universidad de león

Escuela de Ingenierías

Industrial, Informática y Aeroespacial

GRADO EN INGENIERÍA INFORMÁTICA

Trabajo de Fin de Grado

Aplicación web para la gestión de módulos hidropónicos inteligentes

Web application for the management of intelligent hydroponic modules

Autor: José Antonio López Pérez
Tutores: Lidia Sánchez González,
Jesús Fernández Fernández

(Julio, 2025)

UNIVERSIDAD DE LEÓN
Escuela de Ingenierías Industrial, Informática y Aeroespacial

GRADO EN INGENIERÍA INFORMÁTICA
Trabajo de Fin de Grado

ALUMNO: José Antonio López Pérez
TUTOR: Lidia Sánchez González / Jesús Fernández Fernández
TÍTULO: Aplicación web para la gestión de módulos hidropónicos inteligentes.
TITLE: Web application for the management of intelligent hydroponic modules.
CONVOCATORIA: Julio, 2025

RESUMEN:

En un contexto de creciente interés por los sistemas de cultivo sostenible y la optimización de recursos, la hidroponía se presenta como una alternativa eficiente para la producción de alimentos en espacios reducidos y con un menor consumo de agua. Sin embargo, uno de los principales retos de estos sistemas es la necesidad de un monitoreo constante y una gestión precisa de variables como la temperatura, la humedad o los niveles de nutrientes. Automatizar y facilitar este control resulta fundamental para asegurar la viabilidad del cultivo y mejorar su rendimiento. Este Trabajo de Fin de Grado propone una solución innovadora: el desarrollo de una plataforma web de gestión inteligente para sistemas hidropónicos, combinando tecnología IoT, automatización y análisis de datos en tiempo real. Gracias a sensores distribuidos en el módulo de cultivo, los usuarios pueden visualizar y gestionar sus parámetros desde cualquier lugar, a través de una interfaz intuitiva que permite optimizar el rendimiento del sistema con decisiones basadas en datos. De este modo, es posible optimizar el mantenimiento del cultivo mediante la automatización de tareas como la ventilación o el ajuste de la iluminación. La plataforma está organizada en diferentes secciones que ofrecen funcionalidades como la visualización gráfica de los datos registrados, el estado actual del sistema, recomendaciones personalizadas para mejorar el cultivo y paneles de configuración para adaptar el funcionamiento del módulo a las necesidades concretas del usuario. Desde el punto de vista técnico, el sistema sigue una arquitectura basada en microservicios virtualizados mediante contenedores Docker, lo que garantiza su escalabilidad, portabilidad y facilidad de despliegue. Además, el proyecto incluye el diseño y la construcción de un módulo

hidropónico físico, equipado con una serie de sensores y componentes electrónicos gestionados a través de una estructura fabricada mediante impresión 3D. Gracias a esta combinación de hardware y software, se ofrece una solución integral para el control inteligente de cultivos hidropónicos.

ABSTRACT:
In a context of growing interest in sustainable cultivation systems and resource optimization, hydroponics is emerging as an efficient alternative for food production in small spaces and with lower water consumption. However, one of the main challenges facing these systems is the need for constant monitoring and precise management of variables such as temperature, humidity, and nutrient levels. Automating and facilitating this control is essential to ensure crop viability and improve yield. This Final Degree Project proposes an innovative solution: the development of a smart management web platform for hydroponic systems, combining IoT technology, automation, and real-time data analysis. Thanks to sensors distributed throughout the cultivation module, users can view and manage their parameters from anywhere, through an intuitive interface that allows them to optimize system performance with data-driven decisions. This way, crop maintenance can be optimized by automating tasks such as ventilation and lighting adjustment. The platform is organized into different sections that offer functionalities such as graphical visualization of recorded data, the current system status, personalized recommendations for improving cultivation, and configuration panels to adapt the module's operation to the user's specific needs. From a technical perspective, the system follows an architecture based on virtualized microservices using Docker containers, ensuring scalability, portability, and ease of deployment. The project also includes the design and construction of a physical hydroponic module, equipped with a series of sensors and electronic components managed through a 3D-printed structure. This combination of hardware and software offers a comprehensive solution for the intelligent control of hydroponic crops.

Palabras clave: Hidroponía, Automatización, Monitoreo, Sensores, Aplicación web, Microservicios, Docker, Impresión 3D, Optimización de recursos, Control inteligente

Firma del alumno:	**VºBº Tutor/es:**

Índice

Índice de tablas

Índice de imágenes

Glosario de términos

API: Conjunto de definiciones y protocolos que permiten que dos aplicaciones se comuniquen entre sí.

Arduino: Plataforma de hardware libre basada en una placa con microcontrolador y un entorno de desarrollo para crear proyectos electrónicos.

Backend: Parte de una aplicación que gestiona la lógica, la base de datos y el procesamiento de datos en el servidor.

CSS: Lenguaje de hojas de estilo usado para describir la presentación y diseño de documentos HTML.

DaisyUI: Plugin de Tailwind CSS que proporciona componentes predefinidos y estilos personalizables para interfaces web.

Docker: Plataforma que permite crear, distribuir y ejecutar aplicaciones dentro de contenedores ligeros y portables.

Endpoint: Punto de acceso específico dentro de una API donde se pueden realizar operaciones o solicitudes.

Hardware: Conjunto de componentes físicos y tangibles de un sistema informático o electrónico.

IoT: Internet de las Cosas, se refiere a la red de objetos físicos (dispositivos y sistemas) que se conectan a Internet y se comunican entre sí.

Javascript: Lenguaje de programación interpretado, orientado a objetos, usado principalmente para desarrollar funcionalidades en páginas web.

MariaDB: Sistema de gestión de bases de datos relacional, derivado de MySQL, utilizado para almacenar y gestionar datos.

Microservicios: Arquitectura de software que estructura una aplicación como un conjunto de servicios pequeños e independientes que se comunican entre sí.

NFT cultivo: Técnica de cultivo hidropónico llamada "Nutrient Film Technique", en la que las raíces de las plantas crecen en una fina película de solución nutritiva.

PHP: Lenguaje de programación de propósito general ampliamente usado para el desarrollo web del lado del servidor.

React: Framework utilizado para la creación de aplicaciones web.

SSH: Protocolo de red que permite el acceso remoto seguro a través de una red no segura.

Software: Conjunto de programas, instrucciones y reglas informáticas que permiten ejecutar tareas en un dispositivo.

Swagger: Especificación y un conjunto de herramientas de código abierto utilizadas para documentar, diseñar y probar APIs RESTful.

Introducción

Este proyecto plantea una solución enfocada en el desarrollo de un producto software completo, destinado a la gestión y automatización de módulos hidropónicos inteligentes. El objetivo principal es permitir a los usuarios visualizar la información recogida por los sensores instalados en el sistema y facilitar la programación de acciones automáticas que optimicen el mantenimiento del cultivo. A través del procesamiento y presentación adecuada de los datos capturados (como temperatura, humedad o electro-conductividad del agua), se busca garantizar un control preciso que permita mejorar el rendimiento y la viabilidad de los cultivos hidropónicos.

Planteamiento del problema

Con el desarrollo del proyecto, se pretende analizar y aplicar metodologías ágiles durante la gestión del proceso, empleando un enfoque iterativo e incremental. Gracias a este modelo, el producto podrá ir incorporando nuevas funcionalidades de manera progresiva a lo largo de diferentes iteraciones, permitiendo que sea operativo desde las primeras fases y favoreciendo la obtención de retroalimentación por parte de los usuarios. Esto facilita la detección temprana de errores y reduce los costes asociados a posibles cambios, además de asegurar una evolución continua del sistema a medida que se desarrollan nuevas versiones.

En cuanto a la arquitectura, se ha optado por un diseño basado en microservicios virtualizados mediante contenedores Docker, lo que permite dividir el sistema en diferentes componentes independientes, cada uno encargado de una funcionalidad específica, garantizando flexibilidad, escalabilidad y facilidad de mantenimiento. La comunicación entre estos microservicios es fundamental y se realiza a través de peticiones bien definidas que permiten integrar de manera eficiente todas las partes del sistema.

El producto final integra tanto el desarrollo de la aplicación web, que proporciona al usuario acceso a los datos y configuración del módulo hidropónico, como el diseño físico del propio módulo, fabricado mediante impresión 3D y equipado con sensores y componentes electrónicos. Gracias a esta combinación de hardware y software, se ofrece una solución completa que facilita el control inteligente y automatizado de cultivos hidropónicos, proporcionando una alternativa eficiente y sostenible para la producción de alimentos.

Objetivos

El objetivo principal del proyecto es desarrollar una aplicación web que permita la gestión y automatización de módulos hidropónicos inteligentes, facilitando a los usuarios la monitorización de datos en tiempo real y la programación de acciones automáticas para optimizar el mantenimiento del cultivo.

Además del objetivo principal, hay una serie de objetivos específicos:

- Comprender qué es un módulo hidropónico inteligente y cuáles son sus principales componentes.
- Comprender la arquitectura basada en microservicios y su aplicación en sistemas de gestión de datos IoT.
- Conocer los procedimientos y métodos para desarrollar una arquitectura basada en microservicios mediante Docker.
- Comprender el funcionamiento y la integración de sensores en sistemas de monitoreo ambiental (temperatura, humedad, nutrientes).
- Conocer los medios de comunicación entre los microservicios del sistema y cómo garantizar su correcta interacción.
- Evaluar la experiencia del usuario en la aplicación web, analizando la usabilidad y accesibilidad de las funcionalidades ofrecidas.
- Integrar hardware y software de forma eficiente para ofrecer una solución completa de gestión de cultivos hidropónicos.
- Conocer las ventajas que aporta el uso de un bus de datos a la hora de evitar pérdidas de información.
- Desarrollar un módulo hidropónico diseñado específicamente para la aplicación.
- Analizar los datos obtenidos y realizar recomendaciones para el usuario.

Metodología

Para el desarrollo de este proyecto se ha seguido una metodología ágil e incremental, gestionando el ciclo de vida del software a través de la plataforma GitHub como herramienta principal para el control de versiones y seguimiento del progreso. El trabajo se ha dividido en dos grandes etapas de aproximadamente tres meses cada una.

Durante la primera etapa, el objetivo principal fue obtener una primera versión del prototipo funcional, enfocada en desarrollar un Producto Mínimo Viable (MVP) que incluyera las características básicas

necesarias para validar el funcionamiento del sistema. En esta fase inicial se priorizó la integración de los componentes esenciales del sistema, como la captura de datos desde sensores, la visualización básica de información en la aplicación web y la ejecución de acciones automáticas simples.

La planificación del desarrollo se organizó en iteraciones de dos semanas, al final de las cuales se incorporaba al proyecto al menos una nueva funcionalidad que permitiera avanzar de forma continua y medible. Además, esta estructura permitió realizar pruebas parciales y corregir posibles errores antes de continuar con nuevas implementaciones.

Una vez validado el prototipo inicial, se llevó a cabo una segunda etapa, también de tres meses, centrada en el perfeccionamiento del sistema y el desarrollo de la versión final. En esta fase se añadieron mejoras tanto a nivel funcional como visual, incluyendo la optimización de la interfaz de usuario, la ampliación de las opciones de configuración y la integración completa de todas las funcionalidades planificadas para el producto final.

El uso de esta metodología ha permitido mantener un flujo de trabajo flexible y adaptativo, facilitando la incorporación de mejoras continuas y la resolución temprana de incidencias, lo que ha contribuido a obtener un sistema estable y operativo al finalizar cada ciclo de desarrollo.

Estructura del trabajo

Este trabajo se organiza en varios apartados que permiten guiar al lector a través de todas las fases del desarrollo del proyecto, desde su concepción hasta su evaluación final:

Capítulo 1: Análisis del problema

En esta sección se contextualiza la problemática que da origen al proyecto, detallando las necesidades detectadas y los objetivos que se persiguen. También se analizan las alternativas existentes y se justifican las tecnologías seleccionadas para abordar la solución.

Capítulo 2: Planificación y gestión del proyecto

Aquí se describe cómo se ha organizado el desarrollo del sistema, incluyendo la definición del alcance del trabajo, la distribución y calendarización de tareas, así como los recursos empleados a lo largo del proceso. Se detallan las estrategias seguidas para garantizar una gestión eficiente y controlada del proyecto.

Capítulo 3: Desarrollo de la solución

En este apartado se profundiza en la solución implementada, explicando de manera detallada el diseño del sistema, sus principales componentes y la arquitectura adoptada. Se justifica cada decisión técnica tomada y se describe cómo estas decisiones permiten cumplir con los objetivos planteados.

Capítulo 4: Validación y resultados

Esta sección recoge el proceso llevado a cabo para verificar el correcto funcionamiento del sistema, así como las pruebas realizadas. Además, se presentan los resultados obtenidos y se analiza hasta qué punto se han alcanzado los objetivos iniciales.

Capítulo 5: Conclusiones y futuras mejoras

Finalmente, se realiza una reflexión global sobre el trabajo desarrollado. Se exponen los principales retos y dificultades encontrados durante el proyecto, se valoran los resultados alcanzados y se proponen posibles líneas de mejora y evolución futura del sistema.

Anexo 1: Registro del desarrollo

Este anexo recoge de forma cronológica la evolución del proyecto, mostrando cómo ha progresado a lo largo del tiempo y los hitos más relevantes alcanzados.

Anexo 2: Guía de usuario

Incluye instrucciones claras y detalladas para facilitar al usuario la utilización de la aplicación, desde su puesta en marcha hasta el manejo de sus principales funcionalidades.

Anexo 3: Proyecto Ansible

Incluye un proyecto utilizado para la configuración automática del entorno de producción del proyecto. Su desarrollo permite el despliegue continuo en el entorno de despliegue, simplificando tareas complementarias al desarrollo.

Capítulo 1

Estudio del problema

En esta sección se realiza el estudio del contexto del problema, su estado en la actualidad y se define sus aspectos clave.

1.1 El contexto del problema

El Trabajo de Fin de Grado tiene como objetivo el desarrollo de GreenhouseIOT, una aplicación web orientada a la gestión y supervisión de datos generados por un módulo hidropónico. La idea principal surge como una oportunidad para consolidar conocimientos en tecnologías actuales del ámbito del desarrollo web, además de obtener experiencia práctica en la creación de soluciones reales con proyección profesional.

El proyecto propone una herramienta capaz de monitorizar y almacenar los datos obtenidos a partir de sensores instalados en sistemas hidropónicos, permitiendo así un control preciso de variables críticas como temperatura, humedad o nivel de nutrientes. Todo ello está pensado para facilitar el seguimiento de estos cultivos en tiempo real desde una plataforma web intuitiva y accesible.

Gracias al uso de tecnologías modernas como React para el frontend, FastAPI para la lógica del backend, MariaDB como sistema de almacenamiento, junto con Kafka para el envío de datos y Docker para la contenerización de servicios, la aplicación busca ofrecer una solución completa para el seguimiento local de instalaciones hidropónicas.

1.2 El estado del arte

El auge de la agricultura inteligente y, en particular, de los sistemas hidropónicos, ha impulsado el desarrollo de soluciones tecnológicas orientadas al monitoreo y gestión de cultivos mediante sensores IoT. A nivel industrial, existen plataformas robustas que ofrecen servicios avanzados de análisis de datos, control remoto y optimización de recursos, aunque muchas de estas soluciones están orientadas a explotaciones a gran escala y suponen altos costes de implementación.

En cambio, GreenhouseIOT surge como una alternativa pensada para instalaciones locales o de pequeña escala, proporcionando una plataforma sencilla, adaptable y sin grandes requerimientos

técnicos para el usuario final. Mientras que otras herramientas del mercado priorizan la integración de sistemas complejos y centralizados, este proyecto apuesta por una arquitectura distribuida mediante microservicios, facilitando así su mantenimiento, escalabilidad y despliegue en diferentes entornos.

Para que una solución tecnológica de este tipo sea realmente útil en un sistema hidropónico, es fundamental entender las bases agronómicas del proceso. La hidroponía, en esencia, consiste en cultivar plantas sin necesidad de suelo, utilizando en su lugar soluciones nutritivas disueltas en agua. Esto permite un control más preciso sobre los factores que influyen en el desarrollo de las plantas, como la conductividad eléctrica (EC), la oxigenación del agua y la disponibilidad de nutrientes.
Comprender estos principios es vital para interpretar correctamente los datos recogidos por los sensores y realizar ajustes eficaces en tiempo real. Por ejemplo, en sistemas que emplean agua de ósmosis inversa —una opción común debido a su pureza y neutralidad química—, se requiere la adición controlada de sales minerales específicas para hidroponía. Estas sales aportan los macro y micronutrientes esenciales, como nitrógeno (N), fósforo (P), potasio (K), calcio (Ca), magnesio (Mg), hierro (Fe), entre otros.
El monitoreo de estos nutrientes no se realiza midiendo cada elemento por separado en el sistema, sino que se utilizan indicadores indirectos como la conductividad eléctrica (EC) para estimar la concentración total de sales disueltas en la solución nutritiva. Un nivel de EC demasiado bajo indicaría una carencia de nutrientes, mientras que uno excesivamente alto podría generar toxicidad o desequilibrios osmóticos en las raíces. Por ello, el sistema debe ser capaz de registrar y visualizar estos valores de forma accesible y comprensible para el usuario, permitiendo acciones correctivas rápidas y eficientes.
Algunos ejemplos de servicios similares disponibles en el mercado son TerraceLab[1] y Green in Blue[2].
TerraceLab es una startup española que ofrece módulos de cultivo vertical hidropónicos diseñados para ser instalados en entornos urbanos como escuelas, oficinas, comunidades de vecinos o terrazas particulares. Su enfoque se basa en el modelo Farming as a Service, proporcionando tanto el equipamiento como el mantenimiento, junto con una plataforma digital que permite monitorizar parámetros como temperatura, humedad, pH y conductividad eléctrica mediante sensores IoT conectados a la nube. Está orientado principalmente a usuarios no técnicos interesados en introducir prácticas de agricultura sostenible en espacios reducidos, con un fuerte componente educativo, social y de concienciación ambiental.

Green in Blue es una startup catalana (Vilassar de Mar) diseña sistemas de acuaponía con impacto cero, combinando cultivo de plantas y cría de peces. Desarrolla soluciones modulares e IoT-ready, enfocándose en eficiencia hídrica, economía circular y sostenibilidad urbana.

Por su parte, GreenhouseIOT se posiciona como una solución para usuarios domésticos y pequeños productores que busquen una herramienta autónoma, basada en microservicios, contenedorizada, enfocada en hidroponía y con una interfaz web intuitiva. No presenta una suscripción para su uso ni costes elevados del hardware ya que la idea del proyecto es permitir a los usuarios introducirse en el sector de la forma más económica posible, sin renunciar a las ventajas de monitorización por sensores. Los usuarios que lo deseen podrán seguir instrucciones de instalación para la creación de sus propios módulos.

Trabajos académicos similares

Previo al desarrollo, se han tenido en cuenta trabajos académicos similares o relacionados con el sector con el fin de contemplar los retos encontrados y las soluciones aportadas en ellos.

Algunos ejemplos de trabajos han sido los siguientes:

Diseño de una arquitectura IoT para el control de sistemas hidropónicos [3]

En este proyecto se realiza la construcción de un sistema hidropónico basado en la técnica de cultivo NFT. Además, se realizó un servicio web muy simple haciendo uso de PHP y JavaScript.

GreenhouseIOT no usa la técnica NFT de cultivo pero gracias a este trabajo se han tomado en consideración varios elementos utilizados como el DHT11, el sensor de temperatura DS18B20 o el uso de relés para el control de luces.

Sistema IoT para telemetría y riego automático en cultivos de sustrato [4]

En este proyecto se ha desarrollado un sistema iot basado en el cultivo en sustrato. Gracias a los sensores instalados se ha podido analizar y automatizar el riego de los cultivos y se han enviado los datos a la nube con el fin de su visualización. Además el proyecto incluye una demostración de un mes de funcionamiento con resultados muy positivos.

Diseño y construcción de un sistema hidropónico con IoT adaptable a acuaponía [5]

En este proyecto se consiguió desarrollar un sistema NFT adaptable a acuaponía. Además, se realizó un estudio de las variables implicadas, se implementó el módulo iot de tamaño medio y se desarrolló una aplicación para la visualización de datos en dispositivos móviles.

1.3 La definición del problema

En base al análisis previo, el problema principal que aborda este proyecto es la falta de soluciones distribuidas, accesibles y adaptadas a pequeñas instalaciones hidropónicas que permitan gestionar y visualizar de manera eficiente los datos recogidos por sensores IoT.

Con GreenhouseIOT se pretende cubrir las siguientes necesidades:

- Monitorización continua de parámetros ambientales relevantes (temperatura, humedad, nivel de agua, etc.).
- Almacenamiento ordenado y seguro de los datos generados por los sensores.
- Visualización clara e intuitiva de los datos históricos y en tiempo real desde una interfaz web.
- Comunicación eficiente entre los diferentes módulos del sistema.
- Facilitar la escalabilidad del sistema mediante una arquitectura basada en contenedores (Docker) y microservicios.

Además, uno de los objetivos clave de la aplicación es ofrecer una experiencia de usuario sencilla, que permita tanto la instalación como la gestión del sistema sin necesidad de conocimientos avanzados en tecnología o programación. A partir de una plataforma web accesible desde cualquier dispositivo, el usuario puede supervisar su módulo hidropónico, realizar consultas de los registros históricos y recibir recomendaciones basadas en los datos almacenados.

Con todo ello, GreenhouseIOT pretende ser una herramienta útil y práctica para impulsar el uso de la agricultura inteligente en proyectos locales y domésticos.

Capítulo 2

Gestión de proyecto software

2.1 Alcance del proyecto

En este apartado se describen las tareas necesarias para llevar a cabo el desarrollo de GreenhouseIOT, así como los recursos requeridos, la estimación de costes y los posibles riesgos asociados. La finalidad del proyecto es crear una aplicación web que permita monitorizar y gestionar los datos de sensores instalados en un módulo hidropónico, facilitando su visualización, análisis y almacenamiento de manera sencilla y eficiente para el usuario final.
La aplicación busca ofrecer una herramienta de apoyo a pequeñas instalaciones hidropónicas, proporcionando información en tiempo real y registros históricos que permitan optimizar el cultivo y anticipar posibles problemas en el entorno controlado.

2.1.1 Definición del proyecto

El proyecto consiste en el desarrollo de una plataforma web que permita a los usuarios supervisar y gestionar los datos de su módulo hidropónico. La aplicación incluirá funciones como:

- Visualización en tiempo real de datos ambientales (temperatura, humedad, nivel de nutrientes, etc.).
- Consulta de registros históricos y gráficas comparativas.
- Configuraciones para automatizar procesos.
- Gestión del almacenamiento de datos mediante una base de datos relacional.
- Panel de administración básico para supervisar el sistema y verificar el estado de los distintos módulos.

La arquitectura estará dividida en 3 capas: origen de datos, persistencia y explotación, desarrolladas de manera paralela y coordinada, asegurando una integración continua y modular para facilitar su mantenimiento y escalabilidad.
La tecnología elegida para la capa de explotación es React [6], que permitirá diseñar una interfaz interactiva y dinámica para el usuario, en la capa de persistencia encontramos la base de datos

MariaDB [7] y un contenedor desarrollado con FastApi [8] que permite interactuar con ella. Por último, en la capa de origen de datos encontramos un servicio de Python que se encarga de preguntar a los módulos hidropónicos los datos de forma periódica y prepara esos datos para ser transportados por Kafka [9] hacia la capa de persistencia.

El uso de un bus de datos como Kafka resulta fundamental para garantizar la fiabilidad del sistema, ya que permite desacoplar los componentes productores y consumidores de datos, ofreciendo tolerancia a fallos, capacidad de reintentos automáticos y almacenamiento temporal. De esta forma, se evitan pérdidas de información ante problemas como interferencias en la conexión Wi-Fi o caídas puntuales en los servicios de destino.

Además, todo el entorno se desplegará utilizando contenedores Docker [10], asegurando así la portabilidad y estabilidad de la aplicación, independientemente del entorno de ejecución.

El sistema está diseñado para su funcionamiento a nivel local, pensado para pequeñas instalaciones hidropónicas que requieran una solución asequible, funcional y sencilla de implementar.

De forma adicional, se contempla en un anexo [Anexo 3: Proyecto Ansible] un proyecto basado en la tecnología Ansible [11] dedicado a la configuración automática de un servidor VPS Ubuntu para el despliegue de los microservicios.

2.1.2 Estimación de tareas y recursos

Tareas

El desarrollo del proyecto se organizará en tareas semanales, dentro de una metodología incremental que permitirá ir incorporando nuevas funcionalidades progresivamente. A lo largo del desarrollo se realizarán reuniones periódicas para revisar los avances, valorar posibles mejoras y ajustar las prioridades.

Entre las tareas principales previstas se encuentran:

- Diseño de la arquitectura de la aplicación y configuración inicial del entorno.
- Desarrollo del backend con FastAPI.
- Desarrollo del frontend con React.
- Integración de la base de datos MariaDB.
- Configuración y manejo de envió de datos mediante Kafka.
- Creación de pruebas funcionales para asegurar el correcto comportamiento de cada módulo.
- Despliegue de los servicios con Docker.
- Elaboración de documentación técnica y manual de usuario.
- Pruebas finales y validación del sistema.

Cada entrega de una nueva funcionalidad tendrá una duración estimada mínima de dos semanas, dependiendo de la complejidad y los recursos disponibles.

Recursos

Para el desarrollo de GreenhouseIOT se contemplan tanto recursos humanos como físicos:

Humanos

El equipo ideal del proyecto estaría compuesto por cuatro perfiles clave, cada uno enfocado en áreas específicas del desarrollo:

- Jefe de proyecto: encargado de planificar, supervisar y coordinar todas las fases del desarrollo.
- Desarrollador backend: encargado de la implementación de la lógica de negocio, API y conexiones con la base de datos. Se encargaría del mantenimiento y configuración de la base de datos y de las tareas relacionadas con Docker. Desarrollaría prácticas de ci/cd para asegurar el correcto funcionamiento y entrega de las actualizaciones del sistema.
- Desarrollador frontend: responsable del desarrollo de la parte visual e interactiva de la plataforma. Se encargaría de conseguir una interfaz amigable para el usuario haciendo uso de librerías de componentes UI.
- Encargado hardware: responsable del diseño de circuitos electrónicos, diseño de los componentes 3D, procesos de impresión 3D, montaje de módulos hidropónicos, desarrollo del código de microcontroladores esp32 y configuraciones derivadas.

Físicos

Los recursos físicos necesarios para llevar a cabo el proyecto son:

- Hardware: equipos informáticos para cada miembro del equipo, capaces de ejecutar entornos de desarrollo y contenedores de Docker de manera eficiente. El coste de la impresión 3D y materiales asociados a componentes del módulo.
- Software: no se prevén costes adicionales, ya que todas las tecnologías utilizadas son open source (React, FastAPI, MariaDB, Kafka y Docker). Según el despliegue objetivo, puede ser necesario incluir un sobrecoste asociado a un servidor dedicado de despliegue.

2.1.3 Presupuesto

El presupuesto para el desarrollo de GreenhouseIoT se ha estimado considerando los costes asociados tanto a los recursos humanos como a los recursos físicos e infraestructurales necesarios para garantizar el correcto desarrollo del proyecto.

Recursos Humanos

El equipo de trabajo estará formado por los siguientes perfiles profesionales, junto con su coste mensual aproximado y la duración estimada del proyecto (3 meses):

Puesto	Coste por hora(€/h)	Horas de trabajo	Coste total (€)
Jefe de proyecto	30	216	6.480
Backend developer	25	352	8.800
Frontend developer	25	248	6.200
Encargado hardware	25	280	7.000

Figura 2. 1. Coste estimado de recursos humanos

Total, RRHH 28.480€

Los costes por hora se han estimado en función de perfiles con una experiencia intermedia, comprendida entre 2 y 5 años. Este nivel de experiencia garantiza que los profesionales puedan desarrollar sus funciones con autonomía, aplicando buenas prácticas y conocimientos sólidos, sin alcanzar aún los costes asociados a perfiles sénior. La elección de estos perfiles busca un equilibrio entre calidad técnica y viabilidad económica para un proyecto de duración limitada como el planteado.

Nota: Se ha estimado un equipo reducido enfocado en cubrir las áreas críticas del proyecto, prescindiendo de perfiles secundarios para optimizar los costes. Otra opción más económica sería contratar a un desarrollador fullstack y alargar ligeramente la duración del desarrollo.

Recursos Físicos

Hardware:

Aquí se observa el listado de materiales necesarios para la realización de 1 módulo.

Nombre del recurso	Trabajo	Costo
Impresora 3D	1 und	319,00 €
Plástico PLA	3 bobina 850g	63,00 €
portátiles	4 und	1.716,00 €
Cableado	2 metros	22,98 €
DHT22	1 und	4,63 €
Sensor temperatura del agua	1 und	1,39 €
Sensor TDS	1 und	3,49 €
Fuente de alimentación	1 und	7,24 €
Regulador de voltaje	1 und	2,97 €
Foco led	1 und	5,79 €
Tornillería	1 caja	4,79 €
LDR	1 und	1,42 €
Resistencias	1 caja	2,92 €
Modulo 4x Relé	1 und	2,42 €
Conexiones	1 caja	9,80 €
ESP32 S3	1 und	6,03 €
ESP32 Cam	1 und	7,49 €

Contenedores	1 und	4,50 €
Oxigenador	1 und	2,49 €
Ventilador	1 und	1,54 €
Pegamento	1 caja	3,50 €
Tuercas inyección	1 caja	3,99 €
pcb	1 caja	2,84 €
lente 77mm	1 und	6,36 €
Nivel de agua	1 und	2,79 €
Soldador	1 kit	21,00 €
	Coste Total	2.230.37€

Figura 2. 2. Listado de materiales

En el presupuesto anterior vienen incluidos costes materiales asociados a la compra de equipos de montaje o producción. A continuación, se muestra el presupuesto mínimo que requiere la construcción de un segundo módulo, no se incluyen costes como la compra de la impresora 3D o de nueva tornillería ya que el gasto viene por caja y no siempre es necesario comprar más unidades.

Nombre del recurso	Trabajo	Costo
Plástico PLA	3 bobina 850g	63,00 €
Cableado	2 metros	22,98 €
DHT22	1 und	4,63 €
Sensor temperatura del agua	1 und	1,39 €
Sensor TDS	1 und	3,49 €
Fuente de alimentación	1 und	7,24 €
Regulador de voltaje	1 und	2,97 €
Foco led	1 und	5,79 €
LDR	1 und	1,42 €
Modulo 4x Relé	1 und	2,42 €
ESP32 S3	1 und	6,03 €
ESP32 Cam	1 und	7,49 €
Contenedores	1 und	4,50 €
Oxigenador	1 und	2,49 €
Ventilador	1 und	1,54 €
lente 77mm	1 und	6,36 €
Nivel de agua	1 und	2,79 €
	Coste total	146,53 €

Figura 2. 3. Listado de materiales por módulo

Software: No se contemplan costes adicionales debido al uso exclusivo de herramientas y tecnologías open source como FastAPI, React, MariaDB, Kafka y Docker.

Costes Indirectos

Oficina: Se optará por el trabajo remoto por lo que no se aplican costes debidos a oficinas físicas.

Para el desarrollo y testeo de la aplicación será necesaria la construcción del por lo menos 1 módulo hidropónico inteligente.

En este caso se considerará que se construye solo el primer módulo ya que tras el desarrollo el cliente será responsable y propietario de la compra y mantenimiento del módulo físico.

Presupuesto Total

Concepto	Coste (€)
Recursos Humanos	28.480 €
Hardware	2.230.37€
Total	**30.710,37€**

Figura 2. 4. Presupuesto total

Este presupuesto está sujeto a posibles ajustes según la evolución del proyecto, cambios en los plazos o incorporación de nuevos requerimientos.

Coste total

Concepto	Coste (€)
Recursos Humanos Hardware	28.480 € 2.230 €
Costes indirectos (15 %)	4.607 €
Beneficio industrial (15 %)	5.298 €
Impuestos (I.V.A. 21 %)	8.529 €
Total del proyecto	**49.144 €**

Figura 2. 5. Costes con impuestos

2.2 Plan de trabajo

En esta sección se detalla cómo se han establecido las diferentes tareas del desarrollo de la aplicación GreenhouseIoT, así como el tiempo aproximado de dedicación a cada una. El plan de trabajo sigue una metodología incremental, permitiendo validar cada funcionalidad antes de avanzar a la siguiente, asegurando un desarrollo controlado y eficiente.

2.2.1 Identificación de tareas

Tareas de investigación del sector

a) Investigación sobre el sector

Se llevó a cabo un estudio exploratorio para entender el contexto actual de la agricultura inteligente y el uso de módulos hidropónicos en pequeñas instalaciones. Este análisis permitió definir con precisión el alcance del sistema GreenhouseIOT.

b) Investigación sobre las tecnologías

Se investigaron herramientas modernas como React, FastAPI, Docker o Kafka, seleccionando las más adecuadas para la arquitectura distribuida del sistema. Esta fase resultó fundamental para garantizar una integración tecnológica eficiente.

c) Planificación de objetivos esperados

Se definieron los objetivos principales y secundarios del proyecto, delimitando funcionalidades clave a implementar. Esta etapa permitió estructurar las tareas de desarrollo con metas alcanzables en cada iteración.

d) Estudio de viabilidad del proyecto

Se evaluaron aspectos técnicos, económicos y de tiempo para confirmar que el proyecto era ejecutable en el plazo y con los recursos disponibles. Se ajustaron prioridades para optimizar el proceso de desarrollo.

e) Creación de la identidad del producto

Se estableció la imagen de la plataforma, incluyendo nombre, paleta de colores y estilo visual. Esta identidad facilitará su futura expansión y diferenciación frente a soluciones similares.

f) Diseño del primer boceto relativo a la interfaz de la aplicación y estructura

Se elaboraron prototipos iniciales de la interfaz de usuario, priorizando la claridad, accesibilidad y facilidad de uso. Este diseño sirvió de base para el posterior desarrollo del frontend.

g) Planificación de metodología de desarrollo.

Se definió una estrategia basada en metodologías ágiles, con entregas iterativas, control de versiones mediante GitHub y pruebas continuas. Esta elección permitió adaptar el desarrollo a los cambios surgidos durante el proyecto.

Tareas de desarrollo

Nombre de tarea
GreenhouseIOT
Planificación inicial
a) Definición de la idea y requisitos
b) Diseño de la arquitectura base
c) Configuraciones del repositorio de código
d) Creación de la arquitectura del proyecto
Creación del módulo
e) Diseño del circuito eléctrico
f) Montaje del circuito
g) Diseño de la carcasa 3D

h) Impresión de la carcasa
i) Montaje del módulo
j) Programación del módulo
Origen de datos
k) Creación de la API de origen de datos
l) Programación del productor Kafka
Persistencia de datos
m) Diseño de la base de datos
n) Creación de la base de datos
o) Creación de api para la base de datos
p) Programación del consumidor Kafka
Explotación de datos
q) Boceto diseño gráfico de la aplicación web
r) Desarrollo estructura básica de la web
s) Visualización de datos de ultima lectura
t) Visualización de datos haciendo uso de graficas
u) Paneles de configuración para automatizaciones
v) Lógica de gestión de múltiples módulos
Sistema de recomendaciones personalizadas
w) Pagina dedicada para imágenes de los módulos
x) Desarrollo de la sección de peticiones y recomendaciones
y) Sistema de envío de petición personalizada
z) Visualización de recomendación personalizada para el usuario
Control de calidad
aa) Creación test automáticos para Apis
bb) Creación test automáticos para la web

Figura 2. 6. Tareas de desarrollo

Planificación inicial

a) Definición de la idea y requisitos

Se establecieron los objetivos funcionales y técnicos de la plataforma, así como los requisitos mínimos que debía cumplir cada uno de los módulos. Esta fase permitió fijar las expectativas del proyecto y orientar el diseño del sistema hacia la resolución de necesidades reales en el control de cultivos hidropónicos.

b) Diseño de la arquitectura base

Se optó por una arquitectura de microservicios desplegada con Docker, separando las capas de origen de datos, persistencia y explotación. Este enfoque garantiza la escalabilidad y permite mantener cada componente de forma independiente, favoreciendo el mantenimiento y evolución del sistema.

c) Configuraciones del repositorio de código

Se configuró un repositorio Git para almacenar y gestionar el código fuente, estableciendo ramas de desarrollo, revisión y producción. También se añadieron flujos básicos de integración continua para automatizar pruebas y asegurar la calidad del software a lo largo del proyecto.

d) Creación de la arquitectura del proyecto

Se definieron las rutas internas de comunicación entre los servicios y se organizaron las carpetas y módulos del código según buenas prácticas. Esto facilitó el trabajo colaborativo y la modularización del sistema, permitiendo realizar cambios sin afectar otras partes del código.

Creación del módulo

e) Diseño del circuito eléctrico

Se diseñó el esquema electrónico del módulo hidropónico, determinando los sensores y actuadores necesarios para medir variables clave como temperatura, humedad, TDS y nivel de agua. Este diseño sirvió de referencia para el montaje físico y programación posterior del sistema.

f) Montaje del circuito

Tras definir el esquema eléctrico, se procedió a la integración física de los componentes en una placa base. Se soldaron y conectaron sensores, relés y microcontroladores, verificando la correcta comunicación entre todos los elementos del sistema.

g) Diseño de la carcasa 3D

Se creó un diseño 3D personalizado que alojara todos los componentes del sistema de forma compacta y funcional. Se emplearon herramientas de modelado para asegurar el acoplamiento entre piezas, facilitando tanto el montaje como el mantenimiento futuro del hardware.

h) Impresión de la carcasa

Utilizando una impresora 3D y filamento PLA, se imprimieron las distintas piezas del módulo hidropónico. Durante este proceso se ajustaron parámetros de impresión y tolerancias para garantizar la estabilidad estructural y la correcta colocación de los sensores y cableado.

i) Montaje del módulo

Una vez disponibles todas las piezas, se ensambló el módulo completo, incluyendo el encaje de sensores, conexión del circuito y pruebas preliminares de funcionamiento. Esta etapa permitió validar la parte física del sistema antes de comenzar con su programación.

j) Programación del módulo

Se desarrolló el código que permite al ESP32 recoger datos de los sensores, formatearlos y enviarlos al sistema mediante Kafka. También se configuraron rutinas automáticas que permiten ejecutar acciones como activar ventilación o iluminación en función de valores críticos.

Origen de datos

k) Creación de la API de origen de datos

Se desarrolló una API ligera en FastAPI que actúa como puente entre el módulo físico y el sistema. Esta API permite recibir solicitudes periódicas desde el microcontrolador y estructurar los datos antes de su envío, garantizando que la información fluya correctamente hacia las capas superiores del sistema.

l) Programación del productor Kafka

Se implementó un productor Kafka que toma los datos provenientes de la API y los envía a un topic específico en el sistema de mensajería. Este productor se encarga de asegurar que los datos lleguen de forma rápida y confiable al consumidor, permitiendo desacoplar la adquisición de datos de su posterior almacenamiento.

Persistencia de datos

m) Diseño de la base de datos

Se diseñó un esquema relacional en MariaDB para almacenar de forma estructurada todos los datos generados por los sensores. El diseño contempla la escalabilidad del sistema y permite la consulta eficiente de los registros históricos, incluyendo información por módulo y por fecha.

n) Creación de la base de datos

Se llevó a cabo la implementación práctica del modelo de datos definido, creando tablas, relaciones y restricciones necesarias. Además, se realizaron pruebas de integridad y rendimiento para asegurar un almacenamiento fiable y sin pérdidas de información.

o) Creación de api para la base de datos

Se desarrolló una API en FastAPI que actúa como intermediaria entre el frontend y la base de datos. Esta API permite acceder a los datos, aplicar filtros por fecha o tipo de módulo y realizar operaciones CRUD, todo ello respetando las reglas de seguridad y acceso definidas.

p) Programación del consumidor Kafka

El consumidor Kafka recibe los mensajes generados por el productor y los interpreta para insertarlos automáticamente en la base de datos. Se implementó un sistema de validación para asegurar que los datos sean coherentes y se almacenen correctamente sin duplicados ni errores.

Explotación de datos

q) Boceto diseño gráfico de la aplicación web

Se creó un prototipo visual de la interfaz gráfica utilizando principios de usabilidad y diseño centrado en el usuario. Este boceto incluyó la disposición de gráficos, menús y paneles interactivos, sirviendo como guía para el desarrollo del frontend en React.

r) Desarrollo estructura básica de la web

Se implementaron las rutas principales, componentes base y lógica de navegación de la plataforma. Esta estructura permite que el usuario acceda fácilmente a las diferentes funcionalidades, como la visualización de datos, configuración de módulos y consultas históricas.

s) Visualización de datos de ultima lectura

Se implementó un panel principal que muestra en tiempo real los valores más recientes de cada sensor del módulo hidropónico. Esta vista proporciona una lectura rápida del estado actual del cultivo, permitiendo al usuario actuar de inmediato si se detectan valores críticos.

t) Visualización de datos haciendo uso de graficas

Se integraron gráficas dinámicas que permiten al usuario visualizar la evolución de los datos registrados a lo largo del tiempo. Esta funcionalidad facilita el análisis del rendimiento del cultivo y ayuda a identificar patrones o anomalías en el entorno.

u) Paneles de configuración para automatizaciones

Se desarrollaron herramientas para que el usuario pueda definir reglas personalizadas de automatización, como activar el riego o la ventilación si se superan ciertos umbrales. Estos paneles convierten la plataforma en una solución inteligente, adaptable a diferentes tipos de cultivo.

v) Lógica de gestión de múltiples módulos

La aplicación fue diseñada para soportar varios módulos hidropónicos simultáneamente. Se creó una lógica de gestión que permite registrar, supervisar y configurar individualmente cada módulo desde la misma interfaz web, facilitando la expansión del sistema en el futuro.

Sistema de recomendaciones personalizadas

w) Pagina dedicada para imágenes de los módulos

Se incorporó una sección en la aplicación donde se muestran imágenes capturadas por las cámaras instaladas en los módulos. Esta funcionalidad permite al usuario observar el estado visual del cultivo y tener un control adicional mediante visión remota.

x) Desarrollo de la sección de peticiones y recomendaciones

Se llevó a cabo una sección web donde se concentran las acciones relacionadas con peticiones de recomendación personalizada para el módulo seleccionado por el usuario. En esta sección el usuario puede enviar peticiones de revisión del estado del cultivo con el fin de mejorar los resultados del producto.

y) Sistema de envío de petición personalizada

El sistema de envío de peticiones se activa bajo demanda del usuario. El sistema recoge los datos necesarios y disponibles para la realización de una recomendación. Los datos son enviados y procesados, obteniendo una nueva resolución. Estas recomendaciones se almacenan en base de datos.

z) Visualización de recomendación personalizada para el usuario

En la sección de peticiones, el usuario podrá encontrar su listado de recomendaciones de forma ordenada cronológicamente.

Control de calidad

a) Creación test automáticos para Apis

Se diseñaron e implementaron pruebas automatizadas que verifican el correcto funcionamiento de los endpoints de las APIs. Estas pruebas aseguran que las funcionalidades principales de la plataforma se mantengan operativas tras cualquier actualización o modificación del sistema.

b) Creación test automáticos para la web

Se desarrollaron pruebas para comprobar que la interfaz web responde correctamente ante diferentes interacciones del usuario. Se validó la carga de datos, la navegación entre secciones y el correcto despliegue de los gráficos, minimizando así errores en producción.

Tareas finales

Nombre de tarea
a) Documentación de producto
b) Creación de la memoria de desarrollo y producto
c) Corrección de errores
d) Tareas de mantenimiento

Figura 2. 7. Tareas finales

a) Documentación de producto

Se elaboró una documentación técnica que detalla la arquitectura del sistema, los componentes utilizados y la lógica de funcionamiento de cada módulo. Esta documentación está orientada a facilitar el mantenimiento del proyecto y su posible evolución por parte de futuros desarrolladores.

b) Creación de la memoria de desarrollo y producto

Se redactó la memoria oficial del Trabajo de Fin de Grado, recogiendo todo el proceso de desarrollo desde la concepción de la idea hasta la validación final. En ella se incluyen los fundamentos teóricos, decisiones técnicas, metodología utilizada y resultados obtenidos.

c) Corrección de errores

Durante esta fase se revisaron todas las funcionalidades del sistema, corrigiendo errores detectados en pruebas anteriores. Se depuró el código tanto del backend como del frontend, y se ajustaron pequeños detalles visuales y de rendimiento para mejorar la experiencia del usuario.

d) Tareas de mantenimiento

Se implementaron acciones preventivas y de mantenimiento para garantizar la estabilidad del sistema en el tiempo. Esto incluye la actualización de librerías, refactorización de código redundante y mejora de logs, así como una revisión general del entorno Dockerizado para asegurar compatibilidad futura.

2.2.2 Asignación de tareas y estimación

En esta sección se muestra la asignación de tareas y estimación de duraciones organizado por grupo de desarrollo asignado.

Figura 2. 8 Estimación de tareas

Nombre de tarea	Trabajo	Duración
Nombres de los recursos: Desarrollador backend	**288 horas**	**39d**
Creación de la api de origen de datos	16 horas	2 días
Desarrollador backend	*16 horas*	
Programación del productor Kafka	8 horas	1 día
Desarrollador backend	*8 horas*	
Diseño de la base de datos	8 horas	1 día
Desarrollador backend	*8 horas*	
Creación de la base de datos	8 horas	1 día
Desarrollador backend	*8 horas*	
Creación de api para la base de datos	16 horas	2 días
Desarrollador backend	*16 horas*	
Programación del consumidor Kafka	8 horas	1 día
Desarrollador backend	*8 horas*	
Desarrollo de la sección de peticiones y recomendaciones	112 horas	14 días

Desarrollador backend	*112 horas*	
Sistema de envío de petición personalizada	56 horas	7 días
Desarrollador backend	*56 horas*	
Creación test automáticos para Apis	56 horas	7 días
Desarrollador backend	*56 horas*	

Nombre de tarea	Trabajo	Duración
Nombres de los recursos: Desarrollador backend;Desarrollador frontend	**72 horas**	**33d**
Lógica de gestión de múltiples módulos	24 horas	3 días
Desarrollador backend	*8 horas*	
Desarrollador frontend	*16 horas*	
Visualización de recomendación personalizada para el usuario	48 horas	3 días
Desarrollador backend	*24 horas*	
Desarrollador frontend	*24 horas*	

Nombre de tarea	Trabajo	Duración
Nombres de los recursos: Desarrollador frontend	**152 horas**	**52d**
Desarrollo estructura básica de la web	56 horas	7 días
Desarrollador frontend	*56 horas*	
Visualización de datos de ultima lectura	8 horas	1 día
Desarrollador frontend	*8 horas*	
Visualización de datos haciendo uso de graficas	8 horas	1 día
Desarrollador frontend	*8 horas*	
Paneles de configuración para automatizaciones	16 horas	2 días
Desarrollador frontend	*16 horas*	
Pagina dedicada para imágenes de los módulos	8 horas	1 día
Desarrollador frontend	*8 horas*	
Creación test automáticos para la web	56 horas	7 días
Desarrollador frontend	*56 horas*	

Nombre de tarea	Trabajo	Duración
Nombres de los recursos: Desarrollador frontend;Jefe de proyecto	**112 horas**	**7d**
Boceto diseño gráfico de la aplicación web	112 horas	7 días
Desarrollador frontend	*56 horas*	
Jefe de proyecto	*56 horas*	

Nombre de tarea	Trabajo	Duración
Nombres de los recursos: Encargado hardware	**168 horas**	**21d**
Diseño del circuito eléctrico	112 horas	14 días
Encargado hardware	*112 horas*	
Programación del módulo	56 horas	7 días
Encargado hardware	*56 horas*	

Nombre de tarea	Trabajo	Duración
Nombres de los recursos: Encargado hardware	**24 horas**	**3d**
Montaje del circuito	24 horas	3 días

Nombre de tarea	Trabajo	Duración
Nombres de los recursos: Encargado hardware;Jefe de proyecto	**64 horas**	**4d**
Diseño de la carcasa 3D	64 horas	4 días
Encargado hardware	*32 horas*	
Jefe de proyecto	*32 horas*	

Nombre de tarea	Trabajo	Duración
Nombres de los recursos: Encargado hardware	**40 horas**	**5d**
Impresión de la carcasa	40 horas	5 días

Nombre de tarea	Trabajo	Duración
Nombres de los recursos: Encargado hardware	**16 horas**	**2d**
Montaje del módulo	16 horas	2 días

Nombre de tarea	Trabajo	Duración
Nombres de los recursos: Jefe de proyecto	**56 horas**	**12d**
Definición de la idea	40 horas	5 días
Jefe de proyecto	*40 horas*	
Configuraciones del repositorio de Código	16 horas	2 días
Jefe de proyecto	*16 horas*	

Nombre de tarea	Trabajo	Duración
Nombres de los recursos: Jefe de proyecto;Desarrollador backend	**64 horas**	**4d**
Creación de la arquitectura del proyecto	64 horas	4 días
Desarrollador backend	*32 horas*	
Jefe de proyecto	*32 horas*	

Nombre de tarea	Trabajo	Duración
Nombres de los recursos: Jefe de proyecto	**40 horas**	**5d**
Diseño de la arquitectura base	40 horas	5 días

2.2.3 Planificación de tareas

En la siguiente figura se observa un diagrama de Gantt con una vista general de la planificación completa del proyecto

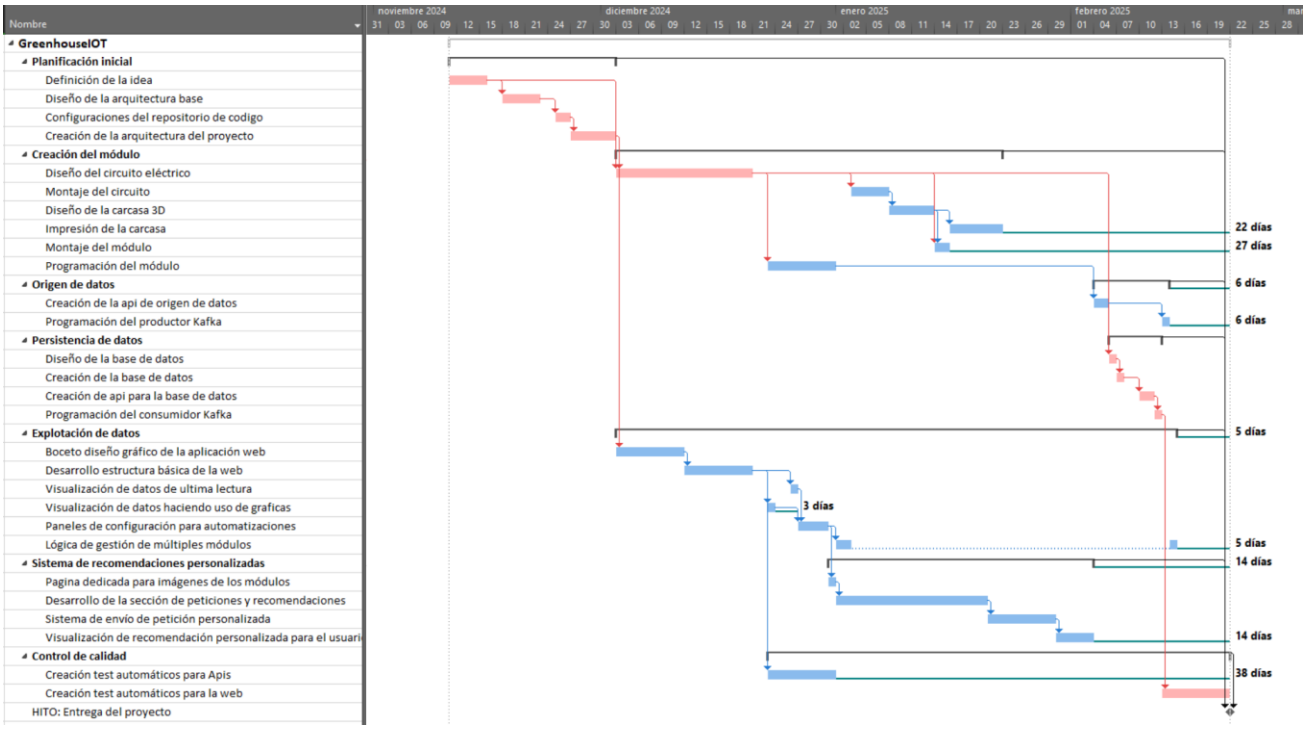


Figura 2. 9. Diagrama de Gantt

En la siguiente tabla se muestra un ejemplo de planificación concreta de desarrollo comprendida entre las fechas 11/11/24 y 21/02/25

Figura 2. 10. Ejemplo de planificación

Nombre	Duración	Comienzo	Fin
GreenhouseIOT	**75 días**	**lun 11/11/24**	**vie 21/02/25**
Planificación inicial	**16 días**	**lun 11/11/24**	**lun 02/12/24**
Definición de la idea	5 días	lun 11/11/24	vie 15/11/24
Diseño de la arquitectura base	5 días	lun 18/11/24	vie 22/11/24
Configuraciones del repositorio de código	2 días	lun 25/11/24	mar 26/11/24
Creación de la arquitectura del proyecto	4 días	mié 27/11/24	lun 02/12/24
Creación del módulo	**37 días**	**mar 03/12/24**	**mié 22/01/25**
Diseño del circuito eléctrico	14 días	mar 03/12/24	vie 20/12/24
Montaje del circuito	3 días	vie 03/01/25	mar 07/01/25
Diseño de la carcasa 3D	4 días	mié 08/01/25	lun 13/01/25

Impresión de la carcasa	5 días	jue 16/01/25	mié 22/01/25
Montaje del módulo	2 días	mar 14/01/25	mié 15/01/25
Programación del módulo	7 días	lun 23/12/24	mar 31/12/24
Origen de datos	**8 días**	**mar 04/02/25**	**jue 13/02/25**
Creación de la api de origen de datos	2 días	mar 04/02/25	mié 05/02/25
Programación del productor Kafka	1 día	jue 13/02/25	jue 13/02/25
Persistencia de datos	**5 días**	**jue 06/02/25**	**mié 12/02/25**
Diseño de la base de datos	1 día	jue 06/02/25	jue 06/02/25
Creación de la base de datos	1 día	vie 07/02/25	vie 07/02/25
Creación de api para la base de datos	2 días	lun 10/02/25	mar 11/02/25
Programación del consumidor Kafka	1 día	mié 12/02/25	mié 12/02/25
Explotación de datos	**54 días**	**mar 03/12/24**	**vie 14/02/25**
Boceto diseño gráfico de la aplicación web	7 días	mar 03/12/24	mié 11/12/24
Desarrollo estructura básica de la web	7 días	jue 12/12/24	vie 20/12/24
Visualización de datos de ultima lectura	1 día	jue 26/12/24	jue 26/12/24
Visualización de datos haciendo uso de graficas	1 día	lun 23/12/24	lun 23/12/24
Paneles de configuración para automatizaciones	2 días	vie 27/12/24	lun 30/12/24
Lógica de gestión de múltiples módulos	3 días	mié 01/01/25	vie 14/02/25
Sistema de recomendaciones personalizadas	**25 días**	**mar 31/12/24**	**lun 03/02/25**
Pagina dedicada para imágenes de los módulos	1 día	mar 31/12/24	mar 31/12/24
Desarrollo de la sección de peticiones y recomendaciones	14 días	mié 01/01/25	lun 20/01/25
Sistema de envío de petición personalizada	7 días	mar 21/01/25	mié 29/01/25
Visualización de recomendación personalizada para el usuario	3 días	jue 30/01/25	lun 03/02/25
Control de calidad	**45 días**	**lun 23/12/24**	**vie 21/02/25**
Creación test automáticos para Apis	7 días	lun 23/12/24	mar 31/12/24
Creación test automáticos para la web	7 días	jue 13/02/25	vie 21/02/25
HITO: Entrega del proyecto	0 días	vie 21/02/25	vie 21/02/25

2.3 Gestión de recursos

En este apartado se van a especificar cuáles han sido los diferentes recursos utilizados en el desarrollo. En la siguiente imagen se muestra el listado completo de recursos. Muchos de ellos cuentan con características particulares especificadas en el siguiente apartado.

Figura 2. 11. Gestión de recursos

Nombre del recurso	Tipo	Etiqueta de material	Iniciales	Capacidad máxima	Tasa estándar
Jefe de proyecto	Trabajo		J	100%	30,00 €/hora
Desarrollador frontend	Trabajo		D	100%	25,00 €/hora
Desarrollador backend	Trabajo		D	100%	25,00 €/hora
Encargado hardware	Trabajo		E	100%	25,00 €/hora
Impresora 3D	Material	und	I		319,00 €
Plástico PLA	Material	bobina 850g	P		21,00 €
portátiles	Material	und	P		429,00 €
Cableado	Material	metros	C		11,49 €
DHT22	Material	und	D		4,63 €
Sensor temperatura del agua	Material	und	S		1,39 €
Sensor TDS	Material	und	S		3,49 €
Fuente de alimentación	Material	und	F		7,24 €
Regulador de voltaje	Material	und	R		2,97 €
Foco led	Material	und	F		5,79 €
Tornillería	Material	caja	T		4,79 €
LDR	Material	und	L		1,42 €
Resistencias	Material	caja	R		2,92 €
Modulo 4x Relé	Material	und	M		2,42 €
Conexiones	Material	caja	C		9,80 €
ESP32 S3	Material	und	E		6,03 €
ESP32 Cam	Material	und	E		7,49 €
Contenedores	Material	und	C		4,50 €
Oxigenador	Material	und	O		2,49 €
Ventilador	Material	und	V		1,54 €
Pegamento	Material	caja	P		3,50 €
Tuercas inyección	Material	caja	T		3,99 €
pcb	Material	caja	p		2,84 €
lente 77mm	Material	und	l		6,36 €
Nivel de agua	Material	und	N		2,79 €

Soldador	Material	kit	S		21,00 €

2.3.1 Especificación de recursos

La elección de los recursos utilizados en este proyecto no se ha basado únicamente en su disponibilidad, sino que ha seguido criterios técnicos, económicos y de compatibilidad con el entorno de desarrollo propuesto.

Por ejemplo, el microcontrolador ESP32 S3 fue seleccionado por su bajo consumo energético, integración nativa de WiFi/Bluetooth y suficiente capacidad de cómputo para manejar tareas de adquisición y envío de datos en tiempo real.

Asimismo, se optó por sensores como el DHT22 y DS18B20 por su fiabilidad y facilidad de integración mediante protocolos digitales, facilitando su conexión directa al ESP32 sin necesidad de conversores adicionales. El uso del sensor DHT22 frente a otras alternativas como el sensor DHT11 ha permitido obtener lecturas mucho más precisas.

En cuanto a la impresión 3D, la impresora Ender-3 V3 KE permite obtener piezas funcionales a bajo coste con una calidad suficiente para soportar las condiciones del entorno hidropónico, especialmente gracias al uso de PLA+, que ofrece mayor resistencia térmica y mecánica que el PLA convencional.

Figura 2. 12. Especificación de recursos

Nombre del recurso	**Tipo**	**Modelo / Especificaciones**	**Observaciones**	**Nombre del recurso**
Impresora 3D	Hardware	Máximo 500 mm/s, extrusor Sprite, admite múltiples filamentos	Control por WiFi, USB y teléfono	Impresora 3D
Filamento PLA+	Consumible	850g, Ø 1.75 mm, 212-230 ºC	Compatible con la impresora	Filamento PLA+
Portátiles	Hardware	HP 255 G9, Ryzen 5 5625U, 8GB RAM, 512GB SSD, 15.6"	Uso en desarrollo	Portátiles
Cableado	Material	Cable rígido 22AWG, 2m, núcleo de cobre sólido	Para conexión entre componentes	Cableado
Sensor DHT22	Electrónico	Temp: -40°C a 80°C, Hum: 0-100% RH, Precisión: ±0.5°C y ±2% RH	Sensor combinado de temperatura y humedad	Sensor DHT22
Sensor de temperatura agua	Electrónico	DS18B20, Rango: -55°C a +125°C, 3-5.5V, resolución 9-12 bits	Apto para inmersión	Sensor de temperatura agua

Sensor TDS	Electrónico	Rango señal: 0-2.3V, Voltaje: 3.3-5.5V, señal AC	Mide sólidos disueltos en agua	Sensor TDS
Fuente de alimentación	Electrónico	12V, 6A, 72W	Suministro general de energía	Fuente de alimentación
Regulador de voltaje	Electrónico	XL4016 DC-DC, entrada: 5-40V, salida: 1.2-35V, máx. 9A	Módulo reductor ajustable	Regulador de voltaje
Foco LED	Iluminación	5V, cuerpo de aluminio, intensidad ajustable	Luz de crecimiento para el cultivo	Foco LED
Tornillería	Consumible	M2/M3, 100 unidades de tornillos y tuercas por tamaño	Para montaje de estructuras	Tornillería
LDR	Electrónico	Fotoresistencia 5528	Detecta intensidad lumínica	LDR
Resistencias	Electrónico	Kit 600 unidades, varios valores	Uso general en circuitos	Resistencias
Módulo 4x Relé	Electrónico	HW-316A, 5V, alimentación unificada, LEDs indicadores	Control de actuadores eléctricos	Módulo 4x Relé
Conexiones	Material	Conectores DC macho-hembra 12V, pack de 5 pares	Compatibles con 5V y 12V	Conexiones
ESP32 S3	Microcontrolador	N16R8, 44 pines, WiFi, Bluetooth, PSRAM	Control principal del módulo	ESP32 S3
ESP32-CAM	Microcontrolador	Módulo con cámara OV2640, WiFi	Para visión remota del cultivo	ESP32-CAM
Contenedores	Estructura	Contenedor plástico estándar	Soporte físico para las plantas	Contenedores
Oxigenador	Electrónico	G613A, 5V, 1W	Mantiene oxigenación del agua	Oxigenador
Ventilador	Electrónico	6010, 5V, 60x60x10 mm, 3500 rpm	Control de temperatura y humedad	Ventilador
Tuercas de inserción	Material	Latón, 220 unidades, varios diámetros	Inserción térmica en PLA	Tuercas de inserción
PCB	Material	Placa 3x7 cm, doble cara, 10 unidades	Para montaje de circuitos	PCB
Lente 77 mm	Óptico	Lente estándar de cámara	Complemento para visión con ESP32-CAM	Lente 77 mm

Sensor de nivel de agua	Electrónico	Sensor sin contacto, 3.3-5V	Para detección de nivel de agua	Sensor de nivel de agua
Soldador	Herramienta	Estación digital 60W, 200-480°C	Ensamblaje de circuitos	Soldador

2.4 Gestión de riesgos

El desarrollo de GreenhouseIOT, al igual que cualquier proyecto tecnológico, está sujeto a una serie de riesgos que pueden afectar negativamente al cumplimiento de plazos, calidad del producto o disponibilidad de recursos. La gestión de riesgos se ha enfocado en su identificación temprana, análisis de impacto y definición de medidas preventivas o correctoras.

2.4.1 Identificación de riesgos

A continuación, se detallan los principales riesgos detectados y las estrategias adoptadas para su mitigación:

Figura 2. 13. Identificación de riesgos

ID	**Riesgo**	**Probabilidad**	**Impacto**	**Medidas preventivas/correctoras**
R1	Fallo de hardware durante pruebas	Media	Alta	Duplicar sensores y componentes críticos. Validación previa en entorno controlado.
R2	Retrasos en el desarrollo de alguna capa del sistema	Alta	Media	Desarrollo paralelo con entregas iterativas. Revisión semanal del progreso.
R3	Dificultades de integración entre microservicios	Media	Alta	Uso de APIs bien documentadas y pruebas automatizadas de integración.
R4	Errores de comunicación entre hardware y software	Media	Alta	Pruebas exhaustivas del protocolo de comunicación y manejo de errores en firmware.
R5	Cambios de requisitos del usuario final	Baja	Media	Aplicación de metodología ágil. Flexibilidad en el desarrollo por iteraciones.
R6	Pérdida o corrupción de datos	Baja	Media	Uso de base de datos con copias de seguridad periódicas. Validación de datos en entrada.

R7	Escasa respuesta de la comunidad sobre tecnologías usadas	Baja	Baja	Documentación oficial, foros de desarrolladores y recursos educativos complementarios.

Matriz de análisis de riesgos

A continuación, se presenta una matriz de riesgos que permite priorizar las amenazas en función de su impacto y probabilidad:

Riesgo	**Probabilidad**	**Impacto**	**Nivel de Riesgo**
Fallo de hardware durante pruebas	Media	Alta	Alto
Retrasos en el desarrollo	Alta	Media	Alto
Problemas de integración	Media	Alta	Alto
Errores en la comunicación HW/SW	Media	Alta	Alto
Cambios de requisitos	Baja	Media	Medio
Pérdida o corrupción de datos	Baja	Media	Medio

Figura 2. 14. Tabla de análisis de riesgos

2.4.2 Ciberseguridad

Uno de los puntos más relevantes en las aplicaciones expuestas a internet es la ciberseguridad de los servicios. La aplicación maneja datos en tiempo real de usuarios y cultivos por lo que es importante asegurar cierto nivel de fiabilidad, eficiencia y seguridad de datos. En GreenhouseIoT los puntos críticos se concentran en la transmisión de los datos y el tratamiento.

Se han tomado las siguientes medidas para incrementar la seguridad del servicio:

- Exposición de APIs sin autenticación: Las APIs REST desarrolladas con FastAPI pueden ser objetivo de accesos no autorizados si no se implementan mecanismos de autenticación y control de acceso. Esto podría permitir la lectura, manipulación o eliminación de datos por parte de terceros. El uso de claves de autenticación ha solucionado este punto.

- Intercepción de datos sensibles: Dado que la plataforma transmite valores como TDS, pH, temperatura o imágenes del sistema, el uso de protocolos inseguros podría permitir ataques, comprometiendo la confidencialidad de los datos. El servicio utiliza protocolos cifrados y

seguros. Auth0 implementa un sistema por el cual nos aseguramos de que todas las secciones de la aplicación naveguen de forma segura ya que si alguna usase un protocolo sin cifrar no permitiría la comunicación.

- Se han limitado los puertos de los servicios expuestos, dejando únicamente los imprescindibles para el correcto funcionamiento del sistema.
- Los dispositivos IoT se han programado para que sean ellos los que recuperar cualquier tipo de información o programación. No se ha permitido la comunicación directa con estos dispositivos por parte del usuario. Un ejemplo es la gestión de la configuración, cuando un usuario modifica o programa alguna configuración no se comunica con los módulos directamente, sino que sube configuraciones a la base de datos. Estas configuraciones son recuperadas por los módulos de forma periódica.
- No se almacenan datos críticos de usuarios. Al hacer uso de Auth0, las credenciales más sensibles de los usuarios son gestionadas por los responsables de Auth0. Delegar el tratamiento de credenciales de usuarios a empresas especializadas incrementa la seguridad de los datos.

2.5 Legislación y normativa

Durante el desarrollo de GreenhouseIOT se han tenido en cuenta diversas normativas aplicables tanto desde el punto de vista técnico como legal. Aunque el sistema se ha desarrollado con fines educativos, se han seguido buenas prácticas contenidas en las siguientes normativas.

Normativa técnica aplicada

- UNE 157001:2004: Normas para la redacción de proyectos, en cuanto a estructura y requisitos formales del documento.
- ISO/IEC 27001 – Seguridad de la información.
    - Aunque no se ha certificado formalmente el sistema bajo esta norma, se han adoptado varios de sus principios para proteger la integridad y disponibilidad de la información.
    - Gestión de riesgos: se han identificado amenazas como pérdida de datos o acceso no autorizado, y se han implementado medidas preventivas como control de acceso a las las APIs.
    - Seguridad en el desarrollo: los microservicios están aislados en contenedores Docker, y las comunicaciones internas se realizan en red privada.

- Reglamento General de Protección de Datos (RGPD - UE 2016/679). Aunque no se almacenan datos personales sensible de los usuarios, se han seguido las siguientes pautas a la hora de desarrollar el sistema:
    - Minimización de datos: el sistema solo almacena los datos necesarios para el funcionamiento del módulo y datos mínimos de usuario. El usuario al registrarse adquiere un identificador numérico que es usado para la gestión del servicio. Las lecturas tienen asociado un módulo y los módulos un identificador numérico de usuario. No se realizan peticiones a Apis internas haciendo uso de datos personales.
    - Control de acceso: la base de datos se encuentra protegida por credenciales y en una red privada Docker no expuesta al exterior.
    - En caso de realizar el despliegue haciendo uso del proyecto Ansible recogido en el Anexo 3, el servidor solo es accesible por ssh haciendo uso de las claves ssh establecidas durante la configuración. Eliminando cualquier intento de inicio de sesión por contraseña.
    - Registro y trazabilidad: cada dato tiene asociado un identificador de módulo y fecha de obtención, lo que facilita auditorías sin requerir información personal.
- Reglamento UE 2024/1689 sobre Inteligencia Artificial.

  Este nuevo reglamento, aprobado por el Parlamento Europeo y el Consejo el 13 de junio de 2024, establece requisitos para garantizar la transparencia y responsabilidad en los sistemas de inteligencia artificial que se implementen en la Unión Europea.

  Aunque GreenhouseIOT no utiliza sistemas de IA de alto riesgo, se han adoptado principios de transparencia, asegurando que cualquier algoritmo o sistema automatizado sea documentado y sus decisiones puedan ser auditadas. Se promueve la trazabilidad y explicabilidad de los procesos automatizados para facilitar la confianza y el cumplimiento normativo. De forma práctica, el uso de inteligencia artificial en el proyecto se concentra en la creación de recomendaciones para el usuario.

Capítulo 3

Solución

En esta sección se va a explicar detalladamente todo el proceso de desarrollo.

3.1 Descripción de la solución

GreenhouseIOT es un sistema distribuido basado en microservicios orientado a la gestión de datos procedentes de invernaderos inteligentes. La solución propuesta responde a la necesidad de supervisión remota de condiciones ambientales en cultivos mediante tecnologías modernas, modulares y escalables.

La arquitectura del sistema se ha dividido en cuatro capas principales:

- Capa de origen de datos (Data Source): implementada mediante microcontroladores ESP32 con sensores DHT22 conectados, encargados de tomar lecturas de temperatura y humedad y transmitirlas a través de la red Wi-Fi. En esta capa englobaríamos una API dedicada a la preparación de los datos para ser enviados a través de un bus de datos Kafka
- Capa de procesamiento (Data Processing): un microservicio desarrollado en Python que consume los datos publicados en el bus Kafka. También se encarga de actuar como interfaz REST para trabajar con la base de datos
- Capa de persistencia (Data Persistence): una base de datos relacional MariaDB que almacena la información ya estructurada para su consulta posterior.
- Capa de explotación (Data Mining): una interfaz de usuario construida con React que permite consultar, visualizar y analizar los datos mediante gráficas e informes.

Cada capa se ha implementado como un contenedor Docker, facilitando el despliegue, aislamiento y escalabilidad del sistema. Se ha optado por esta estructura para permitir actualizaciones independientes y mantenimiento modular.

3.2 El proceso de desarrollo

3.2.1 Análisis

En esta fase se estudiaron los requisitos funcionales del sistema, que incluyen:

- Recolección periódica de datos ambientales.
- Envío automático de los datos al servidor de procesamiento.
- Validación de los datos recibidos (rango, formato, frecuencia).
- Almacenamiento estructurado en una base de datos SQL.
- Visualización de los datos mediante una interfaz web.
- Modularidad para permitir la integración futura de nuevos sensores (luz, CO2, etc.).

El siguiente listado muestra los requisitos que debe cumplir el sistema

Requisitos funcionales

RF 1: El sistema debe permitir la recepción automática de datos desde sensores IoT.

RF 2: El sistema debe procesar y validar los datos antes de almacenarlos.

RF 3: Los datos deben almacenarse en una base de datos estructurada y relacional.

RF 4: La interfaz web debe permitir visualizar la información en tiempo casi real.

RF 5: Los usuarios deben poder seleccionar un invernadero específico para visualizar sus datos.

RF 6: Los datos deben representarse mediante etiquetas, gráficos y tablas.

RF 7: El sistema debe enviar alertas visuales si los valores superan ciertos umbrales.

RF 8: El sistema debe listar los módulos conectados y desconectados.

RF 9: El usuario podrá configurar y programar procesos realizados por los módulos.

RF 10: El sistema debe ofrecer gráficas simplificadas y detalladas de los distintos parámetros medidos.

RF 11: La interfaz debe permitir el filtrado de datos por fecha y tipo de sensor.

RF 12: La API debe aceptar datos en formato JSON desde dispositivos remotos.

RF 13: El sistema debe admitir múltiples invernaderos con datos independientes.

RF 14: Cada módulo debe tener un identificador único para su trazabilidad.

RF 15: El sistema debe registrar la fecha y hora exacta de cada medición.

RF 16: La interfaz debe mostrar la última medición registrada para cada sensor.

RF 17: El sistema debe iniciarse automáticamente cuando se despliegue mediante Docker.

RF 18: La interfaz debe estar adaptada para su uso desde ordenador, Tablet y móvil.

RF 19: El usuario debe poder consultar el histórico de lecturas por invernadero.

RF 20: El sistema debe mostrar un resumen estadístico de los datos.

Requisitos no funcionales

RNF 1: El sistema debe ser tolerante a fallos de red temporales.

RNF 2: El sistema debe tener una latencia de respuesta inferior a 1 segundo en la visualización.

RNF 3: El sistema debe ejecutarse en cualquier sistema operativo que soporte Docker.

RNF 4: El sistema debe ser modular para permitir el reemplazo independiente de cada servicio.

RNF 5: El sistema debe permitir la integración futura con servicios en la nube (por ejemplo, VPS).

RNF 6: La arquitectura debe estar preparada para añadir sensores adicionales sin rediseñar el sistema completamente.

RNF 7: La seguridad debe incluir validación de origen de datos.

RNF 8: El sistema debe ser capaz de manejar múltiples dispositivos simultáneamente.

Especificación de requisitos

En esta sección se especifican los principales casos de uso

Caso de uso 1

Descripción: Recepción automática de datos

Actores: Módulo IoT (sensor), API

Precondiciones: El módulo debe estar encendido y tener conexión de red

Flujo básico:

1. El módulo IoT recoge datos del entorno (temperatura, humedad, etc.)
2. Los datos son enviados vía HTTP en formato JSON a la API
3. La API recibe la información correctamente

   Post condiciones: Los datos se registran en el sistema para su posterior validación y almacenamiento

---

Caso de uso 2

Descripción: Validación y procesamiento de datos

Actores: API, sistema de validación

Precondiciones: Se ha recibido un paquete de datos

Flujo básico:

1. El sistema comprueba si el JSON recibido contiene los campos esperados

2. Se validan los tipos de dato y los rangos válidos
3. Si los datos son correctos, se procesan

Excepciones:

- Si el JSON está malformado, se descarta
- Si los valores son incoherentes, se descarta

  Post condiciones: Solo los datos válidos se almacenan

Caso de uso 3

Descripción: Almacenamiento en base de datos

Actores: Sistema de persistencia, base de datos

Precondiciones: El dato ya ha sido validado

Flujo básico:

1. El sistema genera una consulta SQL
2. Se inserta el dato en la tabla correspondiente del invernadero

Post condiciones: El dato queda persistido en la base de datos relacional

---

Caso de uso 4

Descripción: Visualización en tiempo real

Actores: Usuario, sistema de visualización

Precondiciones: Hay datos recientes en la base de datos

Flujo básico:

1. El usuario accede a la interfaz
2. La interfaz solicita los datos más recientes
3. Los datos se muestran en pantalla

Post condiciones: El usuario ve las últimas mediciones

Caso de uso 5

Descripción: Selección de invernadero

Actores: Usuario

Precondiciones: El usuario debe tener acceso a al menos un invernadero

Flujo básico:

1. Se listan los invernaderos disponibles
2. El usuario selecciona uno
3. El sistema filtra los datos asociados a ese invernadero

Caso de uso 6

Descripción: Representación de datos en gráficos/tablas

Actores: Usuario

Flujo básico:

1. El usuario entra a la vista de datos
2. Se renderizan los datos con etiquetas, gráficos y/o tablas

---

Caso de uso 7

Descripción: Alertas por valores fuera de umbral

Actores: Sistema

Precondiciones: Se ha recibido una medición

Flujo básico:

1. El sistema compara el valor con los umbrales definidos
2. Si se supera, se genera una alerta visual

Post condiciones: Se alerta al usuario en la interfaz

Caso de uso 8

Descripción: Listar módulos conectados/desconectados

Actores: Usuario

Flujo básico:

1. El sistema comprueba qué módulos han enviado datos recientemente
2. Muestra una lista diferenciando los activos y los inactivos

---

Caso de uso 9

Descripción: Navegación a sección de gráficas detalladas

Actores: Usuario

Flujo básico:

1. El usuario selecciona el módulo deseado
2. El usuario accede a la página principal
3. El usuario selecciona el un botón de navegar a vista detallada de gráfica o accede a la sección a través del menú superior izquierdo

---

Caso de uso 10

Descripción: Configuraciones de automatización

Actores: Usuario

Flujo básico:

1. El usuario selecciona el dispositivo a automatizar (Luces, ventiladores, oxigenación…)
2. El sistema almacena la configuración
3. Las configuraciones son recuperadas de forma periódica por los módulos

---

Caso de uso 11

Descripción: Filtrado por fecha y sensor

Actores: Usuario

Flujo básico:

1. El usuario selecciona los filtros
2. El sistema muestra solo los datos que cumplen los criterios

---

Caso de uso 12

Descripción: Recepción de JSON desde dispositivo

Actores: Módulo IoT, API

Flujo básico:

1. El dispositivo envía un JSON con los datos
2. La API lo recibe y lo interpreta

Caso de uso 13

Descripción: Soporte para múltiples invernaderos

Actores: Usuario

Flujo básico:

1. El sistema permite cambiar entre invernaderos
2. Los datos cambian automáticamente

Caso de uso 14

Descripción: Identificador único de módulo

Actores: Sistema

Precondiciones: Cada módulo debe tener un ID asignado

Flujo básico:

1. El sistema utiliza el ID para identificar el origen del dato
2. El dato se asocia correctamente al módulo correspondiente

Caso de uso 15

Descripción: Registro de timestamp de la medición

Actores: Sistema

Flujo básico:

1. Al recibirse el dato, se registra la fecha y hora exacta de captura

---

Caso de uso 16

Descripción: Mostrar última medición

Actores: Usuario

Flujo básico:

1. El sistema obtiene la medición más reciente para cada sensor
2. La presenta en la vista principal

Caso de uso 17

Descripción: Inicio automático con Docker

Actores: Sistema

Flujo básico:

1. El contenedor Docker se levanta
2. Todos los servicios se inician automáticamente

Caso de uso 18

Descripción: Diseño adaptativo

Actores: Usuario

Flujo básico:

1. El usuario accede desde cualquier dispositivo
2. La interfaz se adapta a la pantalla

Caso de uso 19

Descripción: Consultar histórico de datos

Actores: Usuario

Flujo básico:

1. El usuario indica rango de fechas dentro de una gráfica
2. El sistema recupera y muestra los datos

Caso de uso 20

Descripción: Ver resumen estadístico

Actores: Usuario

Flujo básico:

1. El sistema calcula valores medios, máximos y mínimos
2. Los datos son mostrados al usuario en la interfaz

Criterios de calidad del sistema

Fiabilidad y tolerancia a fallos

a) Criterio de calidad RNF 1: El sistema debe ser tolerante a fallos de red temporales, asegurando que los datos no se pierdan si hay desconexiones breves. Para ello, se implementará una lógica de reintentos en los módulos IoT y almacenamiento temporal local si el envío falla.
b) Criterio de calidad RNF 7: La seguridad debe garantizar que solo datos de origen legítimo sean aceptados. Se utilizarán comprobaciones para validar cada módulo.

Rendimiento

a) Criterio de calidad RNF 2: El sistema debe tener una latencia de respuesta inferior a 1 segundo en la interfaz web. Esto se asegura mediante consultas optimizadas y renderizado eficiente con React.
b) Criterio de calidad RNF 8: El sistema debe ser capaz de manejar múltiples dispositivos simultáneamente. Para ello, se utilizará FastAPI por su rendimiento asíncrono y una arquitectura desacoplada mediante microservicios en contenedores.

Portabilidad y despliegue

a) Criterio de calidad RNF 3: El sistema debe ejecutarse en cualquier sistema operativo que soporte Docker, permitiendo compatibilidad multiplataforma (Windows, Linux, macOS).
b) Criterio de calidad RNF 4: El sistema debe ser modular. Cada componente (procesamiento, base de datos, frontend) es independiente y desplegable por separado para facilitar el mantenimiento y la escalabilidad.

Escalabilidad y extensibilidad

a) Criterio de calidad RNF 5: La arquitectura debe permitir integrarse fácilmente con servicios en la nube, como VPS o plataformas como AWS/Azure, gracias al uso de contenedores y estandarización de servicios vía API.

b) Criterio de calidad RNF 6: La arquitectura debe estar preparada para añadir nuevos tipos de sensores sin rediseñar el sistema. Esto se logra mediante un esquema flexible en la base de datos y controladores de entrada adaptables.

3.2.2 Diseño

Durante el diseño se definieron los diagramas de arquitectura, las interfaces entre microservicios, y la estructura de la base de datos. La comunicación entre la capa de origen y procesamiento se realizó inicialmente vía HTTP mediante una API REST sencilla. La base de datos fue diseñada con una tabla principal de lecturas de datos, con claves foráneas hacia una tabla de invernaderos. La arquitectura fue desarrollada a partir del esquema proporcionado por la empresa HPE CDS durante la realización de prácticas.

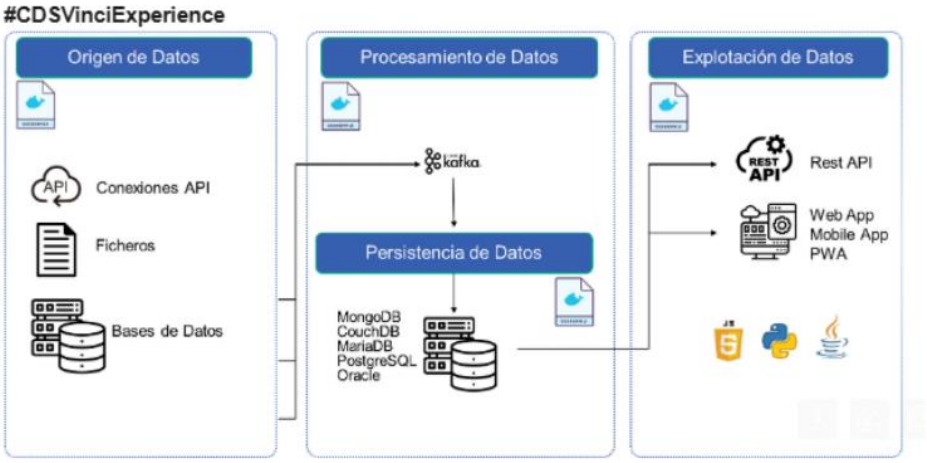


Figura 3. 1. Esquema de capas

La arquitectura se adapta todo lo posible a la imagen anterior, exceptuando algunas implementaciones particulares que fueron necesarias para GreenhouseIOT

Patrones arquitectónicos utilizados

El sistema se ha construido siguiendo principios de arquitectura basada en microservicios, adoptando además ciertos patrones bien establecidos:

- API Gateway: cada cliente (la interfaz web) se comunica exclusivamente con la API de backend, que centraliza y abstrae la lógica interna del sistema.
- Productor-Consumidor asíncrono (Kafka): utilizado para desacoplar la recolección de datos (productor) del almacenamiento persistente (consumidor).
- Separación de responsabilidades (SoC): cada microservicio tiene una función concreta (origen, persistencia, visualización) y puede escalarse de forma independiente.

Esta estrategia facilita no solo la escalabilidad futura del sistema sino también su mantenimiento, ya que cualquier fallo o mejora puede abordarse de manera modular.

3.2.3. Implementación

El desarrollo del sistema se puede organizar en dos versiones diferenciadas. El objetivo de la primera versión fue el desarrollo de un MPV que permitiese ver las funciones más básicas del sistema. Una vez se consiguió esta versión, se analizó los puntos fuertes, los puntos débiles y la proyección a futuro del sistema para terminar consiguiendo una segunda versión mucho más completa y estable.

MPV Desarrollo detallado y resultados

La primera etapa del desarrollo fue la implementación de las diferentes capas. Se profundizó en su correcta contenerización haciendo uso de Docker. Posteriormente se completaron las configuraciones necesarias para el correcto funcionamiento de las comunicaciones entre capas haciendo uso de Docker Compose.

Capa de origen de datos

La capa de datos implementa la API y el productor Kafka. Los datos son almacenados en el bus Kafka. De forma extra, se ha diseñado un prototipo de circuito eléctrico para los sensores del módulo de

micro-invernadero. El prototipo fue montado y programado con un programa simple que recoge los datos de los sensores de forma periódica.

El programa se ejecuta en una placa de desarrollo ESP32 y los datos son enviados a la API para su posterior envío por Kafka.

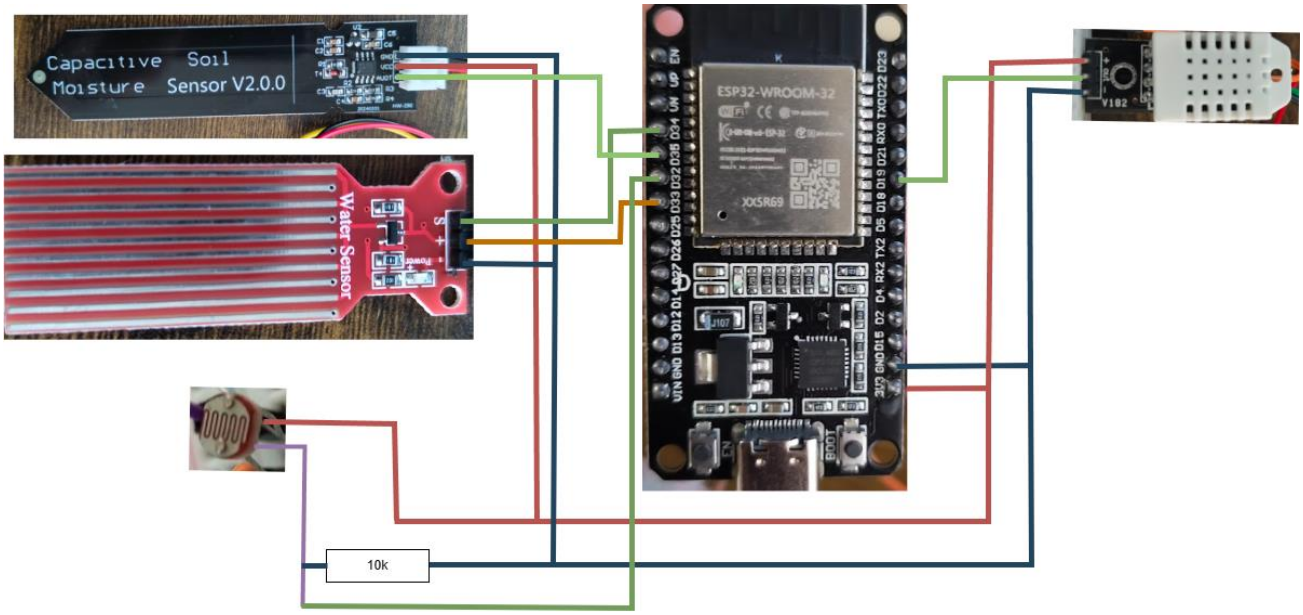
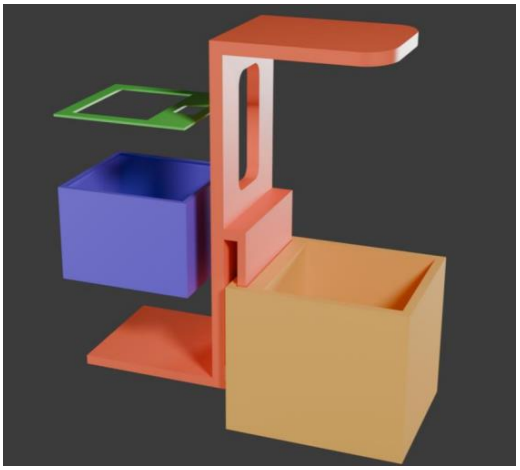


Figura 3. 2. Esquema eléctrico MPV

En la imagen se observan los sensores de humedad del aire, temperatura, luz, agua y humedad de la tierra. Este último sensor fue implementado ya que en la primera versión del sistema se optó por un sistema de cultivo en sustrato debido a tener menor requerimiento en cuanto a número de sensores. Además, se observa un sensor de agua diferente al actual, esto se debe a que el sensor de la imagen superior se degrada de forma mucho más rápida al tener que estar en contacto con el agua. El sensor actual permite incorporarse anclado al contenedor de agua por la parte externa, haciendo mediciones a través de sus paredes. El microcontrolador también es distinto, en esta primera versión se usó una versión menos potente pero más compacta.

Seguidamente se comenzó el diseño 3D del módulo obteniendo el siguiente resultado:



Sección naranja: Contenedor del sustrato
Sección roja: Estructura principal del módulo
Sección azul: Contenedor de circuitería
Sección verde: Tapa con orificios para sacar sensores y cableado.

Figura 3. 3. Renderizado primer prototipo

El módulo fue impreso en 3D y ensamblado. Se realizaron varios intentos de diseño y reimpresión hasta obtener un resultado pulido. El microcontrolador incorpora un servidor web para realizar configuraciones. Además, se incorporaron endpoints para su configuración remota.

Capa de procesamiento y persistencia

La capa de procesamiento y persistencia implementa actualmente la base de datos y la api que permite su acceso. Se desarrollaron algunos endpoints y se comprobó su correcto funcionamiento con parámetros de ejemplo. Se implementó el consumidor kafka y recibe los datos correctamente.

Capa de explotación de datos

La capa de explotación de datos fue uno de los principales puntos débiles de esta primera versión. Se desarrolló usando la tecnología Reflex, framework que permite el desarrollo de páginas web haciendo uso únicamente de Python.

Se consiguió una web funcional que permitía al usuario la visualización de los datos y realizar algunas configuraciones, pero los tiempos de carga de cada sección eran demasiado extensos. Estos tiempos de carga se debían a que Reflex compila los archivos Python a tecnologías web tradicionales cuando el usuario entraba por primera vez a cada página. Los tiempos de carga llegaban a superar los 40 segundos en algunas ocasiones. Para el estilado de la web, no se usó ninguna tecnología específica ya que Reflex fue construido haciendo uso de librerías de componentes populares como ChakraUI y RadixUI.

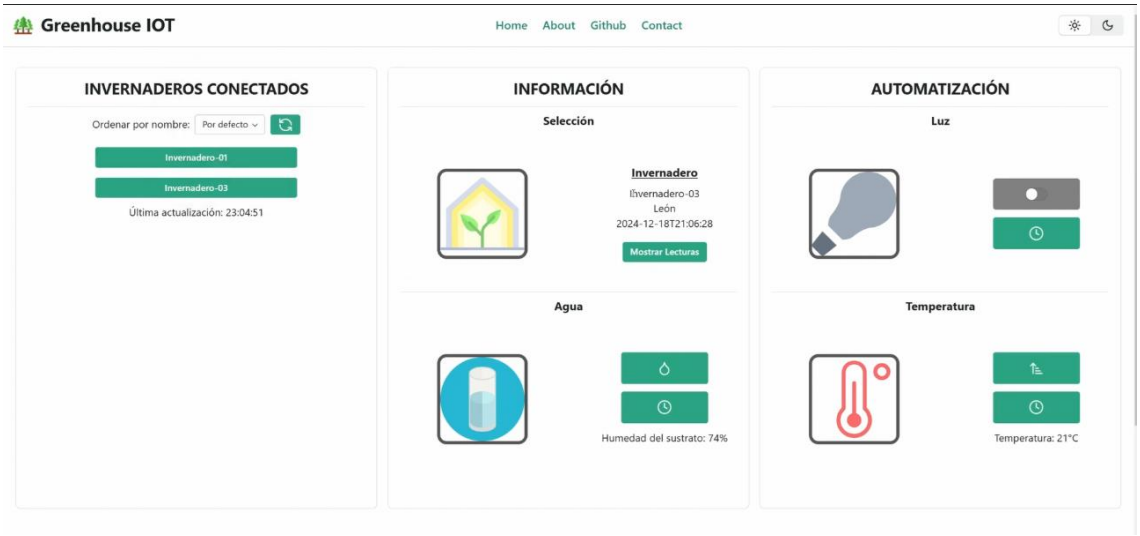


Figura 3. 4. interfaz MPV en modo claro

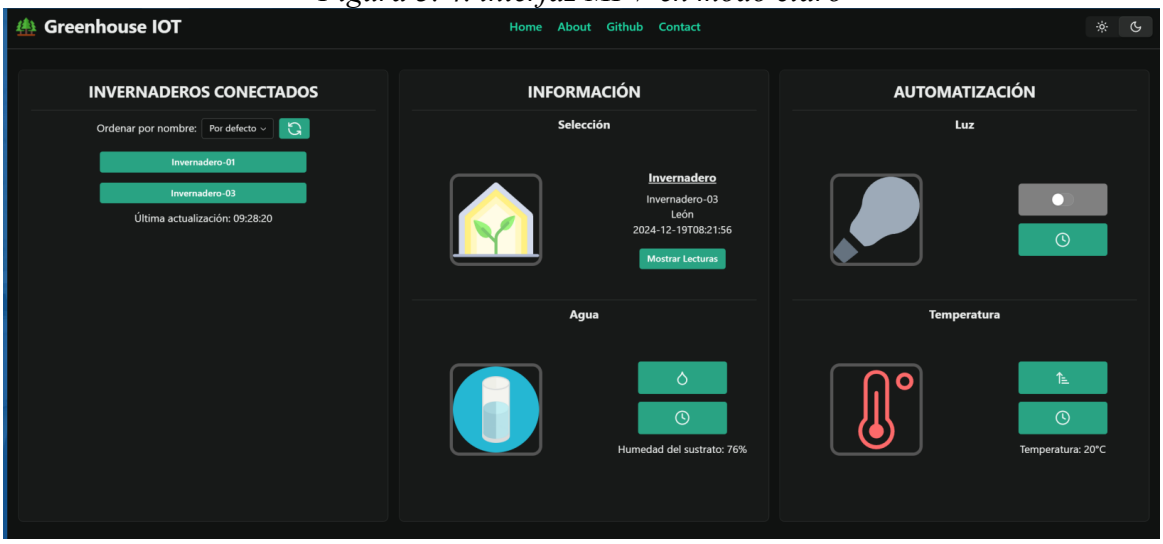


Figura 3. 5 interfaz MPV en modo oscuro

A continuación, se numeran las principales tecnologías usadas:

- Diseño
    - Blender
    - Draw.io
    - Excalidraw
- Arquitectura
    - Docker
    - Kafka
    - MariaDB
- Desarrollo
    - FastAPI
    - Reflex
    - Arduino

Una vez se terminó este prototipo se cerró el desarrollo de esta primera versión creando en el repositorio de GitHub un reléase con la etiqueta de versión 1.0.0

Versión final. Desarrollo detallado y resultados

Para la realización de esta segunda versión se tomaron en cuenta las debilidades de la primera versión y se tomaron decisiones de rediseño en ciertas áreas. En el siguiente diagrama se muestra la estrategia de despliegue de la aplicación de forma general.

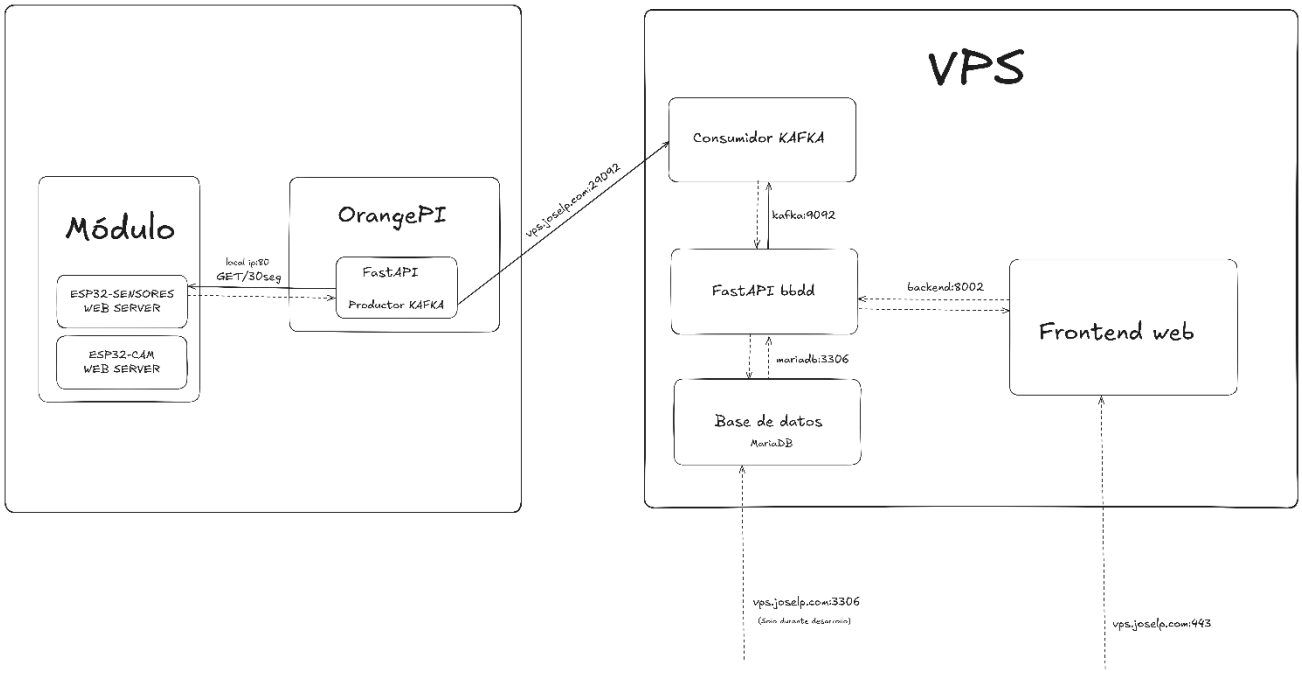


Figura 3. 6. Esquema de despliegue

Estrategia de despliegue automático

Para la estrategia de CI/CD se han realizado diversas automatizaciones comunes en el ámbito DevOps. El proceso comienza al hacer un nuevo push a la rama main del repositorio. Cuando esto sucede, una rutina de Github Actions construye, testea y publica una imagen Docker en DockerHub de cada servicio a desplegar en el servidor VPS.

Para automatizar configuraciones del servidor VPS se ha desarrollado un proyecto basado en Ansible orientado específicamente para este despliegue. El proyecto se desglosa de forma más detallada en un anexo. De forma resumida, es necesaria la ejecución del proyecto una primera vez sobre el servidor VPS con un sistema operativo Ubuntu y tras la ejecución el servidor estará completamente configurado y los servicios levantados. Cuando una imagen nueva se publica es detectado por un servicio llamado WatchTower que reinicia el servicio haciendo uso de la nueva imagen.

Capa de origen de datos

El primer prototipo nos ayudó a entender cómo se podía implementar un sistema de sensores. El desarrollo con Arduino en placas ESP32 permitía la obtención de datos a través de pines digitales y analógicos, pero también permitía la salida de señales eléctricas tanto analógicas como digitales. Una salida digital controlada permite la activación programada de un relé y con ello la automatización de cualquier dispositivo como bombas, ventiladores y oxigenadores cuando el usuario lo desee.

El primer paso fue investigar sobre los requerimientos en cuanto a sensores y activadores que necesita un cultivo hidropónico. Tras obtener un primer listado de requerimientos se comenzó a la búsqueda de los componentes que mejor se adaptasen manteniendo un presupuesto reducido.

Uno de los problemas principales es la alimentación de dispositivos como bombas de agua que requieren potencias mayores a las que un ESP32 puede aportar. La primera versión estaba alimentada por usb pero en esta segunda versión era necesario una fuente de alimentación dedicada y un voltaje muy regulado. Después de varias iteraciones de diseño se consiguió el siguiente circuito resultante.

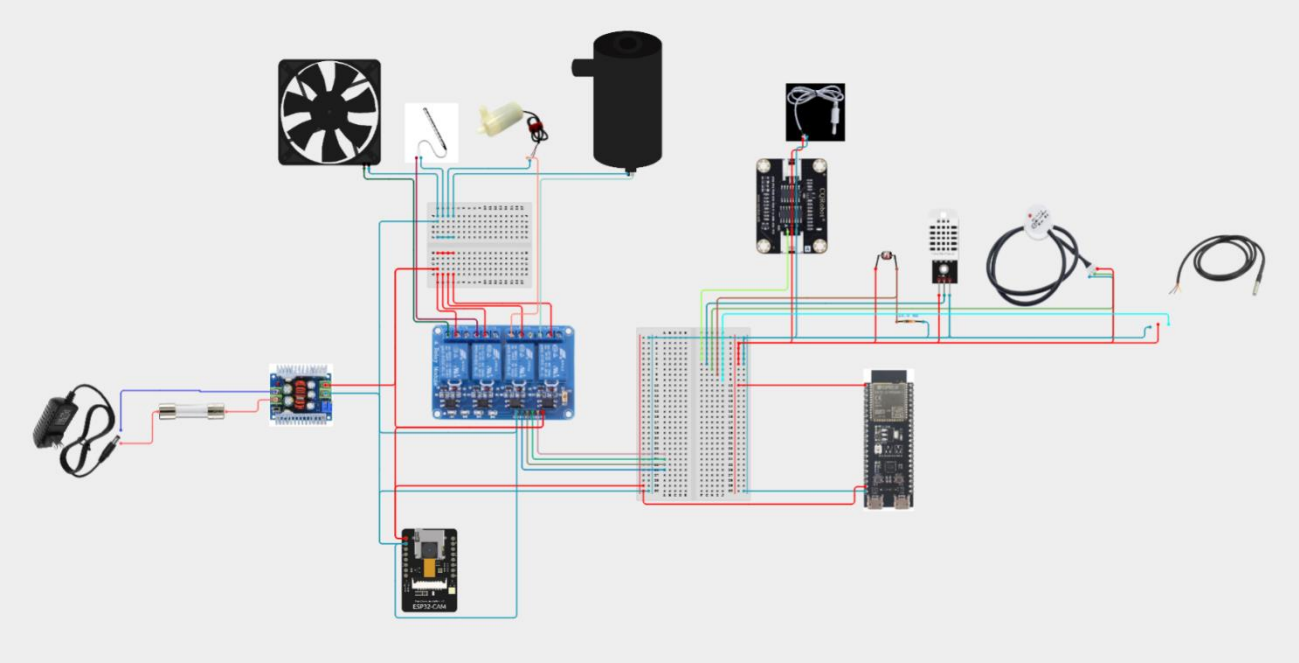


Figura 3. 7. Diagrama del circuito eléctrico

En la imagen observamos un circuito mucho mas completo y complejo. Destaca el uso de un regulador de voltaje configurado a 5V con un amplio margen de intensidad máxima. En la parte superior encontramos los dispositivos de automatización, concretamente ventiladores, luces, oxigenador y bomba de agua. Estos dispositivos pueden ser intercambiables o ampliables de forma sencilla siempre que trabajen a 5V.

Una vez se comprobó que el circuito era válido, se terminó la programación de los microcontroladores y se comenzó con el diseño del nuevo módulo 3D. Para el diseño se intentó que el proceso fuese lo más modular posible. El tamaño del nuevo módulo iba a ser muy superior por lo que reducir las piezas en tamaño y aumentar el número permitía la impresión en paralelo, con piezas más estandarizadas o duplicadas. Además, los errores de impresión eran menos críticos ya que si una pieza falla al imprimirse, cuanto más pequeña sea menores son los retrasos.

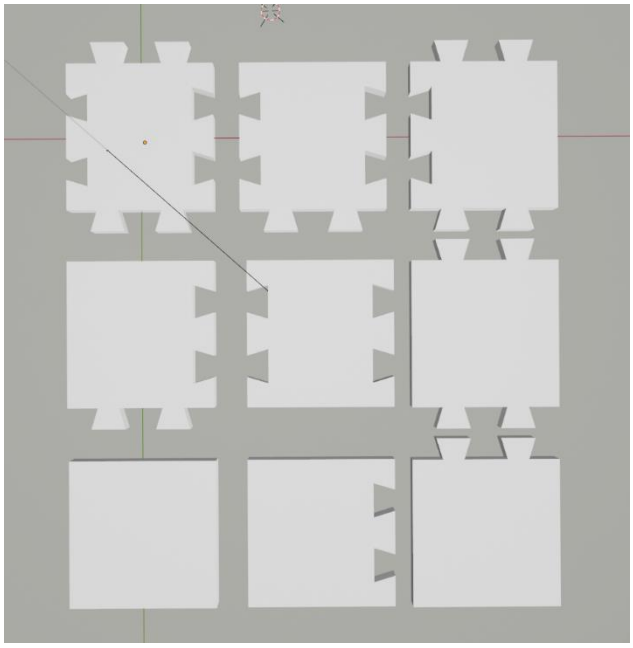


Figura 3. 8. Renderizado de las piezas modulares

Este conjunto de piezas tipo puzle permitió la creación de las mayores secciones de los módulos de una forma mucho más modular. Además, permite la modificación de la carcasa sin el rediseño del módulo en muchas ocasiones. Las piezas tienen un tamaño de 7x7x1cm. Para poder tener cultivos del tamaño de la mayoría de las variedades de lechuga sin restricciones de tamaño, se optó por una superficie base de 4x4 piezas.

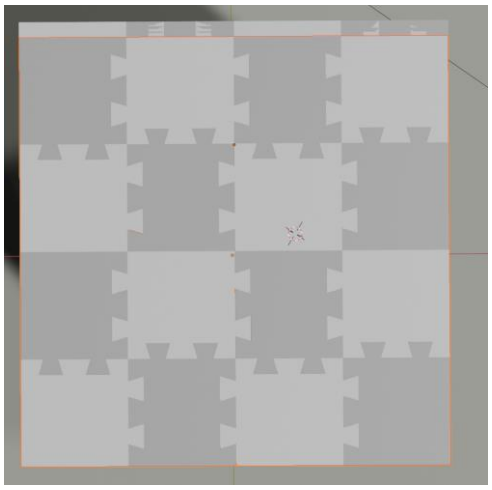


Figura 3. 9. Renderizado ejemplo de estructura modular

Se partió de este primer diseño para comenzar las impresiones. La estructura conceptual se muestra en la siguiente imagen.

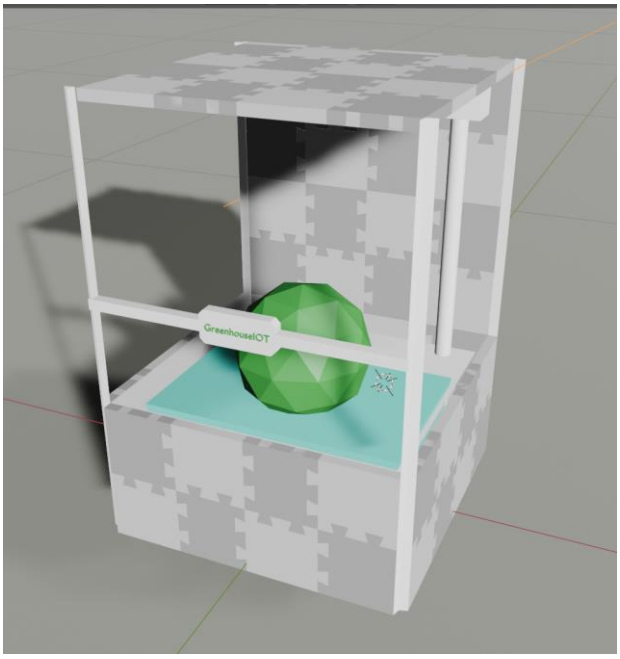


Figura 3. 10. Renderizado del módulo 3D

El módulo final incorpora algunas partes modificadas para dar mayor estabilidad a la estructura como la implementación de pilares mayores.

Figura 3. 11. Imagen del módulo hidropónico

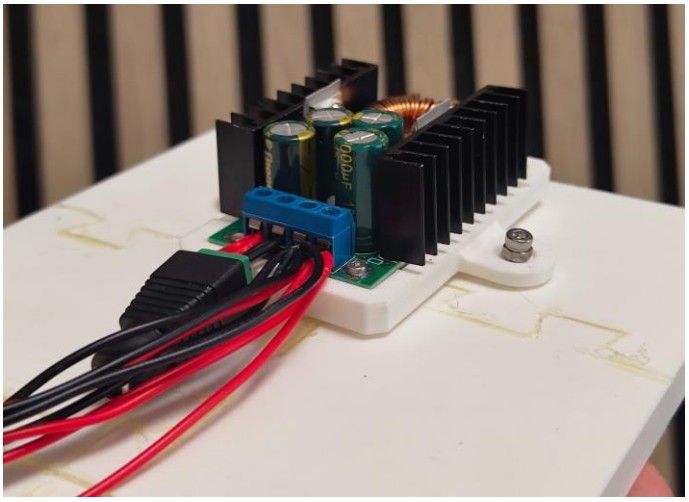


Figura 3. 12. Imagen del regulador de voltaje

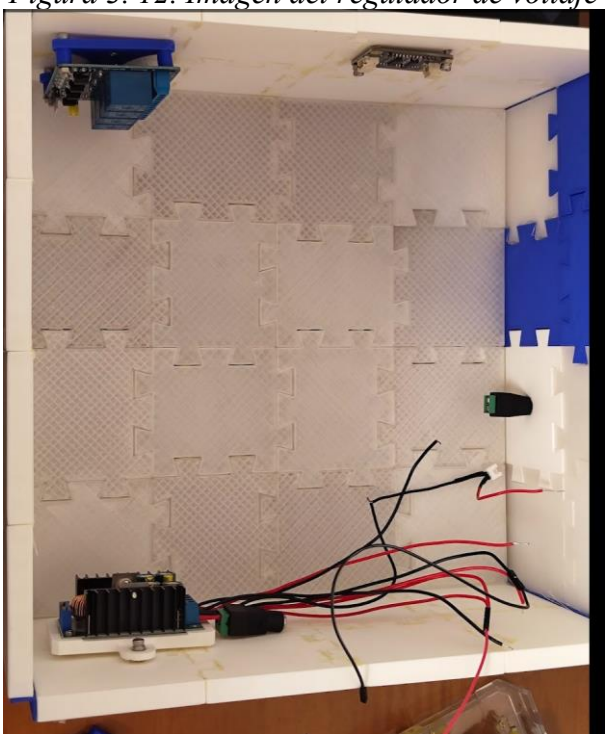


Figura 3. 13. Imagen de construcción

Se cambió el enfoque del funcionamiento del sistema, anteriormente el sistema esperaba a que se solicitase una lectura y posteriormente la enviaba. En esta ocasión se optó por delegar la responsabilidad de envío periódico de datos a los módulos. Si el sistema recibe datos en un margen de tiempo razonable el sistema entiende que el módulo está conectado y está disponible para configuraciones remotas.

Capa de procesamiento y persistencia

La capa de procesamiento siguió el comportamiento explicado en la sección anterior para la lógica de envío de configuraciones. Además, se implementó el soporte para los nuevos sensores y formatos de lecturas.

Con vistas a implementaciones relacionadas con toma de imágenes haciendo uso del nuevo módulo de cámara, se modificó la base de datos para incorporar una tabla dedicada a las lecturas de imágenes.

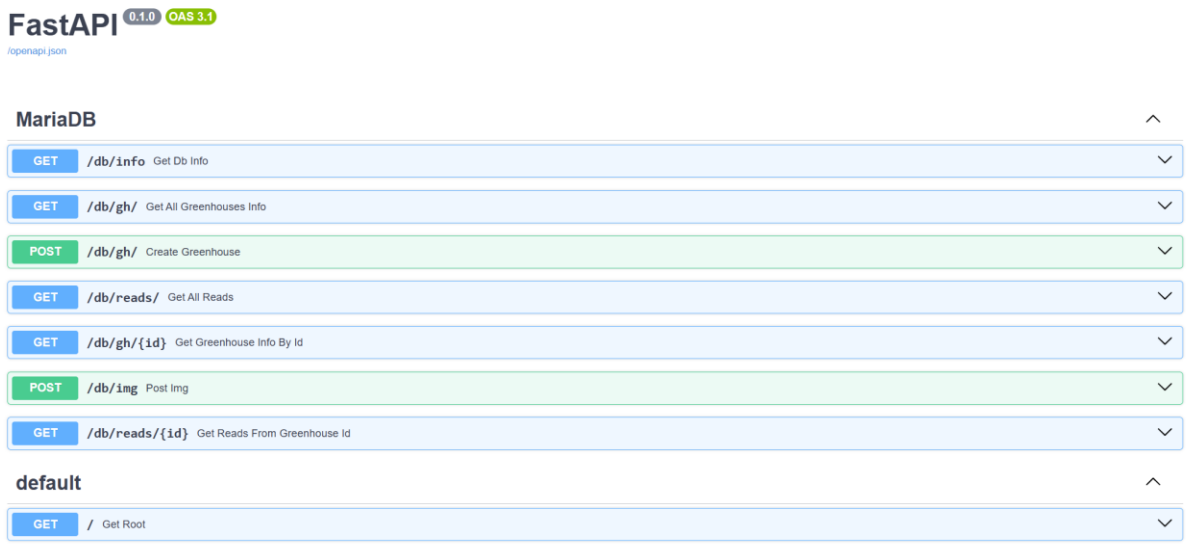


Figura 3. 14. Swagger de la api de la base de datos

Capa de explotación de datos

La capa de explotación de datos sufrió un completo rediseño de arquitectura. Se cambió la tecnología a React funcionando gracias a vite. El lenguaje de desarrollo elegido fue JavaScript. Para el desarrollo de la interfaz se usaron las tecnologías Tailwind y una librería de componentes llamada DaisyUI.

La aplicación cuando llegue a producción debe ser accesible por todos los usuarios y deben por tanto disponer de una cuenta para visualizar solo sus datos y módulos. Para ello se implementó un sistema de autentificación llamado Auth0 que gestiona usuarios y contraseñas. Gracias a que auth0 dispone una capa gratuita, GreenhouseIOT solo debe tener un registro de usuarios y módulos asociados.

Un primer diseño de la interfaz se puede ver en la siguiente imagen.

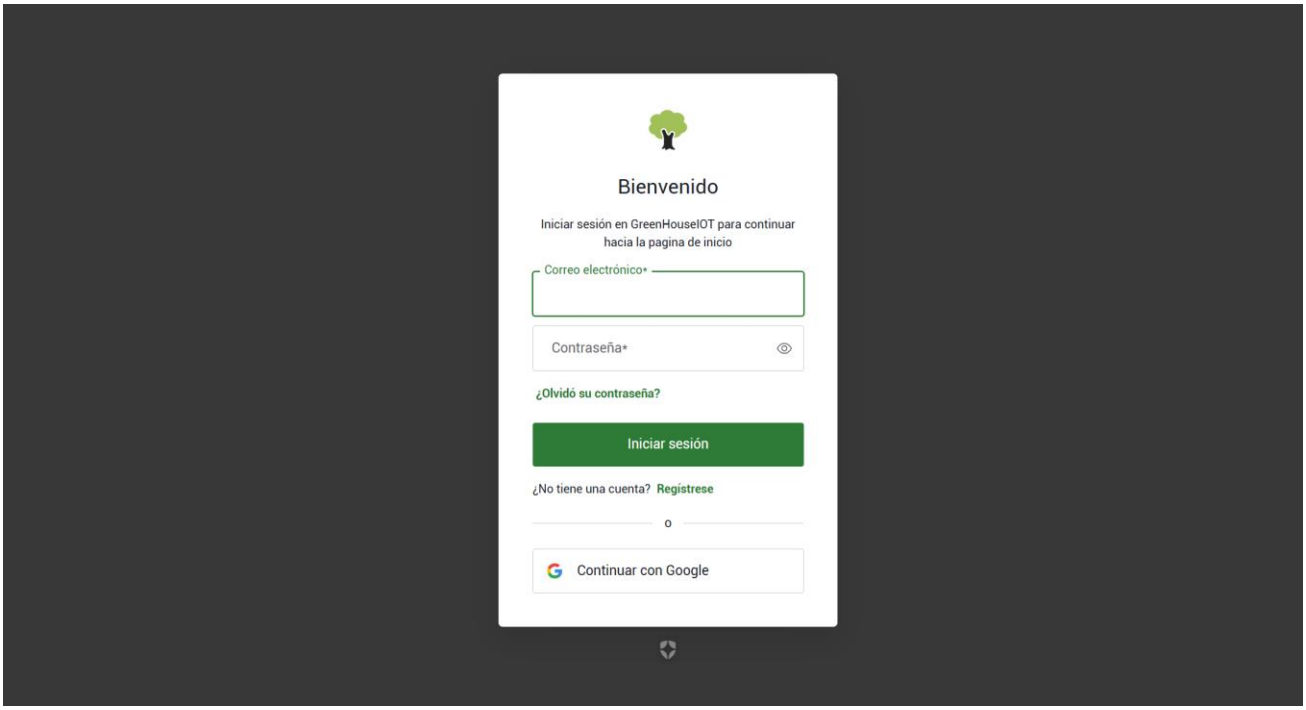


Figura 3. 15. Inicio de sesión en el sistema

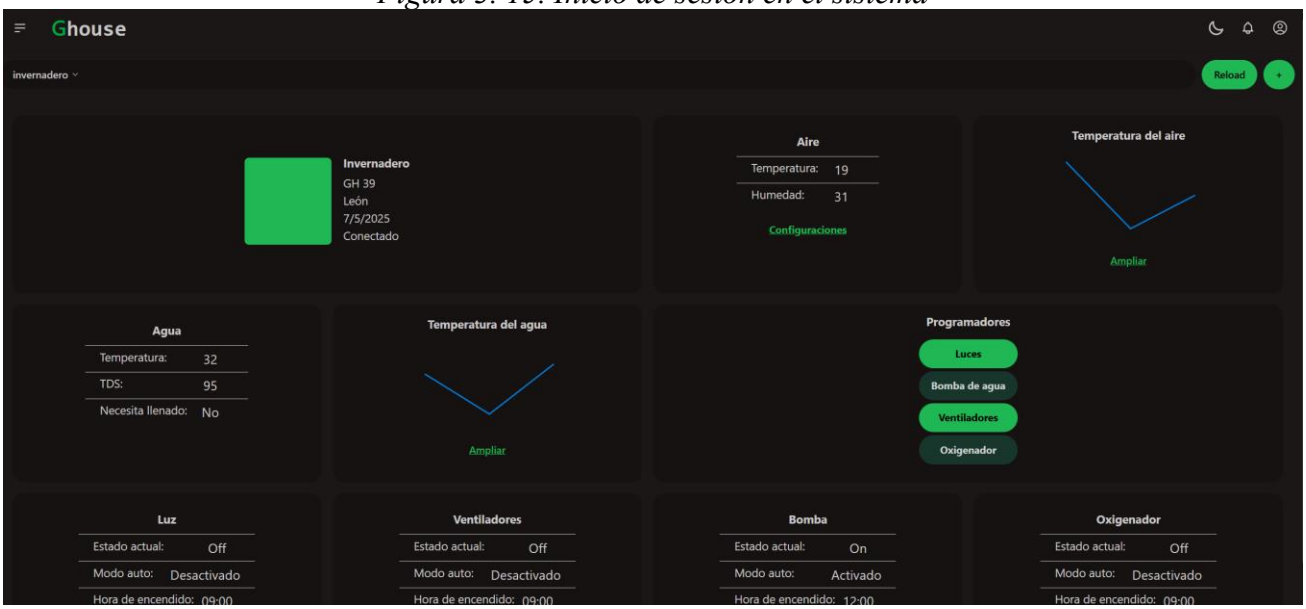


Figura 3. 16. Página de inicio en modo oscuro

Mediante los botones de programación independiente se despliegan formularios con las opciones disponibles.

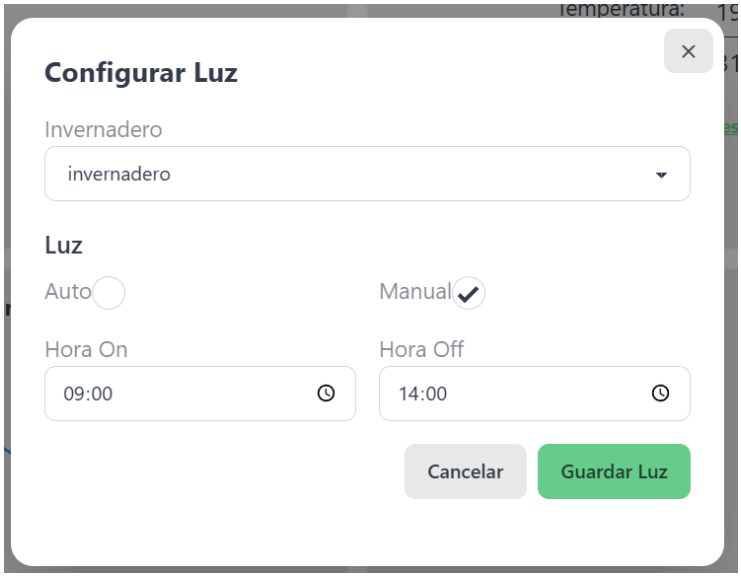


Figura 3. 17 Formulario de configuración de luces

Entre las secciones adicionales encontramos una dedicada a la visualización de gráficas de forma detallada. Se observa la evolución de los parámetros clave a lo largo del tiempo.

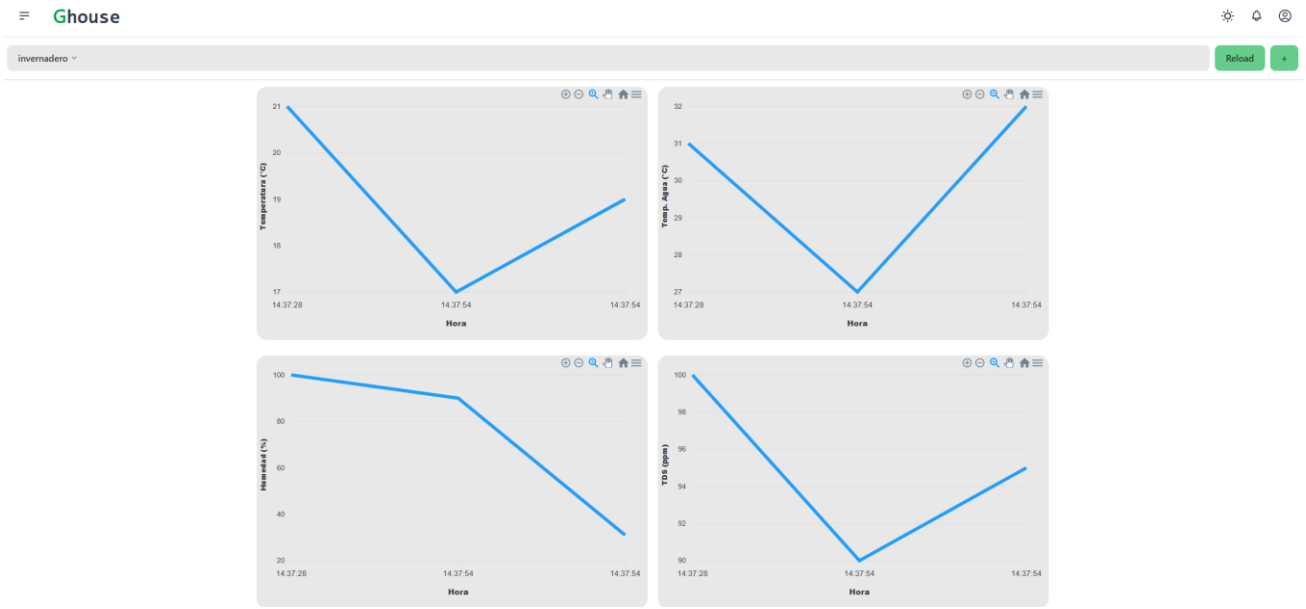


Figura 3. 18 Página de gráficas detalladas

El sistema dispone de un centro de ayuda donde se reúnen ciertas preguntas clave y guías para el usuario. Con esta sección se intenta facilitar al usuario el uso del sistema.

Figura 3. 19. Centro de ayuda para el usuario

3.2.4 Pruebas

Se implementaron pruebas unitarias en el microservicio de procesamiento, asegurando que los datos fuera de rango o en formato inválido no se almacenaran.

Además, se realizaron pruebas de integración entre servicios, mediante el uso de Docker-Compose, validando que los contenedores se comunican correctamente y que los datos se almacenan y se muestran en la interfaz web.

3.3 El producto del desarrollo

El resultado es un sistema funcional desplegable mediante Docker-Compose, capaz de recibir datos de sensores reales, procesarlos, almacenarlos y visualizarlos en una interfaz web moderna. En las secciones anteriores se han incluido capturas de la aplicación desarrollada, así como de todos los componentes que la integran. Este sistema está preparado para escalar tanto en volumen como en funcionalidad, incorporando nuevos tipos de sensores.

Capítulo 4

Evaluación

4.1 Proceso de evaluación

El proceso de evaluación se ha realizado con diferentes estrategias, inicialmente se han realizado pruebas manuales entre entregas bisemanales tratando de comprobar el correcto funcionamiento de las nuevas funcionalidades implementadas. Posteriormente, se aplicaron test específicos para requisitos críticos del sistema y finalmente se evaluó la usabilidad por la metodología de cuestionario con un grupo reducido de usuarios.

4.1.1 Forma de evaluación

La evaluación del sistema desarrollado se ha realizado siguiendo un enfoque integral, que abarca tanto aspectos técnicos como funcionales, y considera distintos niveles del sistema: desde la captura de datos en el hardware hasta la experiencia del usuario en la interfaz web.

El objetivo de esta evaluación es comprobar el correcto funcionamiento, la eficiencia, la estabilidad y la usabilidad del sistema GreenhouseIoT, asegurando que se cumplen los requisitos definidos en la fase de análisis.

A. Criterios de evaluación

La evaluación se ha estructurado en torno a los siguientes criterios clave:

1. **Correctitud funcional**: Verificación de que el sistema cumple las funciones descritas en los requisitos funcionales (recepción de datos, almacenamiento, visualización, automatización, etc.).
2. **Rendimiento**: Análisis de la capacidad del sistema para procesar datos en tiempo real, su tiempo de respuesta y comportamiento ante cargas moderadas.
3. **Fiabilidad y estabilidad**: Pruebas prolongadas para detectar posibles errores, pérdidas de datos o bloqueos del sistema.
4. **Usabilidad**: Evaluación de la interfaz de usuario en términos de claridad, accesibilidad y facilidad de navegación.
5. **Escalabilidad y modularidad**: Comprobación del correcto funcionamiento del sistema al añadir múltiples módulos hidropónicos simultáneamente.

6. **Robustez ante fallos**: Simulación de escenarios de error (fallo de sensores, pérdida de red, datos malformados) para comprobar la tolerancia del sistema.

B. Metodología de evaluación

Para cada uno de los criterios anteriores, se han definido procedimientos específicos que permiten obtener datos objetivos y, en su caso, medir cuantitativamente el rendimiento del sistema:

- Se ha utilizado una plataforma de pruebas local, replicando un entorno doméstico de cultivo, donde se conectó un módulo hidropónico completo (ESP32, sensores, actuadores) a la red local y al sistema completo en contenedores Docker.
- Los microservicios fueron evaluados de forma individual y conjunta, midiendo latencia de respuesta, gestión de errores, y compatibilidad entre APIs.
- Se han desarrollado casos de prueba automatizados para las APIs del backend, validando las operaciones CRUD, la respuesta a entradas no válidas, y el rendimiento bajo múltiples peticiones concurrentes.
- La interfaz de usuario fue probada en distintos navegadores y dispositivos (ordenador de sobremesa, tablet, teléfono móvil) para asegurar adaptabilidad y accesibilidad.
- Se realizaron sesiones de testeo supervisadas con usuarios ajenos al desarrollo (nivel técnico básico), observando posibles bloqueos o dificultades de uso.
- Se emplearon logs detallados y herramientas de monitoreo (como docker stats, curl, y herramientas propias de medición) para recoger métricas durante el uso continuo del sistema.

C. Herramientas utilizadas en la evaluación

- **Postman**: para probar manualmente los endpoints de la API en distintos escenarios.
- **Pytest**: para crear pruebas automatizadas en el backend desarrollado con FastAPI.
- **React Testing Library**: para pruebas funcionales en el frontend.
- **Excel y pandas**: para análisis y visualización de los datos registrados por los sensores durante las pruebas.

D. Escenarios evaluados

A lo largo de las pruebas se reprodujeron múltiples escenarios relevantes:

- **Flujo normal de datos**: sensores enviando valores periódicos durante 48 horas continuas sin interrupciones.
- **Pérdida temporal de conexión WiFi**: reconexión automática del módulo y reanudación del envío de datos.
- **Múltiples módulos activos**: simulación de tres módulos funcionando simultáneamente, con seguimiento independiente.
- **Umbrales superados**: activación automática de ventiladores y alertas por humedad elevada.

- **Carga de datos históricos**: prueba de tiempos de carga en la interfaz al visualizar registros de los últimos 7 días.
- **Errores simulados**: sensores desconectados o mal configurados, recepción de JSON malformado en la API.

4.1.2 Casos de prueba

A. Pruebas del Backend (API y Base de Datos)

ID	Caso de prueba	Entrada	Resultado esperado
B01	Creación de una nueva lectura	JSON válido con datos de temperatura, humedad y TDS	201 Created + objeto insertado correctamente
B02	Lectura de datos por ID	GET `/db/gh/{id}`	Retorno del objeto JSON correspondiente al ID
B03	Envío de datos malformados	JSON incompleto o con tipo de dato erróneo	422 Unprocessable Entity
B04	Solicitud de datos por rango de fechas	GET con parámetros `start_date` y `end_date`	Lista filtrada correctamente
B05	Inserción masiva de lecturas (stress test)	1000 lecturas en lote	Respuesta en < 1s, sin errores

Figura 4. 1. Pruebas del backend

B. Pruebas del Microservicio de Kafka

ID	Caso de prueba	Escenario	Resultado esperado
K01	Producción de datos desde el origen	Sensor envía datos al tópico Kafka	Datos aparecen en el tópico y son recogidos por el consumidor
K02	Interrupción temporal de red	Sensor pierde conexión y la recupera	El servicio se reconecta y reanuda el envío
K03	Datos duplicados en el topic	Doble envío del mismo dato	Solo se almacena una entrada si se detecta duplicado

Figura 4. 2. Pruebas de Kafka

C. Pruebas de Frontend (Interfaz de Usuario)

ID	Caso de prueba	Acción del usuario	Resultado esperado
F01	Acceso a la vista principal	Ingreso en `/dashboard`	Visualización de valores actuales de sensores
F02	Visualización de datos históricos	Selección de un rango de fechas	Gráficas actualizadas correctamente
F03	Acceso desde dispositivos móviles	Navegación desde un smartphone	Interfaz adaptada (responsive)
F04	Creación de regla de automatización	Introducción de valores límite y selección de acción	Regla almacenada y visible en el panel
F05	Fallo en conexión con el backend	Desconexión del contenedor de la API	Mensaje de error visible al usuario

Figura 4. 3. Pruebas de frontend

D. Pruebas de integración entre módulos

ID	Caso de prueba	Interacción	Resultado esperado
I01	Flujo completo de datos	Sensor → API → Kafka → DB → Frontend	Datos visibles en la interfaz en menos de 2 segundos
I02	Uso simultáneo de 3 módulos	Tres sensores activos enviando datos	Módulos identificados correctamente y sin colisión de datos
I03	Inyección de datos erróneos desde sensor simulado	Datos con rango fuera de límites	No se almacenan / se activan alertas

Figura 4. 4. Pruebas de integración

E. Pruebas de robustez y recuperación

ID	Caso de prueba	Condición simulada	Resultado esperado
R01	Reinicio inesperado del contenedor de Kafka	Reinicio manual durante flujo de datos	Reanudación automática sin pérdida de datos
R02	Reinicio de frontend mientras se navega	Recarga forzada del navegador	Persistencia del estado tras reconexión
R03	Pérdida temporal de WiFi en sensor	Desconexión durante 30s	Reconexión automática + reenvío de datos no entregados

Figura 4. 5. Pruebas de robustez

4.2 Análisis de resultados

Para evaluar la interfaz web destinada a los usuarios, se ha utilizado la metodología de evaluación mediante cuestionario. Esta técnica consiste en que un grupo reducido de usuarios, preferentemente imparciales, interactúa con el sistema llevando a cabo una tarea específica. El cuestionario se estructura en tres secciones principales: pre-tarea, tarea y post-tarea.

La sección pre-tarea incluye preguntas generales orientadas a la segmentación de los participantes, lo que permite determinar si el perfil de los encuestados es adecuado para la evaluación del sistema. En la sección de tarea, todos los usuarios deben completar la misma actividad sin recibir asistencia del moderador, lo que facilita la comparación objetiva de resultados. Finalmente, la sección post-tarea recopila las impresiones, dificultades y percepciones de los usuarios sobre la experiencia, permitiendo así identificar puntos fuertes y áreas de mejora en la interfaz evaluada.

La tarea que ha sido asignada a los usuarios es su registro en el sistema, la sincronización de un módulo hidropónico y su configuración de luces. Para ello se ha proporcionado un nombre y código de sincronización a cada participante de forma individual. A continuación, se muestran los resultados obtenidos en la encuesta.

Sección pre-tarea

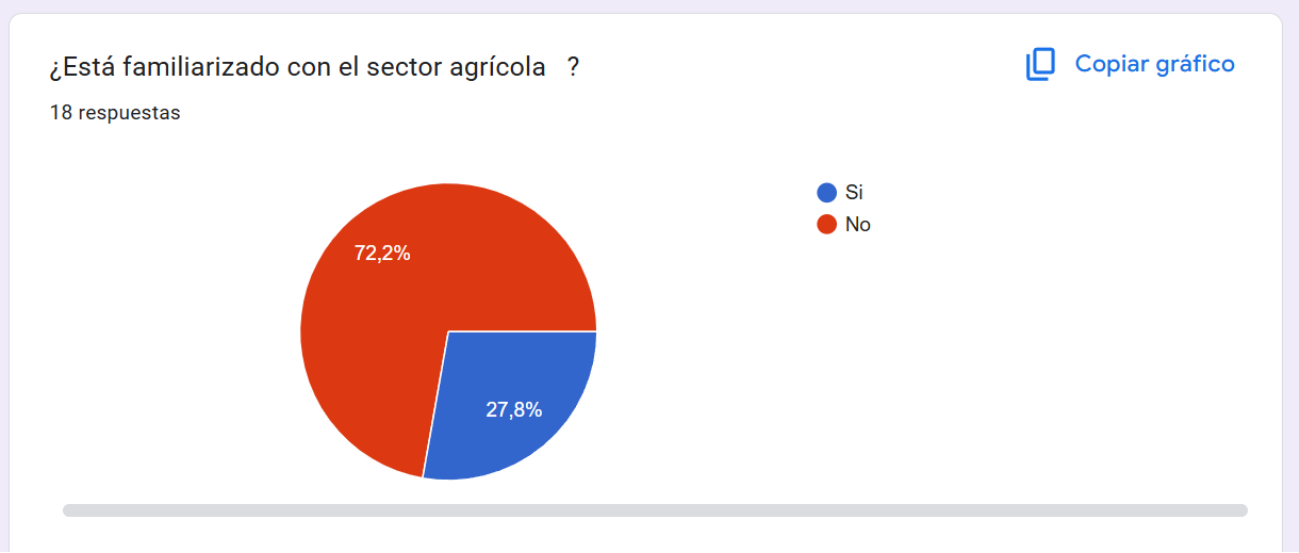


Figura 4. 6. Pregunta de cuestionario sobre sector agrícola

En la primera pregunta observamos que la mayor parte de la población encuestada no está especialmente familiarizada con el sector agrícola, sin embargo, hemos conseguido captar un 27% de usuarios con experiencia en el sector.

Si nuestro objetivo es hacer un producto que permita acercar a las personas al sector agrícola pueden ser buenos resultados.

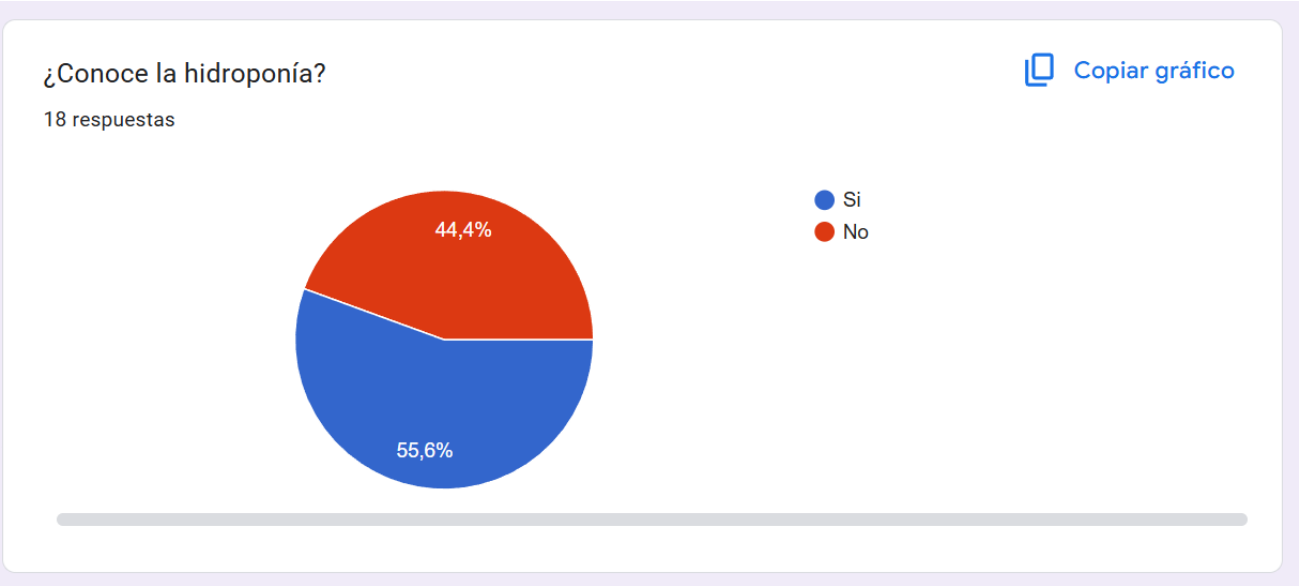


Figura 4. 7. Pregunta de cuestionario sobre hidroponía

La hidroponía es bien conocida por la mayor parte de los encuestados, sin embargo, hay un porcentaje alto de usuarios que no conocen esta modalidad de cultivo. Conociendo este hecho, puede ser buena idea incorporar una sección introductoria para usuarios que no hayan visto ni escuchado nunca que es un módulo hidropónico. Además, si se realizase campañas publicitarias para promocionar este producto sería muy importante enfatizar los beneficios de la hidroponía e invitar a los usuarios a practicar hidroponía de manera accesible.

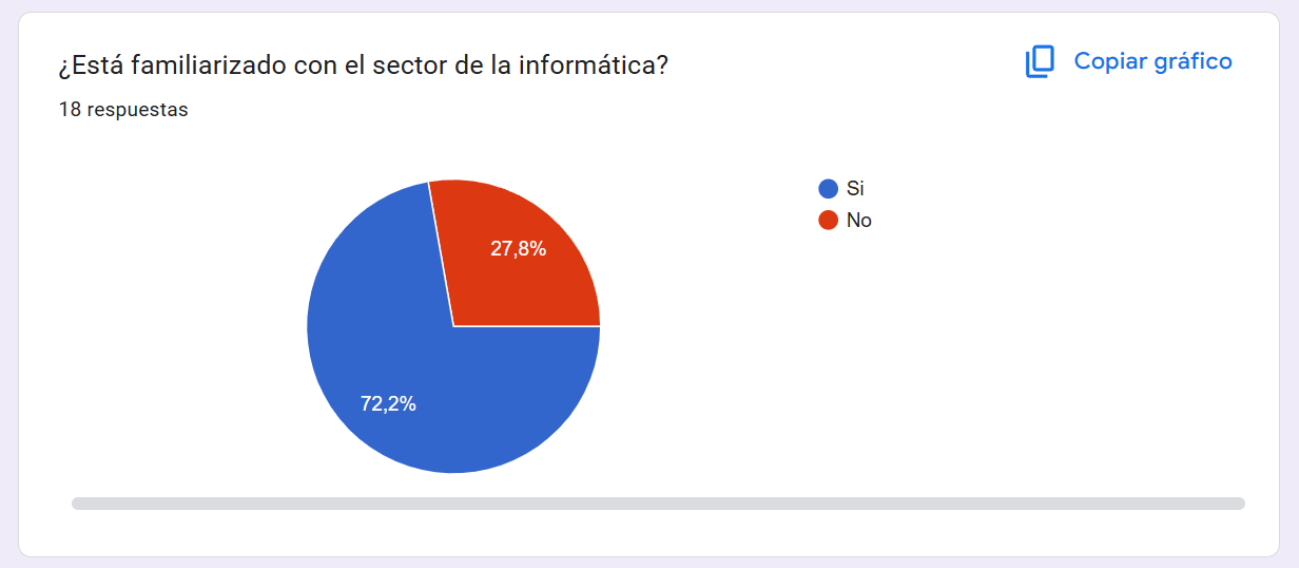


Figura 4. 8. Pregunta de cuestionario sobre sector informático

Observamos que gran parte de los usuarios están relacionados con el sector informático, nos permite valorar la competitividad y robustez de nuestra aplicación al tratar con usuarios mínimamente especializados.

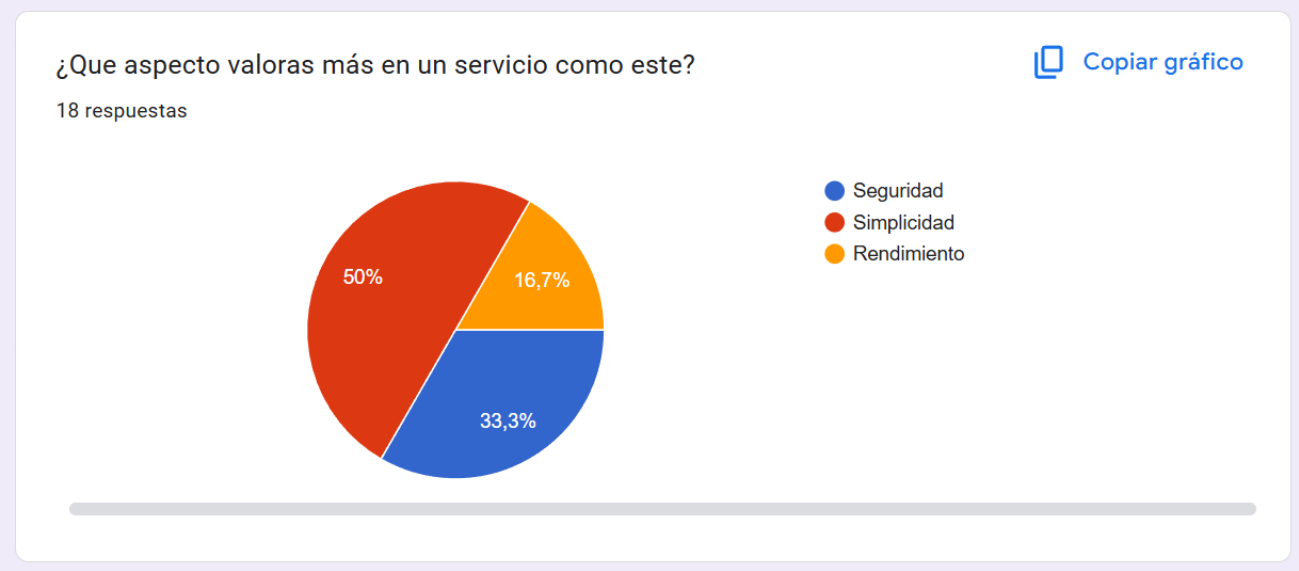


Figura 4. 9Pregunta de cuestionario sobre aspecto a destacar

Observamos que los usuarios valoran la simplicidad del sistema por encima de otras cualidades. Además, observamos que la seguridad es un factor importante para los usuarios.

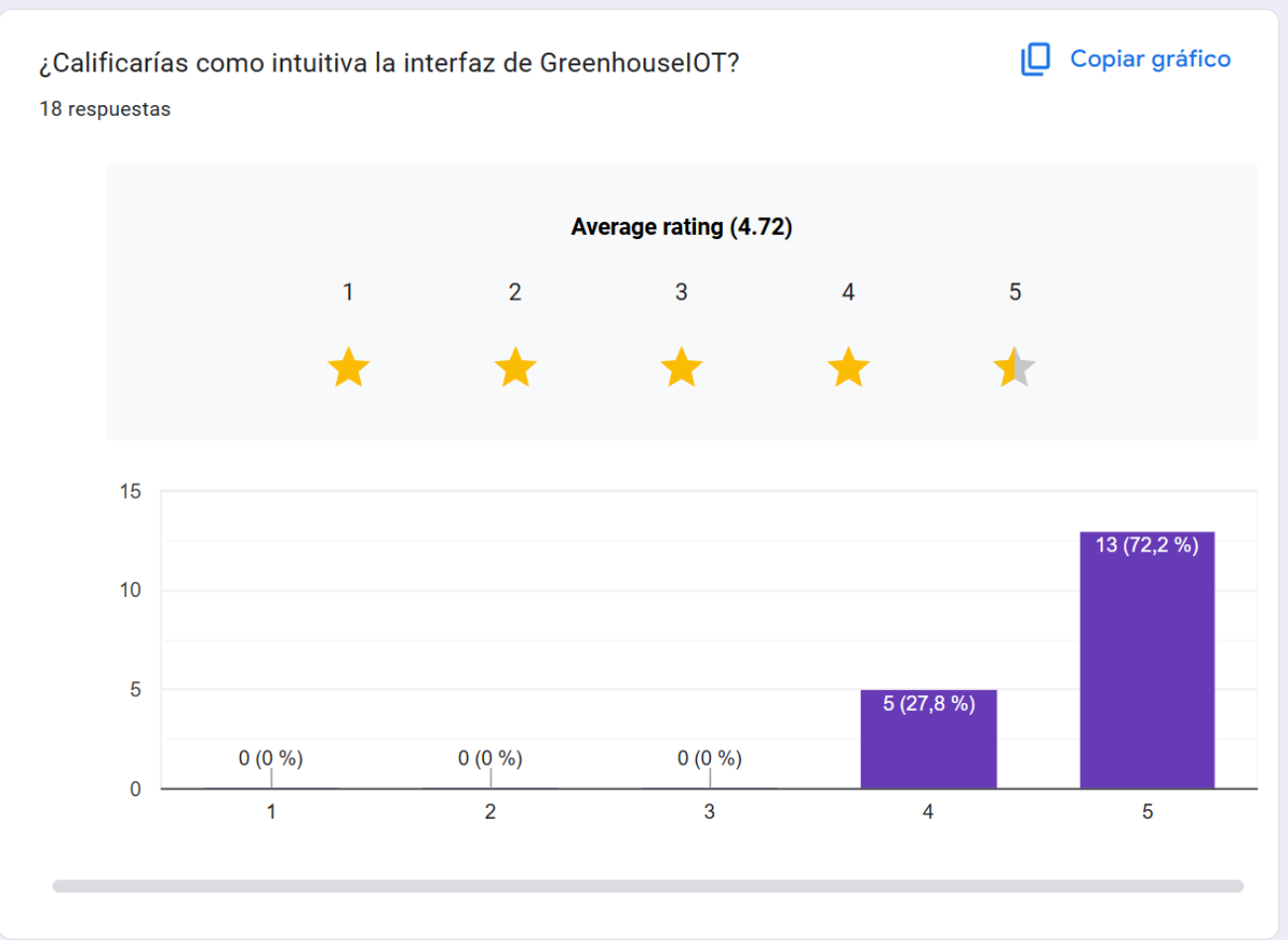


Figura 4. 10 Pregunta de cuestionario sobre interfaz intuitiva

En la calificación de la interfaz intuitiva del sistema encontramos una grata nota de 4.72/5

Los usuarios encuentran sencilla la interfaz en un inicio por lo que estamos consiguiendo ese factor de simplicidad que buscábamos.

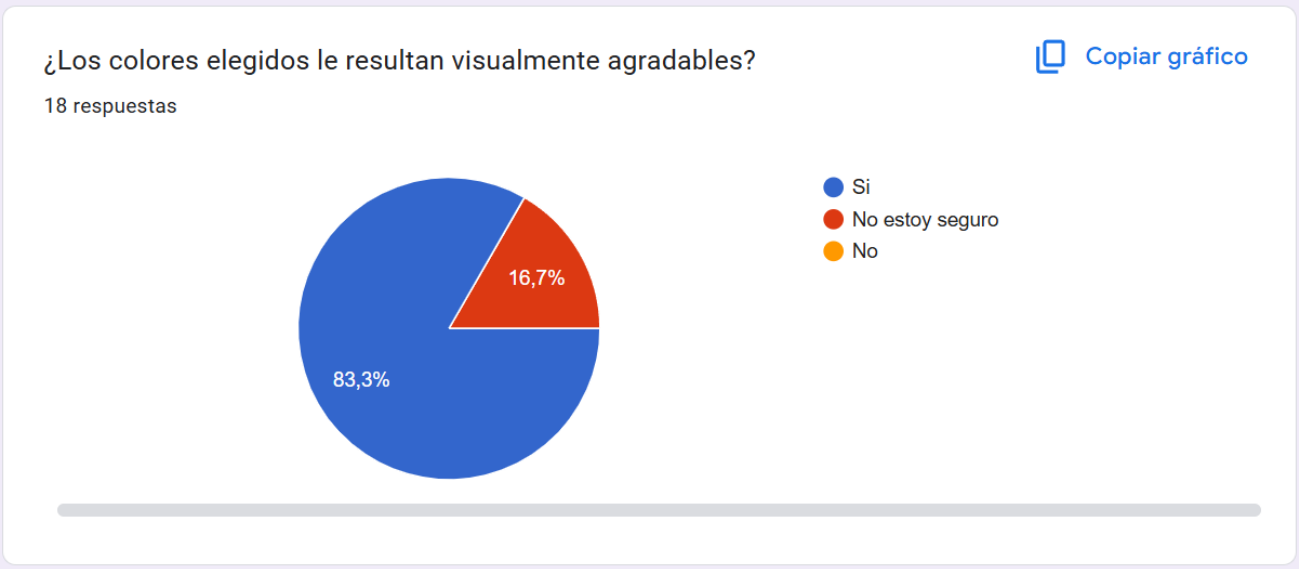


Figura 4. 11.Pregunta de cuestionario sobre la paleta de colores

La mayor parte de los usuarios encuentran agradable los colores elegidos, posiblemente haya sido clave la implementación del sistema de modo claro/oscuro.

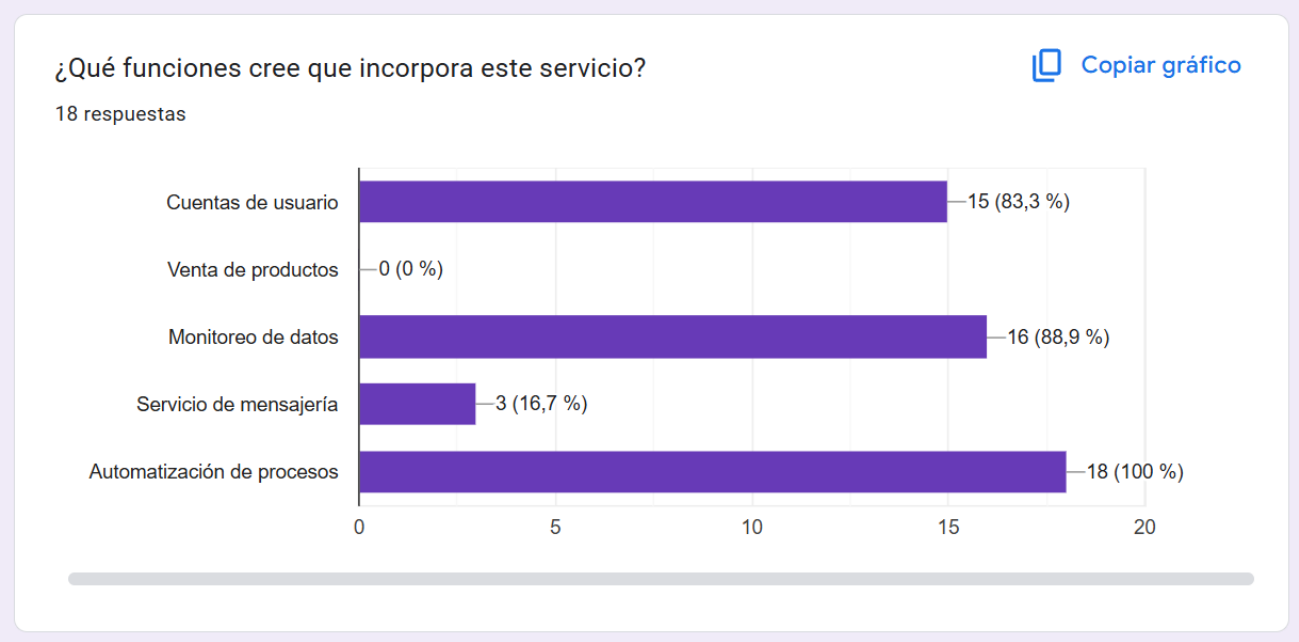


Figura 4. 12.Pregunta de cuestionario sobre funcionalidades

Los usuarios tienen claras las funciones del sistema, sin embargo, encontramos puntuaciones en el servicio de mensajería cuando no es un objetivo principal. Debemos tener presente este hecho, pero puede estar relacionado con la incorporación del sistema de notificaciones en la parte superior izquierda de la interfaz.

Sección post-tarea

Figura 4. 13. Pregunta de cuestionario sobre navegación a la página de registro

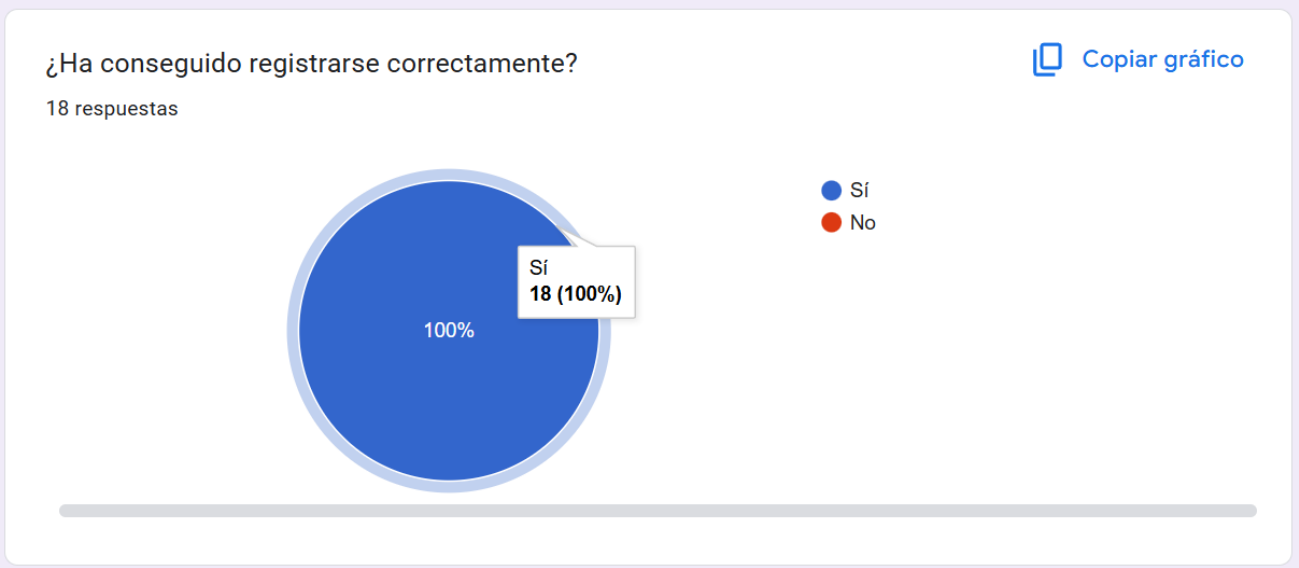


Figura 4. 14. Pregunta de cuestionario sobre proceso de registro

Todos los usuarios han conseguido llegar a la página de registro. Además, han conseguido registrarse en el sistema con éxito. Es una puntuación muy positiva ya que puede ser un punto crítico en aplicaciones similares.

Figura 4. 15. Pregunta de cuestionario sobre puntuación del proceso de registro

La puntuación de navegación al registro ha sido un éxito. Los usuarios están encantados con el sistema de registro.

Figura 4. 16. Pregunta de cuestionario sobre navegación a la sincronización

La sección de sincronización es clara para la mayoría de los usuarios, podemos concluir que es lo suficientemente explícita para la mayor parte de usuarios aunque hay margen de mejora.

Figura 4. 17. Pregunta de cuestionario sobre éxito en sincronización

El proceso de sincronización también ha sido satisfactorio para la mayor parte de usuarios.

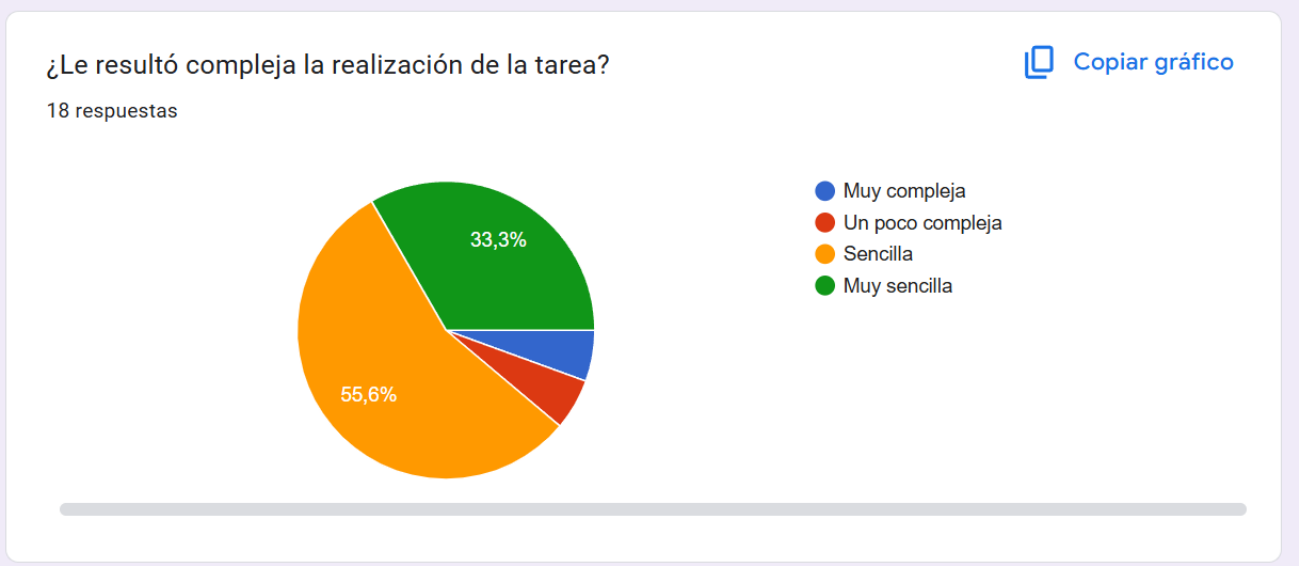


Figura 4. 18. Pregunta de cuestionario sobre la complejidad de la tarea

Los usuarios valoraron como sencilla o muy sencilla la tarea en un porcentaje superior al 85%. Es un muy buen resultado.

Figura 4. 19. Pregunta de cuestionario sobre recomendación

Por todo ello, observamos que los usuarios sí recomendarían el proyecto. Hay usuarios que no están seguros y sería interesante el estudio de sus motivos para mejorar aún más la experiencia de usuario. Es importante destacar que el espacio muestral de la encuesta es muy reducido, pero nos puede ayudar de una manera muy rápida a detectar problemas graves que corregir.

Capítulo 5

Conclusión

5.1 Conclusiones del trabajo

El proyecto GreenhouseIOT ha alcanzado los objetivos planteados, desarrollando una solución integrada para la gestión y automatización de módulos hidropónicos inteligentes a través de una aplicación web.

Entre los principales logros destacan:

- Comprensión y diseño del sistema: Se adquirió un conocimiento detallado sobre los módulos hidropónicos IoT y sus componentes, lo que permitió diseñar un sistema adecuado a sus necesidades.
- Arquitectura basada en microservicios: Se implementó exitosamente una arquitectura modular utilizando Docker, facilitando la escalabilidad, mantenimiento y correcta comunicación entre los servicios.
- Integración de sensores y monitorización: La incorporación de sensores ambientales (temperatura, humedad y nutrientes) posibilitó la monitorización en tiempo real, accesible desde la aplicación web.
- Experiencia de usuario: Se desarrolló una interfaz intuitiva y accesible que permite a los usuarios gestionar y programar acciones automáticas para optimizar el mantenimiento del cultivo.
- Integración hardware-software: Se logró una integración eficiente que garantiza una solución completa y funcional para la gestión del cultivo.
- Fiabilidad y gestión de datos: La implementación de un bus de datos evitó pérdidas de información, mejorando la fiabilidad global del sistema.
- Desarrollo y validación del módulo hidropónico: Se creó un módulo específico para el proyecto, permitiendo recoger datos útiles para el análisis y la mejora continua.
- Análisis y recomendaciones: Los datos obtenidos fueron analizados para ofrecer recomendaciones prácticas que apoyan la toma de decisiones en el cuidado del cultivo.

En conclusión, GreenhouseIOT ofrece una plataforma robusta y escalable que cumple con los objetivos técnicos y funcionales planteados, sentando una base sólida para futuras mejoras y ampliaciones en la agricultura inteligente.

Por otro lado, la realización de este proyecto ha supuesto una serie de retos técnicos y organizativos que han enriquecido considerablemente mi experiencia profesional. Uno de los principales desafíos fue la investigación y comprensión del sector hidropónico IoT, que requirió una revisión exhaustiva de la literatura y de proyectos similares para asegurar que las soluciones adoptadas fueran adecuadas y actuales. Asimismo, el diseño y arquitectura del sistema basado en microservicios implicó aprender a dividir las funcionalidades en componentes independientes, gestionar su comunicación y orquestación dentro de contenedores Docker, lo cual supuso un importante reto técnico.

Además, la integración del hardware con el software fue un aspecto crítico para garantizar la fiabilidad y precisión de los datos recogidos por los sensores, lo que exigió un trabajo minucioso de sincronización y pruebas. Durante el desarrollo, también fue necesario implementar prácticas modernas de desarrollo, como el establecimiento de pipelines de CI/CD, para asegurar una entrega continua, eficiente y con control de calidad.

Gracias a la superación de estos retos, he adquirido un conocimiento integral de todas las fases involucradas en el desarrollo de un servicio digital, desde el diseño del producto hasta la implementación técnica y despliegue final. Este aprendizaje fortalece mis capacidades para futuros desarrollos en entornos tecnológicos complejos y multidisciplinares.

5.2 Futuras líneas de trabajo

El sistema actualmente implementa las principales funcionalidades esperadas en sistemas de su categoría, sin embargo, existen numerosas ampliaciones que podrían agregarse al proyecto para ampliar su alcance y utilidad. Algunas mejoras posibles para futuros desarrollos podrían ser:

- Incorporar comunicaciones directas servidor-módulos haciendo uso del bus de Kafka. Actualmente los módulos recuperan configuraciones de forma periódica de la base de datos, sin embargo, podría implementarse un sistema de actuación más inmediata para casos urgentes.
- Incorporación de un sistema de recopilación de datos anónimos durante el uso de la interfaz web por parte de los usuarios. De esta forma podrían detectarse los puntos clave de la aplicación y descubrir mejoras para hacer la interfaz más accesible.

- Mejoras de seguridad de las APIs. Actualmente la seguridad de las API se centra en tokens secretos encriptados, sin embargo, en caso de quedar expuesto dicho token se podrían producir algunos riesgos de seguridad de importancia moderada. Un generador de tokens dinámicos podría eliminar este problema.
- Implementación de un módulo simulado virtual. Para el uso de la aplicación actualmente se requiere un módulo hidropónico físico. La creación de una simulación de módulo podría permitir la creación de tutoriales guiados para usuarios nuevos.
- Migración del frontend a NextJS con Typescript. Actualemente el frontend está desarrollado con React y JavaScript, la migración mejoraría la integridad durante el desarrollo al incorporar los estándares de programación más usados en la actualidad como el tipado de datos o el enrutado basado en jerarquía de directorios.
- Rango administrador. La incorporación de un rango administrador facilitaría las tareas de moderación de usuarios al permitir la creación de interfaces web dedicadas.
- Incorporación de monitoreo de PH. La creación de una versión más profesional del módulo permitiría incorporar sensores para el monitoreo del pH del agua. La incorporación de dosificadores podría permitir la regulación automática de parámetros y nutrientes.
- Creación de un entorno de testeo previo al despliegue. Permitiría mejorar la integridad de nuevas versiones publicadas.

5.3 Agradecimientos

Me gustaría empezar agradeciendo a mi hermana Clara, mis padres y a mis amigos, por estar ahí en cada etapa de este camino. Gracias por los ánimos cuando las cosas se complicaban, por las palabras de apoyo y por confiar en mí. Sin vuestro cariño y compañía, este proyecto y los años dedicados a mi carrera no habrían sido lo mismo.

Me gustaría agradecer a HPE CDS por darme la oportunidad de formar parte de su equipo durante mis prácticas. Ha sido una experiencia muy valiosa que me ha permitido crecer tanto profesional como personalmente. En especial, gracias a Rubén Rodríguez, por su cercanía, por estar siempre disponible para ayudar y por crear un ambiente de trabajo tan positivo, donde me sentí motivado para tratar de mejorar constantemente.

Finalmente, me gustaría agradecer a Jesús Fernández y Lidia Sánchez por su ayuda durante el desarrollo de mi proyecto. Sus conocimientos y consejos fueron útiles para orientar el trabajo, y gracias a sus revisiones pude pulirlo hasta su etapa final.

Referencias

[1] TerraceLab. https://theterracelab.com/

[2] Green in Blue. https://greeninblue.es/sistemas-de-acuaponia/

[3] Hinojosa Pinto, S. (2019). Diseño de una arquitectura IoT para el control de sistemas hidropónicos. https://riunet.upv.es/handle/10251/127335

[4] Fernández Sánchez-Hermosilla, B. (2024). Implementación de sistema IoT para telemetría y riego automático de sustrato para el cultivo de plantas. (Trabajo Fin de Grado Inédito). Universidad de Sevilla, Sevilla. https://hdl.handle.net/11441/160331

[5] Ávila Guzmán, C. (2018). Diseño y construcción de un sistema hidropónico con IoT adaptable a acuaponía. Universidad de los Andes. Disponible en: https://hdl.handle.net/1992/38769

[6] ReactJS (2024). React – A JavaScript library for building user interfaces. Meta Platforms, Inc. https://reactjs.org

[7] Mariadb Foundation (2024). MariaDB Documentation. https://mariadb.org

[8] FastAPI (2024). FastAPI - Modern Web Framework. https://fastapi.tiangolo.com

[9] Kafka Project Management Committee (2023). Apache Kafka Documentation. Apache Software Foundation. https://kafka.apache.org/documentation/

[10] Docker Inc. (2023). Docker Documentation. https://docs.docker.com

[11] Ansible Project (2024). Ansible Documentation. Red Hat, Inc. https://docs.ansible.com

Anexo 1 - Seguimiento de Proyecto

El seguimiento del proyecto se ha desarrollado mediante reuniones semanales y bisemanales donde se mostraban los avances de varios compañeros en sus proyectos. Se mostraban las funcionalidades implementadas durante el periodo comprendido entre la reunión anterior, se obtenía retroalimentación por parte del coordinador y se establecían los próximos objetivos. Además, se planteaban dudas y el coordinador orientaba a los desarrolladores para abordar los retos.

El periodo final del desarrollo las reuniones pasaron a ser individuales, enfocando principalmente al cierre de los desarrollos satisfaciendo los requisitos de cada proyecto y documentando correctamente el producto final.

Todo el registro de desarrollo queda reflejado en el correspondiente repositorio de código en Github donde se puede ver la evolución completa del proyecto. En el repositorio se agrupan todos los microservicios, el proyecto de despliegue y documentación relativa al proyecto.

El enlace del repositorio de código es el siguiente: https://github.com/jlopep09/GreenHouseIOT

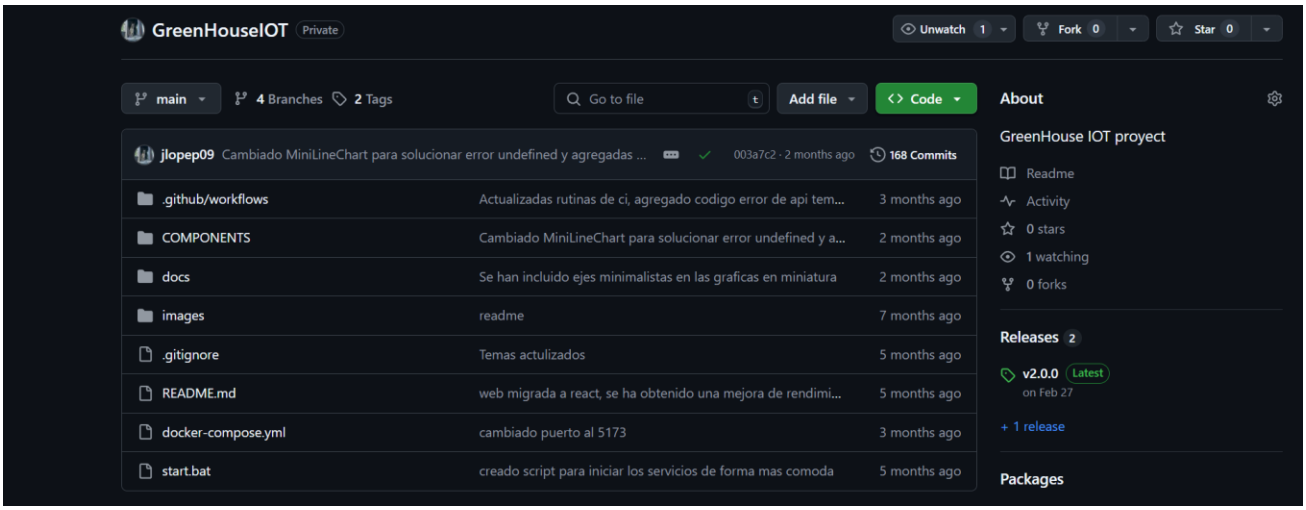


IMAGEN DEL REPOSITORIO DE CÓDIGO

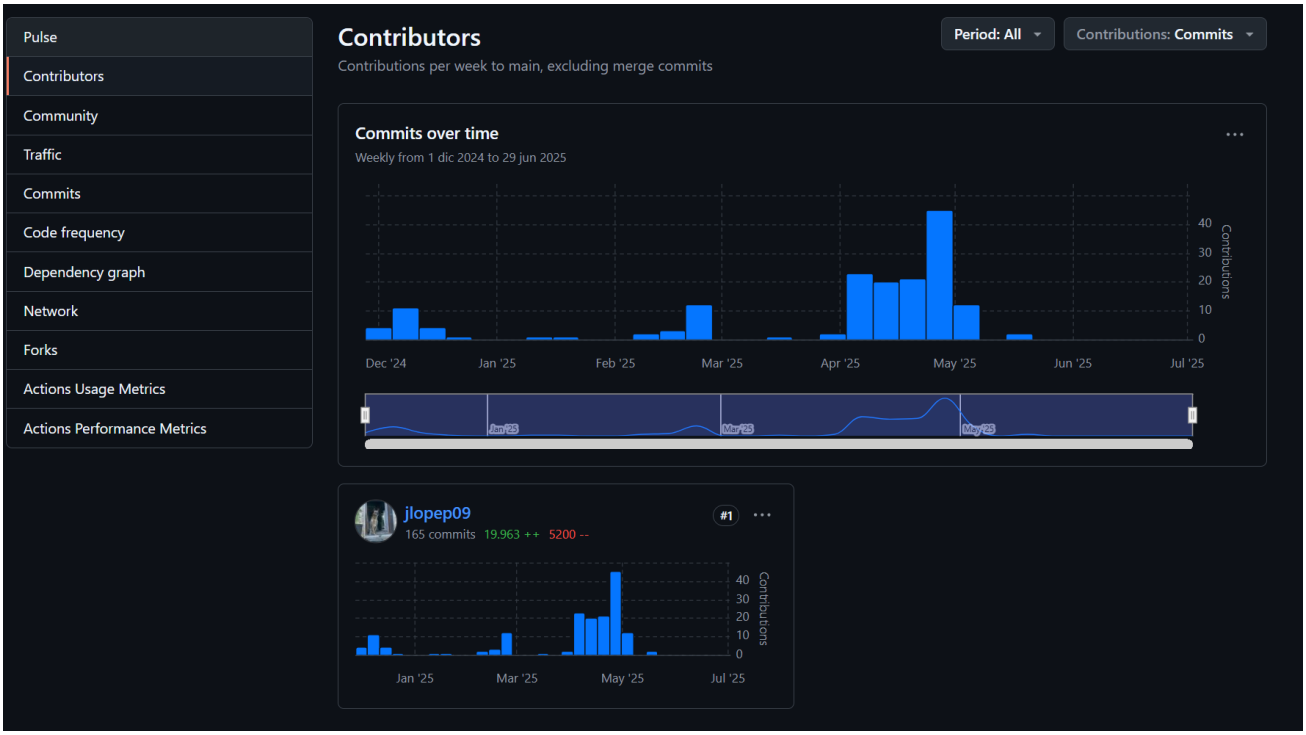


IMAGEN DE LOS CONTRIBUYENTES DEL PROYECTO Y GRÁFICA DE CONTRIBUCIONES POR FECHA.

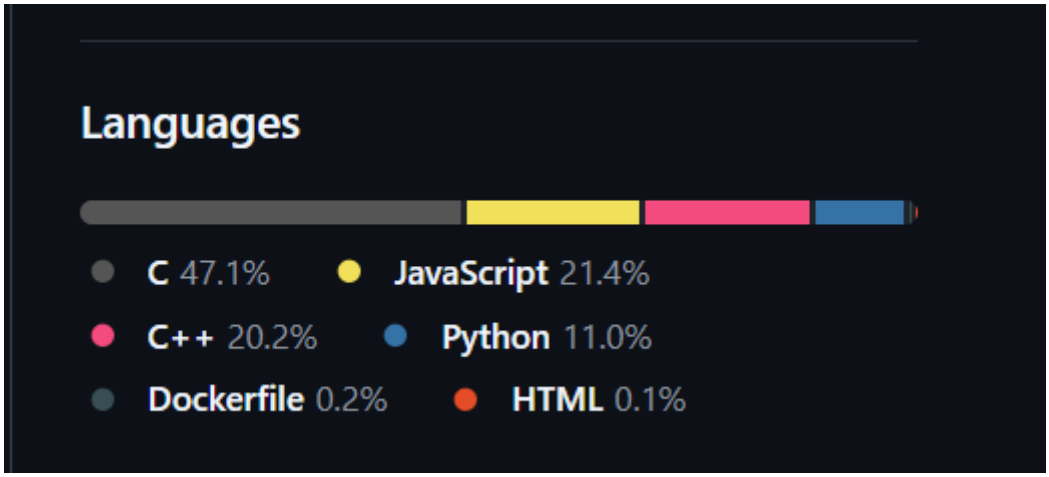


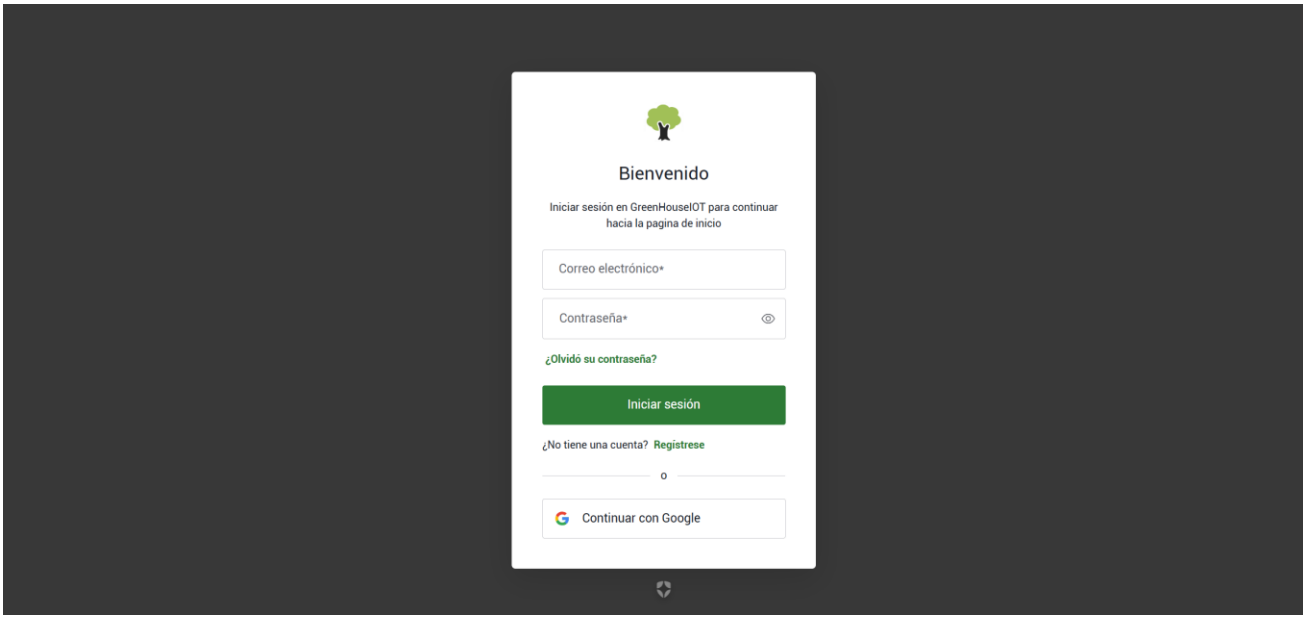
IMAGEN QUE RESUME LOS LENGUAJES UTILIZADOS EN EL PROYECTO.

Anexo 2 - Manual de usuario

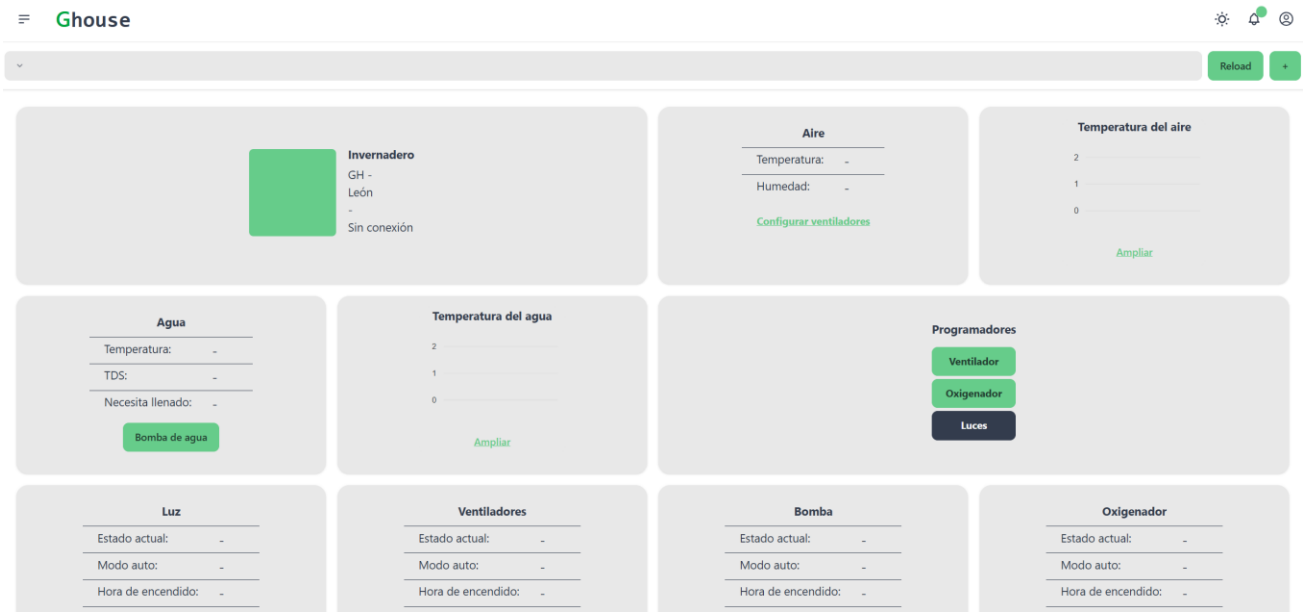
En este anexo se detalla los pasos a seguir para la instalación y uso del sistema por parte del usuario. El primer paso es la adquisición/montaje de un módulo hidropónico. Es requisito necesario la configuración de algunas variables por parte del usuario como las credenciales wifi, la dirección ip local del módulo y diversas configuraciones extra. Una vez encendido todas las configuraciones pueden ser modificadas desde una interfaz web haciendo uso de la ip local del módulo en el puerto 80.

Tras la configuración, se debe encender el servicio de envío de datos en un equipo como un ordenador o una raspberry. Este servicio usa algunas variables de entorno sencillas que pueden modificarse como el servidor donde enviarán los datos. El proceso de configuraciones anteriores puede demorarse pocos minutos si el usuario está habituado al proceso. Con todo ello iniciado, el resto de las acciones se realizarán desde la interfaz web del proyecto, hosteado en el servidor VPS. En este ejemplo se encuentra en la dirección https://vps.joselp.com/

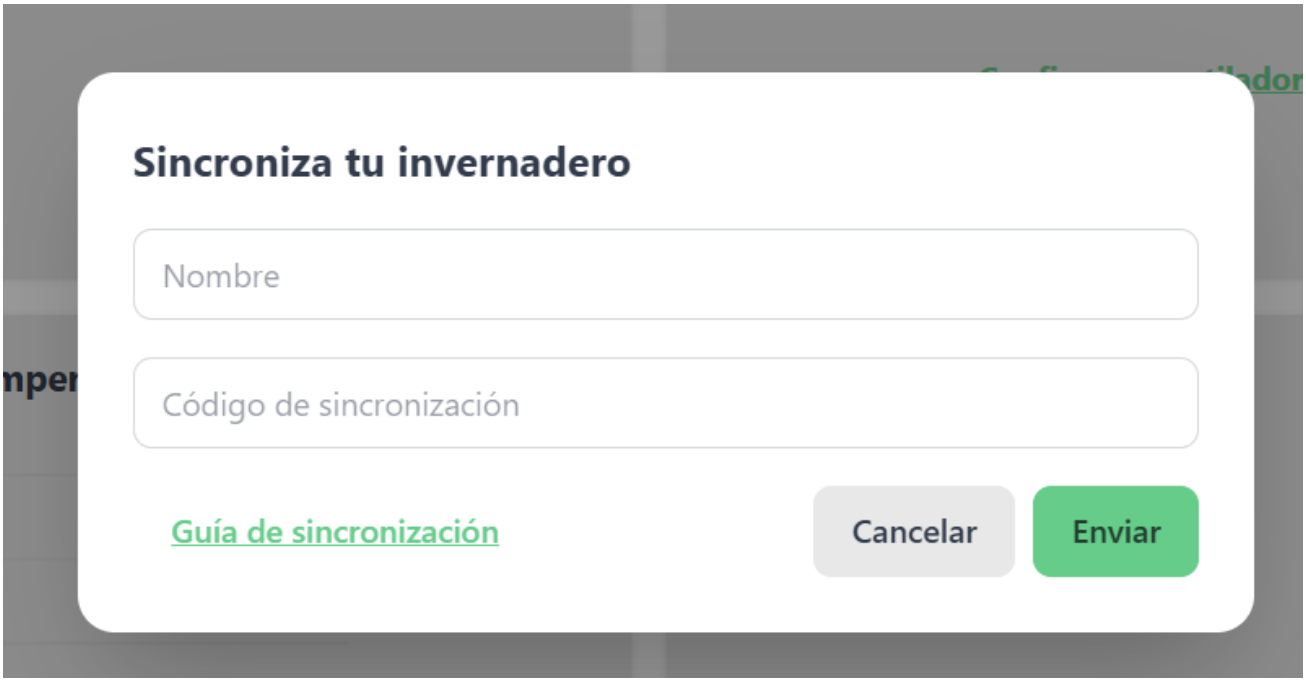
Cuando el usuario accede por primera vez debe registrarse en la aplicación, para ello puede elegir entre usar su cuenta de Google o crear una cuenta específica del servicio.



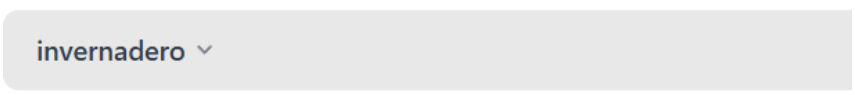
Se encontrará en la página de inicio un menú con información en blanco, ya que no hay ningún módulo sincronizado.



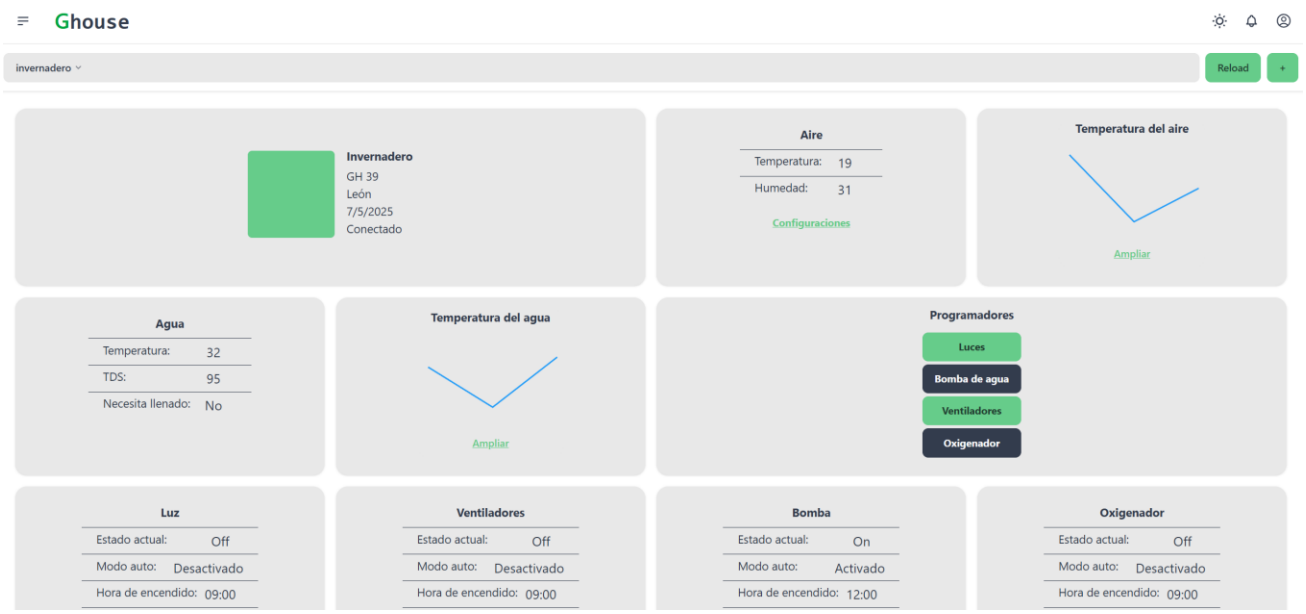
En la sección superior izquierda se encuentra el botón de sincronización representado por el símbolo +. Cuando se acciona el botón aparece un formulario para sincronizar el módulo, el usuario debe introducir las credenciales específicas de su módulo.



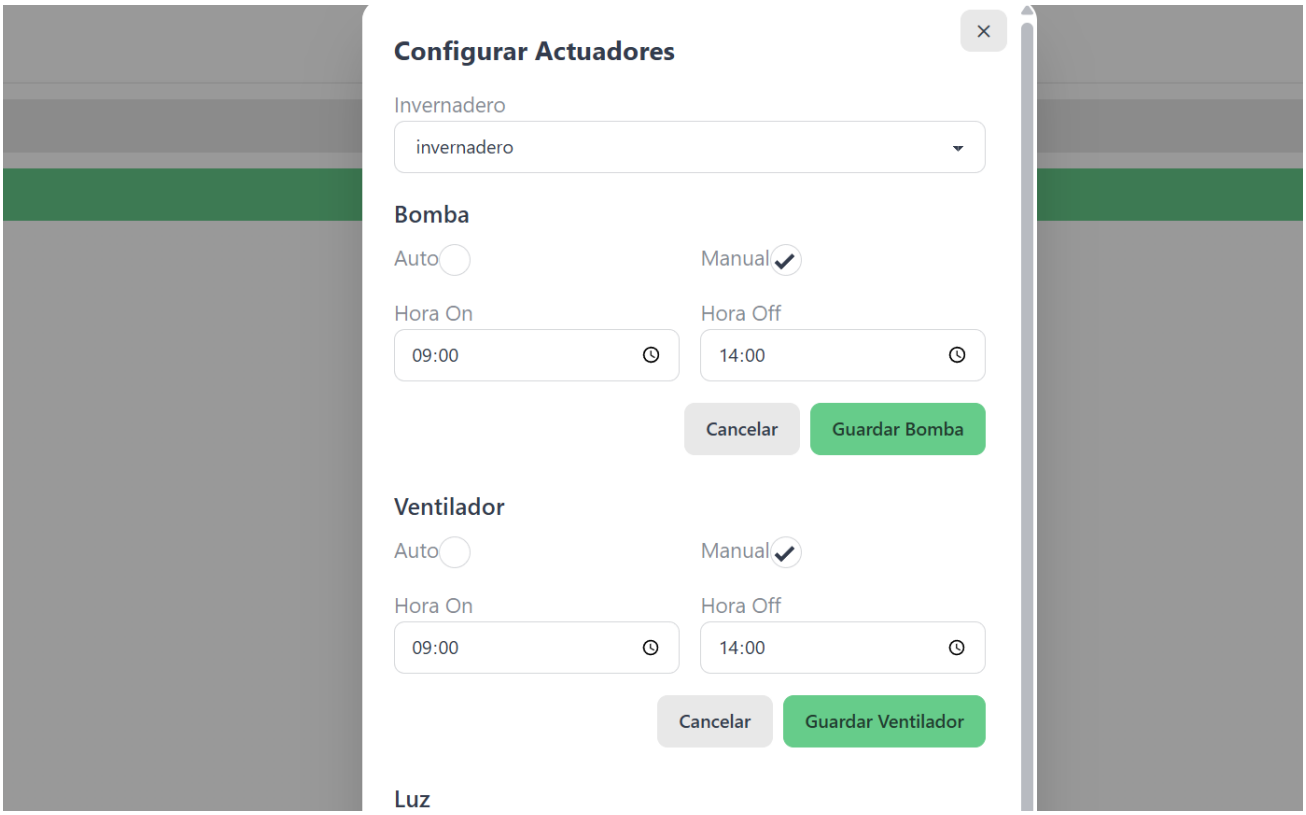
Una vez sincronizado, aparecerá el invernadero en el listado superior.



El contenido disponible del invernadero será mostrado al usuario de forma dinámica.



Las interfaces son intuitivas para que el usuario sepa en todo momento que está viendo/configurando. Además, mediante el menú de navegación puede acceder a secciones como el de configuración para programar sus dispositivos. En la siguiente imagen se muestran algunas configuraciones.



Anexo 3 – Proyecto Ansible

La idea de este proyecto es configurar de forma automatizada un servidor VPS para alojar servicios basados en imágenes Docker. El servidor detectará cuando subes una nueva imagen de tus servicios a DockerHub y hará un nuevo despliegue. El objetivo es configurarlo todo de forma automática en poco tiempo para centrarte en desarrollar el proyecto sin tener que pensar en la infraestructura en exceso. Esta alternativa te permite desplegar tus proyectos de forma más económica respecto a servicios de hosting dedicados al basar su despliegue en un servidor VPS.

Ansible es una herramienta de automatización de TI de código abierto que permite gestionar múltiples servidores y sistemas de forma centralizada. Esta herramienta dispone de una gran cantidad de funcionalidades, pero en este proyecto solo utilizaremos las relacionadas con el despliegue de microservicios Docker y configuraciones básicas.

Ansible se basa en la ejecución de código YAML organizado por tareas dentro de archivos llamados playbooks. Además, permite reunir todos los host destino en un archivo inventory.ini

En nuestro caso, solo hay un host destino.

```
[vps]
vps.joselp.com ansible_user=root ansible_port=22
```

En el playbook principal del proyecto se reúnen las referencias a todos los playbook que agrupan características más concretas, en la siguiente imagen se muestra el orden de ejecución.

```
- import_playbook: playbooks/user.yml
- import_playbook: playbooks/ssh.yml
- import_playbook: playbooks/dependencies.yml
- import_playbook: playbooks/firewall.yml
- import_playbook: playbooks/docker.yml
- import_playbook: playbooks/traefik.yml
- import_playbook: playbooks/greenhouse-db.yml
- import_playbook: playbooks/greenhouse.yml
- import_playbook: playbooks/watchtower.yml
```

Los 4 primeros playbooks se centran en configuraciones más generales, ansible nos automatizará el proceso de creación de un usuario con permisos específicos, configuración de inicio de sesión solo con una clave SSH específica, instalación de dependencias y configuración de un firewall que permita solo los puertos necesarios de nuestro despliegue. Posteriormente se ejecuta el playbook docker.yml,

encargado de su instalación desde el repositorio oficial. Traefik es un servidor proxy inverso que nos permite personalizar y aumentar la seguridad de las comunicaciones. Los siguientes playbooks son los relevantes en nuestra aplicación. Algunas tareas importantes son las siguientes.

```
    - name: Iniciar sesión en DockerHub
      shell: echo "{{ docker_pass }}" | docker login -u "{{ docker_user }}" --password-
stdin
```

En esta tarea se inicia sesión en nuestra cuenta de Dockerhub para poder obtener las imágenes publicadas.

```
- name: Lanzar servicio db (MariaDB)
      docker_container:
        name: mariadb
        image: mariadb:latest
        state: started
        restart_policy: always
        networks:
          - name: mariadb_network
        env:
          MARIADB_ROOT_PASSWORD: "{{ mariadb_root_password }}"
        published_ports:
          - "3306:3306"
```

Se lanza el servicio de MariaDB.

```
    - name: Copiar script SQL al VPS
      copy:
                          src:     /mnt/c/Users/Jose/Desktop/repositoreios/despliegue-
gh/resources/init.sql  # Ruta en tu máquina local
        dest: /tmp/init.sql
      become: yes
      become_method: sudo

    - name: Copiar script SQL al contenedor
      community.docker.docker_container_copy_into:
        container: mariadb
        path: /tmp/init.sql  # Ruta en el VPS
        container_path: /tmp/init.sql

    - name: Ejecutar script SQL dentro del contenedor
       shell: docker exec mariadb mariadb -uroot -p{{ mariadb_root_password }} -e
"source /tmp/init.sql"
      args:
        executable: /bin/bash
```

Para no tener que crear manualmente la estructura de la base de datos, se han creado unas tareas encargadas de copiar y ejecutar un script SQL que crea todas las tablas y elementos.

Una vez tenemos la base de datos, pasaremos a la creación del backend.

```
    - name: Lanzar contenedor de data_processing
      docker_container:
        name: data_processing
        image: jlopep09/greenhouse-backend:latest
        state: started
        restart_policy: always
        networks:
          - name: red_local_compartida
          - name: mariadb_network
          - name: traefik_proxy
        env:
            DB_HOST: "{{ DB_HOST }}"
            DB_USER: "{{ DB_USER }}"
            DB_PASSWORD: "{{ DB_PASSWORD }}"
            DB_NAME: "{{ DB_NAME }}"
            DB_PORT: "{{ DB_PORT }}"
            SECRET_TOKEN: "{{ SECRET_TOKEN }}"
        expose:
          - "8002"
        labels:
          traefik.enable: "true"
          traefik.docker.network: traefik_proxy

          # Enrutado solo desde /api del dominio
                    traefik.http.routers.backend.rule:   "Host(`vps.joselp.com`)   &&
PathPrefix(`/api`)"
          traefik.http.routers.backend.entrypoints: "websecure"
          traefik.http.routers.backend.tls: "true"
          traefik.http.routers.backend.tls.certresolver: "letsencrypt"

          traefik.http.services.backend.loadbalancer.server.port: "8002"
          traefik.http.middlewares.backend-stripprefix.stripprefix.prefixes: "/api"

          traefik.http.routers.backend.middlewares: "backend-stripprefix"
```

Al solo disponer de un servidor vps ha sido necesaria la incorporación del servidor proxy inverso, en la imagen superior observamos que se ha configurado el backend en un router desde el dominio vps.joselp.com/api

Finalmente se crean los contenedores del servicio web y de Kafka. Al querer desplegar de forma pública la aplicación, fue necesaria la incorporación de un sistema de usuarios. Para ello se usó Auth0 con el fin de no almacenar contraseñas en la base de datos. Auth0 está configurado para solo permitir

conexiones por el protocolo https por lo que ha sido necesario configurar el proxy para que solo permita conexiones por este protocolo y nos brinde una certificación SSL/TLS.

```
- name: Lanzar contenedor de web
  docker_container:
    name: web
    image: jlopep09/greenhouse-web:latest
    state: started
    restart_policy: always
    networks:
      - name: red_local_compartida
      - name: traefik_proxy
    env:
      VITE_APP_NAME: "GreenhouseIOT"
      VITE_DDBB_API_IP: "{{ VITE_DDBB_API_IP }}"
      VITE_SECRET_TOKEN: "{{ SECRET_TOKEN }}"

    expose:
      - "5173"
    labels:
      traefik.enable: "true"
      traefik.docker.network: traefik_proxy
      traefik.http.routers.web.rule: "Host(`vps.joselp.com`)"
      traefik.http.routers.web.entrypoints: "web"
      traefik.http.routers.web.middlewares: "https-redirect"



      traefik.http.middlewares.https-redirect.redirectscheme.scheme: "https"

      traefik.http.routers.websecure.rule: "Host(`vps.joselp.com`)"
      traefik.http.routers.websecure.entrypoints: "websecure"
      traefik.http.routers.websecure.tls: "true"
      traefik.http.routers.websecure.tls.certresolver: "letsencrypt"
      traefik.http.services.web.loadbalancer.server.port: "5173"
```

Traefik enruta cualquier petición por el puerto 80 al puerto 443 encriptado por https. Estas peticiones obtienen los datos del servicio Vite ejecutado en el puerto 5173.

Con todo configurado, solo falta recibir datos desde el módulo de invernadero haciendo uso de publicaciones en el topic de Kafka.