
This is the **published version** of the bachelor thesis:

Gutiérrez Vila, Marta. *CityFix : Plataforma de Gestión Colaborativa de Incidencias Urbanas*. Treball de Final de Grau (Universitat Autònoma de Barcelona), 2026 (Enginyeria Informàtica)

This version is available at <https://ddd.uab.cat/record/334472>

under the terms of the  license.

CityFix - Plataforma de Gestión Colaborativa de Incidencias Urbanas

Marta Gutiérrez Vila

30 de junio de 2026

Resumen— CityFix es una aplicación diseñada para optimizar la gestión de incidencias urbanas y mejorar la comunicación entre la ciudadanía y las administraciones locales. La plataforma responde a la necesidad de agilizar los flujos de reporte tradicionales, ofreciendo un sistema automatizado e inteligente gracias a la integración de modelos de lenguaje de gran tamaño. El sistema permite a los ciudadanos reportar desperfectos en la vía pública con solo subir una imagen, la cual es analizada automáticamente por la Inteligencia Artificial para clasificar la categoría, el título y la prioridad de la incidencia, capturando además su ubicación geográfica exacta. Los técnicos municipales disponen de un panel especializado para la asignación y el seguimiento de tareas en tiempo real.

Palabras clave— Large Language Model, Sistema de Despacho Automatizado, geolocalización, smart city, gestión de incidencias, reconocimiento de patrones, eficiencia administrativa

Abstract— CityFix is an application designed to optimize the management of urban incidents and enhance communication between citizens and local authorities. The platform addresses the need to streamline traditional reporting workflows by offering an automated and intelligent system powered by the integration of Large Language Models. The system enables citizens to report public infrastructure issues simply by uploading an image, which is automatically analyzed by Artificial Intelligence to determine the incident's category, title, and priority, while also capturing its exact geographical location. In addition, municipal technicians have access to a specialized dashboard for real-time task assignment and monitoring.

Keywords— Large Language Model, Automated Dispatch System, geo-location, smart city, incident management, pattern recognition, administrative efficiency



1 INTRODUCCIÓN

Si hay un aspecto del que dependa la calidad de vida de cualquier ciudad, independientemente de su localización o población, se trata de la gestión adecuada del espacio público. Aspectos como el pavimento, la iluminación, los muebles urbanos y la señalización requieren un mantenimiento regular para asegurar la seguridad de la urbanización y el bienestar de los ciudadanos.

No obstante, la identificación y el reporte de desperfectos suelen llevarse a cabo mediante métodos tradicionales poco efectivos, tales como las llamadas telefónicas o los correos

electrónicos. Estos factores complican innecesariamente el seguimiento del proceso de resolución de los problemas.

Actualmente, las administraciones públicas buscan la evolución de sus servicios tradicionales mediante la digitalización e impulsan constantemente iniciativas de ciudades inteligentes. Es por este motivo que cualquier comunidad se beneficiaría de contar con herramientas tecnológicas que faciliten una comunicación directa entre los ciudadanos y los servicios municipales.

Este Trabajo de Fin de Grado detalla el diseño y el desarrollo de una plataforma integral para la notificación y gestión de desperfectos en la vía pública. El sistema propuesto permite a los ciudadanos informar sobre incidencias mediante la captura de fotos y la geolocalización automática. Adicionalmente, ofrece a los técnicos municipales un panel de control para manejar el ciclo de vida de cada incidencia a través de una máquina de estados robusta, asignar recursos e informar sobre la resolución definitiva.

El objetivo principal de la iniciativa es optimizar la eficien-

- E-mail de contacto: martagutivila@gmail.com
- Mención: Ingeniería del Software
- Trabajo tutorizado por: Sergio Bachiller Rubia (Área de Ciencias de la Computación e Inteligencia Artificial)
- Curso 2025/26
- Repositorio: <https://github.com/marta-gutierrez/AppCityFix>

cia operativa de los servicios de mantenimiento municipal, incrementar la transparencia en el proceso de resolución y promover la participación ciudadana mediante herramientas digitales fáciles de usar.

Este documento expone el contexto del proyecto, los requisitos funcionales y no funcionales, la arquitectura del sistema, la implementación técnica, las pruebas realizadas y la evaluación de los resultados obtenidos.

2 OBJETIVOS

2.1. Objetivo General

El objetivo principal del proyecto consiste en el diseño y desarrollo de una plataforma web que permita la gestión y el seguimiento de las incidencias en la vía pública, conectando así la ciudadanía con los servicios municipales y optimizando el ciclo de vida de cada incidencia.

2.2. Objetivos específicos

Con tal de alcanzar esta meta, se definen también los siguientes objetivos específicos:

- Definir e implementar una máquina de estados robusta para el control y seguimiento del flujo de cambios de estado, los cuales abarcan desde la apertura hasta el cierre de la incidencia, con tal de enviar notificaciones push al usuario que le informen del progreso en su reporte.
- Integrar un sistema de geolocalización y visualización SIG, utilizando la API de OpenStreetMap para permitir la captura automática de coordenadas y la visualización interactiva de incidencias sobre un mapa en tiempo real.
- Crear un panel de administración para la gestión operativa, el cual permitirá priorizar las incidencias según su gravedad y asignarlas automáticamente a un técnico para su resolución, además de adjuntar evidencias de la reparación.

3 ESTADO DEL ARTE

Actualmente existen múltiples alternativas a este tipo de plataforma, la mayoría de ellas desarrolladas y gestionadas por los propios ayuntamientos de las ciudades donde se distribuyen. No obstante, esto también implica que su funcionalidad se limita únicamente a esa zona geográfica, como es el caso de la aplicación australiana SnapSendSolve [1] y la plataforma estadounidense Find It Fix It [2].

Existe también software perteneciente a empresas que ofrecen sus servicios a los ayuntamientos locales dispuestos a contratarlos. Este es el caso de la plataforma europea Fixi [3]. Todas estas aplicaciones cuentan con las mismas funcionalidades genéricas: geolocalización del usuario mediante el uso de GPS estándar, flujo de estados gestionado internamente por la plataforma y notificaciones estándar.

En el caso de SnapSendSolve y Find It Fix It, estas se ven limitadas por una arquitectura no modular y un panel administrativo limitado al backend municipal. Tomando en cuenta todos estos factores, CityFix propone las siguientes optimizaciones y funcionalidades adicionales:

- Mayor facilidad para los usuarios a la hora de reportar incidencias y a los técnicos a la hora de priorizarlas, agilizando el proceso mediante una clasificación automática de las fallas.
- Apoyo visual en forma de mapas de calor para proporcionar información útil a los ayuntamientos sobre futuras medidas a tomar en las zonas más afectadas.
- Interacción constante entre usuarios y técnicos para poder recibir feedback sobre el proceso y resultado de reparación y asegurar la satisfacción de los ciudadanos.

4 METODOLOGÍA

Dada la naturaleza individual del proyecto, se ha decidido aplicar una metodología ágil Scrum [4] adaptada a una sola persona. Se ha combinado con Kanban [5] para la gestión del flujo del trabajo. Esta es una manera visual y dinámica de observar en todo momento la progresión de cada tarea a realizar, facilitando la priorización de tareas y maximizando la eficiencia.

La herramienta utilizada para aplicar dichas metodologías es Jira [6], una plataforma del entorno Atlassian [7] diseñada para la gestión de proyectos y el seguimiento de incidencias y tareas. De esta manera, el proyecto ha sido dividido en sprints y se ha clasificado cada tarea en 3 columnas: Por hacer, En curso y Finalizado. El uso de Jira como herramienta de gestión de flujo de trabajo ha contribuido a la elección de Confluence [8] como plataforma de gestión de la documentación, puesto que su estatus como herramienta de Atlassian permite una sencilla vinculación entre su propio espacio y el proyecto de Jira.

En cuanto al control de versiones, se ha empleado GitHub [9] como repositorio, facilitando el almacenaje y la gestión del código fuente. Además, el uso de ramas paralelas ha permitido desarrollar nuevas funcionalidades, realizar pruebas y corregir errores de manera aislada sin afectar a la versión principal del proyecto.

5 PLANIFICACIÓN

Con tal de asegurar un desarrollo eficiente del proyecto, se han definido una serie de sprints, cada uno enfocado en una fase del proyecto distinta. A la hora de decidir estas divisiones se han tomado como pauta tanto los objetivos del proyecto como los requisitos funcionales y no funcionales estipulados posteriormente en el apéndice. Se ha dividido el proyecto en un total de cuatro sprints y cinco tipos de actividades (Documentación, Diseño, Frontend, Backend y Testeo):

- Sprint 1 (20/02-08/03): Destinado principalmente a la planificación del proyecto y elaboración de documentación inicial, incluyendo diagramas de casos de uso y una valoración inicial de posibles alternativas para la fase de desarrollo en el futuro.
- Sprint 2 (09/03-12/04): Enfocado en el diseño y maquetado de la interfaz de usuario y *user experience*. Incluye también la creación de la base de datos y las tablas principales.

- **Sprint 3 (13/04-24/05):** Fase de desarrollo que engloba frontend y backend y todas sus tareas específicas. Algunas de estas incluyen definición y enlace de endpoints o implementación de lógica de negocio, entre otros.
- **Sprint 4 (25/05-14/06):** Enfocado en el testeo de la aplicación y la corrección de errores. Adicionalmente, se ha dedicado parte del sprint a refinar la documentación.

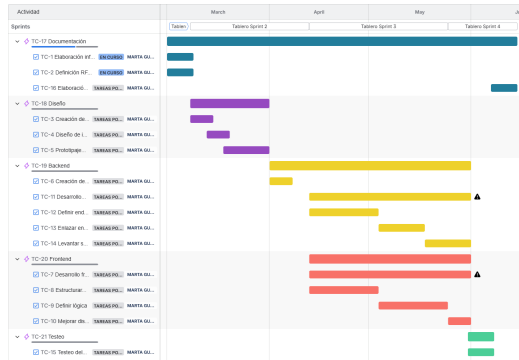


Fig. 1: Diagrama de Gantt

6 REQUISITOS DEL SISTEMA

Para concretar las funcionalidades de CityFix, se han definido los requisitos funcionales y no funcionales del sistema. Esta documentación constituye la base del proyecto y establece de forma clara y precisa todo lo que se espera de la aplicación. Estos se encuentran en el apartado A4 de Apéndices.

7 DISEÑO

7.1. Patrón MVC

En cuanto a la arquitectura del sistema, se ha optado por el uso del patrón Modelo-Vista-Controlador [10]. Este patrón permite separar la lógica de la aplicación en tres componentes principales:

- **Modelo:** Encargado de gestionar los datos y la lógica de negocio.
- **Vista:** Responsable de la presentación de la información al usuario y la interacción con el sistema.
- **Controlador:** Intermediario entre las otras dos capas que procesa las peticiones y actualiza el sistema en consecuencia.

En este caso, las capas de Modelo y Controlador se implementan en el backend mientras que la capa Vista corresponde al frontend, desarrollado con Angular.

7.2. UI/UX

Para el diseño de la interfaz de usuario y la experiencia de usuario, se siguió un proceso iterativo que comenzó con la elaboración de un boceto sencillo. Este primer esquema,

el cual se encuentra en el Apéndice A2, permitió definir de manera rápida la estructura general de la aplicación, la disposición de los elementos principales y los flujos básicos de navegación.

Posteriormente, este boceto inicial se desarrolló con mayor nivel de detalle mediante el uso de la herramienta Figma [11]. En esta fase se concretaron aspectos visuales como la tipografía, la paleta de colores, los componentes interactivos y la organización final de las pantallas.

7.3. Diagramas UML

Con tal de entender y documentar los posibles flujos de ejecución en el sistema, se han confeccionado diversos diagramas de casos de uso, los cuales se encuentran en el Apéndice A1.

8 DESARROLLO

El proyecto está formado principalmente por los siguientes componentes: Frontend, Backend y Supabase [18]. El Frontend se encarga de realizar las solicitudes al Backend, mientras que el Backend procesa dichas peticiones y obtiene los datos desde la base de datos PostgreSQL alojada en Supabase.

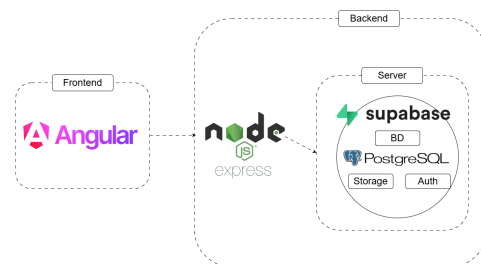


Fig. 2: Arquitectura de CityFix

8.1. Frontend

El cliente de la aplicación se ha construido utilizando Angular, un framework de desarrollo de aplicaciones web de tipo SPA (Single Page Application) basado en TypeScript. Angular ha sido especialmente útil debido principalmente a su ecosistema completo, a su enlace de datos bidireccional y, en especial, a su arquitectura orientada a pruebas [12]. Angular permite la actualización en tiempo real de la aplicación, cosa que ha facilitado enormemente el proceso de desarrollo. Adicionalmente, se ha adoptado el paradigma moderno de Standalone Components, eliminando la dependencia de módulos globales (NgModules) para mejorar el tiempo de carga y la eficiencia de la compilación. Como parte del proceso de desarrollo, se han establecido primeramente las vistas por sí solas con su flujo lógico y métodos propios empleando mock objects. De este modo, se ha aislado la capa de interfaz todavía más del backend, asegurando su funcionamiento independiente correcto en un entorno controlado primero para después poder implementar las llamadas a los servicios posteriormente.

8.1.1. Estructura General

El diseño del frontend implementa un patrón de Separación de Responsabilidades que aísla la interfaz de usuario de las reglas de comunicación de red. Esto se traduce en una división estricta entre Vistas (Componentes) y Capas de Abstracción de Red (Requests/Services).

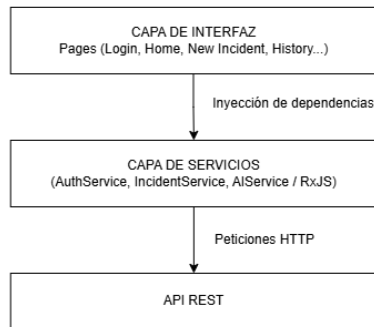


Fig. 3: Arquitectura de separación de responsabilidades

8.1.2. Librerías y Dependencias Clave

- `@angular/common/http` (`HttpClient`): Módulo nativo fundamental para efectuar peticiones asíncronas HTTP hacia el servidor Node.js. Implementa el patrón Observable, sustituido mediante promesas en puntos críticos para un control lineal del flujo.
- `rxjs`: Librería de programación reactiva utilizada para gestionar flujos de datos asíncronos.
- `@angular/router`: Gestor de enrutamiento del framework que permite la navegación dinámica entre las páginas del cliente sin necesidad de recargar el navegador.
- `Leaflet`: Librería para mapas interactivos en JavaScript. Se ha tenido en cuenta su ligereza y simplicidad, combinadas con una amplia gama de features, a la hora de decidir implementarlo como mapa principal de la aplicación y como base para el mapa de calor. [13]
- `signal`: Sistema de reactividad fina de `@angular/core` que permite manejar el estado de la UI con pequeñas unidades reactivas para mejorar el rendimiento. Ha sido especialmente útil para problemas de renderizado en ciertas pantallas del proyecto, especialmente a la hora de obtener datos de Supabase y renderizar marcadores en el mapa. [14]
- `gsap`: Librería de animaciones empleada para mejorar la experiencia del usuario a través de la capa visual sin comprometer el rendimiento. [15]

8.2. Backend

El servidor y núcleo de la lógica de negocio se ha desarrollado sobre Node.js. Node.js es el candidato perfecto en este caso debido a su arquitectura basada en eventos, que proporciona una alta eficiencia y escalabilidad, además de

su gran sinergia con Angular al poder unificar el lenguaje de desarrollo. [16] Se ha empleado el framework Express para la creación de una API REST de alto rendimiento. El backend actúa como una capa intermedia de seguridad y procesamiento pesado. Adicionalmente, es el único componente del sistema con autorización para comunicarse con los servicios de terceros (Google Gemini) y la base de datos (Supabase). Este enfoque ha permitido definir un conjunto de endpoints concretos mediante métodos estándar de HTTP (POST, GET, DELETE, PATCH).

8.2.1. Librerías y Dependencias Clave

- `express`: Framework minimalista encargado de la gestión del servidor HTTP, el ciclo de peticiones/respuestas y la definición de los diferentes endpoints (POST `/api/incidents`, GET `/api/auth/me`, etc.).
- `@supabase/supabase-js`: SDK oficial para la integración de Supabase en entornos de servidor. Permite la persistencia de datos y la gestión del almacenamiento de archivos binarios utilizando privilegios administrativos seguros a través del rol `service_role`. [18]
- `@google/generative-ai`: Cliente oficial de Google para interactuar con los modelos fundacionales de Gemini. Se ha decidido usar Gemini dado que se trata de un modelo multimodal, capaz de procesar texto e imágenes de forma simultánea en la misma red neuronal. Se utiliza para el procesamiento del lenguaje natural y el análisis computerizado de imágenes adjuntas por los ciudadanos. [17]
- `multer`: Middleware diseñado para el procesamiento y manipulación de datos de tipo multipart/form-data. Facilita la recepción de archivos binarios (imágenes) en la memoria del servidor sin colapsar el almacenamiento físico local.
- `dotenv`: Módulo de configuración que inyecta las variables del archivo confidencial `.env` directamente en el entorno de ejecución de Node.js (`process.env`), aislando las claves privadas del código fuente.
- `cors`: Mecanismo de seguridad configurado en Express para permitir el intercambio de recursos de origen cruzado, habilitando de forma controlada que el cliente de Angular (`localhost:4200`) consuma los servicios expuestos por el API (`localhost:3000`).

8.3. Supabase

Para la capa de persistencia de datos y servicios de infraestructura, se ha seleccionado Supabase, una plataforma open-source de tipo BaaS (Backend-as-a-Service) construida sobre PostgreSQL. [18] Las mayores ventajas de implementar Supabase en este proyecto son su robusta base de datos relacional, que garantiza la integridad referencial en el contexto de una aplicación transaccional; su seguridad nativa en la capa de datos mediante Row-Level Security; y, especialmente, su ecosistema integrado, que reúne en una única plataforma la base de datos, el sistema de autenticación de usuarios (Auth) y el almacenamiento de archivos binarios (Storage) para las imágenes adjuntas.

8.3.1. Auth

Se ha empleado el módulo de autenticación de Supabase para gestionar tanto el registro como el inicio de sesión de los usuarios en la aplicación. Este servicio ha permitido implementar un sistema de autenticación seguro basado en correo electrónico y contraseña, evitando la necesidad de desarrollar una solución propia.

8.3.2. Base de Datos

Con tal de almacenar y gestionar los datos relativos al sistema, se ha empleado la base de datos PostgreSQL en Supabase. La estructura completa puede ser consultada en el apartado A3 de la sección Apéndice. La base de datos está compuesta por tablas que representan los elementos principales de la aplicación.

Roles de usuario y privilegios:

Para garantizar la confidencialidad, integridad y disponibilidad del sistema, la base de datos define tres roles de usuario genéricos completamente diferenciados. El control de accesos se coordina mediante la directiva `ENABLE ROW LEVEL SECURITY` en las tablas estratégicas, aplicando políticas basadas en la identidad del usuario autenticado (`auth.uid()`) y su rol asignado.

- Ciudadano (user): Representa al usuario final de la vía pública que reporta los desperfectos urbanos. Es el rol con mayores restricciones operativas. Sus privilegios se limitan mayoritariamente a insertar incidencias y modificar sus datos de perfil y las incidencias creadas por él mismo.
- Técnico (technician): Representa al personal de campo encargado de la reparación física de las averías urbanas. Sus privilegios están supeditados a las tareas que el algoritmo le asigne.
- Administrador (admin): Representa al supervisor municipal o gestor del sistema. Cuenta con inmunidad frente a las restricciones ordinarias del sistema para asegurar la gobernanza de la plataforma.

Tipos de tablas:

- users: Guarda los usuarios del sistema (nombre, rol: usuario, técnico o admin).
- technicians: Subconjunto de usuarios que son técnicos, con su especialización y estado activo.
- incidents: Incidentes reportados: título, descripción, ubicación (lat/lon), estado, prioridad y tipo.
- incident_images: Imágenes asociadas a incidentes (reporte, progreso o resolución).
- incident_assignments: Asignaciones de técnicos a incidentes (quién lo atiende y cuándo).
- incident_updates: Historial de cambios del incidente (estado y comentarios).
- resolutions: Información de resolución de un incidente (quién lo resolvió y cómo).

- resolution_feedback: Valoraciones y comentarios del usuario sobre la resolución.
- notifications: Notificaciones persistentes para la bandeja de notificaciones de la página principal y su estado (abiertas o no abiertas).
- push_subscriptions: Suscripciones de notificaciones push en formato JSON.

8.3.3. Storage

El servicio de almacenamiento en la nube de Supabase se ha utilizado para guardar las imágenes vinculadas al sistema, concretamente las relacionadas con incidencias y reparaciones. El uso de Storage ha permitido gestionar de forma eficiente los recursos multimedia y asegurar un acceso rápido y seguro a los archivos desde cualquier dispositivo.

8.4. Visualización Geoespacial Asíncrona (Leaflet)

El mapa de la pantalla de inicio actúa como el centro operativo del ciudadano. Leaflet se integra en Angular rompiendo el ciclo de renderizado nativo en puntos críticos para optimizar el rendimiento mediante el uso de contenedores del DOM directos (L.map).

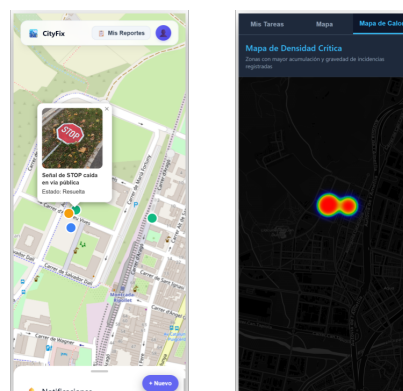


Fig. 4: Vista de marcadores en el mapa Leaflet y vista del mapa de calor

El sistema utiliza la arquitectura de capas de Leaflet para superponer los marcadores dinámicos sobre un mapa base de OpenStreetMap. El servicio de incidencias recupera las coordenadas (latitude, longitude) y el estado (status) desde la API. Leaflet procesa este array en bucle, instanciando objetos `L.marker` y vinculándoles ventanas emergentes (`L.popup`) personalizadas con el título de la avería. Este sistema implementa también escuchas de eventos geoespaciales asíncronos: cuando el usuario arrastra el mapa o ubica una nueva incidencia, Leaflet captura el evento matemático del píxel y lo transforma en coordenadas geográficas decimales utilizables por el backend.

Leaflet también permite al sistema incorporar una capa de mapa de calor, la cual reutiliza las coordenadas de las incidencias para aplicarlas a un algoritmo de densidad térmica. En lugar de limitarse a una representación por acumulación geográfica, el sistema aplica un factor de ponderación dinámico basado en el atributo de prioridad de cada registro.

8.5. Motor de Procesamiento de Incidencias (Google AI SDK)

En el backend, la librería `@google/generative-ai` se trata del núcleo inteligente de la plataforma. La propuesta de valor de CityFix consiste en la transformación de una sola imagen proporcionada por el ciudadano en un reporte refinado y estructurado sin necesidad de esfuerzo por parte del usuario.

Esto se lleva a cabo mediante una inyección de prompts estructurados. Con tal de evitar ambigüedades e inconsistencias, no se realizan consultas abiertas a la IA. El backend formatea la petición encapsulando la imagen (convertida a búfer binario en formato de objeto `InlineData`) dentro de un sistema de reglas estricto. El prompt exige al modelo que actúe como un clasificador urbano profesional y devuelva únicamente un objeto estructurado en formato JSON.

```

1 const model = ai.getGenerativeModel({
2   model: "gemini-2.5-flash",
3   generationConfig: {
4     responseMimeType: "application/json",
5     responseSchema: {
6       type: "object",
7       properties: {
8         title: {
9           type: "string",
10          description:
11            "Título conciso e identificativo. Máximo 50 caracteres.",
12        },
13        description: {
14          type: "string",
15          description:
16            "Descripción detallada de los desperfectos. Máximo 300 caracteres.",
17        },
18        category: {
19          type: "string",
20          description:
21            "Categoría estricta: iluminación, calzada, señalización, zonas verdes, mobiliario.",
22        },
23        priority: {
24          type: "string",
25          description: "Prioridad estricta: low, medium o high.",
26        },
27      },
28      required: ["title", "description", "category", "priority"],
29    },
30  },
31 });

```

Fig. 5: Configuración del modelo

El modelo analiza los patrones visuales de la imagen (por ejemplo, grietas en el asfalto o farolas apagadas) para generar los parámetros necesarios en base a su interpretación. En caso de que el usuario haya proporcionado información manualmente, también verifica si la descripción del usuario se corresponde con la realidad, eliminando el error humano en el triaje de la incidencia.

8.6. Sistema Automatizado de Gestión de Recursos

El objetivo principal de este módulo es optimizar la resolución de incidencias en la vía pública mediante la automatización del proceso de asignación de tareas a los técnicos disponibles. Se busca sustituir la gestión manual tradicional por un Modelo de Puntuación Ponderada ejecutado en el servidor.

8.6.1. Criterios de Filtrado

El algoritmo opera en dos fases diferenciadas: una fase de descarte estricto y una fase de evaluación matemática. En la primera fase, se realiza una consulta a la base de datos aplicando un doble filtro booleano para identificar a los técnicos candidatos legibles:

$$\text{Candidato Elegible} \iff (\text{is_active} = \text{True}) \wedge (\text{Categoría Incidencia} \in \text{specialization})$$

Cualquier técnico que no se encuentre en servicio activo o cuya matriz de competencias no incluya la categoría específica de la incidencia es descartado inmediatamente del proceso con tal de reducir la carga computacional del servidor.

8.6.2. Modelado Matemático de la Heurística de Puntuación

Para los candidatos que superan el filtro inicial, el sistema calcula un índice de idoneidad compuesto, denominado Puntuación Final ($Score_{final}$). Este índice se sitúa en un rango normalizado comprendido entre 0 y 1, donde el valor 1 representa la asignación óptima. El cálculo se desglosa en dos dimensiones independientes:

Factor de Carga de Trabajo ($LoadScore$): Para medir el esfuerzo actual de un técnico, no se evalúa únicamente el número bruto de tareas, sino el impacto de su gravedad. Se asignan Puntos de Esfuerzo (PE) según la prioridad de las órdenes de trabajo activas (Alta = 5, Media = 3, Baja = 1). La carga total acumulada de un técnico se define como:

$$PE = \sum_{i=1}^n w_i \quad \text{donde } w_i \in \{1, 3, 5\}$$

Para transformar este valor en una métrica donde el técnico menos saturado obtenga la mayor puntuación, se aplica una función de proporcionalidad inversa, añadiendo una constante en el denominador para evitar la indeterminación matemática por división entre cero:

$$LoadScore = \frac{1}{PE + 1}$$

Factor de Proximidad Geográfica ($DistanceScore$): La distancia en línea recta entre la ubicación de la nueva incidencia y la última posición conocida del técnico se calcula mediante la Fórmula de Haversine, la cual considera la esfericidad del planeta Tierra:

$$\Delta lat = lat_2 - lat_1$$

$$\Delta lon = lon_2 - lon_1$$

$$a = \sin^2\left(\frac{\Delta lat}{2}\right) + \cos(lat_1) \cdot \cos(lat_2) \cdot \sin^2\left(\frac{\Delta lon}{2}\right)$$

$$c = 2 \cdot \arctan2(\sqrt{a}, \sqrt{1-a})$$

$$d = R \cdot c$$

Donde R representa el radio medio terrestre (6371 km) y d es la distancia resultante en kilómetros. Si el operario no dispone de tareas en curso, se asume disponibilidad geográfica inmediata ($DistanceScore = 1$). En caso contrario, la distancia se normaliza inversamente:

$$DistanceScore = \frac{1}{d + 1}$$

8.6.3. Resolución de la Ecuación Ponderada y Asignación

Una vez obtenidos ambos coeficientes normalizados, se aplica una combinación lineal ponderada mediante pesos

específicos (α y β), que permiten calibrar el comportamiento del sistema según las necesidades estratégicas del municipio:

$$Score_{final} = (\alpha \cdot LoadScore) + (\beta \cdot DistanceScore)$$

$$\text{Restricciones: } \alpha + \beta = 1 \quad \wedge \quad \alpha, \beta \geq 0$$

En la implementación actual del proyecto, se ha establecido una ponderación simétrica ($\alpha = 0,5$ y $\beta = 0,5$), otorgando el mismo nivel de importancia al balanceo de la carga de trabajo que a la reducción de costes por desplazamiento de los operarios. Finalmente, el sistema ordena de forma descendente los resultados y selecciona al técnico que presenta el mayor índice:

$$\text{Técnico Asignado} = \max_j (Score_{final,j})$$

El identificador del técnico seleccionado se inserta automáticamente como una nueva relación transaccional en la tabla de asignaciones, modificando de forma atómica el estado de la incidencia a 'asignada'.

8.6.4. Análisis Comparativo e Inspiración en Sistemas de Despacho Comerciales

El diseño de la heurística de asignación automatizada de este proyecto no constituye un modelo teórico aislado, sino que se encuentra directamente inspirado en los Sistemas de Despacho Automatizado (ADS) empleados por plataformas líderes de la Gig Economy y la logística bajo demanda, tales como Glovo (a través de su motor algorítmico Jarvis), Uber y Deliveroo. [19][20]

Una de las principales inspiraciones respecto a los ADS existentes se trata del concepto de ruta activaz agrupación. La heurística implementada se inspira directamente en este principio al omitir una "base central física" para los operarios. Al calcular el *DistanceScore*, el sistema toma como referencia las coordenadas del técnico y priorizará asignarle una tarea de su especialidad en las cercanías.

Adicionalmente, se han llevado a cabo adaptaciones y diferencias estratégicas. Los mecanismos de emparejamiento de las plataformas comerciales tienden a acumular peticiones durante ventanas de tiempo para resolver un problema de combinatoria global. Dado que en el entorno municipal el volumen de incidencias simultáneas por minuto será mucho menor, se prioriza en el algoritmo la inmediatez de asignación.

También se adapta al proyecto la métrica de distancia, ya que en otros ADS donde el cliente paga por la inmediatez del servicio (como podría ser una entrega) se utiliza el Tiempo Estimado de Llegada (ETA). Dado que en este contexto no existe un límite de tiempo concreto de desplazamiento, se usa la distancia ortodrómica (Haversine).

9 RESULTADOS

9.1. Análisis Comparativo del Comportamiento Operativo del Algoritmo de Asignación Automática

Para evaluar la eficacia de la heurística de asignación propuesta, se ha sometido el sistema a un escenario práctico

simple en un entorno urbano simulado, comparando el comportamiento del algoritmo de asignación de incidencias de CityFix frente a un Sistema de Despacho Automatizado comercial clásico.

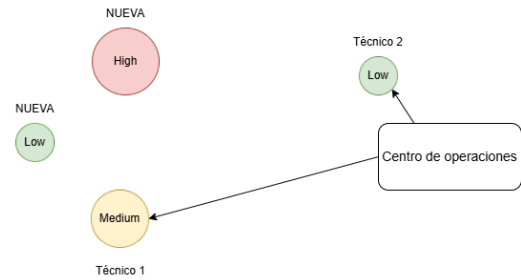


Fig. 6: Situación Inicial

En este caso, se parte de una situación en la que actualmente en una misma zona hay un Técnico 1 ya despachado trabajando en una incidencia de prioridad media y un Técnico 2 despachado más cerca del centro de operaciones trabajando en una incidencia de prioridad baja. Se reportan en un corto periodo de tiempo dos incidencias: una de baja prioridad y una de alta prioridad. Ambas se encuentran más cerca del Técnico 1.

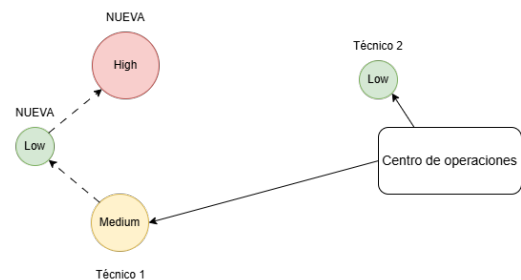


Fig. 7: Algoritmo ADS Comercial

Un Sistema de Despacho Automatizado comercial tendría en cuenta únicamente la proximidad geográfica a la hora de determinar la ruta y coordinar a los técnicos, por lo que asignaría ambas incidencias al Técnico 1. Sin embargo, la incidencia de baja prioridad se encuentra más cerca, por lo que la ruta a seguir establecería que la incidencia de alta prioridad sería atendida tras la resolución de la de baja prioridad. Esto sería lo óptimo en el contexto de, por ejemplo, una entrega a domicilio, donde una vez llegado al destino, las tareas son indistinguibles a nivel de esfuerzo humano y urgencia, pero es insatisfactorio en el contexto de la gestión municipal.

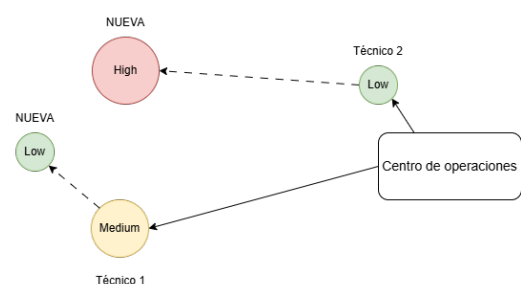


Fig. 8: Algoritmo Propio CityFix

Por otro lado, el modelo de puntuación ponderada de CityFix cuenta con otra métrica: la distribución del trabajo en base al factor de prioridad de cada tarea. De esta manera, la distribución de recursos automatizada no se limita únicamente a valorar la proximidad física, sino que se asegura de que cada reporte tarde el menor tiempo posible en ser atendido. Como se puede observar en los gráficos anteriores, aunque ambos sistemas comparten principios de optimización geoespacial, el comportamiento del modelo desarrollado resulta más adecuado lógicamente en este escenario no matemático debido a la introducción del factor de carga de trabajo y prioridad.

9.2. Análisis de Viabilidad del Uso de Google Gemini como LLM

Se han realizado múltiples pruebas con el LLM de Gemini con el propósito de evaluar la precisión del análisis automático de imágenes. El experimento se ha realizado tanto con incidencias fotografiadas personalmente en la zona de Montcada i Reixach como con imágenes subidas a internet por usuarios de otras aplicaciones de participación ciudadana en otras partes de España. El propósito del experimento consiste en analizar la capacidad de entendimiento del LLM y el impacto en la clasificación de la incidencia generada en diversos escenarios donde el contexto puede ser más o menos evidente en base a una sola fotografía. Se han valorado 10 escenarios fácilmente identificables para el ser humano diversos en clasificación e impacto sobre la vía pública. Estos son los resultados de comparar los análisis hechos por Google AI con la interpretación humana:



Fig. 9: Diagrama Radial de Precisión de Gemini

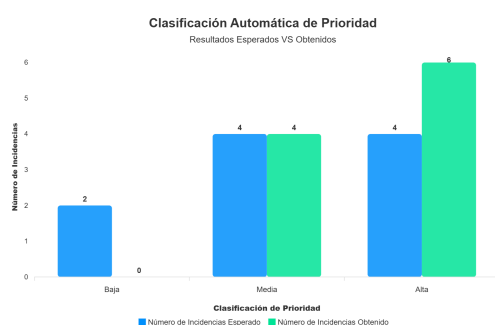


Fig. 10: Diagrama de Barras de Clasificación por Prioridad

Se encuentra que Google Gemini es perfectamente capaz de clasificar incidencias por tipo y, en la gran mayoría de casos, puede definir con exactitud el problema que se le presenta. En cuanto a la clasificación por prioridad, los resultados indican que Gemini tiene una mayor tendencia a catalogar más incidencias como prioritarias que un ser humano.

9.2.1. Casos de Prueba Destacables

En un único caso, Gemini ha sido incapaz de identificar adecuadamente la incidencia mostrada en la imagen. Se trata de una fotografía de una papelerita en la vía pública donde el cubo está desacoplado del soporte y situado en el suelo. La imagen comprende tanto el soporte como el cubo, sin embargo, el modelo de IA no ha sido capaz de hacer la conexión lógica entre ambos en ninguna ocasión. Es posible que este análisis erróneo también haya influido en la asignación de la prioridad. Por otro lado, otro caso similar, el de una farola fundida vista de noche, ha sido definido y clasificado correctamente todas las veces que se ha subido la imagen a la aplicación.

Cabe mencionar que, si bien es cierto que la clasificación de la gravedad del desperfecto puede ser algo subjetivo, en ninguno de los casos de prueba en este experimento ni a lo largo del desarrollo de la solución Gemini ha catalogado una incidencia en prioridad baja. Esto incluye un caso de prueba en el que el reporte a analizar se trataba de un grafiti en la persiana de un local, el cual Gemini catalogó como prioridad media.

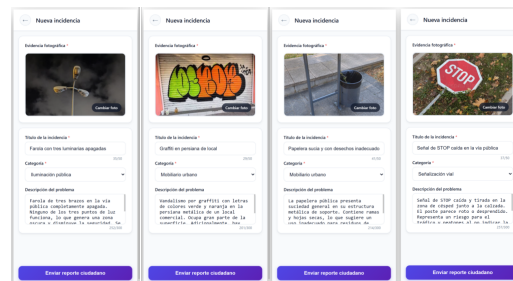


Fig. 11: Ejemplos de imágenes testeadas

9.3. Interpretación y Contextualización de los Resultados

Los resultados obtenidos en las pruebas unitarias y de integración muestran que el sistema no solo resuelve el problema del triaje urbano de forma aislada, sino que aplica principios ya presentes en modelos de gestión utilizados actualmente en la industria tecnológica.

La automatización completa del ciclo de vida de la incidencia, desde la clasificación visual mediante el modelo de Google Gemini hasta la asignación optimizada mediante la heurística de ponderación, sigue una filosofía similar a la utilizada en plataformas de logística inteligente y servicios bajo demanda. Del mismo modo que aplicaciones como Glovo o Uber optimizan recursos y reducen tiempos de coordinación mediante decisiones en tiempo real, este proyecto traslada esa misma lógica operativa al contexto de la administración pública y las Smart Cities.

La interpretación de estos resultados permite concluir que

la incorporación de inteligencia artificial y modelos de optimización matemática en la gestión municipal mejora la eficiencia en la asignación de recursos y proporciona a los ayuntamientos una capacidad de respuesta más ágil y proactiva. En consecuencia, el mantenimiento urbano deja de depender de procesos reactivos tradicionales y pasa a gestionarse desde una perspectiva logística optimizada.

10 CONCLUSIONES

Se ha demostrado la viabilidad técnica de la arquitectura desacoplada basada en Angular y el servidor de lógica de negocio con Node.js y Express. El sistema es capaz de conectar de forma asíncrona los reportes ciudadanos con la lógica del backend con tal de cumplir la base operativa del ciclo de vida de una incidencia.

La implementación de flujo de estados y notificaciones ha sido ejecutada. La máquina de estados es completamente operativa en el backend y en la base de datos a nivel relacional. El sistema cuenta también tanto con una bandeja de entrada de notificaciones persistentes como con un sistema de notificaciones push.

El objetivo de geolocalización se encuentra completamente alcanzado. La integración de la librería open-source Leaflet junto con la API cartográfica de OpenStreetMap ha permitido renderizar con éxito un mapa interactivo en tiempo real en la pantalla de inicio y en las herramientas para operarios. Se ha cumplido con éxito el núcleo algorítmico de la gestión operativa. El backend cuenta con una heurística funcional basada en un Modelo de Puntuación Ponderada que evalúa en tiempo real a los técnicos activos del municipio y proporciona todas las herramientas necesarias para la correcta gestión municipal.

10.1. Líneas de Trabajo Futuras

Pese a haber cumplido con los objetivos principales establecidos en el alcance inicial del proyecto, se pueden identificar diversas áreas de mejora y expansión.

- **Generación automática de informes por zonas geográficas:** Como se definió en la fase de ingeniería de requisitos y se reflejó en uno de los Diagramas de casos de uso del sistema (Fig. 14 en el apartado de Apéndice), se contempló inicialmente el desarrollo de un módulo de generación automática de informes históricos enfocado en el análisis de incidencias agrupadas por zonas geográficas. Sin embargo, durante la fase de modelado de datos, se optó por postergar su implementación. Esto se debe mayoritariamente a la complejidad en la delimitación manual que conllevaría. La introducción manual de polígonos geográficos para definir los límites de cada zona administrativa añade una carga de configuración inicial y un consumo de cómputo geométrico en servidor que excede las capacidades y los objetivos de rendimiento de este proyecto. En un futuro, se podrían investigar maneras de integrar servicios de geocodificación o indexación geoespacial en la aplicación con tal de no comprometer la flexibilidad de la plataforma.
- **Exploración de LLM alternativos:** En base a los resultados de las pruebas de viabilidad de Gemini en el

apartado 9.2, si bien es cierto que su análisis de objetos y categorías tiende a ser preciso, se ha observado que en ocasiones carece de la capacidad requerida para evaluar el contexto de la imagen. Adicionalmente, tiende a exagerar el nivel de urgencia de las incidencias, cosa que podría tener un impacto en la optimización logística de CityFix. Se podría estudiar la implementación de otros LLM y realizar un análisis comparativo con el objetivo de determinar el modelo óptimo para el contexto del proyecto.

- **Estrategias de escalabilidad y optimización de carga:** Actualmente el proyecto está diseñado con entornos pequeños y de baja carga de despliegue. Con el propósito de transformar el prototipo actual en una plataforma de nivel de producción, se proponen posibles iniciativas futuras: Escalabilidad horizontal de la capa de datos para no convertir la BD de Supabase en un cuello de botella y optimización cartográfica/carga diferida para renderizar los marcadores de manera más eficiente de cara a datos masivos.
- **Optimización del algoritmo de asignación automática de cara a una mayor concurrencia de incidencias:** Actualmente el modelo de ponderación puntuada está ajustado a escalas pequeñas donde se asume un volumen reducido de incidencias. De cara a la escalabilidad anteriormente mencionada, se podrían adoptar más rasgos de otros ADS comerciales, como las colas de peticiones.

USO DE LA IA GENERATIVA

Cabe mencionar que, a lo largo de este proyecto, se ha hecho uso de la IA generativa como herramienta de soporte en situaciones puntuales. En cuanto a la redacción del informe, se ha empleado de manera mínima, concretamente para estructurar ciertas secciones y profundizar en conceptos técnicos. De cara a la fase de desarrollo, se ha utilizado principalmente en dos situaciones. Primeramente, para identificar y comparar frameworks y librerías con tal de poder elegir eficientemente el recurso más adecuado para cada situación. En segundo lugar, se ha utilizado para solventar errores técnicos y de programación, particularmente aquellos difíciles de identificar manualmente o que no tuvieran una solución clara al consultar otras fuentes de información. Estos son algunos ejemplos concretos de situaciones donde se ha empleado la IA generativa:

- Errores de renderizado del mapa Leaflet y sus marcadores, concretamente en relación al resto de elementos en la vista.
- Optimización de la configuración de los servicios de Google AI con tal de obtener resultados adaptados a la lógica del proyecto.
- Fallos en los servicios de autenticación que, en ciertas ocasiones, eran incapaces de identificar algunos datos del usuario (como su rol), dependiendo de si estaban registrados con Google o nativamente.
- Apoyo durante la implementación de las notificaciones push en la aplicación, debido a múltiples defectos en el proceso de suscripción y envío.

- Explicación y comparación de posibles librerías a usar. Por ejemplo, la librería de reactividad empleada en el proyecto, Signal.

AGRADECIMIENTOS

En primer lugar quisiera agradecer a mi tutor, Sergio Bachiller Rubia, por sus valiosas indicaciones y orientación, que han sido esenciales para el avance del proyecto. También agradezco el apoyo de mis familiares y amigos, que han sido mis beta testers y una fuente fiable de feedback y sugerencias. Finalmente, quiero hacer una mención especial a Marc García Martín, quien ha estado dispuesto siempre a prestarme ayuda y ofrecerme su opinión durante el desarrollo del proyecto.

REFERENCIAS

- [1] Snap Send Solve Ltd., "Snap Send Solve - Enterprise Platform for Councils." [En línea]. Disponible: <https://www.snapendsolve.com/>. [Accedido: 22-feb-2026].
- [2] City of Seattle, "Find It, Fix It Mobile App – Seattle Customer Service Bureau." [En línea]. Disponible: <https://www.seattle.gov/customer-service-bureau/find-it-fix-it-mobile-app>. [Accedido: 22-feb-2026].
- [3] Decos Information Solutions, "Fixi - Software para la gestión de incidencias en el espacio público." [En línea]. Disponible: <https://decos.com/es/fixi>. [Accedido: 22-feb-2026].
- [4] Atlassian Inc., "¿Qué es Scrum? Metodología ágil explicada." Agile Coach de Atlassian. [En línea]. Disponible: <https://www.atlassian.com/es/agile/scrum>. [Accedido: 23-feb-2026].
- [5] Asana Inc., "¿Qué es Kanban? Principios y beneficios del método visual." Recursos de Asana. [En línea]. Disponible: <https://asana.com/es/resources/what-is-kanban>. [Accedido: 23-feb-2026].
- [6] Atlassian Inc., "Jira Software - Gestión de proyectos para equipos ágiles." [En línea]. Disponible: <https://www.atlassian.com/es/software/jira>. [Accedido: 23-feb-2026].
- [7] Atlassian Inc., "Atlassian - Software para equipos de desarrollo y colaboración." [En línea]. Disponible: <https://www.atlassian.com/>. [Accedido: 23-feb-2026].
- [8] Atlassian Inc., "Confluence - Espacio de trabajo compartido y gestión del conocimiento." [En línea]. Disponible: <https://www.atlassian.com/es/software/confluence>. [Accedido: 23-feb-2026].
- [9] GitHub Inc., "GitHub - Platform for software development and version control." [En línea]. Disponible: <https://github.com/>. [Accedido: 23-feb-2026].
- [10] ThePower Business School, "Modelo Vista Controlador (MVC): qué es y cómo aplicarlo." ThePowerTech Blog. [En línea]. Disponible: <https://thepower.education/blog/tech/modelo-vista-controlador-mvc>. [Accedido: 29-mar-2026].
- [11] Figma Inc., "Figma - La herramienta de diseño de interfaz colaborativa." [En línea]. Disponible: <https://www.figma.com/es-es/>. [Accedido: 29-mar-2026].
- [12] ESIC Business & Marketing School, "Angular: qué es, para qué sirve y ventajas." ESIC Rethink. [En línea]. Disponible: <https://www.esic.edu/rethink/tecnologia/angular-que-es-para-que-sirve-y-ventajas-c>. [Accedido: 16-may-2026].
- [13] V. Agafonkin, "Leaflet - an open-source JavaScript library for mobile-friendly interactive maps." LeafletJS. [En línea]. Disponible: <https://leafletjs.com/>. [Accedido: 16-may-2026].
- [14] Profile Software Services, "Angular Signals: Qué son y cómo cambian la gestión del estado." Profile Blog. [En línea]. Disponible: <https://profile.es/blog/angular-signals/>. [Accedido: 17-may-2026].
- [15] GreenSock Inc., "GSAP - GreenSock Animation Platform." GSAP. [En línea]. Disponible: <https://gsap.com/>. [Accedido: 17-may-2026].
- [16] Solbyte IT Solutions, "Node.js: Qué es, características y para qué sirve." Solbyte Blog. [En línea]. Disponible: <https://www.solbyte.com/blog/nodejs/>. [Accedido: 17-may-2026].
- [17] Google LLC, "Google AI Studio - Quickstart and API Reference." Google AI. [En línea]. Disponible: <https://aistudio.google.com/>. [Accedido: 20-may-2026].
- [18] Supabase Inc., "Supabase - The Open Source Firebase Alternative and Postgres Development Platform." [En línea]. Disponible: <https://supabase.com/>. [Accedido: 29-mar-2026].
- [19] Glovoapp S.L., "Algorithms - Glovo About." [En línea]. Disponible: <https://about.glovoapp.com/algorithms/>. [Accedido: 21-may-2026].
- [20] Uber Technologies Inc., "Über Marketplace Matching." [En línea]. Disponible: <https://www.uber.com/us/en/marketplace/matching/>. [Accedido: 21-may-2026].

APÉNDICE

A.1. Diagramas de casos de uso

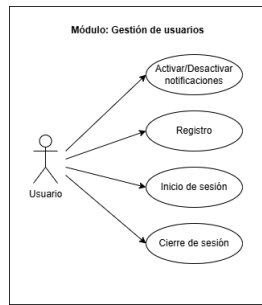


Fig. 12: Gestión de usuarios

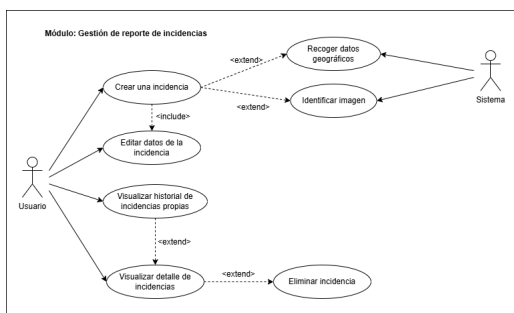


Fig. 13: Gestión de reporte de incidencias

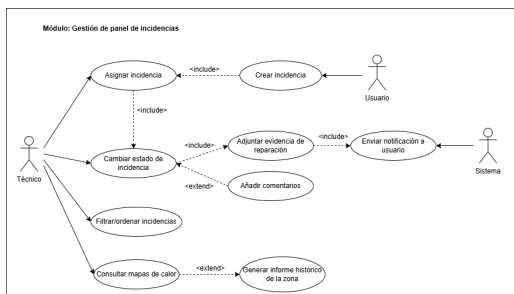


Fig. 14: Gestión del panel de incidencias

A.2. Boceto

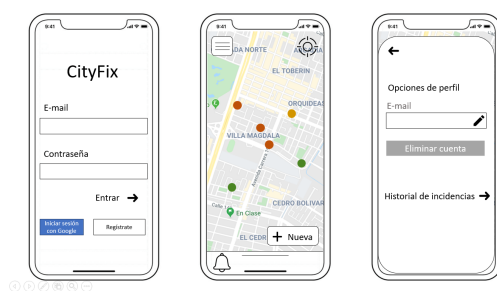


Fig. 15: Pantallas de inicio



Fig. 16: Pantallas de gestión de incidencias



Fig. 17: Pantallas de incidencia

A.3. Pantallas de la Aplicación

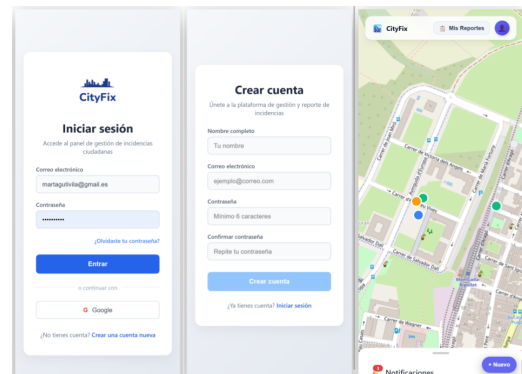


Fig. 18: Login, Registro y Página de inicio

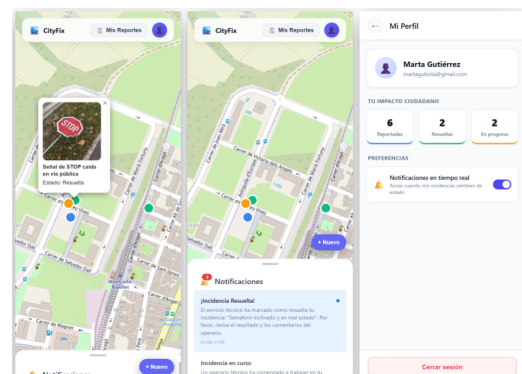


Fig. 19: Vista de marcadores, Bandeja de notificaciones y Perfil

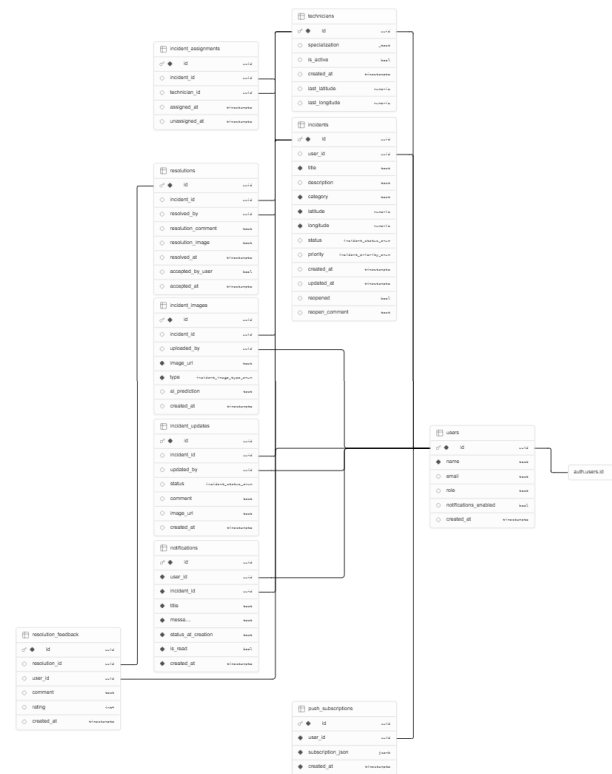
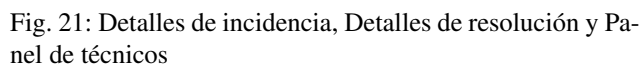


Fig. 23: Esquema de tablas en la base de datos



A.4. Esquema de Supabase

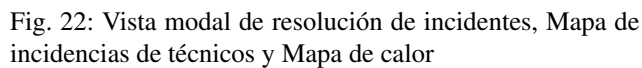


TABLA 1: REQUISITOS FUNCIONALES

Código	Requisito Funcional	Prioridad
RF01	El sistema debe permitir a los ciudadanos reportar incidencias, incluyendo la descripción del problema, la categoría y adjuntar evidencia visual.	Alta
RF02	El sistema debe extraer automáticamente la geolocalización de la incidencia.	Alta
RF03	El sistema debe asignar un identificador único a cada incidencia reportada para su seguimiento.	Alta
RF04	El sistema debe permitir a los técnicos visualizar el listado de incidencias y filtrarlas por estado, gravedad, ubicación o tipo.	Alta
RF05	El sistema debe implementar una máquina de estados para controlar el flujo de cada incidencia.	Alta
RF06	El sistema debe permitir a los técnicos cambiar el estado de una incidencia según su progreso.	Alta
RF07	El sistema debe permitir a los administradores priorizar incidencias según gravedad, impacto o urgencia.	Media
RF08	El sistema debe permitir adjuntar evidencias de resolución, como fotos de la reparación o comentarios adicionales.	Alta
RF09	El sistema debe enviar notificaciones automáticas a los usuarios cuando el estado de su incidencia cambie.	Alta
RF10	El sistema debe permitir a los usuarios consultar el historial y estado de sus incidencias desde la plataforma.	Alta
RF11	El sistema debe permitir a los usuarios reabrir una incidencia si están insatisfechos con la reparación del desperfecto.	Media
RF12	El sistema debe mostrar la ubicación exacta de cada incidencia en un mapa.	Alta
RF13	El sistema debe permitir la visualización de mapas de calor o clústeres para que los administradores identifiquen zonas con mayor número de incidencias.	Media
RF14	El sistema debe ofrecer un panel de control para técnicos y administradores con opciones de filtrado y visualización de incidencias.	Alta
RF15	El sistema debe generar reportes estadísticos sobre incidencias, tiempos de resolución y áreas más afectadas.	Media
RF16	El sistema debe poder rellenar automáticamente la información de una incidencia basándose únicamente en una fotografía.	Media
RF17	El sistema debe permitir a los usuarios entrar a la aplicación con su cuenta de Google.	Baja
RF18	El sistema debe permitir al usuario poder restaurar su contraseña en caso de haberla olvidado.	Baja
RF19	El sistema debe mostrar al usuario estadísticas de su aportación ciudadana en forma de incidencias reportadas para fomentar la participación.	Baja
RF20	El sistema debe permitir al usuario activar o desactivar el envío de notificaciones push.	Baja
RF21	El sistema debe permitir al usuario eliminar incidencias subidas por él mismo siempre que no estén en progreso.	Baja
RF22	El sistema debe permitir al usuario visualizar el título y la imagen de incidencias subidas por otros usuarios en el mapa interactivo.	Alta
RF23	El sistema debe contar con un algoritmo de asignación automática de incidencias.	Media

TABLA 2: REQUISITOS NO FUNCIONALES

Código	Requisito No Funcional	Prioridad
RNF1	El sistema debe poder gestionar al menos 100 incidencias simultáneamente sin degradar el rendimiento.	Media
RNF2	El tiempo de respuesta para mostrar una incidencia o actualizar su estado no debe superar los 2 segundos.	Media
RNF3	El sistema debe garantizar autenticación y autorización de usuarios y técnicos.	Alta
RNF4	Los datos personales de los ciudadanos deben cumplir con la normativa de protección de datos.	Alta
RNF5	Las fotos y evidencias deben ser almacenadas de forma segura y con acceso controlado.	Media
RNF6	La interfaz para ciudadanos debe ser intuitiva, simple y accesible desde móviles.	Alta
RNF7	La interfaz de administración debe permitir a los técnicos gestionar incidencias de manera rápida y eficiente.	Alta
RNF8	La arquitectura del sistema debe permitir agregar nuevas funcionalidades sin reestructurar todo el sistema.	Baja