

# DOCUMENTACIÓN DEL PROYECTO INTEGRADO



# MAPGENDA

# MAPGENDA

Alumno: **Fernando Yucsan Chang Cam**

Centro: IES JULIO VERNE

2025

<b>Registros de cambios</b>	<b>4</b>
<b>1.- Introducción</b>	<b>4</b>
<b>2.- Estudio de Viabilidad</b>	<b>4</b>
2.1 Descripción del sistema actual	4
2.2 - Descripción del Sistema Nuevo	5
2.3 - Identificación de Requisitos del Sistema	5
2.3.1.- Requisitos de información	5
Entidad: Usuario	6
Entidad: Lugar	6
Entidad: Ubicación	7
Entidad: Ruta Personalizada	7
2.3.2 Requisitos Funcionales	8
2.3.3 Otros Requisitos	8
2.4 - Descripción de la Solución	9
2.5 - Planificación del Proyecto	9
2.5.1 - Equipo de Trabajo	9
2.5.2 - Planificación Temporal	10
2.6 - Estudio del Coste del Proyecto	10
Coste de desarrollo (estimado)	10
<b>3. Análisis del Sistema de Información</b>	<b>11</b>
3.1 Identificación del Entorno Tecnológico	11
Entorno de Desarrollo	11
Entorno de Explotación (cuando esté en producción)	12
3.2 Modelado de Datos	12
Base de datos del backend (PostgreSQL)	12
Base de datos local (Room)	12
¿Por qué son diferentes?	13
3.2.1 Modelo Entidad-Relación	13
3.2.2.- Esquema de la base de datos	16
3.1.3.- Datos de prueba	16
3.3.- Identificación de los usuarios participantes y finales	16
A. Usuarios Finales	16
A.1. Usuario Anónimo	16
A.2. Usuario Registrado	16
B. Usuarios Participantes	16
B.1. Usuario Gestor de Contenidos	17
B.2. Usuario Administrador	17
C. Sistemas Externos	17
C.1. Google Maps API	17
C.2. Google Places API	17
C.3. Google Directions API	17
C.4. Sistema de Almacenamiento Offline	17
3.4.- Identificación de subsistemas de análisis	18
3.5. Establecimiento de Requisitos	21
3.6.- Diagramas de Análisis	22

<b>3.7.- Definición de interfaces de usuario</b>	<b>24</b>
3.7.1. Especificación de Principios Generales de Interfaz	24
3.7.2.- Especificación de formatos individuales de la interfaz de pantalla	25
Pantalla de Perfil	26
Funcionalidades disponibles:	26
Sistema de Chips de Categorías	27
Pantallas Principales: Filtro y Mapa	28
Pantalla de Filtros (Selector de Categorías)	28
Pantalla de Mapa	29
Botones flotantes (FABs)	29
1. Resultados de Búsqueda en el Mapa	32
2. Listado de Lugares Offline	33
3. Ficha Detallada del Lugar (Alerta Dialog)	33
Creación y Edición de Lugares Personalizados	34
1. Activación: Botón Flotante “+”	35
2. Creación de un Lugar (Formulario Inicial)	35
3. Edición de un Lugar Existente	35
Funcionalidad de Rutas Offline	36
1. Pantalla Menú de Rutas	37
2. Edición de Ruta	37
3. Visualización en el Mapa	37
4. Edición de Rutas Guardadas	38
Aclaración Importante	38
Panel de Administración (Dashboard)	38
Características principales:	39
Identidad Visual: Nuevo Logo de MAPGENDA	41
3.7.3. Identificación de Perfiles de Usuario	42
3.7.4. Especificación de Formatos de Impresión	42
3.7.5. Especificación de la Navegabilidad entre Pantallas	43
<b>4. Construcción del Sistema</b>	<b>44</b>
Aplicación de Consumo Android	44
Construcción del Backend de la API REST	46
4.2 DESPLIEGUE	49
Despliegue y Entorno Productivo	49
Flujo de CI/CD	50
<b>5. Conclusiones</b>	<b>50</b>
<b>6. GLOSARIO DE TÉRMINOS</b>	<b>51</b>
<b>7. BIBLIOGRAFÍA DEL PROYECTO</b>	<b>52</b>

## Registros de cambios

Versión	causa	Fecha
1.0	Requisitos funcionales	14-04-25
1.1	Estudio del Coste del Proyecto	20-04-25
1.3	Modelo Entidad-Relación	28-04-25
1.4	Diagramas de Análisis	02-05-25
2.0	Últimos retoques	02-06-25
2.1	Integración final	05-06-25

## 1.- Introducción

El proyecto "Mapgenda" es una aplicación móvil de turismo personalizado que permitirá a los usuarios encontrar lugares de interés cercanos según sus preferencias, con la posibilidad de guardar favoritos y acceder a ellos sin conexión. La aplicación se basará en la integración con Google Maps API, Google Places API y Google Directions API para ofrecer una experiencia de exploración óptima, incluyendo rutas para caminar o andar en bicicleta y la descarga de datos para su uso sin conexión.

Los usuarios podrán:

- Visualizar en un mapa los negocios y puntos de interés según sus intereses (bares, cafeterías, librerías LGBTQ+, entre otros).
- Recibir indicaciones en tiempo real mientras caminan o van en bicicleta.
- Marcar lugares favoritos y sincronizarlos con una base de datos en la nube.
- Descargar datos de lugares de otras ciudades para consultarlos sin conexión.
- Recibir notificaciones cuando se acerquen a un lugar previamente guardado.

## 2.- Estudio de Viabilidad

### 2.1 Descripción del sistema actual

Actualmente existen diversas aplicaciones móviles orientadas al turismo y exploración de ciudades. Entre las más destacadas se encuentran **MAPS.ME** y **Visit a City**, ambas con enfoques diferentes pero complementarios.

#### 1. MAPS.ME

Es una app de mapas offline que permite navegar sin conexión a internet. Su principal ventaja es la posibilidad de descargar mapas completos por región o país, facilitando la orientación del usuario incluso en zonas sin cobertura. Sin embargo, **carezce de personalización**: no filtra lugares según intereses específicos del usuario ni permite integrar datos en tiempo real de negocios actualizados, como los que ofrece Google Places. Además, su sistema de favoritos es limitado y no sincroniza con una API externa ni con otras plataformas.

## 2. Visit a City

Está orientada a planificar itinerarios turísticos, ofreciendo rutas prediseñadas y sugerencias de visitas. Aunque es útil para organizar viajes, **no se adapta al contexto inmediato del usuario**, como su ubicación actual o intereses específicos (por ejemplo, lugares LGBTQ+ friendly, cafeterías culturales, etc.). Tampoco cuenta con integración de rutas dinámicas en bicicleta ni ofrece notificaciones personalizadas al estar cerca de un lugar de interés.

### Conclusión

Estas soluciones actuales no logran cubrir la necesidad de una experiencia personalizada, dinámica y conectada en tiempo real. **Nuestro proyecto busca cubrir ese vacío**: ofrecer una app basada en Google Maps y Places API, con rutas personalizadas, filtros temáticos, sincronización con una API propia, modo sin conexión, y notificaciones inteligentes según la ubicación del usuario. Es una solución más flexible, adaptable y centrada en los intereses reales del viajero moderno.

## 2.2 - Descripción del Sistema Nuevo

El nuevo sistema consiste en una aplicación móvil de turismo personalizada que integra **Google Maps, Places y Directions API**, permitiendo al usuario explorar lugares reales cercanos según sus intereses (cafés, bares LGBTQ+, librerías, etc.).

Entre sus funcionalidades clave se incluyen:

- **Mapa interactivo** con filtros temáticos personalizados.
- **Obtención de lugares reales** desde Google Places API.
- **Marcado y guardado de favoritos**, sincronizados con una API propia desarrollada en Spring Boot.
- **Modo sin conexión**, que permite descargar lugares seleccionados y acceder a ellos sin internet.
- **Rutas personalizadas en bicicleta**, calculadas con Google Directions API.
- **Notificaciones inteligentes**, que avisan cuando el usuario está cerca de un lugar de interés.

El sistema busca ofrecer una experiencia de exploración **dinámica, personalizada y accesible**, mejorando las alternativas existentes mediante conectividad en tiempo real y almacenamiento local inteligente.

## 2.3 - Identificación de Requisitos del Sistema

### 2.3.1.- Requisitos de información

A continuación, se identifican las principales entidades y la información que el sistema deberá almacenar.

---

## **Entidad: Usuario**

- Descripción: Información del usuario que utiliza la app. Necesaria para la autenticación, personalización y sincronización de datos.
- Campos:
  - id (obligatorio) – UUID o Long – Identificador único
  - nombre (obligatorio) – String – Nombre visible
  - email (obligatorio) – String – Identificador único para login
  - contraseña (obligatorio) – String – Encriptada (hash)
  - fecha\_registro (obligatorio) – DateTime – Fecha de creación del usuario
  - preferencias - String
  - fotoPerfilUrl - String
  - rol - String
  - telefono - String
  - pais - String
  - ciudad - String
  - direccion - String
  - descripcion - String
  - verificado - Boolean
  - deletedAt - TimeStamp
- Volumen estimado: Bajo (estimado < 1000 usuarios inicialmente)  
Observaciones: Se puede extender para login social (Google) o roles futuros

---

## **Entidad: Lugar**

- Descripción: Datos de negocios o puntos de interés obtenidos desde Google Places, que se pueden mostrar, guardar y consultar offline.
- Campos:
  - id (obligatorio) – String – ID único (Google Place ID)
  - nombre (obligatorio) – String – Nombre del lugar
  - latitud (obligatorio) – Double – Coordenada GPS
  - longitud (obligatorio) – Double – Coordenada GPS
  - direccion (obligatorio) – String – Dirección completa
  - tipo (obligatorio) – String – Categoría (bar, café, etc.)
  - calificacion (opcional) – Double – Rating promedio
  - foto\_url (opcional) – String – URL a imagen del lugar
  - abierto\_ahora (opcional) – Boolean – Estado actual de apertura
  - usuario Usuario - relacion con usuarios
  - fechaCreacion Timestamp

- Volumen estimado: Medio (~500 lugares por ciudad en promedio)
  - Observaciones: Información cacheada o sincronizada desde Google Places
- 

## Entidad: Ubicación

- Descripción: Ciudades que el usuario ha descargado para explorar sin conexión
  - Campos:
    - id (obligatorio) – Long
    - usuario\_id (obligatorio) – FK – Usuario que descargó
    - nombre String
    - latitud double
    - longitud double
    - tipo String
    - fechaCreación – LocalDateTime
    - usuario Usuario
  - Volumen estimado: medio (1–10 ciudades por usuario)
  - Observaciones: Esta entidad fue creada para poder agrupar lugares a un radio de ubicación de 10 km
- 

## Entidad: Ruta Personalizada

- Descripción: Ruta generada con Google Directions API basada en lugares seleccionados
- Campos:
  - id (obligatorio) – Long
  - usuario\_id (obligatorio) – FK
  - nombre (opcional) – String – Ej: "Ruta Cafetera"
  - origen\_lat, origen\_lng – Double – Punto de partida
  - destino\_lat, destino\_lng – Double – Destino
  - modo\_transporte (obligatorio) – Enum/String – bici, caminando, etc.
  - lugares\_intermedios (opcional) – JSON/List – Stops sugeridos
  - polylineCodificada String
  - categoria String
  - ubicacionId Long
- Volumen estimado: Medio Alto (1-20 rutas por usuario)
- Observaciones: Funcionalidad de mucho provecho que se puede consultar offline

### 2.3.2 Requisitos Funcionales

A continuación, se listan los principales requisitos funcionales:

	Requisito Funcional	Descripción breve
1	Visualizar mapa interactivo	El usuario puede ver su ubicación y lugares en tiempo real.
2	Buscar lugares cercanos	Basado en intereses y ubicación actual usando Google Places.
3	Filtrar lugares por categoría	Mostrar sólo cafés, bares, comida, librerías, restaurantes , etc
4	Guardar lugares nuevos	Guarda los lugares nuevos del usuario.
5	Sincronizar Ubicaciones, Lugares y rutas con backend	Para que se guarden en la nube.
6	Descargar ciudades para uso offline	Guarda lugares para exploración sin internet.
7	Generar rutas personalizadas	Calcula y muestra rutas con Directions API.
8	Registrar y autenticar usuarios	Gestión de acceso con JWT o login de Google.
9	Administrar la base de datos desde backend	CRUD para usuarios, Ubicaciones, Lugares y Rutas.

### 2.3.3 Otros Requisitos

Tipo	Descripción
Técnicos	API desarrollada con Spring Boot y base de datos PostgreSQL. App Android en Kotlin con Room DB y Maps SDK.
Integración	Google Maps, Places y Directions API deben estar correctamente configuradas.
Conectividad	Modo offline para lugares guardados, con sincronización posterior.
Seguridad	Acceso mediante autenticación JWT y OAuth2 con Google login.
Usabilidad	Interfaz adaptada a móviles con diseño accesible y simple.

## 2.4 - Descripción de la Solución

La solución propuesta es una **aplicación móvil multiplataforma centrada en el turismo personalizado**, que se apoya en tecnologías modernas para ofrecer una experiencia inteligente, dinámica y útil para el usuario.

La arquitectura general se compone de:

- **Una app Android (Frontend)** desarrollada en Kotlin con Jetpack Compose, que muestra un mapa interactivo, permite buscar lugares personalizados, generar rutas y guardar favoritos.
- **Una API REST (Backend)** desarrollada en Spring Boot, que gestiona usuarios, sincroniza lugares favoritos y ofrece seguridad con autenticación JWT.
- **Una base de datos PostgreSQL**, encargada de almacenar la información estructurada de usuarios, favoritos y rutas.

El sistema aprovecha servicios externos como **Google Maps, Places y Directions API**, y está preparado para funcionar **sin conexión** mediante una base de datos local Room en Android. Se plantea también su despliegue en la nube para pruebas y acceso remoto.

---

## 2.5 - Planificación del Proyecto

### 2.5.1 - Equipo de Trabajo

El desarrollo del proyecto será llevado a cabo de forma individual por el alumno **Fernando Yucsan Chang Cam**, estudiante del ciclo DAW-DAM.

El entorno de desarrollo incluye:

- **Ordenador personal** con:
  - Sistema operativo: Windows 11
  - Procesador: mínimo AMD Ryzen 7
  - Memoria RAM: 8GB o superior
  - Espacio disponible: mínimo 10TB libres
- **Software:**
  - Android Studio (para app móvil)
  - Spring Tools Eclipse IDE (para backend)
  - PostgreSQL Pgadmin
  - Postman, Git
  - **Servicios online:**
    - Repositorio GitHub
    - Google Cloud Console (API Keys)

- Railway para desplegar backend
- 

### 2.5.2 - Planificación Temporal

El proyecto se desarrollará durante el **tercer trimestre del curso 2024-2025**, con una duración estimada de **8 semanas** (marzo a mayo).

Se divide en los siguientes bloques:

Semana	Objetivo principal
1	Setup de entornos y estructura inicial
2	Integración de mapas y lugares (Google Places)
3	Funcionalidad de favoritos (Room + API)
4	Autenticación de usuarios (JWT/OAuth2)
5	Generación de rutas personalizadas
6	Descarga offline de ciudades
7	Notificaciones + mejoras UI
8	Pruebas finales, documentación y entrega

---

## 2.6 - Estudio del Coste del Proyecto

### Coste de desarrollo (estimado)

Recurso	Coste (€)
Tiempo de desarrollo (160 h x 10 €/h)*	1.600 €
Licencias y herramientas (IDE, APIs)	0 € (todas gratuitas o educativas)
Hosting backend (Render gratuito)	0 €
Base de datos PostgreSQL (Railway)	0 €
API Keys de Google (Places, Maps, etc.)	0-10 € (plan gratuito hasta 200 \$/mes)
Hardware personal	0 € (ya disponible)
<b>Total estimado</b>	<b>1.600 €</b>

### 3. Análisis del Sistema de Información

---

#### 3.1 Identificación del Entorno Tecnológico

##### Entorno de Desarrollo

- **Frontend (App móvil):**
  - Lenguaje: Kotlin
  - Framework: Jetpack Compose
  - IDE: Android Studio
  - Librerías: Google Maps SDK, Room, Retrofit
  - Base de datos local: Room Database (SQLite)
  
- **Backend (API REST):**
  - Lenguaje: Java

- Framework: Spring Boot
- ORM: Spring Data JPA
- Seguridad: Spring Security con JWT y OAuth2
- IDE: IntelliJ IDEA / Visual Studio Code
- Testing: JUnit, Postman
- Documentación: Swagger/OpenAPI

- **Base de Datos (Remota):**

- Tipo: Relacional
- Motor: PostgreSQL (Render o Railway)

- **Herramientas de apoyo:**

- Control de versiones: Git + GitHub
- Despliegue: Render (backend) + Firebase opcional (notificaciones)
- Gestión de APIs: Google Cloud Console

### **Entorno de Explotación (cuando esté en producción)**

- Backend desplegado en Render con PostgreSQL gestionado remotamente.
  - Acceso público a endpoints para la app Android mediante HTTPS.
  - La app se distribuirá localmente (APK) o mediante Play Store si se desea publicar.
  - El sistema se diseñó para soportar crecimiento futuro: múltiples usuarios, distintos roles, nuevos tipos de lugares, y analítica básica.
- 

## **3.2 Modelado de Datos**

El sistema "Mapgenda" utiliza **dos bases de datos diferentes**, cada una adaptada a su propósito: una para el **backend/API en PostgreSQL** y otra para el **almacenamiento local en Room (Android)**. Aunque ambas comparten nombres de tablas y estructuras similares, **no son idénticas ni cumplen la misma función**.

### **Base de datos del backend (PostgreSQL)**

- Es una **base de datos relacional completa**.
- Utiliza **claves foráneas, restricciones y relaciones entre tablas** para garantizar integridad referencial.
- Se encarga de gestionar la información centralizada: usuarios, lugares favoritos, rutas, etc.
- Permite compartir y sincronizar datos entre dispositivos.

### **Base de datos local (Room)**

- Es una **base local** que se utiliza en el dispositivo Android para:

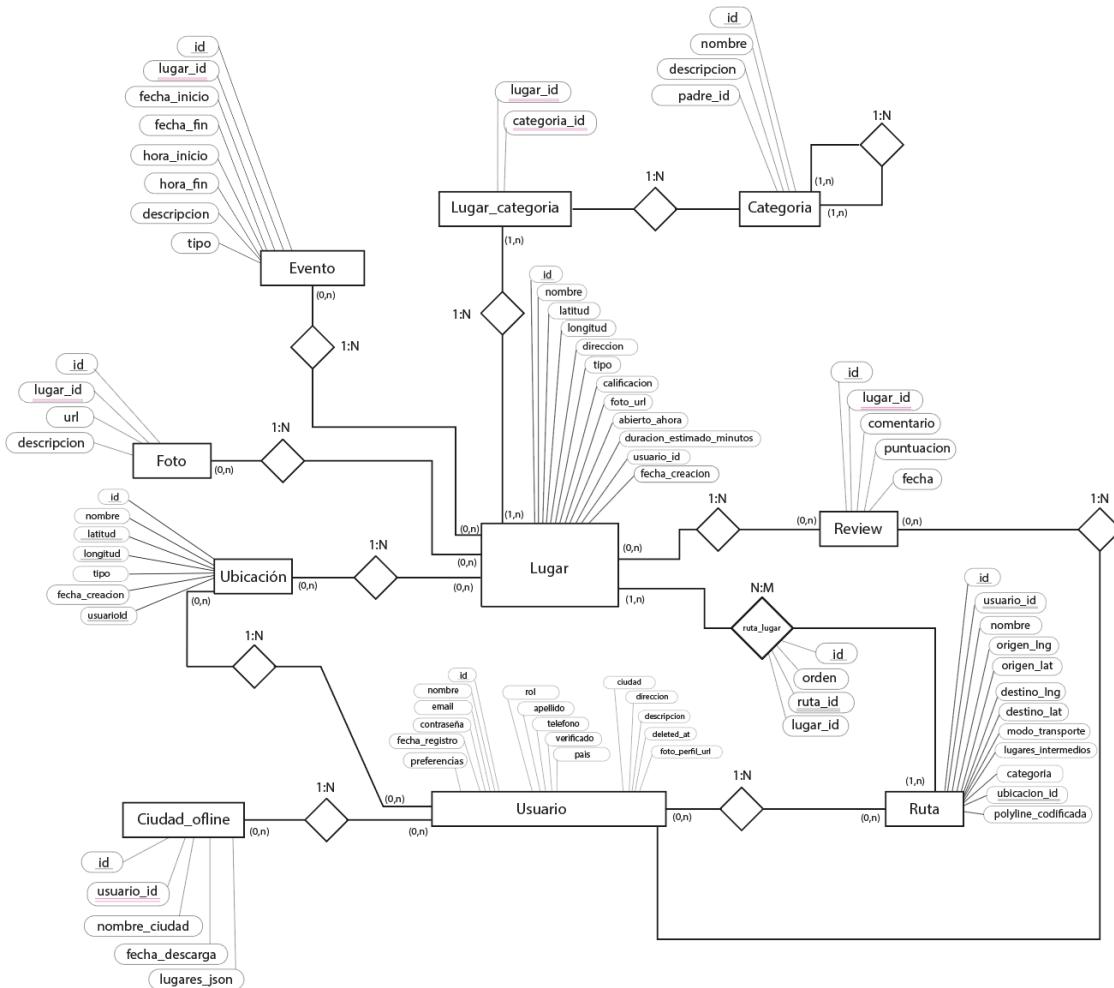
- Acceso rápido a datos recientes o favoritos.
- Funcionamiento **sin conexión** (offline).
- Minimizar llamadas a la API y mejorar rendimiento.
- **No utiliza claves foráneas reales**, ni impone relaciones entre tablas, para simplificar la estructura y hacerla más ligera.
- Se apoya en identificadores (IDs) para simular relaciones cuando es necesario, pero el control lógico lo hace la app.

## ¿Por qué son diferentes?

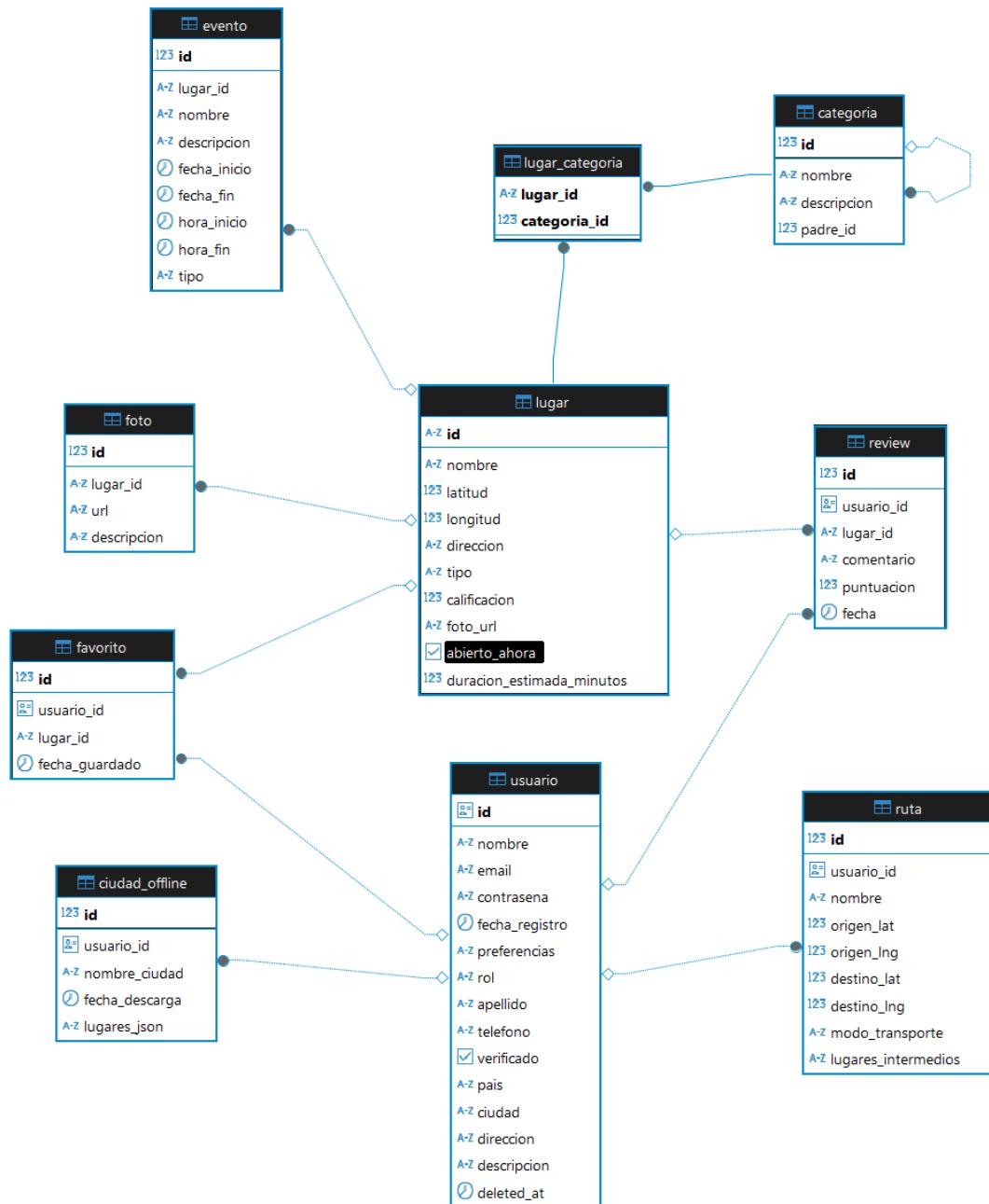
Porque cada base de datos responde a **necesidades distintas**:

- La base de datos del backend está diseñada para **mantener integridad y consistencia global**.
- Room está diseñada para **velocidad, simplicidad y funcionamiento offline**, por eso **evita relaciones estrictas** y es más flexible.

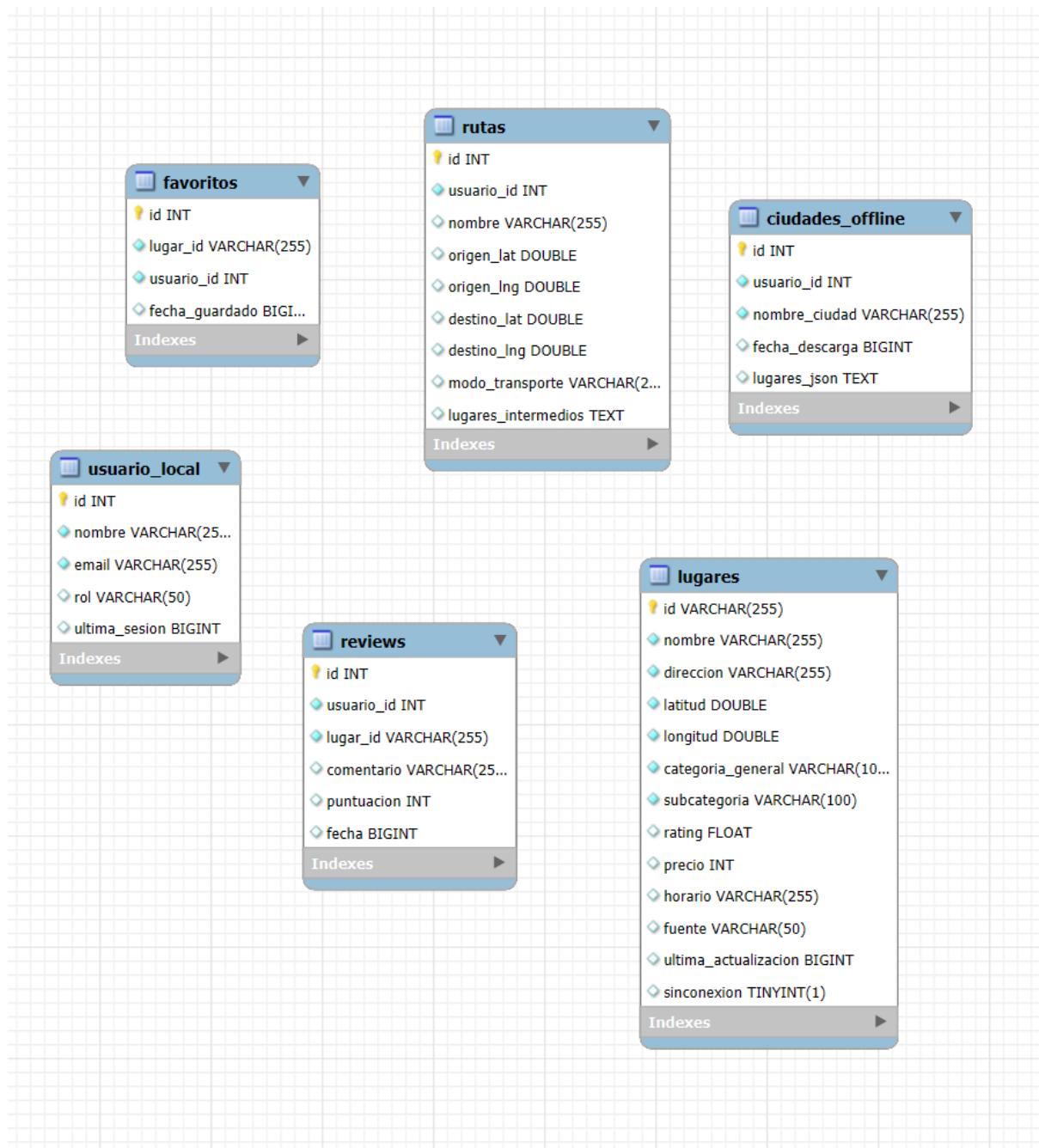
### 3.2.1 Modelo Entidad-Relación



## Base de datos para la API (Postgres SQL) Diagrama de Clases UML



**Base de Datos Room:** Los datos del diagrama han sido exportados a SQL con el objetivo de visualizar las tablas en Workbench. Algunos tipos de datos pueden diferir respecto a los utilizados en Room. El código completo se encuentra en los archivos adjuntos.



### **3.2.2.- Esquema de la base de datos**

Se adjunta junto a la documentación el fichero con nombre ‘mapgenda\_db.sql’ con el script correspondiente para la creación de la base de datos

### **3.1.3.- Datos de prueba**

Se adjunta junto a la documentación el fichero con nombre “data.sql” con el script correspondiente para la inserción de los datos necesarios de la base de datos.

## **3.3.- Identificación de los usuarios participantes y finales**

### **A. Usuarios Finales**

Estos son los usuarios que utilizarán la aplicación para disfrutar del servicio de turismo personalizado. Dentro de esta categoría, se pueden distinguir:

#### **A.1. Usuario Anónimo**

Es aquel que no ha iniciado sesión ni se ha registrado en la aplicación. Sus funciones estarán limitadas a:

- Explorar lugares de interés en el mapa.
- Filtrar lugares según categoría o preferencias generales.
- Visualizar rutas sugeridas para caminatas o recorridos en bicicleta.
- Acceder a información básica de los puntos turísticos.

#### **A.2. Usuario Registrado**

Es un usuario que ha creado una cuenta e iniciado sesión en la aplicación. Tendrá acceso a funcionalidades adicionales:

- Iniciar y cerrar sesión.
- Recuperar contraseña en caso de olvido.
- Editar perfil y preferencias turísticas.
- Guardar lugares como favoritos.
- Descargar datos de lugares guardados para su uso sin conexión.
- Acceder a rutas personalizadas basadas en sus intereses.
- Recibir recomendaciones personalizadas.
- Ver historial de lugares visitados o guardados.

### **B. Usuarios Participantes**

Corresponden a aquellos usuarios que, además de usar la aplicación, tienen permisos especiales para gestionar y mantener la plataforma.

#### **B.1. Usuario Gestor de Contenidos**

Es el encargado de actualizar y curar la información sobre los lugares turísticos y puntos de interés. Sus funciones pueden incluir:

- Iniciar sesión en un panel administrativo.
- Agregar o editar información de lugares.
- Clasificar nuevos puntos de interés según categorías temáticas.
- Revisar y aprobar rutas personalizadas generadas por el sistema.
- Monitorizar estadísticas de uso de los lugares o rutas.

## B.2. Usuario Administrador

Tiene control total sobre la aplicación, incluyendo gestión de usuarios y configuraciones internas. Puede:

- Realizar todas las funciones de los usuarios gestores.
- Administrar permisos y roles de usuarios del sistema.
- Supervisar el estado de las APIs integradas.
- Gestionar el almacenamiento de datos y copias de seguridad.
- Monitorear el rendimiento general de la aplicación.

## C. Sistemas Externos

La aplicación “Mapgenda” se apoya en distintos servicios de Google para brindar funcionalidades avanzadas. Estos sistemas externos interactúan directamente con la aplicación a través de APIs.

### C.1. Google Maps API

- Permite visualizar mapas interactivos en la aplicación.  
Muestra la ubicación del usuario y de los puntos de interés.

### C.2. Google Places API

- Proporciona información detallada sobre lugares (nombre, fotos, puntuaciones, horarios, etc.).
- Permite realizar búsquedas por categoría o cercanía.

### C.3. Google Directions API

- Genera rutas de navegación en tiempo real (caminando o en bicicleta).
- Calcula el tiempo estimado de llegada y distancia entre puntos.

### C.4. Sistema de Almacenamiento Offline

- Permite que los usuarios descarguen los datos de sus favoritos y rutas para acceder a ellos sin conexión.
- Se apoya en la caché del dispositivo o una base de datos local para este propósito.

### C.5 Google Geocoding API

- Convierte direcciones físicas en coordenadas geográficas (latitud y longitud) y viceversa.

- Permite ubicar con precisión direcciones introducidas por el usuario en el mapa y generar marcadores personalizados.

### 3.4.- Identificación de subsistemas de análisis

Dividimos el sistema en partes o subsistemas. Dar una explicación de cada subsistema y mostrar un diagrama de subsistemas en el que se vea cada usuario (u otro sistema) con qué subsistema interactúa.

El sistema general de la aplicación “Mapgenda” se ha descompuesto en cinco subsistemas principales, cada uno de los cuales agrupa un conjunto de funcionalidades relacionadas. Esta división permite estructurar la lógica del proyecto y establecer la interacción entre usuarios, sistemas externos y componentes internos. A continuación, se describen los subsistemas:

#### A. Gestión de Sesiones

Este subsistema se encarga del control de acceso a la aplicación por parte de los usuarios. Incluye las funcionalidades de **inicio de sesión (login)**, **cierre de sesión (logout)** y **gestión de sesiones activas**. Cada vez que un usuario accede a la aplicación, se crea una sesión asociada a su identificador de usuario. Esta sesión permite mantener su estado autenticado y controlar el acceso a funcionalidades restringidas.

Está directamente relacionado con el **subsistema de gestión de usuarios**, ya que:

- Toda sesión debe estar vinculada a un usuario registrado.
- El sistema necesita validar las credenciales del usuario antes de crear la sesión.
- Al cerrar sesión, se elimina la relación entre el usuario y la sesión activa.

#### B. Gestión de Usuarios

Encargado del ciclo de vida de los usuarios registrados en la aplicación.

Funcionalidades:

- Registro de usuarios.
- Modificación de perfil y preferencias turísticas.
- Eliminación de cuentas.
- Gestión de permisos (en el caso de usuarios gestores o administradores).

Interacciones:

- Usuario registrado.
- Usuario gestor.

- Usuario administrador.

Este subsistema se relaciona con todos los demás, ya que las preferencias del usuario impactan la personalización de las rutas y lugares sugeridos.

### C. Descubrimiento y Exploración Turística

Es el núcleo funcional de la aplicación y está orientado a ofrecer la experiencia personalizada de turismo.

Funcionalidades:

- Visualización de lugares de interés.
- Filtros de preferencias y búsqueda.
- Generación de rutas (caminando o en bici).
- Recomendaciones personalizadas.
- Consulta de información de lugares (fotos, reseñas, horarios...).

Interacciones:

- Usuario Google Registrado.
- Google Places API.
- Google Directions API.
- Google Maps API.

Este subsistema se apoya fuertemente en los sistemas externos de Google para alimentar la experiencia de exploración, sin cuenta de google no se puede acceder, esta decisión es porque sin google maps no se pueden disfrutar de la navegación.

### D. Lugares y Uso Offline

Permite a los usuarios guardar lugares y rutas para consultarlos sin conexión.

Funcionalidades:

- Guardar lugares.
- Descargar datos para uso sin conexión.
- Acceder al contenido offline en modo limitado.

Interacciones:

- Usuario registrado.
- Sistema de almacenamiento local.

Se relaciona con el subsistema de descubrimiento, ya que los lugares a guardar provienen de las búsquedas y exploraciones del usuario.

### E. Administración de Contenidos

Permite a los gestores y administradores mantener y enriquecer el contenido de la aplicación.

Funcionalidades:

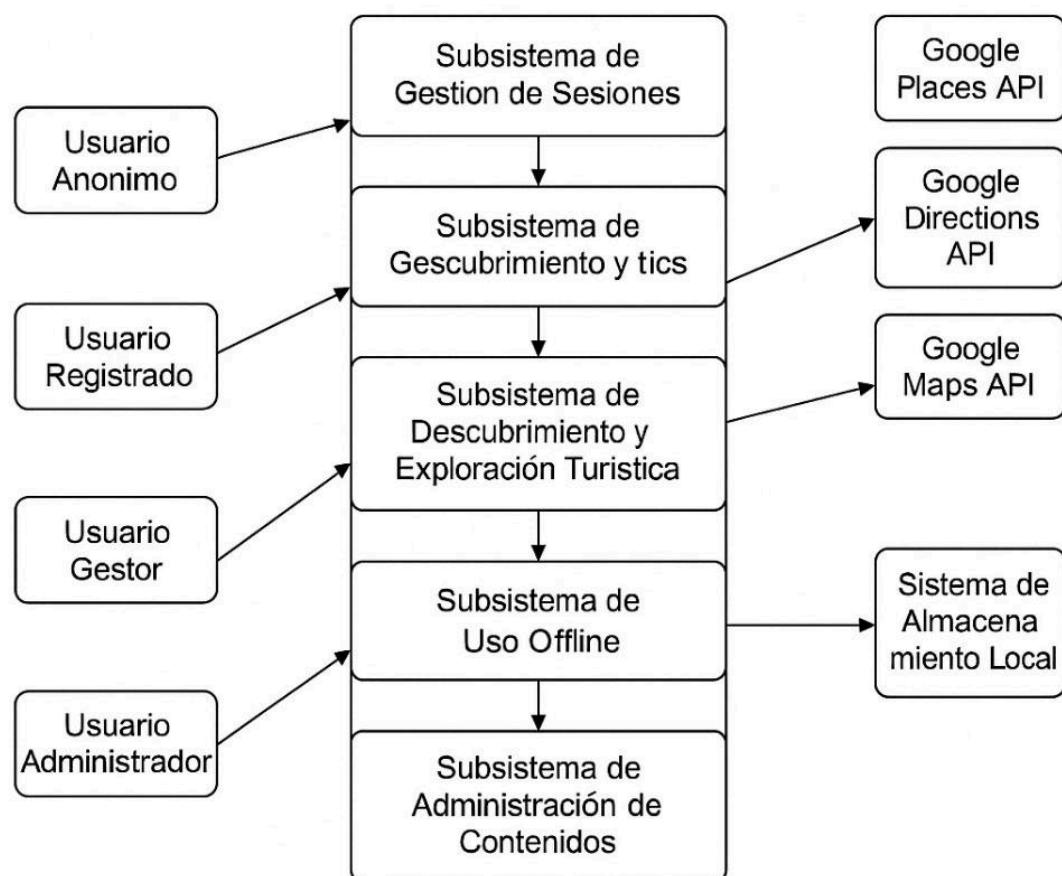
- Alta, edición o eliminación de puntos de interés no disponibles en APIs externas.
- Validación de rutas sugeridas por el sistema.
- Monitorización de estadísticas de uso.

Interacciones:

- Usuario administrador.

Este subsistema permite complementar la información de los lugares con contenido local exclusivo o gestionado manualmente.

### Diagrama de Subsistemas y Usuarios



### **3.5. Establecimiento de Requisitos**

- **Subsistema de Gestión de Sesiones**

- Iniciar sesión: El usuario registrado podrá acceder a su cuenta introduciendo sus credenciales.
- Cerrar sesión: El usuario podrá cerrar su sesión actual para finalizar el uso personalizado de la aplicación.
- Recuperar contraseña: El usuario que haya olvidado su contraseña podrá solicitar un enlace de recuperación a su correo electrónico registrado.

- **Subsistema de Gestión de Usuarios**

- Registro de nuevo usuario: El usuario podrá registrarse creando una cuenta mediante formulario con correo electrónico y contraseña, además de preferencias turísticas.
- Modificación de perfil: El usuario registrado podrá editar sus datos personales (nombre, correo electrónico) y modificar sus intereses turísticos para afinar las recomendaciones.
- Eliminación de cuenta: El usuario registrado podrá solicitar la eliminación definitiva de su cuenta desde la aplicación.

- **Subsistema de Descubrimiento y Exploración Turística**

- Visualización de lugares cercanos: El usuario podrá ver en un mapa los puntos de interés cercanos, según su ubicación y preferencias configuradas.
- Filtros de búsqueda: El usuario podrá aplicar filtros por categoría (naturaleza, cultura, gastronomía, etc.) y nivel de interés.
- Generación de rutas personalizadas: La aplicación mostrará rutas optimizadas para caminar o ir en bicicleta, basadas en los intereses y ubicación actual del usuario.
- Consulta de información detallada: El usuario podrá acceder a descripciones, imágenes, horarios y reseñas de cada punto turístico, gracias a la integración con Google Places.

- **Subsistema de Uso Offline**

- Descargar datos para uso sin conexión: El usuario podrá descargar la información de los lugares favoritos y rutas personalizadas para poder acceder a ellos cuando no tenga conexión a internet.

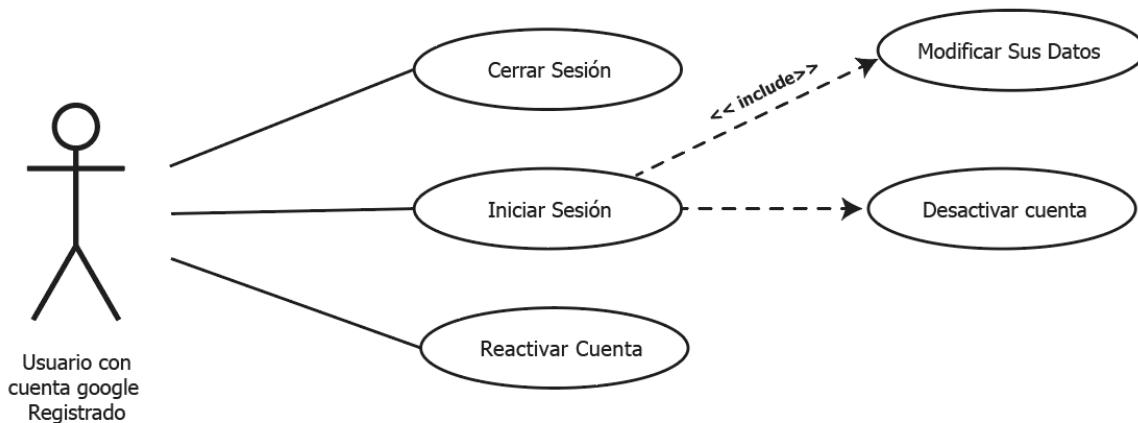
- Acceso a contenido offline: El usuario podrá explorar los datos guardados (lugares y rutas) sin necesidad de estar conectado a internet.

- **Subsistema de Administración de Contenidos**

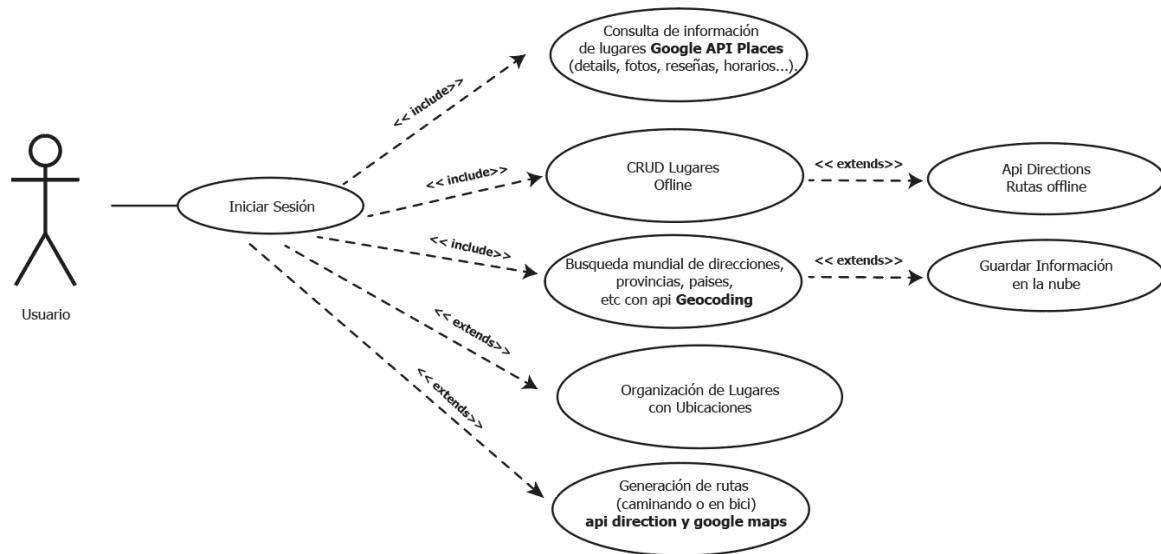
- Alta de nuevos puntos de interés personalizados: El gestor podrá crear lugares turísticos propios que no existan en las APIs externas (por ejemplo, lugares locales exclusivos o temporales).
- Edición de información de lugares: El gestor podrá modificar descripciones, imágenes y categorías de puntos turísticos existentes.
- Gestión de rutas sugeridas: El administrador podrá validar, modificar o eliminar rutas turísticas creadas automáticamente por el sistema o por los usuarios.
- Consulta de estadísticas de uso: El administrador podrá consultar métricas como lugares más visitados, favoritos más comunes o rutas más utilizadas.

### 3.6.- Diagramas de Análisis

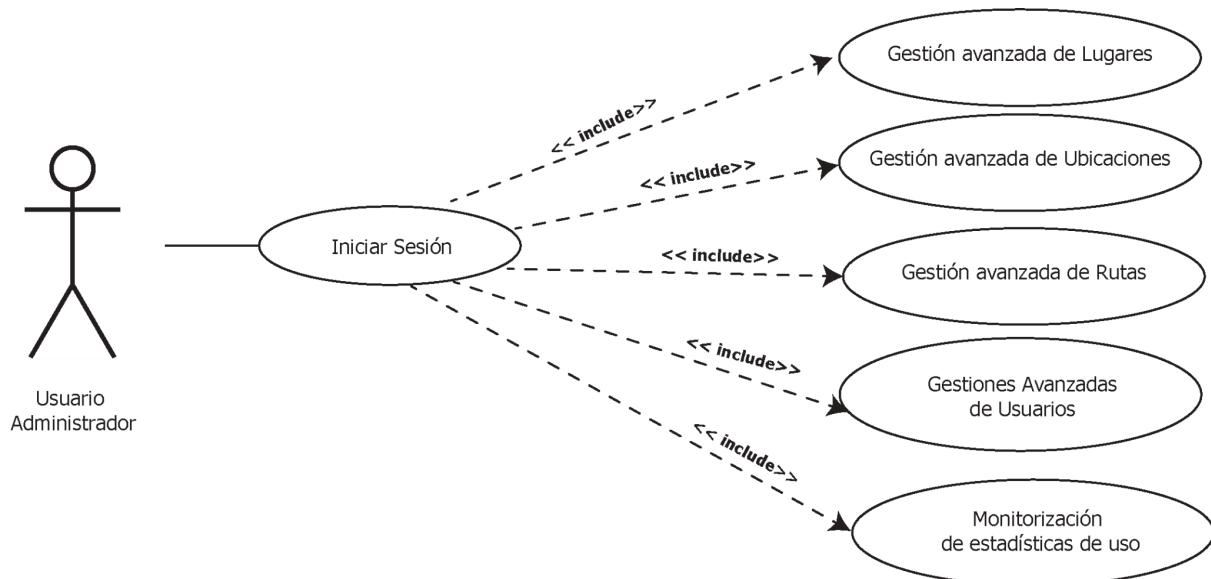
**Diagrama Caso de uso Gestión de Sesiones**



## Diagrama Caso de uso Descubrimiento y Exploración Turística



## Diagrama Caso de uso Administración de Contenidos



## **3.7.- Definición de interfaces de usuario**

### **3.7.1. Especificación de Principios Generales de Interfaz**

La interfaz de usuario (UI) de la aplicación Mapgenda se diseñará siguiendo los siguientes principios generales, con el objetivo de ofrecer una experiencia intuitiva, accesible y placentera al usuario:

#### **1. Simplicidad y claridad**

La interfaz deberá ser simple, con un diseño limpio que minimice la carga cognitiva. Los elementos visuales estarán organizados jerárquicamente para destacar las funciones principales, como la búsqueda de lugares, el acceso a favoritos y la navegación.

#### **2. Consistencia visual y funcional**

Se mantendrá una coherencia en los estilos visuales (colores, tipografías, íconos) y patrones de interacción a lo largo de toda la aplicación. Esto facilitará el aprendizaje del sistema y aumentará la confianza del usuario.

#### **3. Facilidad de navegación**

La navegación será fluida e intuitiva, basada en menús accesibles y rutas claras. Se utilizarán iconos reconocibles y etiquetas descriptivas para guiar al usuario a través de las distintas funcionalidades.

#### **4. Retroalimentación inmediata**

La aplicación proporcionará respuestas visuales o auditivas inmediatas ante cada acción del usuario (por ejemplo, al guardar un favorito o calcular una ruta), asegurando que sus interacciones han sido reconocidas y procesadas correctamente.

#### **5. Accesibilidad**

La interfaz cumplirá con los principios de accesibilidad universal, incluyendo contraste adecuado de colores, textos escalables, y compatibilidad con tecnologías de asistencia como lectores de pantalla.

#### **6. Interactividad contextual**

Se mostrará la información relevante en función del contexto y la localización actual del usuario, evitando sobrecargar la pantalla con opciones innecesarias.

#### **7. Modo sin conexión**

Se garantizará una transición clara entre los modos en línea y sin conexión, con mensajes que indiquen la disponibilidad de funcionalidades offline y acceso directo a los contenidos previamente descargados.

#### **8. Optimización para dispositivos móviles**

Dado que se trata de una aplicación móvil, todos los elementos estarán optimizados

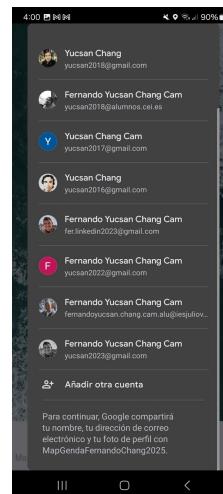
para pantallas táctiles y de distintos tamaños, con tiempos de carga reducidos y diseño responsive.

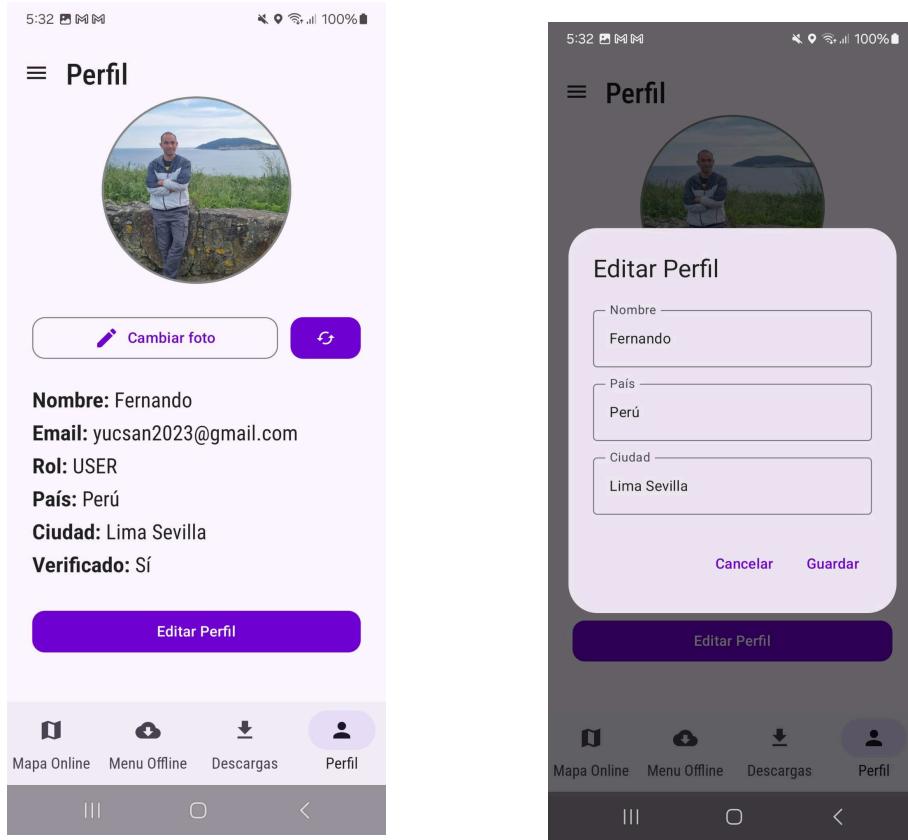
### 3.7.2.- Especificación de formatos individuales de la interfaz de pantalla



#### Pantalla de Inicio

El Login será Exclusivo de para usuarios con cuenta de Google para asegurarnos el poder utilizar todas las funciones de la aplicación





## Pantalla de Perfil

La **pantalla de perfil** permite a los usuarios gestionar opciones personalizadas relacionadas con la experiencia de uso y la descarga de datos para navegación sin conexión. Esta sección solo está disponible para usuarios autenticados mediante cuenta de Google, lo cual garantiza el acceso seguro y completo a todas las funcionalidades de la aplicación.

### Funcionalidades disponibles:

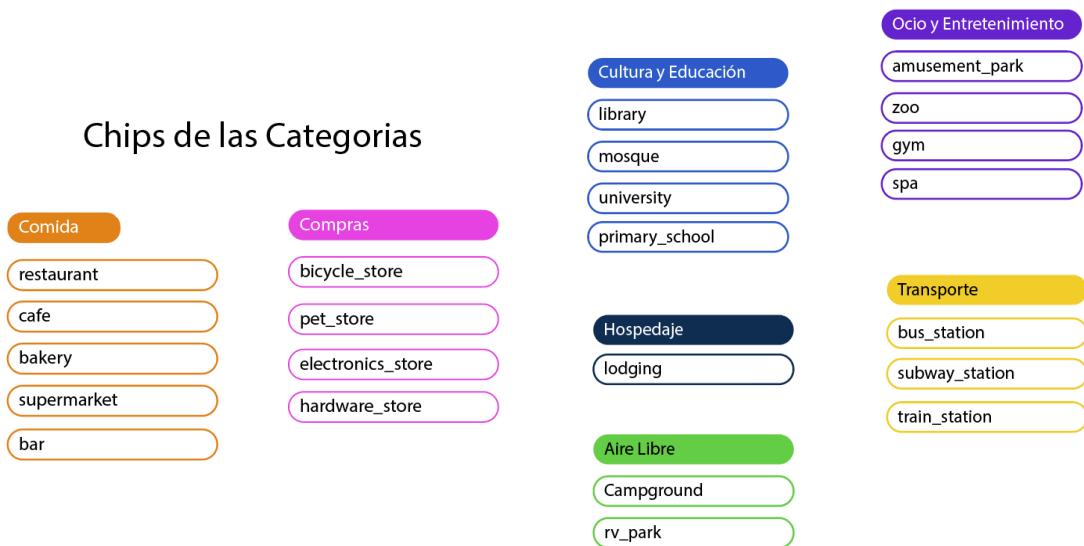
- Gestión de cuenta:
  - Desactivar Cuenta (drawer)
  - Cerrar Sesión (drawer).
  - Visualización del nombre del usuario autenticado.
  - Edición de datos Usuario datos y foto
  
- Descarga personalizada de lugares:
  - El usuario puede realizar hasta tres (3) descargas personalizadas por día.
  - Cada descarga permite seleccionar hasta 17 subcategorías pertenecientes a un conjunto de categorías predeterminadas, tales como:
    - Comida
    - Compras
    - Aire Libre

- Cultura
- Ocio
- Hospedaje
- Transporte
- Custom (categorías personalizadas por el usuario)

- Proceso de descarga:

- El usuario selecciona las categorías deseadas y luego inicia la descarga de los datos correspondientes para su uso sin conexión.
- Una vez seleccionadas, las categorías se procesan y almacenan localmente para su posterior consulta, sin necesidad de conexión a Internet.

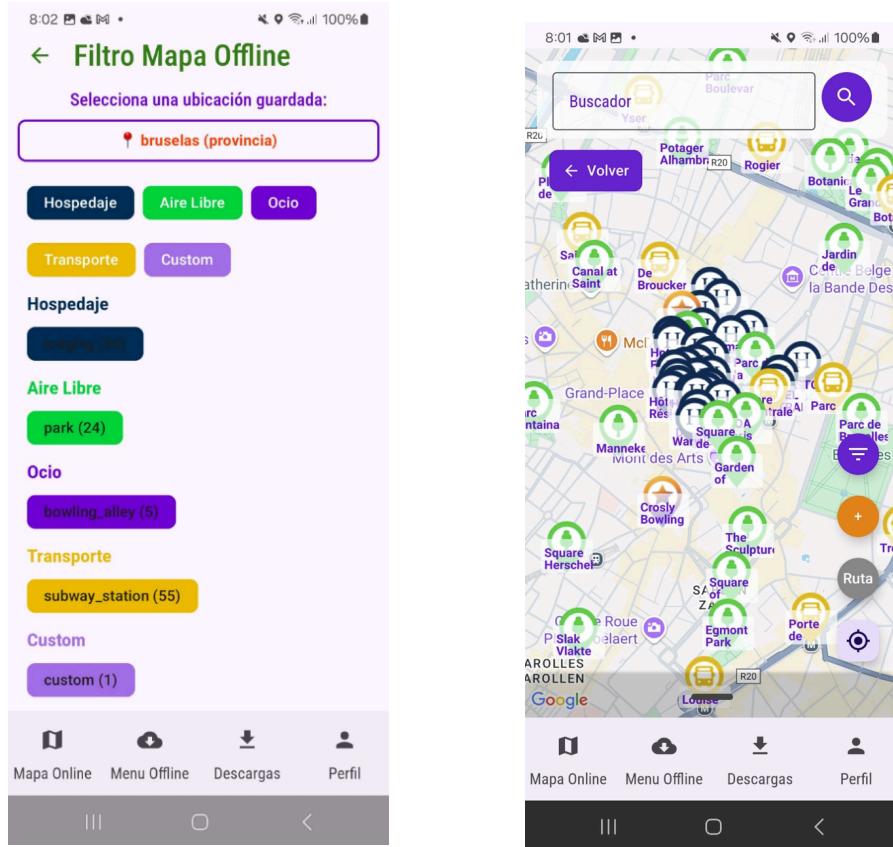
Este módulo de perfil actúa como el **centro de control de preferencias de descarga**, permitiendo que la experiencia de exploración turística esté alineada con los intereses personales del usuario, incluso en entornos con conectividad limitada.



### Sistema de Chips de Categorías

El sistema de **chips de categorías** en Mapgenda está diseñado para facilitar la selección rápida y visualmente intuitiva de intereses turísticos. Cada **categoría principal** se representa con un chip de color sólido, mientras que sus **subcategorías** asociadas comparten el mismo color en el borde, creando una relación visual clara y coherente.

Este uso de color no solo mejora la legibilidad y organización, sino que también guía al usuario en la navegación, reduciendo la carga cognitiva. Los colores fueron elegidos estratégicamente para evocar asociaciones temáticas (por ejemplo, naranja para comida, azul para educación, verde para naturaleza).



Además, la disposición modular de los chips permite escalabilidad y mantiene la interfaz limpia, adaptable a futuras expansiones de contenido.

## Pantallas Principales: Filtro y Mapa

La interfaz principal de Mapgenda se articula entre dos vistas fundamentales: **la pantalla de filtros** (izquierda) y **la pantalla de mapa** (derecha). Ambas funcionan de manera complementaria para permitir una exploración eficiente, personal y visualmente intuitiva del entorno turístico.

### Pantalla de Filtros (Selector de Categorías)

Esta pantalla permite al usuario configurar en detalle **qué tipos de lugares desea explorar** según sus intereses. Está diseñada para ofrecer control total sobre la búsqueda mediante un sistema de activación de **categorías y subcategorías**.

- Las **categorías principales** (como Comida, Cultura, Aire Libre, etc.) se muestran en la parte superior mediante chips de color, y al seleccionarlas se despliegan sus **subcategorías** correspondientes.

- Cada subcategoría puede activarse individualmente para realizar búsquedas precisas.
- El botón de “**Buscar Lugares**” lanza una búsqueda contextual basada en los filtros seleccionados y la ubicación actual del usuario.

Además, se incluye un selector llamado “**minimenú**”, que define qué filtros rápidos estarán disponibles dentro del **bottom sheet (mini filtro)** integrado en la pantalla de mapa. Esta funcionalidad ofrece un acceso ágil a los filtros más usados, sin necesidad de abrir el panel completo.

---

## Pantalla de Mapa

Esta vista representa el centro de interacción geográfica de la aplicación, construida sobre **Google Maps** para brindar precisión y familiaridad. Aquí el usuario puede visualizar los lugares sugeridos, sus favoritos y realizar acciones en tiempo real.

- En la parte inferior, se encuentra un **bottom sheet (scaffold)** colapsado, que actúa como un **minifiltro dinámico**, mostrando las categorías previamente seleccionadas como importantes. Esto permite aplicar y ajustar filtros sin abandonar la vista del mapa.
- Los **íconos sobre el mapa** están categorizados visualmente por tipo de lugar, facilitando la navegación intuitiva.

## Botones flotantes (FABs)

Sobre la interfaz del mapa se ubican tres **botones flotantes principales**, dispuestos verticalmente para un acceso rápido:

### 1. Botón de Filtros

- Abre la pantalla completa de filtros (pantalla izquierda) para realizar ajustes o nuevas búsquedas.

### 2. Botón de Agregar Lugar

- Permite que el usuario cree un nuevo punto de interés manualmente haciendo tap sobre el mapa. Ideal para agregar sitios que aún no aparecen en las fuentes oficiales.

### 3. Botón de Ruta Rápida

- Genera una ruta instantánea hacia un punto seleccionado, enviando un intent directamente a **Google Maps**. Esta acción facilita la navegación espontánea, sin salir de la app.

Con este diseño, Mapgenda logra un equilibrio entre potencia funcional y simplicidad de uso, ideal para usuarios que desean personalizar su experiencia turística al máximo con solo unos toques.



### Sistema de Íconos de Categorías

El sistema de **íconos geolocalizados** de Mapgenda ha sido diseñado con un enfoque **minimalista, directo y visualmente atractivo**, pensado para facilitar la identificación instantánea de los tipos de lugares en el mapa.

Cada **categoría principal** se representa mediante un pin de color distintivo, con un **ícono central simple y reconocible** que comunica su función sin necesidad de texto. Este enfoque mejora la legibilidad en entornos móviles y ofrece una experiencia clara incluso en mapas densamente poblados.

Características destacadas:

- **Claridad visual:** Los íconos están diseñados con formas limpias y siluetas familiares para que el usuario los comprenda de inmediato (por ejemplo, cubiertos para comida, cámara para naturaleza, copa para bares).
- **Color como guía temática:** El color de cada pin coincide con el de su categoría en los chips de filtro, reforzando la coherencia visual en toda la app.
- **Estilo amigable y moderno:** Las líneas curvas y proporciones equilibradas dan un aspecto accesible y profesional, sin caer en lo infantil ni en lo abstracto.

Categorías representadas:

- **Comida** (naranja): tenedor y taza de café.

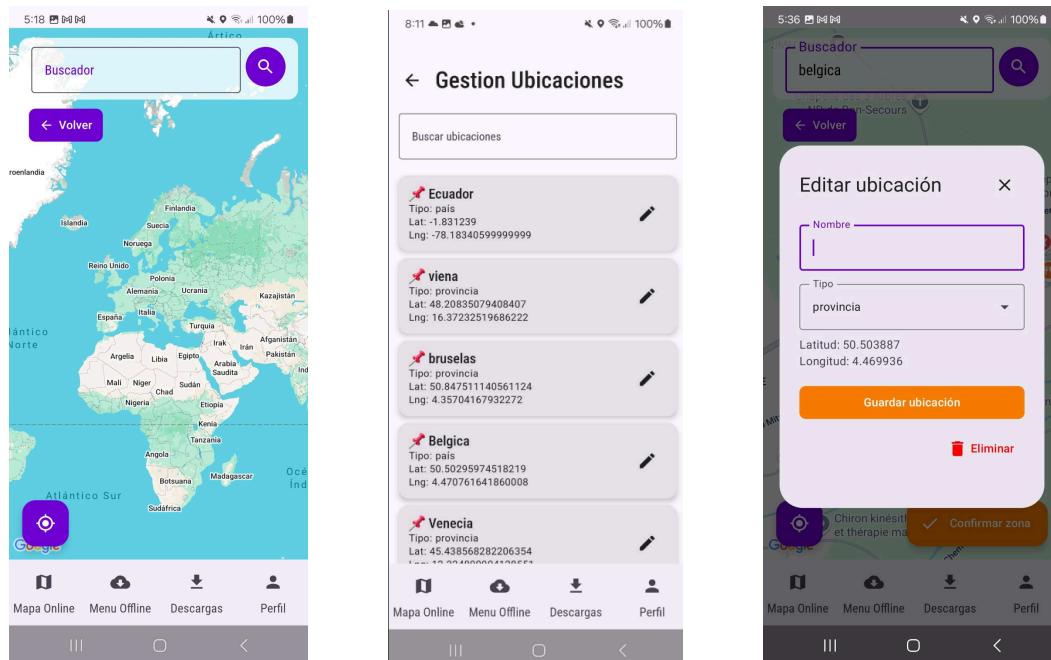
- **Compras** (fucsia): bolsa de compras.
- **Cultura y Educación** (azul): columna clásica, pincel, pez (acuario) y edificio escolar.
- **Ocio y Entretenimiento** (violeta): película, cóctel y naípe.
- **Hospedaje** (azul oscuro): símbolo de hotel.
- **Transporte** (amarillo): autobús y tren.
- **Aire Libre** (verde): árbol y cámara (fotografía de naturaleza).

Este sistema de íconos refuerza la **navegación intuitiva** en el mapa, ofreciendo una experiencia más rica, eficiente y visualmente armoniosa para el usuario.



### Iconos personalizados

Para lugares que se vayan reconociendo con el tiempo y se almacenen en la base de datos y sean parte de las primeras descargas ya con el tiempo.

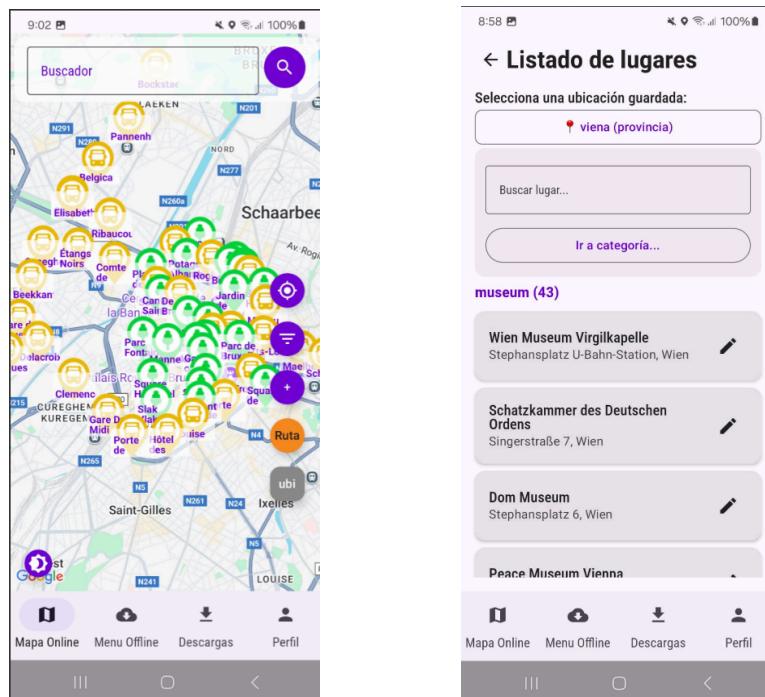


## Gestión de Ubicaciones: Exploración, Información y Control Personalizado

El sistema de gestión de ubicaciones en Mapgenda permite, mediante el buscador, insertar una ubicación válida que redirige automáticamente al lugar seleccionado. Técnicamente, la ubicación ya está marcada, pero es necesario tocar la pantalla para indicar el punto exacto. Al hacerlo, se abre una alerta que permite almacenar la ubicación.

Esta pantalla puede abrirse desde tres lugares: desde el propio **mapaCompose**, desde el menú **Offline** o desde la pantalla de **Descargas**. Las ubicaciones funcionan como puntos de referencia desde los cuales se pueden buscar otros lugares o guardar ubicaciones dentro de un radio de 10 km.

En la segunda pantalla es posible buscar y editar las ubicaciones guardadas.



## Gestión de Lugares: Exploración, Información y Control Personalizado

El sistema de gestión de lugares en Mapgenda ha sido diseñado para que el usuario no solo explore su entorno con libertad, sino que también tenga el **control total** sobre la información que visualiza, guarda y personaliza. Las siguientes pantallas ilustran cómo se integra esta funcionalidad clave dentro de la experiencia de usuario.

### 1. Resultados de Búsqueda en el Mapa

La primera imagen muestra una búsqueda combinada de **parques** (íconos verdes) y **cafeterías** (íconos naranjas) activadas mediante filtros. Esta vista geolocalizada ofrece una respuesta visual instantánea a los intereses seleccionados por el usuario.

- Cada lugar está representado por un **pin con iconografía temática**, facilitando la lectura rápida del mapa.
- Desde aquí, el usuario puede interactuar directamente con los lugares disponibles mediante taps, o usar los **botones flotantes** para ajustar filtros, agregar lugares

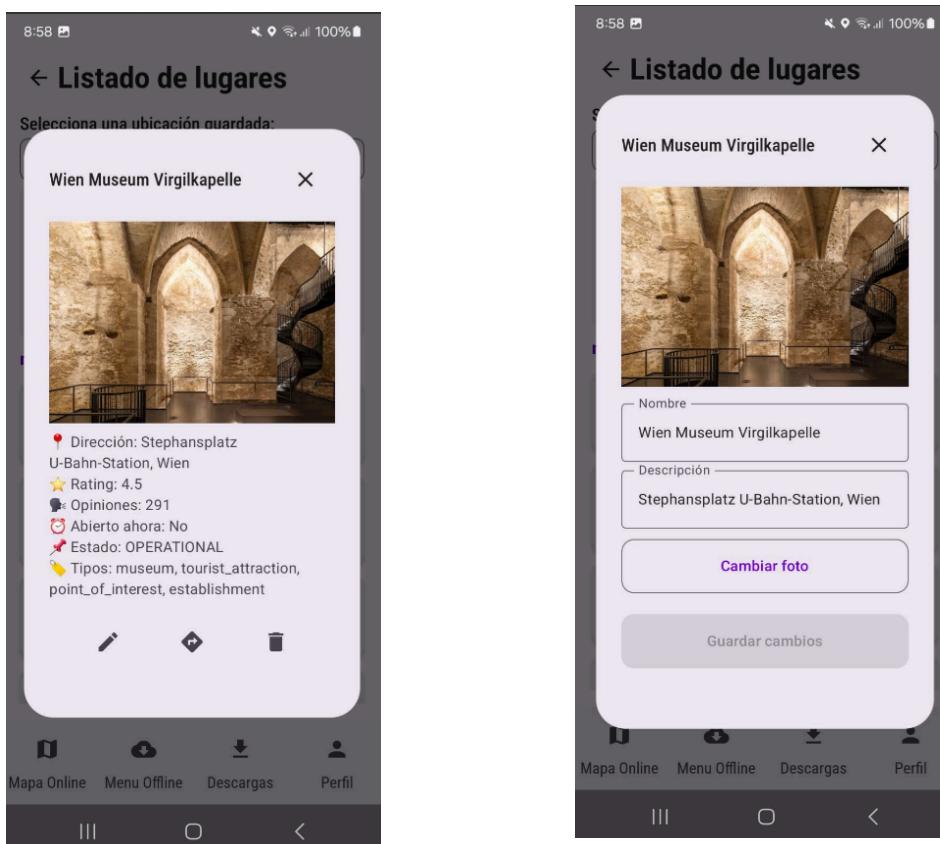
personalizados o generar rutas.

## 2. Listado de Lugares Offline

La tercera pantalla corresponde a la sección **Offline**, donde se almacenan los lugares descargados o guardados como favoritos.

- Cada entrada muestra el **nombre del lugar, rating, comentario personal editable y acceso a filtros** para clasificar u ordenar.
- Es una vista de referencia rápida, útil tanto en modo sin conexión como para planificación previa.

Este enfoque centrado en el usuario hace que Mapgenda no sea solo una app para encontrar sitios, sino una **plataforma de gestión de experiencias urbanas**, personalizada, conectada y lista para usarse tanto online como offline.



## 3. Ficha Detallada del Lugar (Alerta Dialog)

Al seleccionar un lugar del mapa, se despliega una ventana emergente tipo **alert dialog**, que presenta de forma concisa y visualmente clara toda la información descargada desde la **Google Places API**.

- **Imagen del lugar** (cuando esté disponible), nombre y dirección precisa.

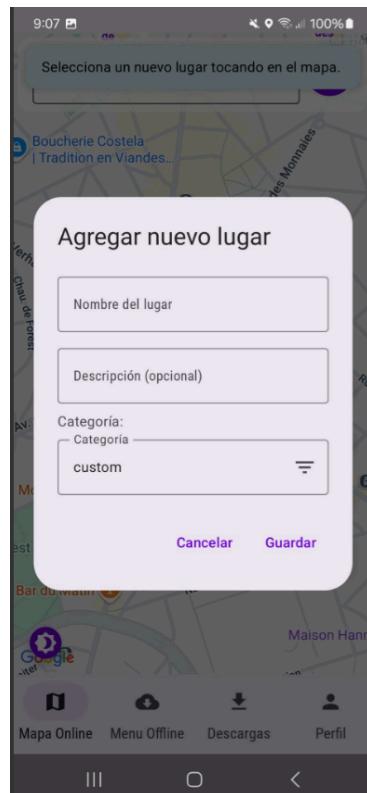
- Información clave como **rating, cantidad de opiniones, estado operativo y horario actual**.
- Lista de tipos de lugar asociados para mayor contexto semántico.

Además, se incluyen tres acciones clave:

- **Editar:** Permite al usuario modificar información o agregar anotaciones personales (ideal para lugares personalizados).
- **Ir con Google Maps:** Lanza un intent externo para navegación rápida y detallada.
- **Eliminar:** Opción para quitar el lugar de favoritos o de los resultados locales.

Esta ficha rápida convierte cada lugar en un **nodo interactivo enriquecido**, optimizando la toma de decisiones en movilidad.

---



## Creación y Edición de Lugares Personalizados

La funcionalidad de creación y edición de lugares en Mapgenda permite a los usuarios **personalizar su mapa turístico**, agregando puntos de interés propios que no figuren en bases de datos públicas o modificando lugares descargados para adaptarlos a su experiencia.

## 1. Activación: Botón Flotante “+”

Desde la **pantalla principal del mapa**, el botón flotante “+” permite al usuario **crear un nuevo lugar** tocando cualquier punto del mapa. Esta interacción abre un **alert dialog** (pantalla 2) que actúa como formulario de ingreso rápido de información.

## 2. Creación de un Lugar (Formulario Inicial)

La segunda pantalla muestra el formulario de creación, que contiene:

- **Campo de imagen:** Permite al usuario asignar una fotografía al lugar desde su galería o cámara, ofreciendo una referencia visual inmediata.
- **Nombre del lugar:** Campo editable para ingresar el título deseado.
- **Categoría:** Menú desplegable para asignar la categoría principal del lugar (Comida, Compras, etc.).
- **Descripción:** Texto libre donde el usuario puede incluir observaciones, comentarios o detalles relevantes.

El diseño limpio y centrado facilita la creación rápida sin distracciones. El botón “Aceptar” guarda el lugar y lo añade al mapa; “Cancelar” descarta los cambios.

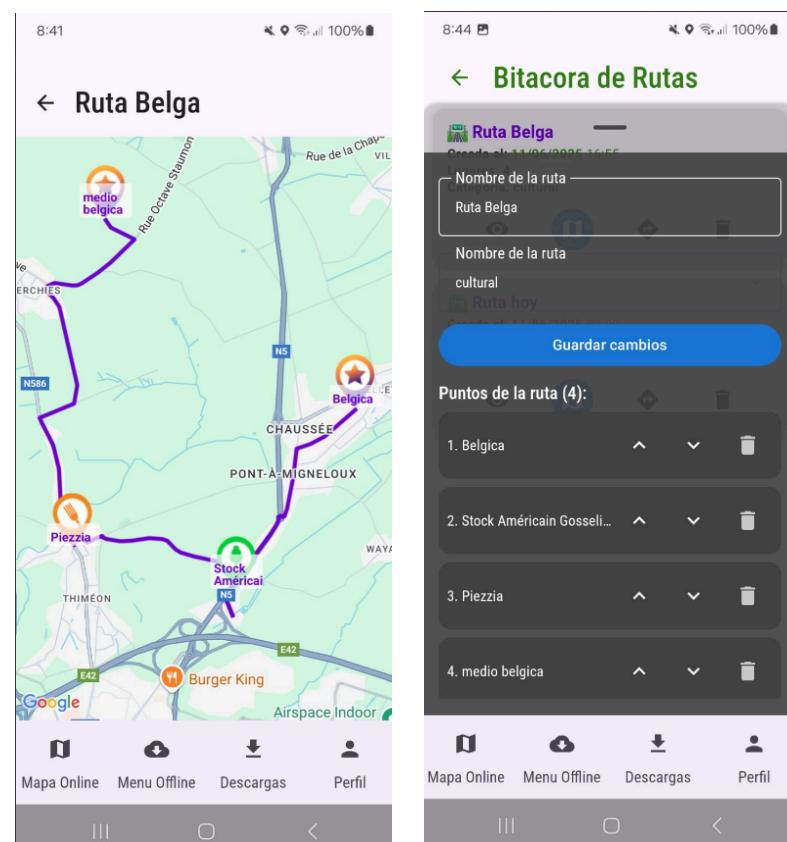
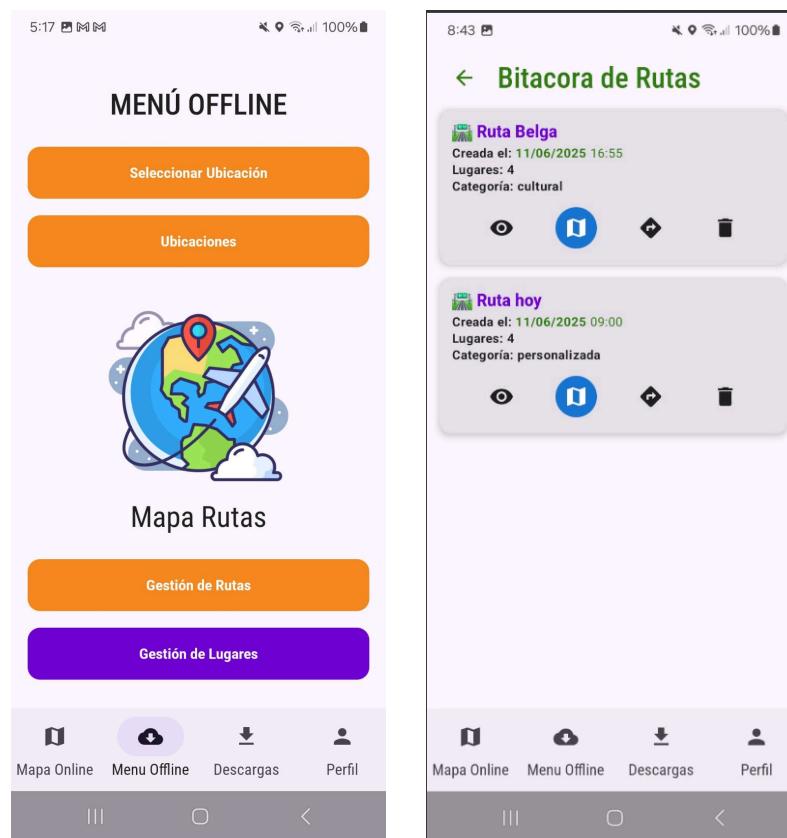
## 3. Edición de un Lugar Existente

La tercera pantalla muestra el mismo formulario pero en modo **edición**, con los datos ya cargados del lugar previamente creado o descargado. Aquí el usuario puede:

- Reemplazar la foto.
- Modificar nombre, categoría o descripción.
- Personalizar cualquier aspecto visual o funcional del punto guardado.

Esta función convierte a Mappenda en una **herramienta flexible**, donde el usuario no se limita a consumir datos, sino que **construye y adapta su propia experiencia geográfica**, generando valor tanto online como offline.

# Funcionalidad de Rutas Offline



La función de **Rutas Offline** permite a

los usuarios planificar, guardar y seguir recorridos turísticos personalizados, incluso en ausencia de conexión a internet. Esta herramienta está pensada para quienes desean explorar sin depender de datos móviles, ofreciendo una experiencia más autónoma y robusta.

---

## 1. Pantalla Menú de Rutas

En la primera pantalla (izquierda, arriba), el usuario accede a un **menú donde puede**

- **Crear Ubicaciones**
- **Gestionar Ubicaciones**
  - Se puede editar el nombre o borrar una ubicación
- **Crear rutas**
  - Se pueden crear rutas entre los lugares y editarlas
- **Gestionar Rutas**
  - Edición o revisión de las rutas
- **Gestionar Lugares**
  - Gestión de Lugares

Hay que aclarar que para la creación de rutas offline si es necesario tener conexión a internet ya para acceder a ésta y verla si está guardada no es necesario.

---

## 2. Edición de Ruta

La segunda pantalla muestra la **interfaz de edición**, donde el usuario puede:

- Reordenar puntos de interés (waypoints).
- Eliminar paradas individuales.
- Guardar o borrar la ruta completa.

Los botones “**Guardar Ruta**” y “**Borrar Ruta**” permiten finalizar la configuración de manera clara y directa.

---

## 3. Visualización en el Mapa

Una vez creada, la ruta puede visualizarse sobre el mapa (pantalla inferior izquierda). Los puntos están conectados por líneas, mostrando el recorrido previsto. Un botón verde flotante permite **iniciar la navegación offline**.

---

#### 4. Edición de Rutas Guardadas

La última pantalla permite acceder a todas las rutas almacenadas por el usuario. Allí se pueden:

- Buscar rutas por nombre.
- Editar recorridos existentes.
- Visualizar detalles rápidamente.

---

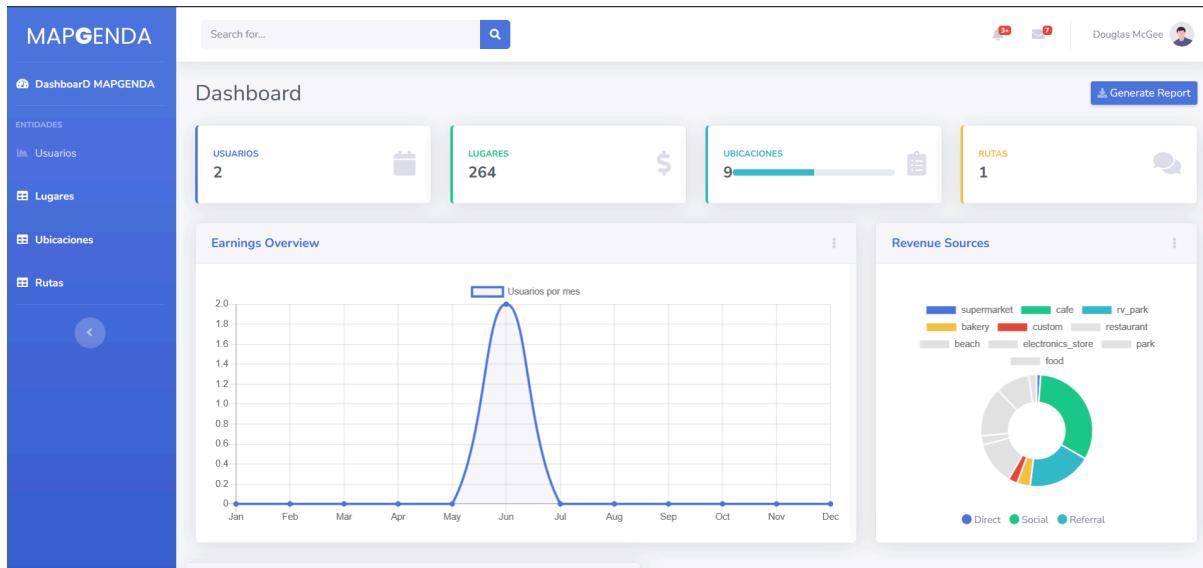
#### Aclaración Importante

Aunque el flujo general de esta funcionalidad es **intuitivo y guiado**, se incluirá una sección específica en la documentación para explicar **cómo descargar los mapas de Google Maps para uso offline**, paso a paso. Esto asegura que incluso en áreas sin cobertura, el usuario pueda contar con navegación y orientación precisa.

#### Panel de Administración (Dashboard)

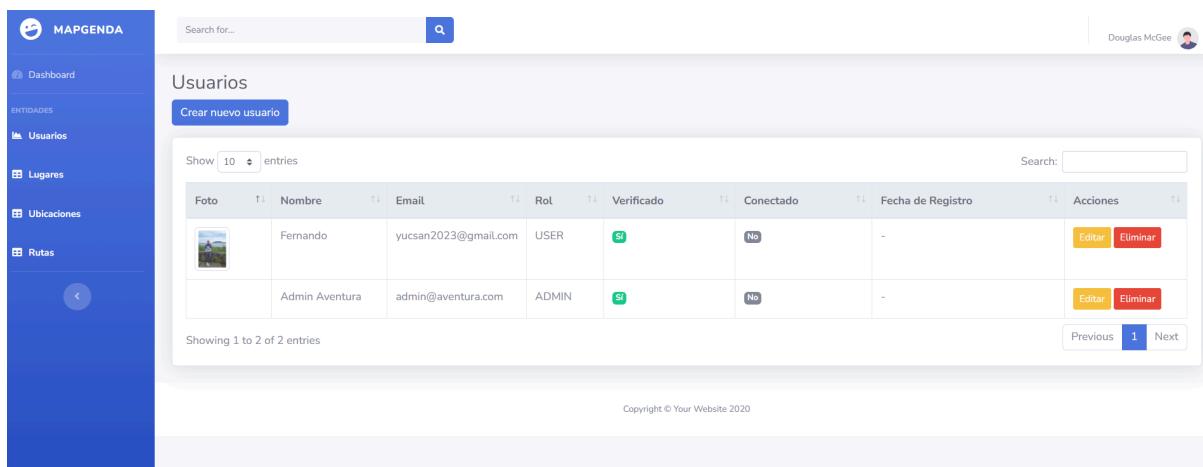


Login



Se desarrollará un **panel de administración web (dashboard)** utilizando el motor de plantillas **Thymeleaf** integrado con el framework **Spring Boot**. Este dashboard permitirá a los usuarios con perfil **administrador** visualizar, gestionar y monitorear los principales datos de la aplicación Mapgenda.

### Características principales:



- **Control de usuarios registrados:**

Visualización detallada de los usuarios activos, incluyendo nombre, rol, actividad, fechas de ingreso y más. Se integrará con una tabla dinámica (como *DataTables*) que permita orden, búsqueda y paginación.

Foto	Nombre	Dirección	Tipo	Calificación	Abierto	Duración	Coordenadas	Acciones
	5 To Go	Strada Gheorghe Baritiu 1A, Brașov	cafe	4.3	<span>Si</span>	0 min	45.6415944, 25.5884242	<button>Editar</button> <button>Eliminar</button>
	ElS[kalt] - 100% natürliches Eis	Piața Sfatului 3, Brașov	cafe	4.7	<span>Si</span>	0 min	45.6427329, 25.589361	<button>Editar</button> <button>Eliminar</button>
	Starbucks	Piața Sfatului Nr. 18, Brașov	cafe	4.2	<span>Si</span>	0 min	45.6415445, 25.5886361	<button>Editar</button> <button>Eliminar</button>
	Crazy Bubble	Strada Apollonia Hirscher 2, Brașov	cafe	3.9	<span>No</span>	0 min	45.6413907, 25.5891929	<button>Editar</button> <button>Eliminar</button>
	MOA	Strada Mureșenilor 3, Brașov	cafe	4.8	<span>Si</span>	0 min	45.6431966, 25.5887612	<button>Editar</button> <button>Eliminar</button>
	La Vatra Ardealului	Strada Gheorghe Baritiu 14, Brașov	cafe	4.5	<span>Si</span>	0 min	45.64134309999999, 25.5874398	<button>Editar</button> <button>Eliminar</button>

- **Gestión de lugares:**

Acceso a la base de datos de lugares guardados, creados o modificados por los usuarios. El administrador podrá editar, eliminar o revisar su información y trazabilidad.

Nombre	Categoría	Modo	Origen	Destino	Acciones
Ruta hoy	personalizada	driving	45.6465, 25.5951	45.6440, 25.5971	<button>Eliminar</button>

- **Supervisión de rutas offline:**

Posibilidad de revisar rutas creadas por los usuarios, cantidad de puntos por ruta, y acciones asociadas a su edición o uso.

- **Acceso restringido por roles:**

Este panel estará protegido mediante un sistema de autenticación y autorización, de manera que **solo usuarios con rol "ADMIN" podrán acceder** a esta funcionalidad. Los demás perfiles (usuarios estándar) no verán esta opción ni podrán ingresar por URL.

- **Interfaz intuitiva:**

El dashboard tendrá una interfaz amigable y estructurada, con menú lateral, buscador superior y navegación clara entre secciones (Usuarios, Lugares, Rutas, Configuración).

Este panel representa el **centro de operaciones internas** de la app, pensado para garantizar la integridad de los datos, facilitar la moderación de contenido y mantener una supervisión activa del uso del sistema.

### Identidad Visual: Nuevo Logo de MAPGENDA



El nuevo logo de **MAPGENDA** refleja una identidad moderna, versátil y profesional, alineada con los valores centrales de la plataforma: exploración, organización y tecnología.

- La palabra se divide visualmente en dos conceptos clave: “**MAP**” (mapa) y “**GENDA**” (de *agenda*), reforzando la idea de una **agenda geográfica personalizada**.
- El uso del **color azul oscuro** en “MAP” transmite confianza, estabilidad y tecnología.
- La letra “**G**” en dorado actúa como punto de anclaje visual y representa **el nexo entre mapa y planificación**, además de sugerir calidad y valor añadido.
- “**ENDA**” en verde simboliza crecimiento, movimiento y naturaleza, vinculándose con el turismo activo y sostenible.

En su versión alternativa con fondo azul, el logo gana fuerza y contraste, ideal para entornos digitales, dashboards o presentaciones institucionales.

Este rediseño busca posicionar a MAPGENDA como una solución confiable, accesible y visualmente coherente dentro del ecosistema de apps inteligentes para el turismo personalizado.

**Prototipo en Figma hay que iniciar session con google**

<https://www.figma.com/proto/eFZmLLPFk355aApuJgl4kl/Mapgenda-Fernando-Chang?node-id=3-22&p=f&t=4LBjSK23lhjd54e-1&scaling=min-zoom&content-scaling=fixed&page-id=0%3A1&starting-point-node-id=3%3A22>

### 3.7.3. Identificación de Perfiles de Usuario

La aplicación contempla dos perfiles de usuario diferenciados:

- **Usuario General (Registrado con cuenta de Google):**

Este perfil tiene acceso completo a todas las funcionalidades destinadas al público:

- Pantalla de inicio y login
- Pantalla de perfil
- Búsqueda de lugares y filtros personalizados
- Mapa interactivo con marcadores
- Gestión de favoritos
- Creación y edición de lugares personalizados
- Gestión de rutas offline
- Visualización y edición de rutas guardadas

- **Administrador (acceso vía dashboard web):**

Este perfil tiene acceso restringido al **panel de administración (Dashboard)** desarrollado con Spring Boot + Thymeleaf. Desde allí puede:

- Visualizar y gestionar usuarios registrados
- Ver, editar o eliminar lugares creados
- Supervisar rutas generadas
- Consultar métricas de uso

Cada perfil está asociado a un nivel de permisos y vistas distintas. El perfil administrador **no interactúa con la app móvil**, sino con la interfaz web del backend.

---

### 3.7.4. Especificación de Formatos de Impresión

La aplicación Mapgenda **no contempla la generación de documentos imprimibles** como parte de su funcionalidad actual.

Su foco está centrado en el uso móvil offline y la experiencia interactiva digital, por lo que no se generan reportes, tickets ni exportaciones en papel.

En caso de requerirse en el futuro, se diseñará un módulo adicional para exportación en PDF desde el dashboard de administración.

### 3.7.5. Especificación de la Navegabilidad entre Pantallas



La navegabilidad en la aplicación sigue un enfoque simple, directo e intuitivo, basado en una **barra de navegación inferior fija** (bottom menu) y un sistema de botones flotantes para acciones contextuales.

- Desde el **menú inferior**, el usuario puede acceder en todo momento a:

- Mapa
- Favoritos
- Modo Offline
- Perfil

- La **pantalla de inicio** (login) lleva directamente al mapa una vez autenticado.
- Desde el mapa, se accede:
  - A filtros avanzados de búsqueda
  - A creación de lugares personalizados
  - A rutas rápidas (vía botones flotantes)
- Desde cualquier lugar mostrado en el mapa se puede:
  - Ver información detallada
  - Guardar como favorito
  - Editar
  - Eliminar
  - Lanzar navegación externa en Google Maps
- El módulo offline permite:
  - Crear rutas paso a paso
  - Descargar mapas para navegación sin conexión
  - Ver y editar rutas guardadas previamente

El flujo entre pantallas es lineal y lógico, diseñado para minimizar pasos innecesarios y mantener al usuario siempre contextualizado dentro de la app.

## 4. Construcción del Sistema



### Aplicación de Consumo Android

La aplicación cliente del sistema ha sido desarrollada para la plataforma **Android**, utilizando como base el lenguaje de programación **Kotlin**. El enfoque de construcción se centró en seguir las prácticas modernas de desarrollo móvil, priorizando la eficiencia, mantenibilidad y experiencia de usuario.

### Tecnologías y Herramientas Empleadas

#### Lenguaje y Base de la Plataforma

- **Kotlin:** Android moderno.
- **Jetpack Compose:** Framework declarativo para la creación de interfaces gráficas.
- **Material 3 + Material Icons:** Se emplea la última especificación de Material Design.

## Arquitectura y Gestión de Estados

- **Navigation Compose:** Para el manejo de la navegación entre pantallas de forma declarativa y desacoplada.
- **Room:** Biblioteca de persistencia local, utilizada para gestionar y almacenar entidades clave de forma offline mediante SQLite, cumpliendo con el patrón DAO.
- **StateFlow / ViewModel:** Los estados de la UI y lógica se gestionan con ViewModel y StateFlow, facilitando la separación de responsabilidades (MVVM).

## Comunicación de Red

- **Retrofit:** Cliente HTTP robusto y extensible para la comunicación con el backend.
- **Gson:** Para la serialización/deserialización de objetos JSON.
- **Kotlin Coroutines:** El acceso a red y operaciones intensivas se realizan de forma asíncrona usando suspend functions y Dispatchers.IO, garantizando fluidez en la UI.

## Servicios de Google

La app integra múltiples APIs de Google para ofrecer capacidades inteligentes y centradas en ubicación:

- **Google Maps SDK:** Visualización de mapas con soporte para interacción de marcadores, rutas y zonas.
- **Places API:** Para sugerencias de lugares, búsqueda predictiva y autocompletado de direcciones.
- **Geocoding API:** Conversión entre coordenadas geográficas y direcciones reales.
- **Location Services:** Gestión precisa de ubicación del dispositivo.
- **Firebase Auth (vía Google Sign-In):** Para la autenticación segura del usuario mediante su cuenta Google.

## Multimedia y Visuales

- **Media3 (ExoPlayer)**: Uso de video en pantalla inicial, mediante la nueva versión modular y eficiente de ExoPlayer.
  - **Cloudinary**: Gestión externa de imágenes y videos subidos por el usuario, con capacidades de transformación y CDN.
  - **Coil**: Carga eficiente de imágenes remotas en la UI, integrada con Compose.
- 

## Consideraciones de Construcción

- El sistema fue diseñado pensando en **offline-first**, con sincronización de datos entre Room y el backend.
- Se siguió el patrón MVVM y se utilizó Coroutines para manejar flujos asíncronos, evitando el uso de callbacks o threads manuales.
- La navegación fue centralizada y desacoplada mediante rutas nombradas.
- El uso de APIs de Google permite una integración avanzada basada en ubicación sin complejidad innecesaria.
- Los assets multimedia fueron optimizados mediante Cloudinary, reduciendo el uso de almacenamiento local.

## Construcción del Backend de la API REST

La aplicación backend se desarrolló utilizando Spring Boot 3.4.4, un framework robusto y ampliamente adoptado en el ecosistema Java, ideal para la construcción de servicios RESTful de forma rápida, segura y modular.

### Tecnologías y Librerías Principales

- **Lenguaje y Entorno de Ejecución**
  - El sistema está construido en Java 21, aprovechando las nuevas capacidades del lenguaje.
  - El manejo del proyecto se realiza con Maven como sistema de construcción (pom.xml).
- **Framework Base**

- spring-boot-starter-web: Proporciona soporte completo para la construcción de APIs REST.
- spring-boot-starter-data-jpa: Facilita la integración con bases de datos mediante JPA (Java Persistence API).
- spring-boot-starter-security: Añade mecanismos de autenticación y autorización para asegurar los endpoints.

- **Persistencia**

- Se utiliza PostgreSQL como sistema gestor de base de datos.
- La conexión se maneja mediante el driver JDBC de PostgreSQL y la implementación de repositorios JPA.

- **DTOs y Mapeo Automático**

- Se emplea MapStruct para realizar conversiones eficientes entre entidades y objetos de transferencia de datos (DTOs), minimizando el código manual y mejorando la mantenibilidad.

- **Lombok**

- Se incorpora la biblioteca Lombok para reducir el boilerplate, como los getters, setters y constructores, manteniendo el código más limpio y conciso.

- **JSON y Serialización**

- El intercambio de datos se realiza en formato JSON, gestionado por la biblioteca Jackson (jackson-databind), usada por defecto por Spring Boot.

- **Autenticación con Google y JWT**

- La autenticación de usuarios utiliza Google Sign-In, validando el idToken con las bibliotecas google-api-client y google-http-client-jackson2.
- Una vez autenticado, el backend genera un token JWT (JSON Web Token) con jjwt, el cual es usado para controlar el acceso a los recursos protegidos.

- **Soporte para JAXB**

- Incluye soporte opcional para APIs que requieren JAXB, especialmente útil para interoperabilidad con XML si es necesario.

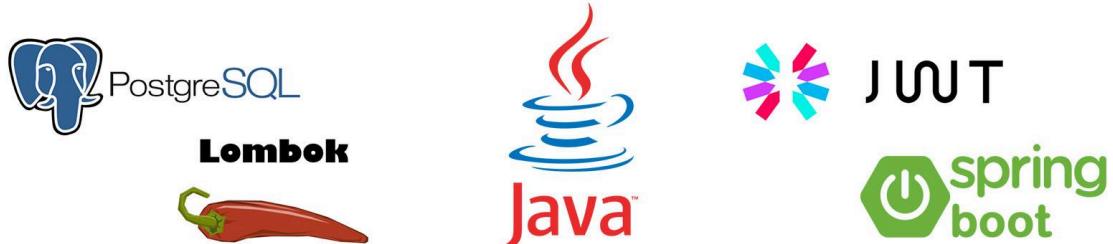
## Testing

Aunque el proyecto incluye las dependencias necesarias para pruebas (spring-boot-starter-test), en esta versión no se han implementado pruebas automatizadas explícitas. Sin embargo, el entorno está preparado para ello, permitiendo en futuras iteraciones agregar:

- Pruebas unitarias con JUnit
- Pruebas de integración
- Validación de endpoints y servicios

## Compilación y Procesamiento

- Se configura el plugin maven-compiler-plugin para compilar con compatibilidad a Java 21, y aplicar correctamente los procesadores de anotaciones (Lombok, MapStruct, Config Processor).
  - El empaquetado se realiza con el spring-boot-maven-plugin, permitiendo construir y ejecutar la API como una aplicación ejecutable (.jar) lista para desplegar.
- 



### Resumen de tecnologías usadas en el backend

Categoría	Tecnología o Librería
Framework	Spring Boot 3.4.4
Lenguaje	Java 21
Construcción	Maven
Persistencia	Spring Data JPA + PostgreSQL
Seguridad	Spring Security + JWT
Autenticación externa	Google Sign-In API
Mapeo DTO ↔ Entidades	MapStruct
Reducción de boilerplate	Lombok

Serialización JSON	Jackson
Testing (soportado)	Spring Boot Test (JUnit, Mockito, etc.)
Compilación avanzada	maven-compiler-plugin

## 4.2 DESPLIEGUE

The screenshot shows the 'Ungrouped Services' section on the Render platform. It displays three services: 'backend\_MAPGENDA' (Deployed, Docker, Oregon, 21h), 'admin-panel' (Deployed, Static, Global, 21h), and 'mapgenda-db' (Available, PostgreSQL 16, Oregon, 17d). A search bar and filter buttons ('Active (3)', 'Suspended (0)', 'All (3)') are also visible.

SERVICE NAME	STATUS	RUNTIME	REGION	DEPLOYED
backend_MAPGENDA	✓ Deployed	Docker	Oregon	21h
admin-panel	✓ Deployed	Static	Global	21h
mapgenda-db	✓ Available	PostgreSQL 16	Oregon	17d

## Despliegue y Entorno Productivo

La arquitectura del sistema Mapgenda ha sido desplegada exitosamente en **Render**, una plataforma de infraestructura como servicio (PaaS) que permite automatizar el alojamiento y escalado de aplicaciones modernas. El entorno consta de tres servicios principales, todos activos, funcionando de manera integrada y conectados a sus respectivos repositorios en GitHub:

### 1. Backend de la API (Spring Boot)

- **Servicio:** backend\_MAPGENDA
- **Estado:** Desplegado (Deployed)
- **Runtime:** Docker
- **Región:** Oregon
- **Descripción:** API REST desarrollada en Java con Spring Boot 3.4, expone los endpoints para autenticación, gestión de usuarios, lugares, rutas y sincronización offline. Se construyó y desplegó usando contenedor Docker conectado directamente al repositorio de GitHub, permitiendo despliegues automáticos con cada push a la rama principal.

### 2. Interfaz de administración (Panel web)

- **Servicio:** admin-panel
- **Estado:** Desplegado (Deployed)
- **Runtime:** Static (HTML, CSS, JavaScript)
- **Región:** Global
- **Descripción:** Aplicación web estática desarrollada con tecnologías web (HTML, JS puro y CSS), que sirve como panel de consulta y prueba de endpoints de la API. Ofrece una vista sencilla pero funcional para gestionar registros o verificar datos de manera visual.

### 3. Base de datos PostgreSQL

- **Servicio:** mappenda-db
- **Estado:** Disponible (Available)
- **Versión:** PostgreSQL 16
- **Región:** Oregon
- **Descripción:** Base de datos en la nube conectada directamente con el backend, utilizada para almacenar usuarios, rutas, ubicaciones y configuraciones. La instancia está configurada con acceso remoto, permitiendo conexión desde herramientas locales como pgAdmin, donde se gestionan manualmente datos en fase de desarrollo o pruebas.

## Flujo de CI/CD

Cada uno de los servicios está vinculado a su respectivo **repositorio en GitHub**, permitiendo un flujo de despliegue automatizado:

- Al realizar un *push* o *merge* en ramas configuradas, Render reconstruye la imagen (en el caso del backend con Docker) o actualiza automáticamente los archivos estáticos del panel web.
- Esto garantiza que las últimas versiones estén siempre reflejadas en producción sin necesidad de despliegues manuales.

## 5. Conclusiones

El proyecto MApagenda ha mantenido desde su concepción una idea clara y coherente: construir una aplicación de turismo personalizado que facilite al usuario la exploración de

lugares de interés cercanos, adaptándose a sus preferencias. Esta visión se ha concretado en una app que permite descubrir espacios culturales, de ocio y diversidad, con funciones clave como visualización en mapas, rutas personalizadas, descarga de contenidos y gestión de lugares guardados para uso offline.

Durante el desarrollo, uno de los retos técnicos más importantes fue la integración de Google Maps y Google Places API, tanto por sus beneficios como por sus limitaciones. Aunque estas APIs ofrecen potentes herramientas de geolocalización, navegación y búsqueda, implican también una fuerte dependencia del ecosistema Google: la app requiere que el usuario tenga instalada la app oficial de Google Maps y una cuenta de Gmail activa para acceder a todas las funcionalidades. Asimismo, el almacenamiento de rutas offline implica guardar fragmentos de mapas directamente en la app de Google Maps, lo que limita la personalización total desde la aplicación.

Otro aprendizaje clave fue trabajar con dos enfoques distintos para la visualización cartográfica:

- El uso de MapView combinado con AndroidView permitió acceder a funciones más avanzadas aún no soportadas por la nueva UI.
- En paralelo, se implementó Google Maps Compose, un enfoque moderno que se integra nativamente con Jetpack Compose, aunque con ciertas restricciones que forzaron soluciones híbridas en algunas pantallas.

La gestión de datos offline representó otro desafío considerable. Fue necesario implementar almacenamiento local con Room, estructurando cuidadosamente las entidades para guardar ubicaciones y rutas descargadas, permitiendo al usuario consultar información sin conexión. Esta funcionalidad ha sido fundamental para cumplir con la promesa de accesibilidad en zonas sin cobertura.

Además, aprendí a integrar el backend construido en Spring Boot con la app móvil mediante Retrofit, usando peticiones HTTP seguras con autenticación por JWT. Esta conexión entre cliente y servidor requirió comprender a fondo la serialización de datos, los flujos de sesión y la sincronización entre los datos locales y remotos.

A lo largo del desarrollo, la idea original del proyecto se mantuvo firme: ofrecer una experiencia de turismo inclusiva, adaptable y accesible, respetando las necesidades del usuario moderno. Aunque no se implementaron pruebas automatizadas, la arquitectura está preparada para incorporar pruebas unitarias y de interfaz en versiones futuras.

## 6. GLOSARIO DE TÉRMINOS

**Botones Flotantes (FABs):** Elementos de UI usados para acceso rápido a funciones como filtros, creación de lugares o rutas.

**Sistema de Chips:** Interfaz de selección visual de categorías y subcategorías de lugares.

**Usuario Registrado:** Usuario autenticado vía cuenta de Google con acceso completo a funcionalidades.

**Gestor de Contenidos:** Usuario con rol especial que edita y valida lugares turísticos desde el panel administrativo.

**Google Directions API:** Servicio para generar rutas (caminando, en bici) entre puntos.

**Firebase Auth:** Sistema de autenticación usado para login con Google.

**Room:** Base de datos local en Android usada para funcionar sin conexión.

**PostgreSQL:** Base de datos relacional usada en el backend para almacenamiento persistente.

**Spring Boot:** Framework en Java utilizado para desarrollar el backend de la API.

**Retrofit:** Cliente HTTP en Android para comunicarse con el backend.

**JWT (JSON Web Token):** Mecanismo de autenticación y autorización segura entre cliente y servidor.

**Jetpack Compose:** Framework de UI declarativa para construir interfaces Android modernas.

**MVVM (Model-View-ViewModel):** Patrón de arquitectura usado en la app para separar lógica de presentación y datos.

**MapStruct:** Herramienta para mapear objetos DTO ↔ Entidades en el backend.

**Panel de Administración (Dashboard):** Interfaz web para administradores, permite gestionar usuarios, lugares y rutas.

**DTO (Data Transfer Object):** Objetos usados para transportar datos entre frontend y backend.

## 7. BIBLIOGRAFÍA DEL PROYECTO

### Servicios Web y Frameworks

Spring Boot

<https://spring.io/projects/spring-boot>

### PostgreSQL

<https://www.postgresql.org/>

### Render (Despliegue Backend + PostgreSQL + Html + Js)

<https://render.com/>

### GitHub (Repositorio y control de versiones)

<https://github.com/>

**Cloudinary (Gestión de imágenes en la nube)**

<https://cloudinary.com/>

Librerías Android

**Jetpack Compose**

<https://developer.android.com/jetpack/compose>

**Room (Base de datos local)**

<https://developer.android.com/jetpack/androidx/releases/room>

**Retrofit (Cliente HTTP)**

<https://square.github.io/retrofit/>

**Gson (Serialización JSON)**

<https://github.com/google/gson>

**Coil (Carga de imágenes en Android)**

<https://coil-kt.github.io/coil/>

**Media3 (ExoPlayer moderno)**

<https://developer.android.com/media/media3>

**Navigation Compose**

<https://developer.android.com/jetpack/compose/navigation>

**Kotlin Coroutines**

<https://developer.android.com/kotlin/coroutines>

**StateFlow / ViewModel**

<https://developer.android.com/topic/libraries/architecture/viewmodel>

APIs de Google

**Google Maps SDK**

<https://developers.google.com/maps/documentation/android-sdk>

**Google Places API**

<https://developers.google.com/maps/documentation/places/web-service/overview>

**Google Directions API**

<https://developers.google.com/maps/documentation/directions/overview>

**Geocoding API**

<https://developers.google.com/maps/documentation/geocoding/overview>

**Google Sign-In (Firebase Auth)**

<https://developers.google.com/identity/sign-in/android/start>

Backend y Seguridad

**Spring Security**

<https://spring.io/projects/spring-security>

**JWT (Autenticación con JSON Web Tokens)**

<https://jwt.io/introduction>

**MapStruct (Mapeo DTO ↔ Entidades)**

<https://mapstruct.org/>

**Lombok**

<https://projectlombok.org/>