

Desarrollo App Web GIS para EIEL.

**Encuesta de Infraestructura y
Equipamientos Locales**



**Juan Ramón Gavilanes
Sánchez**

TFG Ingeniería informática

**Servicios basados en localización y
espacios inteligentes.**

Nombre Tutor/a de TF
Joaquín Torres Sospedra

**Profesor/a responsable de
la asignatura**
Antoni Pérez Navarro

**Universitat Oberta
de Catalunya**

08/04/2024



Esta obra está sujeta a una licencia de
Reconocimiento-NoComercial-SinObraDerivad
a [3.0 España de Creative Commons](#)

FICHA DEL TRABAJO FINAL

Título del trabajo:	<i>Desarrollo de aplicación Web GIS para ejecutar la Encuesta de Infraestructura y Equipamiento Local (EIEL) de la comunidad de Madrid.</i>
Nombre del autor:	<i>Juan Ramón Gavilanes Sánchez</i>
Nombre del consultor/a:	<i>Joaquín Torres Sospedra</i>
Nombre del PRA:	<i>Antoni Pérez Navarro</i>
Fecha de entrega (mm/aaaa):	<i>07/2024</i>
Titulación o programa:	<i>Grado en ingeniería informática</i>
Área del Trabajo Final:	<i>Servicios basados en localización y espacios inteligentes</i>
Idioma del trabajo:	<i>Castellano</i>
Palabras clave	<i>GIS, EIEL, Movilidad.</i>

Resumen del Trabajo

El presente trabajo se centra en el desarrollo de una Aplicación Web GIS (Sistema de Información Geográfica) que asista a la recopilación de datos de la encuesta para la infraestructura y equipamiento locales (EIEL) de la comunidad de Madrid.

La metodología de desarrollo implica el uso de tecnologías modernas tanto en el frontend como en el backend, utilizando un enfoque de escalabilidad y rapidez de entrega mediante el uso de contenedores Linux y técnicas CICD.

Las funcionalidades clave incluyen la recolección de datos geo-referenciados en campo mediante dispositivos móviles, integración con el modelo de datos oficial, y exportación final a formato de geodatabases compatibles con herramientas GIS de escritorio.

Los beneficios esperados es incrementar la eficiencia en la recogida de los datos como mínimo un 75% respecto al ejercicio anterior, y poder finalizar la encuesta antes diciembre del 2024 dando soporte a 179 municipios y 10 operarios en campo.

Abstract

This work focuses on the development of a Web GIS Application to assist in the collection of data for the Local Infrastructure and Equipment Survey (EIEL) of the Community of Madrid. The development methodology involves the use of modern technologies in both the frontend and backend, employing a scalability and rapid delivery approach using Linux containers and CI/CD techniques.

Key functionalities include the collection of geo-referenced data in the field via mobile devices, integration with the official data model, and final export to geodatabase formats compatible with desktop GIS tools.

Expected benefits include increasing data collection efficiency by at least 75% compared to the previous exercise, and being able to complete the survey by December 2024, supporting 179 municipalities and 10 field operators.

Índice

Encuesta de Infraestructura y Equipamientos Locales.....	1
1. Introducción.....	1
1.1. Contexto y justificación del Trabajo.....	1
1.2. Objetivos del Trabajo.....	2
1.3. Impacto en sostenibilidad, ético-social y de diversidad.....	3
1.4. Enfoque y método seguido.....	4
1.5. Planificación del Trabajo.....	6
1.6. Breve sumario de productos obtenidos.....	9
1.7. Breve descripción de los otros capítulos de la memoria.....	10
2. Estado del arte.....	11
2.1. Levantamiento de requerimientos.....	14
2.2. Diseño de la solución.....	15
2.2.1. Aplicación Web.....	15
2.2.2. Aplicación Web Progresiva o Progressive Web App (PWA).....	19
2.2.3. Sistemas de información geográfica (SIG).....	20
2.2.4. MapLibre GL.....	21
2.2.5. Tegola tileserver.....	22
2.2.6. Contenedores Linux con Docker.....	24
2.2.7. Servidor Privado Virtual (VPS).....	25
2.2.8. GitHub y GitHub Actions.....	27
2.2.9. Propuesta arquitectónica del sistema.....	28
3. Implementación.....	29
3.1. Preparar modelo de datos.....	30
3.1. Orígenes de datos.....	30
3.1.1. Modelo de datos oficial.....	30
3.1.2. Nomenclátor oficial de la Comunidad de Madrid.....	31
3.2. Desarrollo Backend.....	33
3.2.1.- Configurar entorno de desarrollo.....	33
3.2.2.- API Rest.....	38
3.2.2.1. Módulo de acceso a base de datos.....	38
3.2.2.2. Módulo de seguridad.....	40
3.2.2.3. Endpoints auxiliares.....	44
3.2.2.4. Endpoints de equipamientos.....	47
3.3. Desarrollo Frontend.....	57
3.3.1. Configurar entorno de desarrollo.....	58
3.3.2. Interfaz UX/UI.....	60
3.3.3. Vistas y formularios.....	65
3.4. Pruebas manuales.....	73
3.5. Crear infraestructura.....	74
4. Glosario.....	75
5. Bibliografía.....	76

Listado de figuras

Figura 1: Visor Geo-EIEL. Fuente: [1].....	1
Figura 2: Metodología en cascada. Fuente: [2].....	5
Figura 3: Metodologías ágiles. Fuente: [3].....	5
Figura 4: Diagrama de Gantt con planificación del proyecto.....	8
Figura 5: Convocatoria anunciada.....	11
Figura 6: Formulario de aplicación gvSIG-EIEL.....	12
Figura 7: Ejemplo de formulario QField para QGis.....	13
Figura 8: Arquitectura básica de una aplicación web. Fuente [11].....	15
Figura 9: Impacto de las PWA en el desarrollo de las aplicaciones web. Fuente [12]..	19
Figura 10: Definición básica de un SIG. Fuente [13].....	20
Figura 11: Imagen del nomenclátor oficial de la Comunidad de Madrid.....	23
Figura 12: Logo Docker. Fuente [14].....	24
Figura 13: Arquitectura del sistema.....	28
Figura 14: Repositorio en GitHub.....	29

1. Introducción

1.1. Contexto y justificación del Trabajo

La necesidad que se busca cubrir es la mejora en la recopilación de datos de espacios públicos en la Comunidad de Madrid

Esto es relevante porque una recopilación eficiente y precisa de estos datos es fundamental para la planificación urbana, el desarrollo sostenible y la toma de decisiones informadas por parte de las autoridades locales.

En el momento de comenzar el trabajo, el problema radica en la falta de una solución tecnológica sencilla y adecuada que permita una recolección de datos georreferenciados eficiente y una integración fluida con el modelo de datos oficial. El ministerio ya proporciona una herramienta web similar de consulta, pero resulta ser altamente compleja y carece de un módulo de toma de datos móvil para los operarios que actúan sobre el terreno.

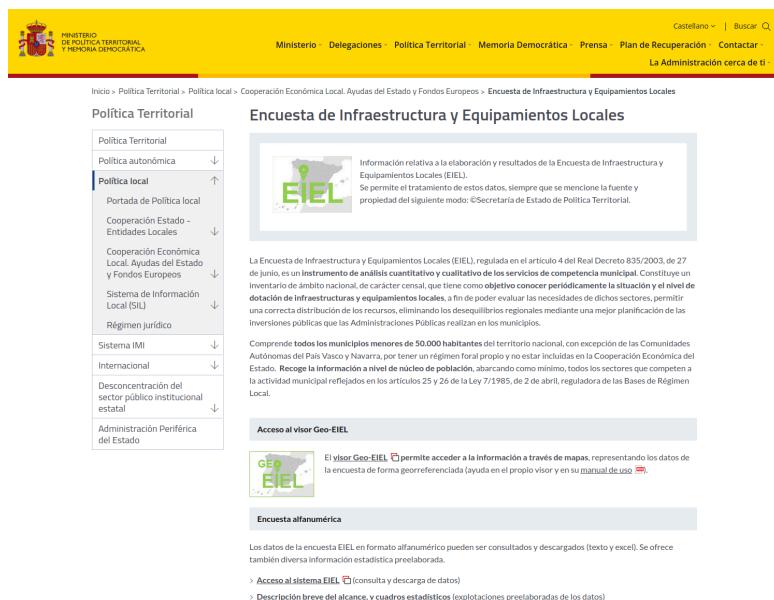


Figura 1: Visor Geo-EIEL. Fuente: [1]

La contribución principal del trabajo es el desarrollo de una Aplicación Web GIS (Sistema de Información Geográfica) que permita la recolección de datos en campo de manera precisa y eficiente utilizando dispositivos móviles. Los datos capturados deben integrarse con el modelo de datos oficial y exportarse a formatos compatibles con herramientas GIS.

Se busca obtener una solución tecnológica completa que mejore significativamente el proceso de recopilación y gestión de datos espaciales para el proyecto EIEL en la Comunidad de Madrid.

Recientemente he comenzado a trabajar en una consultora especializada en este tipo de aplicaciones, y se trata de un proyecto real. Por lo que el proyecto no solo me brinda la oportunidad de aprender y adquirir experiencia práctica en el desarrollo de aplicaciones GIS, sino que también me permite colaborar con expertos en el campo y contribuir al éxito de proyectos reales que tienen un impacto tangible en el mundo real. Esta combinación de aprendizaje, experiencia práctica y contribución a proyectos significativos es lo que hace que este proyecto sea especialmente emocionante y relevante para mi trayectoria profesional.

1.2. Objetivos del Trabajo

El objetivo final del trabajo es el **desarrollo de una aplicación Web GIS** que facilite la recopilación de datos de espacios públicos en la Comunidad de Madrid, y su posterior exportación al modelo definido por el Ministerio de política territorial y memoria democrática.

Objetivos generales

- Desarrollar un proyecto de información geográfica.
- Investigar el estado del arte de proyectos similares ya existentes.
- Documentar el trabajo mediante una memoria.
- Realizar una presentación que resuma las ideas clave del proyecto.
- Defensa del proyecto.

Objetivos específicos.

- Desarrollar una Aplicación Web GIS para la recopilación de datos de espacios públicos en la Comunidad de Madrid.
- Implementar funcionalidades que permitan la recolección eficiente de datos geo-referenciados en campo mediante dispositivos móviles.
- Integrar el modelo de datos oficial del proyecto EIEL para garantizar la consistencia y precisión de la información recopilada.
- Facilitar la revisión y validación de los datos por parte de los ayuntamientos y otros organismos involucrados en el proyecto.
- Exportar los datos recopilados a geodatabases compatibles con herramientas GIS de escritorio para su análisis y utilización posterior.
- Desplegar la aplicación en un entorno de producción en la nube utilizando tecnologías modernas y eficientes.
- Garantizar la seguridad de la aplicación mediante la implementación de protocolos de encriptación y autenticación de datos.

1.3. Impacto en sostenibilidad, ético-social y de diversidad

El trabajo se desarrollará en concordancia con los principios del desarrollo sostenible establecidos en la Agenda 2030 de las Naciones Unidas [4], concretamente:

- Dimensión Ética (ODS 16 - Paz, justicia e instituciones sólidas):
 - Impacto Positivo: El TFG promueve la ética al asegurar la privacidad y confidencialidad de los datos recolectados, contribuyendo así a una sociedad más justa y transparente.
 - Impacto Negativo: Si hay violaciones éticas, como el uso indebido de datos o la falta de transparencia en el proceso, podría socavar la confianza en las instituciones y afectar la paz y la justicia.
- Dimensión Global (ODS 11 - Ciudades y comunidades sostenibles):
 - Impacto Positivo: El TFG tendrá un impacto global positivo al mejorar la gestión de la infraestructura local, contribuyendo así a comunidades más sostenibles y resilientes.
 - Impacto Negativo: Si el proyecto no considera los impactos globales, como la huella ambiental o las necesidades de comunidades marginadas, podría perpetuar desigualdades y contribuir a la degradación ambiental.
- Dimensión de Compromiso (ODS 17 - Alianzas para lograr los objetivos):
 - Impacto Positivo: El TFG podría fortalecer el compromiso al colaborar con diversas partes interesadas, como instituciones académicas, empresas y comunidades locales, para abordar desafíos comunes.
 - Impacto Negativo: Si no se establecen alianzas efectivas o si hay falta de participación de las partes interesadas, podría obstaculizar el progreso hacia los objetivos comunes y limitar el impacto del proyecto.

1.4. Enfoque y método seguido

Existen varias estrategias para llevar a cabo un desarrollo de estas características, como por ejemplo:

1. Desarrollar un producto nuevo desde cero: Esta estrategia implica diseñar y construir la aplicación desde el inicio, aprovechando las tecnologías más actuales y adaptándolas a las necesidades específicas del proyecto.
2. Adaptar un producto existente: Esta estrategia implica tomar una aplicación web GIS existente y modificarla para cumplir con los requisitos del proyecto. Esto puede implicar ajustes en la funcionalidad, el diseño y la integración con otras tecnologías.

La estrategia elegida para este trabajo es desarrollar un producto nuevo desde cero, ya que se necesita un alto grado de **personalización** en la captura de la información para estar alineados con el modelo de datos requerido.

Además, tendremos una mayor **flexibilidad** de poder utilizar las tecnologías más actuales sin tener que depender de antiguos desarrollos.

También dispondremos del **control total** del proceso de desarrollo, lo que nos permite controlar la calidad del código, la seguridad de la aplicación y la eficiencia del rendimiento.

Durante las primeras fases del proyecto, de **inicialización y planificación**, se utilizará el **método en Cascada** o Waterfall. Esta decisión se basa en la necesidad de establecer una estructura clara y detallada para definir los requisitos, alcance y recursos del proyecto antes de avanzar al desarrollo. Este enfoque permite una planificación exhaustiva y una definición precisa de los entregables, lo que es fundamental para el éxito del proyecto, especialmente en sus etapas iniciales.

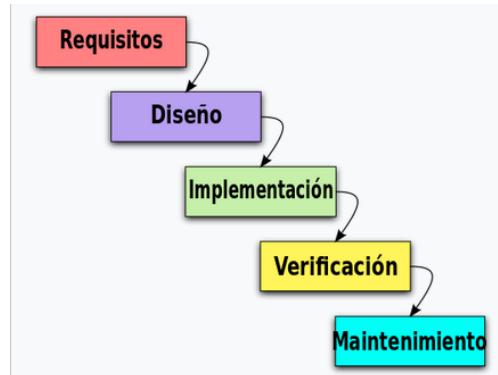


Figura 2: Metodología en cascada. Fuente: [2]

Una vez completadas las fases de inicialización y planificación, la **fase de desarrollo** se llevará a cabo utilizando **metodologías ágiles**. Este cambio de enfoque se debe a la necesidad de adaptabilidad y flexibilidad durante la fase de implementación, donde pueden surgir cambios y ajustes en los requisitos del proyecto. Las metodologías ágiles, como Scrum o Kanban, permiten entregas incrementales y regulares, lo que facilita la adaptación a las necesidades cambiantes del proyecto y proporciona una mayor transparencia y colaboración entre el equipo de desarrollo y los interesados.



Figura 3: Metodologías ágiles. Fuente: [3]

La combinación de metodologías en cascada y ágiles aprovecha las fortalezas de ambos enfoques: la planificación detallada y la estructura del Waterfall para las etapas iniciales, y la flexibilidad y adaptabilidad de las metodologías ágiles para la fase de desarrollo, lo que resulta en un proceso de desarrollo más eficiente y efectivo.

1.5. Planificación del Trabajo

A continuación se detallan los recursos necesarios para la realización del proyecto:

Software

- **GitHub**: Repositorio de código fuente en la nube.
- **Visual Studio Code**: IDE generalista usado para el frontend.
- **Python**: Lenguaje de programación con una versión igual o superior a la 3.10 (LTS)
- **NodeJS**: Lenguaje de programación con una versión superior a la 20.11 (LTS)
- **Pycharm**: IDE de python usado para el backend.
- **Docker**: Tecnología de contenedores para empaquetar y distribuir aplicaciones
- **Postgres/Postgis**: SGBD configurado con extensión espacial.
- **Tegola**: Servidor de datos geoespaciales en forma de teselas de mapa
- **QGis**: Software GIS para visualizar y editar datos espaciales.
- **Almacén de objetos S3**: Para el alojamiento de las imágenes capturadas por los clientes.
- **Google docs**: Para realizar esta memoria.
- **YouTrack**: Herramienta de gestión de proyectos

Hardware

- **Ordenador** con CPU AMD Ryzen 7 5800H, 16 GB RAM DDR4 y Linux Mint 21.3
- **Servidor privado virtual (VPS)** con 1 GB RAM/ 1 Intel vCPU / 35 GB disco/ Ubuntu 22.04 (LTS) x64

El trabajo de desarrollará en cinco fases que se detallan a continuación junto con las tareas que han sido identificadas para cada fase:

Fase 1: Definición y planificación del trabajo final (28/2 al 12/3)

- Determinar la temática del trabajo.
- Describir de manera concisa en qué consistirá el trabajo.
- Explicar razonadamente la motivación y justificación del trabajo elegido
- Definir los objetivos del trabajo.
- Determinar la metodología que se seguirá.
- Realizar una planificación para el desarrollo del trabajo.

Fase 2: Análisis del mercado o Estado del arte (13/3 al 9/4)

- Buscar artículos, investigaciones que intenten resolver la misma problemática.
- Buscar y seleccionar el software que se utilizará.
- Elegir las fuentes de datos que se utilizarán.
- Refinar los objetivos parciales definidos en la actividad anterior.

Fase 3: Implementación (10/4 al 14/5)

- Preparar Modelo de datos.
- Desarrollo Backend.
 - Rest Api.
 - Pruebas manuales.
- Desarrollo PWA.
 - Diseño Mockup.
 - Desarrollo pantallas.
- Pruebas manuales.
- Crear infraestructura.

Fase 4: Entrega final (15/5 al 18/6)

- Repositorio de Github con el código del proyecto.
- Memoria del trabajo final.

Fase 5: Presentación (19/6 al 25/6)

- Video presentación.

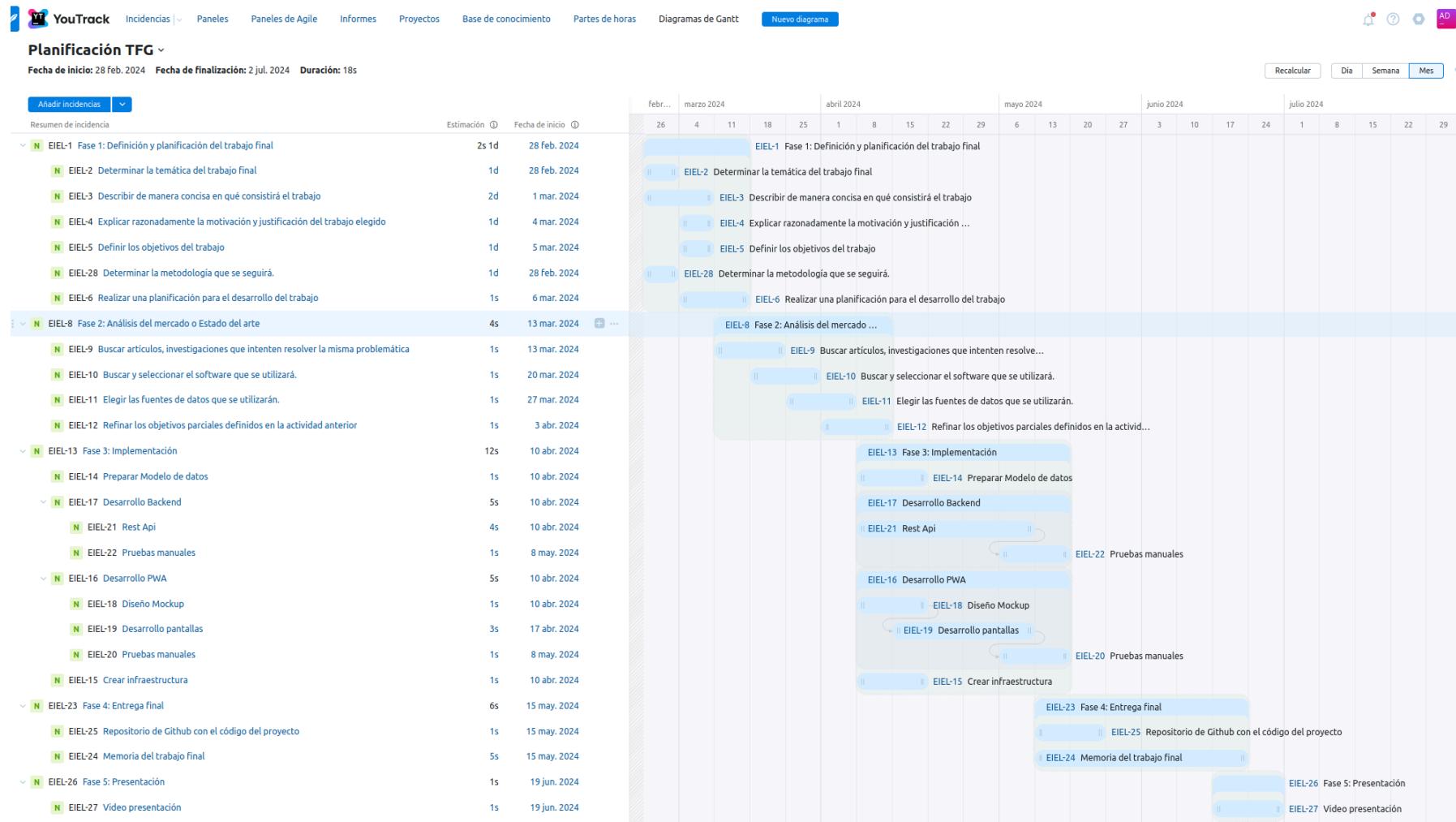


Figura 4: Diagrama de Gantt con planificación del proyecto

1.6. Breve sumario de productos obtenidos

Estos serán los entregables generados tras finalizar el proyecto:

1. Aplicación Móvil PWA diseñada para los operadores encargados de recopilar datos sobre el terreno de manera ágil y eficiente.
2. Aplicación Web de Escritorio que permitirá analizar los datos recolectados desde la aplicación móvil, además de ofrecer la posibilidad de introducir puntos de interés de forma manual.
3. Backend alojado en la nube, encargado de almacenar y gestionar la información recopilada, garantizando un acceso seguro y eficaz para los usuarios.
4. Memoria del proyecto en la que se incluirán tanto detalles técnicos del flujo de trabajo empleado para el desarrollo de la aplicación, así como información básica sobre el dominio de los GIS.

1.7. Breve descripción de los otros capítulos de la memoria

Los próximos capítulos se centrarán en ofrecer un análisis exhaustivo de cada fase del proyecto, mientras que también se proporcionará una visión general de la estructura y contenido de la memoria en su conjunto.

Continuaremos por la Fase 2: Análisis del mercado o Estado del arte. Aquí, examinaremos en profundidad el contexto y las tendencias del mercado relevantes para nuestro proyecto, así como el estado actual de la tecnología o investigación en el área.

Finalmente, abordaremos la Fase 3: Implementación, donde detallaremos el proceso de llevar a cabo nuestro proyecto, desde la planificación hasta la ejecución práctica. Esto incluirá la descripción de los métodos y herramientas utilizados, así como los desafíos encontrados y las soluciones implementadas durante esta etapa.

2. Estado del arte.

Como indica en su propia web [1], "la Encuesta de Infraestructura y Equipamientos Locales (EIEL) es un **instrumento de análisis cuantitativo y cualitativo de los servicios de competencia municipal**".

Constituye un inventario de ámbito nacional, de carácter censal, que tiene como objetivo conocer periódicamente la situación y el nivel de dotación de infraestructuras y equipamientos locales, a fin de poder evaluar las necesidades de dichos sectores, permitir una correcta distribución de los recursos, eliminando los desequilibrios regionales mediante una mejor planificación de las inversiones públicas que las Administraciones Públicas realizan en los municipios."

El **objetivo** de este proyecto es asistir a la captura de datos de los **equipamientos de la encuesta (Lote 3)**, quedando fuera del alcance el resto de los recursos como carreteras, viarios y alumbrado público (Lote 1), abastecimiento de agua y saneamiento (Lote 2)

The screenshot shows the official website for public procurement in Madrid. The main menu includes sections for 'PERFIL DE CONTRATANTE', 'CONTRATACIÓN CENTRALIZADA', 'SERVICIOS Y CONSULTAS', 'INFORMACIÓN GENERAL', and 'AVISOS Y NOVEDADES'. The 'CONTRATACIÓN CENTRALIZADA' section is active. A sidebar on the left lists various documents and information. The central content area is titled 'División en lotes' and specifies three lots: Lote 1 (carreteras, viarios y alumbrado público), Lote 2 (abastecimiento de agua y saneamiento), and Lote 3 (equipamientos).

Figura 5: Convocatoria anunciada

Después de llevar a cabo una exhaustiva revisión de la literatura con el propósito de identificar trabajos previos, proyectos y artículos científicos relacionados [6][7][8][10], hemos observado que aunque existen diversas herramientas concebidas para facilitar la explotación de la información obtenida para la EIEL, presentan un alto nivel de complejidad para nuestro público objetivo, dado que se enfocan en la captura de información de todos los lotes de la encuesta y además, se centran en la captura de datos mediante formularios diseñados para su utilización en equipos de escritorio.



CartoLAB
Laboratorio de Ingeniería Cartográfica
ETS Ingeniería de Caminos, Canales y Puertos
Campus de Elviña - 15071 A Coruña (Spain)
Tel: (+34) 981 167 000 ext 5493
<http://cartolab.udc.es> - cartolab@udc.es

9.5. FORMULARIOS EIEL DE EQUIPAMIENTOS

Núcleo Población	GeId:	Clave:
Fase: 2010	GeId: CC	Clave: CC
Provincia: 36	Orden casa:	Orden casa: Casa do Concello de Padrón
Municipio: municipio	Denominación:	Denominación: Casa do Concello
Entidad:	Tipo instalación:	Tipo instalación: Municipal
Núcleo:	Titular:	Titular: Propiedad municipal
Denominación:	Tenencia:	Tenencia: Bueno
	Estado:	Estado: 560
	Superficie cubierta:	Superficie cubierta: 0
	Superficie aire libre:	Superficie aire libre: 275
	Superficie solar:	Superficie solar: NO
	Acceso con silla de ruedas:	Acceso con silla de ruedas: NO

Figura 6: Formulario de aplicación gvSIG-EIEL

En este sentido, hemos identificado una carencia crucial en estas herramientas: la ausencia de un **módulo que simplifique la recopilación de datos directamente desde el terreno, utilizando dispositivos móviles**, lo que permitiría la captura de fotografías georreferenciadas, así como la capacidad de sincronización en tiempo real con la base de datos central.

Este vacío es precisamente donde nuestro proyecto se destacará, al ofrecer una solución integral que aborda estas limitaciones y proporciona una experiencia de usuario fluida y eficiente.

También hemos evaluado la posibilidad de utilizar plugins como QField para QGIS; sin embargo, esta opción ha sido descartada debido a diversas limitaciones.

En primer lugar, QGIS presenta una compatibilidad limitada con dispositivos móviles, especialmente con iOS, lo que restringe su aplicación en entornos donde se requiera dicha movilidad.

Además, su dependencia de la infraestructura de QGIS puede generar complicaciones adicionales, especialmente en términos de sincronización de datos y compatibilidad entre versiones.

Por otro lado, la gestión del elevado número de formularios requeridos para el proyecto y la falta de flexibilidad en la personalización de los datos a capturar, representan obstáculos significativos que hacen que esta opción no sea adecuada para nuestras necesidades específicas.

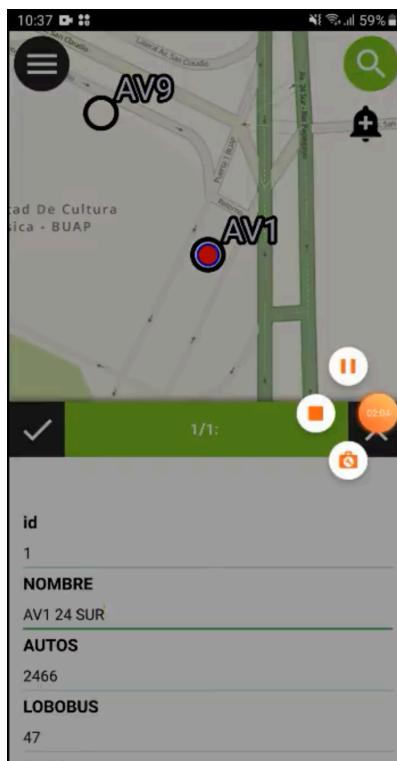


Figura 7: Ejemplo de formulario QField para QGis

2.1. Levantamiento de requerimientos

Como medida inicial realizamos el levantamiento de requerimientos necesarios para la construcción del WEBGIS, durante tres jornadas de reuniones establecidas con el cliente.

A partir de las necesidades detectadas estos fueron los requerimientos obtenidos a muy alto nivel:

- 3 Roles distintos:
 - operario (App Movil: Cumplimentar encuesta)
 - administrador (App Escritorio: administrar usuarios, informes de evolución, etc.)
 - ayuntamiento (App Escritorio: solo consulta y edita los equipamientos de sus municipios)
- Control de acceso de usuarios
 - Login de acceso
 - Registro de actividad de los usuarios
- App Móvil.
 - Cumplimentar formulario de encuesta para cada tipo de equipamiento acorde al modelo de datos solicitado:
 - Casa Consistorial
 - Cementerio
 - Centro Asistencial
 - Centro Enseñanza
 - Centro Sanitario
 - Centro Cultural
 - Edificio público sin uso
 - Instalación deportiva
 - Lonja o mercado o feria
 - Matadero
 - Parque
 - Tanatorio
 - Protección civil
 - Vertedero
 - Subir fotografías georeferenciadas tomada desde el equipamiento encuestado
 - Centrar marcador en la posición actual del dispositivo
 - Buscador de calles
 - Capas auxiliares
 - Open Street Map
 - Ortofotografía
 - Capas del nomenclátor de la Comunidad de Madrid:
 - Municipios, núcleos urbanos, callejero, mercados, parques, residuos, centros sanitarios, etc.

- App Escritorio.
 - Gestión usuarios
 - Alta, Modificación y Baja de usuarios.
 - Informes
 - Evolución por municipios
 - Exportar datos registrados a Shapefile compatible con el modelo de datos solicitado.

2.2. Diseño de la solución

Comenzamos nuestra investigación de herramientas tecnológicas con base en las necesidades identificadas para cumplir con los objetivos del proyecto.

Tras concluir la investigación, hemos seleccionado las siguientes herramientas tecnológicas para su utilización para la construcción del proyecto.

2.2.1. Aplicación Web

Una aplicación web es un software diseñado para ser utilizado a través de un navegador web. En lugar de descargar e instalar la aplicación en un dispositivo, los usuarios pueden acceder a ella simplemente ingresando una dirección URL en su navegador.

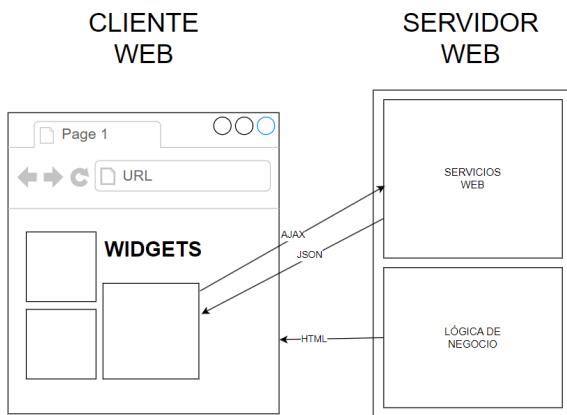


Figura 8: Arquitectura básica de una aplicación web. Fuente [11]

Las aplicaciones web pueden ofrecer una amplia variedad de funcionalidades, desde simples sitios web interactivos hasta complejas plataformas de software como servicios (SaaS) que permiten realizar tareas complejas como procesamiento de datos, gestión de proyectos, redes sociales, compras en línea, entre otras. Estas aplicaciones suelen estar alojadas en servidores remotos y se accede a ellas a través de Internet.

El uso de aplicaciones web implica la interacción entre dos partes, el cliente y el servidor web, a través del protocolo de comunicación HTTP. Se considera comúnmente que estas aplicaciones tienen una estructura compuesta por tres capas.

- **La capa del navegador (frontend)**, responsable de la interacción con los usuarios. Esta capa está formada por software o scripts que pueden ser interpretados por los navegadores web (HTML, CSS, JavaScript), facilitando la comunicación entre el usuario y el servidor web.

En el desarrollo frontend, utilizaremos **Vue.js** (<https://vuejs.org/>), un framework de JavaScript progresivo y de código abierto utilizado para construir interfaces de usuario (UI) interactivas y dinámicas. Se centra en la creación de aplicaciones de una sola página (Single Page Applications, SPA) y en el desarrollo de componentes reutilizables.

Algunas de las ventajas que ofrece Vue.js en el desarrollo del frontend incluyen:

- **Reactividad**: utiliza un sistema de reactividad que permite que los componentes de la interfaz de usuario reaccionen de manera automática a los cambios en los datos. Esto simplifica la gestión del estado de la aplicación y facilita la creación de interfaces dinámicas y fluidas.
 - **Componentes reutilizables**: promueve el desarrollo basado en componentes, lo que permite dividir la interfaz de usuario en piezas más pequeñas y reutilizables. Estos componentes pueden encapsular su propio estado y comportamiento, lo que facilita la modularidad y el mantenimiento del código.
 - **Flexibilidad**: Vue.js es altamente adaptable y se puede integrar fácilmente con otros proyectos y tecnologías. Lo utilizaremos junto con **Tailwind CSS** (<https://tailwindcss.com/>), que es un framework de CSS de utilidad que facilita la creación de interfaces de usuario personalizadas y con un diseño rápido y eficiente.
 - **Rendimiento optimizado**: Vue.js está diseñado para ser liviano y eficiente en términos de rendimiento. Su tamaño reducido y su sistema de compilación optimizado garantizan tiempos de carga rápidos y una experiencia de usuario fluida.
- **La capa del servidor (backend)**, es fundamental en la ejecución de una aplicación web, ya que se encarga de una variedad de tareas críticas como:
 - Implementar la lógica de negocio, que determina cómo se procesan y manipulan los datos recibidos del cliente para llevar a

cabo las funciones específicas de la aplicación.

- La gestión de la seguridad de acceso, asegurando que solo los usuarios autorizados puedan acceder a los recursos y funcionalidades de la aplicación.
- El acceso a la persistencia de datos, lo que implica la comunicación con la base de datos para almacenar y recuperar información de manera eficiente y segura.
- Interacción con otros servicios externos, como APIs de terceros o sistemas internos, para obtener datos adicionales o integrar funcionalidades externas en la aplicación web.
- En nuestro caso para el desarrollo en la capa del servidor, utilizaremos **Python** con el **framework FastAPI** (<https://fastapi.tiangolo.com/>). FastAPI es un framework moderno y de alto rendimiento para la creación de API REST con Python.
- Algunas de las ventajas que proporciona FastAPI en este contexto incluyen:
 - **Rendimiento:** FastAPI está diseñado para ser rápido y eficiente, lo que permite manejar grandes volúmenes de solicitudes de manera ágil y escalable.
 - **Tipado estático:** Utiliza el sistema de tipado estático de Python (mediante Pydantic), lo que proporciona un autocompletado más preciso en los editores de código y ayuda a detectar errores en tiempo de desarrollo.
 - **Documentación automática:** FastAPI genera automáticamente una documentación interactiva (Swagger UI) para la API basada en los tipos de datos y las anotaciones de tipo, lo que facilita su comprensión y uso por parte de otros desarrolladores.
 - **Soporte para asincronía:** Permite el uso de código asíncrono para manejar de manera eficiente operaciones de entrada/salida intensivas, lo que puede mejorar significativamente el rendimiento de la aplicación en entornos de alta concurrencia.
 - **Fácil integración con bases de datos:** Se integra sin problemas con diversas bases de datos, lo que permite acceder y manipular datos de manera eficiente y segura.
 - **Soporte para OpenAPI y GraphQL:** Es compatible con OpenAPI y GraphQL, lo que facilita la exposición de APIs

RESTful y basadas en consultas.

- **Capa de persistencia:** Es la capa en donde se almacenan los datos generados por la aplicación web. Utilizaremos **PostgreSQL** (<https://www.postgresql.org/>), como sistema de gestión de bases de datos relacional de código abierto. Es altamente compatible con SQL y ofrece numerosas características avanzadas que lo convierten en una opción popular para una amplia variedad de aplicaciones, desde pequeñas hasta grandes empresas.

Algunas de sus características y ventajas incluyen:

- **Fiabilidad y robustez:** PostgreSQL es conocido por su confiabilidad y estabilidad. Ha sido probado en aplicaciones críticas de misión durante muchos años y es utilizado por organizaciones de todo el mundo.
- **Amplia compatibilidad:** Es compatible con muchas características de SQL estándar y también ofrece extensiones y características propietarias que lo hacen aún más versátil y útil para una variedad de casos de uso.
- **Escalabilidad:** Es altamente escalable y puede manejar grandes volúmenes de datos y cargas de trabajo exigentes. Ofrece soporte para replicación, particionamiento de tablas y clústeres para distribuir la carga y mejorar el rendimiento.
- **Soporte para datos geoespaciales con PostGIS:** PostGIS (<https://postgis.net/>), es una extensión de PostgreSQL que agrega soporte para datos geoespaciales y operaciones geoespaciales avanzadas. Permite el almacenamiento y la consulta de datos geográficos, como puntos, líneas y polígonos, así como operaciones como la búsqueda de intersecciones, la medición de distancias y la creación de mapas interactivos.
- **Comunidad activa y soporte:** PostgreSQL cuenta con una comunidad de desarrolladores activa y un equipo de soporte dedicado que proporciona actualizaciones regulares, parches de seguridad y documentación detallada.
- **Costo efectivo:** Al ser de código abierto, es una opción rentable para empresas y organizaciones que buscan una solución de base de datos potente y confiable sin incurrir en costos de licencia.

2.2.2. Aplicación Web Progresiva o Progressive Web App (PWA)



Figura 9: Impacto de las PWA en el desarrollo de las aplicaciones web. Fuente [12]

Las PWA son una evolución de las aplicaciones web tradicionales que combina características de las aplicaciones web y las aplicaciones móviles nativas. Se relaciona con una aplicación web en el sentido de que está construida utilizando tecnologías web estándar como HTML, CSS y JavaScript, y se accede a ella a través de un navegador web.

Sin embargo, una PWA ofrece una experiencia de usuario más similar a la de una aplicación nativa, ya que puede ofrecer funcionalidades como notificaciones push, acceso offline, y una interfaz de usuario que se adapta a diferentes dispositivos y tamaños de pantalla.

Algunas de las ventajas de una Progressive Web App incluyen:

- **Acceso desde múltiples plataformas:** Las PWA se pueden ejecutar en cualquier dispositivo que tenga un navegador web compatible, lo que permite llegar a una audiencia más amplia sin necesidad de desarrollar versiones específicas para cada plataforma.
- **Experiencia de usuario mejorada:** Al ofrecer características de una aplicación nativa, como notificaciones push y acceso offline, las PWA proporcionan una experiencia de usuario más rica y atractiva.
- **Mayor velocidad y rendimiento:** Las PWA están diseñadas para cargar rápidamente y ofrecer un rendimiento fluido, lo que mejora la retención de usuarios y reduce la tasa de abandono.
- **Instalación sin fricciones:** Los usuarios pueden "instalar" una PWA en su dispositivo simplemente agregándola a la pantalla de inicio, lo que elimina la necesidad de descargar e instalar desde una tienda de aplicaciones.
- **Actualizaciones automáticas:** Las PWA se actualizan automáticamente cada vez que se accede a ellas, lo que garantiza que los usuarios siempre tengan la última versión sin necesidad de descargar actualizaciones manualmente.

2.2.3. Sistemas de información geográfica (SIG)

Los Sistemas de Información Geográfica (SIG o GIS, por sus siglas en inglés Geographic Information Systems) son sistemas diseñados para capturar, almacenar, manipular, analizar y presentar datos geográficos. Estos datos geográficos incluyen información sobre características físicas y culturales de la Tierra, como montañas, ríos, ciudades, carreteras, límites políticos, entre otros.

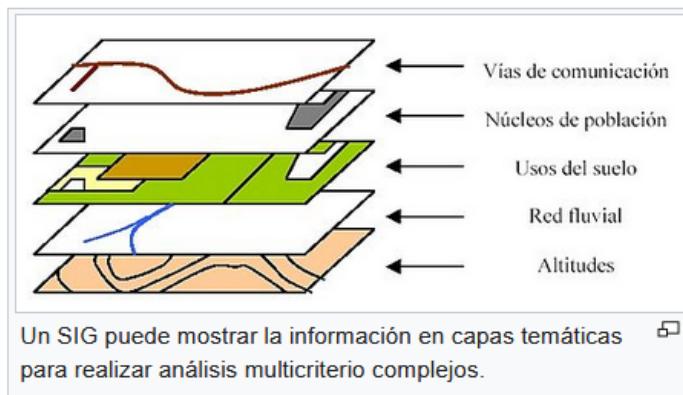


Figura 10: Definición básica de un SIG. Fuente [13]

Los SIG sirven para:

- **Visualización de datos espaciales:** Permiten representar datos geográficos de manera visual en mapas, lo que facilita la comprensión y la toma de decisiones.
- **Análisis espacial:** Proporcionan herramientas para realizar análisis espaciales, como superposición de capas, análisis de proximidad, interpolación, entre otros, que permiten extraer información útil y tomar decisiones informadas.
- **Gestión de datos geográficos:** Permiten almacenar y organizar datos geográficos de manera eficiente, lo que facilita su acceso, consulta y actualización.
- **Integración de datos:** Facilitan la integración de datos geográficos con otros tipos de datos, lo que permite obtener una visión más completa de la información.
- **Planificación y toma de decisiones:** Ayudan en la planificación y toma de decisiones en una variedad de campos, como urbanismo, gestión de recursos naturales, gestión de desastres, agricultura, salud pública, entre otros.

Los **WebGIS** son una extensión de los **SIG tradicionales** que aprovechan las tecnologías web para proporcionar acceso a la información geográfica de manera más accesible, colaborativa y distribuida.

2.2.4. MapLibre GL

MapLibre GL (<https://maplibre.org/maplibre-gl-js/docs/>), es una biblioteca de código abierto y gratuita para la creación de mapas interactivos en aplicaciones web. Se basa en Mapbox GL JS, que es una biblioteca JavaScript para renderizar mapas vectoriales interactivos utilizando WebGL.

Esta biblioteca permite a los desarrolladores crear aplicaciones web con mapas interactivos altamente personalizables y receptivos. Ofrece funciones avanzadas como zoom, rotación, desplazamiento suave, etiquetado dinámico y análisis de datos geoespaciales, todo ello renderizado en el navegador utilizando aceleración por hardware a través de WebGL.

MapLibre GL es ampliamente utilizada en proyectos de cartografía en línea, aplicaciones de seguimiento de ubicación, análisis geoespacial y muchas otras aplicaciones que requieren visualización de datos espaciales. Al ser de código abierto, permite a los desarrolladores adaptar y personalizar el mapa según las necesidades específicas de su proyecto, sin las restricciones de licencia asociadas con las opciones propietarias.

Las ventajas que ofrece MapLibre GL sobre las opciones propietarias serían:

- **Open Source y Gratuito:** MapLibre GL es una biblioteca de código abierto y gratuita, lo que significa que no hay costos de licencia asociados con su uso. Esto reduce significativamente los costos de desarrollo y despliegue de la aplicación.
- **Personalización Completa:** Proporciona una amplia gama de opciones de personalización, lo que permite adaptar el mapa a las necesidades específicas del proyecto. Los desarrolladores tienen control total sobre la apariencia y el comportamiento del mapa, lo que les permite crear experiencias de usuario únicas y personalizadas.
- **Flexibilidad en los Datos:** Es compatible con una variedad de fuentes de datos, incluidos datos vectoriales y rasterizados, así como servicios web de mapas como Mapbox y OpenStreetMap. Esto permite integrar fácilmente diferentes conjuntos de datos y servicios de mapas en la aplicación según sea necesario.
- **Escalabilidad y Rendimiento:** Está diseñado para ofrecer un rendimiento óptimo incluso con grandes cantidades de datos y alta interactividad. Utiliza técnicas de renderizado eficientes y optimización de recursos para garantizar una experiencia de usuario fluida y receptiva, incluso en dispositivos móviles y conexiones de red más lentas.
- **Independencia de Proveedores:** Al utilizar MapLibre GL, no estamos vinculados a un proveedor específico de mapas o servicios de geolocalización. Esto proporciona una mayor independencia y flexibilidad en la elección de proveedores de servicios y la gestión de

datos geoespaciales a lo largo del tiempo.

2.2.5. Tegola tileserver.

Tegola (<https://tegola.io/>), es un servidor de mapas que sirve datos geoespaciales pre-renderizados, conocidos como teselas (tiles), a aplicaciones de mapas en línea.

Estas aplicaciones, como mapas interactivos en sitios web o aplicaciones móviles, solicitan los "tiles" al servidor para mostrar mapas y datos geoespaciales a los usuarios de manera rápida y eficiente.

Cada "tile" es una pequeña imagen cuadrada que representa un fragmento del mapa en diferentes niveles de zoom. Estas imágenes se generan previamente a partir de datos geoespaciales más detallados, como mapas vectoriales o raster, y se almacenan en el servidor para su posterior distribución. Cuando un usuario solicita una vista de mapa específica, el servidor entrega los "tiles" correspondientes, reduciendo la carga de procesamiento en el cliente y optimizando el rendimiento general de la aplicación.

En este proyecto utilizaremos **Tegola**, que es un TileServer escrito en Go, especializado en la generación y distribución de "tiles" vectoriales, lo que significa que puede servir datos geoespaciales vectoriales, como geometrías y atributos, en lugar de imágenes raster. Algunas de las ventajas de Tegola incluyen:

- **Eficiencia en el consumo de ancho de banda:** Al servir datos vectoriales en lugar de imágenes raster, Tegola puede reducir significativamente el tamaño de los "tiles" transmitidos a través de la red, lo que resulta en un consumo de ancho de banda más eficiente y una carga más rápida de los mapas en las aplicaciones del cliente.
- **Flexibilidad y rendimiento:** Está diseñado para ser altamente configurable y escalable, lo que permite adaptarse a una amplia gama de casos de uso y cargas de trabajo. Su arquitectura modular y su capacidad para utilizar datos vectoriales preexistente permiten un rendimiento óptimo incluso en entornos con grandes volúmenes de datos geoespaciales.
- **Interoperabilidad:** Es compatible con estándares abiertos y protocolos comunes en el ámbito de los sistemas de información geográfica (SIG), lo que facilita la integración con otras herramientas y aplicaciones del ecosistema geoespacial.
- **Personalización y extensibilidad:** Ofrece una variedad de opciones de configuración y personalización para adaptarse a las necesidades específicas del proyecto. Además, su código fuente abierto y su comunidad activa de desarrolladores facilitan la creación de nuevas características y extensiones.

En el contexto de nuestro proyecto, el nomenclátor oficial de la Comunidad de Madrid (<https://gestiona.comunidad.madrid/nomecalles/>), nos servirá como fuente de datos geográficos para las capas auxiliares que serán expuestas por Tegola a nuestra aplicación cliente.

Figura 11: Imagen del nomenclátor oficial de la Comunidad de Madrid

2.2.6. Contenedores Linux con Docker

Docker (<https://www.docker.com/>), es una herramienta que nos permite unificar, gestionar y escalar nuestros servicios de aplicación de manera eficiente y consistente en cualquier entorno.



Figura 12: Logo Docker. Fuente [14]

Se trata de una plataforma de código abierto diseñada para facilitar la creación, implementación y ejecución de aplicaciones dentro de contenedores. Un contenedor es una unidad de software que incluye todo lo necesario para ejecutar una aplicación, incluidos el código, las bibliotecas y las dependencias, de manera eficiente y consistente en cualquier entorno.

Docker utiliza tecnologías de virtualización a nivel de sistema operativo para encapsular estas aplicaciones en entornos aislados y portátiles, lo que permite que se ejecuten de manera uniforme en cualquier sistema que ejecute Docker, independientemente de las diferencias en el sistema operativo o la configuración del hardware.

En el contexto de nuestro proyecto, utilizaremos Docker para unificar los servicios desarrollados en un archivo docker-compose.

Docker-compose [19] es una herramienta que permite definir y administrar aplicaciones multi-contenedor utilizando un archivo YAML. Este archivo especifica los servicios que componen la aplicación, así como sus configuraciones y dependencias. Al utilizar docker-compose, podemos definir fácilmente los servicios de nuestra aplicación, como el servidor backend, el frontend, el TileServer (Tegola), la base de datos (PostgreSQL con PostGIS), entre otros, en un solo archivo, lo que facilita la gestión y la configuración de la aplicación en diferentes entornos.

Docker nos permite igualar el entorno de desarrollo y despliegue mediante la creación de imágenes Docker para cada servicio de la aplicación. Estas imágenes contienen todas las dependencias y configuraciones necesarias para ejecutar el servicio de manera consistente en cualquier entorno, asegurando que el entorno de desarrollo sea idéntico al entorno de producción, lo que reduce los errores y facilita la detección y solución de problemas.

Docker facilita el escalado de la aplicación cuando llegue el momento mediante la implementación de múltiples instancias de los servicios en contenedores individuales. Esto se puede lograr fácilmente utilizando herramientas como Docker Swarm o Kubernetes, que permiten administrar y orquestar múltiples contenedores en clústeres distribuidos, lo que garantiza la disponibilidad y la escalabilidad de la aplicación según la demanda.

2.2.7. Servidor Privado Virtual (VPS)

Vamos a desplegar la aplicación en un VPS en la nube. Un VPS, o Servidor Privado Virtual, es un servidor virtualizado que funciona como un servidor físico dedicado, pero que comparte los recursos físicos con otros servidores virtuales en el mismo hardware físico. Esto significa que cada VPS tiene su propio sistema operativo, su propia cantidad dedicada de recursos (como CPU, RAM y almacenamiento) y su propia configuración de software, pero comparte el hardware físico con otros VPS en el mismo servidor físico.

Algunas de las ventajas de desplegar una aplicación en un VPS en la nube incluyen:

- **Escalabilidad:** Los VPS en la nube suelen ofrecer la capacidad de escalar vertical u horizontalmente según sea necesario, lo que permite aumentar o disminuir los recursos asignados a la aplicación para satisfacer la demanda de tráfico en tiempo real.
- **Flexibilidad:** Proporciona un entorno de alojamiento altamente personalizable, lo que permite a los desarrolladores elegir el sistema operativo, la configuración de software y las herramientas de desarrollo que mejor se adapten a las necesidades de su aplicación.
- **Fiabilidad:** Los proveedores de servicios en la nube suelen ofrecer una alta disponibilidad y tiempo de actividad garantizado para sus servicios, lo que reduce la posibilidad de interrupciones del servicio debido a fallos de hardware o mantenimiento programado.
- **Facilidad de gestión:** Los VPS en la nube suelen ofrecer interfaces de gestión intuitivas y herramientas automatizadas para la configuración, supervisión y gestión del servidor, lo que facilita la administración de la aplicación y la resolución de problemas.
- **Costo-efectividad:** Suelen ser más económicos que los servidores físicos dedicados, ya que comparten los costos de infraestructura y mantenimiento con otros clientes. Además, muchos proveedores de servicios en la nube ofrecen modelos de precios flexibles basados en el uso, lo que permite a los desarrolladores pagar solo por los recursos que utilizan.

También hemos de considerar sus posibles desventajas como:

- **Costos:** Aunque los VPS en la nube pueden ser más económicos que los servidores físicos dedicados, todavía implican costos operativos recurrentes que pueden aumentar con el tiempo, especialmente si la aplicación experimenta un crecimiento significativo.
- **Rendimiento no garantizado:** En entornos compartidos como los VPS, el rendimiento del servidor puede verse afectado por otras instancias que comparten el mismo hardware físico. Esto puede resultar en

variaciones en el rendimiento que no pueden ser completamente controladas por el usuario.

- **Dependencia del proveedor:** Aunque es posible migrar entre proveedores de nube, cambiar de proveedor puede ser complejo y llevar tiempo, lo que puede resultar en interrupciones del servicio durante el proceso de migración.
- **Limitaciones de recursos:** Los recursos asignados a un VPS, como la CPU, la memoria y el almacenamiento, pueden ser limitados en comparación con un servidor físico dedicado. Esto puede limitar la capacidad de escalar la aplicación para satisfacer picos de demanda repentinos.
- **Seguridad:** Si bien los proveedores de nube suelen ofrecer medidas de seguridad robustas, los VPS en la nube aún pueden ser vulnerables a ataques cibernéticos y brechas de seguridad, especialmente si no se implementan adecuadamente medidas de seguridad adicionales por parte del usuario.
- **Control limitado sobre la infraestructura física:** A diferencia de tener un servidor físico dedicado, donde se tiene control total sobre el hardware subyacente, en un VPS en la nube, el usuario no tiene control directo sobre la infraestructura física, lo que puede limitar la capacidad de optimizar el rendimiento y la configuración del servidor según las necesidades específicas de la aplicación.

El proveedor elegido para el despliegue de la aplicación es **DigitalOcean** (<https://www.digitalocean.com/>), debido a su facilidad de uso, precios competitivos, flexibilidad, escalabilidad, rendimiento, recursos de la comunidad y soporte técnico

Dada la naturaleza de los Servidores Privados Virtuales en la nube, cambiar de proveedor es una tarea fácil, facilitada por el hecho de que estamos desarrollando la aplicación utilizando contenedores de software.

Esto significa que, aunque hemos seleccionado DigitalOcean como proveedor en este momento, tenemos la flexibilidad de migrar nuestra aplicación a otros proveedores de la nube, como Amazon Web Services (AWS), Microsoft Azure o Google Cloud Platform, en caso de que sea necesario. [15]

Esta capacidad para evitar una gran dependencia con el proveedor de servicios (*vendor locking*) nos brinda una mayor libertad y control sobre nuestra infraestructura de nube, permitiendo adaptarnos a las necesidades cambiantes de nuestro proyecto y aprovechar las ofertas y características únicas de diferentes proveedores.

2.2.8. GitHub y GitHub Actions

Utilizaremos GitHub y GitHub Actions (<https://github.com/>), como repositorio y plataforma de despliegue ágil, respectivamente.

- GitHub es una plataforma de alojamiento de código basada en la nube que permite a los desarrolladores colaborar en proyectos de software mediante el control de versiones utilizando Git.
- GitHub Actions es una funcionalidad de GitHub que permite automatizar flujos de trabajo en respuesta a eventos en tu repositorio de GitHub. Con GitHub Actions, podremos configurar tareas automatizadas, como la construcción y prueba del código fuente, la publicación de artefactos, la notificación de cambios y el despliegue de aplicaciones. [16][17][18]

2.2.9. Propuesta arquitectónica del sistema.

El siguiente diagrama muestra como encajarían las piezas de software descritas en nuestro sistema, una vez esté totalmente desplegado en producción.

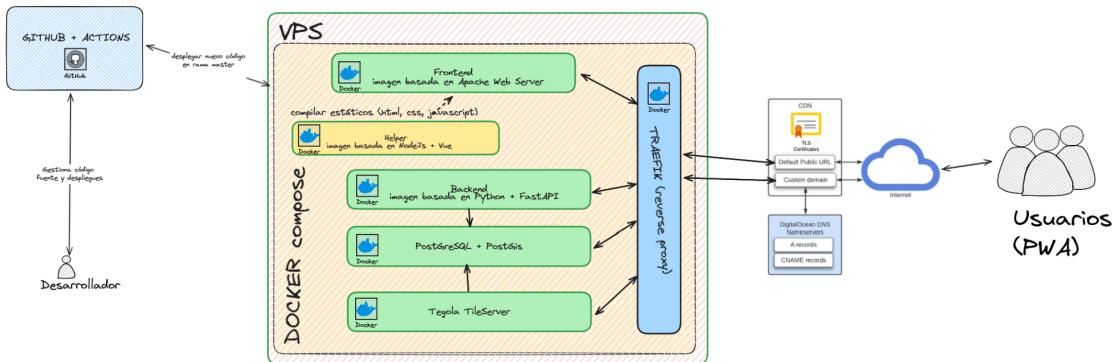


Figura 13: Arquitectura del sistema

La propuesta arquitectónica describe un sistema robusto y escalable que aprovecha tecnologías modernas y buenas prácticas de desarrollo y despliegue. Como hemos comentado, desplegaremos el sistema en un VPS en la nube, lo que proporciona flexibilidad, escalabilidad y fiabilidad para el servicio. Además, se garantizará la seguridad de las comunicaciones mediante HTTPS y se utilizará un dominio apuntando al servidor para los usuarios puedan acceder fácilmente al servicio.

Mediante la utilización de Docker y Docker Compose se simplifica el proceso de empaquetar, distribuir y ejecutar la aplicación en contenedores, facilitando así la gestión de la infraestructura y la escalabilidad de la aplicación. En la arquitectura propuesta cada servicio se ejecuta en un contenedor independiente. Esto promueve la modularidad, la flexibilidad y un mantenimiento simplificado del sistema. De esta manera, se optimiza los recursos y facilita la gestión y el mantenimiento de cada servicio.

Mediante la utilización de un proxy reverso (basado en Traefik), se simplifica la gestión de las peticiones y proporciona una capa adicional de seguridad al sistema.

La integración con GitHub y GitHub Actions facilita la colaboración en el desarrollo de la aplicación y automatiza el proceso de despliegue en producción. Esto garantiza una entrega continua y una mayor eficiencia en el ciclo de desarrollo.

En caso de necesitar escalar algún servicio, se puede extraer fácilmente del VPS principal y contratar un VPS dedicado específicamente para ese servicio. Esta capacidad de escalar de manera independiente cada componente del sistema permite adaptarse rápidamente a cambios en la demanda y garantiza un rendimiento óptimo en todo momento.

3. Implementación

En esta sección mostraremos los pasos seguidos para construir la solución software, acorde con la planificación establecida en la figura 4.

Para este proyecto, hemos creado un repositorio de código abierto en GitHub en la siguiente dirección: <https://github.com/jrgavilanes/TFG-EIEL>

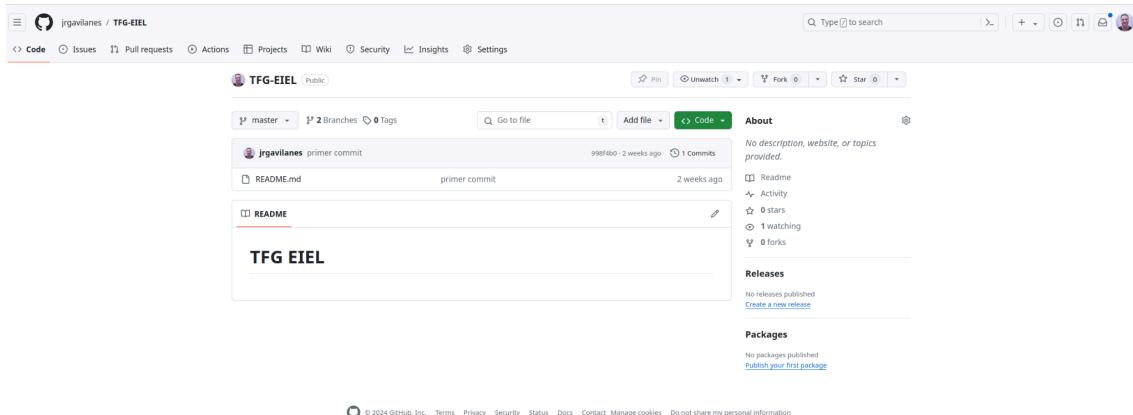


Figura 14: Repositorio en GitHub

Utilizaremos Gitflow para gestionar nuestro flujo de trabajo. Tendremos una rama `master` destinada al despliegue final, una rama `develop` para el desarrollo continuo, donde se integrarán todas las ramas `feature` con las nuevas características que desarrollemos, y ramas `release` para probar la versión en un entorno de preproducción antes de desplegarla en la rama `master`.

Ramas de publicación

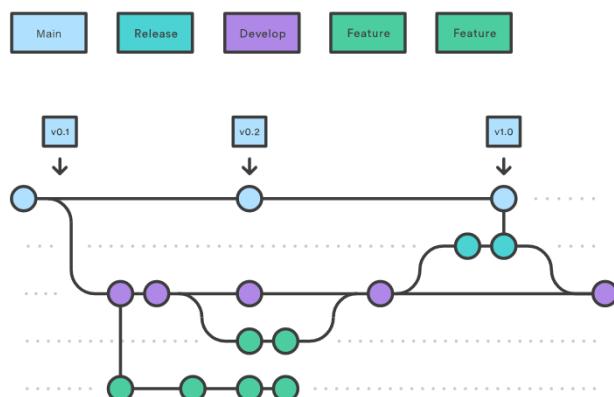


Figura 15: Esquema Gitflow

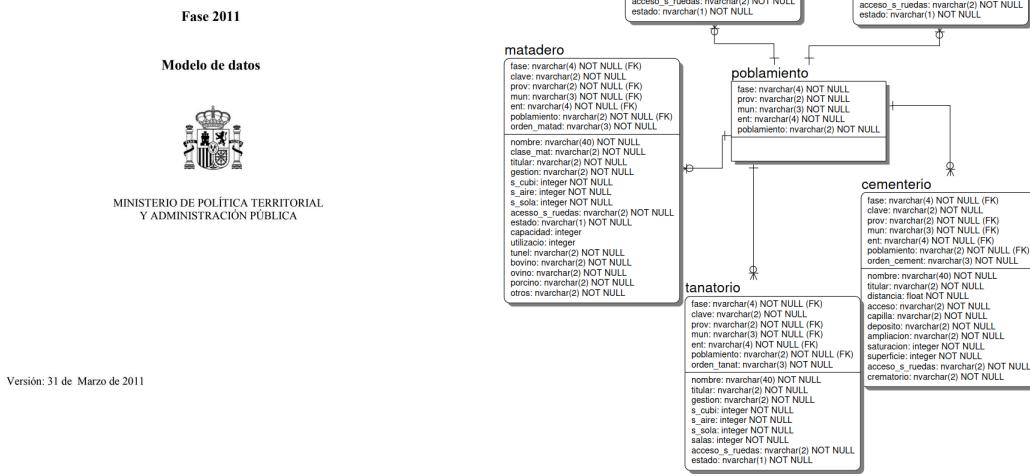
3.1. Preparar modelo de datos

3.1. Orígenes de datos

Los orígenes de datos que emplearemos será la descripción del modelo de datos solicitado por la EIEL y el nomeclátor de la Comunidad de Madrid.

3.1.1. Modelo de datos oficial

Encuesta de Infraestructura y Equipamientos Locales

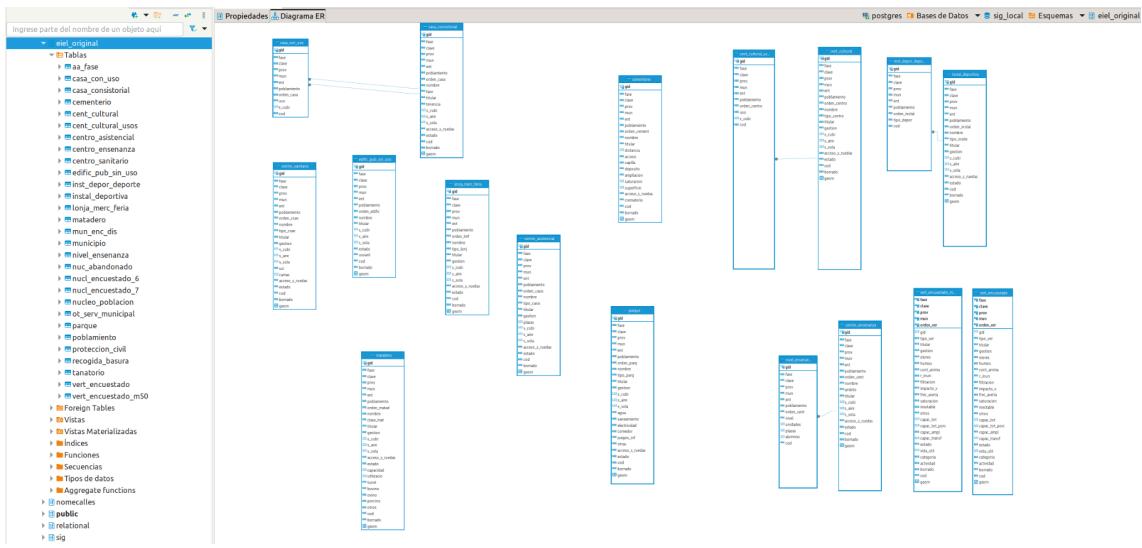


Modelo de datos (TO-DO: MEJORA ESTA SECCIÓN)

Validaciones

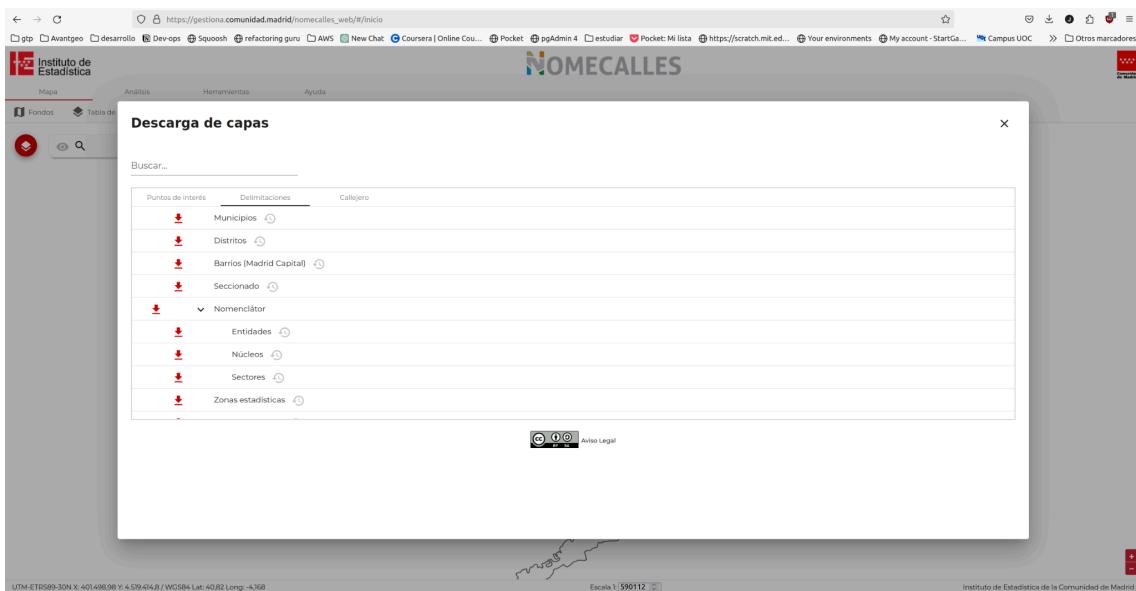
Manual de instrucciones

Partiendo de la descripción del modelo y sus validaciones específicas construiremos un esquema en nuestra base de datos que lo represente.



3.1.2. Nomenclátor oficial de la Comunidad de Madrid

Por otra parte, nos apoyaremos en el nomenclátor para obtener capas auxiliares con los puntos de interés de la comunidad de Madrid, y también para obtener los puntos de referencia de provincia, municipio, entidad y poblamiento al pulsar sobre el mapa, e intersectando la coordenada en la que pulsamos con las registradas en la tabla de núcleos y entidades.



Desde la web descargaremos cada capa en formato shapefile, y con el siguiente proceso importaremos sus datos en postgres. Por ejemplo para el caso de **núcleos**, estos serán los ficheros que obtendremos desde el noménclator

ficheros shapefile	Proceso de importación
<pre> nucl2023.cpg nucl2023.dbf nucl2023.prj nucl2023.sbn nucl2023.sbx nucl2023.shp nucl2023.shp.xml nucl2023.shx </pre>	<pre> # dependencia necesaria \$ sudo apt install postgis # Crea tabla nucleos en esquema nomecalles \$ shp2pgsql -s 25830 nucl2023.shp nomecalles.nucleos psql -U [user] -d [database] # cambiar de EPSG 25830 a EPSG 3857 sql: ALTER TABLE nomecalles.nucleos ADD COLUMN geom2 public.geometry(multipolygon, 3857); UPDATE nomecalles.nucleos SET geom2 = ST_Transform(geom, 3857); ALTER TABLE nomecalles.nucleos DROP COLUMN geom; ALTER TABLE nomecalles.nucleos RENAME COLUMN geom2 TO geom; </pre>

De esta forma obtendremos el esquema *nomecalles* con todas las tablas

3.2. Desarrollo Backend

Necesitaremos tener instalado en nuestro sistema una versión de Python igual o superior a la 3.7, y Docker engine con una versión igual o superior a la 24.0

3.2.1.- Configurar entorno de desarrollo

```
git clone git@github.com:jrgavilanes/TFG-EIEL.git  
git checkout feature/01_configuracion_entorno
```



Esta es la estructura inicial del proyecto

```
├── backend  
│   ├── requirements.txt  
│   └── database  
│       └── initdb  
│           └── 01.init.sql  
├── docker-compose-dev.yml  
├── .env-template  
└── frontend  
    ├── Dockerfile  
    ├── README.md  
    ├── salida.txt  
    └── tegola  
        └── config.toml
```

Arrancamos nuestro entorno de desarrollo con el siguiente archivo **docker-compose**, que establece toda la infraestructura necesaria para el proyecto:

```
# docker-compose-dev.yml  
  
version: '3'  
services:  
  
  # Servicio de base de datos PostgreSQL con PostGIS  
  db:  
    image: postgis/postgis:13-3.1  
    container_name: tfg-postgis-eiel  
    restart: always  
    environment:  
      POSTGRES_USER: ${DATABASE_USER}  
      POSTGRES_PASSWORD: ${DATABASE_PASSWORD}  
      POSTGRES_DB: sig_local  
    volumes:  
      - ./database/initdb:/docker-entrypoint-initdb.d  
      - tfg-pgdata:/var/lib/postgresql/data  
    ports:  
      - "8003:5432"  
  
  # Servicio de Redis para almacenamiento en caché  
  redis:  
    image: redis:6.2.6  
    container_name: tfg-redis  
    restart: always  
    ports:  
      - 8379:6379  
  
  # Servicio de proveedor de mapas vectoriales a través de Tegola y PostGIS  
  tegola-mvt-postgis-provider:  
    image: gospatial/tegola:v0.16.0  
    container_name: tfg-tegola-mvt-postgis-provider  
    depends_on:  
      - db  
      - redis
```

```

restart: always
ports:
  - 8005:8080
volumes:
  - ./tegola:/data
command: >
  serve --config /data/config.toml
environment:
  DB_HOST: db
  DB_PORT: 5432
  DB_NAME: sig_local
  DB_USER: ${DATABASE_USER}
  DB_PASSWORD: ${DATABASE_PASSWORD}

volumes:
  # Volumen persistente para los datos de PostgreSQL
  tfg-pgdata:

```

Este archivo define tres servicios principales:

- Un servicio de base de datos PostgreSQL con PostGIS, que alojará nuestra base de datos principal.
- Un servicio de proveedor de mapas vectoriales que utiliza Tegola y PostGIS para servir datos geoespaciales vectoriales.
- Un servicio de Redis para almacenamiento en caché, utilizado para mejorar el rendimiento de ciertas operaciones con Tegola.

Este entorno de desarrollo proporciona una configuración completa y lista para usar, lo que nos permite comenzar a trabajar en nuestro proyecto de manera eficiente y sin complicaciones.

El entorno de producción final será muy similar a este por lo que nos aseguramos que estamos trabajando con el mismo sistema tanto en desarrollo y producción.

La principal diferencia son los servicios que dan soporte al backend y al frontend, ya que en el entorno de desarrollo no los tendremos contenerizados, sino que los ejecutaremos directamente con sus frameworks de desarrollo, FastAPI y Vite, respectivamente.

Despues crear el fichero .env partiendo de .env-template, y de configurar las variables DATABASE_USER y DATABASE_PASSWORD, arrancamos y visualizamos el entorno de desarrollo

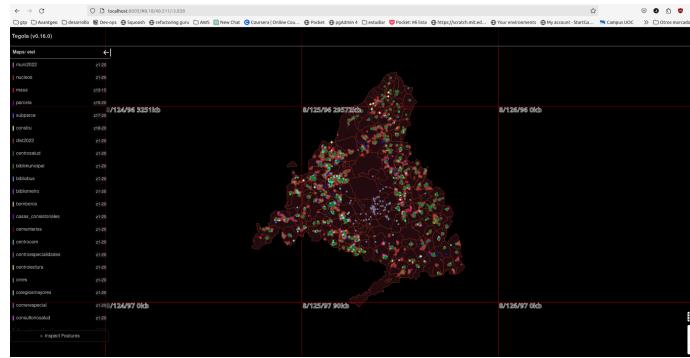
```

● janrax@janrax-home-linux:~/Escritorio/code/tfg-eiel$ docker-compose -f docker-compose-dev.yml up -d && docker-compose -f docker-compose-dev.yml start
tfg-postgis-eiel is up-to-date
tfg-redis is up-to-date
tfg-tegola-mvt-postgis-provider is up-to-date
Starting db ... done
Starting redis ... done
Starting tegola-mvt-postgis-provider ... done
● janrax@janrax-home-linux:~/Escritorio/code/tfg-eiel$ docker-compose -f docker-compose-dev.yml ps
          Name           Command           State           Ports
----- 
tfg-postgis-eiel      docker-entrypoint.sh postgres Up    0.0.0.0:8003->5432/tcp,:::8003->5432/tcp
tfg-redis             docker-entrypoint.sh redis ... Up    0.0.0.0:8379->6379/tcp,:::8379->6379/tcp
tfg-tegola-mvt-postgis-provider /opt/tegola serve --config ... Up    0.0.0.0:8005->8080/tcp,:::8005->8080/tcp
● janrax@janrax-home-linux:~/Escritorio/code/tfg-eiel$ 

```

Vemos que la base de datos escucha por el puerto 8003, Redis por 8379 y Tegola por el 8005.

Como ya hemos cargado en la base de datos la información del nomenclátor, con el procedimiento que hemos visto en el punto 3.1.2, ya podremos ver los puntos desde su cliente web.



El servidor Tegola y los datos almacenados en la base de datos, se relacionan mediante la configuración establecida en el fichero config.toml del directorio /tegola

```
config.toml
...
tegola > config.toml
1 [webserver]
2 port = "8080"
3
4 [webserver.headers]
5 Cache-Control = "s-maxage=3600"
6
7 [cache]
8 type="redis"
9 address="redis:6379"
10 password=""
11 ttl=15
12
13 [[providers]]
14 name = "eiel"
15 type = "mvn_postgis"
16 url = "postgres://$(DB_USER)$(DB_PASSWORD)@$(DB_HOST)$(DB_PORT)/$(DB_NAME)" # PostGIS connection string (required)
17 srid = 3857
18
19 [[providers.layers]]
20 name = "muni2022"
21 geometry_fieldname = "geom"
22 geometry_type="polygon"
23 id_fieldname = "gid"
24 srid = 3857
25 sql = SELECT ST_AsMVTGeom(geom, !bbox!) AS geom, * gid FROM nomecalles.muni2022 WHERE geom && !bbox!
26
27 [[providers.layers]]
28 name = "equipamientos_municipales"
29 geometry_fieldname = "geom"
30 geometry_type="point"
31 id_fieldname = "gid"
32 srid = 3857
33 sql = SELECT ST_AsMVTGeom(geom, !bbox!) AS geom, *, etiqueta AS mi_etiqueta FROM nomecalles.equipamientos_municipales WHERE used = 'N' and geom && !bbox!
34
35 [[providers.layers]]
36 name = "nucleos"
37 geometry_fieldname = "geom"
38 geometry_type="point"
39 id_fieldname = "gid"
40 srid = 3857
41 sql = SELECT ST_AsMVTGeom(geom, !bbox!) AS geom, * gid FROM nomecalles.nucleos WHERE geom && !bbox!
42
43
44 [[providers.layers]]
45 name = "masa"
46 geometry_fieldname = "geometry"
47 geometry_type="polygon"
48 id_fieldname = "gid"
49 srid = 3857
50 sql = SELECT ST_AsMVTGeom(geometry, !bbox!) AS geometry, *, gid FROM nomecalles.masa WHERE geometry && !bbox!
51
52 [[providers.layers]]
53 name = "parcela"
54 geometry_fieldname = "geometry"
55 geometry_type="polygon"
```

Para comenzar a programar el Backend, crearemos un entorno virtual en Python con las dependencias necesarias del proyecto.

The screenshot shows a terminal window with the following command history:

```
janrax@janrax-home-linux:~/Escritorio/code/tfg-eiel/backend$ python3 -m venv venv
janrax@janrax-home-linux:~/Escritorio/code/tfg-eiel/backend$ . venv/bin/activate
(venv) janrax@janrax-home-linux:~/Escritorio/code/tfg-eiel/backend$ pip install -r requirements.txt
Collecting fastapi==0.109.0
  Using cached fastapi-0.109.0-py3-none-any.whl (92 kB)
Collecting uvicorn==0.26.0
  Using cached uvicorn-0.26.0-py3-none-any.whl (60 kB)
Collecting jinja2
  Using cached jinja2-3.1.3-py3-none-any.whl (133 kB)
Collecting psycopg2-binary
  Using cached psycopg2-binary-2.9.9-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.0 MB)
Collecting shapely
  Using cached shapely-2.0.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (2.5 MB)
Collecting sqlalchemy==2.0.25
  Using cached SQLAlchemy-2.0.29-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (3.1 MB)
Collecting gealchemy2
  Using cached gealchemy2-0.15.0-py3-none-any.whl (73 kB)
Collecting uuid
  Using cached Gealchemy2-0.15.0-py3-none-any.whl (73.57/73.5 KB 4.0 MB/s eta 0:00:00)
```

Crear un entorno virtual en Python proporciona un ambiente limpio y controlado para el desarrollo de proyectos, asegurando la consistencia, la portabilidad y la facilidad de gestión de dependencias, ya que facilita:

- **Isolación de dependencias:** Puedes aislar las dependencias de tu proyecto de las instaladas globalmente en tu sistema. Esto evita conflictos entre versiones de paquetes y asegura que tu proyecto funcione de manera consistente, independientemente de las dependencias de otros proyectos o del sistema en general.
- **Facilidad de gestión de dependencias:** Con un entorno virtual, puedes instalar y gestionar las dependencias específicas de tu proyecto de forma independiente. Esto hace que sea más fácil mantener un registro claro de las dependencias utilizadas y sus versiones correspondientes.
- **Portabilidad del entorno de desarrollo:** Al compartir tu código con otros desarrolladores o al desplegar tu aplicación en diferentes entornos, un entorno virtual garantiza que todas las dependencias necesarias estén disponibles y sean consistentes en todos los sistemas.
- **Experimentación segura:** Puedes probar nuevas bibliotecas o versiones de bibliotecas en un entorno virtual sin afectar el entorno global de Python en tu sistema. Esto te permite experimentar con nuevas herramientas y tecnologías de forma segura.

Aunque en nuestro caso con Docker, ya se proporciona un entorno aislado para el desarrollo de aplicaciones, sin embargo, crear entornos virtuales en Python sigue siendo una práctica recomendada para garantizar la consistencia, la portabilidad y el aislamiento de las dependencias del proyecto.

Iniciaremos el IDE Pycharm cargando el entorno recién creado, y comprobaremos con un ejemplo básico de API.

```

main.py <
1  from fastapi import FastAPI
2
3  # Crea una instancia de la aplicación FastAPI
4  app = FastAPI()
5
6  # Define una ruta raíz que responde con un mensaje
7  @app.get("/")
8  def read_root():
9      return {"message": "Hello, world!"}
10
11  # Define una ruta que responde un gabinete u lo devuelve
12  @app.get("/items/{item_id}")
13  def read_item(item_id: int, q: str = None):
14      return {"item_id": item_id, "q": q}
15
16  # Ejecuta el servidor solo si este archivo se ejecuta directamente
17  if __name__ == "__main__":
18      import uvicorn
19
20      # Inicia el servicio con uvicorn
21      uvicorn.run(app, host="127.0.0.1", port=8000)
22
23
24 if __name__ == "__main__":
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
417
418
419
419
420
421
422
423
423
424
425
425
426
426
427
427
428
428
429
429
430
430
431
431
432
432
433
433
434
434
435
435
436
436
437
437
438
438
439
439
440
440
441
441
442
442
443
443
444
444
445
445
446
446
447
447
448
448
449
449
450
450
451
451
452
452
453
453
454
454
455
455
456
456
457
457
458
458
459
459
460
460
461
461
462
462
463
463
464
464
465
465
466
466
467
467
468
468
469
469
470
470
471
471
472
472
473
473
474
474
475
475
476
476
477
477
478
478
479
479
480
480
481
481
482
482
483
483
484
484
485
485
486
486
487
487
488
488
489
489
490
490
491
491
492
492
493
493
494
494
495
495
496
496
497
497
498
498
499
499
500
500
501
501
502
502
503
503
504
504
505
505
506
506
507
507
508
508
509
509
510
510
511
511
512
512
513
513
514
514
515
515
516
516
517
517
518
518
519
519
520
520
521
521
522
522
523
523
524
524
525
525
526
526
527
527
528
528
529
529
530
530
531
531
532
532
533
533
534
534
535
535
536
536
537
537
538
538
539
539
540
540
541
541
542
542
543
543
544
544
545
545
546
546
547
547
548
548
549
549
550
550
551
551
552
552
553
553
554
554
555
555
556
556
557
557
558
558
559
559
560
560
561
561
562
562
563
563
564
564
565
565
566
566
567
567
568
568
569
569
570
570
571
571
572
572
573
573
574
574
575
575
576
576
577
577
578
578
579
579
580
580
581
581
582
582
583
583
584
584
585
585
586
586
587
587
588
588
589
589
590
590
591
591
592
592
593
593
594
594
595
595
596
596
597
597
598
598
599
599
600
599
600
601
601
602
602
603
603
604
604
605
605
606
606
607
607
608
608
609
609
610
610
611
611
612
612
613
613
614
614
615
615
616
616
617
617
618
618
619
619
620
620
621
621
622
622
623
623
624
624
625
625
626
626
627
627
628
628
629
629
630
630
631
631
632
632
633
633
634
634
635
635
636
636
637
637
638
638
639
639
640
640
641
641
642
642
643
643
644
644
645
645
646
646
647
647
648
648
649
649
650
650
651
651
652
652
653
653
654
654
655
655
656
656
657
657
658
658
659
659
660
660
661
661
662
662
663
663
664
664
665
665
666
666
667
667
668
668
669
669
670
670
671
671
672
672
673
673
674
674
675
675
676
676
677
677
678
678
679
679
680
680
681
681
682
682
683
683
684
684
685
685
686
686
687
687
688
688
689
689
690
690
691
691
692
692
693
693
694
694
695
695
696
696
697
697
698
698
699
699
700
699
700
701
701
702
702
703
703
704
704
705
705
706
706
707
707
708
708
709
709
710
710
711
711
712
712
713
713
714
714
715
715
716
716
717
717
718
718
719
719
720
720
721
721
722
722
723
723
724
724
725
725
726
726
727
727
728
728
729
729
730
730
731
731
732
732
733
733
734
734
735
735
736
736
737
737
738
738
739
739
740
740
741
741
742
742
743
743
744
744
745
745
746
746
747
747
748
748
749
749
750
750
751
751
752
752
753
753
754
754
755
755
756
756
757
757
758
758
759
759
760
760
761
761
762
762
763
763
764
764
765
765
766
766
767
767
768
768
769
769
770
770
771
771
772
772
773
773
774
774
775
775
776
776
777
777
778
778
779
779
780
780
781
781
782
782
783
783
784
784
785
785
786
786
787
787
788
788
789
789
790
790
791
791
792
792
793
793
794
794
795
795
796
796
797
797
798
798
799
799
800
799
800
801
801
802
802
803
803
804
804
805
805
806
806
807
807
808
808
809
809
810
810
811
811
812
812
813
813
814
814
815
815
816
816
817
817
818
818
819
819
820
820
821
821
822
822
823
823
824
824
825
825
826
826
827
827
828
828
829
829
830
830
831
831
832
832
833
833
834
834
835
835
836
836
837
837
838
838
839
839
840
840
841
841
842
842
843
843
844
844
845
845
846
846
847
847
848
848
849
849
850
850
851
851
852
852
853
853
854
854
855
855
856
856
857
857
858
858
859
859
860
860
861
861
862
862
863
863
864
864
865
865
866
866
867
867
868
868
869
869
870
870
871
871
872
872
873
873
874
874
875
875
876
876
877
877
878
878
879
879
880
880
881
881
882
882
883
883
884
884
885
885
886
886
887
887
888
888
889
889
890
890
891
891
892
892
893
893
894
894
895
895
896
896
897
897
898
898
899
899
900
899
900
901
901
902
902
903
903
904
904
905
905
906
906
907
907
908
908
909
909
910
910
911
911
912
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
920
921
921
922
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
930
931
931
932
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
940
941
941
942
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
950
951
951
952
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
960
961
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
970
971
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
980
981
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
990
991
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1
```

3.2.2.- API Rest

3.2.2.1. Módulo de acceso a base de datos

```
git clone git@github.com:jrgavilanes/TFG-EIEL.git  
git checkout feature/02_configurar_acceso_base_datos
```



Desde database.py, se configura la conexión a una base de datos PostgreSQL utilizando SQLAlchemy y FastAPI, proporcionando funciones y dependencias para interactuar con la base de datos en las rutas de API.

```
database.py ✘  
1 import os  
2 from typing import Annotated  
3  
4 from fastapi import Depends  
5 from sqlalchemy import create_engine  
6 from sqlalchemy.orm import sessionmaker, Session  
7  
8 POSTGRES_USER = os.getenv('POSTGRES_USER', 'janrax')  
9 POSTGRES_PASSWORD = os.getenv('POSTGRES_PASSWORD', 'janrax')  
10 POSTGRES_DB = os.getenv('POSTGRES_DB', 'sig_local')  
11 POSTGRES_HOST = os.getenv('POSTGRES_HOST', 'localhost')  
12 POSTGRES_PORT = os.getenv('POSTGRES_PORT', '9003')  
13  
14 POSTGRES_DATABASE_URL = (  
15     f'postgresql+psycopg2://{{POSTGRES_USER}}:  
16     {{POSTGRES_PASSWORD}}@'  
17     f'{{POSTGRES_HOST}}:{{POSTGRES_PORT}}'  
18     f'/{{POSTGRES_DB}}'  
19 )  
20 engine_postgres = create_engine(POSTGRES_DATABASE_URL)  
21  
22 SessionLocalPostgres = sessionmaker(autocommit=False, autoflush=False, bind=engine_postgres)  
23  
24  
25 usage ± jrgavilanes  
26 def get_db_postgres():  
27     db = SessionLocalPostgres()  
28     try:  
29         yield db  
30     finally:  
31         db.close()  
32  
33 ± jrgavilanes  
34 def fetch_records_and_convert(db, query, values=None):  
35     records = db.execute(query, values)  
36     return [dict(zip(records.keys(), row)) for row in records.fetchall()]  
37  
38 db_dependency_postgres = Annotated[Session, Depends(get_db_postgres)]  
39
```

Recuperará la configuración de conexión desde las variables de entorno, y utilizará unas por defecto si no se proporcionan.

Con `get_db_postgres()`, devuelve una sesión de base de datos PostgreSQL. Utiliza el decorador `yield` para convertirla en un generador, lo que permite que la sesión se utilice como una dependencia en rutas de API de FastAPI. Se asegura de cerrar la sesión de manera adecuada utilizando un bloque `finally`.

Con `db_dependency_postgres`, se define una dependencia que se utilizará en las rutas de API para injectar una sesión de base de datos PostgreSQL. Utiliza `Annotated` para proporcionar anotaciones de tipo, especificando que la dependencia es una instancia de `Session` y depende de la función `get_db_postgres`.

3.2.2.2. Módulo de seguridad

```
git clone git@github.com:jrgavilanes/TFG-EIEL.git  
git checkout feature/03_modulo_seguridad
```



Almacenaremos los usuarios en la tabla *users* del esquema *auth*, con la siguiente estructura.

```
--CREATE TABLE auth.users (  
    id uuid DEFAULT uuid_generate_v4() NOT NULL,  
    name varchar(255) NOT NULL,  
    "password" varchar(255) NOT NULL,  
    "role" varchar(255) DEFAULT 'operator'::character varying NOT NULL,  
    municipality varchar(255) NULL,  
    active bool DEFAULT true NOT NULL,  
    CONSTRAINT municipality_not_null CHECK (((role)::text <> 'cityhall'::text) OR (((role)::text = 'cityhall'::text) AND (length((municipality)::text) >= 1))),  
    CONSTRAINT name_length CHECK (length((name)::text) >= 1),  
    CONSTRAINT users_name_key UNIQUE (name),  
    CONSTRAINT users_password_check CHECK ((length((password)::text) >= 1)),  
    CONSTRAINT users_pkey PRIMARY KEY (id),  
    CONSTRAINT users_role_check CHECK (((role)::text = ANY (ARRAY[('admin'::character varying)::text, ('operator'::character varying)::text, ('cityhall'::character varying)::text]))))  
) ;
```

A modo de recordatorio, la aplicación tendrá tres roles de acceso: admin, operator y cityhall. En el caso del rol cityhall, será obligatorio que se indique su municipio, ya que sólo tendrá acceso a los equipamientos registrados en su localidad. Los roles operator y admin, tendrán acceso a todos los equipamientos, pero sólo este último tendrá acceso a las tareas administrativas como crear usuarios, generar listados, etc.

En el modulo auth.py se agruparán las funciones que gestionan toda la seguridad.

El sistemas de seguridad entre backend y frontend será a traves de JWT con una duración del token de 8 horas.

JWT se utiliza comúnmente para autenticación y autorización en aplicaciones web y APIs. Cuando un usuario se autentica con éxito, el servidor genera un JWT y lo devuelve al cliente.

El cliente incluye este JWT en cada solicitud subsiguiente, ya sea en el encabezado de Autorización o como un parámetro de consulta. El servidor verifica la firma del JWT para garantizar que no haya sido manipulado y luego extrae la información del usuario de la carga útil para determinar si tiene acceso a los recursos solicitados.

Una de las ventajas clave de JWT es su capacidad para mantener el estado de la sesión del usuario en el cliente, lo que reduce la carga en el servidor y simplifica la implementación de servicios web escalables y seguros. Además, al ser autónomo, no requiere almacenar información de sesión en el servidor, lo que lo hace útil en arquitecturas distribuidas y sin estado.

En el siguiente código, vemos la función que se inyectará en los endpoints securizados y que se encargará de validar el token enviado en cada llamada.

```
auth.py ×
16     from database import db_dependency_postgres, fetch_records_and_convert
17
18     router = APIRouter(
19         prefix="/api/auth",
20         tags=['/api/auth']
21     )
22
23     JWT_SECRET_KEY = os.getenv('JWT_SECRET_KEY',
24                                "d1fc798b7e37a3e3fdfc42773aa449ba523b2139f52c09809eaf12a98b37de93") # $ openssl rand -hex 32
25     ALGORITHM = 'HS256'
26     oauth2_bearer = OAuth2PasswordBearer(tokenUrl="token")
27
28     bcrypt_context = CryptContext(schemes=["bcrypt"], deprecated="auto")
29
30
31     usage = jrgavilanes +1
32     async def get_current_user(token: Annotated[str, Depends(oauth2_bearer)]):
33         try:
34             payload = jwt.decode(token, JWT_SECRET_KEY, ALGORITHM)
35             username: str = payload.get("username")
36             user_id: int = payload.get("id")
37             role: str = payload.get("role")
38             municipality: str = payload.get("municipality")
39             is_desktop: bool = payload.get("is_desktop")
40             if username is None or user_id is None or role is None:
41                 raise HTTPException(status_code=status.HTTP_401_UNAUTHORIZED,
42                                     detail="No autorizado - Token inválido o caducado")
43             return {
44                 "username": username,
45                 "user_id": user_id,
46                 "role": role,
47                 "municipality": municipality,
48                 "is_desktop": is_desktop
49             }
50         except JWTError:
51             raise HTTPException(status_code=status.HTTP_401_UNAUTHORIZED,
52                                 detail="No autorizado - Token inválido o caducado")
53
54     user_dependency = Annotated[dict, Depends(get_current_user)]
55
56
```

A continuación vemos un ejemplo de uso, en el caso del endpoint de creación de un nuevo usuario. Previamente comprobamos que el token enviado es válido y corresponde a un administrador. Despues almacenaremos la contraseña de forma encriptada en la base de datos.

```
1 usage = jrgavilanes
class CreateUserRequest(BaseModel):
    name: str
    password: str
    role: str
    municipality: Optional[str]

    @jrgavilanes
    @router.post(path="/", status_code=201)
    async def create_user(
        db_postgres: db_dependency_postgres,
        user_request: CreateUserRequest,
        user: user_dependency
    ):
        if user.get("role") != "admin":
            raise HTTPException(status_code=403, detail="No eres Rol admin. Acción no permitida")

        query = sql.text("""
            insert into auth.users (name, password, role, municipality)
            values (:name, :password, :role, :municipality)
            returning id;
        """)
        values = {
            "name": user_request.name,
            "password": bcrypt_context.hash(user_request.password),
            "role": user_request.role,
            "municipality": user_request.municipality
        }
        try:
            user_id = fetch_records_and_convert(db_postgres, query, values)
            db_postgres.commit()
            return {"id": user_id[0].get("id")}
        except Exception as e:
            raise HTTPException(status_code=400, detail=str(e.orig))
```

A continuación hacemos la llamada para crear un usuario

POST http://127.0.0.1:7000/api/auth

http://127.0.0.1:7000/api/auth

Send

Params Auth Headers (9) Body Pre-request Tests Settings Cookies Beautify

raw JSON

```
1 "name": "demo",
2 "password": "demo",
3 "role": "admin",
4 "municipality": ""
5
6 
```

Body

201 Created 352 ms 175 B Save Response

Pretty Raw Preview Visualize JSON

```
1 "id": "2ea...8cb"
```

La llamada anterior, ha generado el nuevo registro en la base de datos.

Propiedades		Datos		Diagrama ER				tfg-sig_local	Bases de Datos
users		Enter a SQL expression to filter results (use Ctrl+Space)							
		id	name	password	role	municipality	active		
Grilla	to	1	zeaf00b-793f-4595-a43-86fafdd2a8cb	demo	\$2b\$12\$XuJgen3RCFViSyBSFDye6KGQsbMp1UxeR2aVtv6upclX9yKW1lm	admin	[v]		

Ahora comprobamos desde en entorno de prueba FastApi, que validándonos con el nuevo usuario, el sistema nos devuelve un jwt token.

FastAPI 0.1.0 OAS 3.1

/api/openapi.json

Authorize 

/api/auth

POST /api/auth/ Create User 

POST /api/auth/validate Validate User 

Parameters

No parameters

Request body required  

application/json

{
 "name": "demo",
 "password": "demo"
}

Execute 

Responses

Curl

```
curl -X "POST" \
  "http://127.0.0.1:7000/api/auth/validate" \
  -H "Content-Type: application/json" \
  -d {"name": "demo",  
       "password": "demo"}  
{
```

Request URL

http://127.0.0.1:7000/api/auth/validate

Server response

Code **Details**

200

Response body

```
"eyJhbGciOiJIUzI1NiJ9.aScC1lkpXVC29_rwJ2c7vlnfzt5L81e81m81lCjB2CTC1j1YnG1M0E11t2SP7+000N1n1z2Q1LtgZpZn05Dy7ThjYi1sIn>sd01s1hZC1ph1xTm1bm1jxRbhk016e1611s1s1x2ZB1r1t9B3d01RyDm01s1v4cC10HTcxHDYxMD80H6_8ngZGdF8FD9E82unf4aB_Xt0q1hp1Mcpct2}seew0"
```



Response headers

```
content-length: 258  
content-type: application/json  
date: Mon, 19 Dec 2023 16:49:07 GMT  
server: uvicorn
```

Y finalmente comprobamos que el token generado contiene los atributos correctamente.

The screenshot shows the JUNT (JWT Debugger) tool interface. At the top, there are tabs for 'Encoded' (selected) and 'Decoded'. Below the tabs is a text input field labeled 'PASTE A TOKEN HERE' containing a long JWT string. To the right, under the 'Decoded' tab, is a detailed breakdown of the token's structure:

- HEADER: ALGORITHM & TOKEN TYPE**

```
{  
  "alg": "HS256",  
  "typ": "JWT"  
}
```
- PAYOUT: DATA**

```
{  
  "username": "demo",  
  "id": "2eafb00b-793f-4595-af49-86fafdd2a8cb",  
  "role": "admin",  
  "municipality": "",  
  "is_desktop": true,  
  "exp": 1714610948  
}
```
- VERIFY SIGNATURE**

```
HMACSHA256(  
  base64UrlEncode(header) + "." +  
  base64UrlEncode(payload),  
  your-256-bit-secret  
)  secret base64 encoded
```

Below the main area, two warning messages are displayed:

- Warning:** Looks like your JWT signature is not encoded correctly using base64url (<https://tools.ietf.org/html/rfc4648#section-5>). Note that padding ("=") must be omitted as per <https://tools.ietf.org/html/rfc7515#section-2>
- Warning:** Looks like your JWT header is not encoded correctly using base64url (<https://tools.ietf.org/html/rfc4648#section-5>). Note that padding ("=") must be omitted as per <https://tools.ietf.org/html/rfc7515#section-2>

3.2.2.3. Endpoints auxiliares

```
git clone git@github.com:jrgavilanes/TFG-EIEL.git  
git checkout feature/04_endpoints_auxiliares
```



En esta sección daremos de alta endpoints que darán soporte a los endpoints de equipamientos.

Por ejemplo se guardará un registro con la actividad de los usuarios en la tabla de auditorias

```
-- auth.audits definition

-- Drop table
-- DROP TABLE auth.audits;

CREATE TABLE auth.audits (
    user_id uuid NOT NULL,
    access_time timestamp NOT NULL,
    equipment varchar NOT NULL,
    function_name text NOT NULL,
    parameters json NOT NULL,
    CONSTRAINT fk_user_id FOREIGN KEY (user_id) REFERENCES auth.users(id)
);
```

Cada operación que modifique el estado de algún equipamiento hará uso de la siguiente función que registrará el movimiento de usuario.

```
audit.py ×

1  from sqlalchemy import sql
2  from datetime import datetime
3  import json
4
5
6  7 usages  ± jrgavilanes@gmail.com <jrgavilanes@gmail.com>
7  async def register_user_access(db_postgres, user_id, equipment, function_name, parameters, commit=False):
8      query = sql.text("""
9          INSERT INTO auth.audits (user_id, access_time, equipment, function_name, parameters)
10         VALUES (:user_id, :access_time, :equipment, :function_name, :parameters)
11     """)
12      values = {
13          "user_id": user_id,
14          "access_time": datetime.now(),
15          "equipment": equipment,
16          "function_name": function_name,
17          "parameters": json.dumps(parameters)
18      }
19      db_postgres.execute(query, values)
20      if commit:
21          db_postgres.commit()
```

En uploader.py se gestionará todo lo relacionado con el manejo de las imágenes, que se minimizarán y se subirán a un contenedor de objetos (bucket S3)

```
uploader.py ×
new *
109 @router.post( path: "/{equipment_type}/{gid}/{cod}/{lat}/{lng}", status_code=200)
110 async def upload_file(
111     background_tasks: BackgroundTasks,
112     db_postgres: db_dependency_postgres,
113     user: user_dependency,
114     equipment_type: str,
115     gid: int,
116     cod: str,
117     lat: float,
118     lng: float,
119     new_file: UploadFile = File(...),
120 )
121     if user.get("role") not in ["admin", "operator"]:
122         raise HTTPException(status_code=403, detail="acción no permitida")
123
124     file_contents = await new_file.read()
125     filename = str(uuid.uuid4())
126     extension = os.path.splitext(new_file.filename)[1]
127
128     background_tasks.add_task(upload_to_bucket, *args: file_contents, S3_BUCKET_NAME, f'{filename}{extension}', new_file.content_type)
129
130     background_tasks.add_task(minify_image_and_upload, *args: file_contents, S3_BUCKET_NAME,
131                             f'{S3_PHOTO_MINIFY_BASE_URL.split("/")[-1]}/{filename}{extension}', new_file.content_type)
132
133     query = sql.text("""
134         INSERT INTO eiel.equipamiento_fotos (idef, tipo_equipamiento, tipo_equipamiento_gid, original_path, minify_path, fecha_hora, lat, lng, cod)
135         VALUES (:idef,:tipo_equipamiento, :tipo_equipamiento_gid, :original_path, :minify_path, :fecha_hora, :lat, :lng, :cod)
136     """)
137     values = {
138         "idef": filename,
139         "tipo_equipamiento": equipment_type,
140         "tipo_equipamiento_gid": gid,
141         "original_path": f'{S3_PHOTO_BASE_URL}/{filename}{extension}',
142         "minify_path": f'{S3_PHOTO_MINIFY_BASE_URL}/{filename}{extension}',
143         "fecha_hora": datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
144         "lat": lat,
145         "lng": lng,
146         "cod": cod
147     }
148     db_postgres.execute(query, values)
149
150     await register_user_access(db_postgres=db_postgres,
151                               user_id=user.get("user_id"),
152                               equipment=TABLE_NAME,
153                               function_name="upload_file",
154                               parameters={"equipment_type": equipment_type,
155                                           "gid": gid,
```

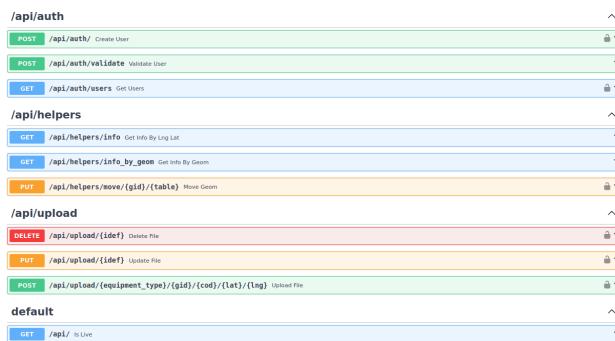
Almacenaremos la referencia y la geolocalización de la imagen en la tabla eiel.equipamiento_fotos

```
CREATE TABLE eiel.equipamiento_fotos (
    idef uuid NOT NULL,
    cod varchar(50) NULL,
    original_path varchar(250) NULL,
    minify_path varchar(250) NULL,
    tipo bpchar(2) DEFAULT 'PG'::bpchar NULL,
    fecha_hora bpchar(14) NULL,
    comentario text NULL,
    lat float8 NULL,
    lng float8 NULL,
    tipo_equipamiento varchar(2) NULL,
    tipo_equipamiento_gid int4 NULL,
    CONSTRAINT equipamiento_fotos_pkey PRIMARY KEY (idef)
);
```

En helpers.py, tendremos funciones de propósito general comunes a todos los equipamientos, así como la obtención del municipio madrileño a partir de la latitud y de la longitud transmitida al crear un markpoint en el mapa de la aplicación frontend, como veremos posteriormente.

```
helpers.py
2 usages new *
27     async def info_by_lng_lat(db_postgres, lat, lng):
28         fase = "2023"
29         entidad_colectiva = "00"
30         provincia = "28"
31         entidad = ""
32         municipio = ""
33         poblamiento = "99"
34         query = sql.text(f"""
35             select codine, geocodigo
36             from nomecalles.nucleos
37             where ST_INTERSECTS(geom, st_transform(ST_SetSRID(ST_MakePoint(:lng,:lat), 4326),3857));
38         """)
39         values = {
40             "lng": lng,
41             "lat": lat
42         }
43         nucleo = fetch_records_and_convert(db_postgres, query, values)
44         if len(nucleo) > 0:
45             poblamiento = nucleo[0].get("geocodigo")
46             poblamiento = poblamiento[-2:]
47             query = sql.text(f"""
48                 select geocodigo
49                 from nomecalles.entidades
50                 where ST_INTERSECTS(geom, st_transform(ST_SetSRID(ST_MakePoint(:lng),:lat), 4326),3857);
51             """)
52             values = {
53                 "lng": lng,
54                 "lat": lat
55             }
56             qentidad = fetch_records_and_convert(db_postgres, query, values)
57             if len(qentidad) > 0:
58                 geocodigo = qentidad[0].get("geocodigo")
59                 entidad = f"00{geocodigo[-2:]}"
60                 municipio = geocodigo[:3]
61
62             return {
63                 "lng": lng,
64                 "lat": lat,
65                 "fase": fase,
66                 "provincia": provincia,
67                 "entidad_colectiva": entidad_colectiva,
68                 "entidad": entidad,
69                 "municipio": municipio,
70                 "poblamiento": poblamiento,
71                 "qnucleo": nucleo,
72                 "qentidad": qentidad,
73             }
74         }
```

Una vez compilado el proyecto con el nuevo código, estos serán los endpoints que tendremos disponibles



3.2.2.4. Endpoints de equipamientos

```
git clone git@github.com:jrgavilanes/TFG-EIEL.git  
git checkout feature/05_endpoints_equipamientos
```



En esta sección daremos de alta los 14 endpoints con los equipamientos a gestionar.

Cada commit de la rama contendrá un equipamiento completo.

El código de los endpoints de cada equipamiento es muy similar, salvando la diferencia de campos de las tablas, por lo que es muy susceptible de refactorizarse en futuras iteraciones del proyecto.

Veremos en profundidad el código del primer equipamiento, y será extrapolable al resto de ellos.

El primer endpoint de equipamientos que crearemos, será el de cementerios (`cemeteries.py`), donde comenzaremos importando el código de acceso a base de datos, autorización, auditoría y funciones comunes vistas en las secciones anteriores de este documento.

```
➊ cemeteries.py ×  
1  from collections import OrderedDict  
2  from typing import Optional  
3  
4  from fastapi import APIRouter, HTTPException  
5  from pydantic import BaseModel  
6  from sqlalchemy import sql  
7  
8  from database import db_dependency_postgres, fetch_records_and_convert  
9  from routers.audit import register_user_access  
10 from routers.auth import user_dependency  
11 from routers.helpers import exists_equipment, update_record_geom_with_nomecalles_geom, release_nomecalles_layer, \  
12     delete_images_from_s3, recalculate_cod, get_images, check_municipality, get_all_my_records_by_table_name, \  
13     get_new_code, update_nomecalles  
14  
15 router = APIRouter(  
16     prefix='/api/cemeteries',  
17     tags=['/api/cemeteries'])  
18 )  
19  
20 EIEL_TABLE_NAME = "cementerio"  
21 ORDER_FIELD = "orden_cement"  
22 NOMECALLES_TABLES_NAMES = [  
23     "cemeterios"  
24 ]  
25 TIPO_EQUIPAMIENTO = "CE"  
26
```

Creamos constantes con los nombres de tablas, campo de ordenación y tipo de equipamiento para, más adelante, facilitar la construcción de queries parametrizadas.

Estos son dos modelos de solicitud (request) para la creación y actualización de registros de cementerios. Los modelos se definen utilizando Pydantic, que es una biblioteca de validación y serialización de datos en Python.

```
1 usage new *
28 class CemeteryCreateRequest(BaseModel:
29     fase: str
30     clave: str
31     prov: str
32     mun: str
33     ent: str
34     poblamiento: str
35     orden_cement: str
36     nombre: Optional[str]
37     titular: Optional[str]
38     distancia: Optional[float]
39     acceso: Optional[str]
40     capilla: Optional[str]
41     deposito: Optional[str]
42     ampliacion: Optional[str]
43     saturacion: Optional[float]
44     superficie: Optional[float]
45     acceso_s_ruedas: Optional[str]
46     crematorio: Optional[str]
47     cod: Optional[str]
48     borrado: Optional[str]
49     field_nomenclales: Optional[str]
50     lat: float
51     lng: float
52
53
54     1 usage new *
55     class CemeteryUpdateRequest(BaseModel:
56         gid: int
57         nombre: Optional[str]
58         titular: Optional[str]
59         distancia: Optional[float]
60         acceso: Optional[str]
61         capilla: Optional[str]
62         deposito: Optional[str]
63         ampliacion: Optional[str]
64         saturacion: Optional[float]
65         superficie: Optional[float]
66         acceso_s_ruedas: Optional[str]
67         crematorio: Optional[str]
```

Ambos modelos tienen campos opcionales para permitir la actualización parcial de los registros y reflejar el hecho de que no todos los campos necesitan ser proporcionados en cada solicitud. Esto ofrece flexibilidad en el manejo de los datos de cementerios dentro de la aplicación.

La actualización parcial se trata de un cambio que introdujimos el cliente cuando le entregamos el primer MPV del desarrollo. En un principio pensábamos que los usuarios disponían de toda la información a llenar sobre el terreno, y no dejábamos almacenar hasta que se insertarán todos los campos obligatorios del modelo, pero nos hicieron modificarlo para que pudieran guardar datos de forma parcial y luego en oficina finalizar completamente la inserción.

Para indicarle al usuario fácilmente la cantidad de equipamientos totalmente cumplimentados, hemos creado un campo calculado llamado *completo*, que se actualiza mediante un trigger que valida que los campos obligatorios no tengan valores vacíos o nulos en cada inserción o actualización del registro.

```
create trigger check_cementerio_completo_trigger before
insert
or
update
on
eiel.cementerio for each row execute function eiel.check_cementerio_completo()

CREATE OR REPLACE FUNCTION eiel.check_cementerio_completo()
RETURNS trigger
LANGUAGE plpgsql
AS $function$
BEGIN
    IF NEW.orden_cement IS NULL OR NEW.orden_cement = '' OR
    NEW.nombre IS NULL OR NEW.nombre = '' OR
    NEW.titular IS NULL OR NEW.titular = '' OR
    NEW.distancia IS NULL OR
    NEW.acceso IS NULL OR NEW.acceso = '' OR
    NEW.capilla IS NULL OR NEW.capilla = '' OR
    NEW.deposito IS NULL OR NEW.deposito = '' OR
    NEW.ampliacion IS NULL OR NEW.ampliacion = '' OR
    NEW.saturacion IS NULL OR
    NEW.superficie IS NULL OR
    NEW.acceso_s_ruedas IS NULL OR NEW.acceso_s_ruedas = '' OR
    NEW.crematorio IS NULL OR NEW.crematorio = ''
    THEN
        NEW.completo = FALSE;
    ELSE
        NEW.completo = TRUE;
    END IF;
    RETURN NEW;
END;
$function$
;
```

El siguiente código gestiona el borrado del equipamiento.

```
69      ...
70      @router.delete( path: "/{gid}", status_code=204)
71      async def delete_by_gid(
72          user: user_dependency,
73          gid: int,
74          db_postgres: db_dependency_postgres
75      ):
76          if user.get("role") not in ["admin", "operator"]:
77              raise HTTPException(status_code=402, detail="acción no permitida")
78
79          record = await exists_equipment(db_postgres=db_postgres, gid=gid, table_name=EIEL_TABLE_NAME)
80
81          cod = record.get("cod")
82
83          await release_nomenclature_layer(db_postgres, gid, table_name=EIEL_TABLE_NAME,
84                                           nomenclature_tables_names=NOMENCLATURE_TABLES_NAMES)
85
86          query = sql.text("""
87              delete from eiel.cementerio
88              where gid = :gid;
89          """)
90          values = {
91              "gid": gid
92          }
93          db_postgres.execute(query, values)
94          await register_user_access(db_postgres=db_postgres,
95                                      user_id=user.get("user_id"),
96                                      equipment=f"eiel.{EIEL_TABLE_NAME}",
97                                      function_name="delete_by_gid",
98                                      parameters={"gid": gid})
99
100         await delete_images_from_s3(db_postgres=db_postgres, gid=gid, tipo_equipamiento=TIPO_EQUIPAMIENTO)
101         await recalculate_cod(cod=cod, db_postgres=db_postgres, table_name=EIEL_TABLE_NAME, order_field=ORDER_FIELD)
102         db_postgres.commit()
```

Sólo puede eliminar equipamientos los usuarios con rol admin u operador. Al eliminar se recalcula la codificación ordenada del equipamiento, y se libera el punto de la capa auxiliar del nomenclátor para que vuelva a aparecer en el mapa del frontend.

Finalmente se elimina el registro de la base de datos, se registra el movimiento y si tiene imágenes asociadas las elimina de la base de datos y del bucket S3.

El siguiente endpoint devuelve el detalle del equipamiento solicitado junto con sus fotos asociadas. Mediante la función `check_municipality`, se chequea que si el usuario que consulta es del rol cityhall, sólo pueda ver el detalle de equipamientos de su municipio.

```
104     @router.get("/{gid}")
105     async def get_by_gid(
106         user: user_dependency,
107         gid: int,
108         db_postgres: db_dependency_postgres
109     ):
110         await exists_equipment(db_postgres, gid, table_name=EIEL_TABLE_NAME)
111         await check_municipality(db_postgres, gid, user, table_name=EIEL_TABLE_NAME)
112         images = await get_images(db_postgres=db_postgres, gid=gid, tipo_equipamiento=TIPO_EQUIPAMIENTO)
113
114         query = sql.text("""
115             select
116                 gid,
117                 nombre,
118                 titular,
119                 distancia,
120                 acceso,
121                 capilla,
122                 deposito,
123                 ampliacion,
124                 saturacion,
125                 superficie,
126                 acceso_s_ruedas,
127                 crematorio,
128                 cod,
129                 ST_AsGeoJSON(ST_Transform(geom, 4326)) as geom
130             from eiel.cementerio
131             where gid = :gid;
132         """)
133         values = {
134             "gid": gid
135         }
136         record = fetch_records_and_convert(db_postgres, query, values)
137         if len(record) > 0:
138             record = record[0]
139             record["images"] = images
140             return record
141         else:
142             raise HTTPException(status_code=404, detail="Equipamiento no encontrado")
143
```

Actualización del equipamiento

```
146     async def update_by_gid(
147         user: user_dependency,
148         gid: int,
149         equipment: CemeteryUpdateRequest,
150         db_postgres: db_dependency_postgres
151     ):
152         if gid != equipment.gid:
153             raise HTTPException(status_code=400, detail="El gid del equipamiento no coincide con el gid del objeto")
154
155         await exists_equipment(db_postgres, gid, table_name=EIEL_TABLE_NAME)
156         await check_municipality(db_postgres, gid, user, table_name=EIEL_TABLE_NAME)
157
158         query = sql.text(f"""
159             update eiel.cementerio
160             set nombre = :nombre,
161                 titular = :titular,
162                 distancia = :distancia,
163                 acceso = :acceso,
164                 capilla = :capilla,
165                 deposito = :deposito,
166                 ampliacion = :ampliacion,
167                 saturacion = :saturacion,
168                 superficie = :superficie,
169                 acceso_s_ruedas = :acceso_s_ruedas,
170                 crematorio = :crematorio
171             where gid = :gid;
172         """)
173         values = {
174             "gid": gid,
175             "nombre": equipment.nombre,
176             "titular": equipment.titular,
177             "distancia": equipment.distancia,
178             "acceso": equipment.acceso,
179             "capilla": equipment.capilla,
180             "deposito": equipment.deposito,
181             "ampliacion": equipment.ampliacion,
182             "saturacion": equipment.saturacion,
183             "superficie": equipment.superficie,
184             "acceso_s_ruedas": equipment.acceso_s_ruedas,
185             "crematorio": equipment.crematorio
186         }
187         try:
188             db_postgres.execute(query, values)
189             await register_user_access(db_postgres=db_postgres,
190                                         user_id=user.get("user_id"),
191                                         equipment=f"eiel.{EIEL_TABLE_NAME}",
192                                         function_name="update_by_gid",
193                                         parameters=equipment.model_dump())
194             db_postgres.commit()
195         except Exception as e:
196             raise HTTPException(status_code=400, detail=str(e.orig))
```

Tenemos el endpoint de mostrar todos los equipamientos de este tipo, en el que como en anteriores endpoints, se revisa el rol de usuario y si es ayuntamiento solo se devuelven los de su municipio configurado.

```

new *
199 @router.get("/")
200 async def get_all(
201     db_postgres: db_dependency_postgres,
202     user: user_dependency,
203 ):
204     return await get_all_my_records_by_table_name(db_postgres, user, EIEL_TABLE_NAME)
205

```

```

2 usages  ± jrgavilanes@gmail.com <jrgavilanes@gmail.com>
449 async def get_all_my_records_by_table_name(
450     db_postgres: db_dependency_postgres,
451     user: user_dependency,
452     table_name: str
453 ):
454     ORDER_FIELD = {...}
455     if user.get("role") not in ["cityhall"]:
456         query = sql.text(f"""
457             select
458                 gid,
459                 '{table_name}' as tabla,
460                 {ORDER_FIELD.get(table_name)} as orden,
461                 completo,
462                 concat('',{ORDER_FIELD.get(table_name)},') ', nombre) as mi_etiqueta,
463                 ST_AsGeoJSON(ST_Transform(geom, 4326)) AS geom
464             from eiel.{table_name};
465         """)
466         equipments = fetch_records_and_convert(db_postgres, query)
467     else:
468         query = sql.text(f"""
469             select
470                 gid,
471                 '{table_name}' as tabla,
472                 {ORDER_FIELD.get(table_name)} as orden,
473                 completo,
474                 concat('',{ORDER_FIELD.get(table_name)},') ', nombre) as mi_etiqueta,
475                 ST_AsGeoJSON(ST_Transform(geom, 4326)) AS geom
476             from eiel.{table_name} where mun = :mun;
477         """)
478         values = {
479             "mun": user.get("municipality")
480         }
481         equipments = fetch_records_and_convert(db_postgres, query, values)
482     if len(equipments) > 0:
483         geojson = {
484             "type": "FeatureCollection",
485             "features": [
486                 {
487                     "type": "Feature",
488                     "geometry": json.loads(feature["geom"]),
489                     "properties": {
490                         "gid": feature["gid"],
491                         "tabla": feature["tabla"],
492                         "orden": feature["orden"],
493                         "completo": feature["completo"],
494                         "mi_etiqueta": feature["mi_etiqueta"],
495                         "geom": feature["geom"]
496                     }
497                 }
498             ]
499             for feature in equipments
500         }
501     return equipments
502
503
504
505
506
507
508
509
510
511
512
513

```

El último endpoint es el de inserción de equipamiento, en el que sólo puede ser ejecutado por un rol operator o admin, nunca por un ayuntamiento. Si la inserción tiene éxito, se recalcula el código de ordenación, se registra en la base de datos, se refleja el movimiento en la tabla de auditorias y actualizamos las capas auxiliares del nomenclátor para que oculte esa coordenada.

```
207     @router.post( path: "/", status_code=201)
208     async def insert_equipment(
209         db_postgres: db_dependency_postgres,
210         equipment: CemeteryCreateRequest,
211         user: user_dependency,
212     ):
213         if user.get("role") not in ["admin", "operator"]:
214             raise HTTPException(status_code=402, detail="Rol ayuntamiento no puede insertar equipamientos")
215
216         await recalculate_cod(db_postgres=db_postgres,
217                               cod=equipment.cod,
218                               table_name=EIEL_TABLE_NAME,
219                               order_field=ORDER_FIELD)
220
221         result = await get_new_code(db_postgres=db_postgres, cod=equipment.cod, table_name=EIEL_TABLE_NAME)
222         equipment.cod = result.get("new_cod")
223         equipment.orden_cement = result.get("order_field")
224         gid = await insert_equipment_in_db(equipment, db_postgres)
225         parameters = OrderedDict()
226         parameters["gid"] = gid.get("inserted_id")
227         parameters.update(equipment.model_dump())
228
229         await register_user_access(db_postgres=db_postgres,
230                                    user_id=user.get("user_id"),
231                                    equipment=f"eiel.{EIEL_TABLE_NAME}",
232                                    function_name="insert_equipment",
233                                    parameters=parameters)
234
235         await update_record_geom_with_nomecalles_geom(
236             db_postgres=db_postgres,
237             eiel_table_name=EIEL_TABLE_NAME,
238             eiel_gid=gid.get("inserted_id"),
239             field_nomecalles=equipment.field_nomecalles
240         )
241
242         db_postgres.commit()
243         return gid
244
245
```

Una vez creado el nuevo enrutador del equipamiento y enlazado en el programa principal,

```
main.py ×
1 import os
2 import uvicorn
3 from fastapi import FastAPI
4
5 from routers import auth, helpers, uploader, cemeteries
6
7 PRODUCTION = os.getenv('PRODUCTION', "false")
8 if PRODUCTION.lower().strip() == "true":
9     app = FastAPI(debug=False, docs_url=None, redoc_url=None)
10 else:
11     app = FastAPI(debug=True, docs_url="/api/docs", openapi_url="/api/openapi.json")
12
13 app.include_router(auth.router)
14 app.include_router(helpers.router)
15 app.include_router(uploader.router)
16 app.include_router(cemeteries.router)
17
18 # jrgavilanes@gmail.com <jrgavilanes@gmail.com>
19 @app.get("/api/")
20 async def is_live():
21     return {"status": "OK"}
22
23
24 if __name__ == '__main__':
25     uvicorn.run(app, port=7000, host="127.0.0.1")
26
```

ya podremos operar con él desde la web de soporte.

The screenshot shows the Redoc API documentation interface. It displays three main sections: **/api/upload**, **/api/cemeteries**, and **default**.

- /api/upload** section:
 - DELETE /api/upload/{idef}** Delete File
 - PUT /api/upload/{idef}** Update File
 - POST /api/upload/{equipment_type}/{gid}/{cod}/{lat}/{lng}** Upload File
- /api/cemeteries** section:
 - DELETE /api/cemeteries/{gid}** Delete By Gid
 - GET /api/cemeteries/{gid}** Get By Gid
 - PUT /api/cemeteries/{gid}** Update By Gid
 - GET /api/cemeteries/** Get All
 - POST /api/cemeteries/** Insert Equipment
- default** section:
 - GET /api/** Is Live

Tras integrar todos los endpoints de los 14 equipamientos, este será el aspecto que presentará el asistente.

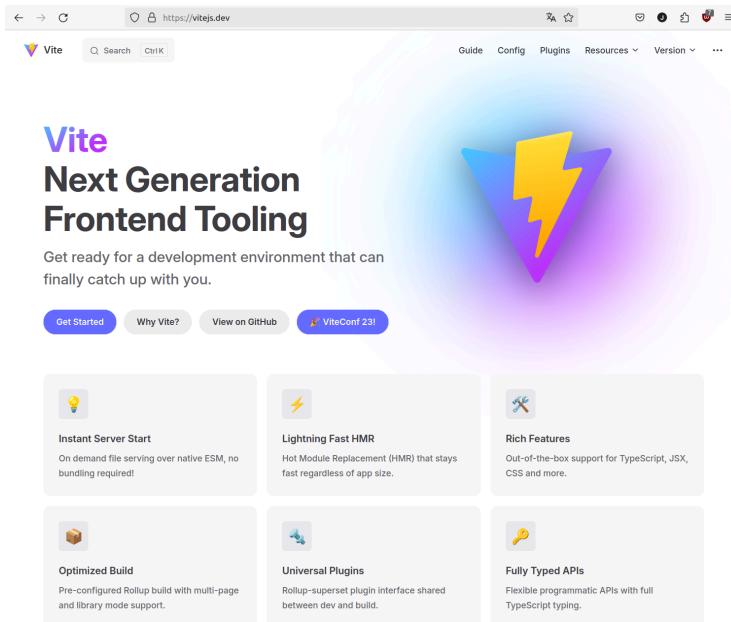
/api/cemeteries		
DELETE	/api/cemeteries/{gid}	Delete By Gid
GET	/api/cemeteries/{gid}	Get By Gid
PUT	/api/cemeteries/{gid}	Update By Gid
GET	/api/cemeteries/	Get All
POST	/api/cemeteries/	Insert Equipment
/api/assistance-centers		
DELETE	/api/assistance-centers/{gid}	Delete By Gid
GET	/api/assistance-centers/{gid}	Get By Gid
PUT	/api/assistance-centers/{gid}	Update By Gid
GET	/api/assistance-centers/	Get All
POST	/api/assistance-centers/	Insert Equipment
/api/civil-protections		
DELETE	/api/civil-protections/{gid}	Delete By Gid
GET	/api/civil-protections/{gid}	Get By Gid
PUT	/api/civil-protections/{gid}	Update By Gid
GET	/api/civil-protections/	Get All
POST	/api/civil-protections/	Insert Equipment
/api/cultural-centers		
DELETE	/api/cultural-centers/{gid}	Delete By Gid
GET	/api/cultural-centers/{gid}	Get By Gid
PUT	/api/cultural-centers/{gid}	Update By Gid
GET	/api/cultural-centers/	Get All
POST	/api/cultural-centers/	Insert Equipment
/api/educational-centers		
DELETE	/api/educational-centers/{gid}	Delete By Gid
GET	/api/educational-centers/{gid}	Get By Gid
PUT	/api/educational-centers/{gid}	Update By Gid
GET	/api/educational-centers/	Get All
POST	/api/educational-centers/	Insert Equipment
/api/landfills		
DELETE	/api/landfills/{gid}	Delete By Gid
GET	/api/landfills/{gid}	Get By Gid

En el siguiente apartado ya pasaremos a la parte frontend del proyecto.

3.3. Desarrollo Frontend

Vite es un potente y rápido constructor de proyectos JavaScript diseñado especialmente para Vue.js y React. Ofrece una configuración predeterminada óptima, incluyendo soporte para preprocesadores de CSS, así como TypeScript.

Vite se ha convertido en una herramienta popular en el ecosistema de Vue.js debido a sus características únicas. Una de las principales razones es su rapidez en el tiempo de desarrollo. Utiliza una arquitectura de servidor de desarrollo con cargas instantáneas, lo que significa que el tiempo de compilación y recarga es extremadamente rápido. Esto se debe a su capacidad para utilizar ESM (Módulos JavaScript) nativos, lo que permite una resolución de dependencias muy rápida.



Además, ofrece un soporte nativo para ESM tanto en tiempo de desarrollo como de producción. Esto significa que puedes importar módulos de JavaScript de forma nativa, sin necesidad de transpilación, lo que mejora aún más la velocidad de desarrollo y la eficiencia en el consumo de recursos.

Su HMR (Hot Module Replacement), proporciona una experiencia de desarrollo fluida, donde los cambios realizados en el código se reflejan instantáneamente en el navegador sin necesidad de recargar la página completa. Esto mejora significativamente el flujo de trabajo y la productividad del desarrollador.

3.3.1. Configurar entorno de desarrollo

```
git clone git@github.com:jrgavilanes/TFG-EIEL.git  
git checkout feature/06_configuracion_entorno_frontend
```

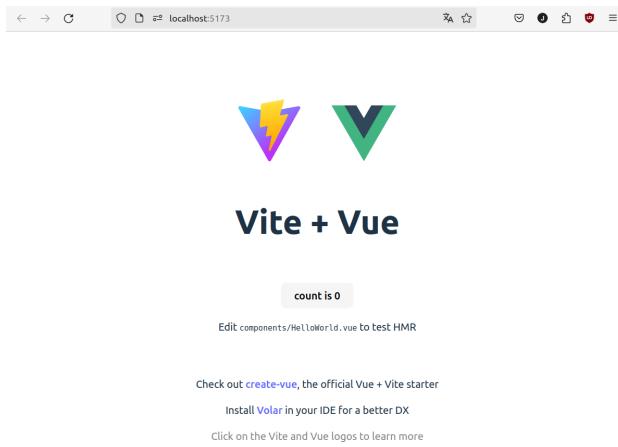


Introduciremos el código del frontend en el directorio **/frontend** del proyecto, para ello, seguiremos las instrucciones desde la propia página web de vite (<https://vitejs.dev/guide/>) para construir un proyecto base, generar toda la estructura de ficheros necesaria, y arrancar el entorno de desarrollo en el puerto 5173

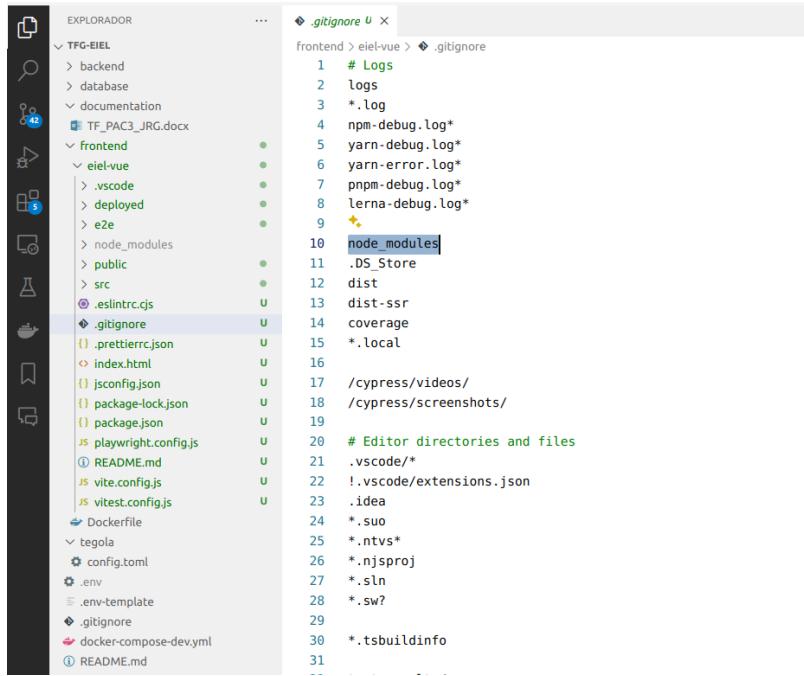
The screenshot shows the Visual Studio Code interface with the title bar "tfg-eiel - Visual Studio Code". The left sidebar shows a file tree with a project named "TFG-EIEL" containing "backend", "database", "documentation" (with a file "TF_PAC3_JRG.docx"), and "frontend" (which is expanded to show "eiel-vue" with files ".vscode", "node_modules", "public", ".gitignore", "index.html", "package-lock.json", "package.json", "README.md", "vite.config.js", "Dockerfile", "tegola", "config.toml", ".env", and "esquema"). The right side has tabs for "PROBLEMAS", "SALIDA", "CONSOLA DE DEPURACIÓN", "TERMINAL", and "PUERTOS". The "TERMINAL" tab is active, displaying the following command-line session:

```
janrax@janrax-home-linux:~/Escritorio/code/tfg-eiel/frontend$ node -v  
v20.11.1  
janrax@janrax-home-linux:~/Escritorio/code/tfg-eiel/frontend$ npm create vite@latest eiel-vue -- --template vue  
Scaffolding project in /home/janrax/Escritorio/code/tfg-eiel/frontend/eiel-vue...  
Done. Now run:  
cd eiel-vue  
npm install  
npm run dev  
janrax@janrax-home-linux:~/Escritorio/code/tfg-eiel/frontend$ cd eiel-vue/  
janrax@janrax-home-linux:~/Escritorio/code/tfg-eiel/frontend/eiel-vue$ npm install  
added 27 packages, and audited 28 packages in 2s  
4 packages are looking for funding  
  run `npm fund` for details  
found 0 vulnerabilities  
janrax@janrax-home-linux:~/Escritorio/code/tfg-eiel/frontend/eiel-vue$ npm run dev  
> eiel-vue@0.0.0 dev  
> vite  
  
VITE v5.2.11 ready in 349 ms  
+ Local: http://localhost:5173/  
+ Network: use --host to expose  
+ press h + enter to show help
```

Ya podremos acceder a la página inicial de prueba del entorno recién instalado. Cualquier cambio que realicemos en los ficheros del programa, se verán reflejado inmediatamente en la página.



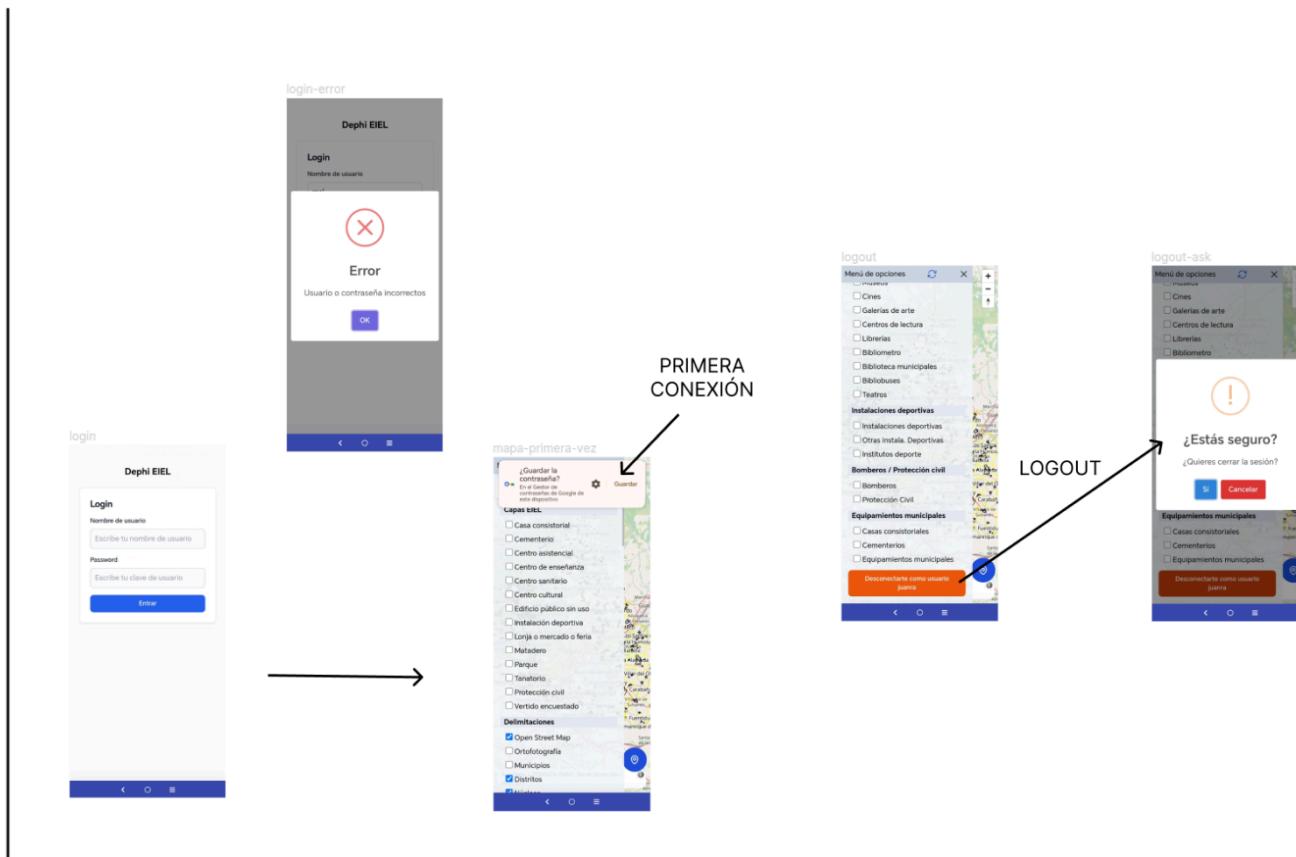
No debemos olvidar comprobar que el directorio `/node_modules`, que ha sido generado automáticamente al construir el proyecto, está en `.gitignore` para que no se haga seguimiento de su contenido, ya que éste se generará a partir del fichero `package.json` cada vez que construyamos un proyecto nuevo al ejecutar `npm install`.



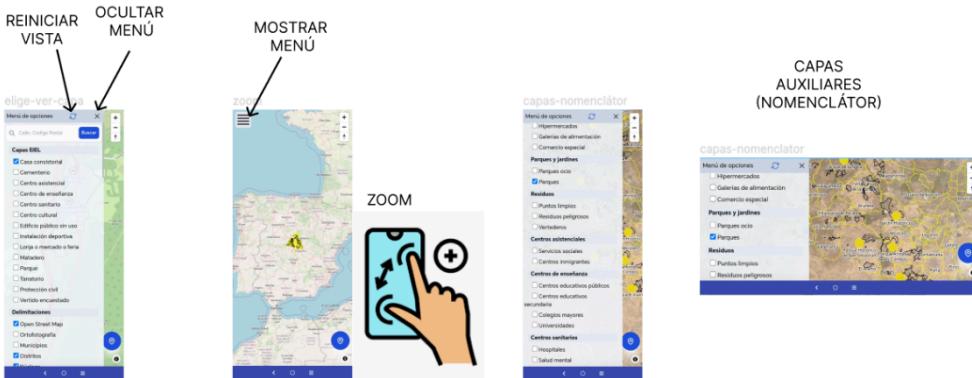
```
frontend > eiel-vue > .gitignore
1 # Logs
2 logs
3 *.log
4 npm-debug.log*
5 yarn-debug.log*
6 yarn-error.log*
7 pnpm-debug.log*
8 lerna-debug.log*
9 +
10 node_modules
11 .DS_Store
12 dist
13 dist-ssr
14 coverage
15 *.local
16
17 /cypress/videos/
18 /cypress/screenshots/
19
20 # Editor directories and files
21 .vscode/*
22 !.vscode/extensions.json
23 .idea
24 *.suo
25 *.ntvs*
26 *.njsproj
27 *.sln
28 *.sw?
29
30 *.tsbuildinfo
31
```

3.3.2. Interfaz UX/UI

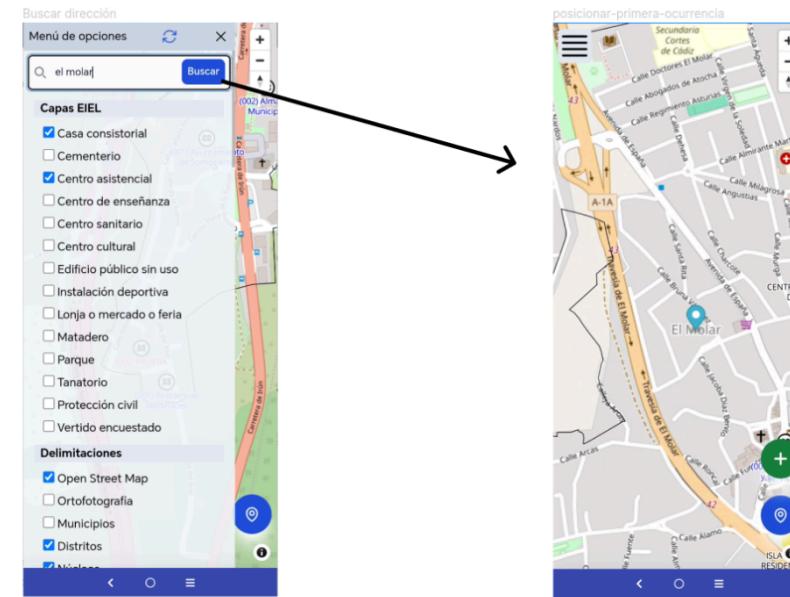
LOGIN



NAVEGACIÓN BÁSICA



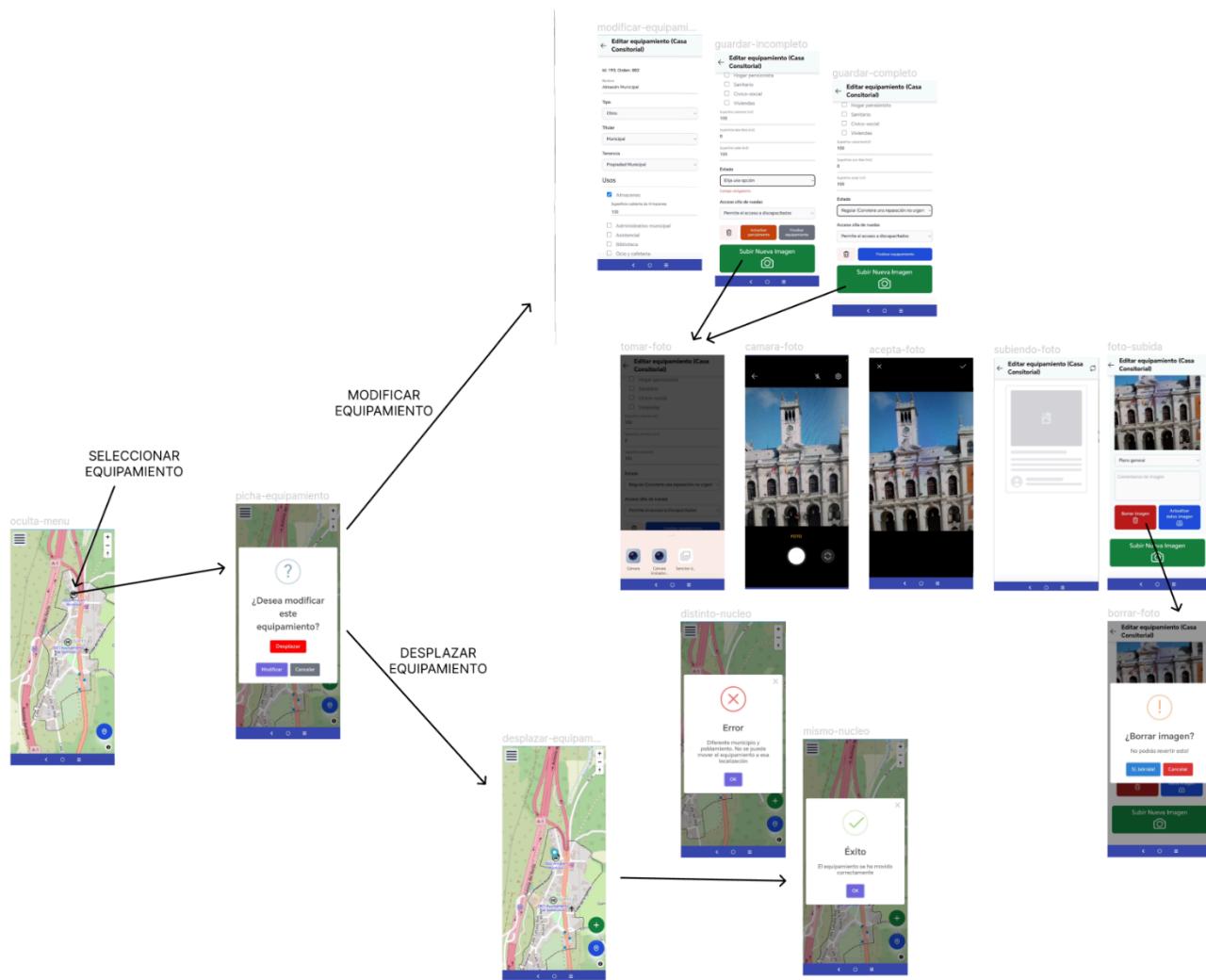
BUSCAR DIRECCIÓN



CREAR EQUIPAMIENTO



EDITAR EQUIPAMIENTO

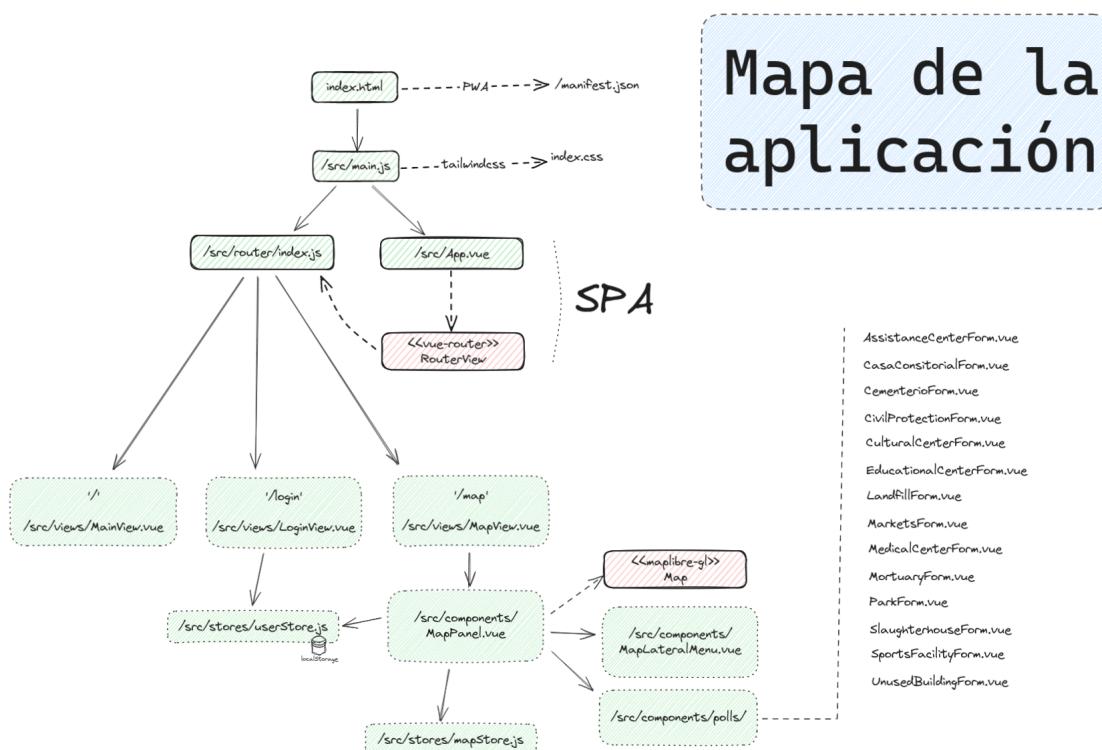


3.3.3. Vistas y formularios

```
git clone git@github.com:jrgavilanes/TFG-EIEL.git
git feature/07_desarrollo_vistas_y_componentes
```



En esta sección desarrollaremos las vistas del proyecto partiendo del siguiente diagrama.

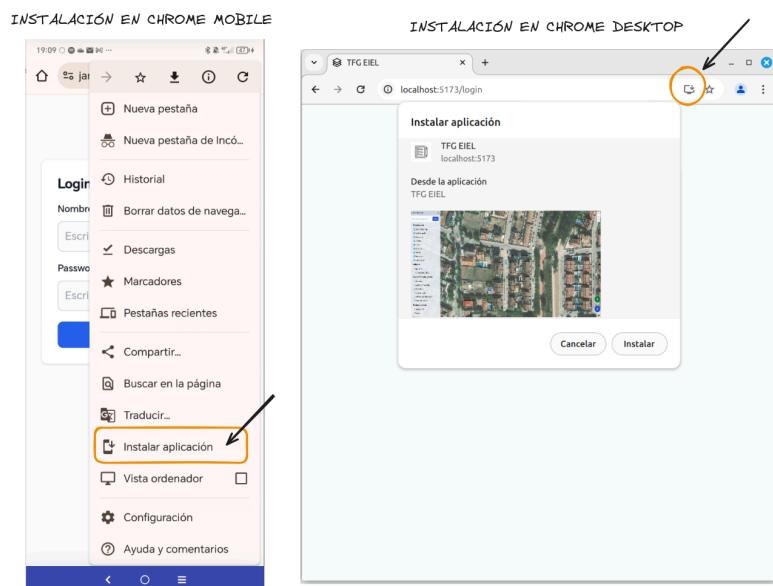


Cuando un navegador web visite la url de la aplicación, cargará por defecto el fichero `index.html`.

```
index.html x
frontend > eiel-vue > index.html > html
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="UTF-8">
5      <link rel="icon" type="image/svg+xml" href="/icon.svg" />
6      <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1.0, user-scalable=no">
7      <link rel="manifest" href="/manifest.json">
8      <title>TFG EIEL</title>
9    </head>
10   <body class="overscroll-none">
11     <div id="app"></div>
12     <script type="module" src="/src/main.js"></script>
13   </body>
14 </html>
```

Este será el punto de partida de la aplicación, donde enlazamos con el fichero `manifest.json` que es donde definimos todas las propiedades de la PWA, y hacemos que la aplicación web sea instalable en navegadores Chrome y Safari, ocultando la barra de navegación y dándole el aspecto de APP.

```
(manifest.json) manifest.json x
Frontend > eiel-vue > public > manifest.json > ...
1  {
2    "id": "/",
3    "name": "TFG EIEL",
4    "short_name": "EIEL",
5    "description": "TFG EIEL",
6    "start_url": "/",
7    "display": "standalone",
8    "orientation": "any",
9    "background_color": "#ffffff",
10   "theme_color": "##007bff",
11   "icons": [
12     {
13       "src": "/images/pwa-logo.png",
14       "sizes": "256x256",
15       "type": "image/png"
16     }
17   ],
18   "screenshots": [
19     {
20       "src": "/images/pwa-horizontal.jpg",
21       "sizes": "1280x720",
22       "type": "image/jpg",
23       "form_factor": "wide"
24     },
25     {
26       "src": "/images/pwa-vertical.jpg",
27       "sizes": "720x1280",
28       "type": "image/jpg"
29     }
30   ]
31 }
```



El fichero `index.html` también enlazará con `src/main.js`, que configura la aplicación Vue.js, integrando el complemento Pinia para manejar el estado centralizado y el enrutador para gestionar la navegación entre las distintas vistas de la aplicación. Luego, monta la aplicación en el elemento HTML con el

id `app`, lo que permite que la interfaz de usuario se renderice en esa ubicación específica dentro del documento HTML. En `index.css` se carga Tailwind CSS para aplicar estilos globales a la aplicación.

```
JS main.js  X
frontend > eiel-vue > src > JS main.js > ...
1 import { createApp } from 'vue'
2 import { createPinia } from 'pinia'
3
4 import App from './App.vue'
5 import router from './router'
6
7 import './index.css'
8
9 const app = createApp(App)
10
11 app.use(createPinia())
12 app.use(router)
13
14 app.mount('#app')
```

El archivo `App.vue` configura la estructura principal de la aplicación Vue.js.

```
▼ App.vue  X
frontend > eiel-vue > src > ▼ App.vue > {} script setup
1 <script setup>
2 import { RouterView } from 'vue-router'
3 </script>
4
5 <template>
6   <div class="w-full h-full select-none">
7     |   <RouterView />
8   </div>
9 </template>
10
11 <style scoped></style>
```

Utiliza `RouterView` de Vue Router para renderizar dinámicamente las vistas correspondientes a las rutas definidas en el enrutador. La plantilla define un contenedor que ocupa toda la ventana y el componente `RouterView` se utiliza para mostrar las vistas según la navegación del usuario.

El archivo `router/index.js` configura el enrutador para la aplicación. Aquí se definen las rutas de la aplicación, donde cada ruta tiene un `path` que especifica la URL correspondiente y un componente asociado que se renderizará cuando se acceda a esa ruta. Por ejemplo, la ruta `'/'` corresponde a la página principal y se renderiza el componente `Main`. La ruta `'/login'` renderiza el componente `Login`, y la ruta `'/map'` renderiza el componente `MapView`.

```
JS index.js  x
frontend > eiel-vue > src > router > JS index.js > ...
1 import { createRouter, createWebHistory } from 'vue-router'
2
3 import MapView from '@/views/MapView.vue'
4 import Login from '@/views/LoginView.vue'
5 import Main from '@/views/MainView.vue'
6
7 const router = createRouter({
8   history: createWebHistory(import.meta.env.BASE_URL),
9   routes: [
10     {
11       path: '/',
12       component: Main,
13     },
14     {
15       path: '/login',
16       name: 'Login',
17       component: Login,
18     },
19     {
20       path: '/map',
21       name: 'MapView',
22       component: MapView
23     },
24   ]
25 })
26
27 export default router
```

El archivo `MainView.vue` se encarga de dirigir la navegación del usuario según el estado de la sesión. Utiliza la función `useRouter` para acceder al enrutador de Vue Router y la función `useUserStore` para acceder al almacenamiento de datos del usuario. Comprueba si hay un token de usuario almacenado. Si no hay un token, redirige al usuario a la página de inicio de sesión (`'/login'`). Si hay un token, redirige al usuario a la página del mapa (`'/map'`). La plantilla está vacía ya que todo el trabajo funcional se maneja en la sección de configuración `<script setup>`.

```
▼ MainView.vue ×
frontend > eiel-vue > src > views > ▼ MainView.vue > {} template > ⚙ div
  1  <script setup>
  2
  3  import { useRouter } from 'vue-router';
  4  const router = useRouter();
  5
  6  import { useUserStore } from '@/stores/userStore';
  7  const userStore = useUserStore();
  8
  9  userStore.loadToken();
 10
 11 if (!userStore.user.token) {
 12   router.replace('/login');
 13 } else {
 14   router.replace('/map');
 15 }
 16
 17 </script>
 18
 19 <template>
 20   <div>
 21     </div>
 22 </template>
```

`LoginView.vue` representa la vista de inicio de sesión de la aplicación. En la sección `<script setup>`, importa el módulo `Swal` de SweetAlert2 para mostrar alertas, y utiliza la función `ref` para crear referencias reactivas a las variables `username` y `password`.

Utiliza `useUserStore` para acceder al almacenamiento de datos del usuario. Define la función `login`, que realiza una solicitud de inicio de sesión al servidor y gestiona la respuesta.

La plantilla HTML en la sección `<template>` muestra un formulario de inicio de sesión con campos para el nombre de usuario y la contraseña. Cuando se envía el formulario, se llama a la función `login`. Si hay un error en la autenticación, se muestra una alerta de error utilizando SweetAlert2.



```
▼ LoginView.vue M ●
frontend > eiel-vue > src > views > ▼ LoginView.vue > {} template
  1  <script setup>
  2  import Swal from 'sweetalert2';
  3
  4  import { ref } from 'vue';
  5
  6  import { useUserStore } from '@/stores/userStore';
  7  const userStore = useUserStore();
  8
  9  const username = ref('');
 10  const password = ref('');
 11
 12  const login = async () => {
 13
 14    const response = await fetch(`api/auth/validate`, {
 15      method: 'POST',
 16      headers: {
 17        'Content-Type': 'application/json',
 18      },
 19      body: JSON.stringify({
 20        name: username.value,
 21        password: password.value,
 22      }),
 23    );
 24
 25    if (!response.ok) {
 26      Swal.fire({
 27        icon: 'error',
 28        title: 'Error',
 29        text: 'Usuario o contraseña incorrectos',
 30      });
 31      return;
 32    }
 33
 34    const jwt = await response.json();
 35    userStore.login(jwt);
 36
 37  };
 38  </script>
```

En `userStore.js` he encapsulado la mayoría de la lógica de negocio relacionada con la autenticación y la gestión del usuario.

Esto incluye funciones para iniciar y cerrar sesión, así como para cargar automáticamente el token de inicio de sesión al cargar la aplicación. Esta práctica de encapsulamiento en los almacenes de datos ayuda a mantener un código modular y fácil de mantener, al tiempo que proporciona un lugar centralizado para gestionar el estado de la aplicación relacionado con el usuario.

```
JS userStore.js M ×
frontend > eiel-vue > src > stores > JS userStore.js > ...
1 import { computed, reactive } from 'vue'
2 import { defineStore } from 'pinia'
3 import { useRouter } from 'vue-router';
4
5 export const useUserStore = defineStore('userStore', () => {
6
7   const router = useRouter();
8
9   const data = reactive({ ... });
10
11
12   const user = computed(() => {
13     return {
14       id: data.id,
15       username: data.username,
16       role: data.role,
17       token: data.token,
18       municipality: data.municipality,
19       is_desktop: data.is_desktop,
20     };
21   });
22
23
24   const login = (jwt) => {
25     localStorage.setItem('token', jwt);
26     const tokenParts = jwt.split('.');
27     const decodedPayload = atob(tokenParts[1]);
28     const payloadObj = JSON.parse(decodedPayload);
29     data.id = payloadObj.id;
30     data.username = payloadObj.username;
31     data.role = payloadObj.role;
32     data.token = jwt;
33     data.municipality = payloadObj.municipality;
34     data.is_desktop = payloadObj.is_desktop;
35     router.replace({ name: 'MapView' });
36   }
37
38
39   const logout = () => {
40     ...
41   }
42
43   const loadToken = () => {
44     const jwt = localStorage.getItem('token');
45     if (jwt) {
46       login(jwt);
47       return;
48     }
49     router.replace({ name: 'Login' });
50   }
51
52
53   return {
54     login,
55     loadToken,
56     logout,
57     user,
58   };
59
60
61
62
63
64
65
66
67
68 }
```

En `MapPanel.vue`, se presenta la parte principal de la aplicación: el mapa donde se georeferencian los equipamientos registrados, aprovechando los almacenes de datos (`MapStore.vue`) para gestionar la lógica de negocio de manera eficiente.

Se encarga de la configuración del mapa utilizando la biblioteca `maplibre-gl`, definiendo las capas base y vectoriales, así como los controles de navegación, y cargando datos del mapa desde fuentes externas.

Facilita la interacción del usuario con el mapa, permitiendo acciones como agregar marcadores o activar el menú lateral para explorar opciones adicionales. También posibilita el inicio de nuevas encuestas para registrar diferentes tipos de equipamiento en el mapa, presentando formularios específicos para cada tipo de equipamiento seleccionado.



```
▼ MapPanel.vue M X
Frontend > eiel-vue > src > components > ▼ MapPanel.vue > {} script setup
1 <script setup>
2 import { onMounted } from 'vue'
3 import Swal from 'sweetalert2';
4 import { Map, NavigationControl } from 'maplibre-gl'
5 import 'maplibre-gl/dist/maplibre-gl.css'
6
7 import iconMenu from './icons/iconMenu.vue'
8 import MapLateralMenu from './MapLateralMenu.vue'
9
10 import { useMapStore } from '../stores/mapStore.js'
11
12 import CementerioForm from './polls/CementerioForm.vue'
13 import CasaConsitorialForm from './polls/CasaConsitorialForm.vue'
14 import AssistanceCenterForm from './polls/AssistanceCenterForm.vue'
15 import ParkForm from './polls/ParkForm.vue';
16 import EducationalCenterForm from './polls/EducationalCenterForm.vue';
17 import MedicalCenterForm from './polls/MedicalCenterForm.vue';
18 import CulturalCenterForm from './polls/CulturalCenterForm.vue';
19 import UnusedBuildingForm from './polls/UnusedBuildingForm.vue';
20 import SportsFacilityForm from './polls/SportsFacilityForm.vue';
21 import MarketsForm from './polls/MarketsForm.vue';
22 import SlaughterhouseForm from './polls/SlaughterhouseForm.vue';
23 import MortuaryForm from './polls/MortuaryForm.vue';
24 import CivilProtectionForm from './polls/CivilProtectionForm.vue';
25 import LandfillForm from './polls/LandfillForm.vue';
26
27 import { useUserStore } from '@/stores/userStore';
28 const userStore = useUserStore();
29
30
31 const mapStore = useMapStore()
32
33 > async function checkToken() { ...
55 }
56
57 > const startPoll = async () => { ...
103 }
104
105 onMounted(() => {
106   checkToken();
107   mapStore.map = new Map({
108     container: 'map',
109     style: {
110       version: 8,
111       glyphs: 'https://demotiles.maplibre.org/font/{fontstack}/{range}.pbf',
112       sources: {
113         open_street_map: {
114           type: "raster",
115           tiles: [
116             "https://tile.openstreetmap.org/{z}/{x}/{y}.png"
117           ],
118           tileSize: 256,
119           attribution:
120             'TFG EIEL ${new Date().getFullYear()}. Server:${window.location.hostname}'}
121       }
122     }
123   });
124 }
125
126 > <NavigationControl position="top-right" :map="mapStore.map" />
127 <MapLateralMenu :store="userStore" />
```

El archivo `CasaConsistorialForm.vue` es un componente, que hemos tomado a modo de ejemplo, y está diseñado para manejar la información específica de las casas consistoriales, pero sigue una estructura general que es similar en todos los equipamientos.

El componente comienza definiendo sus propiedades, que incluyen datos como la latitud y longitud del equipamiento, así como información específica del tipo de equipamiento y otros campos necesarios para su identificación.

Luego, importa varios módulos y funciones de utilidad y almacenes de datos específicos para el mapa y el usuario.

Incluye una serie de funciones para realizar acciones como obtener información sobre un equipamiento a partir de su ID o coordenadas, eliminar el equipamiento junto con todas sus fotos, eliminar imágenes asociadas al equipamiento, actualizar información de una imagen, y enviar información del equipamiento al servidor para su almacenamiento o actualización.

También se proporciona una función para manejar el cambio de archivos de imagen, así como una función para cerrar el formulario de equipamiento.

```
▼ CasaConsistorialForm.vue M ×
frontend > eiel-vue > src > components > polls > ▼ CasaConsistorialForm.vue > {} script setup
  1 <script setup>
  2
  3 const props = defineProps({
  4   lat: Number,
  5   lng: Number,
  6   geom: String,
  7   mi_etiqueta: String,
  8   tipo_equipamiento: String,
  9   field_nomenclatura: String,
 10   gid: String
 11 })
 12
 13 import { ref, reactive, onMounted } from 'vue';
 14 import { useRouter } from 'vue-router';
 15 import {
 16   aa_estado,
 17   aa_acceso_s_ruedas, aa_fase
 18 } from '@/composables/dominioComposable.js';
 19
 20 import {
 21   casa_consistorial_tenencia,
 22   casa_consistorial_tipo, casa_consistorial_titular, casa_consistorial_uso
 23 } from '@/composables/casaConsistorialComposable.js';
 24
 25 import Swal from 'sweetalert2';
 26
 27 import { useMapStore } from '@/stores/mapStore.js'
 28 const mapStore = useMapStore();
 29
 30 import { useUserStore } from '@/stores/userStore';
 31 const userStore = useUserStore();
 32 const router = useRouter();
 33
 34 const isLoading = ref(false);
 35
 36 > const formData = reactive({-
 62 });
 63
 64 const isValidForm = () => {
 65
 66   return formData.nombre &&
 67     formData.titular &&
 68     formData.tipo &&
 69     formData.acceso_s_ruedas &&
 70     formData.nombre && formData.nombre.length <= 50 &&
 71     formData.tenencia &&
 72     !isNaN(formData.s_cubi) && formData.s_cubi &&
 73     !isNaN(formData.s_aire) && formData.s_aire &&
 74     !isNaN(formData.s_sola) && formData.s_sola
 75
 76 }
 77
 78 > const getInfoByEquipmentId = async (gid) => {-
132 }
```

3.4. Pruebas manuales

3.5. Crear infraestructura

4. Glosario

EIEL: Encuesta de Infraestructura y Equipamientos Locales

GIS: Geographic Information System o SIG, sistema de información geográfica. Un SIG captura, almacena, analiza, gestiona y representa datos vinculados a una locación.

NOMENCLÁTOR: Registro oficial que contiene una lista detallada de nombres de lugares, calles, vías, edificios u otras entidades geográficas dentro de una región específica.

PWA: Aplicaciones Web Progresivas

5. Bibliografía

- [1] https://mpt.gob.es/politica-territorial/local/coop_econom_local_estado_fondos_europeos/eiel.html (Marzo. 2024)
- [2] Desarrollo en cascada: https://es.wikipedia.org/wiki/Desarrollo_en_cascada (Marzo. 2024)
- [3] Desarrollo ágil: <https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/agile-development/> (Marzo. 2024)
- [4] Objetivos del desarrollo sostenible (ODS):
<https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/> (Marzo. 2024)
- [5] Ostadabbas, H., Weippert, H., & F-J Behr. (2019). *Database Transformation, Cadastre Automatic Data Processing In Qgis And Implementation In Web Gis.* Gottingen: Copernicus GmbH.
doi:<https://doi.org/10.5194/isprs-archives-XLII-4-W14-175-2019>
- [6] LUACES, Miguel R., et al. WebEIEL: un SIG basado en Web para la EIEL.
- [7] GONZÁLEZ, Pedro A., et al. gisEIEL, un SIG para la explotación de la EIEL de A Coruña.
- [8] ISERN, Àgueda, et al. Implementación del aplicativo gisEIEL en el Consell Insular de Mallorca.
- [9] KURIA, Ezekiel; KIMANI, Stephen; MINDILA, Agnes. A framework for web GIS development: a review. *International Journal of Computer Applications*, 2019, vol. 178, no 16, p. 6-10.
- [10] <https://cartolab.udc.es/gvsig-eiel/index.html> (Marzo 2024)
- [11] https://es.wikipedia.org/wiki/Aplicaci%C3%B3n_web (Abril 2024)
- [12] Tandel, Sayali & Jamadar, Abhishek. (2018). Impact of Progressive Web Apps on Web App Development. 10.15680/IJIRSET.2018.0709021.
- [13] https://es.wikipedia.org/wiki/Sistema_de_informaci%C3%B3n_geogr%C3%A1fica (Abril 2024)
- [14] [https://es.wikipedia.org/wiki/Docker_\(software\)](https://es.wikipedia.org/wiki/Docker_(software)) (Abril 2024)
- [15] <https://latitudetechnolabs.com/aws-vs-google-cloud-vs-digital-ocean/> (Abril 2024)
- [16] <https://www.martinfowler.com/articles/continuousIntegration.html> (Abril 2024)
- [17] https://es.wikipedia.org/wiki/Entrega_continua (Abril 2024)
- [18] <https://docs.github.com/es/actions/learn-github-actions/understanding-github-actions> (Abril 2024)
- [19] <https://docs.docker.com/compose/intro/features-uses/> (Abril 2024)
- [20] FLUJO DE TRABAJO DE GITFLOW:
<https://www.atlassian.com/es/git/tutorials/comparing-workflows/gitflow-workflow> (Abril 2024)