe actualizar la ubicación del usuario en tiempo real durante el recorrido.

RF8: Gestión del mapa del espacio

El sistema debe contar con un servidor central que permita gestionar el mapa del espacio, incluyendo la actualización de cambios en la disposición y la definición de zonas temporalmente restringidas al acceso de los usuarios.

RF9: Recepción de señales RSSI

El sistema debe recibir señales RSSI de al menos dos dispositivos beacons simultáneamente.

RF10: Prototipo

El prototipo debe incorporar un ESP32 con un sistema SAI (Sistema de Alimentación Ininterrumpida) que tenga la capacidad de cambiar automáticamente la fuente de alimentación de la tensión de línea (220 V AC) a una batería de respaldo. Se debe

garantizar un reducido tiempo de inhabilitación, no más de 30 seg.

RF11: Guía visual en tiempo real

El sistema debe proporcionar instrucciones visuales claras para la navegación (flechas, puntos de interés) y retroalimentación cuando el usuario se desvía de la ruta calculada (RF6).

RF12: Detección de interferencia de señal

El ESP32 debe implementar un mecanismo para detectar y reportar niveles de RSSI inusualmente bajos o variables, que puedan indicar interferencia o fallos en los beacons, afectando la precisión (RF1).

El RF12 **no fue implementado**. Aunque inicialmente se asignó esta responsabilidad al ESP32, durante el desarrollo se determinó que la detección debía realizarse principalmente sobre las mediciones obtenidas por la aplicación móvil. La funcionalidad requería definir y validar criterios adicionales para diferenciar interferencias, variaciones normales del entorno y fallos físicos. Debido a su baja prioridad, quedó fuera de la versión final del prototipo.

Reformulación del requerimiento:

RF12: Detección de anomalías en las señales recibidas. El sistema deberá analizar las mediciones de RSSI obtenidas por la aplicación móvil para detectar variaciones anómalas que puedan indicar interferencias, pérdida de cobertura o fallos en las balizas.

RF13: Gestión de beacons

El servidor debe registrar y monitorear el estado de cada beacon (identificador, ubicación configurada, última comunicación) para el mantenimiento.

2.3.2. Requerimientos no funcionales

RNF1: Claridad de la interfaz

La interfaz de usuario debe ser intuitiva y de fácil uso para personas con conocimiento no técnico.

RNF2: Compatibilidad

La app debe funcionar en dispositivos Android con versión ≥ 9.0 .

RNF3: Latencia

La latencia entre la recepción de la señal y la actualización en la pantalla no debe superar los 3 segundos.

RNF4: Autonomía

Los dispositivos deben tener una autonomía mínima de 24 horas ante cualquier interrupción de la línea de 220V, utilizando para ello una batería de litio 18650 recargable de 6800 mAh, 3.7 V.

RNF5: Escalabilidad

El sistema debe permitir la incorporación de nuevos dispositivos beacons o la ampliación del espacio sin necesidad de rediseñar completamente la arquitectura.

El RNF5 se considera **parcialmente cumplido**. La arquitectura permite incorporar nuevos dispositivos y adaptar el sistema a otros espacios sin rediseñar completamente sus componentes centrales. Sin embargo, cada entorno requiere la construcción de un nuevo modelo espacial, la configuración de la cuadrícula de navegación y una nueva distribución de las balizas. Asimismo, no se realizaron pruebas a una escala superior a la del escenario definido.

RNF6: Seguridad

El acceso al servidor de gestión del mapa (RF8) debe requerir autenticación de usuario y cifrado para prevenir modificaciones no autorizadas.

2.4. Plan de trabajo

El proyecto se estructuró originalmente en seis bloques de trabajo. Durante su desarrollo se incorporó un séptimo bloque, destinado a la aplicación web de administración:

- **Preliminar:** investigación, capacitación y vinculación inicial.
- **Hardware:** adquisición, configuración y construcción del prototipo.
- **Software I:** estimación de posición y cálculo de rutas.
- **Software II:** diseño e implementación de la base de datos.
- **Software III:** desarrollo del servidor/API.
- **Software IV:** desarrollo de la aplicación móvil InMap.
- **Software V:** desarrollo de la aplicación web de administración.

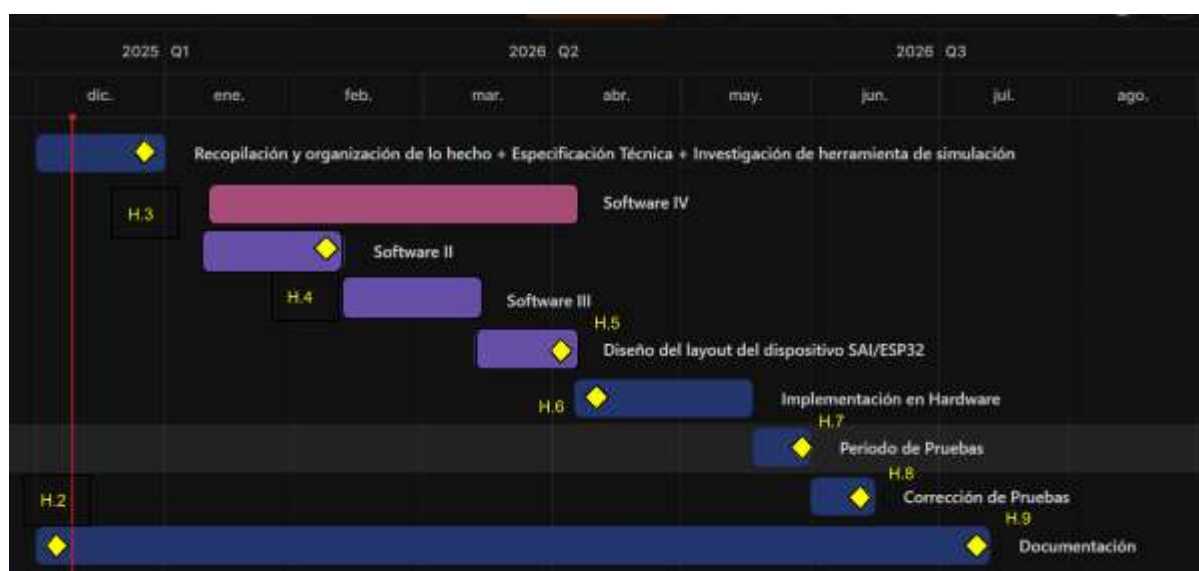


Figura 2.2. Diagrama de Gantt correspondiente a la planificación inicial del proyecto.

La Figura 2.2 presenta el diagrama de Gantt correspondiente a la planificación inicial, anterior a la incorporación del último bloque. Las tareas se distribuyeron según la experiencia y las responsabilidades principales de cada integrante. Constanza Maña se encargó principalmente de las aplicaciones finales y de la integración de los algoritmos en la aplicación móvil, mientras que Camila Baldomá tuvo a su cargo la base de datos y el desarrollo de la API. La investigación, los algoritmos de posicionamiento y rutas, la construcción del hardware, las pruebas y la documentación se realizaron en conjunto.

Capítulo 3

Diseño e implementación del sistema

3.1. Subsistema físico

El subsistema físico está formado por el conjunto de balizas BLE distribuidas en posiciones conocidas del espacio. Cada baliza constituye un nodo de operación autónoma basado en un microcontrolador ESP32, encargado de emitir periódicamente paquetes de publicidad que son recibidos por la aplicación móvil InMap para estimar la posición del usuario.

Los dispositivos se alimentan normalmente desde la red eléctrica del establecimiento y disponen de una batería de respaldo para mantener su funcionamiento ante interrupciones temporales del suministro. Cada unidad está compuesta por:

- una placa principal con el ESP32, dos indicadores LED y un divisor resistivo para medir la tensión de la batería;
- un módulo 18650 Battery Shield con una batería de litio de 6800 mAh y 3.7 V, encargado de la carga, protección y conmutación de la alimentación;
- una fuente que convierte los 220 V de alterna en 5 V de continua para alimentar el conjunto.

Durante su funcionamiento, las balizas emiten paquetes BLE con la información que permite identificarlas. La aplicación móvil mide la intensidad recibida de estas señales mediante el indicador RSSI y aplica los algoritmos de filtrado y estimación necesarios para calcular la ubicación del usuario.

Además, cada dispositivo utiliza conectividad Wi-Fi para enviar a la API un reporte periódico de su estado operativo. Si, al iniciar, detecta que el reinicio fue causado por una condición anómala, envía un reporte extraordinario sin esperar al período regular de 24 horas. Esta información puede consultarse desde la aplicación de administración.

3.1.1. Implementación de una baliza BLE

Para la emisión de los paquetes se utiliza el **modo de publicidad**. Durante este modo, el dispositivo transmite periódicamente paquetes que contienen información sobre su identidad, servicios y características [17].

El formato utilizado para estandarizar la estructura y el contenido de los paquetes de publicidad emitidos por la baliza es **iBeacon**. Fue desarrollado por Apple en 2013 y, al ser uno de los primeros estándares de balizas, cuenta con amplia aceptación entre los desarrolladores y soporte de hardware [16].

Para implementar una baliza iBeacon se utiliza una **placa ESP32**, en este caso una ESP32-WROOM-32 o ESP32-32X de Espressif, cuyo firmware fue desarrollado mediante Arduino IDE [18].

Cada paquete de publicidad, emitido a intervalos de tiempo fijos (*Advertising Interval*), contiene una carga útil (*payload*) estructurada con cuatro elementos principales:

1. **UUID (*Universally Unique Identifier*)**: identificador global de 128 bits que vincula la baliza con el sistema InMap.
2. **Major**: valor numérico de 16 bits utilizado para identificar una agrupación geográfica o estructural de balizas.
3. **Minor**: valor numérico de 16 bits que identifica de manera unívoca cada baliza dentro del grupo definido por el Major.
4. **Measured Power o Tx Power**: valor de referencia que representa la intensidad esperada de la señal RSSI a un metro del emisor y permite estimar su atenuación.

El firmware desarrollado para el ESP32 tiene como propósito implementar una baliza BLE de tipo iBeacon con bajo consumo energético y capacidad de reporte periódico mediante Wi-Fi. El dispositivo se identifica por medio de un UUID fijo y de los valores Major y Minor, que permiten distinguir cada nodo dentro del sistema.

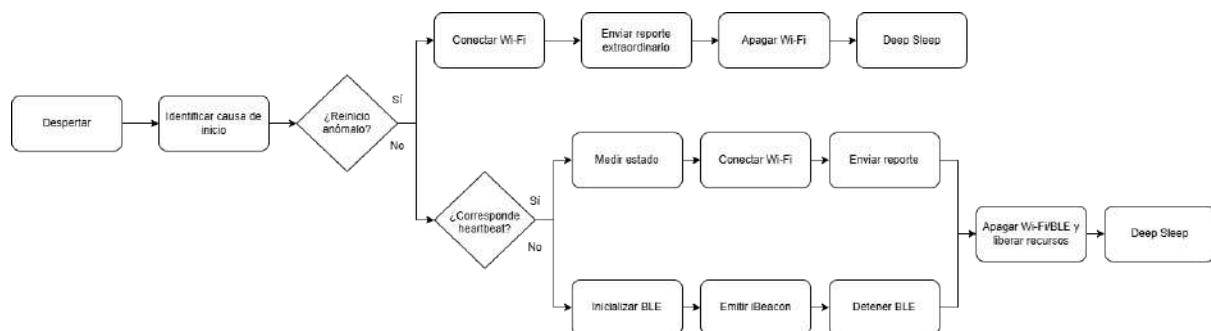


Figura 3.1. Diagrama de funcionamiento del dispositivo.

Para reducir el consumo energético, el ESP32 utiliza el modo **Deep Sleep**. En cada ciclo, el dispositivo despierta y evalúa si corresponde realizar un reporte por Wi-Fi. Si no corresponde, inicializa BLE y mantiene la publicidad activa durante aproximadamente 0,5 segundos. Luego detiene BLE, libera sus recursos y vuelve a entrar en Deep Sleep. El tiempo de sueño se encuentra configurado y es de 2 segundos.

El firmware identifica arranques que no corresponden a un despertar normal por temporizador. Entre ellos se consideran el encendido, *brown-out*, *panic* y distintos reinicios provocados por el *watchdog*. Ante estas situaciones, el dispositivo intenta enviar inmediatamente un **reporte extraordinario** al servidor, sin esperar el intervalo del *heartbeat* periódico.

El envío de datos a la API se realiza mediante una rutina de *heartbeat*. Su objetivo es que el dispositivo informe periódicamente su estado, incluyendo el identificador de la baliza, el nivel de batería, la memoria libre, el tiempo activo, la causa de despertar y la razón de reinicio, entre otros datos. El intervalo del *heartbeat* se configuró en 24 horas, equivalentes a 86.400 segundos.

Cuando corresponde enviar el *heartbeat*, el firmware omite la emisión BLE durante ese ciclo para reservar memoria y recursos para la conexión Wi-Fi.

La medición de la batería se realiza mediante el pin analógico GPIO34. Como la tensión de una celda 18650 puede alcanzar aproximadamente 4,2 V, se utiliza un divisor resistivo para reducirla antes de que ingrese al ESP32.

El sistema utiliza dos LED de estado. El LED verde indica una emisión BLE correcta, mientras que el LED rojo señala errores críticos.

3.1.2. Diseño del prototipo

El prototipo físico fue diseñado para integrar en una única unidad la emisión BLE, el reporte de estado mediante Wi-Fi y un sistema de alimentación con respaldo de batería. Su estructura está formada por una placa principal que contiene el ESP32 y los elementos de medición y señalización, un módulo 18650 Battery Shield con una batería de litio y una fuente encargada de convertir la tensión de red de 220 VCA a 5 VCC.

La Figura 3.2 y la Figura 3.3 presentan el esquema eléctrico del prototipo final y las conexiones entre sus componentes.

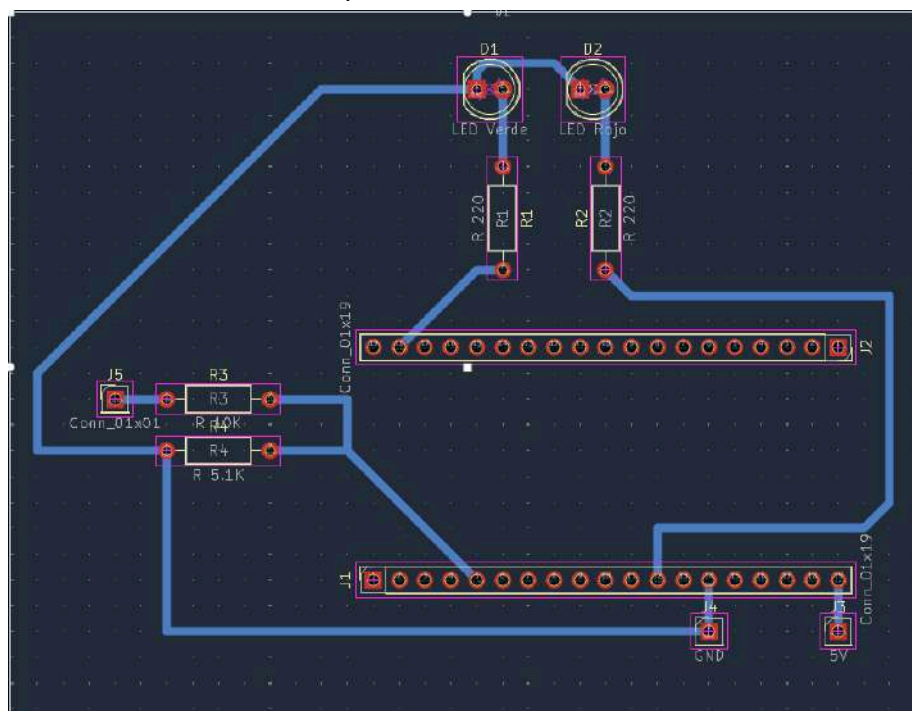


Figura 3.2. Esquemático de la placa principal. (Nota: Los conectores J1 y J2 corresponden a los zócalos donde se aloja el ESP32).

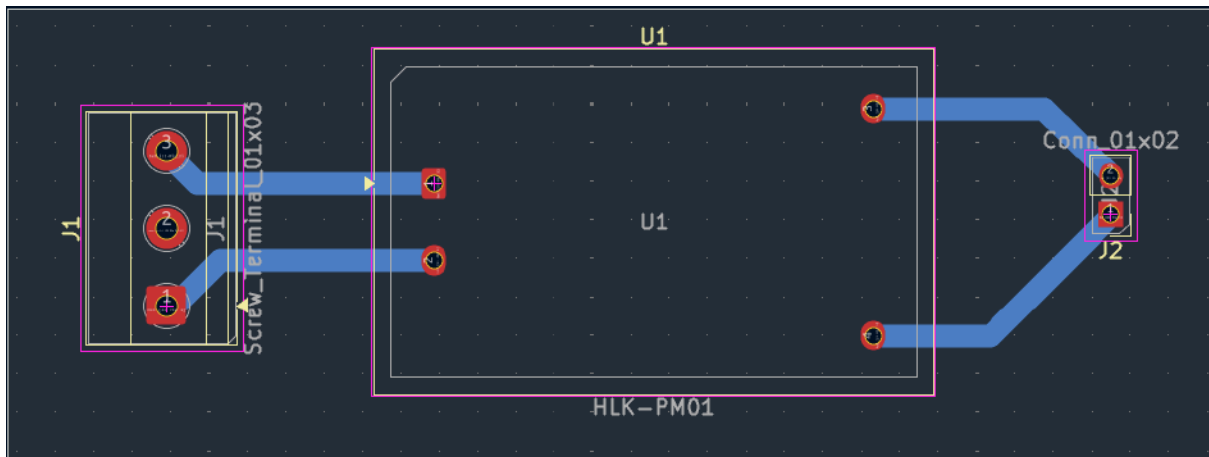


Figura 3.3. Esquemático del módulo de alimentación (Fuente regulada Hi-Link)

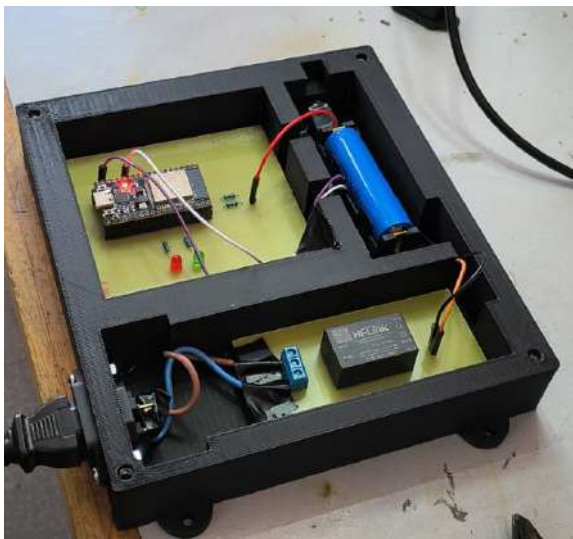


Figura 3.4. Interior del prototipo final.

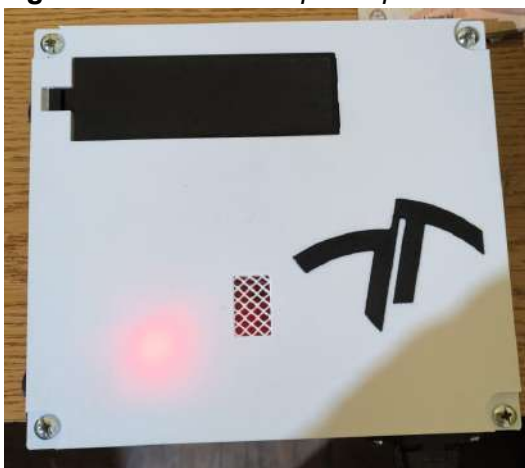


Figura 3.5. Exterior del prototipo final.

En la Figura 3.4 y en la Figura 3.5 se muestra la disposición física de las placas y el montaje final de la unidad. En ella se pueden observar los siguientes componentes:

- **ESP32:** ejecuta el firmware, emite los paquetes BLE, realiza las mediciones y envía los reportes por Wi-Fi.

- **Battery Shield y batería 18650:** administran la carga y permiten mantener la alimentación ante una interrupción de la red eléctrica.
- **Fuente de alimentación:** convierte los 220 VCA de la red en 5 VCC para alimentar el sistema y cargar la batería.
- **Divisor resistivo:** adapta la tensión de la batería al rango admitido por la entrada analógica del ESP32.
- **LED verde y LED rojo:** indican respectivamente el funcionamiento normal y la aparición de errores.

En condiciones normales, la fuente toma energía de la red eléctrica y entrega 5 V al Battery Shield, que alimenta el ESP32 y mantiene cargada la batería. Ante una interrupción del suministro, el módulo conmuta la alimentación a la batería sin modificar el funcionamiento lógico de la baliza.

3.1.3. Limitaciones y problemas detectados

Durante la implementación y las pruebas del subsistema físico se detectaron diferentes problemas relacionados con la estabilidad del firmware, la alimentación, la medición de la batería y la comunicación con la API. La Tabla 3.1 resume los principales inconvenientes, las soluciones aplicadas y su estado al finalizar el desarrollo.

Tabla 3.1. *Problemas detectados y soluciones implementadas.*

Problema detectado	Causa principal	Solución aplicada	Estado
Reinicios provocados por el <i>watchdog</i>	Cierre asíncrono de BLE antes de ingresar en Deep Sleep	Se incorporó la desinicialización completa de BLE y un tiempo de espera antes del Deep Sleep	Resuelto
Uso simultáneo de BLE y Wi-Fi	Memoria y recursos limitados del ESP32	En los ciclos de reporte se omite temporalmente la emisión BLE	Resuelto
Fallas en el envío de reportes durante las pruebas locales	El servidor local no era accesible de forma estable desde el ESP32	Se desplegó la API en Render mediante una URL pública	Resuelto
Lecturas inestables de la tensión de batería	Ruido del conversor analógico y variaciones posteriores al despertar	Se promedian múltiples mediciones y se descartan las primeras lecturas	Resuelto
Estimación lineal poco representativa de la batería	La descarga de una celda 18650 no presenta un comportamiento lineal	Se reemplazó el porcentaje por los estados de batería alta, media y baja	Resuelto

Cambios repetidos entre estados de batería	Fluctuaciones de tensión cercanas a los umbrales	Se incorporó histéresis entre los cambios de estado	Resuelto
Tensión de batería superior al rango analógico del ESP32	La celda puede alcanzar aproximadamente 4,2 V	Se incorporó un divisor resistivo conectado al GPIO34	Resuelto
Aumento del consumo ante ausencia de Wi-Fi	Intentos prolongados de conexión	Se estableció un tiempo máximo de conexión antes de regresar a Deep Sleep	Resuelto
Capacidad y autonomía reales inciertas	La capacidad nominal de la batería y la eficiencia del Battery Shield pueden diferir de los valores efectivos	Se realizó una caracterización experimental de descarga	Verificado experimentalmente.

3.1.4. Costos del prototipo

Con el propósito de obtener una primera aproximación al costo de fabricación de las balizas desarrolladas, se relevaron los precios de los componentes utilizados en la construcción de una unidad. La Tabla 3.2 presenta los costos unitarios considerados y el costo total estimado del prototipo.

Los valores corresponden a precios minoristas consultados en agosto de 2026. Por lo tanto, representan el costo de construcción de una unidad experimental y no contemplan posibles reducciones derivadas de compras mayoristas, fabricación seriada o ensamblado de placas en volumen. Tampoco se incluyen las horas destinadas al diseño, la programación, el montaje y las pruebas.

Tabla 3.2. *Costos unitarios del prototipo.*

Componente	Costo unitario (ARS)
ESP32	\$8.000
Battery Shield	\$8.500
Batería	\$3.000
4 Resistencias	\$1.320

2 LEDs	\$600
Fuente Hi-Link	\$8.500
Placa de Cobre (100x100mm)	\$4.500
Placa de Cobre (50x100mm)	\$2.250
Cable Interlock	\$4.700
Caja	\$15.000
Total Estimado	\$56.370

De acuerdo con los componentes relevados, el costo estimado de fabricación de una baliza fue de \$56.370 ARS. Este valor se comparó de manera preliminar con los precios minoristas de algunas balizas BLE comerciales disponibles en el mercado. Las alternativas encontradas presentaron valores aproximados de entre \$18.167 y \$82.000 ARS por unidad.

Como referencia de mercado, se relevaron los precios minoristas de tres balizas BLE comercializadas a través de Mercado Libre: SenseCAP BC02, Blue Charm BC021 y un dispositivo genérico basado en el nRF52810 [24], [25], [26].

Sin embargo, no se encontró un producto directamente equivalente al prototipo desarrollado. Por esta razón, los precios de las balizas comerciales se utilizaron únicamente como referencias minoristas de mercado. La alternativa de menor precio correspondía a una baliza BLE básica importada, alimentada únicamente mediante batería y sin conectividad Wi-Fi propia para el envío de reportes. En cambio, la solución desarrollada incorpora alimentación desde la red eléctrica, batería de respaldo y monitoreo remoto del estado operativo.

Por lo tanto, no puede afirmarse que el prototipo sea la alternativa BLE de menor precio disponible. No obstante, su costo resulta competitivo frente a algunos de los productos relevados si se consideran las prestaciones adicionales que integra. Asimismo, el costo unitario podría reducirse mediante la compra mayorista de componentes y la fabricación de placas en serie, aunque esta reducción deberá comprobarse mediante cotizaciones para diferentes volúmenes de producción.

3.2. Estimación de la posición

El bloque de procesamiento o estimación de la posición se encarga de determinar en tiempo real la posición bidimensional del usuario dentro del edificio. Para ello, se utiliza una técnica denominada **posicionamiento basado en RSSI**, o posicionamiento basado en la intensidad de la señal recibida, que consiste en medir la potencia con la que se reciben señales de radio emitidas desde ubicaciones conocidas. El principio general es que la intensidad recibida tiende a disminuir a medida que aumenta la distancia respecto de la fuente, aunque puede verse afectada por las características del entorno.

A partir de este principio, el bloque de procesamiento recibe las mediciones de RSSI captadas por el teléfono móvil desde balizas BLE de posición conocida y devuelve una coordenada válida sobre el mapa de navegación. La solución satisface el requerimiento RF1, que establece un error de localización inferior a 2,5 m en el escenario de evaluación definido.

Para implementar este bloque de posicionamiento se evaluaron dos enfoques principales: modelos de propagación de señal y el mapeo de radio (fingerprinting). Si bien el fingerprinting procesado mediante una red neuronal densa (MLP) obtuvo métricas superiores durante la fase de datos simulados, el **algoritmo del Centroides Ponderado (WCL)** demostró un desempeño más robusto frente a las fluctuaciones de señal en el entorno real de prueba. Por este motivo, la arquitectura de “InMap” utiliza actualmente la estimación por WCL con restricciones topológicas del mapa para resolver la ubicación del usuario en tiempo real.

En las siguientes subsecciones se describen las técnicas desarrolladas y los criterios utilizados para seleccionar la solución final.

3.2.1. Métodos basados en propagación

Como se mencionó previamente, el posicionamiento basado en RSSI parte de que, en términos promedio, la potencia recibida tiende a disminuir cuando aumenta la distancia entre el transmisor y el receptor. En interiores, esta relación es sólo aproximada: las paredes, las reflexiones, la presencia de personas, la orientación del teléfono y las interferencias provocan variaciones que impiden interpretar una lectura aislada como una distancia exacta [10].

Para convertir el RSSI de una baliza en una distancia estimada se utiliza la siguiente ecuación:

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

donde n es una constante que depende de las características del entorno, $TxPower$ es un valor preestablecido para el dispositivo emisor que representa el RSSI esperado a un metro de distancia, y $RSSI$ es la intensidad medida por el teléfono móvil.

En el entorno de prueba se definieron diez balizas BLE con posiciones conocidas sobre el plano, como se puede observar en la Figura 3.6. Es importante aclarar que los círculos son alcances operativos teóricos (5m de radio). En la práctica se confirmó que efectivamente cada punto fuera alcanzado por la señal de al menos dos balizas.

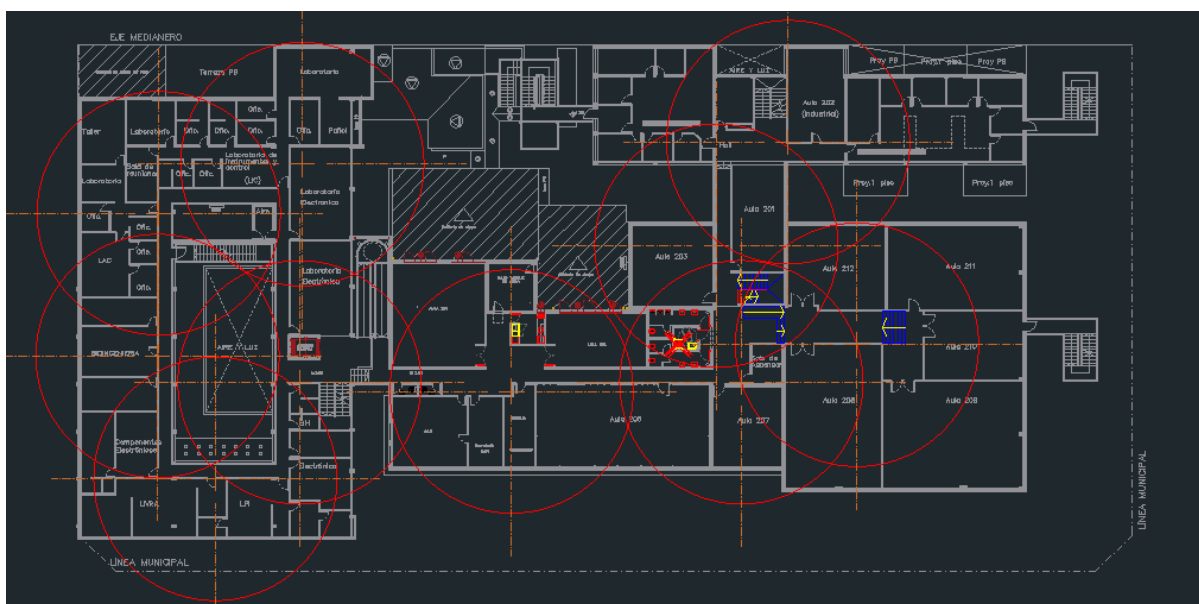


Figura 3.6. Distribución de las balizas en el espacio prevista inicialmente.

La **trilateración** es un método geométrico que permite estimar la posición de un punto desconocido a partir de sus distancias respecto de tres o más puntos de referencia conocidos. En el caso bidimensional, cada distancia define una circunferencia centrada en la baliza correspondiente y, bajo condiciones ideales, su intersección determina la posición del receptor, como se observa en la Figura 3.7.

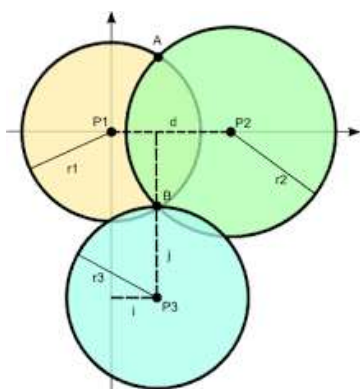


Figura 3.7. Círculos típicos trazados en la técnica de Trilateración.

Sin embargo, debido a la variabilidad del RSSI, las circunferencias asociadas con las distancias estimadas rara vez se intersectan en un único punto. Tradicionalmente, este problema se aborda mediante un procedimiento analítico de mínimos cuadrados, el cual busca calcular la coordenada que minimice el error respecto de todas las distancias obtenidas en lugar de buscar una intersección exacta.

Si bien este enfoque fue evaluado inicialmente, su dependencia de un modelo geométrico rígido lo vuelve susceptible a fluctuaciones en entornos con alto nivel de ruido. Para superar esta limitación, en este trabajo se descartó la trilateración geométrica en favor del algoritmo de **Centroide Ponderado (WCL)**. Este es un método utilizado principalmente en redes de sensores inalámbricos (WSN) para localizar un dispositivo desconocido estimando su

posición a través del promedio de las coordenadas espaciales de las balizas, otorgándole mayor "peso" gravitacional a aquellas que presentan una señal más fuerte [14].

Los algoritmos tradicionales de centro de masa puro son susceptibles a errores de medición de RSSI y a problemas asociados con la distribución física de las balizas, lo que puede resultar en grandes errores de posicionamiento debido a su dependencia de cálculos geométricos simples. Además, los algoritmos que buscan una precisión extrema suelen ir acompañados de un alto consumo de recursos computacionales, dificultando el equilibrio entre precisión y eficiencia energética. Para superar este desafío en entornos complejos, se implementó un algoritmo de centro de masa ponderado basado en la corrección empírica de pesos.

Para adaptar el WCL a las condiciones del edificio, se diseñaron e implementaron optimizaciones heurísticas:

- **Selección de nodos críticos (Top-K):** El algoritmo restringe el cálculo a las tres balizas con mayor intensidad, descartando señales débiles propensas al multitrayecto.
- **Transformación exponencial:** Convierte la escala logarítmica de pérdida de señal en un peso estadístico positivo, acentuando la influencia de la baliza más cercana.
- **Validación de umbral:** Exige un mínimo de señales concurrentes válidas para habilitar el cálculo, evitando estimaciones erráticas en puntos ciegos.

El método se evaluó utilizando un conjunto de registros con coordenadas reales conocidas. En la primera evaluación matemática, el WCL presentó un **error medio de 2,8 m**. A partir de este resultado, se acopló una restricción topológica del mapa, forzando mediante algoritmos de ajuste geométrico (snap) que las posiciones estimadas se limitaran estrictamente a los ejes centrales de los pasillos transitables. Esta integración redujo la dispersión de las estimaciones y permitió obtener un **error medio final de 2,2 m**, cumpliendo el requerimiento máximo de 2,5 m establecido por RF1.

En consecuencia, el algoritmo WCL con información espacial del entorno resulta adecuado para estimar la posición del usuario dentro del escenario de evaluación.

3.2.2. Otras técnicas analizadas

Con el propósito de obtener mejores resultados que con el método de trilateración, analizado en la sección anterior, se consideró la técnica de **Fingerprinting** y se evaluaron distintos algoritmos.

En la técnica de *Fingerprinting*, también conocido como método de huella digital o huella de radio, se registran, durante un estudio preliminar, los valores RSSI en diversas ubicaciones conocidas para crear un "mapa de radio". Posteriormente, las lecturas RSSI en tiempo real se comparan con este mapa mediante algoritmos como el vecino más cercano o métodos de aprendizaje automático para estimar la ubicación del usuario basándose en la mejor coincidencia entre los datos observados y registrados.

Su principal ventaja es que puede aprender características particulares del entorno, como atenuaciones, reflexiones y obstáculos, sin convertir explícitamente cada RSSI en una distancia. Por el contrario, su mayor desventaja radica en el tiempo y los recursos

requeridos para la etapa inicial de recolección de datos, lo cual dificulta la escalabilidad y el mantenimiento del sistema ante cambios físicos en el edificio.

Este método se divide típicamente en dos fases principales: fuera de línea y en línea [8], [13].

Fase fuera de línea: creación de la Base de Datos de Huellas.

Durante esta etapa inicial, el objetivo es construir un mapa (o grilla) detallado del entorno, conocido como base de datos de huellas dactilares. Para ello:

1. Se definen Puntos de Referencia (RP) estratégicos en el área a mapear.
2. En cada RP, se miden y registran las distribuciones de la Intensidad de la Señal Recibida (RSSI) de diferentes dispositivos Bluetooth Low Energy (BLE).
3. Con estas mediciones, se genera una grilla o mapa de huellas dactilares que correlaciona cada RP con sus características de señal únicas.

Para la implementación de esta etapa, se creó una aplicación móvil sencilla, con el objetivo de facilitar la toma de datos. El objetivo de ésta fue construir la Base de Datos de Huellas, correlacionando coordenadas físicas conocidas con vectores de intensidad de señal (RSSI). La aplicación permite seleccionar la celda correspondiente a la ubicación real y ejecutar un escaneo BLE continuo. Su objetivo es funcional: asegurar que cada vector de señales quede asociado con una coordenada conocida.

Fase en línea: localización del punto desconocido.

Una vez establecida la base de datos, la fase en línea permite determinar la posición de un dispositivo en tiempo real:

1. El dispositivo en la ubicación desconocida adquiere sus propias mediciones de RSSI de los BLE cercanos.
2. Estas mediciones se comparan con la grilla preestablecida en la fase fuera de línea.
3. Utilizando algoritmos como K Vecinos Más Cercanos (KNN), algoritmos similares o redes neuronales más complejas, se identifican los RPs de la base de datos que mejor coinciden con los RSSI actuales, permitiendo así estimar la posición del dispositivo.

Durante esta etapa, se construyeron modelos y analizaron distintos algoritmos. Para ello, se combinaron dos conjuntos de datos, obtenidos a partir de dos dispositivos diferentes, en un único conjunto utilizado para el entrenamiento. De esta manera, el modelo resultante logra una mayor capacidad de generalización y una menor sensibilidad a las variaciones propias de un solo dispositivo.

Posteriormente, se analizaron diferentes algoritmos y modelos de redes neuronales, como se observa en la Tabla 3.3.

Tabla 3.3. *Métodos analizados para la estimación de la posición.*

Técnica	Algoritmo o modelo	Información utilizada	Configuración o característica principal	Error medio	Conclusión
Trilateración	Mínimos cuadrados	Distancias estimadas a partir del RSSI y posiciones conocidas de las balizas	Resolución de la posición mediante tres o más balizas	3,43 m	El ruido del RSSI produjo estimaciones fuera de las zonas transitables.
Trilateración restringida	Mínimos cuadrados con restricción espacial	RSSI, posiciones de las balizas y geometría del mapa	La posición estimada se limita a pasillos y zonas transitables	2,2 m	Redujo el error euclídeo, pero mantuvo inestabilidad (saltos bruscos) ante fluctuaciones de la señal.
Centroide Ponderado (WCL)	Promedio ponderado por RSSI (Heurística Top-3)	Vector RSSI actual y posiciones conocidas de las balizas	Transformación exponencial del RSSI de las 3 señales más fuertes	2,8 m	Mitigó los saltos bruscos del modelo geométrico, pero conservó estimaciones fuera del área transitable.
WCL con restricción topológica	WCL + filtros espaciales del mapa	RSSI, posiciones de las balizas y geometría del mapa	La posición estimada se limita a pasillos y zonas transitables	2,2 m	Presentó la mayor estabilidad y coherencia espacial en pruebas físicas; fue el método definitivo integrado en la aplicación final.
<i>Fingerprinting</i>	WKNN	Vector RSSI actual y mapa de huellas	Tres vecinos, ponderados según su distancia respecto de la medición actual	2,29 m	Mejóro levemente a KNN, aunque presentó estimaciones máximas alejadas.
<i>Fingerprinting</i>	SVR	Vector RSSI actual y mapa de huellas	Regresión mediante máquinas de vectores de soporte	2,23 m	Fue el mejor algoritmo clásico de <i>fingerprinting</i> , pero no superó a la trilateración restringida.
Red neuronal recurrente	RNN	Secuencia temporal de mediciones RSSI	Incorpora memoria de observaciones anteriores	—	Se analizó conceptualmente, pero no se implementó como modelo final debido a sus limitaciones para conservar dependencias temporales extensas.

Red neuronal recurrente	LSTM	Secuencias de 30 mediciones RSSI	Dos capas LSTM, seguidas por capas densas; optimizador AdamW [12]	1,25 m	Obtuvo buenos resultados sobre los datos de evaluación, pero presentó mayor costo computacional y riesgo de sobreajuste.
Red neuronal densa	MLP sin contexto temporal	Vector RSSI actual	Capas densas para estimar directamente las coordenadas (x,y)	2,53 m	El RSSI actual por sí solo no aportó suficiente información para obtener una estimación estable.
Red neuronal densa	MLP con posición anterior	Vector RSSI actual y posición anterior conocida	Capas de 64, 64 y 32 neuronas; salida lineal de dos coordenadas	1,05 m	Alcanzó el menor error sobre los datos evaluados, pero durante las pruebas en el entorno real produjo saltos e inestabilidad en la posición.

A partir del análisis realizado y los resultados obtenidos, la red neuronal densa, enriquecida con la información del estado anterior, se estableció inicialmente como la solución **más óptima teórica**, ofreciendo el mejor equilibrio entre rendimiento, simplicidad y eficiencia computacional para el problema de la localización con balizas BLE.

Sin embargo, este comportamiento no se reprodujo con la misma estabilidad durante las pruebas en tiempo real. Por este motivo, se seleccionó el algoritmo del **centroide ponderado WCL** restringido por la geometría del mapa, que presentó un error medio de 2,2 m, cumplió el requerimiento RF1 y proporcionó un comportamiento más estable en el entorno real de implementación.

3.2.3. Evaluación en el entorno real

Para evaluar el funcionamiento de la aplicación "InMap", se realizaron pruebas en el segundo piso de la Facultad de Ingeniería. La distribución de las diez balizas prevista inicialmente se presentó en la sección anterior. Sin embargo, para estas pruebas se utilizó únicamente la mitad derecha del plano, con cinco balizas: tres prototipos completos y dos dispositivos compuestos solamente por el ESP32 conectado a una fuente de alimentación. Esta diferencia no afectó la emisión de las señales BLE necesarias para estimar la posición.

De acuerdo con los resultados obtenidos mediante los datos utilizados en las evaluaciones anteriores, inicialmente se consideró que la alternativa más adecuada era el método de fingerprinting procesado mediante una red neuronal densa (MLP) con información de la posición anterior. Por este motivo, las primeras pruebas en el entorno real se realizaron con dicho modelo.

Durante estas pruebas, la posición estimada no se mantuvo estable. Si bien por momentos coincidía con la ubicación real del usuario, posteriormente presentaba saltos hacia posiciones alejadas dentro del mapa. Debido a la naturaleza de "caja negra" propia de las

redes neuronales, diagnosticar y ajustar el origen exacto de estos desvíos resultó complejo. Se incorporaron filtros y restricciones espaciales para reducir estas fluctuaciones, pero no fue posible eliminarlas satisfactoriamente.

La implementación utilizaba inicialmente el cálculo WCL para obtener la primera posición y, posteriormente, la red neuronal para realizar el seguimiento de la trayectoria. Debido a que la posición inicial obtenida mediante el algoritmo ponderado presentaba un comportamiento más estable, se decidió evaluar esta técnica durante todo el proceso de estimación.

Con este cambio desaparecieron los principales saltos observados durante las pruebas con la red neuronal. A diferencia del modelo neuronal, el WCL ofrece una estructura matemática transparente que permite un control directo sobre el cálculo. Además, mediante ajustes en las restricciones topológicas asociadas a paredes y pasillos transitables, se obtuvo una estimación coherente con el desplazamiento real del usuario.

Por lo tanto, la versión actual de “InMap” utiliza el algoritmo de **Centroide Ponderado (WCL)** restringido por la geometría del mapa para estimar la posición en tiempo real. Esta elección no implica que la red neuronal sea intrínsecamente inferior, sino que el WCL, al ser un modelo parametrizable y predecible, fue la alternativa que mejor satisfizo, bajo las condiciones de evaluación, los requisitos de estabilidad del sistema.

3.3. Persistencia y modelo espacial

Para el funcionamiento del sistema es necesario almacenar tanto la representación espacial del entorno como la información asociada a este, entre ella aulas, materias, horarios, personal, usuarios y balizas. La aplicación móvil recupera estos datos para mostrar la información solicitada y realizar los cálculos de navegación, mientras que la aplicación web permite su actualización por parte de los administradores.

Debido a que estas entidades presentan relaciones claramente definidas, se adoptó un modelo de datos relacional. Este modelo organiza la información en tablas compuestas por filas y columnas y permite vincular las distintas entidades mediante relaciones. De esta manera, resulta apropiado para representar de forma estructurada y consistente los datos gestionados por el sistema [15].

La Figura 3.8 presenta el Diagrama Entidad-Relación (DER) definido para el proyecto, donde se observan las principales entidades, sus atributos y las relaciones existentes entre ellas.

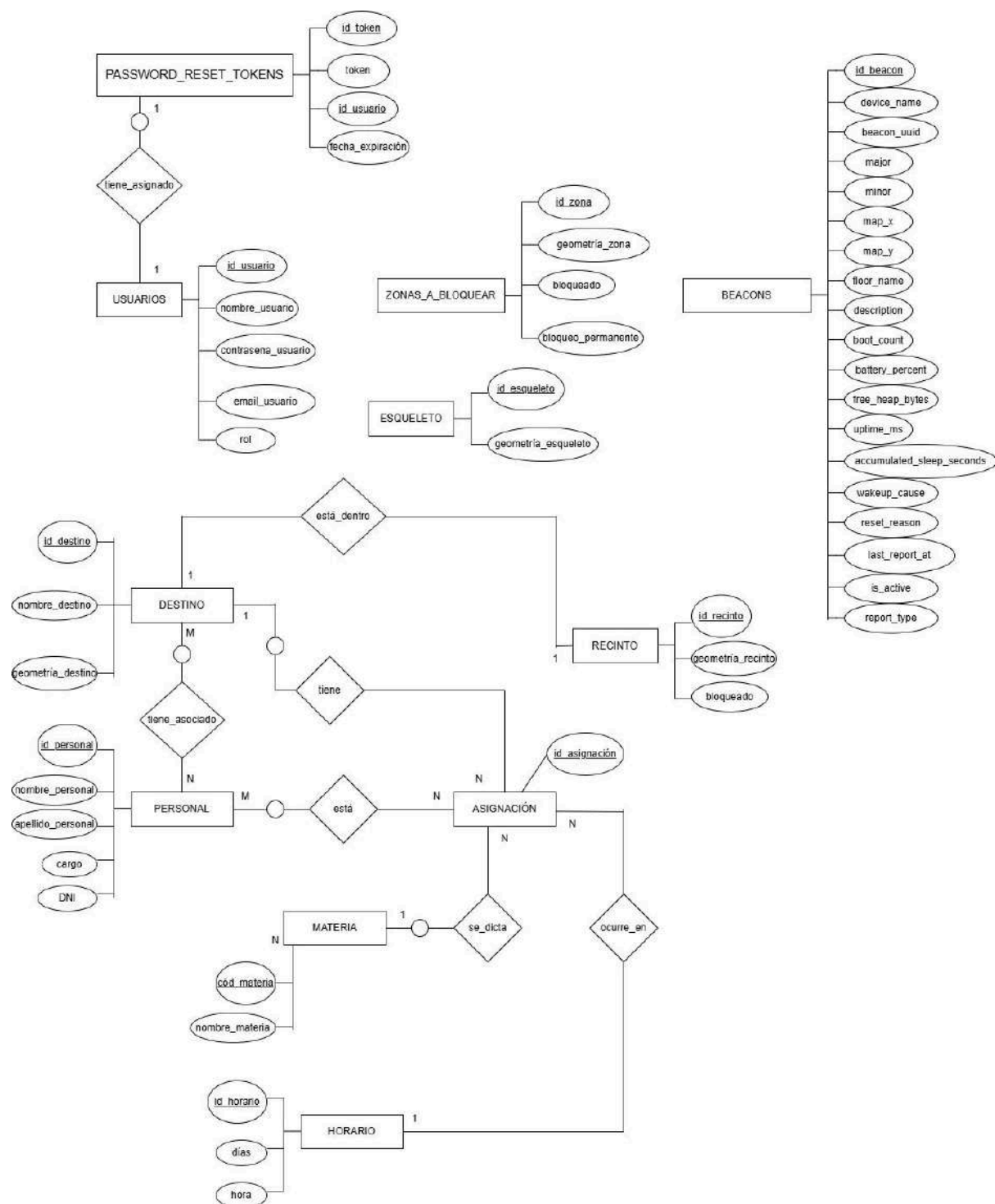


Figura 3.8. Diagrama Entidad-Relación (DER)

La base de datos implementada centraliza la información utilizada por las aplicaciones móvil y web. El acceso a los datos se realiza mediante la API del sistema, que aplica las validaciones y reglas de negocio correspondientes. De esta manera, las aplicaciones cliente no interactúan directamente con el motor de base de datos.

3.3.1. Elección del motor de Base de Datos

Entre los sistemas de gestión de bases de datos relacionales (SGBD) considerados se encuentran: Oracle, MySQL, Microsoft SQL y PostgreSQL. Si bien todos permiten almacenar la información relacional del sistema, se seleccionó PostgreSQL principalmente por su integración con PostGIS y por tratarse de una solución de código abierto.

PostGIS es una extensión que incorpora a PostgreSQL capacidades específicas para almacenar, consultar y procesar información espacial [19], [33]. Entre las funcionalidades relevantes para el proyecto se encuentran:

- Almacenamiento de geometrías, como puntos, líneas y polígonos.
- Representación del mapa, las zonas transitables y la ubicación de los distintos elementos del espacio.
- Realización de consultas y operaciones espaciales, como cálculos de distancia, intersección y contención.
- Integración de la información espacial con las restantes entidades del modelo relacional.

Estas capacidades permiten mantener en una misma base de datos tanto la información administrativa del establecimiento como su representación espacial. Por este motivo, el motor seleccionado para el sistema fue **PostgreSQL [33] con la extensión PostGIS [19]**.

3.4. API

El software del sistema adopta una arquitectura cliente-servidor de tres niveles, como se muestra en la Figura 3.9. La capa de presentación está compuesta por la aplicación móvil “InMap” y la aplicación web de administración, mientras que la capa de persistencia corresponde a la base de datos “InMap_DB”, implementada en PostgreSQL con la extensión PostGIS. Entre ambas se encuentra la capa de aplicación, cuya implementación se describe en esta sección.

Esta capa se implementó mediante una API RESTful con comunicación sobre HTTPS. Su incorporación permite evitar el acceso directo de las aplicaciones cliente a la base de datos y centralizar las reglas de negocio, las validaciones, el acceso a la información y los mecanismos de autenticación y autorización. Además, proporciona una interfaz común para que las aplicaciones móvil y web utilicen los mismos datos y operaciones.

Se eligió el estilo REST por su compatibilidad con aplicaciones web y móviles, su utilización de los métodos estándar del protocolo HTTP y su facilidad de integración y mantenimiento. De esta manera, cada nivel conserva una responsabilidad definida y puede modificarse con menor impacto sobre los restantes componentes.

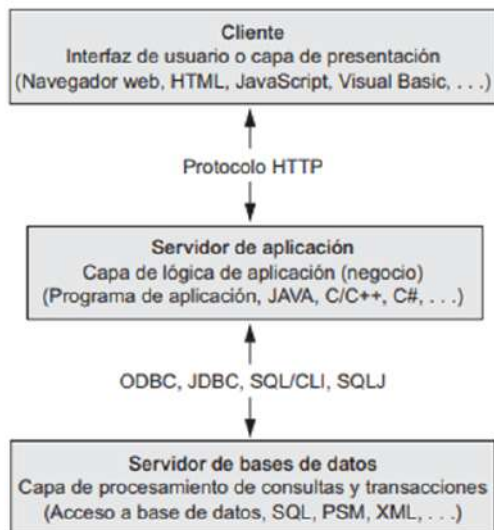


Figura 3.9. *Arquitectura cliente-servidor de tres niveles*

3.4.1. Arquitectura de software y estilo RESTful

La API fue desarrollada en Java mediante el framework Spring Boot. Este framework facilita la configuración del proyecto, la gestión de dependencias y la ejecución mediante un servidor web embebido. Asimismo, proporciona herramientas para implementar servicios RESTful, acceder a la base de datos y aplicar los mecanismos de autenticación y autorización requeridos por el módulo administrativo [21].

La API expone recursos mediante endpoints que utilizan los métodos estándar de HTTP. En el entorno desplegado, las comunicaciones se realizan de forma cifrada mediante HTTPS. Las operaciones principales son:

- **GET:** consulta de información desde las aplicaciones móvil y web.
- **POST:** creación de recursos desde la aplicación web y recepción de los reportes enviados por las balizas.
- **PUT:** actualización de información por parte de usuarios administradores autenticados.
- **DELETE:** eliminación de recursos autorizada desde la aplicación web.

La implementación del backend se organiza en tres capas, representadas en la Figura 3.10:

- **Capa de controladores:** recibe las solicitudes HTTP, valida su formato inicial, delega el procesamiento correspondiente y construye las respuestas enviadas a los clientes.
- **Capa de servicios:** contiene la lógica de negocio y coordina los casos de uso del sistema, incluidas las validaciones y operaciones sobre las distintas entidades.
- **Capa de acceso a datos:** contiene las entidades que representan la información persistida y los repositorios encargados de interactuar con PostgreSQL mediante Spring Data JPA [4], [22] .

Esta separación distribuye las responsabilidades del backend y facilita el mantenimiento y la modificación independiente de sus componentes [6], [3].

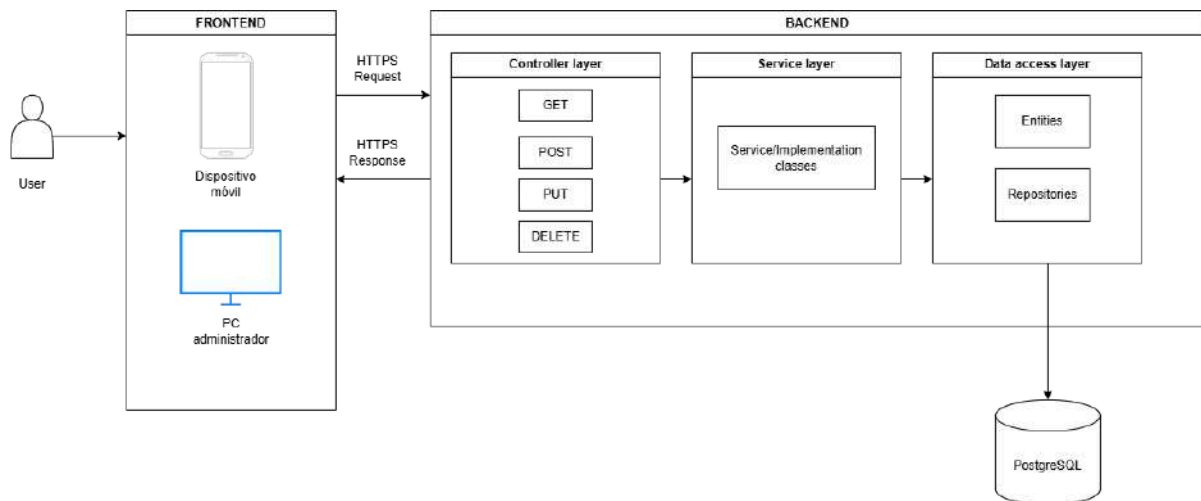


Figura 3.10. *Arquitectura RESTful*

Para el desarrollo se utilizó IntelliJ IDEA 2026.1.1 [27], mientras que Postman [28] se empleó para ejecutar solicitudes y comprobar las respuestas proporcionadas por los endpoints.

3.4.2. Despliegue

Los servicios centrales fueron desplegados en la plataforma Render [20]. Como muestra la Figura 3.11, el despliegue comprende la API restful-api-inmap, ejecutada en un contenedor Docker [37]; la base de datos PostgreSQL inmap-db; y la aplicación web de administración inmap-admin, publicada como sitio estático. La API y la base de datos se encuentran alojadas en la misma región, mientras que la interfaz web dispone de distribución global.

SERVICE NAME 3	STATUS	RUNTIME	REGION	UPDATED ↓	
restful-api-inmap	✓ Deployed	Docker	Oregon	6d	...
inmap-db	✓ Available	PostgreSQL 18	Oregon	6d	...
inmap-admin	✓ Deployed	Static	Global	1mo	...

Figura 3.11. *Despliegue de los componentes de software en Render.*

3.4.3. Autenticación y seguridad

La aplicación móvil “InMap” está destinada a usuarios generales y realiza principalmente consultas de lectura de información, por lo que no requiere autenticación. En cambio, la aplicación web de administración permite crear, modificar y eliminar información del sistema. Por este motivo, estas operaciones se encuentran restringidas a usuarios autenticados con los permisos correspondientes.

Para proteger estos recursos se implementaron mecanismos de autenticación y autorización mediante **Spring Security** [23]. La autenticación verifica la identidad del usuario, mientras que la autorización determina qué operaciones puede realizar según su rol.

Se consideraron mecanismos como HTTP Basic, claves de API, OAuth 2.0 y autenticación basada en tokens. Para el prototipo se seleccionó esta última alternativa mediante **JSON Web Tokens (JWT)**, principalmente porque:

- Permite implementar una autenticación sin mantener sesiones en el servidor.
- Se integra directamente con Spring Security y con la aplicación web [1], [5].
- Incluye información como la identidad, el rol y el tiempo de expiración del usuario.
- Permite validar la integridad del token mediante una firma digital.
- Facilita el envío de las credenciales en cada solicitud mediante el encabezado Authorization: Bearer.

Cuando el administrador inicia sesión con credenciales válidas, la API genera un token con una vigencia de tres horas. La aplicación web almacena este token y lo incorpora en las solicitudes dirigidas a los recursos protegidos. El backend verifica su firma, vigencia y permisos antes de ejecutar la operación solicitada. Una vez expirado, el usuario debe autenticarse nuevamente.

La comunicación se realiza mediante HTTPS para proteger las credenciales y los tokens durante su transmisión. Al cerrar sesión, el token se elimina del almacenamiento de la aplicación web, impidiendo que vuelva a utilizarse desde ella.

El sistema también permite cambiar la contraseña, crear nuevos perfiles administrativos y recuperar el acceso mediante un enlace enviado al correo registrado. Las contraseñas no se almacenan directamente, sino mediante el mecanismo de codificación configurado en Spring Security.

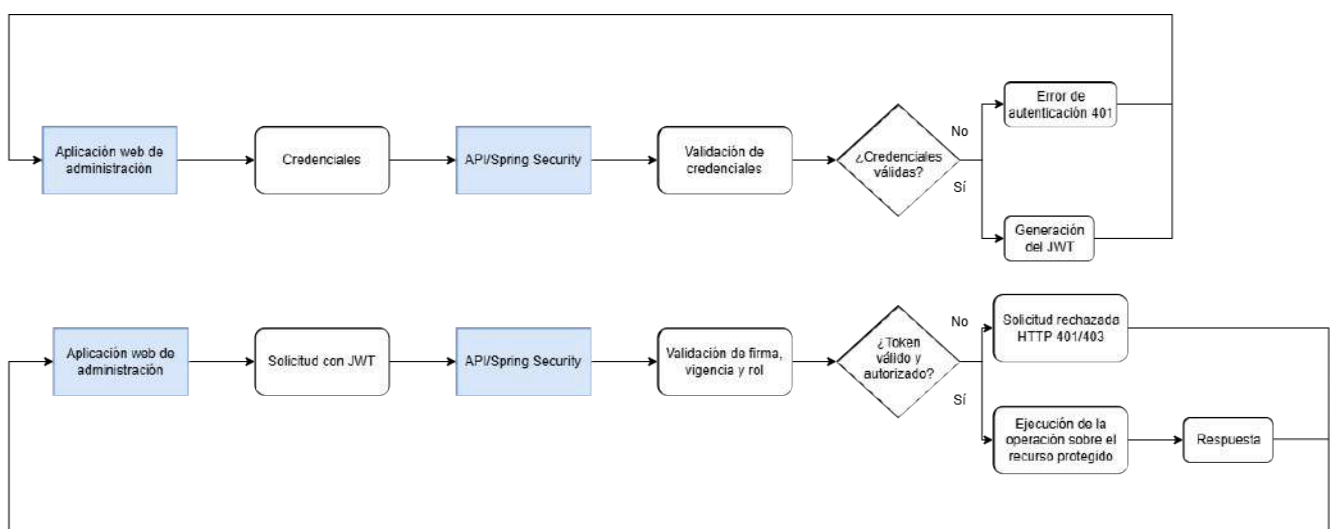


Figura 3.12. Flujo de autenticación con Spring Security.



Figura 3.13. *Pantalla de login, aplicación web de administrador.*

3.5. Aplicación móvil

La aplicación móvil “InMap”, desarrollada para dispositivos Android, constituye la interfaz principal del usuario. La aplicación captura las señales emitidas por las balizas BLE y obtiene sus mediciones de RSSI, que son procesadas mediante el método de estimación descrito en la [Sección 3.2](#) para determinar la posición del usuario en tiempo real. A partir de esta ubicación, permite seleccionar un destino, calcular una ruta y proporcionar indicaciones visuales para guiar al usuario dentro del espacio.

En esta sección se presentan la arquitectura, las tecnologías empleadas y las principales funcionalidades de la aplicación.

3.5.1. Navegación

La aplicación móvil fue desarrollada en **Dart** [35] mediante el framework **Flutter** [34], utilizando Android Studio como entorno de trabajo. Flutter permitió integrar la interfaz de usuario con la lógica de posicionamiento y navegación, además de incorporar elementos gráficos dinámicos sin afectar significativamente la fluidez de la aplicación.

Para representar el interior del establecimiento se construyó un modelo tridimensional mediante **SketchUp** [29]. Este modelo fue incorporado a **Unity** [30], utilizado como motor de renderizado para mostrar el mapa y controlar los elementos visuales de la navegación, como la posición del usuario, el destino seleccionado, la ruta calculada y las zonas bloqueadas. Unity se encuentra integrado dentro de la aplicación móvil y se comunica con Flutter, que conserva el control de la interfaz, la lógica de negocio y el acceso a la información del sistema.

Antes de calcular una ruta, el plano se transforma en una representación que pueda ser procesada por el algoritmo. Para ello, el espacio se divide en una cuadrícula y cada celda se clasifica como transitable u obstáculo, según corresponda a pasillos, aulas, paredes u

otras zonas no accesibles. De esta manera, se genera una representación navegable del entorno que evita que las rutas atravesasen espacios físicamente imposibles de recorrer.

A partir de la posición estimada y del destino seleccionado, el algoritmo **A*** utiliza esta cuadrícula para calcular una ruta válida. El resultado se envía a Unity, donde se representa sobre el modelo tridimensional y se actualiza visualmente para guiar al usuario hasta el destino [11].



Figura 3.14. Ubicación actual del usuario representada en la aplicación.

3.5.2. Cálculo de la ruta más corta

La etapa de direccionamiento tiene como objetivo encontrar una ruta válida entre la posición estimada del usuario y el destino seleccionado, considerando las restricciones físicas del entorno y las zonas transitables representadas en la cuadrícula.

Para esta tarea se evaluaron los algoritmos de Dijkstra y A*. Ambos permiten obtener la ruta más corta, pero A* incorpora una heurística basada en la distancia Manhattan al destino, lo que orienta la búsqueda y reduce la exploración de nodos innecesarios.

Por este motivo, se seleccionó el **algoritmo A*** para el cálculo de rutas del sistema. Su funcionamiento resulta apropiado para la representación del entorno mediante una grilla y permite obtener una ruta transitable con un menor costo de búsqueda.

El cálculo de la ruta entre la posición del usuario y el destino seleccionado se implementó mediante el algoritmo A*, como se explicó anteriormente. Durante las primeras pruebas, las trayectorias presentaron segmentos en zigzag y una tendencia a aproximarse demasiado a las paredes, debido a la representación cuadriculada del entorno. Para obtener recorridos más claros y naturales se incorporaron dos optimizaciones:

- **Penalización por proximidad a obstáculos (*penalty grid*):** se asigna un costo adicional a las celdas cercanas a paredes y otros obstáculos. Este costo disminuye progresivamente a medida que aumenta la distancia, favoreciendo recorridos alejados de los límites y más próximos al centro de los pasillos.
- **Simplificación por línea de visión (*string pulling*):** una vez calculada la ruta, se utiliza el algoritmo de Bresenham para comprobar si existe un trayecto directo y libre de obstáculos entre puntos no consecutivos [9]. Cuando esto es posible, se eliminan los puntos intermedios redundantes y se reemplazan por segmentos rectos.

La combinación de ambas técnicas permite conservar una ruta válida y transitable, reducir los cambios innecesarios de dirección y mejorar su representación visual en la aplicación.

Luego de realizar el cálculo y obtener la ruta más corta, se visualiza en la aplicación con instrucciones claras para que el usuario pueda llegar a su destino, el cual se representa mediante una bandera roja, como se puede ver en la Figura 3.15.

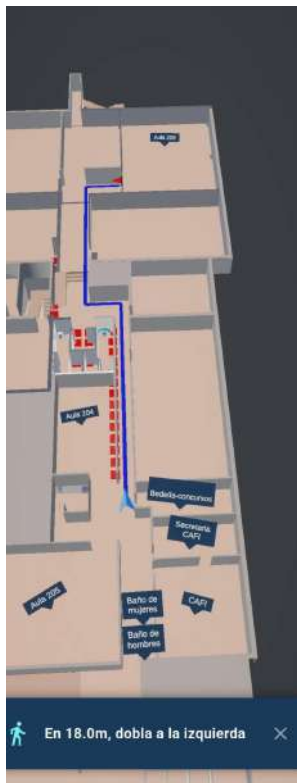


Figura 3.15. *Mejor ruta proporcionada por la aplicación para llegar al destino.*

3.5.3. Interacción con el usuario y experiencia de búsqueda

La aplicación dispone de un buscador que permite localizar aulas, docentes y materias, como se muestra en la Figura 3.16. La información necesaria se carga de manera asíncrona al iniciar la aplicación, lo que permite realizar las búsquedas sin interrumpir la interacción con el mapa tridimensional. Una vez seleccionado un resultado, la aplicación identifica el destino correspondiente, calcula la ruta y ajusta suavemente la cámara para facilitar su visualización.

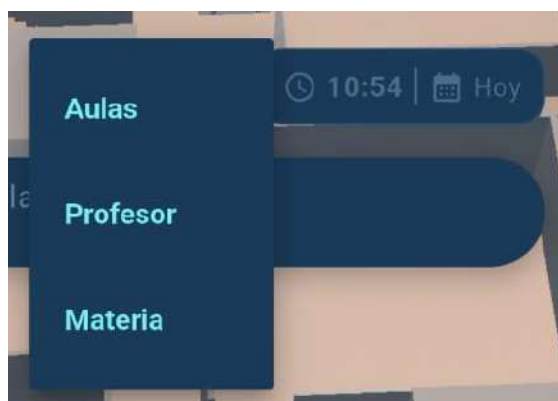


Figura 3.16. Opciones del buscador.

Para representar la ubicación y orientación del usuario se sustituyó el punto utilizado inicialmente por un ícono con forma de flecha. Su dirección se actualiza mediante la brújula del dispositivo móvil, lo que permite relacionar la orientación del usuario con la representación del espacio.

Además, se incorporó la visualización contextual de los nombres de los recintos cercanos. A medida que el usuario se desplaza, la aplicación muestra las referencias relevantes de su entorno y actualiza el seguimiento de la cámara. Estas funcionalidades permiten reconocer con mayor claridad la ubicación actual, la dirección de desplazamiento y los espacios próximos.

3.5.4. Evaluación en entorno real y problemas detectados

Durante la evaluación de la aplicación y del sistema completo en el entorno real, se detectaron distintas dificultades relacionadas con la estimación de la posición, la navegación y su representación visual. La Tabla 3.4 presenta los problemas identificados y las soluciones implementadas en cada caso.

Tabla 3.4. Problemas encontrados y soluciones propuestas.

Módulo/bloque	Problema detectado	Solución implementada
Estimación de la posición	La red neuronal MLP presentó estimaciones inestables en el entorno real, con saltos hacia ubicaciones alejadas de la posición efectiva del usuario.	Se reemplazó la red neuronal implementando un sistema de localización basado en el cálculo de un Centroide Ponderado. La estimación resultante se estabiliza aplicando restricciones geométricas basadas en la topología del mapa y límites de velocidad humana, garantizando una trayectoria coherente.
Cámara y visualización 3D	La cámara cambiaba su orientación ante pequeñas oscilaciones de la posición estimada, lo que generaba una visualización inestable y dificultaba el seguimiento del recorrido.	Se eliminó el giro automático de la cámara. Se conservaron el acercamiento, la brújula y la visualización de los nombres de los recintos cercanos.

Representación y orientación del usuario	La incorporación de una flecha orientada mediante la brújula generó un volumen excesivo de actualizaciones continuas. Este tráfico constante saturó el puente de comunicación asíncrona entre Flutter y Unity, produciendo un retraso entre la posición calculada y su representación visual en el modelo tridimensional.	Para reducir la carga de procesamiento, se introdujeron umbrales espaciales (6°) y temporales (250 ms) en el envío de datos magnéticos desde Flutter.
Cálculo y representación de la ruta	Durante la generación de las indicaciones aparecían segmentos diagonales que no representaban correctamente el recorrido esperado.	Se delegó el control geométrico del trazado al motor gráfico. Flutter procesa y envía únicamente los vértices de la ruta ortogonal. En Unity, la línea se renderiza con estos vértices fijos y solamente se actualiza de forma dinámica la posición del primer vértice (índice 0) anclándolo a la ubicación actual del usuario.
Suavizado del desplazamiento	Durante el desplazamiento, la flecha avanzaba y retrocedía de manera simultánea o realizaba movimientos que no se correspondían con el recorrido del usuario.	Se ajustó el filtro de suavizado aplicado a las actualizaciones de posición.
Grilla de navegación y zonas bloqueadas	Después de incorporar la flecha, tanto el indicador del usuario como la línea de ruta tendían a aproximarse demasiado a las paredes y, en algunos sectores, podían ingresar en zonas no transitables.	Se ajustó la definición de las zonas bloqueadas, especialmente en sectores conflictivos como el acceso a los baños, para impedir el ingreso de la ruta en espacios no transitables.
Inicialización y concurrencia del sistema	En ocasiones, la aplicación quedaba bloqueada en el estado de "buscando balizas" al arrancar, ya que el escáner Bluetooth comenzaba a operar antes de que se completara la descarga del mapa y las zonas bloqueadas.	Se reestructuró el ciclo de vida inicial de la aplicación resolviendo la condición de carrera. Se encadenaron las funciones asíncronas para garantizar que el hardware de posicionamiento solo se active una vez que la topología del mapa esté completamente cargada en memoria.
Adquisición y filtrado de señales BLE	Las lecturas crudas de RSSI de los ESP32 presentaban fluctuaciones severas y pérdidas temporales de paquetes, lo que desestabilizaba el cálculo de posicionamiento.	Se diseñó un sistema de procesamiento por ventanas de tiempo con un buffer de memoria a corto plazo. Ante la ausencia temporal de lectura de una baliza, el sistema retiene su último promedio válido por un máximo de 2 segundos antes de descartarlo, mitigando los picos de ruido y las caídas de señal.

3.6. Aplicación de gestión (administrador)

La aplicación web de administración fue desarrollada para permitir que usuarios autenticados y autorizados gestionen la información del establecimiento y consulten el

estado operativo de las balizas. Desde esta plataforma es posible crear, consultar, modificar y eliminar los datos utilizados por la aplicación móvil “InMap”.

La aplicación se comunica con la API RESTful mediante solicitudes HTTPS. La API procesa cada petición, aplica las validaciones y reglas de negocio correspondientes y realiza las operaciones necesarias sobre la base de datos. Las respuestas se devuelven a la aplicación web en formato JSON para su presentación al usuario.

3.6.1. Arquitectura de software

La aplicación de administración fue desarrollada como una plataforma web basada en una arquitectura de componentes. Para su implementación se utilizó **React** [36] con JavaScript y sintaxis JSX, mientras que la presentación visual se construyó mediante **Tailwind CSS**, lo que facilitó la creación de una interfaz consistente y adaptable a diferentes tamaños de pantalla.

La interfaz se organiza mediante una barra de navegación lateral que proporciona acceso a los distintos módulos del sistema: plano, materias, horarios, personal, reportes y configuración. La pantalla principal, presentada en la Figura 3.17, resume parte de la información gestionada y permite acceder a las operaciones principales. Las funcionalidades de cada módulo se describen en la sección siguiente.

La aplicación se comunica con la API RESTful mediante solicitudes HTTPS para consultar y modificar la información almacenada. Como se indicó en la [Sección 3.4.2](#), actualmente se encuentra desplegada en Render como un sitio web estático.

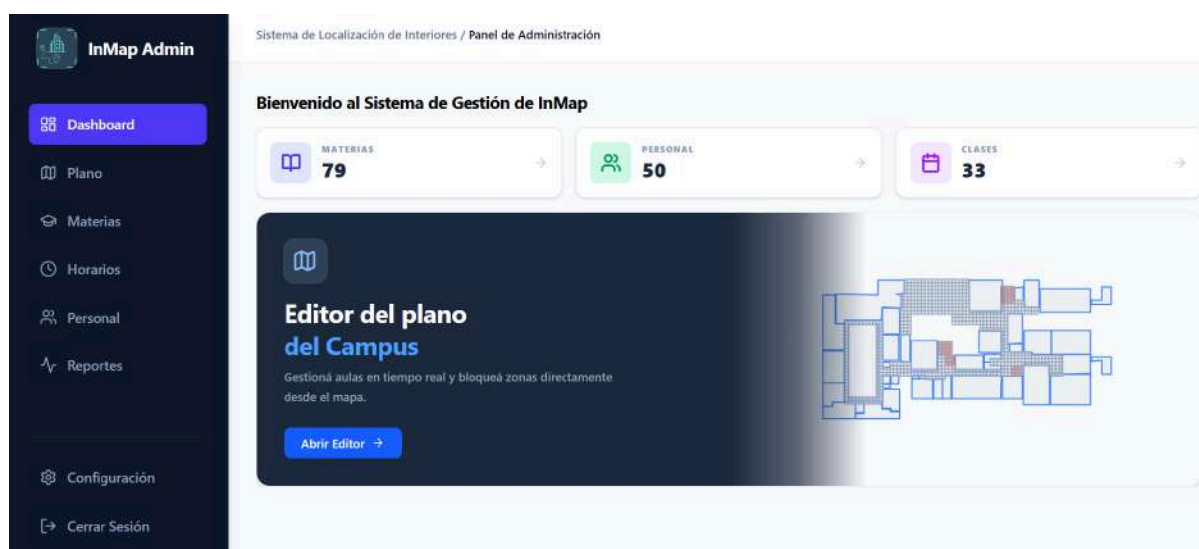


Figura 3.17. Pantalla principal de la aplicación de gestión.

Al momento de elaboración del informe, la aplicación se encuentra disponible en <https://inmap-admin.onrender.com/>. Se puede acceder con las siguientes credenciales de acceso para pruebas:

Usuario: nuevo_admin

Contraseña: nuevo_admin

3.6.2. Funcionalidades específicas.

La aplicación implementa un modelo de acceso basado en roles para proteger la información administrada. Como se explicó en la [Sección 3.4.3](#), la sesión se gestiona mediante JSON Web Tokens (JWT), almacenados en el cliente e incorporados en el encabezado de las solicitudes dirigidas a recursos protegidos. Se definieron dos perfiles:

- **Administrador:** puede consultar, crear, modificar y eliminar información mediante las operaciones habilitadas por la API. También puede gestionar usuarios y actualizar la configuración disponible.
- **Visualizador:** dispone únicamente de permisos de consulta. La interfaz oculta las acciones de edición y la API rechaza cualquier operación que no se corresponda con los permisos del usuario.

Las principales funcionalidades de la aplicación son las siguientes:

Plano interactivo: Permite visualizar la distribución de los recintos y la ubicación de las balizas BLE. Además, permite bloquear temporalmente sectores transitables para impedir que la aplicación móvil calcule rutas a través de ellos.

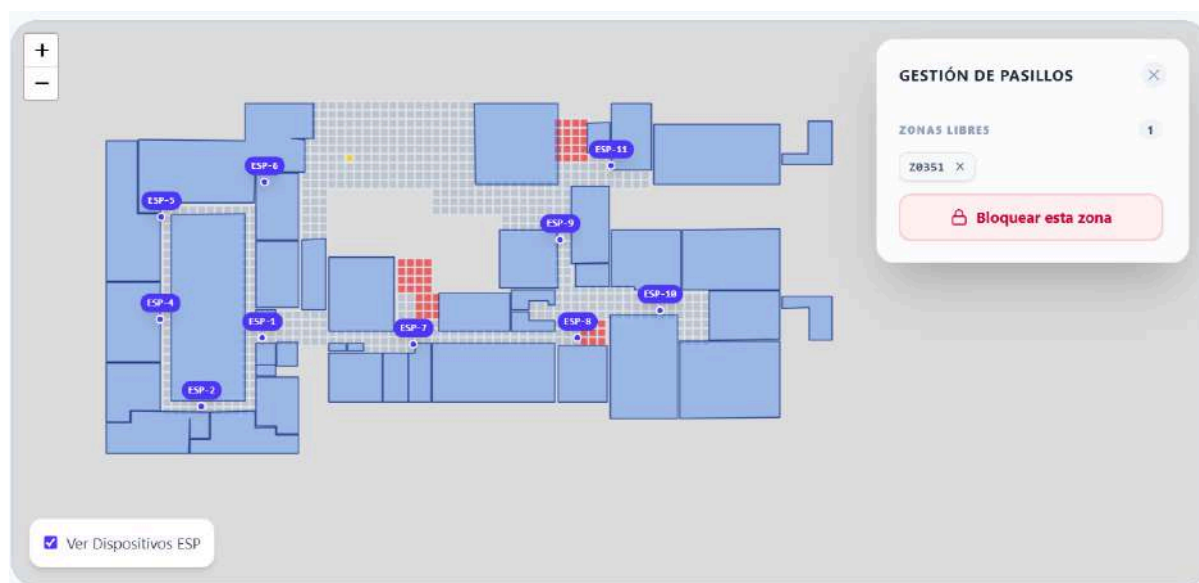


Figura 3.18. Mapa interactivo en formato 2D.

Gestión de información: los módulos de personal, materias, horarios y asignaciones permiten realizar las operaciones de creación, consulta, actualización y eliminación necesarias para mantener la información utilizada por el sistema:

- **Personal:** administra la información del personal registrado.



Figura 3.19. Gestión del personal.

- **Materias:** gestiona el catálogo de asignaturas.



Figura 3.20. Gestión de materias.

- **Horarios y asignaciones:** relaciona materias, personal, recintos y franjas horarias. También permite definir turnos para organizar la grilla horaria.

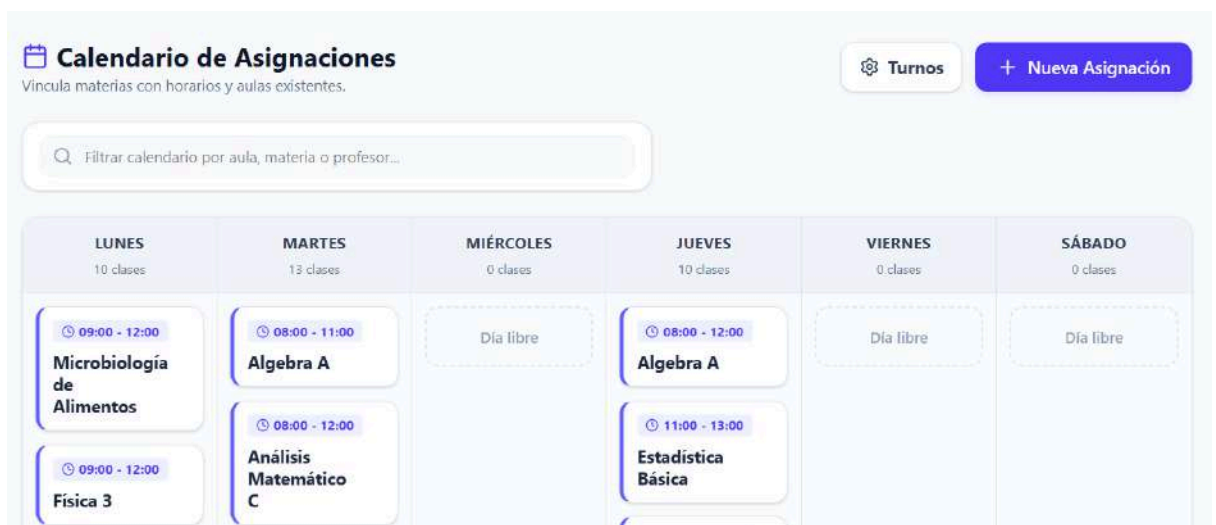


Figura 3.21. Gestión de horarios y asignaciones.

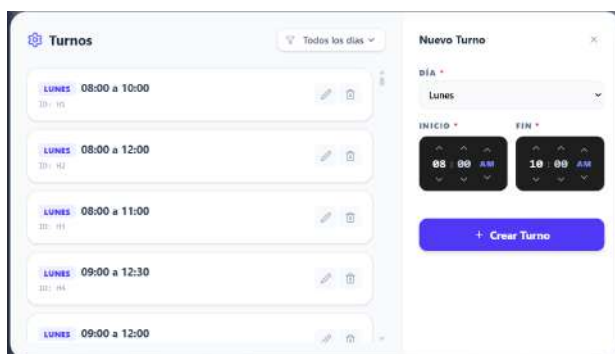


Figura 3.22. Gestión de turnos.

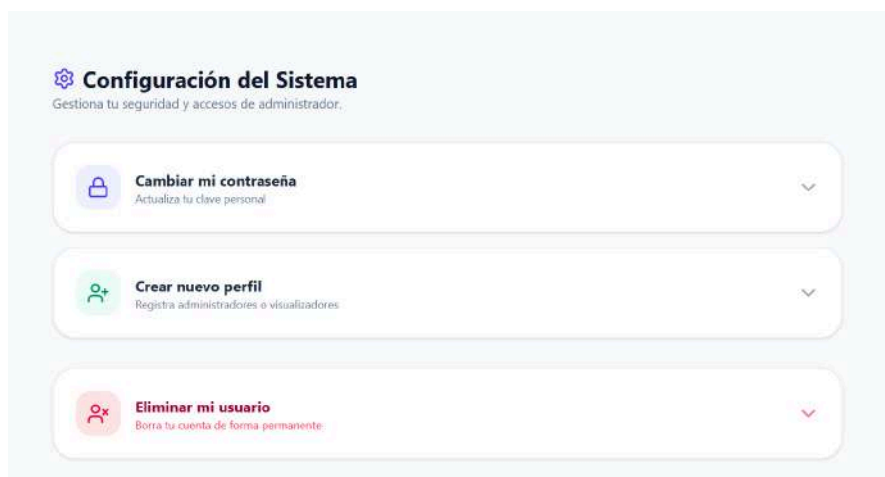
Esta información permite que los usuarios de “InMap” busquen una materia o una persona y obtengan el recinto correspondiente según el día y el horario de la consulta.

Reportes y monitoreo de nodos BLE: presenta los reportes enviados por los dispositivos ESP32 y permite consultar información como el nivel estimado de batería, el tiempo de actividad, el motivo del último reinicio y el estado operativo calculado para cada baliza.



Figura 3.23. Monitoreo balizas BLE.

Configuración: Permite que los usuarios puedan cambiar su contraseña, crear nuevo perfil (solo para Administradores) y dar de baja la cuenta activa de forma permanente.



Capítulo 4

Verificación y resultados

4.1. Estrategia y ambiente de pruebas

La estrategia de pruebas tuvo como objetivo verificar el cumplimiento de los requerimientos funcionales y no funcionales establecidos para el prototipo, así como evaluar el comportamiento individual y conjunto de sus principales componentes. Para ello, se aplicó un enfoque progresivo: primero se comprobaron los módulos de manera aislada, luego se verificó la comunicación entre los distintos subsistemas y, finalmente, se evaluó el funcionamiento completo en el entorno físico seleccionado.

Los componentes evaluados fueron las balizas BLE basadas en ESP32, la API RESTful, la base de datos PostgreSQL con PostGIS, la aplicación web de administración y la aplicación móvil InMap. Las pruebas se organizaron en tres ambientes principales: desarrollo y pruebas automatizadas, integración del sistema y evaluación en el entorno real.

En el ambiente de desarrollo se realizaron pruebas unitarias y de componentes en las estaciones de trabajo de las desarrolladoras. Para el backend se utilizaron JUnit [31] y Mockito [32], con el propósito de validar de forma aislada la lógica de negocio y simular sus dependencias sin requerir una conexión permanente con la base de datos. En la aplicación web de administración se emplearon Vitest [38], React Testing Library [39] y JSDOM para verificar la lógica de los componentes, su renderizado y las interacciones principales desde la perspectiva del usuario.

En el subsistema físico se efectuaron comprobaciones manuales de componentes mediante el monitor serie de Arduino IDE. Estas pruebas permitieron observar la inicialización del ESP32, el funcionamiento del temporizador, los ciclos de actividad y reposo, la medición de la batería y los reportes de estado. La emisión de los paquetes BLE y sus identificadores se verificó mediante una aplicación de escaneo instalada en un dispositivo móvil.

Las pruebas de API se realizaron mediante Postman, desde donde se enviaron solicitudes HTTP a los diferentes endpoints y se comprobaron los códigos de estado, los datos enviados y recibidos, el formato de las respuestas JSON y las restricciones de autenticación y autorización.

En el ambiente de integración se verificó la comunicación entre la API y PostgreSQL, entre la aplicación web y la API, entre la aplicación móvil y la API, y entre los dispositivos ESP32 y el backend. Para estas pruebas se utilizaron los servicios desplegados en Render: la API ejecutada en un contenedor Docker, la base de datos PostgreSQL con PostGIS y la aplicación web publicada como sitio estático. También se comprobó el intercambio de información mediante HTTPS y la correcta persistencia de los datos.

La evaluación del sistema completo se llevó a cabo en el segundo piso de la Facultad de Ingeniería de la Universidad Nacional de Mar del Plata. Este espacio permitió probar el prototipo en un escenario interior real, compuesto por aulas, pasillos, intersecciones, paredes, mobiliario, circulación de personas y otros dispositivos inalámbricos capaces de afectar la propagación de las señales. Las balizas se distribuyeron en posiciones previamente definidas sobre el plano y las pruebas se realizaron utilizando dispositivos móviles con sistema operativo Android.

Como criterio general de aprobación, cada prueba se consideró satisfactoria cuando el resultado obtenido coincidió con el comportamiento esperado y con los requerimientos asociados. En particular, para la estimación de la posición se adoptó como criterio un error medio menor o igual a 2,5 m dentro del escenario definido. Cuando se detectaron resultados incorrectos o comportamientos inestables, se registró la incidencia, se modificó el componente involucrado y se repitieron las pruebas correspondientes, incluyendo pruebas de regresión sobre las funcionalidades previamente verificadas.

Las pruebas manuales y sus resultados se documentaron en la planilla de [Casos de prueba-Baldoma-Maña](#), mientras que las pruebas automatizadas quedaron registradas en los respectivos proyectos de software. En esta planilla se registraron sobre todo pruebas funcionales, que indican que se valida una función observable del sistema desde el punto de vista del usuario o de un requerimiento. La definición de los ambientes, tipos y criterios utilizados se encuentra desarrollada en el [Plan de Pruebas](#) del sistema.

4.2. Tipos de pruebas

A partir de la estrategia y los ambientes descritos anteriormente, se ejecutaron distintos tipos de pruebas para verificar el funcionamiento aislado de los componentes, su integración y el comportamiento del sistema completo. La Tabla 4.1 resume las pruebas realizadas, los componentes evaluados y las herramientas o mecanismos empleados en cada caso.

Tabla 4.1. *Tipos de pruebas ejecutadas sobre los componentes del sistema.*

Tipo de prueba	Componentes evaluados	Herramientas o procedimiento utilizado
Unitarias y de componentes	API RESTful y aplicación web de administración	JUnit y Mockito para la lógica del backend; Vitest, React Testing Library y JSDOM para los componentes y la lógica de la aplicación web.
API y backend	Endpoints, servicios, validaciones, manejo de errores y autenticación	Solicitudes ejecutadas mediante Postman y pruebas automatizadas con JUnit. Se verificaron operaciones GET, POST, PUT y DELETE, códigos de estado y respuestas JSON.

Integración	Aplicación web-API, aplicación móvil-API, API-PostgreSQL y ESP32-API	Ejecución conjunta de los componentes desplegados en Render y verificación de las comunicaciones mediante HTTPS y mensajes JSON.
Frontend web	Interfaz de administración, formularios, tablas, navegación y control de acceso según el rol	Vitest, React Testing Library y JSDOM, complementados con pruebas manuales en el navegador.
Aplicación móvil	Captura de señales BLE, carga del mapa, búsqueda de destinos, representación de la posición y navegación	Pruebas manuales en dispositivos Android y observación del intercambio de información entre Flutter y Unity.
Sistema o extremo a extremo	Balizas, aplicación móvil, API, base de datos y aplicación web	Recorridos completos en el segundo piso de la Facultad: emisión BLE, estimación de posición, selección de destino, cálculo de ruta, actualización de información y monitoreo.
Posicionamiento y navegación	Algoritmo WCL, filtros espaciales, restricción topológica y algoritmo A*	Mediciones en puntos y recorridos conocidos, comparación entre posición estimada y posición real, y verificación visual de las rutas sobre el mapa.
Rendimiento en el entorno real	Aplicación móvil y comportamiento general del sistema	Ejecución con circulación de personas y múltiples dispositivos Bluetooth activos. Se observaron la continuidad de las actualizaciones, los tiempos de respuesta y la fluidez de la representación gráfica.
Consumo, autonomía y recuperación	Sistema de alimentación, Battery Shield, batería y ESP32	Mediciones de descarga, simulación de cortes del suministro eléctrico y comprobación de la continuidad de la emisión BLE durante la conmutación.
Estabilidad del hardware	Firmware, BLE, Wi-Fi, temporizador y ciclos de Deep Sleep	Monitor serie de Arduino IDE, observación de causas de reinicio, memoria disponible, reportes periódicos y aplicación BLE Scanner.
Seguridad	API RESTful y aplicación web de administración	Pruebas de inicio de sesión, validación y vencimiento de JWT, control de roles, acceso a endpoints protegidos, recuperación y cambio de contraseña.

Usabilidad y aceptación	Aplicación móvil y aplicación web de administración	Ejecución de tareas representativas y demostración del sistema ante los directores y responsables de la validación.
-------------------------	---	---

4.3. Resultados obtenidos

La Tabla 4.2 resume los principales resultados obtenidos durante las pruebas y su estado respecto de los criterios de aceptación establecidos. Estos resultados corresponden al prototipo evaluado en las condiciones y el entorno descritos previamente.

Tabla 4.2. *Síntesis de los resultados obtenidos.*

Área evaluada	Resultado principal	Estado
API y base de datos	Las operaciones y consultas evaluadas devolvieron las respuestas esperadas y persistieron correctamente la información.	Aprobado
Aplicación web	Se verificaron las operaciones de consulta y administración, junto con las restricciones correspondientes a cada rol.	Aprobado
Hardware	Las balizas mantuvieron la emisión BLE y enviaron los reportes operativos previstos.	Aprobado con ajustes
Respaldo energético	La batería mantuvo temporalmente el funcionamiento ante interrupciones del suministro.	Aprobado
Posicionamiento	El WCL con restricción topológica obtuvo un error medio de 2,2 m, inferior al máximo requerido de 2,5 m.	Aprobado
Navegación	Se calcularon rutas transitables y se evitaron las zonas bloqueadas.	Aprobado con ajustes
Aplicación móvil	La integración Flutter–Unity permitió representar la posición y las rutas, aunque requirió filtros para reducir retrasos e inestabilidad.	Aprobado con observaciones
Sistema completo	Se verificó el flujo desde la emisión BLE hasta la representación y administración de la información.	Aprobado en el escenario evaluado

Capítulo 5

5.1. Conclusiones generales

El desarrollo del proyecto permitió demostrar la viabilidad técnica de construir un prototipo propio de localización y navegación en interiores. Durante el trabajo se integraron las balizas BLE, la aplicación móvil, la API RESTful, la base de datos espacial y la aplicación web de administración. Las pruebas realizadas permitieron verificar el funcionamiento conjunto de estos componentes y el cumplimiento de los requerimientos evaluados en el escenario definido.

Respecto del objetivo de localización, el algoritmo de Centroide Ponderado (WCL) con restricciones topológicas presentó el comportamiento más estable entre las alternativas analizadas. Este método obtuvo un error medio de 2,2 m en el entorno de prueba, por debajo del máximo de 2,5 m establecido en el requerimiento funcional RF1. Por lo tanto, se verificó el cumplimiento de dicho requerimiento en el sector evaluado del segundo piso de la Facultad de Ingeniería.

En relación con la continuidad del hardware, el Sistema de Alimentación Ininterrumpida integrado en cada prototipo permite mantener temporalmente la emisión BLE ante interrupciones del suministro eléctrico. Asimismo, los reportes enviados mediante Wi-Fi permiten supervisar desde la aplicación web el estado operativo de las balizas y consultar indicadores como el nivel estimado de batería, el tiempo de actividad y el motivo del último reinicio.

En cuanto a la gestión y ejecución del proyecto, el desarrollo respetó satisfactoriamente el orden lógico de las tareas y la división de responsabilidades establecidas en la planificación inicial. Cada integrante cumplió con los roles asignados originalmente, lo que permitió avanzar de manera estructurada en la investigación, el desarrollo de software y la construcción del hardware. Sin embargo, la fecha de finalización del proyecto se extendió más allá de lo estipulado en el diagrama de Gantt original, requiriendo un margen adicional de aproximadamente un mes y medio. Este desvío temporal se debió, en gran medida, a una expansión en el alcance del sistema para incrementar su robustez y funcionalidad. Durante el ciclo de desarrollo se identificaron e incorporaron tareas sustanciales que no estaban contempladas en la planificación preliminar, destacándose la integración de un entorno tridimensional (3D), el desarrollo de la aplicación web para la administración y el monitoreo del sistema, y la resolución de problemas detectados durante la fase de pruebas físicas. Estas incorporaciones resultaron fundamentales para entregar una solución tecnológica funcionalmente más completa, escalable y tolerante a fallos ante posibles cortes en el suministro eléctrico.

El análisis de costos mostró que el prototipo puede construirse utilizando componentes disponibles comercialmente y tecnologías de código abierto. El costo calculado corresponde a la fabricación de una unidad con precios minoristas, por lo que constituye una estimación preliminar y no un costo de producción en serie. Aunque existen balizas BLE básicas de menor precio, estas no incorporan necesariamente las mismas prestaciones de alimentación, respaldo energético y monitoreo remoto. En consecuencia, los resultados

permiten considerar que la solución presenta un costo potencialmente competitivo para las funcionalidades implementadas, aspecto que deberá validarse mediante cotizaciones por volumen.

Finalmente, la arquitectura y los componentes desarrollados pueden reutilizarse para implementar el sistema en otros espacios interiores. Sin embargo, la adaptación no es automática: cada nueva instalación requiere construir el modelo tridimensional del entorno, representar sus zonas transitables y bloqueadas, definir la cuadrícula de navegación, distribuir las balizas y validar nuevamente el algoritmo de posicionamiento. Por lo tanto, el proyecto proporciona una base tecnológica reutilizable, pero cada despliegue exige una etapa particular de configuración y evaluación.

5.2. Trabajos futuros

A partir de los resultados obtenidos, se proponen diversas líneas de trabajo futuro para continuar la evolución del sistema. Entre ellas, se destaca la ampliación de la recolección de datos RSSI en distintas condiciones reales, con el propósito de reconsiderar el uso de modelos neuronales para la estimación de la posición, así como también probar algunos nuevos. También se propone extender las pruebas a espacios de mayor superficie y con una mayor cantidad de balizas, usuarios y dispositivos Bluetooth activos. Esta ampliación requerirá construir y configurar el modelo espacial correspondiente a cada nuevo entorno.

En relación con el subsistema físico, queda pendiente implementar un mecanismo para detectar y reportar niveles de señal anormalmente bajos o variables, de acuerdo con el requerimiento RF12. Asimismo, podría estudiarse el desarrollo de herramientas que simplifiquen la configuración de las balizas, la construcción del modelo tridimensional y la definición de las zonas transitables.

Desde el punto de vista del software, se plantea integrar la información académica con sistemas institucionales, como SIU Guaraní, y desarrollar una versión de la aplicación móvil compatible con iOS, con el fin de ampliar el alcance y la utilidad de la solución.

Bibliografía

Páginas web/blogs de tecnología

- [1] Aggarwal, B. (s.f.). Master authentication using Spring Security + JWT. Medium. <https://medium.com/@bharataggarwal143/master-authentication-using-spring-security-jwt-411fef32b6e8>
- [2] Blueiot. (s.f.). *GPS vs. indoor positioning*. <https://www.blueiot.com/blog/gps-vs-indoor-positioning.html>.
- [3] Bootcamp 10x. (s.f.). Arquitectura de backend. <https://bootcamp.10x.gt/docs/desarrollo-web-y-movil/net-core-web-api/arquitectura-de-backend>.
- [4] CampusMVP. (s.f.). Acceso a datos con Java: JDBC, JPA, Spring Data JPA y Hibernate. ¿Qué es cada uno y cuál elegir?

<https://www.campusmvp.es/recursos/post/acceso-a-datos-con-java-jdbc-jpa-spring-d-ata-jpa-y-hibernate-que-es-cada-uno-y-cual-elegir.aspx>.

- [5] Edicom Group. (s.f.). Spring Security: Conceptos y estrategias. Tech Blog. <https://careers.edicomgroup.com/blogtech/spring-security-conceptos-y-estrategias/>
- [6] Negrin, J. (2023, 23 de noviembre). Construyendo una RESTful API con Spring Boot: Integración de DDD y arquitectura hexagonal. Medium. <https://medium.com/@juannegrin/construyendo-una-restful-api-con-spring-boot-integraci%C3%B3n-de-ddd-y-arquitectura-hexagonal-af824a3a4d05>
- [7] The Walt Disney Company. (s.f.). *Frequently asked questions about Bluetooth technology at Disney Parks and Experiences*. <https://privacy.thewaltdisneycompany.com/en/resortble/>

Artículos de revistas científicas

- [8] Bahl, P. y Padmanabhan, V. N. (2000). RADAR: An in-building RF-based user location and tracking system. En IEEE INFOCOM 2000. Conference on Computer Communications. Proceedings. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Vol. 2, pp. 775-784). IEEE. <https://www.cse.iitb.ac.in/~varsha/allpapers/radar.pdf>
- [9] Bresenham, J. E. (1965). Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1), 25–30. <https://doi.org/10.1147/sj.41.0025>
- [10] Ferrero-López, A., Gallego, A. J. y Lozano, M. Á. (2025). Posicionamiento en interiores mediante Bluetooth de baja energía: un enfoque de red neuronal basado en huellas digitales. *Internet de las cosas*, 31, Artículo 101565. <https://www.sciencedirect.com/science/article/pii/S2542660525000782>
- [11] Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2), 100-107. <https://doi.org/10.1109/TSSC.1968.300136>.
- [12] Loshchilov, I., & Hutter, F. (2017). *Decoupled weight decay regularization*. arXiv. <https://doi.org/10.48550/arXiv.1711.05101>
- [13] Singh, N., Choe, S., & Punmiya, R. (2021). Machine learning based indoor localization using Wi-Fi RSSI fingerprints: An overview. *IEEE Access*, 9, 127115–127131. <https://doi.org/10.1109/ACCESS.2021.3111083>
- [14] Yu, W., & Yu, C. (2025). Application of weighted centroid algorithm based on weight correction in node localization of wireless sensor networks. *Scientific Reports*, 15, 23400. <https://www.nature.com/articles/s41598-025-08336-0>

Libros

- [15] Elmasri & Navathe (Fundamentos de sistemas de bases de datos): Elmasri, R. y Navathe, S. B. (2007). Fundamentos de sistemas de bases de datos (5.ª ed.). Pearson/Addison Wesley.

Documentación oficial

- [16] Apple Inc. (2014). *Getting started with iBeacon* (versión 1.0). <https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf>

- [17] Bluetooth SIG. (2024). *The Bluetooth Low Energy primer* (Version 1.3.0). <https://www.bluetooth.com/bluetooth-le-primer/>
- [18] Espressif Systems. (2026). *ESP32 series datasheet* (Version 5.3). https://documentation.espressif.com/esp32_datasheet_en.pdf
- [19] PostGIS Development Group. (s.f.). *PostGIS documentation*. <https://postgis.net/docs/>
- [20] Render. (s.f.). *Render documentation*. <https://render.com/docs>
- [21] Spring. (s.f.-a). *Spring Boot reference documentation*. <https://docs.spring.io/spring-boot/reference/>
- [22] Spring. (s.f.-b). *Spring Data JPA reference documentation*. <https://docs.spring.io/spring-data/jpa/reference/>
- [23] Spring. (s.f.-c). *Spring Security reference*. <https://docs.spring.io/spring-security/reference/>

Mercado

- [24] Mercado Libre. (s.f.-a). *Beacon Bluetooth Outdoor BC02 IP68 SenseCAP BLE 5.0 Semakiot* [Publicación de producto]. Recuperado el 3 de septiembre de 2026, de <https://www.mercadolibre.com.ar/beacon-bluetooth-outdoor-bc02-ip68-sensecap-ble-50-semakiot/up/MLAU3934280483>
- [25] Mercado Libre. (s.f.-b). *Bluetooth BLE iBeacon Blue Charm Beacons BC021 Multibeacon* [Publicación de producto]. Recuperado el 3 de septiembre de 2026, de <https://www.mercadolibre.com.ar/bluetooth-ble-ibeacon-blue-charm-beacons-bc021-multibeacon/p/MLA2061850599>
- [26] Mercado Libre. (s.f.-c). *Módulo de bajo consumo nRF52810 5.0 Beacon I* [Publicación de producto]. Recuperado el 3 de septiembre de 2026, de <https://www.mercadolibre.com.ar/modulo-de-bajo-consumo-nrf52810-50-beacon-i-ble/p/MLA2050793226>
- [40] Mercado Libre (s.f.-d). *Pila 18650 recargable*. Recuperado el 1 de octubre de 2026, de <https://www.mercadolibre.com.ar/bateria-pila-18650-recargable-x4-unidades-37-vol-c-on-teton/up/MLAU253064773>

Software

- [27] JetBrains s.r.o., IntelliJ IDEA, versión 2026.1.1 [software], 2026. [En línea]. Disponible: <https://www.jetbrains.com/idea/>
- [28] Postman, Inc., Postman [software]. [En línea]. Disponible: <https://www.postman.com/>
- [29] Trimble Inc., SketchUp [software]. [En línea]. Disponible: <https://www.sketchup.com/>
- [30] Unity Technologies, *Unity* [software]. [En línea]. Disponible: <https://unity.com/>
- [31] JUnit Team, *JUnit 5* [biblioteca de software]. [En línea]. Disponible: <https://junit.org/junit5/>.

- [32] Mockito Project, *Mockito* [biblioteca de software]. [En línea]. Disponible: <https://site.mockito.org/>
- [33] PostgreSQL Global Development Group, *PostgreSQL Documentation*. [En línea]. Disponible: <https://www.postgresql.org/docs/>
- [34] Google, *Flutter* [framework de software]. [En línea]. Disponible: <https://docs.flutter.dev/>
- [35] Google, *Dart* [lenguaje de programación]. [En línea]. Disponible: <https://dart.dev/guides>
- [36] Meta Platforms, Inc., *React* [biblioteca de software]. [En línea]. Disponible: <https://react.dev/>
- [37] Docker, Inc., *Docker Documentation*. [En línea]. Disponible: <https://docs.docker.com/>
- [38] Vitest Team, *Vitest* [framework de pruebas]. [En línea]. Disponible: <https://vitest.dev/>
- [39] Testing Library, *React Testing Library*. [En línea]. Disponible: <https://testing-library.com/docs/react-testing-library/intro/>

Apéndice

A1. Documentos anexos

Plan de proyecto

[Plan de proyecto-Baldoma-Maña.](#)

Especificación de requerimientos

[Especificacion de requerimientos-Baldoma-Maña.](#)

Especificaciones funcionales

[Especificacion funcional-Baldoma-Maña.](#)

Especificaciones técnicas

[Especificacion tecnica-Baldoma-Maña.](#)

Plan de pruebas

[Plan de pruebas-Baldoma-Maña.](#)

Casos de prueba

[Casos de prueba-Baldoma-Maña](#)

Implementación de los códigos

- Código de ESP32 en: https://github.com/camibaldoma/beacons_BLE.git
- Código de la aplicación móvil en:
<https://github.com/ConstanzaMana/TrabajoFinal-Sistema-Localizacion-de-interiores/tree/main/AppInMap>
Motor 3D (Unity)
<https://github.com/ConstanzaMana/TrabajoFinal-Sistema-Localizacion-de-interiores/tree/main/InMap3D>
- Código de la aplicación de gestión (administrador):
<https://github.com/ConstanzaMana/TrabajoFinal-Sistema-Localizacion-de-interiores/tree/main/inmap-admin>
- Código de la API en: <https://github.com/camibaldoma/restful-api-inmap.git>