| **Desarrollo App Web GIS para EIEL.** **Encuesta de Infraestructura y Equipamientos Locales** | |
| --- | --- |
|  | |
| Shape, rectangle  Description automatically generated | **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  **08/04/2024** |

  
Esta obra está sujeta a una licencia de Reconocimiento-NoComercial-SinObraDerivada [3.0 España de Creative Commons](http://creativecommons.org/licenses/by-nc-nd/3.0/es/)

**FICHA DEL TRABAJO FINAL**

| **Título del trabajo:** | *Desarrollo de aplicación Web GIS para ejecutar la Encuesta de Infraestructura y Equipamiento Local (EIEL) de la comunidad de Madrid.* |
| --- | --- |
| **Nombre del autor:** | *Juan Ramón Gavilanes Sánchez* |
| **Nombre del consultor/a:** | *Joaquín Torres Sospedra* |
| **Nombre del PRA:** | *Antoni Pérez Navarro* |
| **Fecha de entrega (mm/aaaa):** | *07/2024* |
| **Titulación o programa:** | Grado en ingeniería informática |
| **Área del Trabajo Final:** | *Servicios basados en localización y espacios inteligentes* |
| **Idioma del trabajo:** | *Castellano* |
| Palabras clave | *GIS, EIEL, Movilidad.* |
| **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 modelos 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](#_heading=h.ys47yd76hvkl)

[1. Introducción 1](#_heading=h.30j0zll)

[1.1. Contexto y justificación del Trabajo 1](#_heading=)

[1.2. Objetivos del Trabajo 2](#_heading=)

[1.3. Impacto en sostenibilidad, ético-social y de diversidad 3](#_heading=)

[1.4. Enfoque y método seguido 4](#_heading=)

[1.5. Planificación del Trabajo 6](#_heading=)

[1.6. Breve sumario de productos obtenidos 9](#_heading=)

[1.7. Breve descripción de los otros capítulos de la memoria 10](#_heading=)

[2. Estado del arte. 11](#_heading=h.ixgha2pt582n)

[2.1. Levantamiento de requerimientos 14](#_heading=h.2cy3xv1jd36p)

[2.2. Diseño de la solución 15](#_heading=h.br8k6n59f0jm)

[2.2.1. Aplicación Web 15](#_heading=h.17dp8vu)

[2.2.2. Aplicación Web Progresiva o Progressive Web App (PWA) 19](#_heading=h.ffoklvkjx49)

[2.2.3. Sistemas de información geográfica (SIG) 20](#_heading=h.s47rtwlmz9lj)

[2.2.4. MapLibre GL 21](#_heading=h.h0f8onvfh27)

[2.2.5. Tegola tileserver. 22](#_heading=h.g37lcaqrdkc7)

[2.2.6. Contenedores Linux con Docker 24](#_heading=h.he1ws5ar0eve)

[2.2.7. Servidor Privado Virtual (VPS) 25](#_heading=h.aftd83pd8jbp)

[2.2.8. GitHub y GitHub Actions 27](#_heading=h.sgim0ufmc0z9)

[2.2.9. Propuesta arquitectónica del sistema. 28](#_heading=h.endq606whidt)

[3. Implementación 29](#_heading=h.k48oe0bocgw5)

[3.1. Preparar modelo de datos 30](#_heading=h.39abl4krnmik)

[3.1. Orígenes de datos 30](#_heading=h.t672hk2qyk5t)

[3.1.1. Modelo de datos oficial 30](#_heading=h.vyhub08bwqow)

[3.1.2. Nomenclátor oficial de la Comunidad de Madrid 31](#_heading=h.lzve03xulgd0)

[3.2. Configurar Backend 33](#_heading=h.fzqapijolda)

[3.3. Rest Api 38](#_heading=h.ikice8gvo27i)

[3.3.1 Módulo de acceso a base de datos 38](#_heading=h.qih82gh2gclf)

[3.3.2. Módulo de seguridad 40](#_heading=h.k8sf2gtnvdgv)

[3.3. Desarrollo Pantallas 44](#_heading=h.lzggte8d9nip)

[3.4. Pruebas manuales 45](#_heading=h.byz12btkklq9)

[3.5. Crear infraestructura 46](#_heading=h.7et5kku98xqt)

[4. Glosario 47](#_heading=h.35nkun2)

[5. Bibliografía 48](#_heading=h.1ksv4uv)

Lista de figuras

[Figura 1: Visor Geo-EIEL. Fuente: [1] 1](#_heading=h.cq8vxcdhua93)

[Figura 2: Metodología en cascada. Fuente: [2] 5](#_heading=h.xrqftvjljc6p)

[Figura 3: Metodologías ágiles. Fuente: [3] 5](#_heading=h.ggrw1e2mm14l)

[Figura 4: Diagrama de Gantt con planificación del proyecto 8](#_heading=h.2r8gn2v8s01f)

[Figura 5: Convocatoria anunciada 11](#_heading=h.379f9h4ia3cl)

[Figura 6: Formulario de aplicación gvSIG-EIEL 12](#_heading=h.itnc6m5ur6vv)

[Figura 7: Ejemplo de formulario QField para QGis 13](#_heading=h.ahpo6zhmcxhc)

[Figura 8: Arquitectura básica de una aplicación web. Fuente [11] 15](#_heading=h.n4v02zyjvd43)

[Figura 9: Impacto de las PWA en el desarrollo de las aplicaciones web. Fuente [12] 19](#_heading=h.rokt0mp67w4)

[Figura 10: Definición básica de un SIG. Fuente [13] 20](#_heading=h.nzy5mvujzgan)

[Figura 11: Imagen del nomenclátor oficial de la Comunidad de Madrid 23](#_heading=h.phrgaisbjvcn)

[Figura 12: Logo Docker. Fuente [14] 24](#_heading=h.4najt253fivg)

[Figura 13: Arquitectura del sistema 28](#_heading=h.ybenpahlxscv)

[Figura 14: Repositorio en GitHub 29](#_heading=h.i6nvih3e0s5c)

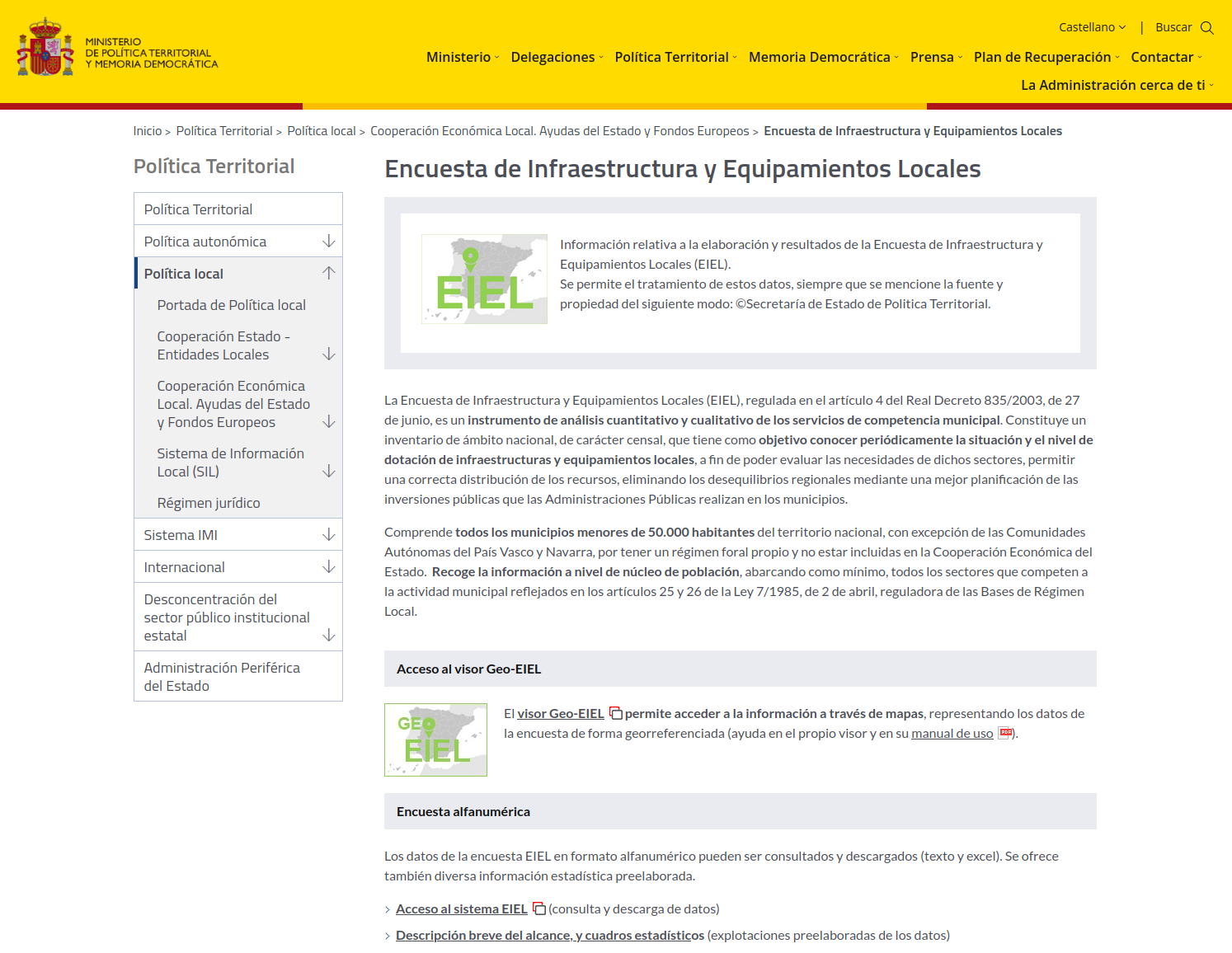
# Introducción

## 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.

## 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 memoría 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 específicicos.**

* 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.

## 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.

## Enfoque y método seguido

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

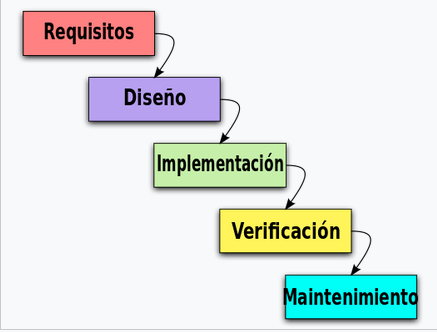
1. Desarrollar un producto nuevo desde cero: Esta estrategia implica diseñar y construir la aplicación desde el inicio, aprovechando las tecnologías más actuales y adaptándolas a las necesidades específicas del proyecto.
2. Adaptar un producto existente: Esta estrategia implica tomar una aplicación web 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 tecnologias 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.

## 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.
* **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 virtua**l (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)

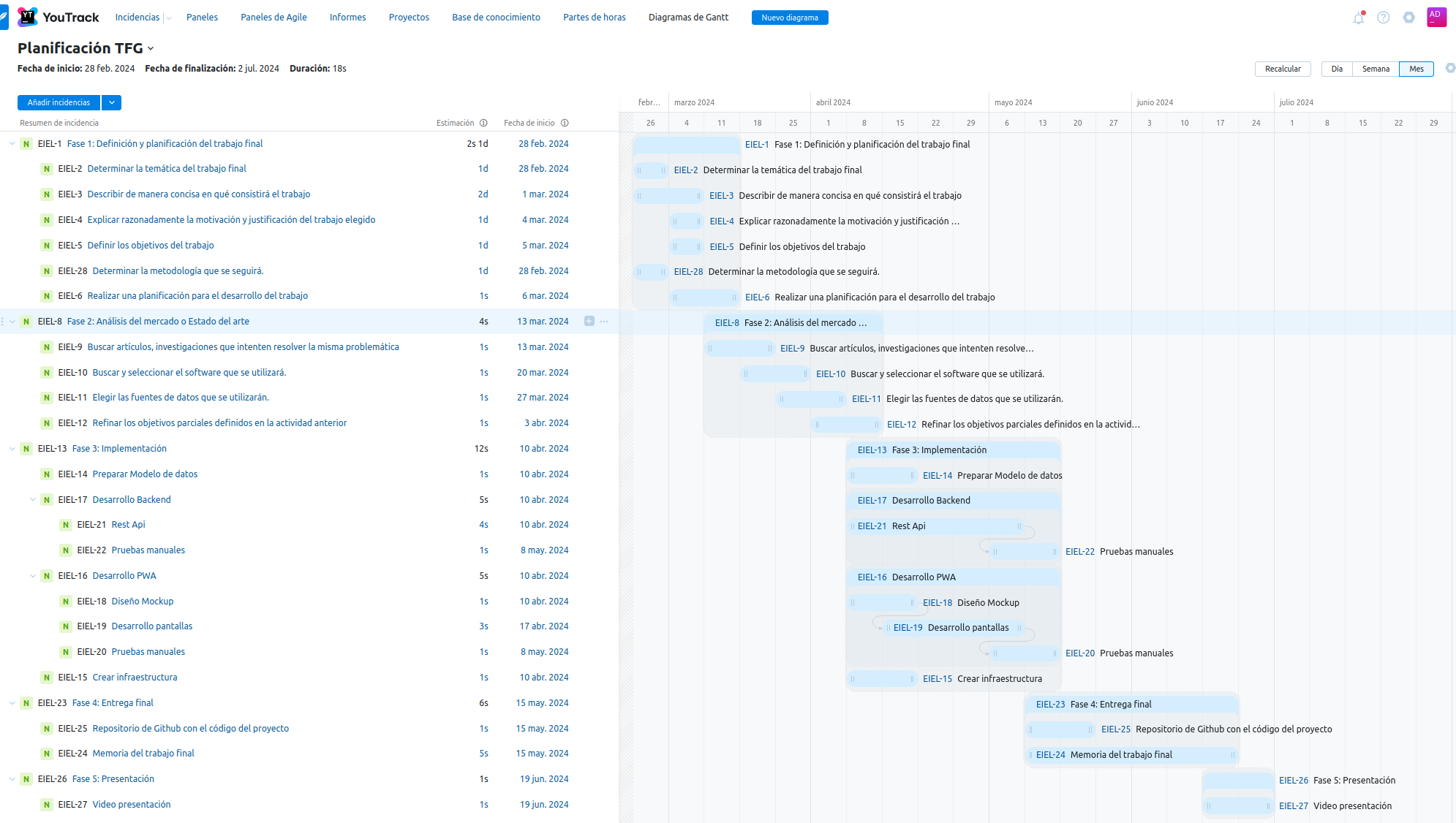
* Preparar Modelo de datos.
* Desarrollo Backend.
  + Rest Api.
  + Pruebas manuales.
* Desarrollo PWA.
  + Diseño Mockup.
  + Desarrollo pantallas.
* Pruebas manuales.
* Crear infraestructura.

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

## 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.

## 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.

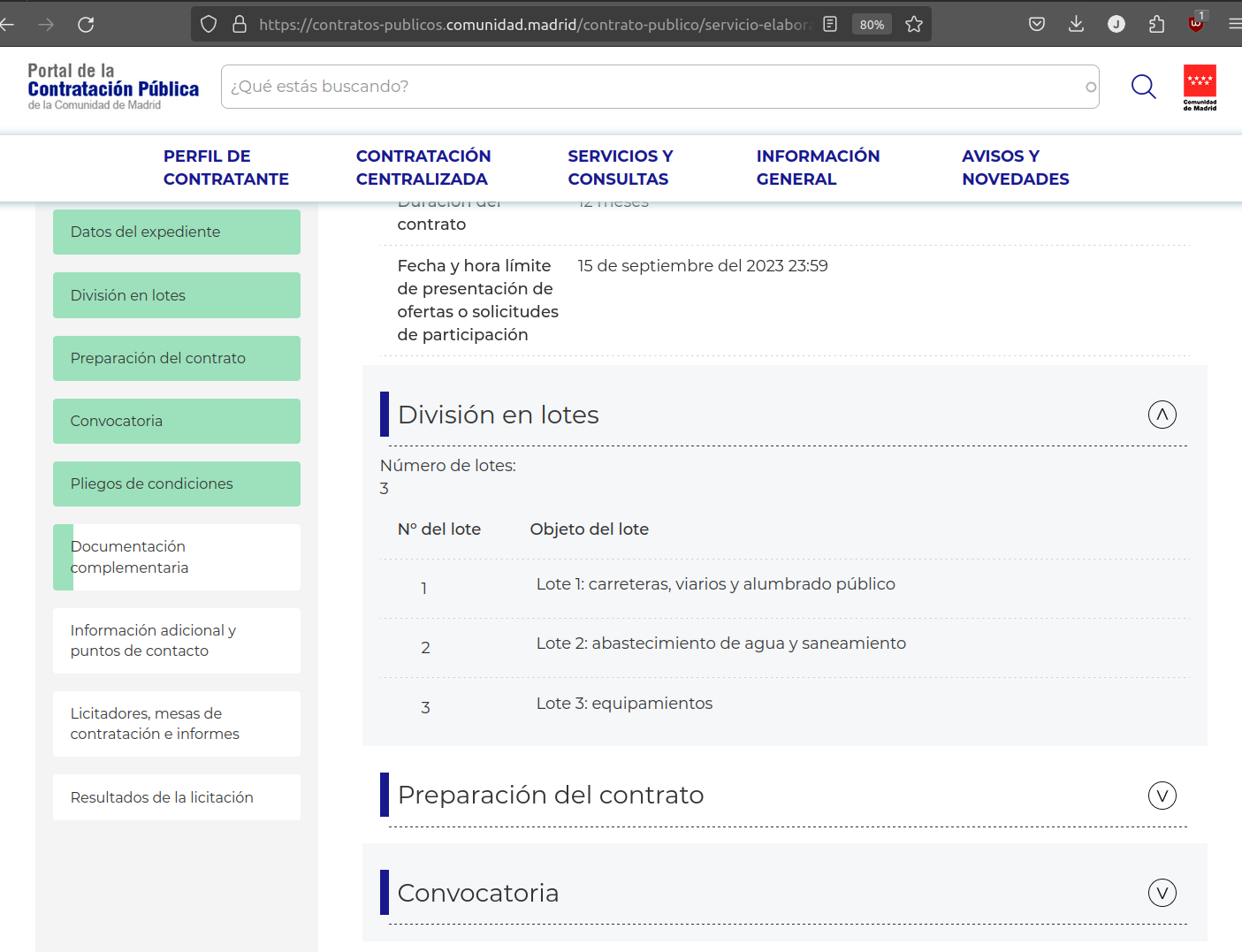
# Estado del arte.

# 

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)



###### 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.



###### 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.

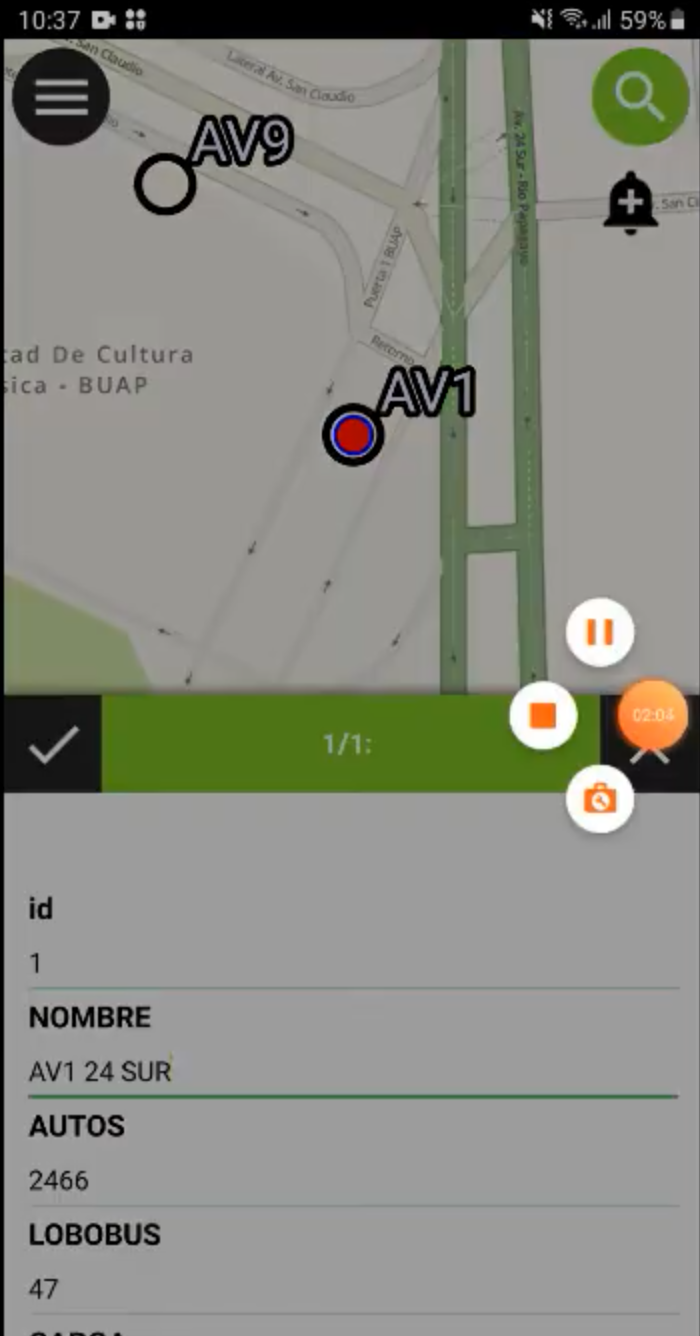
Este vacío es precisamente donde nuestro proyecto se destacará, al ofrecer una solución integral que aborda estas limitaciones y proporciona una experiencia de usuario fluida y eficiente.

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:
      * Muncicipios, 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 1**1**:** 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 1**3**:** 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.

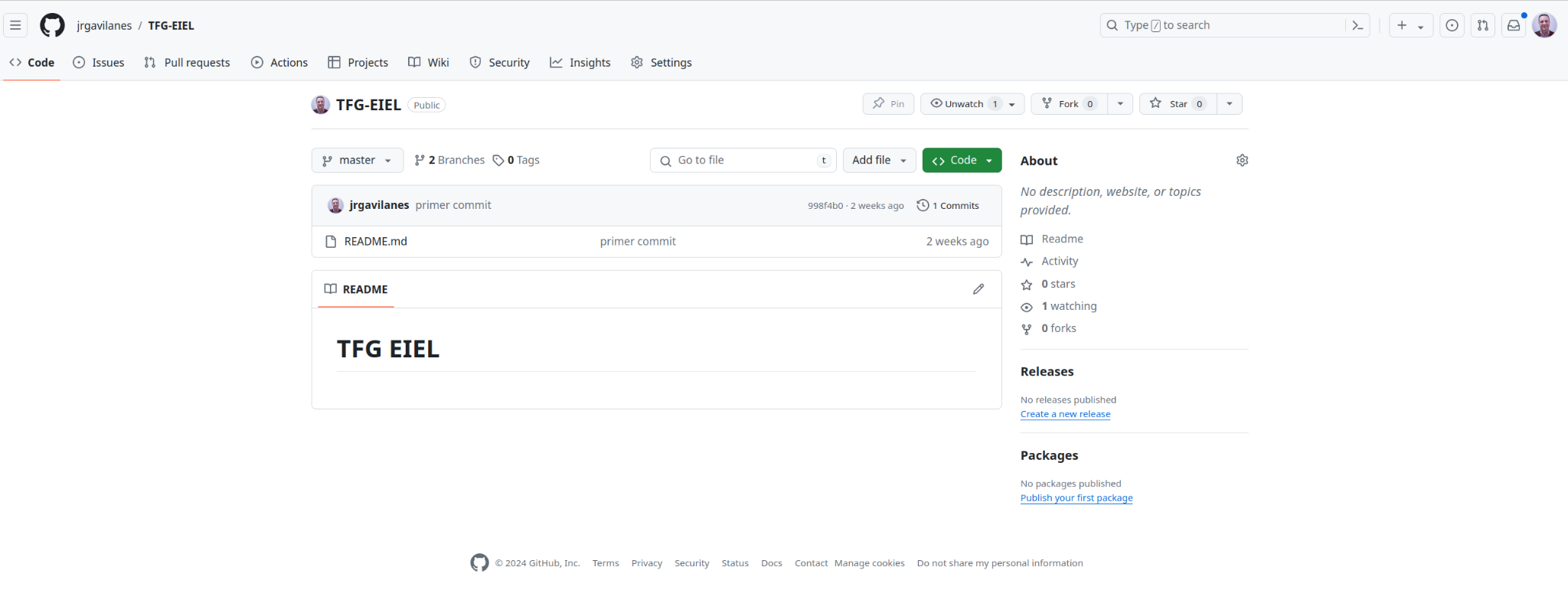
La integración con GitHub y GitHub Actions facilita la colaboración en el desarrollo de la aplicación y automatiza el proceso de despliegue en producción. Esto garantiza una entrega continua y una mayor eficiencia en el ciclo de desarrollo.

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.

# Implementación

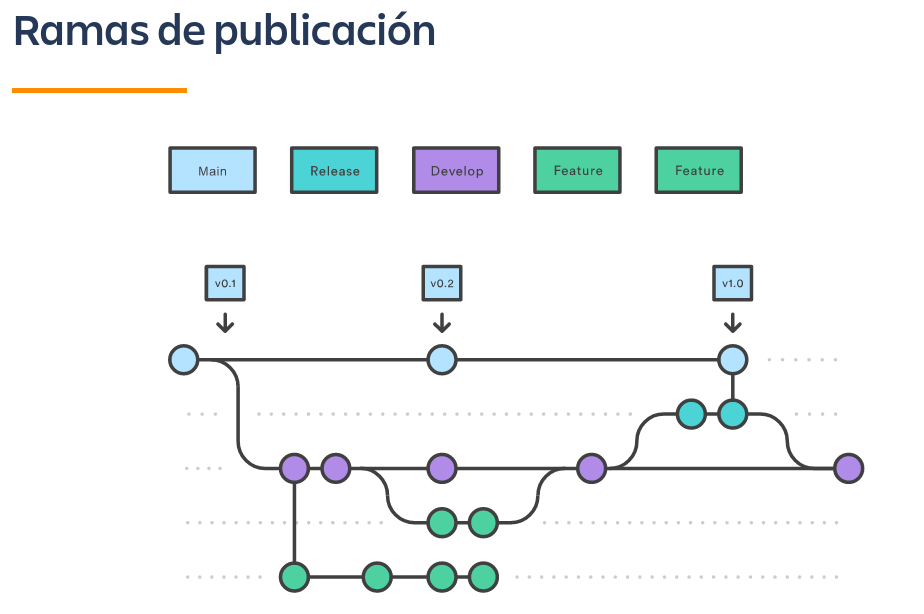
En esta sección mostraremos los pasos seguidos para construir la solución software, acorde con la planificación establecida en la figura 4.

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.

  
**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

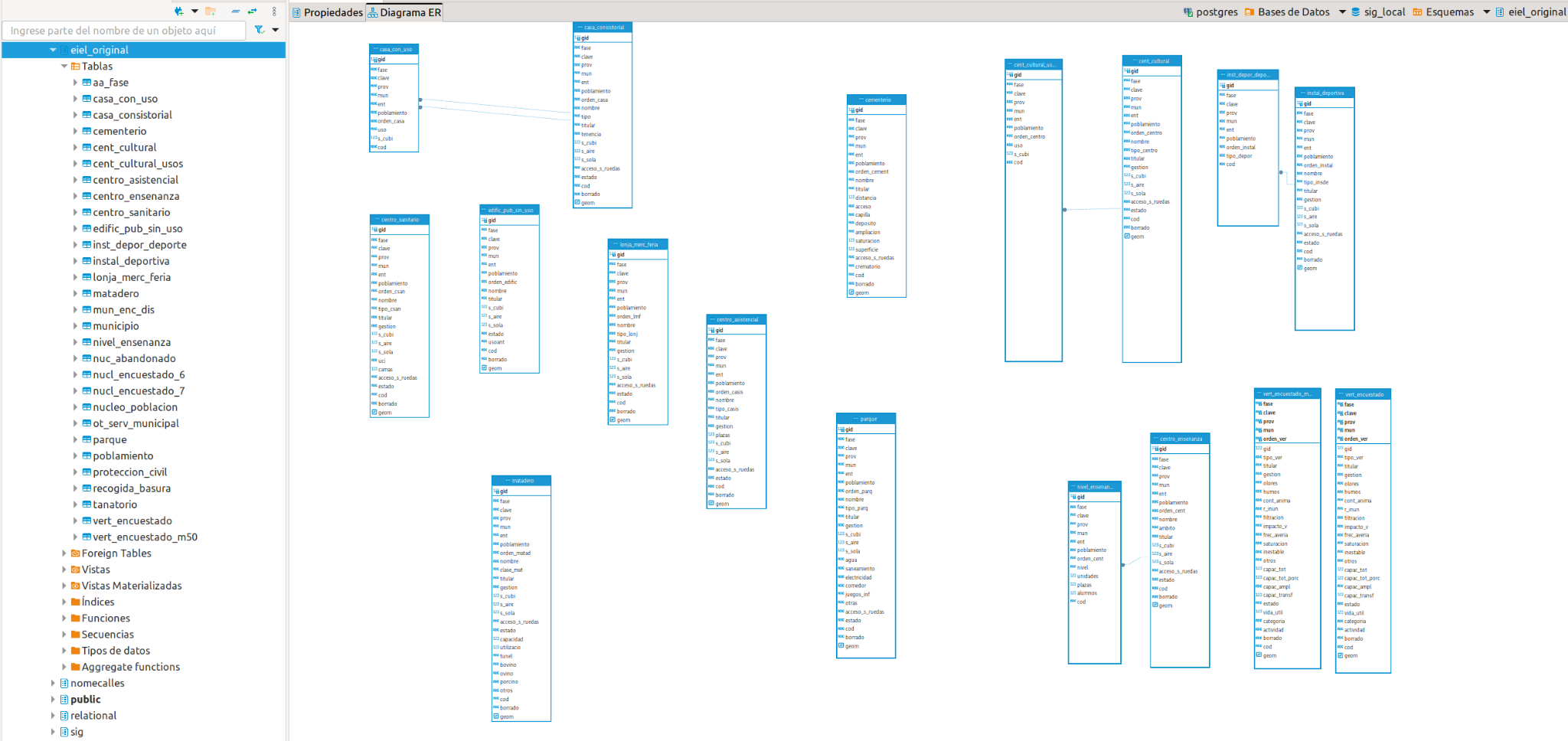
|  |  |
| --- | --- |

[Modelo de datos](https://drive.google.com/file/d/1pXh-vePFqRCias9Cz0h8xzVzBlSutItq/view?usp=sharing)

[Validaciones](https://drive.google.com/file/d/1IduwDfQmZ6jMATaeYD9jJCcHuE7JISfU/view?usp=sharing)

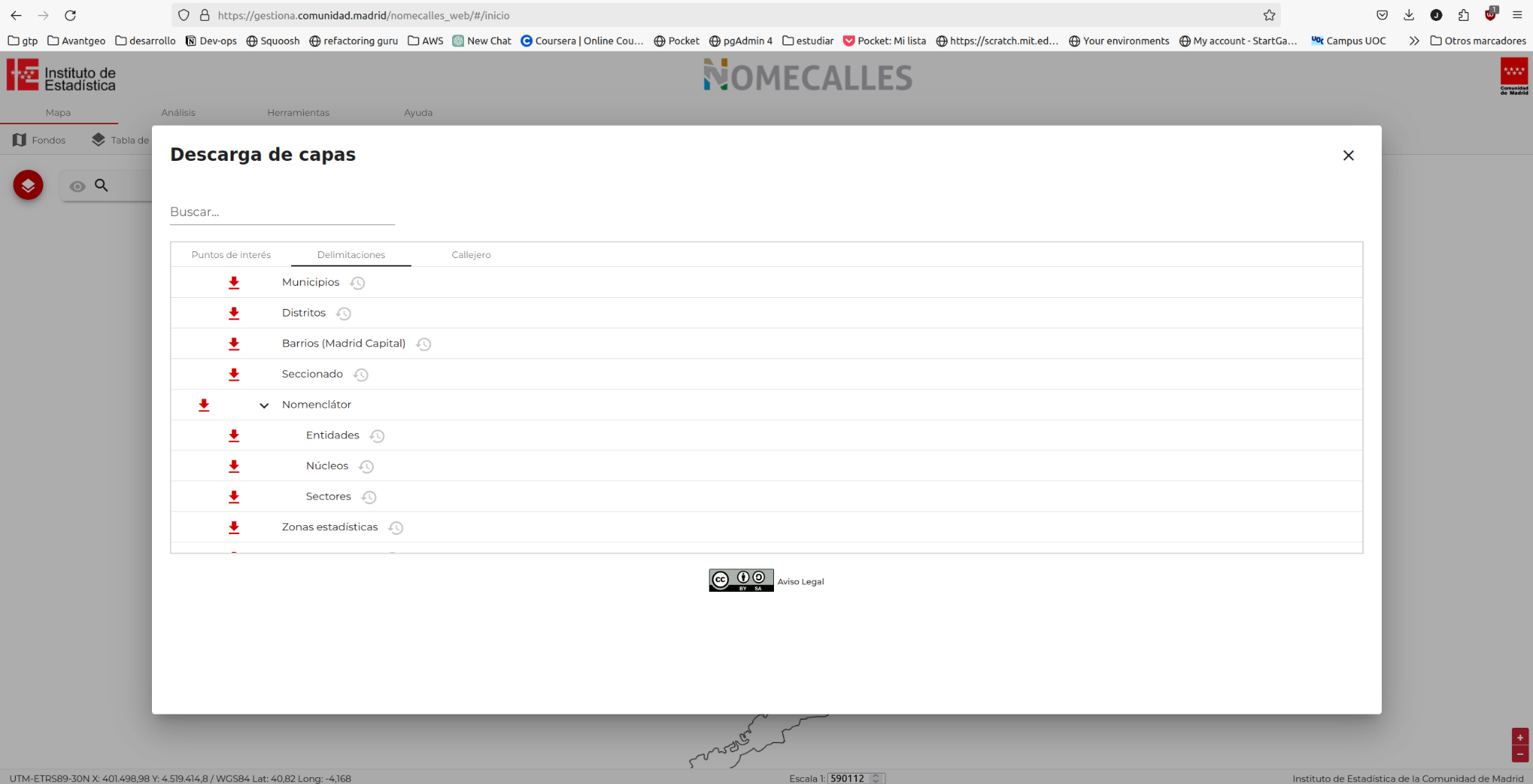
[Manual de instrucciones](https://drive.google.com/file/d/1bTVmvwmA4E5ft3-l66mhyQ-ueTZfDkig/view?usp=sharing)

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 instersectando 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 |
| --- | --- |
|  | # 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; |

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

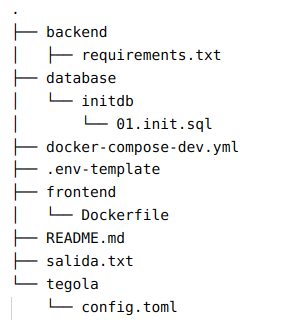


## 

## 3.2. Configurar Backend

| *git clone git@github.com:jrgavilanes/TFG-EIEL.git*  *git checkout feature/01\_configuracion\_entorno* |  |
| --- | --- |

Esta es la estructura inicial del proyecto



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

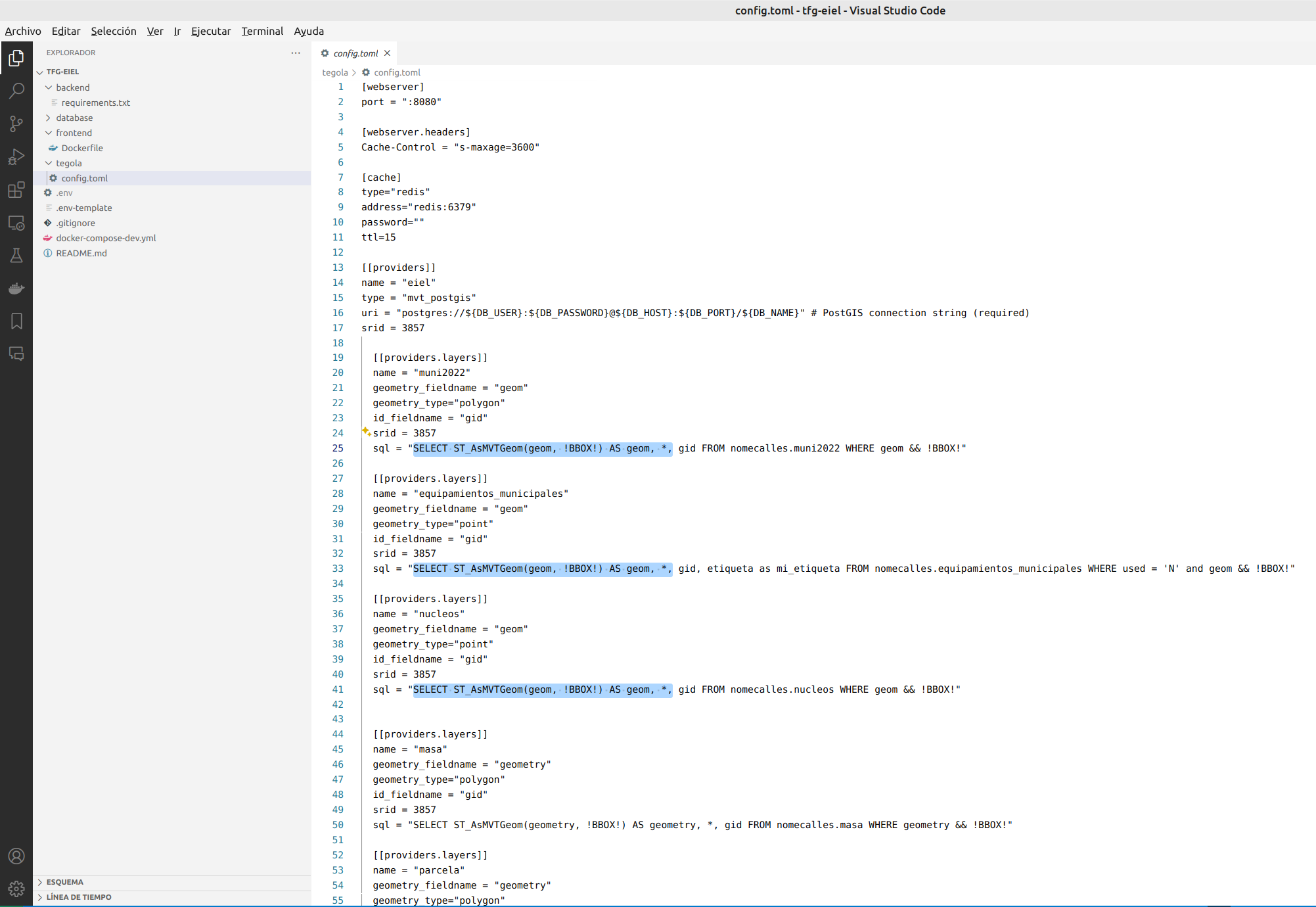


Vemos que la base de datos escucha por el puerto 8003, Redis por 8379 y Tegola por el 8005.

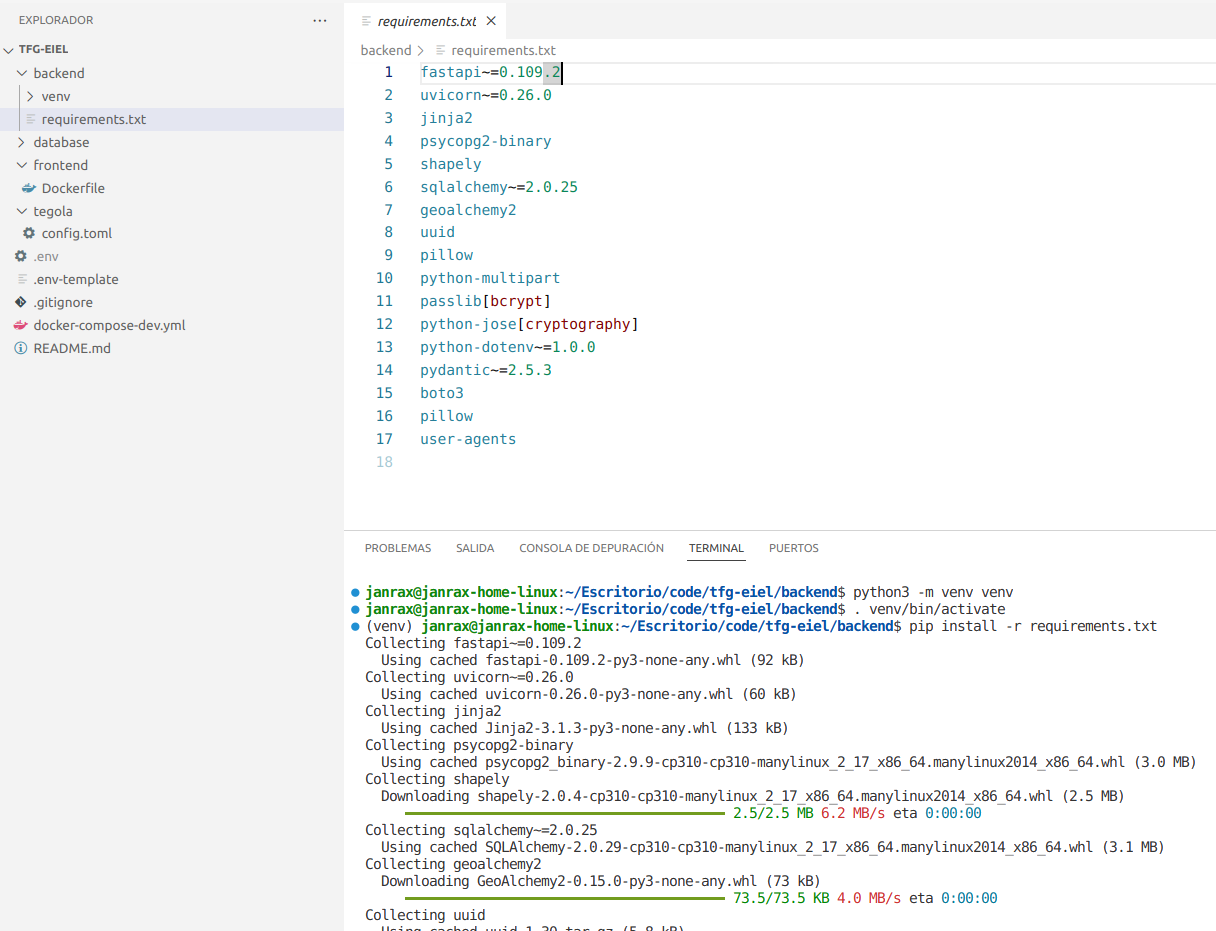
Como ya hemos cargado en la base de datos la información del nomenclátor, con el procedimiento que hemos visto en el punto 3.1.2, ya podremos ver los puntos desde su cliente web.



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



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



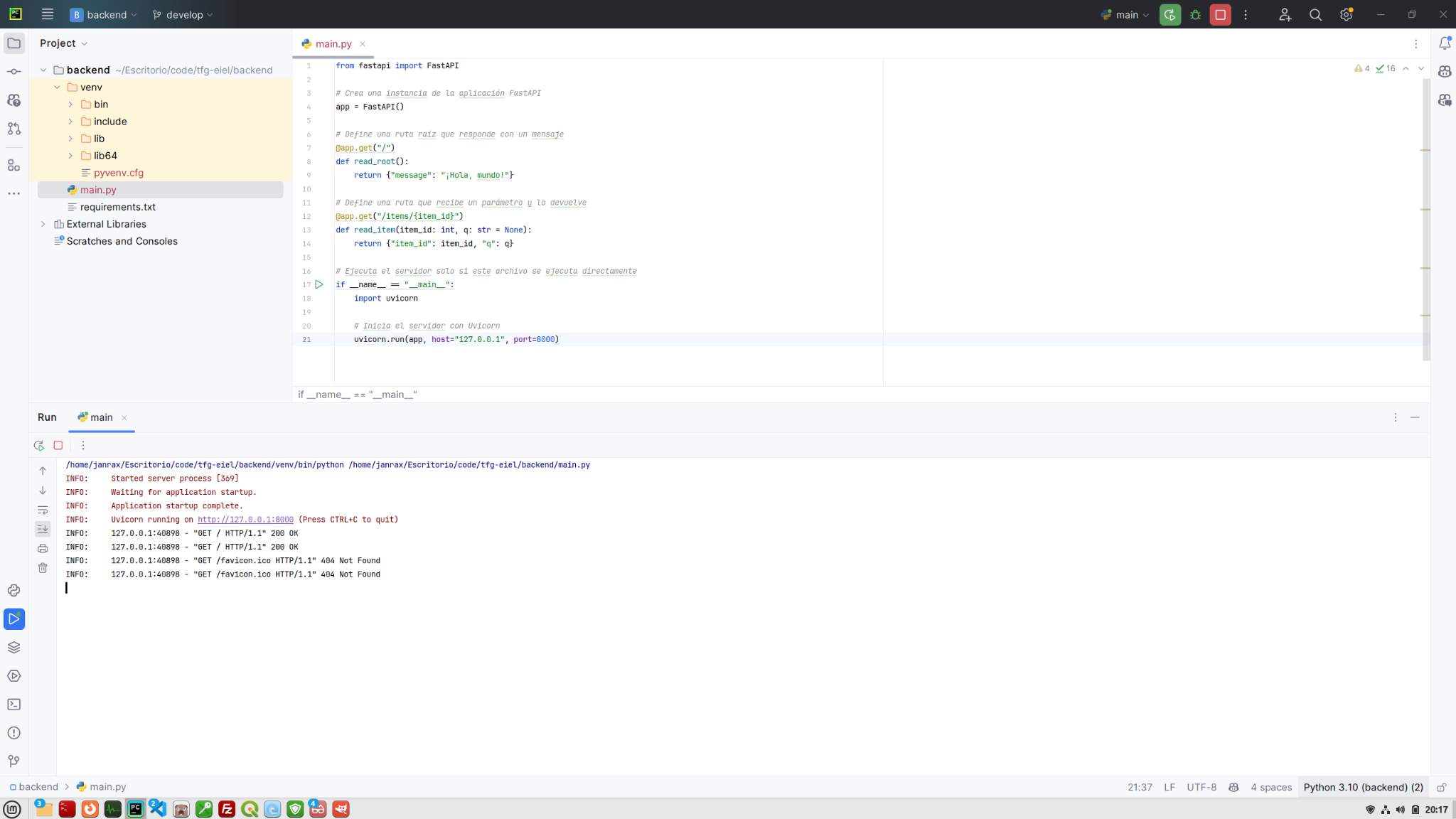
Crear un entorno virtual en Python proporciona un ambiente limpio y controlado para el desarrollo de proyectos, asegurando la consistencia, la portabilidad y la facilidad de gestión de dependencias, ya que facilita:

* **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 .



## 

Vemos que ya podemos acceder desde el navegador al endpoint recién creado

## 

El framework FastApi, en la url /docs, nos ofrece por defecto un entorno OpenApi donde visualizar, documentar y testear nuestros endpoints.

## 

## 3.3. Rest Api

### 3.3.1. 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.



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.

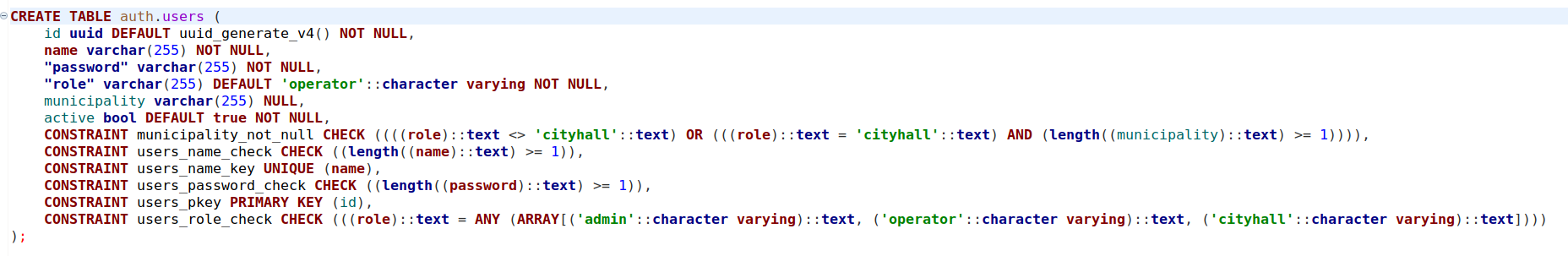
Con db\_dependency\_postgres, se define una dependencia que se utilizará en las rutas de API para inyectar una sesión de base de datos PostgreSQL. Utiliza Annotated para proporcionar anotaciones de tipo, especificando que la dependencia es una instancia de Session y depende de la función get\_db\_postgres.

### 

### 3.3.2. 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.



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.

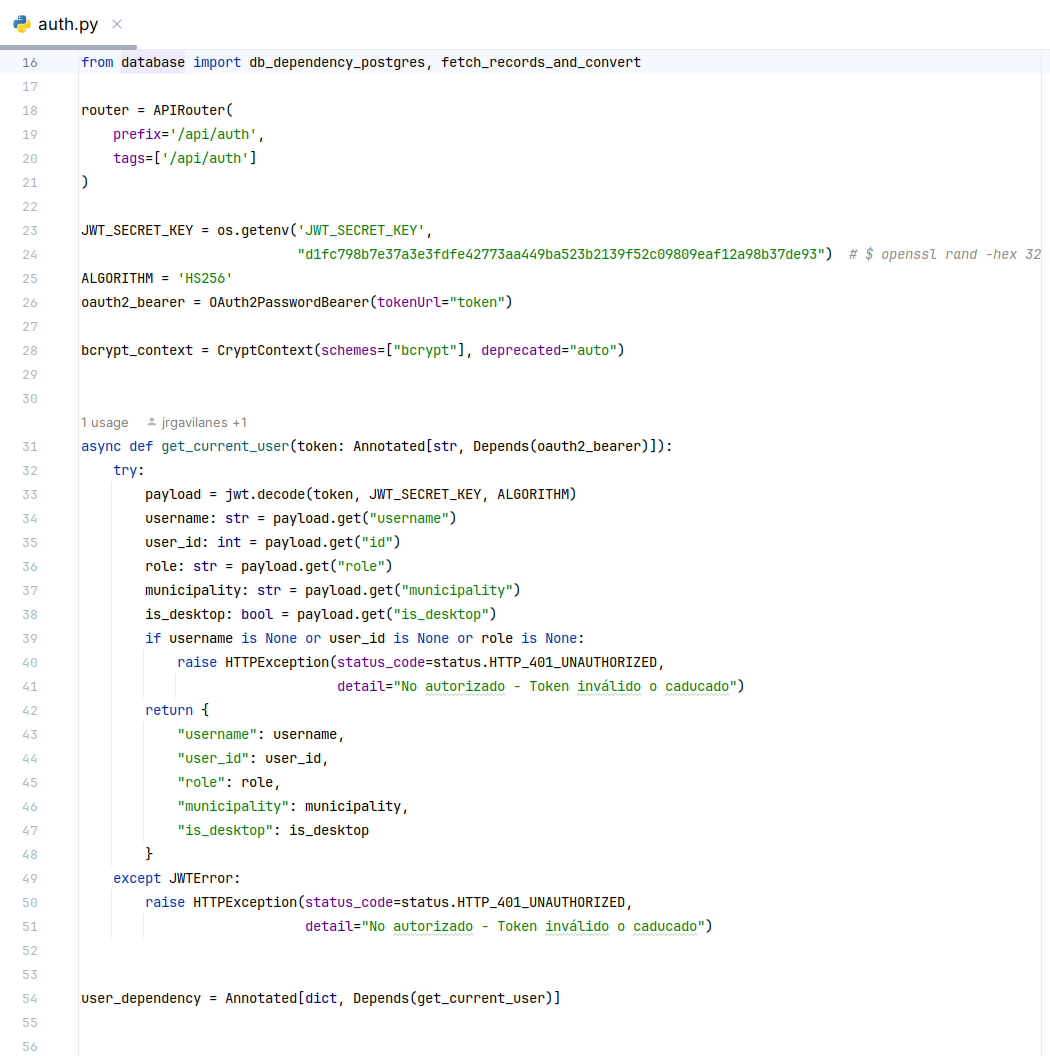
El sistemas de seguridad entre backend y frontend será a traves de JWT con una duración del token de 8 horas.

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.

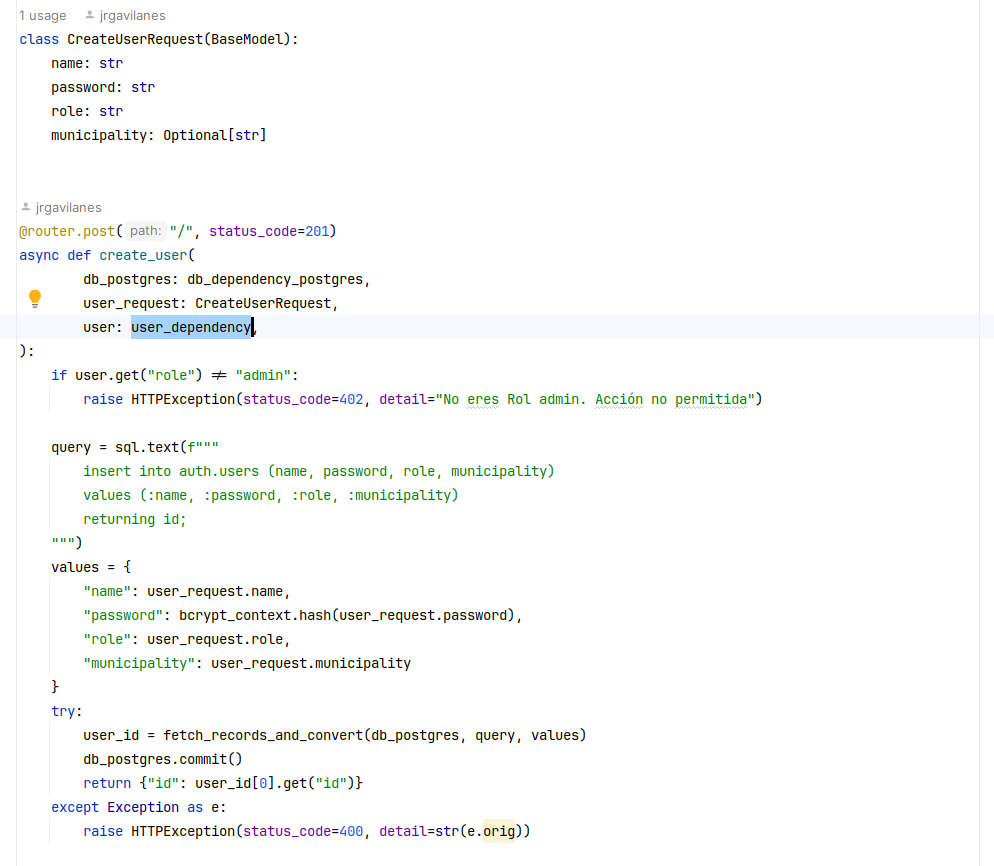
El cliente incluye este JWT en cada solicitud subsiguiente, ya sea en el encabezado de Autorización o como un parámetro de consulta. El servidor verifica la firma del JWT para garantizar que no haya sido manipulado y luego extrae la información del usuario de la carga útil para determinar si tiene acceso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

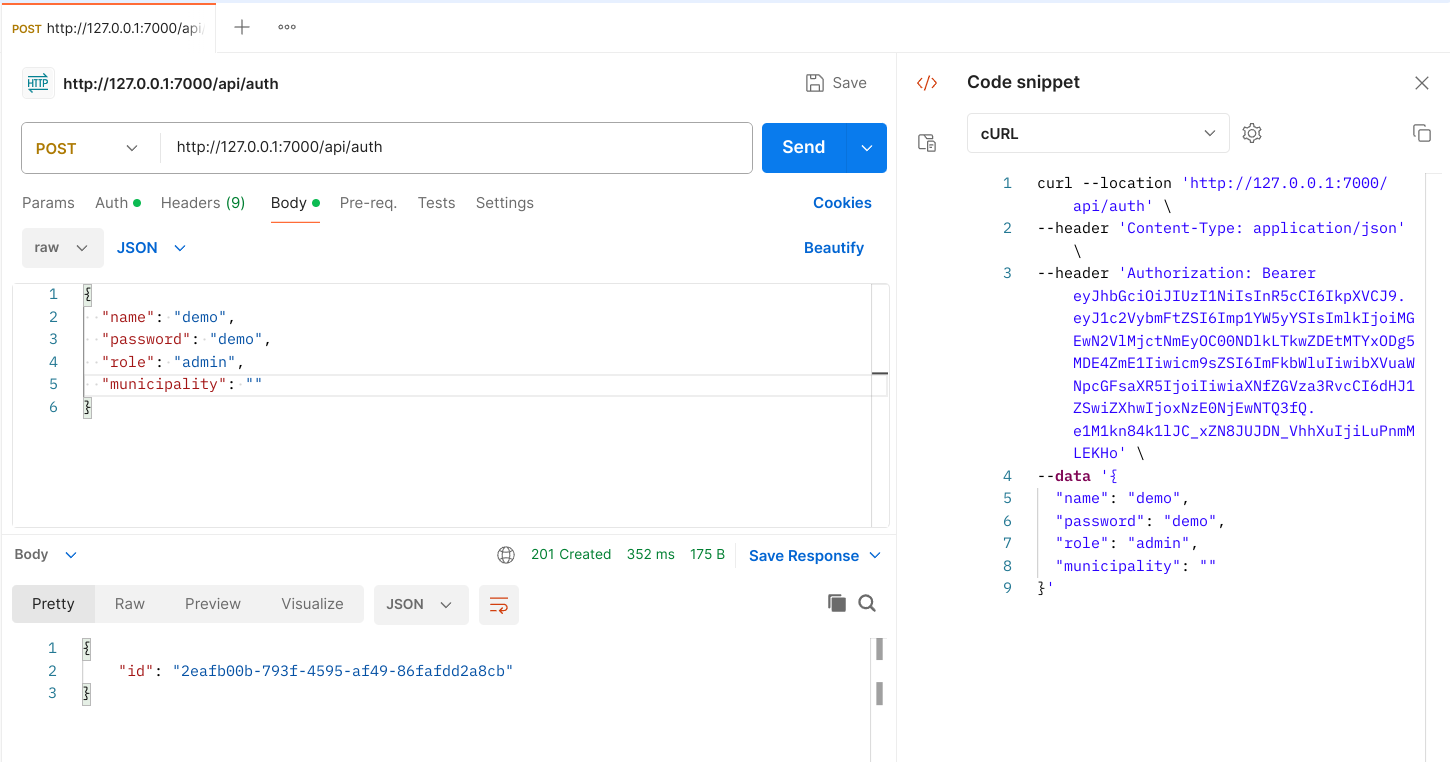
En el siguiente código, vemos la función que se inyectará en los endpoints securizados y que se encargará de validar el token enviado en cada llamada.



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. Después almacenaremos la contraseña de forma encriptada en la base de datos.



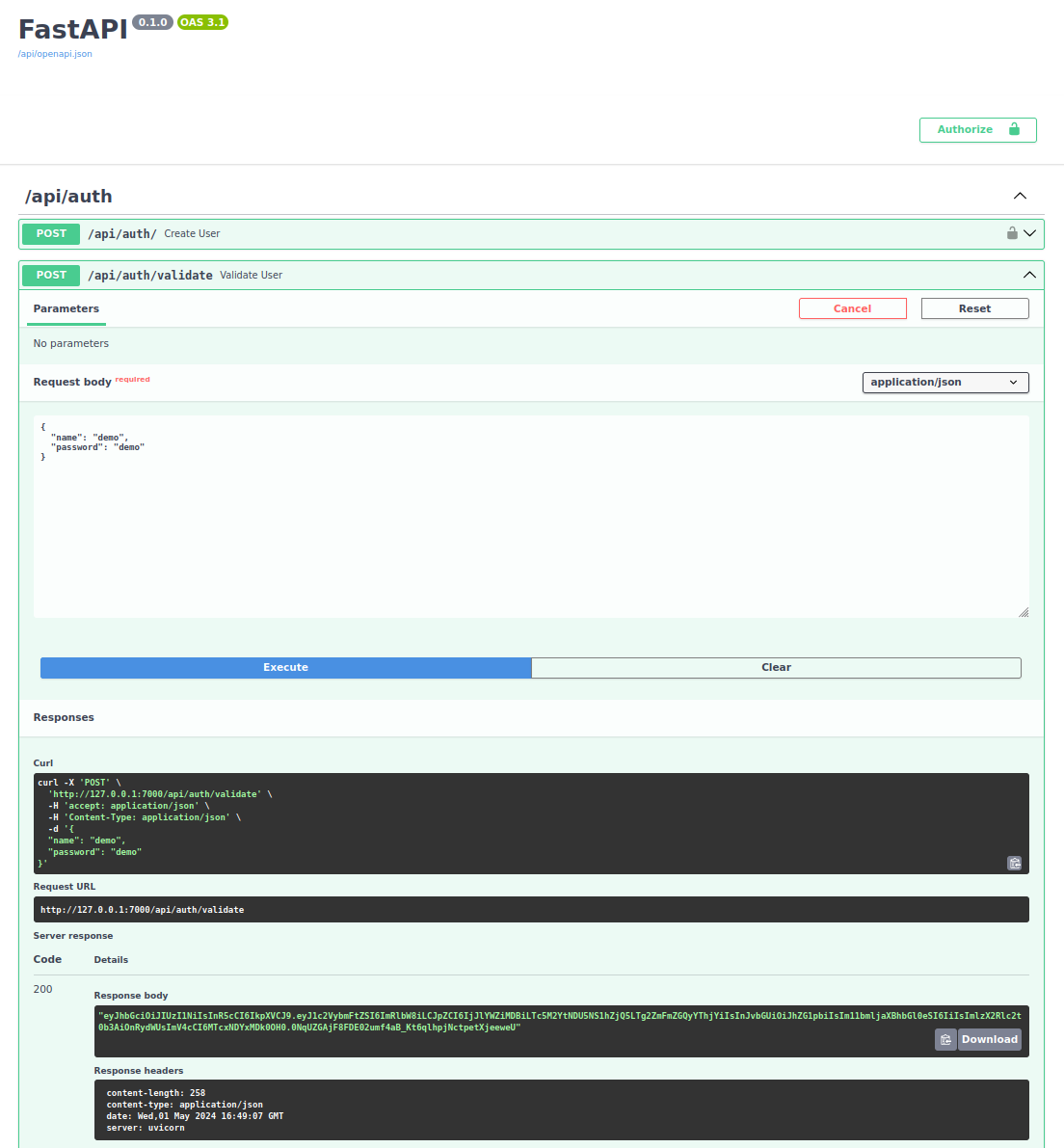
A continuación hacemos la llamada para crear un usuario



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

## 

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



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

## 

## 

## 

## 

## 3.3. Desarrollo Pantallas

## 

## 3.4. Pruebas manuales

## 

## 3.5. Crear infraestructura

# 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

# 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. Implantación 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)> (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)