

Índice de ilustraciones

1.	Arquitectura de la Aplicación Web	15
2.	Diagrama de la base de datos	16
3.	Inicio aplicación	20
4.	Página de Contacto	21
5.	Inicio de sesión	21
6.	Inicio de sesión con error	22
7.	Restablecer contraseña	22
8.	Datos comunidad de regantes	23
9.	Datos usuario del agua	23
10.	Parcelario	24
11.	Usos en parcela(Parcelario)	24
12.	Contadores y medidas (Parcelario)	25
13.	Consumos(Parcelario)	25
14.	Datos adicionales(Parcelario)	25
15.	Control de pagos(Parcelario)	26
16.	Petición turnos de riego	26
17.	Facturas	27
18.	Factura para imprimir	27
19.	Control de Pagos	28
20.	Control de pagos factura	28
21.	Listado del libro diario	29
22.	Registro de documentos 1	29
23.	Registro de documentos 2	30
24.	Incidencia y mantenimiento	30
25.	Información cultivos	31
26.	Estadística de consumo de agua del usuario	31
27.	Emitir circular	32
28.	backend principal	34
29.	backend modificar usuario	35
30.	backend gestionar informes	35
31.	Página web de descarga Linux Debian	36
32.	Debian instalado en ordenador	37
33.	Página web de descarga Visual Studio Code [10]	38
34.	Visual Studio Code Instalado	38
35.	Terminal con la versión de Python utilizada	39
36.	Django VScode instalación	39
37.	Version instalada de Node.js	40
38.	VSCODE instalación docker	40
39.	Version Git instalada	41

40.	Versiones instaladas de NPM y YARN	41
41.	GESTAGUA-CAP	42
42.	VScode instalación DJango	43
43.	Estructura react	44
44.	Docker	46
45.	VScode Frontend ejecución docker	47
46.	VScode Backend ejecución docker	48
47.	http://localhost:3000	48
48.	http://localhost:8000/admin	49
49.	http://localhost:8000/swagger	49
50.	Swagger 1	53
51.	Swagger 2	53
52.	Obtener token de acceso	54
53.	Datos con token de acceso	54
54.	Autorización Token	55
55.	Petición 'GET /community/all/'	55
56.	Resultado petición 'GET /community/all/'	56
57.	Postman	57

CAPÍTULO 1

Introducción

1.1. Antecedentes

La gestión de recursos hídricos es un desafío crítico en diversas regiones, especialmente en áreas como Andalucía, que son propensas a la sequía y cuentan con recursos hídricos limitados. La ineficiencia en la gestión del agua, junto con las frecuentes roturas en infraestructuras desarrolladas por generaciones pasadas y la falta de capacitación para manejar los complejos programas de gestión existentes, han llevado a un desperdicio significativo de este recurso vital. Las comunidades de regantes, encargadas de gestionar el agua para la agricultura, enfrentan grandes dificultades debido a la falta de herramientas accesibles y efectivas.

Personalmente, tengo familiares que forman parte de comunidades de regantes y una estrecha relación con la Asociación de Regantes Provincial de Málaga (APREMA). En mi día a día, soy testigo de los problemas que enfrentan al regar sus cultivos, desde roturas de tuberías hasta pérdidas de agua. Estos desafíos subrayan la necesidad de una solución más eficiente y fácil de usar para la gestión de los recursos hídricos.

1.2. Marco del proyecto

El presente Trabajo de Fin de Grado aborda el desarrollo de una aplicación web que resuelve un desafío específico en la gestión de las comunidades de regantes. La problemática central consiste en gestionar y facilitar la gestión de las actividades y recursos dentro de estas comunidades, con el propósito de mejorar la organización, eficiencia y comunicación entre los miembros, y reducir costos asociados a la administración y operatividad. De esta forma, se puede controlar mejor las comunidades y organizar eficientemente los recursos hídricos disponibles.

1.3. Motivación

La gestión eficiente de los recursos hídricos es crucial para la sostenibilidad agrícola, especialmente en regiones como Andalucía, donde es limitada la disponibilidad de aguas y su uso eficiente es algo esencial, por esto surge la necesidad de proporcionar a las comunidades de regantes una herramienta de software accesible y fácil de usar, permitiéndoles gestionar eficientemente sus recursos administrativos e hídricos. Se ha buscado diseñar e implementar una aplicación web inspirada en GESTAGUA-CAP[1] pero que sea completamente independiente, a pesar de sus ventajas y adopción por más de 250 comunidades de regantes, este software ofrecido a las comunidades presenta una interfaz poco intuitiva, unido al hecho de que los usuarios no son expertos en tecnología, impide que sea aprovechado plenamente su potencial, lo cual puede conducir a una gestión ineficiente de los recursos hídricos.

Este proyecto ha contado con el apoyo activo de las comunidades de regantes y la asociación provincial de regantes, las cuales han estado involucradas en cada etapa del proyecto, satisfaciendo así las necesidades reales de los usuarios finales.

1.4. Objetivos

El objetivo principal de este Trabajo de Fin de Grado (TFG) es desarrollar una aplicación web accesible y fácil de usar para la gestión de recursos hídricos en comunidades de regantes, inspirada en las funcionalidades de GESTAGUA-CAP[1], pero con mejoras significativas en términos de usabilidad y accesibilidad. Para lograr este objetivo general, junto con representantes de las comunidades de regantes, se han definido varios objetivos específicos que guiarán el desarrollo del proyecto:

- **Registro y Gestión de Usuarios y Parcelas:**

- Desarrollar una interfaz que permita a los usuarios registrar y gestionar una lista de comuneros y sus respectivas parcelas.
- Facilitar la visualización y edición de los datos de los comuneros y parcelas, asegurando la integridad y consistencia de la información.

- **Gestión de Contadores y Peticiones de Agua:**

- Simplificar el proceso de registro y gestión de contadores.

- **Generación de Facturas:**

- Proporcionar opciones de facturación personalizables por la comunidad, incluyendo la posibilidad de configurar IVA, recargos e intereses.
- Desarrollar la gestión de anticipos y atrasos en los pagos, facilitando el seguimiento y control financiero.

- **Gestión de Cobros y Pagos:**

- Permitir a los usuarios registrar y seguir los cobros y pagos realizados por los miembros de la comunidad de regantes.

- **Registro de Documentos:**

- Mantener un registro detallado de todos los documentos que entran y salen de la comunidad, asegurando una gestión documental eficiente y organizada.

- **Emisión de Comunicaciones:**

- Implementar funcionalidades para la emisión de cartas o circulares personalizadas para cada comunero, mejorando la comunicación interna de la comunidad.

- **Inventario de Cultivos:**

- Registrar y gestionar un inventario de los cultivos de la comunidad.
- Facilitar el seguimiento del estado de los cultivos y la gestión de los recursos necesarios para su mantenimiento.

- **Gestión de Incidencias y Mantenimiento:**

- Registrar y gestionar las incidencias ocurridas en la red de distribución y las actuaciones de mantenimiento realizadas.

- **Estadísticas de Consumo de Agua:**

- Proporcionar herramientas para el análisis estadístico de los consumos de agua de la comunidad, apoyando la toma de decisiones informada.
- Desarrollar informes y visualizaciones que faciliten la comprensión y análisis de los datos de consumo.

- **Consultas Geográficas:**

- Incluir un Sistema de Información Geográfica (SIG) para la consulta geográfica de toda la información almacenada en el programa.
- Facilitar la creación y gestión del parcelario y la red de distribución, apoyándose en ortofotografías digitales y otras fuentes de datos geoespaciales.

Cada uno de los objetivos específicos contribuirán a alcanzar el objetivo general de desarrollar una aplicación web para mejorar la eficiencia en la gestión de recursos hídricos, optimizando procesos técnicos y administrativos de las comunidades de regantes.

1.5. Estructura de la presente memoria

En el segundo capítulo, se realiza un análisis del estado del arte, explorando desde la problemática de la gestión de las comunidades de regantes hasta las herramientas fundamentales como HTML[5], CSS[6] y Python[8].

El tercer capítulo expone la metodología empleada en el desarrollo del proyecto y detalla la definición del mismo, abarcando aspectos como el alcance y desarrollo del proyecto.

Posteriormente, el cuarto capítulo se enfoca en el diseño y arquitectura de la aplicación, incluyendo la interfaz de usuario, la arquitectura de la base de datos y las especificaciones técnicas.

En el quinto capítulo, se explica cómo se ha llevado a cabo la implementación de la aplicación, presentando el proceso de desarrollo, la codificación de funcionalidades, la integración de sistemas y módulos.

El sexto capítulo se dedica a las pruebas y evaluación de la aplicación, describiendo las pruebas realizadas, la evaluación de la funcionalidad y usabilidad, el análisis de resultados y la retroalimentación de los usuarios.

Finalmente, el séptimo capítulo presenta las conclusiones derivadas del proceso de elaboración del TFG, así como perspectivas para futuras mejoras, lecciones aprendidas y recomendaciones para el desarrollo futuro de la aplicación.

CAPÍTULO 2

Estado Actual

La gestión de comunidades de regantes enfrenta desafíos similares a los problemas de optimización y distribución en diversas industrias. La necesidad de asignar y planificar eficientemente el uso de los recursos hídricos, cumpliendo con restricciones específicas y maximizando la eficiencia operativa, es fundamental para mejorar la gestión y reducir costos. En este contexto, el desarrollo de estrategias eficientes es crucial. A continuación se enumeran los problemas actuales.

2.1. Problema de gestión de comunidades de regantes

El problema de gestión de recursos hídricos en comunidades de regantes busca optimizar el uso y distribución del agua, además de gestionar las actividades administrativas y operativas. El objetivo es maximizar la eficiencia, garantizar la sostenibilidad y facilitar la administración de las comunidades. Las áreas más críticas de este problema son:

- **Registro y Gestión de Comuneros y Parcelas**
 - Registro detallado de comuneros y sus parcelas.
 - Mantener control actualizado de los miembros y sus tierras.
 - Desafíos: asegurar precisión y actualización de datos, facilitar acceso y modificación de información.
- **Gestión de Contadores y Peticiones de Agua**
 - Registro y gestión de contadores de agua y solicitudes de comuneros.
 - Controlar el consumo y gestionar solicitudes eficientemente.
 - Desafíos: asignación justa del agua, evitar desperdicios.
- **Generación y Gestión de Facturas**
 - Emisión de facturas personalizadas con IVA, recargos.
 - Gestión de anticipos y atrasos en pagos.
 - Desafíos: asegurar exactitud en la facturación, mantener registro claro de pagos.

■ **Seguimiento de Cobros y Pagos**

- Registro detallado de cobros y pagos de miembros.
- Desafíos: mantener registros precisos y actualizados, gestionar retrasos en pagos, garantizar transparencia financiera.

■ **Gestión de Inventario de Cultivos**

- Registro y gestión de inventario de cultivos.
- Desafíos: asegurar información precisa y actualizada.

■ **Registro y Gestión de Incidencias y Mantenimiento**

- Registro de incidencias en la red de distribución y gestión de mantenimiento.
- Desafíos: asegurar respuesta rápida a incidencias, mantener registros detallados de mantenimiento, minimizar impacto en distribución de agua.

■ **Estadísticas**

- Proporcionar estadísticas sobre consumo de agua.
- Desafíos: presentar información clara.

■ **Sistema de Información Geográfica (SIG)**

- Consulta geográfica de información almacenada y creación del parcelario con ortofotografía digital.
- Desafíos: integrar eficientemente información geográfica, asegurar precisión de datos, proporcionar herramientas intuitivas para gestión de información geográfica.

2.2. Problema de Optimización de la Distribución del Agua

La optimización de la distribución del agua es fundamental para asegurar que cada parcela reciba la cantidad adecuada de agua en el momento correcto, minimizando el desperdicio y aumentando la eficiencia. Las áreas más críticas de este problema son:

■ **Programación de Riegos**

- Planificación de horarios de riego para maximizar la eficiencia y minimizar la evaporación y el escurrimiento.
- Desafíos: coordinar la programación entre múltiples parcelas y sistemas de riego para evitar conflictos y sobrecargas.

■ **Gestión de Recursos Hídricos**

- Control y asignación de recursos hídricos disponibles según las necesidades y prioridades de la comunidad.

- Desafíos: asegurar una distribución equitativa y justa del agua, especialmente durante períodos de escasez.
- **Sostenibilidad y Conservación**
 - Implementación de prácticas sostenibles para la conservación del agua y la reducción de su uso.
 - Desafíos: promover la adopción de prácticas sostenibles entre los comuneros y medir el impacto de estas prácticas.

2.3. Problema de Escalabilidad y Rendimiento

A medida que la comunidad de regantes crece y se añaden más datos y usuarios, es vital que el software mantenga un rendimiento óptimo y pueda escalar adecuadamente. Los desafíos incluyen gestionar grandes volúmenes de datos, optimizar el desempeño de la aplicación para tiempos de respuesta rápidos y permitir la escalabilidad horizontal y vertical del sistema. Las áreas más críticas de este problema son:

- **Gestión de Grandes Volúmenes de Datos**
 - Almacenar y procesar los datos de forma eficiente de forma que se pueda acceder rápidamente cuando esto sea necesario.
 - Desafíos: asegurar la integridad y disponibilidad de los datos en caso de fallo del sistema o durante las actualizaciones.
- **Escalabilidad Horizontal y Vertical**
 - La escalabilidad horizontal distribuye la carga entre múltiples máquinas y la vertical mejora las capacidades existentes.
 - Desafíos: asegurar un rendimiento óptimo mediante el equilibrio de carga y la optimización del software, manteniendo baja la latencia.

2.4. Problema de Usabilidad y Accesibilidad

El software debe ser intuitivo y fácil de usar para todos los miembros de la comunidad de regantes, independientemente de su habilidad técnica. Los desafíos incluyen diseñar una interfaz de usuario intuitiva, asegurar la accesibilidad para personas con discapacidades y ofrecer soporte para múltiples idiomas. Las áreas más críticas de este problema son:

- **Diseño de Interfaz de Usuario Intuitiva**
 - Diseñar interfaces que faciliten una navegación fácil y eficiente para los usuarios.
 - Desafíos: asegurar que la interfaz sea intuitiva y accesible para todos los usuarios, independientemente de su nivel de experiencia tecnológica.

CAPÍTULO 3

Metodología y Fases del trabajo

Para el desarrollo del Trabajo de Fin de Grado se ha implementado una metodología ágil, específicamente Scrum, que permite una gestión flexible y eficiente del proyecto, asegurando así el cumplimiento de los objetivos establecidos y la obtención de un producto de alta calidad. Sin embargo, también se puede considerar que ha habido un enfoque iterativo incremental debido a las fases claras y la evolución progresiva del proyecto a través de iteraciones.

3.1. Metodología Scrum con enfoque iterativo incremental

Scrum es una metodología ágil que se centra en la gestión flexible y eficiente de proyectos, facilitando el cumplimiento de objetivos y la creación de productos de alta calidad. Esta metodología se complementa con un enfoque iterativo incremental, donde el desarrollo del proyecto se realiza en fases claras y evoluciona progresivamente a través de iteraciones. Cada iteración, o sprint, permite revisar y mejorar el producto, asegurando que se ajusta a las necesidades y expectativas del usuario final. En conjunto, Scrum y el enfoque iterativo incremental proporcionan una estructura robusta para la gestión y desarrollo continuo del proyecto, garantizando resultados óptimos. A continuación se detallan las fases principales y los métodos empleados:

1. Planificación Inicial:

- **Definición del Alcance:** Determinar los objetivos específicos del proyecto, realizando una identificación de funcionalidades y requisitos del software.
- **Configuración del Entorno de Trabajo:** Preparar el entorno de desarrollo, incluyendo la configuración de herramientas y tecnologías necesarias.
- **Planificación de Sprints:** Establecer un cronograma de trabajo basado en sprints, definiendo las tareas y los plazos para cada fase del proyecto.

2. Análisis:

- **Estudio de las tecnologías a utilizar:** Investigar y seleccionar los frameworks, lenguajes de programación y otras tecnologías que mejor se adapten al

proyecto.

- **Extracción de requisitos mediante entrevistas y análisis del software GESTAGUA-CAP[1]:** Realizar entrevistas y análisis del software GESTAGUA-CAP[1] para identificar los requisitos funcionales y no funcionales.
- **Extracción de los requisitos del proyecto:** Documentar todos los requisitos y obtener la validación de los usuarios finales.
- **Elaboración del Documento General de Requisitos (DGR):** Crear un documento detallado que contenga todos los requisitos funcionales y no funcionales del proyecto.
- **Validación de los requisitos por usuarios:** Obtener la aprobación y retroalimentación de los usuarios finales para asegurar que los requisitos documentados cumplen con sus necesidades.

3. Diseño y Arquitectura:

- **Diseño y modelado de la Aplicación:** Crear los modelos de datos y diseñar la arquitectura general de la aplicación.
- **Estudio, diseño y modelado de la Base de Datos:** Desarrollar el esquema de la base de datos que soportará la aplicación.
- **Maquetado de la aplicación web:** Definir la estructura y diseño de la interfaz de usuario.

4. Desarrollo e implementación:

- **Implementación de Funcionalidades:** Desarrollar las funcionalidades específicas descritas en los objetivos del proyecto.
- **Implementación de los Mapas e integración con los Datos:** Implementar los componentes del Sistema de Información Geográfica (SIG) y su integración con los datos de la aplicación.
- **Creación y desarrollo de las Interfaces:** Diseñar e implementar las interfaces de usuario para asegurar una experiencia de uso intuitiva y eficiente.
- **Revisión y Testing Continuo:** Realizar pruebas continuas para identificar y corregir errores durante el desarrollo.

5. Verificación y pruebas:

- **Realización de Pruebas software sobre las funcionalidades de la aplicación:** Ejecutar pruebas exhaustivas de todas las funcionalidades implementadas para asegurar su correcto funcionamiento.
- **Realización de Pruebas de usabilidad y aceptación con usuarios finales:** Colaborar con las comunidades de regantes para realizar pruebas de usabilidad y obtener retroalimentación, asegurando que la aplicación satisface sus necesidades.

6. Documentación:

- **Elaboración del Manual de Usuario:** Crear un manual detallado para guiar a los usuarios en el uso de la aplicación.
- **Elaboración del Manual de Instalación:** Documentar el proceso de instalación y configuración de la aplicación.
- **Elaboración de la Memoria del TFG:** Redactar la memoria del proyecto, incluyendo todos los aspectos relevantes del desarrollo y los resultados obtenidos.

3.2. Iteraciones y Reuniones

Cada iteración (sprint) ha tenido una duración de dos semanas. Las reuniones y comunicaciones se han llevado a cabo de la siguiente manera ordenada:

- **Tutora Lola Burgueño:**
 - **Reunión inicial presencial (20 minutos):**
 - **Objetivo:** Explicar y aprobar la idea del proyecto.
 - **Contenido:** Se discutió la idea del proyecto y se aclararon varias dudas sobre el enfoque y los objetivos. Tras esta reunión, se tomó una semana adicional para reflexionar y asegurar que la idea estaba bien planteada antes de comenzar el desarrollo del proyecto.
 - **Segunda reunión presencial (45 minutos):**
 - **Objetivo:** Informar sobre el progreso y resolver dudas.
 - **Contenido:** Se presentó el progreso inicial del proyecto y se resolvieron dudas sobre la metodología y el enfoque técnico. La profesora ofreció recomendaciones y se establecieron los próximos pasos a seguir.
 - **Tercera reunión presencial (45 minutos):**
 - **Objetivo:** Mostrar avances y resolver dudas específicas.
 - **Contenido:** Se presentaron los avances en la implementación y se discutieron dudas específicas sobre ciertos aspectos técnicos y metodológicos del proyecto. Se recibieron recomendaciones adicionales para mejorar el proceso.
 - **Tres reuniones telemáticas (20 minutos cada una):**
 - **Objetivo:** Mostrar la aplicación y el código, recibir recomendaciones.
 - **Contenido:** En estas reuniones, se mostró el progreso de la aplicación y el código desarrollado en remoto por cuestiones de infraestructura. La profesora revisó el trabajo, ofreció recomendaciones y ajustes necesarios, y se discutieron los siguientes pasos a seguir.

- Reunión para mostrar el proyecto terminado (45 minutos):
 - **Objetivo:** Presentar el proyecto terminado y revisar el código final.
 - **Contenido:** Se mostró el proyecto finalizado, incluyendo el código completo. La profesora revisó el trabajo, ofreció feedback final y se discutieron los últimos detalles antes de la defensa.
- Reunión para resolver preguntas sobre la defensa (45 minutos):
 - **Objetivo:** Preparar la defensa del TFG.
 - **Contenido:** Se discutieron posibles preguntas y temas que podrían surgir durante la defensa, y se proporcionaron recomendaciones sobre cómo presentar y defender el proyecto de manera efectiva.
- Intercambios breves de correos electrónicos
 - **Objetivo:** Mostrar novedades y consultas cortas del software y recibir recomendaciones.
 - **Contenido:** Se utilizaron para consultas rápidas y actualizaciones sobre el progreso del proyecto. La profesora brindó orientación y respuestas a las dudas planteadas.
- Aprema y Comunidad de Regantes Acequia de Granadillos:
 - Reuniones semanales (15 minutos cada una):
 - **Objetivo:** Mostrar novedades del software y recibir recomendaciones.
 - **Contenido:** En estas reuniones semanales, se mostraban las novedades y avances del software. Los stakeholders proporcionaban feedback y recomendaciones, que se incorporaban en las siguientes iteraciones del proyecto.

CAPÍTULO 4

Diseño y arquitectura

4.1. Requisitos

Los requisitos tanto funcionales como no funcionales de este proyecto han surgido tras numerosas reuniones con la Asociación Provincial de Regantes de Málaga (APREMA) y diversas comunidades de regantes como Acequia de Granadillos. Durante estas reuniones, se identificaron y discutieron los problemas y desafíos que enfrentan estas comunidades en la gestión de los recursos hídricos.

4.1.1. Proceso de Identificación de Requisitos

1. Reuniones y Entrevistas:

- Se llevaron a cabo reuniones periódicas con representantes de APREMA y miembros de diversas comunidades de regantes.
- A través de entrevistas y sesiones de brainstorming, los usuarios compartieron sus experiencias y dificultades con los sistemas actuales de gestión de recursos hídricos.

2. Análisis de Problemas:

- Los problemas principales identificados incluyen la dificultad para gestionar usuarios y parcelas, la complejidad en la gestión de contadores y facturación, y la falta de herramientas eficaces para el seguimiento de cobros y pagos.
- Otros desafíos incluyen la necesidad de mejorar la usabilidad y accesibilidad del software, y la importancia de contar con un sistema de información geográfica (SIG) para gestionar las parcelas y la red de distribución.

3. Documentación de Requisitos:

- Con la información recopilada, se procedió a documentar los requisitos funcionales y no funcionales.
- Estos requisitos fueron validados con los usuarios finales para asegurar que reflejan adecuadamente sus necesidades y expectativas.

4.1.2. Requisitos funcionales

Los requisitos funcionales describen las funciones y características específicas que debe tener un sistema para cumplir con las necesidades y expectativas de los usuarios. Estos requisitos detallan qué debe hacer el sistema y cómo debe comportarse en situaciones específicas.

- **Registro y Gestión de Usuarios y Parcelas:**

- Desarrollar una interfaz que permita a los usuarios registrar y gestionar una lista de comuneros y sus respectivas parcelas.
- Facilitar la visualización y edición de los datos de los comuneros y parcelas, asegurando la integridad y consistencia de la información.

- **Gestión de Contadores y Peticiones de Agua:**

- Simplificar el proceso de registro y gestión de contadores.
- Permitir a los usuarios realizar peticiones de agua de manera eficiente.

- **Generación de Facturas:**

- Proporcionar opciones de facturación personalizables por la comunidad, incluyendo la posibilidad de configurar IVA, recargos e intereses.
- Desarrollar la gestión de anticipos y atrasos en los pagos, facilitando el seguimiento y control financiero.

- **Registro de Documentos:**

- Mantener un registro detallado de todos los documentos que entran y salen de la comunidad, asegurando una gestión documental eficiente y organizada.

- **Emisión de Comunicaciones:**

- Implementar funcionalidades para la emisión de cartas o circulares personalizadas para cada comunero, mejorando la comunicación interna de la comunidad.

- **Inventario de Cultivos:**

- Registrar y gestionar un inventario de los cultivos de la comunidad.
- Facilitar el seguimiento del estado de los cultivos y la gestión de los recursos necesarios para su mantenimiento.

- **Gestión de Incidencias y Mantenimiento:**

- Registrar y gestionar las incidencias ocurridas en la red de distribución y las actuaciones de mantenimiento realizadas.

- **Estadísticas de Consumo de Agua:**

- Proporcionar herramientas para el análisis estadístico de los consumos de agua de la comunidad, apoyando la toma de decisiones informada.
- Desarrollar informes y visualizaciones que faciliten la comprensión y análisis de los datos de consumo.

■ **Estadísticas de Consumo de Agua:**

- Incluir un Sistema de Información Geográfica (SIG) para la consulta geográfica de toda la información almacenada en el programa.
- Facilitar la creación y gestión del parcelario y la red de distribución, apoyándose en ortofotografías digitales y otras fuentes de datos geoespaciales.

4.1.3. Requisitos no funcionales

Los requisitos no funcionales definen los criterios que se utilizan para evaluar el funcionamiento del sistema en áreas como el rendimiento, la usabilidad, la fiabilidad, la seguridad y la escalabilidad. Estos requisitos no describen funciones específicas, sino que establecen estándares y restricciones bajo los cuales el sistema debe operar.

■ **Escalabilidad:**

- La aplicación debe ser escalable para gestionar el crecimiento de la comunidad y la cantidad de datos.

■ **Usabilidad:**

- La interfaz debe ser intuitiva y fácil de usar para todos los miembros de la comunidad, independientemente de su habilidad técnica.

■ **Rendimiento:**

- El tiempo de respuesta para consultas y actualizaciones debe ser rápido.

■ **Seguridad:**

- Asegurar la protección de los datos almacenados y transmitidos mediante medidas de seguridad adecuadas.

■ **Mantenibilidad:**

- La aplicación debe ser fácilmente actualizable para incorporar nuevas características y corregir errores.

■ **Eficiencia:**

- Establecer niveles de rendimiento en aspectos como tiempo de respuesta y uso de recursos.

■ **Fiabilidad:**

- Establecer niveles de recuperabilidad y tolerancia a fallos para asegurar la continuidad del servicio.

■ **Portabilidad:**

- Garantizar que el sistema puede ser instalado, sustituido y adaptado a diferentes entornos de hardware y software.

4.2. Arquitectura de la Aplicación Web

La arquitectura de la aplicación web está diseñada para ser modular y escalable, asegurando que todos los componentes interactúen de manera eficiente. La aplicación se divide en tres capas principales: base de datos, frontend, backend. A continuación, se presenta un diagrama simplificado y describe cada componente clave de la arquitectura:



4.2.1. Frontend

El frontend, desarrollado en React.js[3], proporciona una interfaz de usuario interactiva y fácil de usar. Los componentes principales incluyen:

- **Dashboard:** Un panel de control para que los usuarios y administradores gestionen sus datos.
- **Formularios de Gestión:** Formularios que permiten la gestión de usuarios, parcelas, contadores, etc.
- **Visualización Geográfica:** Un mapa interactivo para la visualización y gestión de datos geográficos.

4.2.2. Backend

El backend, construido con Django[4], maneja la lógica de negocio y se comunica con el frontend a través de API RESTful. Los módulos principales incluyen:

- **API RESTful:** Proporciona endpoints para realizar operaciones CRUD.
- **Autenticación y Autorización:** Gestiona la seguridad del acceso a diferentes partes del sistema.
- **Gestión de Facturación:** Módulo para la creación y administración de facturas.
- **Módulo de Informes:** Genera informes y estadísticas sobre el consumo de agua.

4.2.3. Base de Datos

La base de datos PostgreSQL[9], con la extensión PostGIS[7], almacena datos estructurados y geoespaciales. Los componentes clave son:

- **Tablas de Datos:** Almacenan información de usuarios, parcelas, contadores, etc.
- **Índices Geoespaciales:** Mejoran la eficiencia de las consultas geográficas.

Ilustración 1: Arquitectura de la Aplicación Web

4.3. Base de datos

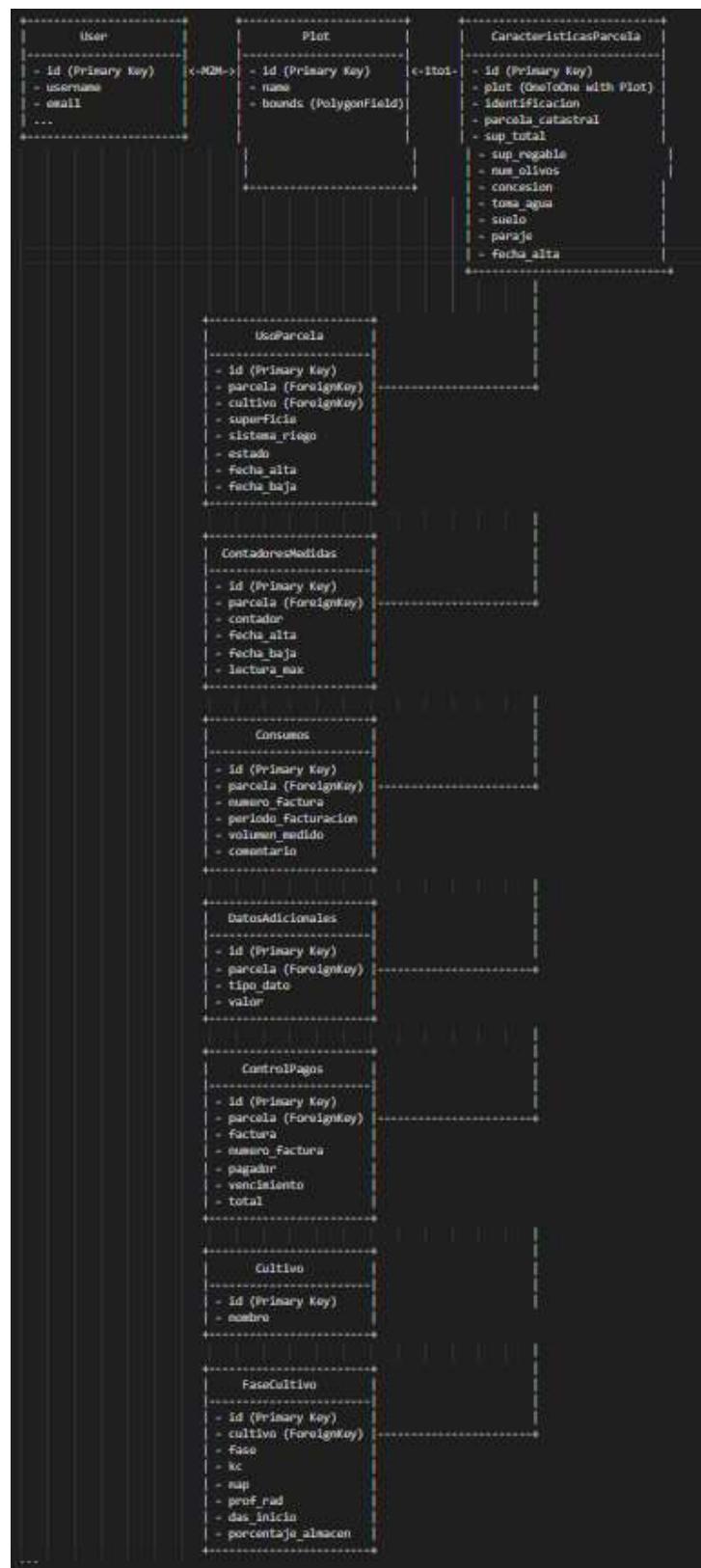


Ilustración 2: Diagrama de la base de datos

La base de datos de este proyecto es una base de datos geoespacial, específicamente utilizando PostGIS[7] sobre PostgreSQL[9]. Esta elección se debe a las capacidades avanzadas de PostgreSQL[9] para gestionar datos relacionales y la extensión PostGIS[7], que añade soporte para datos geoespaciales, lo que es esencial para el manejo de parcelas agrícolas.

PostGIS es una extensión de PostgreSQL[9] que convierte esta base de datos relacional en un sistema de información geográfica (SIG)[2]. Proporciona soporte para datos geoespaciales y permite realizar operaciones geográficas complejas. Las características clave incluyen:

- **Tipos de Datos Espaciales:** Incluye tipos de datos como `Geometry`, `Geography`, y subtipos específicos como `Point`, `LineString`, `Polygon`, etc.
- **Funciones Espaciales:** Proporciona una amplia gama de funciones para realizar operaciones espaciales como cálculos de distancias, intersecciones, uniones, etc.
- **Índices Espaciales:** Soporte para índices espaciales como GIST y SP-GIST para mejorar el rendimiento de las consultas espaciales.

4.3.1. Tablas Principales

- **User (Usuarios)**
 - **Campos:**
 - `id`: Clave primaria.
 - `username`: Nombre de usuario.
 - `email`: Correo electrónico del usuario.
 - **Relación:**
 - Un usuario puede estar asociado con múltiples parcelas (relación 1:N con la tabla Plot a través de UserParcela).
- **Plot (Parcelas)**
 - **Campos:**
 - `id`: Clave primaria.
 - `name`: Nombre de la parcela.
 - `bounds`: Representación geográfica de los límites de la parcela, utilizando el tipo de dato `PolygonField` de PostGIS.
 - **Relación:**
 - Cada parcela tiene un conjunto de características específicas (relación 1:1 con CaracteristicasParcela).
- **CaracteristicasParcela**
 - **Campos:**
 - `id`: Clave primaria.
 - `plot_id`: Clave foránea hacia Plot.

- **identificacion:** Identificación de la parcela.
- **parcela_catastral:** Información catastral de la parcela.
- **sup_total:** Superficie total.
- **sup_regable:** Superficie regable.
- **num_olivos:** Número de olivos.
- **concesion:** Información sobre concesiones.
- **toma_agua:** Información sobre toma de agua.
- **suelo:** Información adicional.
- **paraje:** Localización específica.
- **fecha_alta:** Fecha de alta de las características.

4.3.2. Tablas Relacionales

■ UsoParcela

• Campos:

- **id:** Clave primaria.
- **parcela:** Clave foránea hacia Plot.
- **tipo_uso:** Tipo de uso de la parcela.
- **cultivo:** Tipo de cultivo en la parcela.
- **superficie:** Superficie de cultivo.
- **sistema_riego:** Sistema de riego utilizado.
- **fecha_alta:** Fecha de alta de la relación.
- **fecha_baja:** Fecha de baja de la relación.

■ ContadoresMedidas

• Campos:

- **id:** Clave primaria.
- **parcela_id:** Clave foránea hacia Plot.
- **contador:** Identificación de contadores de la parcela.
- **fecha_alta:** Fecha de alta del contador.
- **fecha_baja:** Fecha de baja del contador.
- **lectura_max:** Registro de lectura de contador.

■ Consumos

• Campos:

- **id:** Clave primaria.

- parcela_id: Clave foránea hacia Plot.
- nuevo_facturado: Nueva cantidad facturada.
- periodo_facturacion: Período de facturación.
- volumen_medio: Consumo medio.
- comentario: Comentarios adicionales sobre el consumo.

■ **DatosAdicionales**

• **Campos:**

- id: Clave primaria.
- parcela_id: Clave foránea hacia Plot.
- tipo_dato: Tipo de dato adicional.
- valor: Valor del dato adicional.

■ **ControlPagos**

• **Campos:**

- id: Clave primaria.
- parcela_id: Clave foránea hacia Plot.
- factura: Número de factura.
- numero_factura: Número único de factura.
- pagador: Nombre del pagador.
- vencimiento: Fecha de vencimiento del pago.
- total: Total a pagar.

4.3.3. Relaciones

- **User (Usuario)** tiene una relación de 1:N con **Plot (Parcela)** a través de la tabla **UserParcela**, lo que indica que un usuario puede estar asociado con múltiples parcelas.
- **Plot (Parcela)** tiene una relación de 1:1 con **CaracteristicasParcela**, indicando que cada parcela tiene un conjunto de características específicas.
- **Plot (Parcela)** tiene relaciones 1:N con las tablas **ContadoresMedidas**, **Consumos**, **DatosAdicionales** y **ControlPagos**, indicando que cada parcela puede tener múltiples registros en estas tablas asociadas a contadores, consumos, datos adicionales y control de pagos, respectivamente.

4.4. Diseño de la Interfaz

4.4.1. Frontend

Para mejorar la experiencia y la conexión del usuario final con el campo, se ha utilizado un fondo verde claro en el diseño del fondo, en el encabezado de la web un color gris oscuro con letras blancas. Esta elección de color no solo busca hacer la interfaz más agradable y accesible, sino también reflejar el entorno natural y agrícola al que está destinado el sistema.

- **Página principal:**

- **Encabezado (Header):**

Fondo Gris oscuro con letras en blancas para darle mayor visibilidad.
Nombre de la organización: 'Junta Central de Usuarios GUADALUSERS'.
Menú de navegación: 'Inicio', 'Login' y 'Contacto'.

- **Barra Lateral (Sidebar):**

Enlaces de navegación a recursos externos: 'Junta de Andalucía', 'Consejería de Agricultura, Pesca, Agua y Desarrollo Rural', 'El Tiempo', 'Catastro' y 'Google Maps'.

- **Contenido Principal:**

Imagen Principal: Imagen destacada relacionada con la temática agrícola.
Sección de Historia: 'Historia de la Junta Central de Usuarios'.
Texto descriptivo: Explica la misión y el origen de la organización.

- **Tablón de Publicidad:**

Notificaciones importantes sobre eventos y anuncios relevantes.



Ilustración 3: Inicio aplicación

■ **Contacto:**

Esta página proporciona un formulario de contacto para que los usuarios puedan enviar mensajes al administrador de la comunidad, está compuesta por los campos para introducir nombre, email, el mensaje y un botón para enviar el formulario.

Ilustración 4: Página de Contacto

■ **Inicio de sesión:**

Esta página permite a los usuarios iniciar sesión en el sistema introduciendo su DNI y contraseña. Consta de campos para ingresar el DNI y la contraseña, un botón para iniciar sesión y un enlace debajo de este último para la recuperación de la contraseña.

Ilustración 5: Inicio de sesión

■ **Inicio de sesión con error:**

En caso de introducir incorrectamente algún dato en el inicio de sesión, aparecerá un mensaje de error.

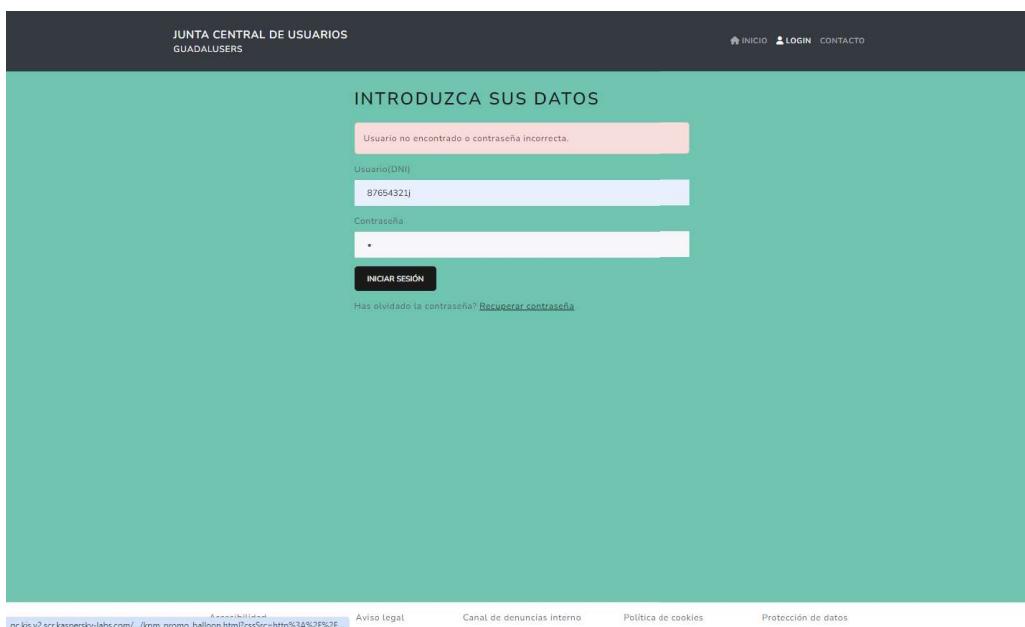


Ilustración 6: Inicio de sesión con error

■ **Restablecer contraseña:**

Al pulsar en el enlace de restablecer contraseña, se accede a esta página, la cual incluye un campo para introducir el email y un botón para enviarlo.

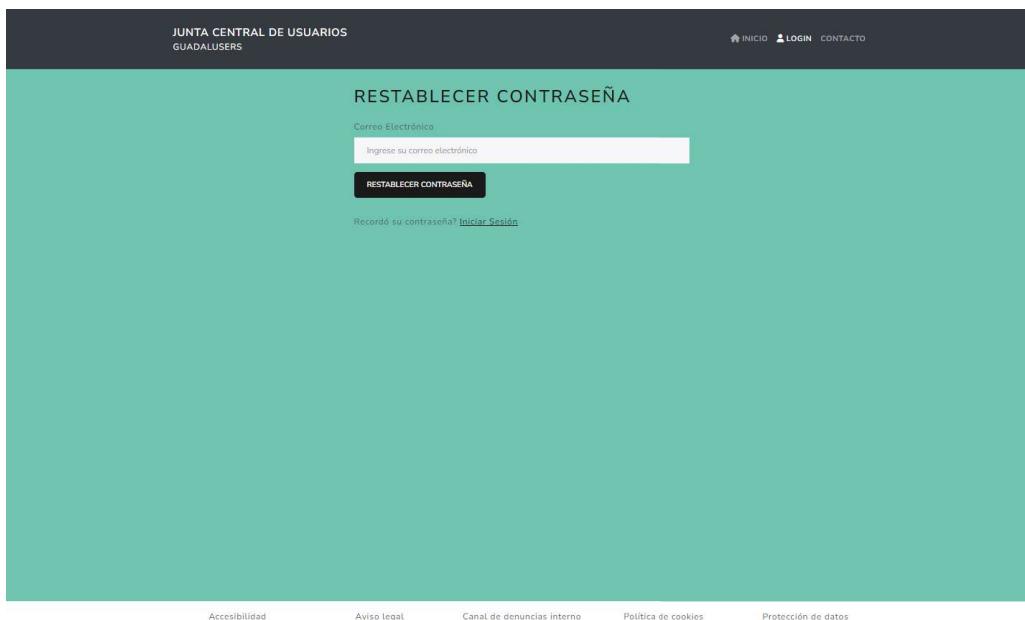


Ilustración 7: Restablecer contraseña

■ **Datos Comunidad de Regantes:**

Esta página muestra información detallada sobre una comunidad de regantes específica, incluyendo los datos de la comunidad (nombre, CIF, código postal, localidad, dirección, teléfonos y correo electrónico), datos bancarios (nombre de la entidad bancaria, número de banco, número de sucursal, dígito de control y número de cuenta) y del representante de la comunidad (CIF, nombre del cargo, teléfono y correo electrónico de contacto).

DATOS COMUNIDAD DE REGANTES

COMUNIDAD DE REGANTES ACEQUIA DE GRANADILLOS

Datos de la Comunidad	
Nombre: Comunidad de Regantes Acequia de Granadillos	CIF: 15896320G
Localidad: MÁLAGA	Código Postal: 29590
Dirección: ANTIGUA CARRETERA DE CARTAMA 23	Teléfono Móvil: 658202252
	Teléfono Fijo: 952178956
Email Alternativo: cracequidegranadillos@gmail.com	

Datos Bancarios				
Nombre de la entidad:	Número de banco:	Número de sucursal:	Dígito de control:	Número de cuenta:
CAJA RURAL DE GRANADA	0049	0023	17	0009876547

Cargos Comunidad		
CIF:	Nombre:	Cargo:
15896320G	Comunidad de Regantes Acequia de Granadillos	Gestión de Recursos
Telefono:	Email:	Móvil:
600987654	admincrgranadillos@gmail.com	658202252

Ilustración 8: Datos comunidad de regantes

■ **Datos del Usuario del Agua:**

Esta información incluye datos de la comunidad, como el nombre, CIF, dirección, código postal, localidad, teléfonos y correo electrónico. También proporciona datos bancarios, incluyendo el nombre de la entidad bancaria, número de banco, número de sucursal, dígito de control y número de cuenta. Además, muestra la ubicación de la parcela en un mapa.

DATOS COMUNIDAD DE REGANTES

DATOS DEL USUARIO DEL AGUA

Datos del Usuario

Nombre: Juan	CIF: 87654321L	C.P.: 05412
Dirección: calle manzanares 12	Localidad: malaga	
Teléfono Fijo: 987564324	Móvil: 654987234	
Email: juan@gmail.com		

Datos Bancarios

Nombre de la entidad:	Número de banco:	Número de sucursal:	Dígito de control:	Número de cuenta:
La caixa	2034	300	6	0001237654

Parcela Ejemplo 4

A map showing a specific parcel (Parcela Ejemplo 4) located in a rural area with various roads and landmarks labeled.

Ilustración 9: Datos usuario del agua

■ **Parcelario:**

Esta página permite seleccionar y visualizar información detallada sobre las parcelas. Los usuarios pueden seleccionar una parcela específica y obtener datos como la identificación, superficie total, número de árboles, parcela catastral, superficie regable, propietario, concesión, toma de agua, suelo, paraje y fecha de alta.

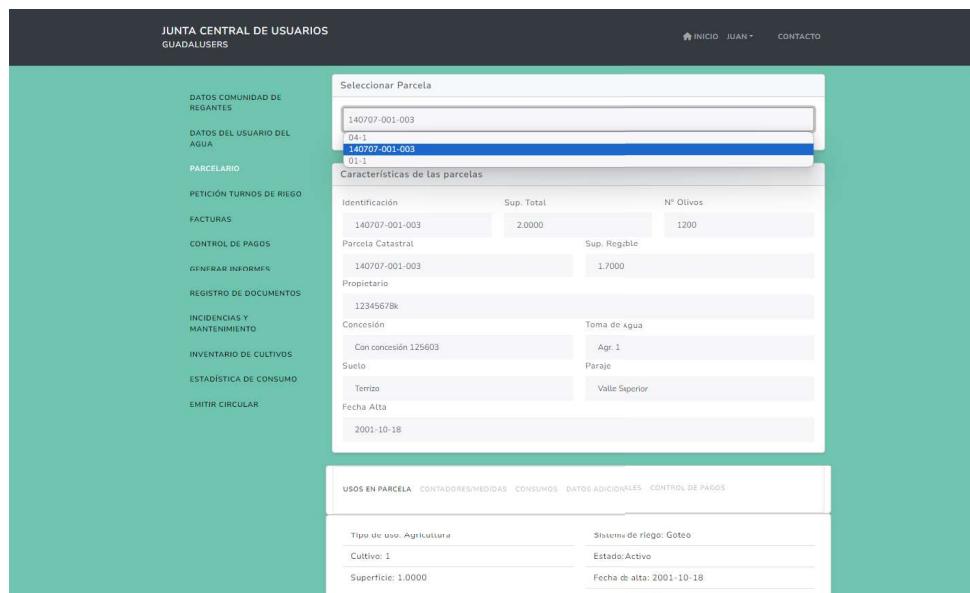


Ilustración 10: Parcelario

■ **Usos en parcela(Parcelario):**

Esta página presenta los diferentes usos de una parcela, incluyendo información sobre el tipo de uso, el cultivo, la superficie, el sistema de riego, el estado, y las fechas de alta y de baja.

USOS EN PARCELA		CONTADORES/MEDIDAS	CONSUMOS	DATOS ADICIONALES	CONTROL DE PAGOS
Tipo de uso: Agricultura				Sistema de riego: Goteo	
Cultivo: 1				Estado: Activo	
Superficie: 1.0000				Fecha de alta: 2001-10-18	
				Fecha de baja: Vacio	
Tipo de uso: Agricultura				Sistema de riego: Goteo	
Cultivo: 47				Estado: Activo	
Superficie: 3.0000				Fecha de alta: 2010-08-20	
				Fecha de baja: Vacio	

Ilustración 11: Usos en parcela(Parcelario)

■ **Contadores y medidas(Parcelario):**

Esta página presenta información sobre los contadores y sus medidas, incluyendo el identificador del contador, la fecha de alta, la fecha de baja y las lecturas máximas.

CONTADOR	FECHA ALTA	FECHA BAJA	LECTURA MAX
001003	2002-11-15	No proporcionado	1000000
001003	2002-11-15	No proporcionado	1000000
004007	2023-02-11	No proporcionado	750000

Ilustración 12: Contadores y medidas (Parcelario)

■ **Consumos(Parcelario):**

Esta página muestra información sobre las facturas de consumo, incluyendo el número de factura, el período facturado, el volumen medido y los comentarios.

Nº FACTURA	PERÍODO FACTURADO	VOL. MEDIDO	COMENTARIO
FAC001	2011-09	15000.00	Consumo de riego
FAC001	2011-09	15000.00	Consumo de riego

Ilustración 13: Consumos(Parcelario)

■ **Datos adicionales(Parcelario):**

Esta página presenta información sobre datos adicionales, incluyendo el nombre del dato y su valor correspondiente.

TIPO DE DATO	VALOR
PH del suelo	7.0
Nivel de Nitrógeno	3.5
Salinidad del agua	0.5

Ilustración 14: Datos adicionales(Parcelario)

■ **Control de pagos(Parcelario):**

Esta página muestra una tabla que permite a los usuarios ver y gestionar los pagos realizados y pendientes. La tabla contiene información detallada, incluyendo el tipo de factura, el número de factura, el ID/Usuario, el pagador, la fecha de vencimiento y el total en euros.

The screenshot shows a table with the following columns: FACTURA, N° FACTURA, ID USUARIO, PAGADOR, VENCIMIENTO, and TOTAL [EUROS]. There is one row of data:

FACTURA	N° FACTURA	ID USUARIO	PAGADOR	VENCIMIENTO	TOTAL [EUROS]
1	1	1	Juan Perez	2011-09-30	1450.50

Ilustración 15: Control de pagos(Parcelario)

■ **Petición Turnos de Riego:**

Esta página permite a los usuarios solicitar turnos de riego mediante un formulario que incluye un campo para ingresar la solicitud y un botón para enviarla. Más abajo, se muestra un listado de las peticiones con el mensaje, la fecha y el estado de procesamiento.

The screenshot shows a form for requesting irrigation turns. It includes a message input field labeled 'Mensaje' with placeholder text 'Ingresar su petición' and a 'ENVIAR' button. Below the form is a table titled 'MIS PETICIONES' with columns: MENSAJE, FECHA, and PROCESADO. One row of data is shown:

MENSAJE	FECHA	PROCESADO
Solicito riego del 10 al 14 de Agosto 9am a 12am	4/9/2024, 21:27:32	No

Ilustración 16: Petición turnos de riego

■ **Facturas:**

Esta página presenta un listado de facturas, proporcionando detalles específicos como el número de factura, la fecha de emisión y un botón para visualizar la factura.

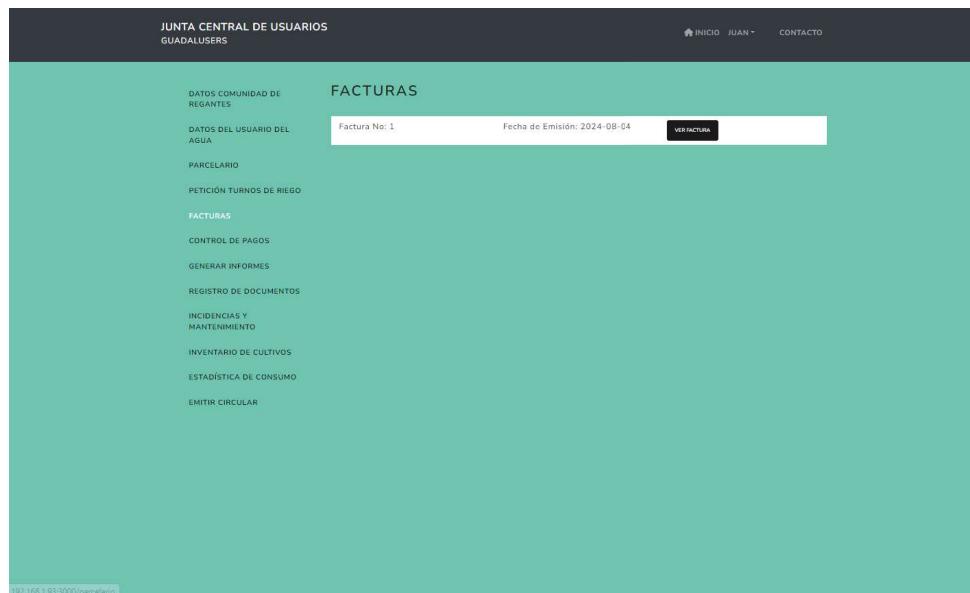


Ilustración 17: Facturas

■ **Mostrar factura:**

Al hacer clic en el botón "Ver Factura", se abre una nueva pantalla que muestra el documento de la factura. Este documento incluye información detallada como la fecha de emisión, la fecha de vencimiento, el estado de pago, los datos de la comunidad, los datos del usuario y los conceptos imputados. Más abajo, se presenta una gráfica del consumo de agua. En la parte superior del documento, se encuentra una barra de herramientas que permite imprimir o descargar la factura en formato PDF, además de ofrecer otras opciones adicionales.

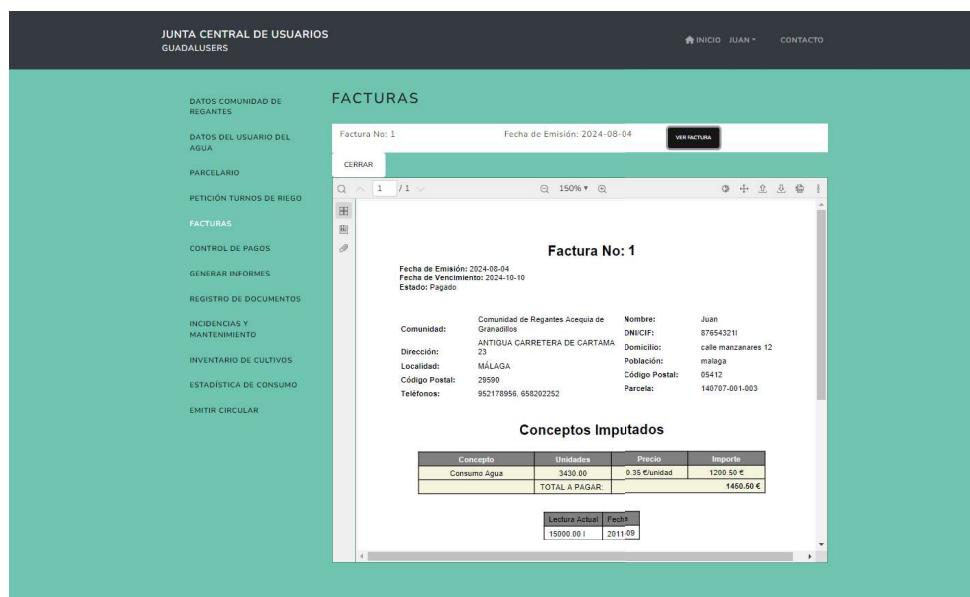


Ilustración 18: Factura para imprimir

■ Control de Pagos:

Esta página permite a los usuarios ver y gestionar los pagos realizados y pendientes mediante una tabla interactiva. La tabla incluye varios elementos: el Número, que es el identificador del pago; la Cuantía, que indica el monto del pago; la Fecha de Emisión, que muestra la fecha en que se emitió el pago; la Fecha de Pago, que indica cuándo se realizó el pago; la Forma de Pago, que detalla el método utilizado para realizar el pago; el estado de Pagado (Sí/No), que señala si el pago ha sido realizado; y el estado de Recargo (Sí/No), que indica si se ha aplicado algún recargo. Además, hay un botón en la columna de Acción que permite ver detalles de la factura.

NÚMERO	CUANTÍA	FECHA EMISIÓN	FECHA PAGO	FORMA DE PAGO	PAGADO (SÍ/NO)	RECARGO (SÍ/NO)	ACCIÓN
1	1450.50 €	2024-08-04	2024-09-09	N/A	Sí	Sí	VER FACTURA

Ilustración 19: Control de Pagos

■ Vista de Factura:

Al seleccionar "Ver Factura," se muestra un detalle de la factura que incluye la fecha de emisión y vencimiento, el estado del pago, los datos de la comunidad y del usuario, y los conceptos imputados.

Factura No: 1

Fecha de Emisión: 2024-08-04
Fecha de Vencimiento: 2024-10-10
Estado: Pagado

Comunidad:	Nombre:
Granadillas	Juan
ANTIGUA CARRETERA DE CÁRTAMA	DNI/CIIF: 67854321H
Dirección: 20000	Domicilio: calle manzanares 12
Localidad: MÁLAGA	Población: malaga
Código Postal: 29590	Código Postal: 05412
Teléfonos: 952178956, 658202252	Parcela: 140707-001-003

Conceptos Imputados

Concepto	Unidades	Precio	Importe
Consumo Agua	3430.00	0.35 Unidad	1200.50 €
		TOTAL A PAGAR	1450.50 €

Ilustración 20: Control de pagos factura

■ **Vista de Factura:**

Esta página permite a los usuarios generar informes en formato PDF. Incluye una tabla con los siguientes elementos: número de asiento, fecha del asiento, número de subcuenta, descripción del concepto, monto que se debe (Debe), monto a favor (Haber), nombre del titular de la subcuenta y un botón para descargar el informe en formato PDF. Este listado corresponde al libro diario que se visualiza en la página.

The screenshot shows a web application interface for 'JUNTA CENTRAL DE USUARIOS GUADALUSERS'. At the top, there's a navigation bar with links for 'INICIO', 'payment_report.pdf', 'Descargar', and a download warning message: 'Se ha bloqueado una descarga no segura'. The main content area has a teal header 'LISTADO DEL LIBRO DIARIO'. Below it is a table with columns: ASIENTO, FECHA, SUBCUENTA, CONCEPTO, DEBE, HABER, and TÍTULO DE LA SUBCUENTA. A single row of data is shown: ASIENTO 1, FECHA 4/8/2024, SUBCUENTA 4300001, CONCEPTO Factura 1/2024, DEBE 1450.50, HABER, and TÍTULO DE LA SUBCUENTA Juan. To the left of the table is a sidebar with various menu items: DATOS COMUNIDAD DE REGANTES, DATOS DEL USUARIO DEL AGUA, PARCELARIO, PETICIÓN TURNOS DE RIEGO, FACTURAS, CONTROL DE PAGOS, GENERAR INFORMES, REGISTRO DE DOCUMENTOS, INCIDENCIAS Y MANTENIMIENTO, INVENTARIO DE CULTIVOS, ESTADÍSTICA DE CONSUMO, and EMITIR CIRCULAR. A 'GENERAR PDF' button is located above the table.

Ilustración 21: Listado del libro diario

■ **Registro de documentos:**

En esta página se encuentra el formulario para el registro de documentos incluye los siguientes campos: número de documento para ingresar el número del documento, fecha para ingresar la fecha del documento, asunto para ingresar el asunto del documento, localización para ingresar la localización relacionada con el documento, dirigido a para ingresar el destinatario del documento, ruta de archivo para seleccionar un archivo a adjuntar, observaciones para ingresar observaciones adicionales y un checkbox para marcar si es un documento de entrada o salida.

The screenshot shows a web application interface for 'JUNTA CENTRAL DE USUARIOS GUADALUSERS'. At the top, there's a navigation bar with links for 'INICIO', 'JUAN', and 'CONTACTO'. The main content area has a teal header 'REGISTRO DE DOCUMENTOS'. Below it is a form with fields: 'Número de Documento' (with placeholder 'Ingrese número de documento'), 'Fecha' (with placeholder 'dd/mm/aaaa'), 'Asunto' (with placeholder 'Ingrese asunto'), 'Localización' (with placeholder 'Ingrese localización'), 'Dirigido a' (with placeholder 'Ingrese destinatario'), 'Ruta de Archivo' (with placeholder 'Ruta de Archivo'), 'Observaciones' (with placeholder 'Ingrese observaciones'), and checkboxes for 'Entrada' and 'Salida'. A 'ENVIAR' button is at the bottom. Below the form is a section titled 'LISTADO DE DOCUMENTOS' with a table showing one document entry: Número de Documento REG045632, Asunto MALAGA, Fecha 2024-08-04, Entrada SI, and Salida NO. A link 'Detalles del Documento' is also present. To the left of the form is a sidebar with the same menu items as in Illustration 21.

Ilustración 22: Registro de documentos 1

- **Visor documentos registrados:**

Debajo de la página de registro se encuentra el listado de documentos presentados incluye los siguientes elementos: número de documento, que sirve como identificador del documento; asunto, que indica el tema del documento; fecha del documento; entrada/salida, que especifica si es un documento de entrada o salida; y detalles del documento, que muestran información detallada como localización, destinatario, observaciones y archivo adjunto.

Ilustración 23: Registro de documentos 2

- **Incidencia y mantenimiento:**

La página de Incidencias y Mantenimiento permite a los usuarios reportar y visualizar incidencias relacionadas con el mantenimiento. Incluye un campo de texto para escribir la incidencia o el mantenimiento necesario, un botón Enviar para enviar el reporte, y una tabla de incidencias que lista las incidencias reportadas por el usuario, detallando el mensaje, la fecha y si han sido procesadas.

Ilustración 24: Incidencia y mantenimiento

■ **Inventario de Cultivos:**

Esta página muestra un listado de diferentes tipos de cultivos, cada uno con un botón de 'Detalles' asociado. Al hacer clic en 'Detalles', se despliega información adicional sobre el cultivo, incluyendo la fase de crecimiento, el coeficiente de cultivo (K_c), el nivel de agua en el perfil (NAP), la profundidad radicular, los días después de la siembra (DAS) y el porcentaje de almacén.

Ilustración 25: Información cultivos

■ **Inventario de Cultivos:**

Esta página de Estadística de Consumo presenta un gráfico que muestra la estadística de consumo de agua del usuario a lo largo del tiempo. El gráfico de consumo indica el volumen de agua medido en metros cúbicos (m^3) en diferentes períodos, con una línea que muestra la tendencia de consumo.



Ilustración 26: Estadística de consumo de agua del usuario

■ Emitir Circular:

Esta página permite emitir circulares informativas para los usuarios, e incluye un campo de texto enriquecido para escribir el contenido de la circular y un botón Imprimir para generar e imprimir la circular.

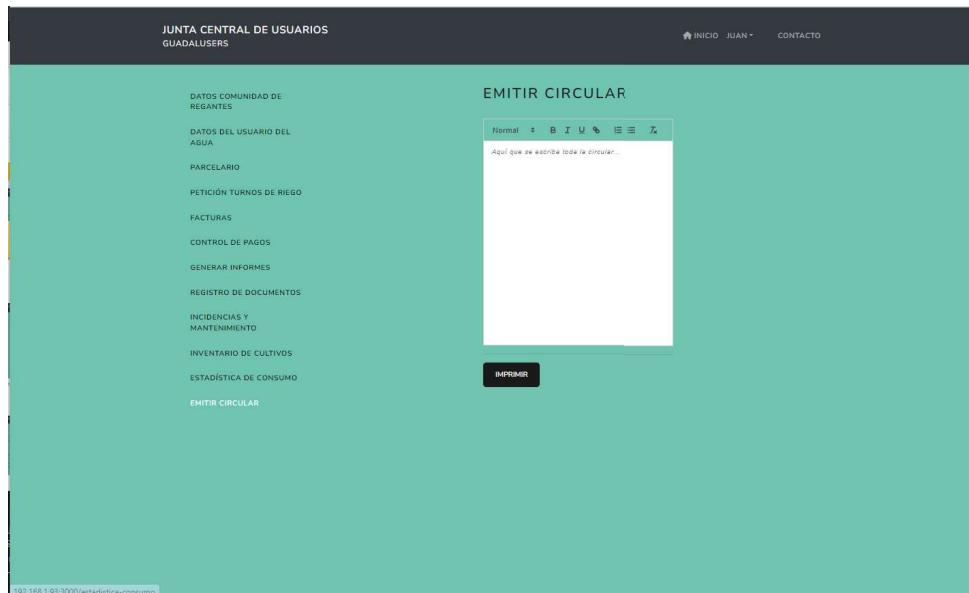


Ilustración 27: Emitir circular

4.4.2. Backend

Descripción General

Esta captura muestra la interfaz de administración del framework Django^[4], donde se pueden gestionar las diferentes entidades del sistema. El panel de administración es una herramienta potente que permite a los administradores del sitio realizar acciones CRUD (Crear, Leer, Actualizar, Eliminar) sobre los datos de la aplicación.

Secciones Principales

Autenticación y Autorización

- **Groups:** Permite la gestión de grupos de usuarios, facilitando la asignación de permisos.
- **Users:** Gestión de usuarios, incluyendo la creación, modificación y eliminación de cuentas de usuario.

Backend

- **Características Parcelas:** Gestión de las características asociadas a las parcelas.
- **Comunidades de Regantes:** Administración de las comunidades de regantes.
- **Consumos:** Control y registro de los consumos de agua.
- **Contact Messages:** Administración de los mensajes de contacto enviados por los usuarios.
- **Contadores Medidas:** Gestión de los contadores y sus mediciones.
- **Control Pagos:** Supervisión y registro de los pagos realizados por los usuarios.
- **Cultivos:** Administración de los diferentes tipos de cultivos gestionados.
- **Datos Adicionales:** Gestión de datos adicionales relevantes para la administración.
- **Facturas:** Registro y administración de las facturas emitidas.
- **Fase Cultivos:** Gestión de las diferentes fases de los cultivos.
- **Incidencias:** Registro y administración de incidencias reportadas.
- **Locations:** Administración de las ubicaciones geográficas relacionadas.
- **Perfiles:** Gestión de los perfiles de usuario.
- **Peticiones:** Administración de las peticiones de los usuarios.
- **Plots:** Gestión de las parcelas registradas.
- **Registros de Documentos:** Administración de los documentos registrados en el sistema.

- **Uso Parcelas:** Gestión de los diferentes usos asignados a las parcelas.

Acciones Recientes

En la parte derecha de la pantalla, se muestra una sección titulada "Recent actions" que lista las acciones recientes realizadas por el administrador.

Análisis de la Interfaz

El panel de administración de Django [4] proporciona una interfaz intuitiva y organizada para la gestión de los datos del sistema. Las entidades están categorizadas de manera clara, permitiendo un acceso rápido y eficiente a las diferentes secciones. La posibilidad de ver acciones recientes facilita el seguimiento de las modificaciones y operaciones realizadas, mejorando la capacidad de auditoría y administración del sistema.

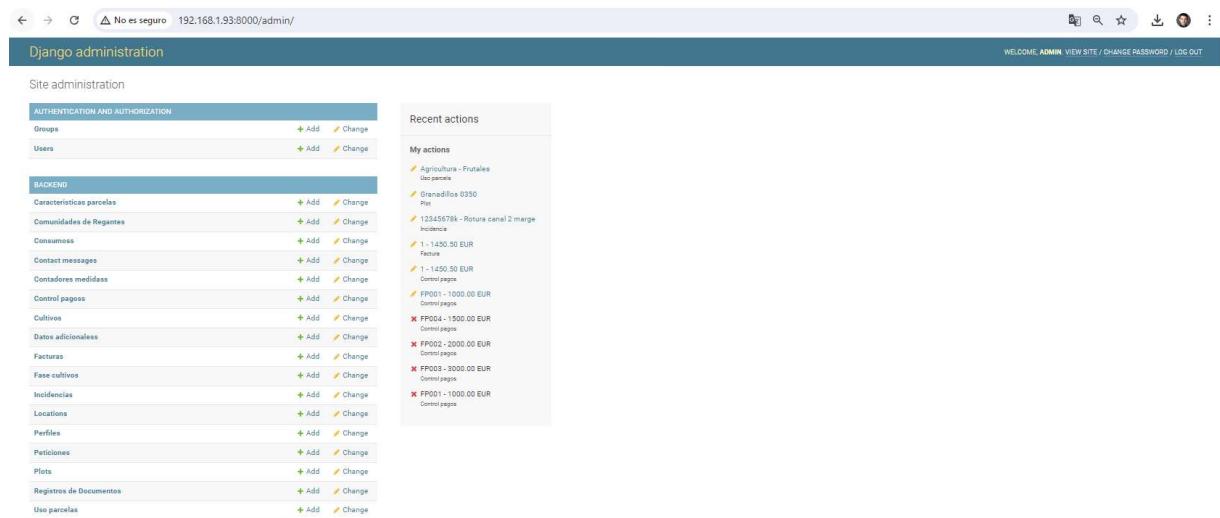


Ilustración 28: backend principal

Usuario:

En la pantalla de usuarios, se pueden gestionar de manera eficiente todos los aspectos relacionados con los usuarios del sistema: introducir nuevos datos, eliminar usuarios existentes, crear nuevos perfiles, modificar información actual y visualizar una lista completa de todos los usuarios registrados. Aunque solo muestro esta interfaz, todas las demás pantallas del sistema son prácticamente iguales en términos de diseño y funcionalidad.

USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
12345678k	juan@gmail.com	Juan		✗
23456789h	luis@gmail.com	Luis		✓
87654321j	maria@gmail.com	Maria		✗
admin	admin@example.com			✓

4 users

Ilustración 29: backend modificar usuario

Ilustración 30: backend gestionar informes

CAPÍTULO 5

Infraestructura de desarrollo

Para desarrollar y utilizar la aplicación, es esencial instalar y configurar diversas herramientas.

5.1. Linux Debian

Debian es una distribución del sistema operativo GNU/Linux, reconocida por su estabilidad, seguridad y vasto repositorio de software. Fue fundada en 1993 por Ian Murdock y es mantenida por una comunidad de desarrolladores voluntarios. Debian sirve como base para otras distribuciones populares, como Ubuntu.

Para la instalación de Linux Debian se ha utilizado un ordenador personal en el que se ha realizado una partición del disco duro. A continuación, se procedió a realizar una instalación desde cero utilizando la imagen oficial proporcionada en la web oficial de Debian, específicamente la versión 12 (Bookworm).

<https://www.debian.org/download>



Ilustración 31: Página web de descarga Linux Debian

En la página de descarga (Ilustración 31), seleccionamos el tipo de dispositivo que

utilizamos. En este caso, se selecciona la imagen de instalación para PC de 64 bits (amd64).

Al hacer clic en el enlace de descarga, se abrirá una nueva ventana donde comenzará automáticamente la descarga de la imagen ISO para la versión 12.6.0 (netinst).

▪ Creación de un Medio de Instalación:

- Una vez descargada la imagen ISO, se utiliza un programa de creación de medios de instalación (como Rufus) para crear un USB de arranque. Se debe seguir el proceso indicado por el programa para transferir la imagen ISO al USB.

▪ Instalación de Debian:

- Reiniciar el ordenador y arrancar desde el USB creado. Seguir las instrucciones del instalador de Debian, seleccionando las opciones adecuadas para particionar el disco duro y configurar el sistema.
- Asegurarse de seleccionar la versión adecuada y configurar los parámetros de red, usuario y contraseñas según las necesidades del proyecto.
- Al finalizar la instalación, Debian Linux versión 12 (Bookworm) estará listo para su uso. Esta configuración básica proporciona un entorno robusto y seguro para el desarrollo y la implementación de proyectos de software.

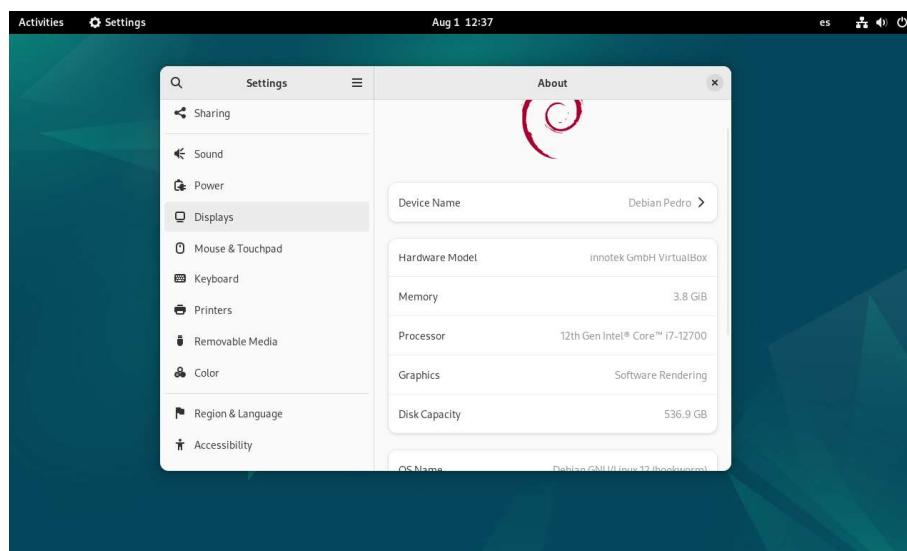


Ilustración 32: Debian instalado en ordenador

5.2. Visual Studio Code

Para el desarrollo del proyecto, se utilizará Visual Studio Code (VS Code)[\[10\]](#) como entorno de desarrollo integrado (IDE). VS Code[\[10\]](#) es ampliamente utilizado debido a sus interesantes plugins que facilitan el desarrollo de software y a su terminal integrada que permite ejecutar comandos directamente desde el editor.

Para descargar Visual Studio Code, se debe acceder a la web oficial a través del siguiente enlace:

<https://code.visualstudio.com/Download>



Ilustración 33: Página web de descarga Visual Studio Code [10]

Se realiza la instalación en linux mediante comandos y se obtiene el programa instalado:

En la página de descarga (Ilustración 30), seleccionamos el tipo de dispositivo que utilizamos y, al hacer clic en el botón correspondiente, se abrirá una nueva ventana donde comenzará automáticamente la descarga del instalador. Para este caso, hay que descargar la versión para Debian .deb e instalarlo mediante comandos de terminal sudo apt install code.deb en este caso, puede variar dependiendo versión y sistema usado.

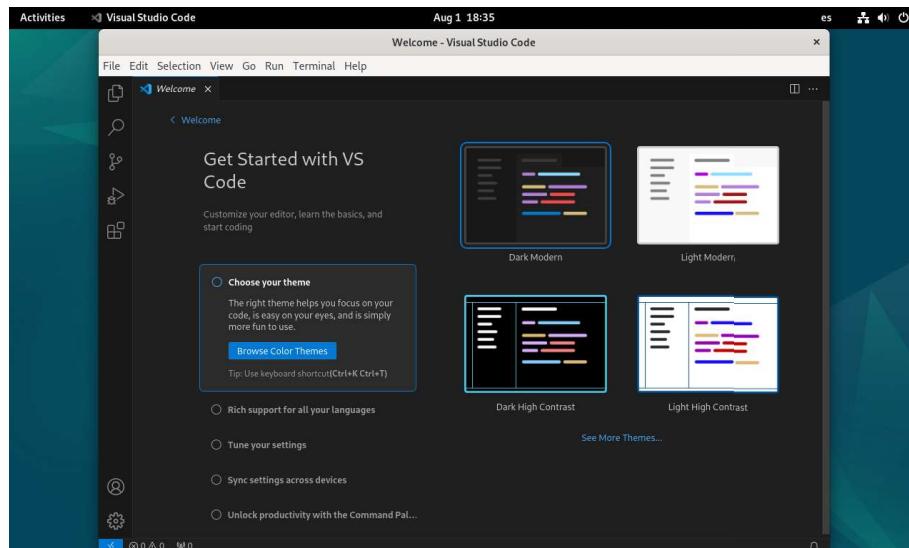


Ilustración 34: Visual Studio Code Instalado

5.3. Python

Python[8] es un lenguaje de programación interpretado, ampliamente utilizado por su simplicidad y legibilidad. Es ideal tanto para principiantes como para expertos debido a su

sintaxis clara y su extenso ecosistema de bibliotecas y herramientas. En esta versión de Debian, Python[8] ya viene preinstalado, eliminando la necesidad de realizar una instalación adicional. Esto facilita el desarrollo de aplicaciones, ya que el entorno de programación está listo para usar inmediatamente después de instalar el sistema operativo.



```
osboxes@osboxes:~$ python3 --version
Python 3.11.2
osboxes@osboxes:~$
```

Ilustración 35: Terminal con la versión de Python utilizada

5.4. Django

Django[4] es un framework de alto nivel para el desarrollo de aplicaciones web, conocido por su simplicidad y eficiencia. Escrito en Python[8], facilita la creación de aplicaciones web seguras y mantenibles.

Para instalar Django[4] en Debian, puedes utilizar dos métodos: desde el panel de extensiones de Visual Studio Code [10] o mediante comandos en la terminal.

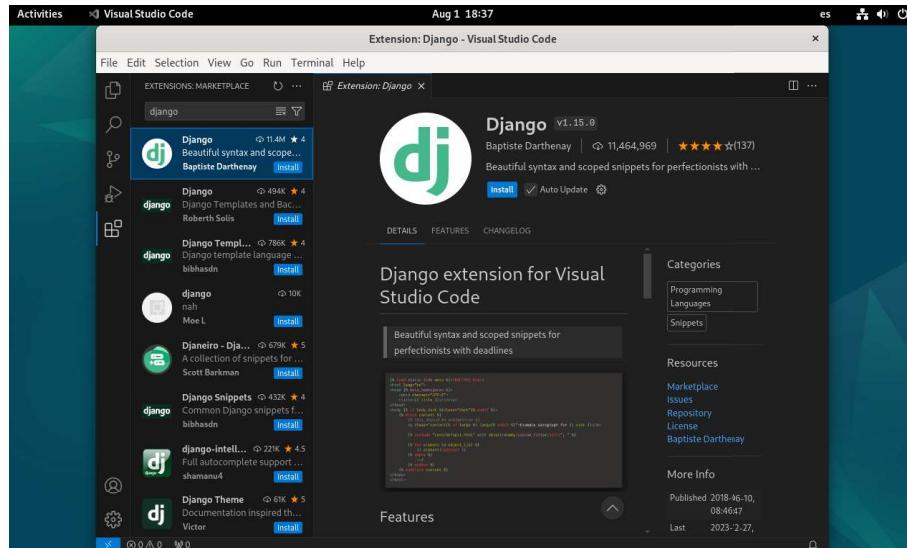


Ilustración 36: DJango VScode instalación

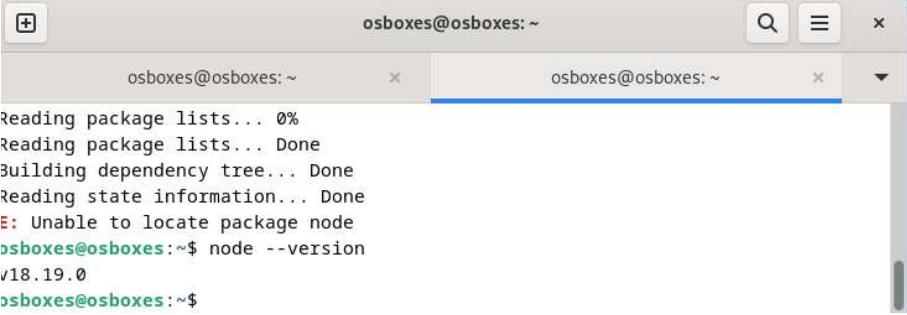
5.5. JavaScript

JavaScript[11] es un lenguaje de programación interpretado que se utiliza ampliamente para crear contenido web interactivo. Es una tecnología clave junto con HTML[5] y CSS[6] para el desarrollo web, estas dos últimas no requieren instalación específica ya que son lenguajes de marcado y estilo que se pueden escribir en cualquier editor de texto.

Para instalar y utilizar JavaScript[11] en tu entorno de desarrollo, puedes seguir estos métodos:

Instalación mediante Visual Studio Code [10] Puedes instalar extensiones relacionadas con JavaScript[11] desde el panel de extensiones de Visual Studio Code para mejorar la productividad y obtener características adicionales como linters y formateadores de código.

En este caso la instalación se realiza mediante terminal, para eso hay que actualizar primero los repositorios , a continuación instalar Node.js[11].



```
osboxes@osboxes: ~
osboxes@osboxes: ~
Reading package lists... 0%
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package node
osboxes@osboxes:~$ node --version
v18.19.0
osboxes@osboxes:~$
```

Ilustración 37: Version instalada de Node.js

5.6. Docker

Para la instalación de Docker[12] es mas rapido y sencillo instalarlo desde el panel de extensiones de Visual Studio Code[10], igualmente se puede realizar sin problemas su instalación mediante comandos.

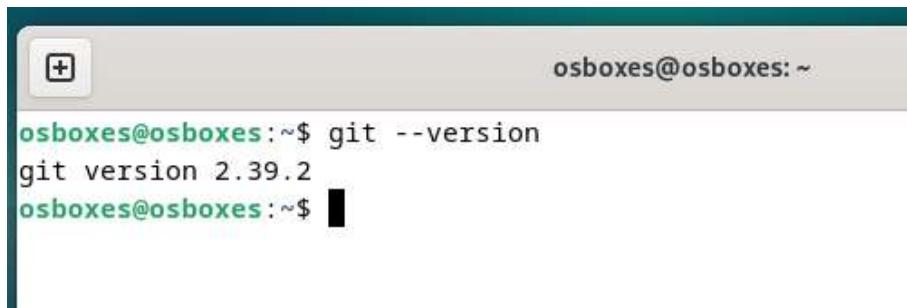


Ilustración 38: VSCode instalación docker

5.7. Git

Git es un sistema de control de versiones distribuido, utilizado para gestionar el código fuente a lo largo de su desarrollo. Permite a los desarrolladores rastrear cambios, colaborar en proyectos y mantener un historial detallado del desarrollo.

En esta versión de Debian se encuentra ya instalada.



```
osboxes@osboxes:~$ git --version
git version 2.39.2
osboxes@osboxes:~$
```

Ilustración 39: Version Git instalada

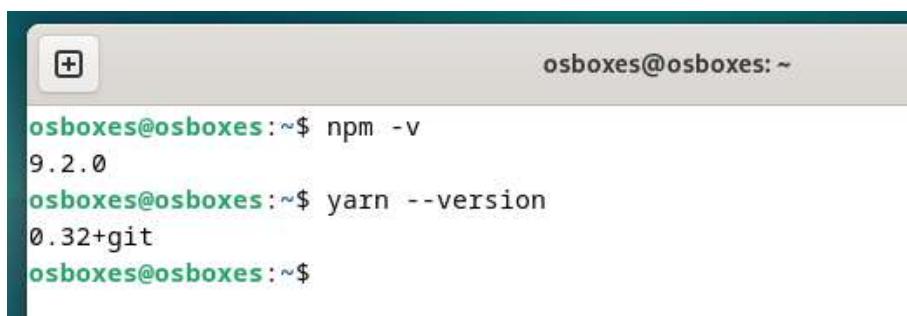
5.8. npm/yarn

Npm[13] y Yarn[14] son gestores de paquetes para JavaScript[11] que facilitan la instalación y gestión de dependencias en proyectos de Node.js[11].

Para este proyecto se han instalado ambos gestores de paquetes por terminal mediante los comandos en terminal:

```
sudo apt install npm
```

```
sudo apt install yarn
```



```
osboxes@osboxes:~$ npm -v
9.2.0
osboxes@osboxes:~$ yarn --version
0.32+git
osboxes@osboxes:~$
```

Ilustración 40: Versiones instaladas de NPM y YARN

5.9. GESTAGUA-CAP

GESTAGUA-CAP[1] es un software especializado para la gestión integral de comunidades de regantes. A continuación, se detallan los pasos para su adquisición, para el

desarrollo de la aplicación ya se encontraba instalado con anterioridad en el ordenador utilizado, al trabajar conjuntamente con las comunidades de regantes:

■ **Adquisición de GESTAGUA-CAP:**

- Para adquirir GESTAGUA-CAP[1], es necesario enviar un correo electrónico a la Consejería de Agricultura, Pesca y Desarrollo Rural de Andalucía, indicando el nombre de la comunidad de regantes y un teléfono de contacto.
- La dirección de correo es:

aplicacion.regadios.agapa@juntadeandalucia.es



Ilustración 41: GESTAGUA-CAP

La Consejería de Agricultura, Ganadería, Pesca y Desarrollo Sostenible ofrece esta aplicación de forma totalmente gratuita a todas aquellas Comunidades de Regantes que se encuentren interesadas en hacer uso de la misma para llevar a cabo su gestión.

5.10. WSL

Para desarrollar y utilizar la aplicación en un entorno Windows que permita ejecutar herramientas y entornos de Linux, es esencial instalar y configurar el Subsistema de Windows para Linux (WSL), se puede instalar desde Visual Studio Code.

WSL es una capa de compatibilidad para ejecutar binarios de Linux de forma nativa en Windows 10 y Windows Server 2019. Proporciona una forma de utilizar un entorno Linux directamente en Windows, sin la necesidad de una máquina virtual.

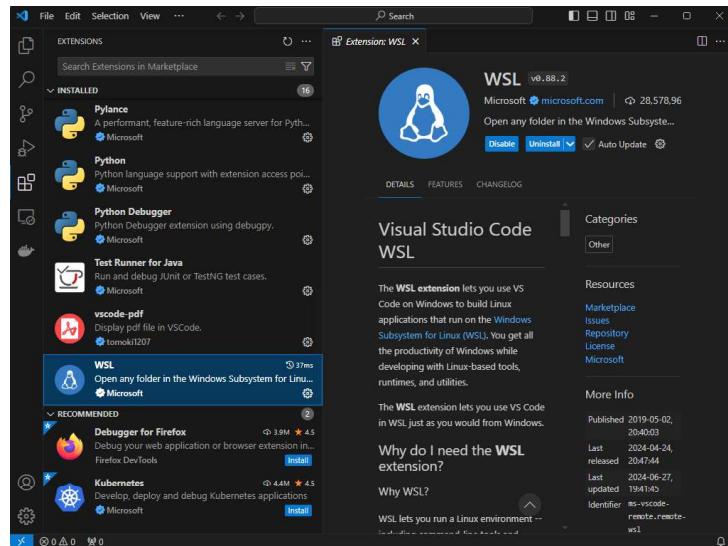


Ilustración 42: VScode instalación DJango

Con WSL instalado y configurado, tendrás acceso a un entorno de desarrollo Linux completo directamente en tu máquina Windows. Esto te permitirá utilizar herramientas y flujos de trabajo de Linux mientras trabajas en un sistema Windows, proporcionando lo mejor de ambos mundos para el desarrollo de software.

5.11. REACT

React^[3] es una biblioteca de JavaScript desarrollada por Facebook que se utiliza para construir interfaces de usuario (UI). Es especialmente útil para desarrollar aplicaciones web de una sola página (SPA) donde los datos cambian con el tiempo. React^[3] permite construir componentes reutilizables que gestionan su propio estado.

- **Requisitos Previos:**

Antes de instalar React^[3], asegúrate de tener instalado Node.js^[11] y Npm (Node Package Manager)^[13] en tu sistema, ya que estos son necesarios para gestionar paquetes y dependencias de JavaScript^[11].

- **Instalación de React usando NPM:**

Create React^[3] App es una herramienta oficial que te permite crear un nuevo proyecto de React con una configuración predeterminada ('npx create-react-app my-app').

- **Iniciar el servidor de desarrollo:**

Una vez en el directorio del proyecto mediante terminal usando comando 'cd + directorio del proyecto', ejecutar 'npm start', este comando abrirá tu nueva aplicación de React^[3] en el navegador web, generalmente en 'http://localhost:3000'.

Estructura del proyecto:

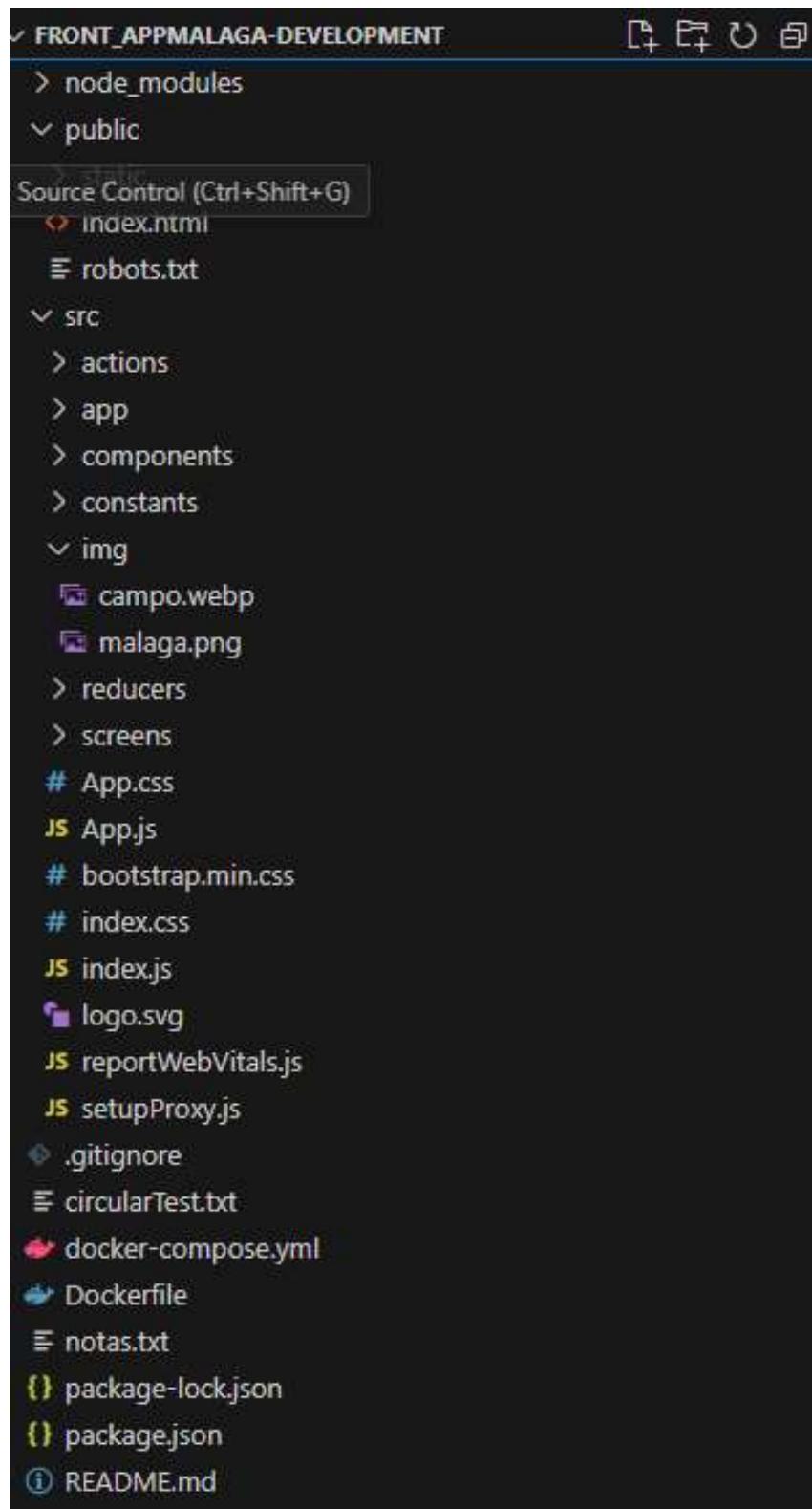


Ilustración 43: Estructura react

- **src/:**

Contiene los archivos de código fuente de React [3], como componentes y estilos.

5.12. REDUX

Redux[15] es una biblioteca de administración del estado para aplicaciones JavaScript[11]. Es comúnmente usada con React para manejar estados globales de una aplicación. Redux[15] ayuda a gestionar estados de la aplicación de manera predecible y centralizada, lo que facilita el desarrollo de aplicaciones complejas con estados compartidos entre múltiples componentes.

- **Instalación de Redux usando NPM:**

Navegar al directorio de tu proyecto de React[3] 'cd + directorio del proyecto', una vez en el ejecutar desde la terminal 'npm install redux react-redux'

CAPÍTULO 6

Implementación

La estructura de la aplicación está basada en una arquitectura web moderna, donde el cliente hace una petición de la página web a través de su navegador y el servidor le responde enviándole la página HTML[5] generado dinámicamente. A partir de ahí, comienza una comunicación cliente-servidor basada en el protocolo HTTP[16] (Hypertext Transfer Protocol) para la comunicación entre el servidor web (Django[4]) y los clientes. La base de datos utilizada es PostgreSQL[9] con la extensión PostGIS [7] para datos geoespaciales.

Se utiliza docker para simplificar la implementación:

Docker[12] es una plataforma de software que facilita la creación y administración de entornos de desarrollo y producción en contenedores. Un contenedor es una unidad estándar de software que empaqueta el código y todas sus dependencias para que la aplicación se ejecute de manera rápida y confiable en diferentes entornos de computación. Docker[12] es especialmente útil para desarrolladores web que necesitan un entorno consistente para probar y desarrollar sus aplicaciones antes de desplegarlas en producción.

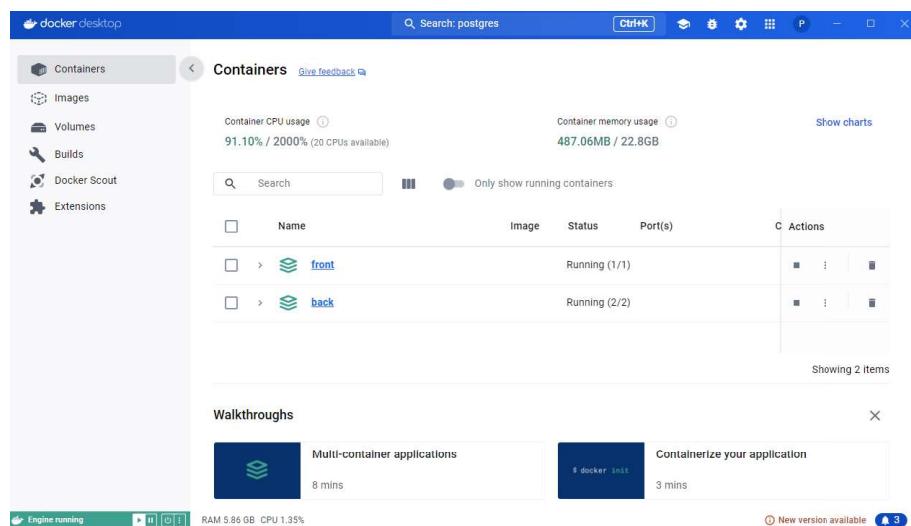


Ilustración 44: Docker

La implementación del proyecto utilizando tecnologías modernas como Django[4], React[3], Docker[12] y PostgreSQL[9] con PostGIS[7] ha permitido desarrollar un sistema eficiente y

escalable para la gestión de recursos hídricos. La arquitectura modular del sistema facilita la adición de nuevas funcionalidades y la mantenibilidad del código, asegurando que el proyecto pueda evolucionar para satisfacer futuros requisitos.

6.1. Implementación en Debian o Windows:

A continuación, se presenta un ejemplo detallado de cómo implementar la aplicación en un entorno Debian o Windows. Este ejemplo asume que ya has instalado y configurado todo el software necesario tal como se describió en la sección de Infraestructura de Desarrollo. A continuación se muestran los pasos ordenadamente:

■ Abrir el Proyecto en VS Code:

- Inicia Visual Studio Code (VS Code[10]) y abre las carpetas correspondientes al frontend y al backend del proyecto.
 - Asegúrate de que Docker[12] esté ejecutándose en tu sistema.

■ Ejecutar Docker Compose:

- Abre una terminal dentro de VS Code[10].
 - Ejecuta el siguiente comando para levantar los contenedores de Docker[12] 'docker-compose up'. Este comando construirá y ejecutará los contenedores definidos en el archivo docker-compose.yml.

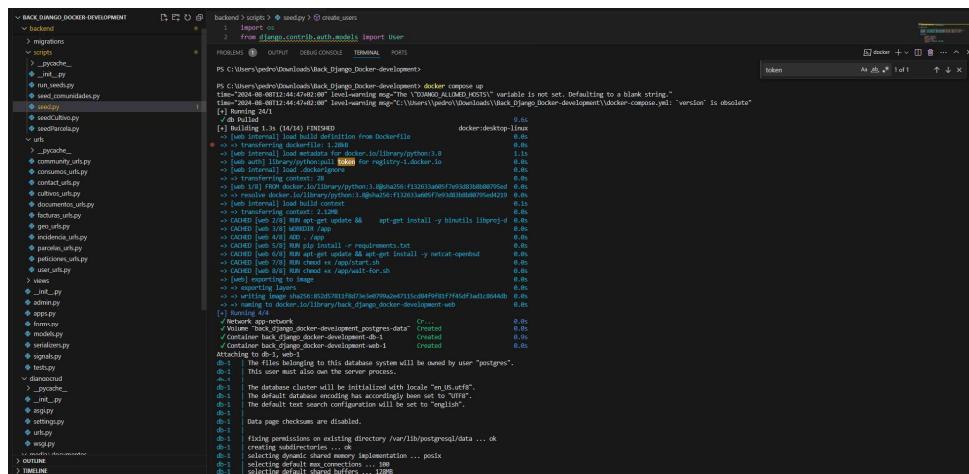


Ilustración 45: VScode Frontend ejecución docker

Ilustración 46: VScode Backend ejecución docker

■ Acceso a la Aplicación:

- **Frontend:** Una vez que Docker[12] haya iniciado correctamente los contenedores, abre tu navegador web y dirígete a 'http://localhost:3000'.



Ilustración 47: <http://localhost:3000>

- **Backend:** Para acceder a la interfaz de administración de Django[4], abre tu navegador y dirígete a 'http://localhost:8000/admin'.

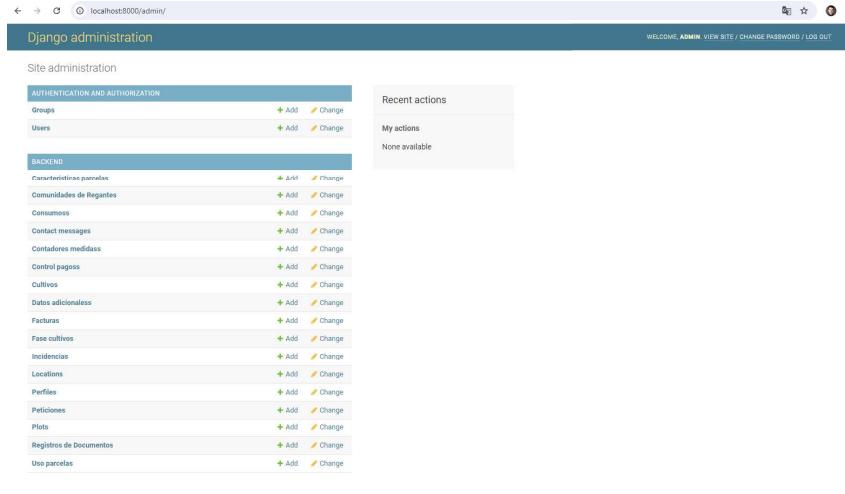


Ilustración 48: http://localhost:8000/admin

- **Documentación de la API:** Si necesitas acceder a la generada con Swagger[17], puedes hacerlo navegando a 'http://localhost:8000/swagger'.

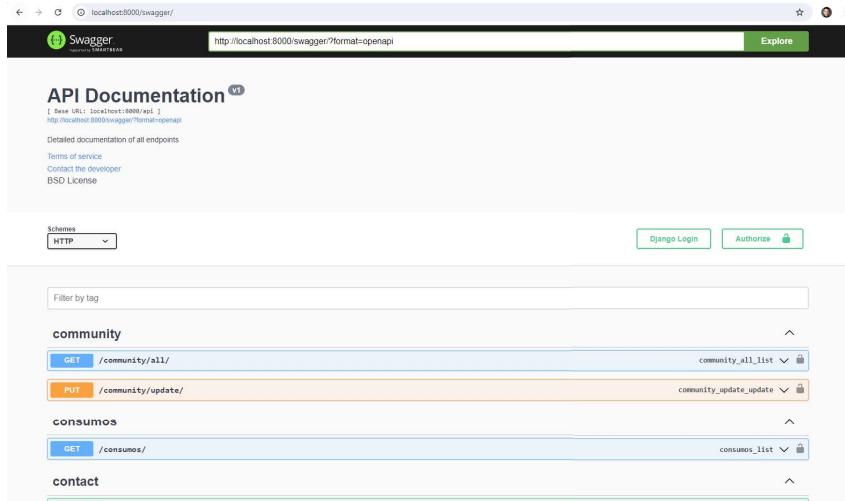


Ilustración 49: http://localhost:8000/swagger

Implementar este proyecto en un entorno Debian o Windows es un proceso directo gracias al uso de Docker[12]. Docker Compose[18] facilita la orquestación de los contenedores necesarios para el frontend y el backend, permitiendo a los desarrolladores centrarse en el desarrollo y la prueba de la aplicación en un entorno controlado y consistente. Además, el uso de VS Code[10] proporciona una interfaz de desarrollo amigable y poderosa, que soporta todas las necesidades de este proyecto.

Este enfoque no solo agiliza la configuración y el despliegue, sino que también asegura que el entorno de desarrollo sea lo más cercano posible al entorno de producción, reduciendo así las discrepancias y potenciales problemas que puedan surgir al mover la aplicación de desarrollo a producción.

CAPÍTULO 7

Conclusiones y trabajo futuro

El desarrollo de este Trabajo de Fin de Grado ha dado como resultado una aplicación web eficiente para la gestión de recursos hídricos en comunidades de regantes. La aplicación, que utiliza tecnologías modernas como Django[4], React[3], Docker[12] y PostgreSQL[9] con PostGIS[7], ha demostrado ser capaz de manejar de manera efectiva la información relacionada con la distribución de agua, permitiendo una gestión optimizada de los recursos. El sistema es escalable y modular, lo que facilita su mantenimiento y la adición de nuevas funcionalidades en el futuro. Las pruebas realizadas han validado su correcto funcionamiento, cumpliendo con los objetivos planteados al inicio del proyecto. El proyecto tiene un amplio potencial de mejora y expansión. Algunas de las áreas de trabajo futuro incluyen:

- **Optimización del Algoritmo de Gestión:**
Mejorar el algoritmo para incluir más variables y condiciones que reflejen situaciones más realistas, como la gestión en tiempo real del sistema de riego.
- **Implementación de Nuevas Funcionalidades:**
Integrar características adicionales como la predicción del consumo de agua basada en datos históricos y condiciones climáticas.
- **Mejoras en la Seguridad:**
Fortalecer la seguridad de la aplicación, tanto en la capa de servidor como en la interfaz web, para proteger mejor la información sensible manejada por el sistema.
- **Expansión de la Base de Datos:**
Ampliar la base de datos para incluir más información geoespacial y datos históricos que mejoren la capacidad de análisis y toma de decisiones.
- **Desarrollo de una Aplicación Móvil:**
Crear una versión móvil de la aplicación para facilitar el acceso y gestión por parte de los usuarios desde dispositivos móviles.
- **Recolección de Datos mediante Monitorización:**
Implementar sistemas de monitorización que recolecten datos en tiempo real sobre el uso de agua, el estado de las infraestructuras, y otros factores relevantes, permitiendo un análisis más detallado y una gestión proactiva.

■ **Conexión Directa con el Catastro:**

Establecer una conexión directa con el Catastro para la obtención automática de datos de parcelas y propiedades, lo que simplificaría el proceso de actualización de la base de datos y mejoraría la precisión de la información disponible en el sistema.

CAPÍTULO 8

Pruebas

En este capítulo se detallan las pruebas realizadas a la API desarrollada para el proyecto de fin de grado, utilizando Swagger[17] como herramienta principal para la documentación y prueba de los endpoints. El objetivo de estas pruebas es asegurar que la API funcione correctamente, cumpla con los requisitos especificados y maneje adecuadamente las diferentes solicitudes y respuestas.

8.1. Swagger:

Para llevar a cabo las pruebas de la API se utilizó Swagger[17], una herramienta que permite documentar y probar de manera interactiva los endpoints de una API. Swagger[17] facilita la visualización de la estructura de la API, así como la ejecución de pruebas directamente desde la interfaz web.

8.1.1. Configuración Inicial:

- **Acceso a Swagger:**

La API está documentada y accesible a través de la interfaz de Swagger[17] en la siguiente URL: '<http://192.168.1.93:8000/swagger/>'. Esta interfaz proporciona una visión completa de todos los endpoints disponibles, permitiendo realizar solicitudes de prueba y observar las respuestas.

- **Autenticación:**

Algunos endpoints requieren autenticación. Para acceder a estos, es necesario utilizar el botón 'Authorize' en Swagger [17] e ingresar las credenciales proporcionadas. Esto asegura que solo usuarios autorizados puedan acceder a funciones sensibles de la API. Si ves una cerradura al lado del método, significa que necesitas estar autenticado. Ve a la parte superior derecha de la interfaz de Swagger [17] y haz clic en 'Authorize'. Aparecerá una ventana emergente donde debes introducir el Bearer Token. Este token es generalmente obtenido al hacer login a través de un endpoint de autenticación (por ejemplo, POST /users/login/).

- Pantalla principal Swagger:

The screenshot shows the Swagger UI interface for a RESTful API. At the top, there's a header with the Swagger logo, the URL `http://192.168.1.93:8000/swagger/?format=openapi`, and a green "Explore" button. Below the header, the title "API Documentation v1" is displayed, along with a note about detailed documentation of all endpoints and links to terms of service, developer contact, and BSD license.

The main area is organized into sections: "community", "consumos", "contact", "documentos", "facturas", "geo", and "incidencias". Each section contains a list of API endpoints with their methods (e.g., GET, POST, PUT) and URLs. For example, under "consumos", there are endpoints for listing consumos (`/consumos/`) and creating a new consumos (`/consumos/nuevo/`). Under "facturas", there are endpoints for generating payment PDFs (`/facturas/generate/payment/pdf/`) and getting PDF lists (`/facturas/{factura_id}/pdf/`). Each endpoint has a dropdown menu next to it, showing its associated operation ID (e.g., `consumos_list`, `consumos_update`, etc.) and a lock icon indicating security status.

Ilustración 50: Swagger 1

This screenshot shows another view of the Swagger API Documentation interface, likely a different version or a subset of the endpoints. It includes sections for "consumos", "contact", "documentos", "facturas", "geo", and "incidencias". The endpoints listed are similar to those in Illustration 50, such as `/consumos/` for consumos and `/facturas/generate/payment/pdf/` for facturas. Each endpoint is shown with its method, URL, and associated operation ID and security status.

Ilustración 51: Swagger 2

8.1.2. Obtener el token de acceso:

- Usa el endpoint de login para obtener el token. Por ejemplo, 'POST /users/login/'.
 - Introduce los parámetros necesarios en el cuerpo de la solicitud, como 'username' y 'password'.
 - Haz clic en Execute.

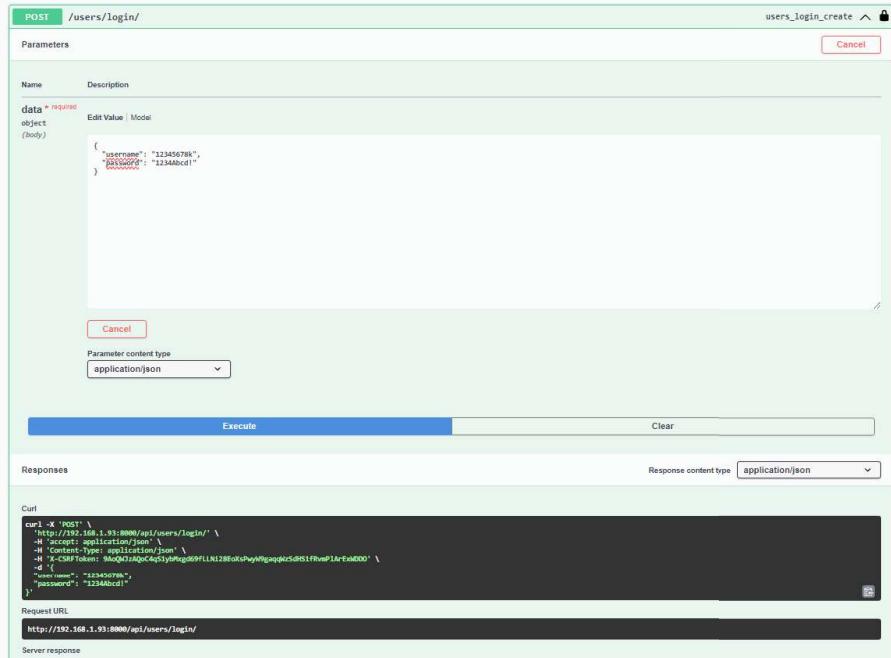


Ilustración 52: Obtener token de acceso

- La respuesta incluirá un 'access token' y un 'refresh token'. El 'access token' es lo que utilizarás para autenticarte.

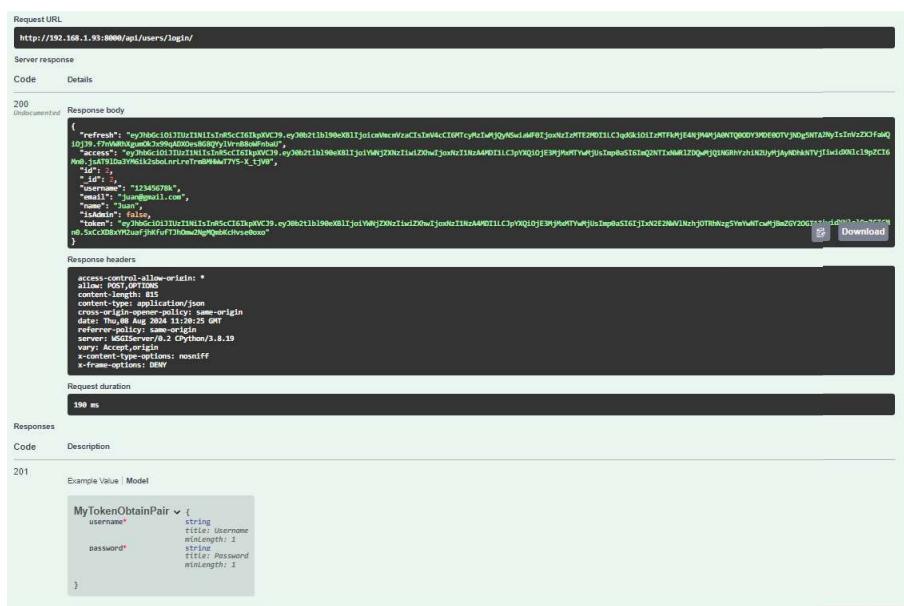


Ilustración 53: Datos con token de acceso

8.1.3. Autorización con el Token:

- Copia el access token de la respuesta del login.
- Vuelve a la ventana de Authorize en Swagger.
- Introduce el token en el formato Bearer token.
- Haz clic en Authorize y luego en Close.

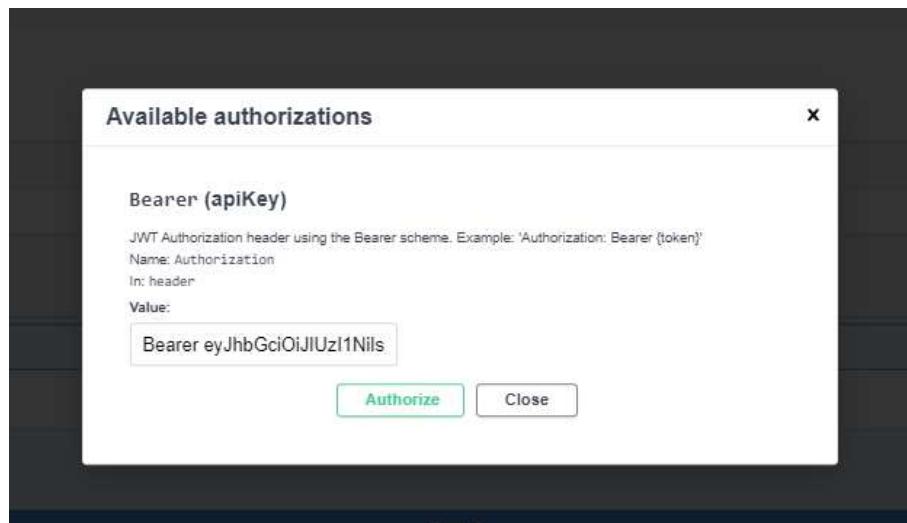


Ilustración 54: Autorización Token

8.1.4. Pruebas de Endpoints:

Durante las pruebas, se realizaron solicitudes a diversos endpoints de la API para verificar su funcionalidad y comportamiento. A continuación se detallan los resultados de algunas de las pruebas más significativas.

- Selecciona el endpoint que deseas probar nuevamente, por ejemplo, GET /community/all/.
- Haz clic en Execute para ejecutar la petición. Si todo está correcto, deberías ver la respuesta del servidor.

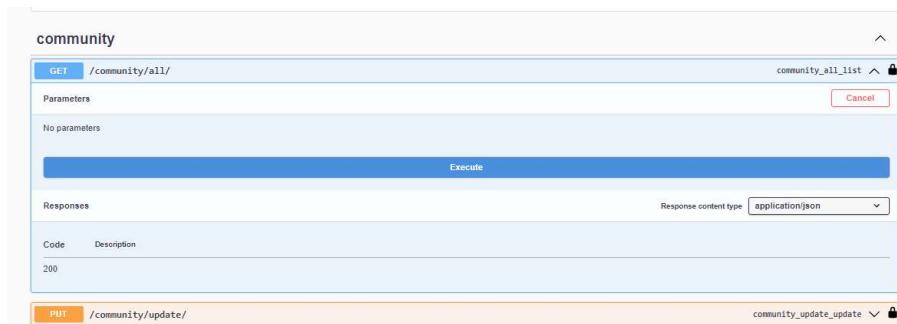


Ilustración 55: Petición 'GET /community/all/'

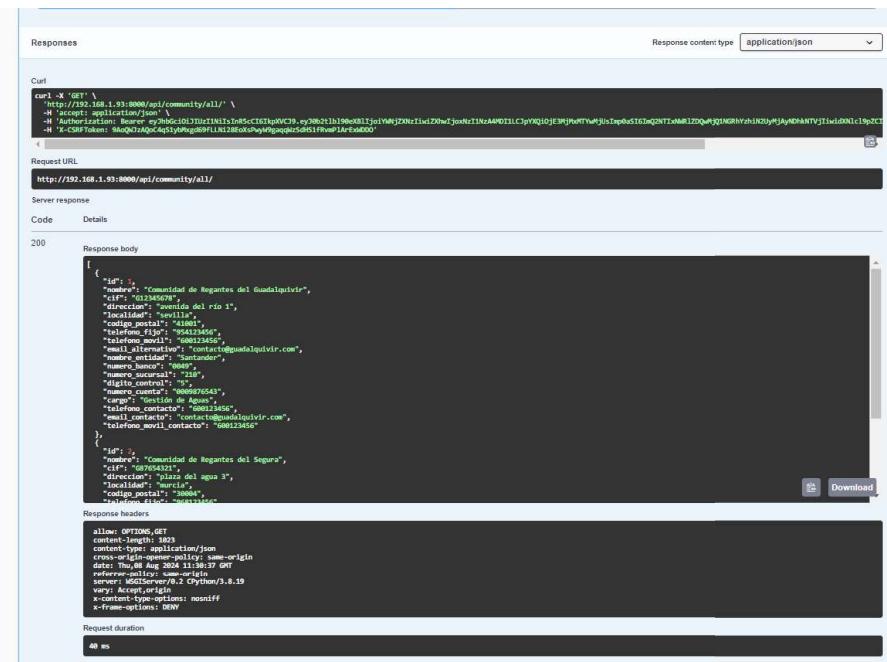


Ilustración 56: Resultado petición 'GET /community/all/'

8.1.5. Resultados y Observaciones:

Las pruebas realizadas con Swagger [17] han demostrado que los endpoints de la API funcionan correctamente, devolviendo los códigos de estado adecuados y los datos esperados. No se encontraron errores significativos durante las pruebas, y la API respondió de manera eficiente a todas las solicitudes.

Las pruebas realizadas a la API utilizando Swagger [17] confirmaron que la API cumple con los requisitos establecidos y maneja correctamente las diferentes operaciones de CRUD (Crear, Leer, Actualizar, Eliminar). Swagger [17] se demostró como una herramienta invaluable para la documentación y prueba de APIs, facilitando tanto el desarrollo como la verificación de la funcionalidad de la API. Cualquier problema detectado durante las pruebas fue abordado y solucionado, asegurando que la API esté lista para su uso en producción.

8.2. Postman:

Postman es una herramienta de desarrollo de API (Interfaz de Programación de Aplicaciones) que permite a los desarrolladores diseñar, probar, documentar y compartir APIs de manera rápida y sencilla. Es una aplicación popular entre desarrolladores porque simplifica el proceso de interacción con APIs, proporcionando una interfaz gráfica donde puedes enviar solicitudes HTTP y ver las respuestas del servidor sin necesidad de escribir código adicional.

Uso de Postman para Probar la API:

- **Configuración de Postman:**

Postman es una herramienta útil para probar las rutas de la API REST que hayas desarrollado en el backend.

- **Creación de una Nueva Solicitud:**

Abre Postman y crea una nueva solicitud (New Request). Selecciona el método HTTP apropiado (GET, POST, PUT, DELETE, etc.) y en la barra de dirección introduce la URL de la API que deseas probar, por ejemplo, <http://localhost:8000/api/ruta-deseada>.

- **Autenticación y Headers:**

Si tu API requiere autenticación, asegúrate de configurar los encabezados (Headers) y cualquier token de autenticación necesario en Postman.

- **Enviar Solicitud:**

Haz clic en Send para enviar la solicitud y ver la respuesta del servidor.

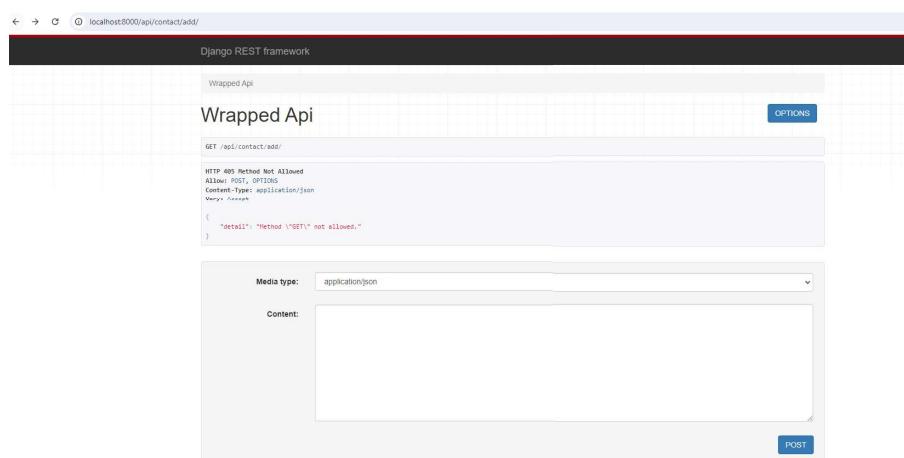


Ilustración 57: Postman

Bibliografía

- [1] J. de Andalucía, *Gestagua-cap*, Accessed: 2024-03, 2024. dirección: <https://www.juntadeandalucia.es/organismos/agriculturapescaaguaydesarrollorural/areas/infraestructurasagrarias/regadios/paginas/gestagua-cap2.html>.
- [2] Q. D. Team, *QGIS: Information and User Manuals*, Accessed: 2024-03, 2024. dirección: <https://qgis.org/es/site/>.
- [3] I. Meta Platforms, *React – A JavaScript library for building user interfaces*, Accessed: 2024-06, 2024. dirección: <https://react.dev/>.
- [4] D. S. Foundation, *Django Documentation*, Accessed: 2024-06, 2024. dirección: <https://docs.djangoproject.com/>.
- [5] W. W. W. C. (W3C), *HTML - HyperText Markup Language*, Accessed: 2024-06, 2024. dirección: <https://www.w3.org/html/>.
- [6] W. W. W. C. (W3C), *CSS - Cascading Style Sheets*, Accessed: 2024-06, 2024. dirección: <https://www.w3.org/Style/CSS/>.
- [7] P. D. Team, *PostGIS Documentation*, Accessed: 2024-06, 2024. dirección: <https://postgis.net/documentation/>.
- [8] P. S. Foundation, *Python Documentation*, Accessed: 2024-06, 2024. dirección: <https://www.python.org/doc/>.
- [9] P. G. D. Group, *PostgreSQL Documentation*, Accessed: 2024-06, 2024. dirección: <https://www.postgresql.org/docs/>.
- [10] Microsoft, *Visual Studio Code Documentation*, Accessed: 2024-06, 2024. dirección: <https://code.visualstudio.com/docs>.
- [11] M. D. N. (MDN), *JavaScript Guide*, Accessed: 2024-06, 2024. dirección: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide>.
- [12] Docker, *What is a Container? / Docker*, 2023. dirección: <https://www.docker.com/resources/what-container/>.
- [13] I. npm, *npm - Node Package Manager*, Accessed: 2024-06, 2024. dirección: <https://www.npmjs.com/>.
- [14] Yarnpkg.com, *Yarn - Package Manager*, Accessed: 2024-06, 2024. dirección: <https://yarnpkg.com/>.
- [15] Redux, *Redux - A Predictable State Container for JS Apps*, Accessed: 2024-06, 2024. dirección: <https://redux.js.org/>.
- [16] M. D. N. (MDN), *HTTP - Hypertext Transfer Protocol*, Accessed: 2024-06, 2024. dirección: <https://developer.mozilla.org/en-US/docs/Web/HTTP>.
- [17] Swagger, *Swagger Documentation*, Accessed: 2024-06, 2024. dirección: <https://swagger.io/docs/>.
- [18] Docker, *Docker Compose Documentation*, Accessed: 2024-06, 2024. dirección: <https://docs.docker.com/compose/>.

Apéndices