



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



**TFG del Grado en Ingeniería
Informática**

**Green In House
Maceta educativa**



Presentado por Oscar Valverde Escobar
en Universidad de Burgos — 6 de julio de 2023
Tutores: Dr. Raúl Marticorena Sánchez
Dr. Antonio J. Canepa Oneto
D. Yeray Pescador Calleja (Técnico de
Emprendimiento UBU)



UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería Informática



D. Raúl Marticorena Sánchez y D. Antonio J. Canepa Oneto, profesores del departamento de Ingeniería Informática del área de Lenguajes y Sistema Informáticos.

Exponen:

Que el alumno D. Oscar Valverde Escobar, con DNI 71309017R, ha realizado el Trabajo Final de Grado en Ingeniería Informática titulado “Green In House - Maceta educativa”.

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección de los que suscriben, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 6 de julio de 2023

Vº. Bº. del Tutor:

Dr. Raúl Marticorena Sánchez

Vº. Bº. del Tutor:

Dr. Antonio J. Canepa Oneto

Resumen

Cultivar una planta es un proceso largo, en el que las acciones no tienen un efecto inmediato, sino que se demoran varias horas o días en hacer efecto. Debido a ello, mantener un control temporal de los factores que afectan al crecimiento y desarrollo de una planta es un factor clave para poder hacerla crecer de manera óptima y evitar así su marchitamiento.

Green In House es un proyecto educativo enfocado principalmente a niños, el cual les permite mantener un registro de los factores principales que influyen en el desarrollo de las plantas, de manera prolongada en el tiempo. Para ello se utilizan sensores que permiten medir estos parámetros esenciales para la supervivencia y buen mantenimiento de una planta.

Como factores más importantes a medir, se han seleccionado la temperatura ambiente, la humedad relativa del ambiente, la humedad relativa de la tierra de la maceta y la luminosidad ambiente.

Green In House no cuenta con actuadores como sistemas de riego automáticos o de iluminación artificial, ya que pretende implicar activamente al niño en el cuidado diario de su planta y hacerle tomar la responsabilidad de mantenerla viva.

Para poder interaccionar de manera sencilla y agradable con Green In House, se proveerá de una aplicación móvil. Dicha aplicación obtendrá a través de conexión WiFi, los datos del servidor integrado en la maceta.

Con esto se consigue que el usuario pueda tener información detallada en el tiempo de los factores que influyen en el desarrollo de las plantas y pueda tomar mejores decisiones a la hora de realizar sus cuidados.

Descriptores

Maceta educativa, IoT, aplicación Android, Raspberry, Flutter, Python, base de datos, API REST.

Abstract

Growing a plant is a long process, where actions do not have an immediate effect, but rather take several hours or days to take effect. Because of this, maintaining a temporal control of the factors that affect the growth and development of a plant is a key factor in being able to grow it optimally and thus prevent its wilting.

Green In House is an educational project mainly aimed at children, which allows them to keep a record of the main factors that influence the development of plants, over an extended period of time. To do this, sensors are used to measure these essential parameters for the survival and good maintenance of a plant.

The most important factors to measure have been selected as the ambient temperature, the relative humidity of the environment, the relative humidity of the potting soil, and the ambient light.

Green In House does not have actuators such as automatic watering or artificial lighting systems, as it aims to actively involve the child in the daily care of their plant and make them take responsibility for keeping it alive.

To interact easily and pleasantly with Green In House, a mobile application will be provided. This application will obtain, through a WiFi connection, the data from the server integrated into the pot.

This allows the user to have detailed information over time about the factors that influence the development of plants and can make better decisions when it comes to their care.

Keywords

Educational pot, IoT, Android app, Raspberry, Flutter, Python, database, REST API.

Índice general

| | |
|---|------------|
| Índice general | iii |
| Índice de figuras | vi |
| Índice de tablas | vii |
| Introducción | 1 |
| 1.1. Materiales adjuntos | 3 |
| Objetivos del proyecto | 4 |
| 2.1. Objetivos sociales | 4 |
| 2.2. Objetivos técnicos | 6 |
| 2.3. Objetivos personales | 8 |
| Conceptos teóricos | 9 |
| 3.1. Fotosíntesis | 9 |
| 3.2. Factores que influyen en el crecimiento de una planta | 11 |
| 3.3. Raspberry Pi: Microcomputador con entradas y salidas | 13 |
| 3.3.1. Raspbian: Sistema operativo de Raspberry Pi | 14 |
| 3.4. Bash: Lenguaje de Programación Shell Linux | 15 |
| 3.4.1. Utilización de Bash en Green In House | 16 |
| 3.5. Python: Lenguaje de programación de aplicaciones de carácter general | 16 |
| 3.5.1. Utilización de Python en Green In House | 17 |
| 3.6. Flutter: lenguaje de programación de aplicaciones multiplataforma | 17 |
| 3.6.1. Dart: el lenguaje de programación de Flutter | 18 |

| | |
|--|-----------|
| 3.6.2. Utilización de Flutter en Green In House | 18 |
| 3.7. API REST: | 19 |
| Técnicas y herramientas | 21 |
| 4.1. Técnicas y metodologías | 21 |
| 4.1.1. Patrones de Diseño | 21 |
| 4.1.2. Scrum: Metodología de gestión ágil | 22 |
| 4.2. Herramientas | 23 |
| 4.2.1. Venv: Entornos Virtuales en Python | 23 |
| 4.2.2. SQLAlchemy: librería de base de datos en Python . . | 24 |
| 4.2.3. OpenAPI: librería de API REST en Python | 25 |
| 4.2.3.1. Microframework Connexion | 25 |
| 4.2.3.2. JSON: Archivos de datos con formato específicos | 25 |
| 4.2.3.2.1. Formato de JSON | 26 |
| 4.2.4. Tkinter: librería de interfaces gráficas en Python . . | 26 |
| 4.2.5. Adafruit - CircuitPython: librería de sensores y actuadores en Python | 27 |
| 4.2.6. Git: sistema de control de versiones | 28 |
| 4.2.6.1. GitHub: Plataforma <i>open source</i> de repositorios | 29 |
| 4.2.6.1.1. Utilización de GitHub en Green In House . | 30 |
| 4.2.7. VSCode: IDE de programación multilenguaje | 30 |
| 4.2.8. Conexión SSH: Control remoto | 31 |
| 4.2.9. FlutterFlow: IDE de programación GUI para Flutter | 32 |
| Aspectos relevantes del desarrollo del proyecto | 34 |
| 5.1. Decisiones de diseño de Green In House | 34 |
| 5.1.1. Decisión de utilizar microcomputador o microcontrolador | 35 |
| 5.1.2. IDE a utilizar | 36 |
| 5.1.3. Respaldo en la nube con control de versiones | 37 |
| 5.1.4. Almacenar datos en ficheros o en base de datos | 37 |
| 5.1.5. Comunicación de datos con sistemas externos | 39 |
| 5.2. Problemas encontrados | 39 |
| 5.2.1. Necesidad de instalar un adaptador WiFi USB | 40 |
| 5.2.2. Necesidad de ingresar las credenciales de la red WiFi . | 40 |
| 5.2.3. Necesidad de incorporar un módulo ADC (<i>analogic digital converter</i>) externo | 41 |
| 5.2.4. Problemas de permisos de ficheros | 41 |
| 5.2.5. Problemas al actualizar el <i>firmware</i> de la Raspberry Pi | 42 |

| | |
|--|-----------|
| 5.2.6. Problemas por la actualización de dependencias internas de librerías utilizadas | 43 |
| 5.2.7. Necesidad de establecer una dirección IP estática . . | 43 |
| 5.2.8. Limitaciones en las gráficas ofrecidas por Flutter Flow | 44 |
| Trabajos relacionados | 45 |
| Conclusiones y Líneas de trabajo futuras | 47 |
| 7.1. Conclusiones | 47 |
| 7.2. Líneas de trabajo futuras | 49 |
| Bibliografía | 51 |

Índice de figuras

| | |
|--|----|
| 3.1. dibujo aclaratorio del ciclo de la fotosíntesis extraído de Wikipedia | 10 |
| 3.2. dibujo aclaratorio de las fases de la fotosíntesis extraído de Wikipedia | 11 |
| 3.3. fotografía de una Raspberry Pi extraída de internet | 13 |
| 3.4. captura de pantalla del escritorio de Raspbian de mi Raspberry | 15 |
| 3.5. captura de pantalla del Swagger de la API REST de Green In House desarrollada con OpenAPI | 20 |
| 4.1. resumen de funcionamiento de metodología Scrum | 23 |
| 4.2. Captura de pantalla de la app gráfica de Green In House desarrollada en TKinter | 27 |

Índice de tablas

| | |
|---|----|
| 4.1. Modelos de sensores utilizados en Green In House | 28 |
| 6.1. Comparativa de Green In House con la competencia | 46 |

Introducción

Green In House es una maceta educativa móvil diseñada para fomentar el aprendizaje de los niños y promover su conexión con la naturaleza. Es un producto que combina la tecnología y la educación sobre el medioambiente de una manera interactiva y práctica.

Green In House se trata de una maceta fabricada con material ecológico (PETG reciclado), el cual se imprime utilizando tecnología de impresión 3D. Green In House no es una simple maceta para plantas, ya que incorpora sensores y una aplicación móvil interactiva para brindar una experiencia de aprendizaje completa a su usuario.

El objetivo de Green In House es mantener un registro temporal de los factores clave que influyen en el crecimiento de las plantas y enseñar a los niños sobre el cuidado de las mismas, concienciándolos sobre la importancia de la naturaleza. A través de esta maceta educativa, los niños pueden aprender a asumir responsabilidades, como regar las plantas y proporcionar un entorno adecuado.

La aplicación móvil complementaria permite a los usuarios acceder a información detallada sobre los factores que afectan al desarrollo de las plantas, recibir y generar consejos de cuidado, facilitando al usuario poder realizar un seguimiento del crecimiento de sus plantas.

Green In House se enfoca en mantener un control sobre aspectos clave para el crecimiento de las plantas, como la temperatura, la humedad y la cantidad de luz. Los sensores incorporados en la maceta educativa permiten a los usuarios monitorizar y ajustar estos factores para garantizar un entorno óptimo para el desarrollo de las plantas.

A continuación se describe la estructura de la memoria y del resto de materiales entregados.

- **Introducción:** breve descripción del problema a resolver y de la solución propuesta. Estructura de la memoria y listado de materiales adjuntos.
- **Objetivos del proyecto:** explicación de los objetivos que persigue el proyecto.
- **Conceptos teóricos:** breve explicación de los conceptos teóricos clave para la comprensión del proyecto.
- **Técnicas y herramientas:** listado de técnicas metodológicas y herramientas utilizadas para gestión y desarrollo del proyecto.
- **Aspectos relevantes del desarrollo:** exposición de aspectos relevantes que hubo que valorar antes y durante la realización del proyecto.
- **Trabajos relacionados:** productos competidores actualmente existentes en el mercado
- **Conclusiones y líneas de trabajo futuras:** conclusiones obtenidas tras la realización del proyecto y posibilidades de mejora de la solución aportada.

Junto a la memoria se proporcionan los siguientes anexos:

- **Plan del proyecto software:** planificación temporal y estudio de viabilidad del proyecto.
- **Especificación de requisitos del software:** se describe la fase de análisis; los objetivos generales, el catálogo de requisitos del sistema y la especificación de requisitos funcionales y no funcionales.
- **Especificación de diseño:** se describe la fase de diseño desglosada en el diseño de datos, el diseño procedimental y el diseño arquitectónico.
- **Manual del programador:** recoge los aspectos más relevantes relacionados con el código fuente de la aplicación (estructura, compilación, instalación, ejecución, pruebas, etc.).
- **Manual de usuario:** guía de usuario para el correcto manejo de la aplicación.

1.1. Materiales adjuntos

Los materiales que se adjuntan con la memoria son:

- Aplicación para Raspberry Pi con scripts de instalación y configuración del entorno y código fuente de la aplicación.
- Aplicación multiplataforma desarrollada en Flutter utilizando Flutter Flow.

Todos estos recursos están accesibles públicamente desde los siguientes repositorios:

- Repositorio del proyecto para la maceta [21].
- Repositorio del proyecto para la App móvil [19].
- Repositorio del proyecto con la documentación [20].

Objetivos del proyecto

Tal y como se ha comentado anteriormente, Green In House es un proyecto que trata de unir el uso de la tecnología y la interacción de las personas con el entorno que nos rodea. Debido a eso, este proyecto persigue tanto objetivos sociales como objetivos tecnológicos, a parte de mis propios objetivos personales, los cuales se describen en los siguientes apartados.

2.1. Objetivos sociales

Green In House es un proyecto que integra tecnología, ecología y educación de una manera práctica e interactiva, brindando a los niños la oportunidad de aprender sobre la importancia de cuidar el medio ambiente y asumir responsabilidades. Por lo tanto, Green In House es un instrumento de aprendizaje que promueve la comprensión de conceptos científicos como la fotosíntesis, la biología de las plantas y la interacción con el medio ambiente. Los sensores incorporados en la maceta ayudan a los niños a entender cómo diferentes factores como la luz, la humedad y la temperatura afectan el crecimiento de las plantas.

Además, la integración con una aplicación móvil proporciona una experiencia de aprendizaje más completa. La aplicación ofrece a los niños acceso a información detallada sobre las plantas, consejos de cuidados y la capacidad de hacer un seguimiento del crecimiento de sus plantas en base a los factores ambientales a los que están sometidas las plantas. Debido a esto, Green In House puede ser una herramienta educativa muy valiosa, ya que puede ayudar al aprendizaje infantil de manera significativa, vivencial y con aporte

al aprendizaje académico de varias formas, a la vez que desarrolla nuevas habilidades transversales gracias a la motivación en distintos aspectos como:

- **Aprendizaje significativo:** Green In House proporciona a los niños una experiencia práctica y tangible sobre el cuidado de las plantas. Esto permite a los niños establecer conexiones significativas entre los conceptos teóricos y la realidad. Al hacerles participar activamente en el proceso de crecimiento de las plantas, los niños pueden comprender mejor los conceptos relacionados con la biología, la ecología, la fotosíntesis y otros temas relacionados.
- **Aprendizaje vivencial:** mediante la obligación de los niños a interactuar directamente con las plantas y experimentar las consecuencias de sus acciones, Green In House consigue hacer que aprendan observando el crecimiento de las plantas, experimentando con diferentes condiciones ambientales y tomando decisiones sobre el cuidado de las mismas. Esta experiencia vivencial refuerza el aprendizaje.
- **Apporte al aprendizaje académico:** Green In House se puede integrar en el ámbito académico, complementando y enriqueciendo las lecciones tradicionales. Las actividades relacionadas con las plantas pueden ser utilizadas para abordar diferentes áreas del conocimiento, como ciencias naturales, lenguaje (a través de la comunicación y expresión de ideas sobre las plantas), arte (mediante la creación de proyectos artísticos inspirados en la naturaleza),... Esto amplía el alcance del aprendizaje académico y lo hace más relevante para los niños.
- **Desarrollo de habilidades transversales:** Green In House promueve el desarrollo de habilidades como la responsabilidad, el pensamiento crítico, la resolución de problemas, la toma de decisiones y la autonomía. Los niños aprenden a cuidar de las plantas, a tomar decisiones basadas en la observación y a solucionar problemas relacionados con el crecimiento de las mismas. Estas habilidades se pueden extraer a otras áreas de su vida y contribuyen a su desarrollo integral.
- **Motivación y compromiso:** mediante su enfoque lúdico y práctico, Green In House pretende capturar el interés de los niños y motivarlos a participar activamente en el proceso. Al estar involucrados en el cuidado de las plantas, los niños se sentirán más comprometidos y conectados con el proceso de aprendizaje, lo que favorece un mayor nivel de retención y comprensión de los contenidos.

Green In House, a pesar de estar basado en la tecnología e incorporar el uso de sensores y dispositivos móviles, está diseñado para promover la interacción con el mundo real y mejorar el desarrollo cognitivo de los niños. Algunas razones por las cuales este proyecto podría tener un impacto positivo, en lugar de fomentar una dependencia negativa de la tecnología, son las siguientes:

- **Interacción con el mundo real:** aunque Green In House usa sensores y una aplicación móvil, la totalidad de las acciones necesarias para el cuidado de la planta se realizan de manera manual. Los niños tienen que regar la planta, colocarla en un lugar donde reciba suficiente luz y monitorizar los parámetros del ambiente físico. Esto significa que pasan un tiempo significativo interactuando con el mundo real, no solo con la tecnología.
- **Consecuencias tangibles:** a diferencia de la mayoría de videojuegos, donde las acciones no suelen tener consecuencias reales (a excepción de realizar compras con dinero real, lo cual afecta directamente a sus cuentas de crédito, o en este caso, a las de sus padres), en Green In House las decisiones y acciones de los niños tienen resultados tangibles. Si olvidan regar su planta, verán cómo se marchita. Esto puede ayudarles a comprender que sus acciones tienen consecuencias en el mundo real.
- **Fomenta la responsabilidad:** cuidar regularmente de una planta puede enseñar a los niños sobre la responsabilidad y la gestión del tiempo. Tienen que recordar regar la planta y comprobar sus condiciones regularmente. Este es un hábito que puede extrapolarse a otras áreas de sus vidas.
- **Uso de la tecnología con un determinado propósito:** Green In House no promueve el uso de la tecnología por el simple hecho de usarla. En este proyecto la tecnología se utiliza como una herramienta para aprender y cuidar de un organismo vivo. Esto puede ayudar a los niños a entender que la tecnología tiene muchos usos útiles y beneficiosos, más allá del entretenimiento.

2.2. **Objetivos técnicos**

A continuación se describen los objetivos técnicos cubiertos en el proyecto:

- Generar y mantener un registro temporal de datos de temperatura, luminosidad y humedad ambiente, así como la humedad de la maceta. Estos son los factores principales de influencia en el desarrollo de una planta en los que se centrará Green In House.
- Utilizar Git como sistema de control de versiones junto con una plataforma de repositorios *Open Source*.
- Crear uno o varios servidores en una Raspberry Pi capaz de desplegar y controlar la aplicación completa de Green In House.
- Generar una serie de *scripts* en Bash que permitan realizar fácilmente la instalación de la aplicación en cualquier Raspberry Pi y su iniciación automática al encenderla.
- Leer diferentes sensores electrónicos que miden factores ambientales de temperatura, humedad y luminosidad.
- Desarrollar una base de datos que almacene toda la información recogida de los sensores y la interrelacione con la planta a la que dichos sensores están asociados a lo largo de diferentes periodos temporales y a los consejos asociados a dicha planta.
- Generar una estructura modular en la que se puedan incorporar fácilmente y de manera dinámica, nuevos sensores, plantas, tipos de plantas y consejos de mantenimiento, así como definir interrelaciones entre ellos, sin necesidad de modificar el código de la aplicación.
- Desarrollar una aplicación para generar una interfaz gráfica que permita realizar determinadas acciones de control de la maceta desde una pantalla táctil incorporada a la misma.
- Implementar un sistema API REST capaz de comunicar datos entre el servidor alojado en la Raspberry Pi y aplicaciones externas que hagan uso de ella.
- Desarrollar una aplicación multiplataforma en Flutter capaz de ser desplegada en diferentes plataformas como Android, iOS,... y que permita leer los registros de los sensores y graficarlos, así como generar nuevas plantas, incorporar sensores, modificar los consejos, etc.
- Utilizar patrones de diseño para mejorar la arquitectura y facilitar el diseño y entendimiento del código. Emplear un sistema MVC (Modelo-Vista-Controlador) definido por capas.
- Aplicar metodología ágil en el desarrollo del software

2.3. Objetivos personales

A continuación se describen mis objetivos personales que buscaba cubrir con la realización de este proyecto:

- Investigar diferentes tecnologías existentes utilizadas comúnmente para desarrollar las tareas de:
 - Gestión de bases de datos en Raspberry Pi
 - Comunicación entre sistemas mediante API REST en Raspberry Pi
 - Generación de aplicaciones de interfaz gráfica para Raspberry Pi.
 - Realizar lecturas de diferentes sensores electrónicos desde una Raspberry Pi.
 - Generación mediante Flutter de aplicaciones móviles con comunicación con otros sistemas mediante API REST.
- Aportar un enfoque lúdico al cuidados de plantas mediante el uso de nuevas tecnologías.

Conceptos teóricos

Green In House es un proyecto que combina diversas tecnologías para facilitar el cuidado de las plantas, por lo que es necesario tener claro tanto los conceptos teóricos referentes a los factores que influyen en el crecimiento de las plantas, como los conceptos teóricos de las diferentes tecnologías involucradas en el sistema.

3.1. Fotosíntesis

La fotosíntesis [25] es un proceso vital que ocurre en las plantas y algunos organismos como las algas y algunas bacterias. Como se muestra en la Figura 3.1, es el proceso mediante el cual los organismos capturan la energía del sol y la convierten en energía química, almacenándola en forma de compuestos orgánicos, como los azúcares. Esta energía química es esencial para el crecimiento y desarrollo de los seres vivos y también para mantener el equilibrio ecológico en nuestro planeta.

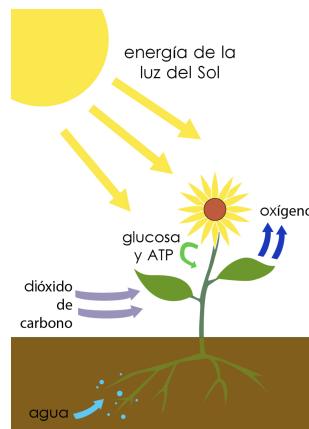


Figura 3.1: dibujo aclaratorio del ciclo de la fotosíntesis extraído de Wikipedia

La fotosíntesis es un proceso fundamental para la vida en la Tierra, ya que las plantas son la base de la cadena alimentaria. Los organismos heterótrofos, como los animales, obtienen la energía necesaria al consumir plantas u otros organismos que se alimentan de ellas. Además, la fotosíntesis tiene un papel crucial en la producción de oxígeno, ya que durante el proceso se libera este gas que es vital para la respiración de los seres vivos.

La fotosíntesis se lleva a cabo en los cloroplastos, orgánulos presentes en las células de las plantas. Estos contienen un pigmento verde llamado clorofila, el cual es el responsable de capturar la energía lumínica del sol. Como se muestra en la Figura 3.2, el proceso de fotosíntesis se puede dividir en dos etapas:

- **Fase luminosa [23]:** la energía lumínica del sol es absorbida por la clorofila y se utiliza para dividir las moléculas de agua en oxígeno y protones. Este proceso libera energía en forma de ATP (adenosín trifosfato), el cual es una molécula portadora de energía. Además, se genera NADPH (nicotinamida adenina dinucleótido fosfato reducido), la cual es otra molécula portadora de energía.
- **Fase oscura [24]:** los ATP y el NADPH generados en la fase luminosa se utilizan para convertir el dióxido de carbono (CO_2) en moléculas orgánicas, como los azúcares. Este proceso se conoce como fijación del carbono y es fundamental para la producción de biomasa

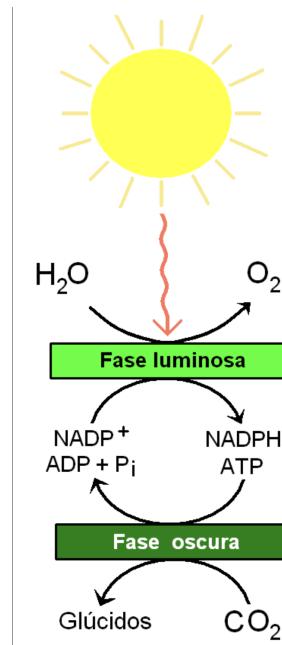


Figura 3.2: dibujo aclaratorio de las fases de la fotosíntesis extraído de Wikipedia

3.2. Factores que influyen en el crecimiento de una planta

Es esencial mantener un control adecuado de la temperatura, la humedad y la luminosidad para fomentar un desarrollo correcto de una planta. [13]

- **La luminosidad** es fundamental para la fotosíntesis. Este es el proceso a través del cual las plantas convierten la luz solar, el agua y los nutrientes en alimento (glucosa). Sin suficiente luz, una planta no puede producir la energía que necesita para crecer. Pero de nuevo, demasiada luz (especialmente la luz solar directa y fuerte), puede ser perjudicial y causar quemaduras en las hojas de la planta. Por eso es esencial garantizar que la planta reciba la cantidad adecuada de luz.
- **La humedad** juega un papel vital en la salud de las plantas. La humedad ambiental afecta a la tasa de transpiración. Este es el proceso mediante el cual el agua se evapora de las hojas de las plantas. Este proceso es esencial para el transporte de nutrientes a través de la planta, pero si la humedad es demasiado baja, la planta puede perder agua más

rápidamente de lo que puede absorberla, llevándola a la deshidratación. Por otro lado, demasiada humedad puede promover el crecimiento de hongos y otras enfermedades. Por lo tanto, mantener el equilibrio correcto es crucial. La humedad del suelo es uno de los factores más críticos para el crecimiento de una planta. Todas las plantas necesitan agua para sobrevivir y crecer. Las raíces de las plantas toman agua del suelo y la transportan al resto de la planta para realizar una variedad de funciones vitales. Entre estas funciones se incluyen la fotosíntesis, el transporte de nutrientes desde el suelo a través de la planta, y la regulación de la temperatura de la planta a través de la transpiración. Si el suelo está demasiado seco, las plantas no pueden obtener suficiente agua para llevar a cabo estas funciones vitales y pueden empezar a marchitarse. Por contra, si el suelo está demasiado húmedo, las raíces de las plantas pueden llegar a saturarse de agua, lo que puede impedir que las raíces reciban el oxígeno que necesitan para sobrevivir. En el peor de los casos, un exceso de agua puede llevar a la putrefacción de la raíz, resultando fatal para la planta. Además, la cantidad de humedad en el suelo también puede afectar la capacidad del suelo para retener nutrientes. Algunos nutrientes son solubles en agua y pueden ser lavados del suelo si hay demasiada humedad. Por otro lado, si el suelo está demasiado seco, puede ser difícil para las raíces de las plantas absorber los nutrientes que necesitan.

- **La temperatura** es un factor crítico en el crecimiento de las plantas. Las plantas, al igual que los humanos, tienen un rango de temperaturas máximas y mínimas en las que pueden sobrevivir y prosperar. Si la temperatura es demasiado baja, una planta puede entrar en un estado de latencia, en el cual el crecimiento se ralentiza o incluso se detiene, marchitando la planta. Por el contrario, si la temperatura es demasiado alta, puede dañar la estructura celular de la planta y causar estrés, lo cual puede resultar en un crecimiento pobre o en la marchitación de la planta. Por lo tanto, mantener la temperatura dentro del rango óptimo es crucial para garantizar un crecimiento saludable.

Además de estos tres factores en los que se centra Green In House, existen muchos más factores que influyen en el crecimiento de las plantas como por ejemplo la calidad del suelo, la cantidad de nutrientes que tiene la tierra, el abono que se utiliza, el espacio que dispone la planta para crecer, la necesidad de podar las ramas, etc. En esta memoria no se incide más sobre ellos ya que no serán controlados por Green In Houe, pero es relevante destacar la existencia de los mismos.

3.3. Raspberry Pi: Microcomputador con entradas y salidas

Raspberry Pi [4] es una serie de computadoras de placa única, con bajo coste y del tamaño de una tarjeta de crédito, la cual fue desarrollada en el Reino Unido por la Raspberry Pi Foundation, con el objetivo de promover la enseñanza de informática básica en las escuelas. Sin embargo, debido a su gran versatilidad y potencia de cómputo, se ha vuelto increíblemente popular en una variedad de aplicaciones, desde servidores web caseros hasta prototipos de productos electrónicos. En la Figura 3.3 se puede apreciar como es físicamente una Raspberry Pi.



Figura 3.3: fotografía de una Raspberry Pi extraída de internet

La Raspberry Pi ofrece varias ventajas frente a computadores y microcontroladores convencionales, al hacer una fusión de lo mejor de ambos mundos en una única placa. Algunas de las características que hacen destacar a esta placa son las siguientes:

- **Interfaz de Entrada/Salida General (GPIO):** uno de los aspectos más atractivos es su conjunto de pines de Entrada/Salida General (GPIO). Estos pines permiten la interacción con una amplia variedad de componentes electrónicos, como sensores y actuadores. Los pines GPIO de la Raspberry Pi son muy flexibles y pueden configurarse para leer o escribir datos digitales, utilizar buses de comunicación y en algunos modelos para leer y generar señales analógicas, lo que la convierte en una plataforma ideal para proyectos de automatización caseros.
- **Rendimiento y Almacenamiento:** a pesar de su pequeño tamaño, es una computadora completamente funcional, la cual cuenta con

procesadores de varios núcleos, permitiendo realizar tareas paralelas, y una tarjeta gráfica que permite desplegar un entorno gráfico y utilizarla como si de un ordenador convencional se tratase. Puede ejecutar varios sistemas operativos de carácter general como Windows y Linux, aunque dispone de su propia distribución Linux optimizada para su hardware, conocida como Raspbian. Esta es la distribución de sistema operativo que se ha elegido para Green In House. Además, dispone de un slot de tarjeta SD, el cuál funciona como un disco duro, lo que permite tener un almacenamiento de datos local de gran tamaño, lo cuál permite a Green In House no tener problema de espacio de almacenamiento de su base de datos.

- **Red y servidor:** puede conectarse a las red a través de Ethernet o Wi-Fi (en los modelos que cuentan con esta característica). Esto permite a Green In House proporcionar y recoger datos a través de una API REST. El modelo utilizado para el proyecto no dispone de conexión WiFi, por lo que ha sido necesario instalar un adaptador WiFi y sus correspondientes controladores.
- **Costo y comunidad:** este microcomputador es una opción atractiva debido a su bajo costo (aunque actualmente debido a los problemas en la obtención de semiconductores y la inflación, su precio se ha visto muy incrementado respecto a cuando salieron al mercado). A pesar de su potencia y flexibilidad, es significativamente más barata que la mayoría de las computadoras o microcontroladores especializados. Además, cuenta con una comunidad de usuarios activa y entusiasta que puede ser una excelente fuente de ayuda y recursos, al ser de carácter Open Source.

3.3.1. Raspbian: Sistema operativo de Raspberry Pi

Como se ha comentado anteriormente, existen diferentes versiones de sistema operativo soportados por la Raspberry Pi, o más bien dicho, modificados para poder funcionar en ella [3]. Para la realización de Green In House se ha optado por el uso de su distribución oficial Raspbian, en su versión 6.27, ya que está completamente diseñada para sacar el máximo partido a las características de la Raspberry Pi. Aunque existan otros sistemas operativos optimizados que prometan dar mayor rendimiento, he decidido utilizar Raspbian en su versión estable, para evitar posibles problemas derivados de inconsistencias internas del sistema operativo. En la Figura 3.4 se puede ver como es el escritorio de Raspbian.

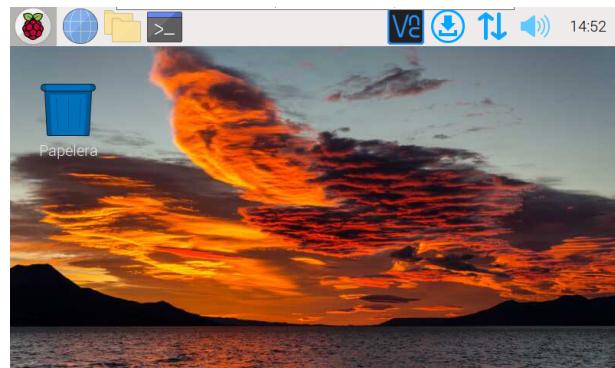


Figura 3.4: captura de pantalla del escritorio de Raspbian de mi Raspberry

3.4. Bash: Lenguaje de Programación Shell Linux

Bash [22] es un intérprete de comandos de Unix y es el acrónimo de Bourne Again SHell. Se trata de una evolución del shell Bourne, que fue uno de los primeros intérpretes de comandos del sistema operativo Unix. Bash fue creado por Brian Fox y lanzado por primera vez en 1989.

Bash es el shell por defecto en la mayoría de las distribuciones de Linux actuales y también se puede utilizar en otros sistemas operativos, como Windows y Mac, a través de software como CMD y Terminal. Bash se utiliza tanto para ejecutar comandos directamente en la terminal, como para crear scripts de shell, los cuales son programas que ejecutan una serie de comandos.

- **Comandos:** Bash es un intérprete de comando, lo que significa que los usuarios pueden escribir comandos, los cuales Bash interpreta y ejecuta. Esto permite a los usuarios interactuar directamente con el sistema operativo y realizar tareas como gestionar archivos, iniciar y detener programas, configurar el entorno del sistema, etc
- **Scripts de Shell:** Bash también es un lenguaje de programación, lo que significa que los usuarios pueden escribir programas (denominado scripts) que ejecutan una serie de comandos. Estos scripts son especialmente útiles para automatizar tareas repetitivas. Un script de Bash puede incluir funciones, bucles, condicionales, y otras estructuras de control de flujo comunes de los lenguajes de programación.

3.4.1. Utilización de Bash en Green In House

Green In House cuenta con varios scripts Bash desarrollados para facilitar las tareas de instalación, despliegue, ejecución y parada de la aplicación.

3.5. Python: Lenguaje de programación de aplicaciones de carácter general

Python [26] es un lenguaje de programación interpretado, de alto nivel y de propósito general. Fue creado por Guido van Rossum y lanzado por primera vez en 1991. Algunas de sus características principales son:

- **Legibilidad y sintaxis clara:** fue diseñado con una gran énfasis en la legibilidad y la simplicidad, lo que facilita el aprendizaje del lenguaje. Las estructuras de control de Python, las funciones, los bucles y las sentencias condicionales, están diseñadas para ser fácilmente comprensibles.
- **Interpretado:** es un lenguaje interpretado, lo que significa que el código Python se ejecuta línea por línea. Esto facilita la depuración de los programas y la experimentación interactiva en la terminal de Python o en entornos como Jupyter Notebook.
- **Multiparadigma:** admite varios paradigmas de programación, incluyendo programación orientada a objetos, programación imperativa y programación funcional, lo que le da una gran flexibilidad para resolver problemas de diferentes maneras.
- **Ecosistema de librerías:** tiene un ecosistema de librerías de terceros extremadamente rico, que lo hacen adecuado para una amplia gama de tareas. Librerías como NumPy y SciPy para la computación científica, Django y Flask para el desarrollo web, y TensorFlow y PyTorch para el aprendizaje profundo, SQLAlchemy para la gestión de base de datos, OpenAPI para el desarrollos de servidores API REST, Tkinter para el desarrollo de interfaces gráficas, Adafruit para el uso de sensores electrónicos... hacen de Python una opción popular en muchos campos de la informática y la ciencia de datos.

Tal y como se describe en los puntos anteriores, Python es un lenguaje de programación dinámico y fuertemente tipado, que combina una sintaxis

clara, una semántica simple y un ecosistema de librerías de terceros amplio y robusto. Estas características han llevado a Python a ser uno de los lenguajes de programación más populares en el mundo. Además, debido a estas características, es uno de los lenguajes más utilizados para realizar prototipados y desarrollo ágil de proyectos, al permitir a los programadores realizar programas utilizando un número menor de líneas de código de las que se necesitarían escribir en otros lenguajes como C++ o Java. Debido a su extremada versatilidad, es muy apropiado para una amplia gama de tareas, desde el desarrollo de programas simples, pasando por el desarrollo web, hasta el análisis de datos y la inteligencia artificial.

3.5.1. Utilización de Python en Green In House

Todo el código funcional de Green In House está desarrollado en Python, apoyándose en el uso de librerías de terceros para la gestión de los diferentes servicios y tecnologías que utiliza e implementa.

3.6. Flutter: lenguaje de programación de aplicaciones multiplataforma

Flutter [11] es un framework de desarrollo de software multiplataforma de código abierto creado por Google. Algunas de sus características más importantes son:

- **Recarga en caliente:** Uno de los aspectos más útiles es su capacidad de recarga en caliente. Esta característica permite a los desarrolladores experimentar, construir interfaces de usuario, agregar características y corregir errores más rápido. Los cambios en el código se reflejan en tiempo real sin necesidad de reiniciar la aplicación.
- **Material Design y Cupertino:** está estrechamente integrado con el diseño Material de Google y el diseño Cupertino de Apple, permitiendo a los desarrolladores crear interfaces de usuario que siguen las directrices de diseño de cada plataforma.
- **Rendimiento:** utiliza el lenguaje de programación Dart, que se compila en código nativo, lo que significa que las aplicaciones tienen un rendimiento cercano al nativo en todas las plataformas.

- **Widgets:** utiliza un sistema de widgets para crear la interfaz de usuario. Los widgets son bloques de construcción modulares que incluyen elementos como botones, textos, barras de desplazamiento, conmutadores, etc.
- **Soporte multiplataforma:** está diseñado para permitir a los desarrolladores crear aplicaciones de alto rendimiento y alta fidelidad para múltiples sistemas como iOS, Android y web, todo desde un mismo código
- **Integración de código nativo:** permite la integración de código nativo de los distintos sistemas que soporta, en caso de que se necesite realizar alguna funcionalidad específica o crítica que requiera de desarrollo nativo.

3.6.1. Dart: el lenguaje de programación de Flutter

El lenguaje de programación utilizado en Flutter es Dart [10] , el cual también ha sido desarrollado por Google. Dart es un lenguaje orientado a objetos que se compila *Just in Time* (JIT) para el desarrollo y *Ahead of Time* (AOT) para la producción. Esto significa que durante el desarrollo, el código se puede compilar y cambiar sobre la marcha (JIT), lo que permite la recarga en caliente. Para la producción, el código se compila antes de tiempo (AOT) en código nativo, lo que resulta en un rendimiento muy eficiente.

Dart es un lenguaje fuertemente tipado, lo que significa que una vez que se ha creado una variable de un tipo, no es posible cambiarlo. Esto tiene sus desventajas, pero a la vez ayuda a evitar errores comunes en tiempo de ejecución, los cuales podrían hacer caer la aplicación. A pesar de su seguridad de tipos, Dart sigue siendo un lenguaje fácil de aprender y usar, especialmente para aquellos que ya están familiarizados con otros lenguajes de programación orientados a objetos, como Java, C++ o Python.

3.6.2. Utilización de Flutter en Green In House

Flutter ha sido utilizado para desarrollar una aplicación móvil capaz de utilizar los *endpoints* de la API REST integrada en Green In House. Gracias a ello puede recoger los datos de la base de datos alojada en la Raspberry y puede trabajar con ellos y graficarlos, para mostrar al usuario información de manera intuitiva.

3.7. API REST:

Una API REST [17] (Representational State Transfer) es una forma estándar de diseñar servicios web, los cuales permiten la comunicación y transferencia de datos entre diferentes aplicaciones. Se basa en los principios del protocolo HTTP (*Hypertext Transfer Protocol*) y se utiliza ampliamente en el desarrollo de aplicaciones web y móviles. Una API REST se compone de entidades o conjuntos de datos específicos. Se accede a ellos mediante una URL (*Uniform Resource Locator*) única, esto se conoce como *endpoint*. Para realizar la petición y envío de los datos se utilizan los métodos estándar de HTML:

- **GET:** utilizado para consultar datos existentes.
- **POST:** utilizado para crear nuevos datos.
- **PUT:** utilizado para actualizar datos existentes.
- **DELETE:** utilizado para borrar datos existentes.

En la Figura 3.5 se puede ver un ejemplo del empleo de estos métodos en Green In House. La comunicación con una API REST se realiza a través de peticiones HTTP. El cliente envía una solicitud al servidor y este responde con los datos solicitados. Los datos se suelen enviar en formato JSON , que es un formato ligero y legible para el intercambio de datos, así como independiente del lenguaje de programación.

La principal ventaja de utilizar una API REST es la interoperabilidad entre diferentes sistemas y plataformas. Permite que aplicaciones desarrolladas en diferentes lenguajes de programación y ejecutándose en diferentes plataformas, puedan comunicarse de manera fácil, eficiente y consistente. Además, al seguir los principios del protocolo HTTP, se aprovechan las características y funcionalidades de dicho protocolo, como la caché, la autenticación y la seguridad.

GIH backend service REST API 1.0 OAS3

/api/v1/openapi.json

REST API del servicio de backend.

Autor: Oscar Valverde Escobar.

Green In House. Grado en Ingeniería Informática, Universidad de Burgos, 2022-2023.

Servers: /api/v1

Servidor Operaciones sobre el propio servidor (query de estado del servidor).

HEAD / Health test for the service

Registros Sensores Operaciones relacionadas con los sensores.

Sensores Operaciones relacionadas con los sensores.

Plantas Operaciones relacionadas con los sensores.

GET /Plantas/All Obtener todas las plantas.

GET /Plantas/All/Active Obtener todas las plantas vivas actualmente.

GET /Plantas/All/Active/FromType Obtener todas las plantas vivas actualmente de un tipo especificado.

GET /Plantas/All/FromType Obtener todas las plantas de un tipo especificado.

DELETE /Plantas/One Dar por marchitada la planta especificada.

GET /Plantas/One Obtener la planta especificada.

POST /Plantas/One Crear la planta especificada.

PUT /Plantas/One Modificar la planta especificada.

Figura 3.5: captura de pantalla del Swagger de la API REST de Green In House desarrollada con OpenAPI

Técnicas y herramientas

En este capítulo se describen las técnicas y metodologías que se han empleado para realizar el diseño de la aplicación de Green In House, así como las diferentes tecnologías que se han empleado para su desarrollo.

4.1. Técnicas y metodologías

Para poder realizar de manera óptima y eficiente el diseño de Green In House, con un carácter iterativo y con posibilidades de ampliar sus funcionalidades continuamente en cada iteración, se ha recurrido a la utilización de patrones de diseño y a una metodología ágil.

4.1.1. Patrones de Diseño

Los patrones de diseño [2] son soluciones probadas a problemas comunes que nos encontramos en el diseño de software. Son como plantillas que se pueden aplicar en diferentes situaciones, proporcionando un esquema que puede ayudar a resolver un problema en particular. Básicamente son una forma de reutilizar conocimientos y experiencias previas para hacer el código de una aplicación más eficiente, mantenible y comprensible. En Green In House se ha hecho uso de los siguientes patrones de diseño, los cuales están más detallados en el apartado de anexos:

- Patrón de diseño **MVC** (Modelo-Vista-Controlador)
- Patrón de diseño **Adaptador**

- Patrón de diseño **Factoría**
- Patrón de diseño **Estrategia**
- Patrón de diseño **Plantilla**
- Patrón de diseño **Fachada**
- Patrón de diseño **Singleton**

4.1.2. Scrum: Metodología de gestión ágil

Scrum [15] es un marco de trabajo de metodología ágil ampliamente utilizado en el desarrollo de software. Se basa en principios colaborativos, flexibilidad y entrega incremental. El objetivo principal de Scrum es permitir a los equipos de trabajo desarrollar productos de alta calidad de manera eficiente. Además permite realizar una entrega temprana y continua de valor al cliente, a la vez que se fomenta la mejora continua del proceso de desarrollo.

Scrum se basa en un enfoque iterativo e incremental, en el que el trabajo se divide en ciclos continuos de desarrollo llamados *sprints*. Cada sprint tiene una duración fija, generalmente de 1 a 4 semanas, y al final de cada sprint se entrega un incremento funcional del producto. Esta metodología permite ajustar y cambiar los requisitos a medida que se realizan los incrementos y se obtiene retroalimentación del cliente.

Durante el *sprint*, se llevan a cabo reuniones diarias llamadas *Daily Scrum*, en las cuales el equipo comparte el progreso realizado, identifica posibles problemas y planifica las tareas para el día. También se realizan reuniones de planificación de *sprint*, revisión de *sprint* y retrospectivas, que ayudan al equipo a refinar y mejorar continuamente su trabajo. El equipo de desarrollo de Scrum se organiza de manera autónoma y autoorganizada. No hay jerarquías rígidas, sino que se fomenta la colaboración y la toma de decisiones conjunta. El equipo se compone de roles claramente definidos:

- **Product Owner:** es el responsable de representar los intereses del cliente y definir los requisitos del producto. También se encarga de mantener el *product backlog*, priorizar las tareas y asegurar que se cumplan las expectativas del cliente.

- **Scrum Master:** es el facilitador del equipo Scrum. Su rol es asegurar que se sigan los principios y prácticas de Scrum, eliminar problemas y garantizar un ambiente de trabajo colaborativo y productivo.
- **Equipo de Desarrollo:** está formado por profesionales que llevan a cabo el trabajo de desarrollo del producto. Son multidisciplinares y autorganizados, asumiendo la responsabilidad de entregar un incremento de software funcional al final de cada *sprint*.

En la figura 4.1 se puede ver un diagrama de los roles de Scrum y como intervienen en las diferentes fases de proceso.

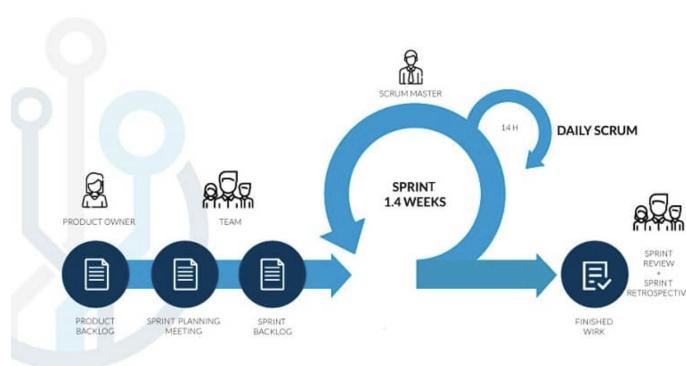


Figura 4.1: resumen de funcionamiento de metodología Scrum

4.2. Herramientas

Green In House no es una simple maceta, si no que es un proyecto que aglutina una gran cantidad de funcionalidades y tecnologías diferentes, las cuales han sido desarrolladas utilizando las siguientes herramientas.

4.2.1. Venv: Entornos Virtuales en Python

Uno de los aspectos más críticos al desarrollar proyectos en Python es la gestión de dependencias. Con frecuencia, diferentes proyectos requieren diferentes versiones de las mismas librerías. La instalación global de dichas librerías se hace inviable, ya que puede generar conflictos, debido a que

determinadas versiones pueden integrar métodos que otras no, o requerir otra sintaxis, etc. Debido a esto, los entornos virtuales de Python, o *venv* desempeñan un papel crucial. [7]

El módulo *venv* de Python proporciona un soporte para crear entornos virtuales ligeros con sus propios directorios de sitio, aislados del intérprete Python global del sistema. Cada entorno virtual tiene su propio intérprete de Python, lo que permite instalar y gestionar paquetes de Python independientes para cada proyecto.

4.2.2. SQLAlchemy: librería de base de datos en Python

SQLAlchemy [18] es una librería de Python que facilita la comunicación entre programas desarrollados en Python y las bases de datos SQL. Proporciona un conjunto completo de operaciones de persistencia de datos de alto nivel, así como un marco de abstracción de base de datos de SQL completo para aportar una mayor flexibilidad.

- **ORM (Object Relational Mapper):** una de sus características más poderosas es su ORM (Object-Relational Mapping). Un ORM es una técnica de programación que facilita la conversión de datos entre sistemas incompatibles (en este caso, entre objetos Python y tablas SQL). En otras palabras, un ORM nos permite trabajar con bases de datos SQL en un lenguaje orientado a objetos, como Python. Esto significa que con el ORM de SQLAlchemy, se pueden crear clases en Python que se correspondan directamente con las tablas de la base de datos. Cada instancia de una clase representa una fila en la tabla correspondiente. Esta correspondencia entre objetos Python y tablas SQL nos permite interactuar con los datos de la base de datos de una manera más intuitiva y similar a como normalmente lo hace Python.
- **Abstracción de base de datos:** además del ORM, también proporciona un nivel de abstracción que nos permite cambiar entre diferentes sistemas de bases de datos SQL con pocos o ningún cambio en el código. Esto es útil si en un futuro se decidiese cambiar la base de datos subyacente de Green In Hosue o quisiese hacer el código más general para que funcione con diferentes sistemas de bases de datos.

4.2.3. OpenAPI: librería de API REST en Python

OpenAPI [14], anteriormente conocido como Swagger, es una especificación para archivos de definición de API REST. Estos archivos de definición pueden estar escritos en YAML o JSON y describen los detalles de la API REST, entre los cuales por ejemplo se encuentran:

- Endpoints disponibles.
- Métodos HTTP soportados.
- Parámetros de las solicitudes.
- Respuestas esperadas.
- Modelos de los datos a comunicar.

La especificación OpenAPI se utiliza para documentar y describir APIs REST de forma estandarizada y fácil de entender, tanto para las personas, como para las máquinas. Esto facilita el desarrollo, la prueba, y la integración de servicios basados en APIs. Un beneficio adicional de la especificación OpenAPI es que puede generar automáticamente una interfaz de usuario interactiva, permitiendo a los usuarios o desarrolladores explorar y probar la API de manera intuitiva.

4.2.3.1. Microframework Connexion

Green In House utiliza el Microframework Connexion, el cual mapea el API especificado en el archivo `spec.yml` con OpenAPI en Python, facilitando la creación de los endpoints para que otras aplicaciones puedan conectar con el servicio, el cual por defecto es lanzado en el puerto 5000. Para poder acceder a estos endpoints hay que hacerlo utilizando de base la siguiente url: `http://192.168.1.240:5000/api/v1/`

4.2.3.2. JSON: Archivos de datos con formato específicos

JSON, que significa JavaScript Object Notation, es un formato ligero de intercambio de datos. JSON es un formato de texto que es completamente independiente del lenguaje pero utiliza convenciones que son familiares para los programadores de diversos lenguajes como C, C++, C#, Java, JavaScript,

Perl, Python, y muchos otros. Se utiliza como sistema de almacenaje y compartición de datos entre diferentes aplicaciones, al tener un formato independiente de la aplicación.

4.2.3.2.1 Formato de JSON

Un objeto JSON es un conjunto de pares clave-valor rodeado por llaves {}. Las claves son cadenas de caracteres y los valores pueden ser:

- Números.
- Cadenas de caracteres.
- Booleanos (true o false).
- Otros objetos JSON.
- Arrays de cualquiera de las opciones anteriormente mencionadas. Un array JSON es una lista de valores rodeada por corchetes [].

4.2.4. TKinter: librería de interfaces gráficas en Python

TKinter [6] es una librería de Python utilizada para el desarrollo de interfaces gráficas de usuario (GUI). Es la interfaz estándar de Python para el kit de herramientas Tk, y es tanto simple como eficaz para la creación de una variedad de programas. TKinter proporciona un poderoso conjunto de widgets de interfaz de usuario:

- Botones.
- Cajetines de texto.
- Listas.
- Barras de desplazamiento.
- Etiquetas.
- Menús.
- Ventanas emergentes.

- Frames.

Todo esto permite crear interfaces de usuario altamente personalizables y funcionales. En la Figura 4.2 se muestra la interfaz de la aplicación desarrollada con TKinter para que el usuario pueda introducir los datos de su red WiFi.



Figura 4.2: Captura de pantalla de la app gráfica de Green In House desarrollada en TKinter

4.2.5. Adafruit - CircuitPython: librería de sensores y actuadores en Python

Adafruit CircuitPython [16] es una implementación de Python diseñada para simplificar la experimentación con hardware de bajo coste. CircuitPython es una implementación de Python 3 diseñada para microcontroladores. Esta librería está desarrollada por Adafruit Industries, una empresa reconocida en el mundo de la electrónica y hardware de código abierto.

Una de las características clave de CircuitPython es su simplicidad. Proporciona una excelente plataforma para aquellos que están empezando a programar y trabajar con hardware. CircuitPython es capaz de interactuar directamente con el hardware. Esto incluye leer entradas y salidas digitales, utilizar la comunicación I2C y SPI, y controlar y leer sensores y actuadores. Esto permite a los desarrolladores y estudiantes explorar la interacción entre el software y el hardware de una manera intuitiva y fácil de entender. En la tabla 4.1 se muestran los diferentes módulos y sensores electrónicos que se han utilizado en el desarrollo de Green In House mediante la librería Circuit Python de AdaFruit.

| Sensor | Función | Tipo de medición | Zona | Tipo de Señal |
|---------|-----------|----------------------|----------|---------------------|
| MCP3008 | Conversor | - | - | Analógico a Digital |
| DHT11 | Sensor | Humedad, Temperatura | Ambiente | Digital |
| LM35 | Sensor | Temperatura | Ambiente | Analógica |
| FC28 | Sensor | Humedad | Maceta | Analógica |
| LDR | Sensor | Luminosidad | Ambiente | Analógica |
| BH1750 | Sensor | Luminosidad | Ambiente | Digital |

Tabla 4.1: Modelos de sensores utilizados en Green In House

4.2.6. Git: sistema de control de versiones

Git [1] es un sistema de control de versiones distribuido de código abierto que permite gestionar y ver los cambios que se han ido realizando un proyecto a lo largo del tiempo. Está diseñado para manejar todo tipo de proyectos (ya sean pequeños o muy grandes), con una gran velocidad y eficiencia. Algunas de las características más destacadas de Git son:

- **Sistema distribuido:** su característica más importante es que es un sistema de control de versiones distribuido. Esto significa que cada desarrollador tiene una copia completa del historial de cambios del proyecto en su máquina local. Esta característica permite trabajar de manera descentralizada y autónoma, sin tener que estar constantemente sincronizado con un servidor central.
- **Integridad de los datos:** está diseñado con un fuerte énfasis en la integridad de los datos. Cada cambio o *commit* tiene una suma de comprobación asociada que se utiliza para verificar la integridad de los datos. Esto permite a los desarrolladores asegurarse de que, una vez que se ha registrado un cambio, los archivos no se alterarán sin su conocimiento.
- **Ramas:** ofrece un soporte robusto y flexible para la ramificación y fusión de código. Esta característica permite a los desarrolladores trabajar en características y experimentos de manera aislada, sin afectar el código principal del proyecto. Una vez que una característica está lista, proporciona herramientas para fusionar ese trabajo con la rama principal.
- **Desempeño:** ha sido diseñado con un fuerte enfoque en el rendimiento. A pesar de que maneja historiales de cambios complejos y grandes cantidades de archivos y cambios, las operaciones son generalmente

muy rápidas. Esta es una característica de gran valor ya que permite realizar un desarrollo ágil y eficiente.

- **Plataforma de colaboración:** Combinado con plataformas de alojamiento de código como GitHub, se convierte en una poderosa plataforma de colaboración, permitiendo a los equipos de desarrolladores revisar y discutir cambios, gestionar tareas e integrar de manera continua el código. Este sistema de colaboración es esencial para mantener la calidad del código y coordinar el trabajo en equipo.

4.2.6.1. GitHub: Plataforma *open source* de repositorios

GitHub [8] es una plataforma de desarrollo de software basada en la nube que utiliza el sistema de control de versiones Git. Con el paso del tiempo se ha convertido en una herramienta esencial para muchos desarrolladores y organizaciones debido a las capacidades que ofrece para colaborar y gestionar proyectos de software.

GitHub permite a los desarrolladores almacenar sus proyectos en repositorios, que son esencialmente directorios de proyectos. Estos repositorios pueden ser públicos (cualquier persona puede ver y contribuir al código) o privados (restringe el acceso sólo a personas específicas).

Una característica principal de GitHub es su enfoque en la colaboración. Los desarrolladores pueden hacer un *fork* de un repositorio, lo que crea una copia del proyecto (de la versión seleccionada) en su propia cuenta de GitHub. Después de eso pueden hacer cambios en esa copia y, cuando estén listos, pueden enviar una *pull request* al repositorio original. El propietario del repositorio original puede revisar estos cambios y optar por descartarlos o por fusionarlos con el proyecto principal.

Github también integra un control de ramas dentro de un mismo repositorio. Una rama en Git es una referencia a uno de los *commits*. Cuando creamos una rama, en realidad estamos creando una nueva línea de desarrollo en base al código existente en un determinado *commit*. Esto significa que podemos realizar cambios en esa rama sin afectar a otras ramas. Esto permite a los desarrolladores trabajar en características específicas, pruebas o experimentos en aislamiento, lo que facilita la organización del código y reduce el riesgo de introducir errores en el código de producción.

Las ramas son especialmente útiles en el desarrollo colaborativo, ya que cada colaborador puede trabajar en una rama separada sin interferir con el trabajo de los demás. Una vez que un colaborador ha terminado su trabajo

en una rama, puede solicitar que se integre de nuevo en la rama principal (a menudo llamada *master* o *main*) a través de un *pull request*.

Además de estas capacidades de colaboración y control de versiones, GitHub ofrece características como la gestión de incidencias para rastrear y resolver problemas, y acciones de GitHub para automatizar flujos de trabajo de desarrollo.

4.2.6.1.1 Utilización de GitHub en Green In House

Git ha sido utilizado en el desarrollo de Green In House junto con la plataforma GitHub para realizar el seguimiento de cambios, garantizar la integridad del código y detección de problemas tras la realización de cambios. También ha sido una herramienta útil para poder mantener en la nube una copia de seguridad del proyecto actualizada continuamente en caso de que se estropease el entorno de desarrollo, evitando la pérdida del código del proyecto.

4.2.7. VSCode: IDE de programación multilenguaje

VSCode [12] es la abreviatura de Visual Studio Code, el cual es un editor de código fuente desarrollado por Microsoft. Es una herramienta extremadamente popular y versátil utilizada por desarrolladores de software en todo el mundo. Algunas de las características clave que tiene son:

- **Editor de código:** proporciona un editor de código altamente personalizable con resaltado de sintaxis acorde al lenguaje utilizado, sugerencias inteligentes, sangría automática y otras características útiles para facilitar la escritura de código.
- **Extensibilidad:** hay miles de extensiones disponibles que agregan funcionalidades adicionales, como soporte para diferentes lenguajes de programación, integración con sistemas de control de versiones, etc.
- **Depuración:** incluye capacidades de depuración integradas que permiten ejecutar y depurar el código directamente desde el editor. Esto es especialmente útil para identificar y solucionar problemas de manera ágil en el código.
- **Integración con Git:** tiene integración con el sistema de control de versiones de Git. Permite realizar de manera cómoda y ágil tareas

comunes de Git, como confirmar cambios, crear ramas y fusionar código, etc.

- **Terminal integrada:** dispone de una terminal integrada que permite ejecutar comandos de línea de comandos directamente dentro del editor. Esto evita tener que alternar entre el editor y la terminal del sistema operativo, lo que mejora la eficiencia en el flujo de trabajo.

4.2.8. Conexión SSH: Control remoto

SSH [9] es la abreviatura de *Secure Shell*. Es un protocolo de red que permite una comunicación segura entre dos sistemas a través de una red insegura. Proporciona un mecanismo de autenticación y cifrado robusto para proteger la información transmitida entre ambos sistemas. Algunos de los usos más comunes de SSH son:

- **Acceso remoto:** permite acceder a un sistema remoto de forma segura. Esto es útil cuando necesitas administrar un servidor o realizar tareas en un sistema al que no tienes acceso físico.
- **Transferencia de archivos segura:** permite transferir archivos de manera segura entre sistemas utilizando herramientas como *scp* o *sftp*. Esto es útil cuando necesitas mover archivos de un sistema a otro de forma segura y confiable.
- **Túneles de red:** permite crear túneles de red seguros que redirigen el tráfico a través de una conexión cifrada. Esto se utiliza comúnmente para acceder de forma segura a servicios de red, como bases de datos o servidores web, a través de una red no confiable.
- **Ejecución de comandos remotos:** permite ejecutar comandos en un sistema remoto sin tener que estar físicamente presente en él. Esto es especialmente útil cuando necesitas automatizar tareas o administrar múltiples sistemas de forma remota.

Para poder establecer una conexión SSH es necesario contar con los siguientes elementos:

- **Servidor SSH:** el sistema al que se desea acceder debe tener un servidor SSH instalado y configurado correctamente.

- **Cliente SSH:** es necesario tener instalado y configurado un cliente SSH en el sistema local desde el cual te conectarás al servidor SSH remoto. En sistemas basados en Unix, como Linux o macOS, el cliente SSH suele estar disponible de forma nativa. En sistemas Windows, se pueden utilizar programas como PuTTY, Git Bash o VSCode, que incluyen clientes SSH.
- **Credenciales de acceso:** para poder establecer la comunicación entre cliente y servidor hay que contar con las credenciales de acceso adecuadas, como un nombre de usuario y una contraseña, o una clave SSH privada si se ha configurado.

4.2.9. FlutterFlow: IDE de programación GUI para Flutter

Flutter Flow [5] es una plataforma online que permite diseñar y desarrollar aplicaciones móviles utilizando Flutter de manera visual y sin necesidad de escribir código. Proporciona una interfaz intuitiva y fácil de utilizar que te permite crear interfaces de usuario y lógica de aplicación de manera rápida y sencilla. Para los desarrolladores experimentados, es importante destacar que también permite el diseño de funcionalidades propias más complejas de las que inicialmente ofrece. Algunas de sus características principales son:

- **Diseño visual:** ofrece una forma sencilla de diseñar la interfaz de usuario de una aplicación de forma visual, arrastrando y soltando componentes en un lienzo. Esto te permite ver instantáneamente cómo se verá la aplicación mientras se va construyendo.
- **Sin necesidad de codificación:** una de las ventajas más destacadas de Flutter Flow de cara al público general es que no requiere conocimientos de programación para crear aplicaciones móviles. Permite construir la lógica de la aplicación utilizando una interfaz gráfica intuitiva en lugar de escribir código manualmente.
- **Generación automática de código Flutter:** aunque no es necesario escribir código, Flutter Flow genera automáticamente el código Flutter correspondiente a medida que se diseña la aplicación. Esto permite obtener un proyecto de Flutter completamente funcional, el cual puede ser personalizado y mejorado a mayores según las necesidades de la aplicación.

- **Colaboración en tiempo real:** permite la colaboración en tiempo real, lo que significa que puedes trabajar en equipo en el diseño y desarrollo de una aplicación simultáneamente. Esto es especialmente útil cuando se trabaja junto con otros desarrolladores o diseñadores en un mismo proyecto.
- **Integraciones y personalizaciones:** admite diversas integraciones y personalizaciones para ampliar las capacidades de la aplicación diseñada. Permite agregar servicios externos, como bases de datos o servicios en la nube, y personalizar la lógica de la aplicación para adaptarse a requisitos específicos.

En sus versiones de pago ofrece integración con el repositorio GitHUB, sistema de traducción automática de textos y despliegue en las tiendas de aplicaciones de Android e iOS. Para el desarrollo de Green In House he contado con una licencia especial de estudiante, la cual me permite utilizar las funciones anteriormente nombradas. Para solicitar dicha licencia tuve que solicitarla a Flutter Flow entregando los documentos necesarios que me acreditan como alumno universitario y en 48 horas me la concedieron.

Aspectos relevantes del desarrollo del proyecto

Para poder desarrollar Green In House a sido necesario tomar una serie de decisiones de diseño que permitieran realizar todas las funcionales que se pretendían incluir en el proyecto. Durante el desarrollo del mismo, me he ido encontrando con varios problemas relevantes, los cuales no estaban previstos inicialmente. A continuación se exponen estos apartados de manera detallada.

5.1. Decisiones de diseño de Green In House

En las siguientes páginas se detallan las principales decisiones de diseño que tuve que tomar antes de poder comenzar con el desarrollo de la aplicación de Green In House. Gracias a esta toma inicial de decisiones, pude encauzar correctamente el desarrollo de la aplicación sin tener que ir implementando y desecharndo funcionalidades. Para la realización de la estructura inicial de Green In House separando las funcionalidades en diferentes capas y servicios, reutilicé la estructura básica de la práctica final de la asignatura "Diseño y mantenimiento del software". De esta práctica reutilicé algunos archivos de código fuente originalmente desarrollados por Jesús Alonso Abad, para definir los servicios básicos de configuración de directorios de la base de datos y de la API REST, así como la instalación de dependencias declaradas para cada entorno virtual, adaptándolos al código y estructura de Green In House.

5.1.1. Decisión de utilizar microcomputador o microcontrolador

Todos los programas que hemos desarrollado durante el Grado han corrido siempre internamente en un ordenador, sin necesidad de comunicarse con sensórica real. Sin embargo, Green In House cuenta con sensores electrónicos con los que ha de comunicarse, por lo que necesitaba contar con interfaces de comunicación dedicadas a interactuar con sensores electrónicos. Para realizar esta tarea lo mejor era optar por un microcontrolador Arduino o un microcomputador Raspberry, los cuales ambos cuentan con estas interfaces para comunicación con sensórica real, pero presentan grandes diferencias.

- Arduino es un microcontrolador, lo que quiere decir que internamente no ejecuta ningún sistema operativo, sino un único programa de manera cíclica. Por esa razón, no necesita tener grandes capacidades de núcleos de procesamiento, ni demasiada memoria, ya que está diseñado para el control específico de hardware electrónico mediante un programa optimizado para la tarea que tiene que realizar. Cuenta con una gran cantidad de pines digitales, analógicos y PWM, los cuales se utilizan para leer sensores y comandar actuadores. Además cuenta con múltiples sistemas de comunicación digital mediante varios protocolos como puerto serie, I2C y SPI, muy útil para poder comunicarse con otras placas Arduino u otros sistemas. Es una placa de bajo coste y muy fácil de programar, por lo que es ideal en proyectos de robótica. Mediante interrupciones de sistema se puede romper el flujo estándar del programa para ejecutar otras funciones en un determinado momento, pero en ningún momento podrá ejecutar tareas de manera simultánea. Además, tiene ciertas limitaciones en determinados áreas, como por ejemplo en el manejo de grandes cantidades de datos persistentes, ya que no está pensado para albergar una base de datos junto con su sistema gestor. Es una herramienta pensada para el control de hardware en tiempo real.
- Raspberry es un microcomputador y ejecuta un sistema operativo completo, por lo que necesita contar con una cantidad considerable de recursos para poder ejecutar dicho sistema operativo y sobre él ejecutar otros programas. Al tener un sistema operativo, permite ejecutar acciones de alto nivel de manera mucho más cómoda que en Arduino, al delegar la ejecución de dichas funciones en el sistema operativo. Otro punto importante, es que es capaz de albergar y manejar una base de datos y su sistema gestor, además de permitir

utilizar intermediarios ORM. Además, al tener múltiples núcleos, es capaz de ejecutar tareas de manera paralela, delegando el control de estos hilos en el programador de tareas del sistema operativo. Esto lo hace más eficiente que Arduino a la hora de querer montar varios servidores y ejecutar tareas de manera paralela en una misma placa. Esto es importante en Green In House, ya que se necesita ejecutar simultáneamente una tarea cíclica de lectura de sensores y almacenamiento de registros en la base de datos, un servidor API REST con conexión a la base de datos para lectura y escritura de registros y una pantalla de control. Otra gran ventaja que tiene es que soporta diversos lenguajes de programación como Python, C y C++, por lo que ofrece alternativas si se quiere ejecutar código interpretado (más simple de programar) o código compilado (con mayor eficiencia de recursos). También cuenta con pines de conexión digital para leer sensores y comandar actuadores, además de múltiples interfaces de comunicación para diversos protocolos como puerto serie, I2c y SPI. Esta placa está diseñada para ejecutar tareas en tiempo diferido, donde no es tan extremadamente importante el tiempo de respuesta como en los sistemas de tiempo real, aunque optimizando el código y dando gran prioridad a los procesos, se pueden ejecutar tareas en tiempo real.

Debido a las ventajas que aporta Raspberry Pi sobre Arduino para las características que requiere Green In House, he optado por utilizar una Raspberry Pi, ya que el tiempo de lectura de los sensores no es crítico, pero si me es indispensable contar con todas las funciones anteriormente mencionadas. Si el tiempo de control sobre la lectura de sensores fuera crítico, una buena opción sería crear un sistema híbrido, en el que un Arduino se comunicase en tiempo real con los sensores, recogiera sus datos en memoria, y se los enviase cada cierto tiempo a la Raspberry para que lo almacenase en su base de datos. De esta manera podría aprovechar lo mejor de ambas placas, pero no ha sido necesario para la realización del proyecto.

5.1.2. IDE a utilizar

Para poder desarrollar el código de Green In House de manera eficiente y cómoda, una de las principales decisiones que tuve que tomar fue la del IDE a utilizar. Entre las diferentes posibilidades valoré la utilización del IDE Geany (al venir incorporado en Raspbian), pero tenían un control muy pobre sobre la estructura de directorios del programa y no tenía integración con GitHub, por lo que lo finalmente lo descarté. La siguiente opción que

valoré fue la utilización del IDE VSCode, el cuál me permite tener control sobre la estructura de directorios del programa, cuenta con herramientas de refactorización de código, tiene integración con GitHub para poder tener mi código sincronizado con un repositorio con control de versiones y cuenta con un cliente SSH. Este cliente SSH me permite programar desde mi ordenador utilizando VSCode pero actuando de manera remota sobre los archivos almacenados en mi Raspberry, permitiéndome hacer un despliegue de los cambios en mi aplicación remotamente, y controlar la Raspberry por línea de comandos. Por estas razones elegí VSCode como IDE a utilizar para desarrollar Green In House

5.1.3. Respaldo en la nube con control de versiones

A la hora de desarrollar Green In House, una prioridad que tenía en mente era poder tener un respaldo en la nube del código de mi proyecto, por si por alguna razón perdía algún archivo o el proyecto completo. Además quería poder realizar un seguimiento de los cambios que iba realizando en cada *sprint*, por lo que necesitaba utilizar un repositorio que integrase un control de versiones como Git. Me decanté por utilizar GitHub ya que es el que he utilizado siempre y no he tenido nunca problemas o limitaciones con él. Habría sido buena idea aprovechar el sistema de control de ramas que tiene GitHub, pero al ser el único desarrollador del proyecto no lo consideré necesario, ya que no iba a haber otras personas trabajando simultáneamente, con las que pudiera colapsar mi trabajo desarrollado, ni había ninguna release realizada acorde a alguna versión de commit. Una vez desarrollada la primera release, sería mejor empezar a trabajar en ramas paralelas e ir realizando nuevas *releases* a medida que se fueran realizando las fusiones de las ramas.

5.1.4. Almacenar datos en ficheros o en base de datos

Para poder almacenar de manera persistente los datos que se van generando en la aplicación (registros de sensores, sensores del sistema, plantas, etc) tenía dos opciones, utilizar un sistema de ficheros o una base de datos. Utilizar un sistema de ficheros propio parecía tener una mayor facilidad de uso, pero a la vez presentaba varios problemas como por ejemplo:

- Posible corrupción de archivos sin sistema de solvencia de problemas.

- Necesidad de leer completamente los archivos para realizar las búsquedas de datos, por lo que cuando creciera mucho la cantidad de registros presentaría problemas de eficiencia.
- Complejidad a la hora de realizar cambios en los registros existentes para añadir nuevos campos.
- Imposible manejar múltiples consultas simultáneamente al mismo archivo al encontrarse bloqueado.
- Controlar que no se puedan generar registros duplicados.

Todos estos problemas anteriormente mencionados, ya están solventados de manera eficiente y cómoda por los actuales gestores de bases de datos, por lo que la mejor opción era implementar una base datos con su correspondiente gestor. A pesar de las mejoras que ofrece esta alternativa, también presenta algunos problemas como por ejemplo:

- Dificultad añadida de tener que manejar el gestor de la base de datos mediante un *framework* que permita implementación ORM para poder utilizar los registros como objetos de manera cómoda en el código Python.
- Necesidad de programar las transacciones de manera correcta haciendo uso de las herramientas proporcionadas por el *framework*.
- Necesidad de definir las estructuras de las tablas y su interrelación entre ellas.

Al querer desarrollar una aplicación bastante grande y con posibilidades de crecer continuamente, las ventajas de utilizar una base de datos superaban con creces a sus inconvenientes, por lo que decidí utilizar este sistema para Green In House. La base de datos utilizada es SQLite3, y es manejada mediante el *framework* SQLAlchemy, el cuál tiene implementado un modelo ORM que permite manejar los registros de las tablas como si fueran objetos dentro de Python. Este ORM se encarga tanto de generar el objeto en base al registro, como de generar el registro de la tabla en base a los datos del objeto, y lo hace de manera transparente al usuario. Además, SQLAlchemy permite realizar consultas SQL mediante un sistema de orientación a objetos, para hacer más consistente todo el código, sin necesidad de generar consultas SQL, ya que el mismo *framework* se encarga de convertir los métodos utilizados del objeto *query* en una consulta SQL.

5.1.5. Comunicación de datos con sistemas externos

Para poder comunicar con sistemas externos los datos recogidos por los sensores de Green In House y almacenados en su base de datos, he recurrido a la utilización de un servidor API REST a través de consultas realizadas mediante HTML. Este es uno de los sistemas más comúnmente utilizados para comunicar datos entre diferentes sistemas, ya que no requiere conocer como trabaja internamente dicho sistema. Simplemente se hace una petición de los datos que se quieren recoger haciendo una llamada a un *endpoint*, el cuál puede estar configurado para recibir determinados parámetros de entrada. Con los datos de la llamada, la API REST procesa internamente los datos que tiene que devolver y los encapsula en un archivo JSON, el cuál es enviado como respuesta de la petición junto con un código de respuesta HTML, que indica si la petición se pudo resolver correctamente o si hubo algún problema durante el procesamiento. Las operaciones estándar permitidas por este sistema mediante *endpoints* son las siguientes:

- **POST:** Creación de un recurso nuevo.
- **PUT:** Modificación de un recurso existente.
- **GET:** Consulta la información de un recurso.
- **DELETE:** Eliminación de un recurso existente.

En el caso de Green In House, la operación *DELETE* no borra como tal el recurso de la tabla, sino que le asigna la fecha actual como fecha de eliminación. Esto he decidido realizarlo así para mantener la integridad de la base de datos, permitiendo almacenar datos históricos de sensores y plantas que hayan sido eliminados del sistema. Si no, al borrar dicho elemento, habría que borrar en cascada todos los elementos que hagan referencia al elemento borrado.

5.2. Problemas encontrados

Durante el desarrollo de Green In House me fui encontrando problemas que inicialmente no había valorado, por lo que tuve que hacerles frente a medida que iban apareciendo. A continuación detallo una lista de los problemas más importantes que tuve que ir solucionando.

5.2.1. Necesidad de instalar un adaptador WiFi USB

El modelo de Raspberry que utilicé para desarrollar Green In House es una Raspberry Pi 2b, el cuál no cuenta con conexión WiFi, por lo que fue necesario utilizar un adaptador WiFi conectado por USB. Este problema no existen en el nuevo modelo Raspberry Pi 4, pero utilicé el modelo 2B debido a que ya disponía de dicha placa, por lo que consideré innecesario comprar una nueva placa. La explicación de su instalación se detalla en el apartado de anexos.

5.2.2. Necesidad de ingresar las credenciales de la red WiFi

Una de las principales características de Green In House es que dispone de un servidor API REST utilizado para enviar y recibir datos mediante comunicación HTTP, por lo que es necesario tener acceso a una red LAN o WLAN. Como se quiere que la maceta sea portátil y se pueda transportar fácilmente, la mejor opción es utilizar una red WLAN y conectarse a ella mediante WiFi, pero para ello se necesita contar con las credenciales dicha red. Para permitir al usuario ingresar los datos de dicha red valoré dos opciones:

- Utilización de bluetooth desde la App en Flutter: al tener que desarrollar una App para dispositivos móviles para poder ver los registros e interactuar con Green In House, la primera opción que valoré fue incluir una ventana de configuración de red en la aplicación y utilizar comunicación Bluetooth para enviar esta información desde el móvil a la Raspberry Pi. Esta idea finalmente fue descartada debido al uso de Flutter Flow para desarrollar la aplicación multiplataforma, ya que debido a limitaciones del entorno de desarrollo, no permite realizar la comunicación serie con dispositivos Bluetooth.
- Utilización de una pantalla táctil con una App con interfaz gráfica: al no poder utilizar Bluetooth para comunicarme con la Raspberry Pi desde la App móvil, decidí buscar una forma de controlar la Raspberry Pi directamente, y la solución por la que opté fue incluir una pantalla táctil en el sistema y desplegar una App que me permitiese configurar los datos de la red WiFi. Además, esta pantalla táctil se puede utilizar en líneas futuras para desarrollar un visualizador en tiempo real de los valores de los sensores de Green In House.

5.2.3. Necesidad de incorporar un módulo ADC (*analogic digital converter*) externo

Hasta el momento había trabajado con Arduino en varias ocasiones, pero nunca con Raspberry Pi, por lo que suponía que tenía las mismas posibilidades de lectura de sensores que Arduino. Sin embargo, al intentar leer el sensor analógico FC28 y los LDR (*Light Dependent Resistor*), me di cuenta de que no podía hacerlo, ya que la Raspberry Pi no contaba con un ADC incorporado, por lo que no podía leer sensores analógicos. Debido a ello, tuve que buscar alternativas para poder leer dicho sensor, y valoré las siguientes opciones:

- Sustitución de estos sensores de lectura analógica por otros sensores que me permitiera realizar una lectura digital de ellos, por ejemplo a través de una interfaz I2C (*inter integrated circuits*).
- Incorporación de un ADC externo para leer sensores analógicos. Para ello, una opción era incoporar un módulo MCP3008, el cual a través de comunicación SPI me permitía leer un total de 8 sensores analógicos. Este módulo ADC está incluido en la librería CircuitPython de Adafruit, la cual ya estaba utilizando para leer el sensor DHT11, por lo que presentaba una manera muy simple de controlarlo.

Finalmente me decanté por la utilización del módulo MCP3008, ya que me permitía incluir 8 sensores analógicos en el sistema y necesitaba por lo menos utilizar un FC28 y varios LDR.

5.2.4. Problemas de permisos de ficheros

Al declarar la raíz del sistema como ruta de instalación de Green In House, es necesario proporcionar permisos de superusuario durante la instalación. Además, es necesario modificar archivos de sistema para poder declarar una dirección IP (*Internet Protocol*) estática para la red WiFi y para poder definir como tareas de arranque de sistema el lanzamiento de la App gráfica de la pantalla táctil, la lectura cíclica de sensores y el despliegue del servidor API REST.

Otro problema añadido a esto, es que el archivo WPAConf (el cual guarda las contraseñas de las redes WiFi almacenadas en el sistema), requería permisos de superusuario para modificarlo. La aplicación de la pantalla

táctil se lanza con permisos de usuario normal, por lo que fue necesario dar acceso a dicho archivo a todos los usuarios del sistema. Esto presenta un problema de seguridad, pero es tolerable ya que en este archivo únicamente se guardan las credenciales de la red WiFi a la que está conectada Green In House. Si alguien externo ha sido capaz de conseguir acceso a Green In House, lo ha hecho a través de la red WiFi, por lo que ya dispone de dichas credenciales.

5.2.5. Problemas al actualizar el *firmware* de la Raspberry Pi

Un problema con el que me tuve que enfrentar en dos ocasiones fue la actualización del firmware de la Raspberry Pi. Esto fue un problema porque en ambas ocasiones el programa de Green In House dejó de funcionar y de compilar, debido a que necesitaba actualizar las librerías y dependencias para que funcionasen correctamente en la nueva versión.

- La primera vez que me ocurrió este problema fue al actualizar el *firmware* de la versión 5.14 a la 6.21. Esta actualización me obligó a reinstalar todos los entornos virtuales y las dependencias de cada entorno, así como los drivers del adaptador WiFi USB que había instalado anteriormente en el sistema. Como tenía desarrollado un script para la instalación de los entornos virtuales, esta parte no me dio mayor problema. Sin embargo, reinstalar los drivers del adaptador WiFi fue algo más problemático ya que tuve que buscar una versión compatible con la versión 6 de Raspbian y tuve que probar varios drivers en versión beta hasta que encontré uno que me funcionó.
- La segunda vez que me ocurrió este problema fue al actualizar el *firmware* de la versión 6.21 a la 6.27 y en este caso, fue más complicado resolverlo, ya que el *script* de instalación que tenía desarrollado daba error de *headers*, y esto impedía que pudiera instalar nuevos módulos o actualizar los actuales. Tras intentar borrarlas y reinstalarlas sin éxito un montón de veces y de maneras diferentes, tal y como sugerían en varios foros de internet, el error seguía persistiendo. Finalmente investigando por mi cuenta en los archivos de sistema, descubrí que los archivos de *headers* se almacenaban en la ruta `/usr/src/` y la versión que tenían estos archivos era 6.21 y no 6.27. Aunque borrase los archivos de *headers* y volviera a reinstalarlas utilizando los comandos de sistema correctos, seguían descargándose la versión 6.21 y el fallo

de instalación de los módulos por las *headers* seguía persistiendo. El problema era que al buscar el archivo de **headers**, intentaba encontrar la versión 6.27 (la cual no existía en el sistema). Debido a eso se me ocurrió probar a hacer una copia a mano de estos archivos de *headers* y cambiar en el nombre de los archivos de 6.21 a 6.27. Finalmente esta solución por fin funcionó y me dejó reinstalar los entornos virtuales y las dependencias necesarias para que el código volviese a funcionar correctamente.

5.2.6. Problemas por la actualización de dependencias internas de librerías utilizadas

Durante la realización del código otro problema que sufrí fue debido a la actualización de una de las librerías que utilizaba internamente otra librería de mi programa. En concreto, la librería que me empezó a dar problemas era *connexion*, la cual utilizo en Green In House para la gestión del servidor API REST. Al intentar reinstalar el entorno virtual, me saltaba un error de que no podía regenerar las dependencias de la librería *cryptography* ya que no tenía instalado *rust*. Intenté instalar *rust* pero me decía que ya le tenía instalado en su última versión, por lo que no me dejaba actualizar el paquete, ni terminar de instalar el entorno virtual de Green In House. Como no podía actualizar *rust*, comencé a invertigar el changelog de *cryptography* y descubrí que habían sacado una nueva versión 41 que obligaba a tener instalada la versión 1.5.6 de *rust*. Creo que el problema de que no pudiera actualizar *rust* puede venir derivado del problema anterior que tuve con las *headers* que no me dejaba actualizarla a la nueva versión del sistema operativo. Finalmente conseguí resolver el problema de dependencias forzando la instalación de la librería *cryptography* en su versión 39.0.0. Tras esto, conseguir realizar la instalación de dependencias en los entornos virtuales y Green In House volvió a funcionar correctamente.

5.2.7. Necesidad de establecer una dirección IP estática

Durante las primeras pruebas que realicé tras desplegar el servidor API REST en la Raspberry Pi, accedía a dicho servidor a través de su *hostname*, lo cuál me parecía mucho más útil que hacerlo a través de su dirección IP. De esta manera me podía asegurar que cuando el usuario registrase Green In

House en su red WiFi, se iba a poder hacer las peticiones a los *endpoint* por nombre de *hostname* y no iba a haber problemas de que se pudiera solapar la IP de Green In House con la de otro dispositivo ya conectado a la red.

Esta solución funcionaba perfectamente desde mi ordenador Windows y desde mi máquina virtual Linux. El problema vino al comenzar a desarrollar la App en Flutter Flow y configurar la realización de las *API calls* a mi servidor API REST utilizando el *hostname* como base de la URL (*Uniform Resource Locator*) de las consultas. La comunicación no funcionaba. Intenté realizar una consulta desde el navegador del teléfono para ver si recuperaba el objeto JSON, pero Android no era capaz de resolver las peticiones mediante el hostname. Probé a realizar la misma petición utilizando de base de la consulta la dirección IP asignada a la Raspberry Pi y la consulta se resolvió sin problemas.

Debido a esto, tuve que configurar una IP estática para la red WiFi y desarrollar un nuevo *script* que se encargase de realizar dicha configuración automáticamente durante la instalación de Green In House, para asegurar su correcto funcionamiento. La dirección IP que establecí para Green In House es 192.168.1.240, ya que está dentro del rango por defecto de direcciones privadas de redes de clase C (utilizada comúnmente para los routers de hogares). Elegí utilizar la dirección 240 ya que es una dirección bastante alta que normalmente suele estar fuera del rango de direcciones asignables de los servidores DHCP (*Dynamic Host Configuration Protocol*) incluidos en los routers.

5.2.8. Limitaciones en las gráficas ofrecidas por Flutter Flow

A la hora de realizar la aplicación multiplataforma en Flutter Flow me encontré con que su representación de gráficos es muy limitada. Únicamente me permitía representar valores enteros en el eje Y, si intentaba representar valores decimales, aunque especificase el formato correctamente, el gráfico no se visualizaba. Otra limitación que me encontré es que en el eje X no me permitía utilizar utilizar un formato de fecha corto, ni me permitía representar las fechas para dar una idea aproximada de cuando se había realizado la muestra visualizada. Debido a esto tuve que adaptar los datos que enviaba desde el servidor API REST, enviando únicamente valores enteros para el eje Y, y formatos de fecha larga para el eje X, para que los registros quedasen ordenados por la conversión a formato *datetime* que parece realizar internamente.

Trabajos relacionados

Actualmente en el mercado ya se ofrecen diversas alternativas a este producto, aunque todas tienen un enfoque más autónomo puramente dicho, lo cuál en Green In House se ha eliminado esa parte de autosuficiencia, para involucrar lo máximo posible al usuario en el desarrollo de su planta (en este caso el niño):

Click and Grow: es un jardín interior inteligente que permite cultivar plantas en casa. Viene con cápsulas de semillas que se insertan en el jardín inteligente para un crecimiento sin problemas. Los sensores y el sistema automatizado de riego aseguran que las plantas reciban la cantidad adecuada de agua, luz y nutrientes. Los precios varían desde alrededor de 100€ hasta 200€ dependiendo del tamaño del jardín inteligente.

AeroGarden: es otro sistema de jardín interior inteligente que permite cultivar plantas en interiores durante todo el año. También utiliza semillas en cápsulas y tiene un sistema de iluminación LED ajustable y un panel de control para recordar cuándo agregar agua y nutrientes. Los precios oscilan entre 100€ y 300€ dependiendo del modelo.

Planty Square: es un jardín modular inteligente que permite a los usuarios cultivar varias plantas a la vez. Cada módulo puede cultivar una planta y los módulos se pueden conectar entre sí para formar un jardín más grande. Cuenta con una aplicación que envía notificaciones para regar las plantas. El precio ronda los 100€ .

SproutsIO: otro sistema es un jardín interior de alta tecnología que permite cultivar plantas sin tierra. Utiliza la hidroponía y la aeroponía para cultivar plantas y tiene una aplicación que permite a los usuarios monitorear y controlar su jardín desde su teléfono. El precio es más elevado, alrededor de 800€.

Es importante aclarar que estos precios pueden variar por región debido a factores como los impuestos, los costos de envío y la tasa de cambio. Todos estos productos tienen en común la idea de utilizar la tecnología para facilitar el cultivo de plantas en interiores, pero cada uno tiene sus propias características únicas, al igual que Green In House. Para determinar un precio de venta para Green In House, habría que tener en cuenta el costo de producción, las características del producto, el precio de los competidores y la disposición a pagar del público, lo cual se detallará más en la parte de anexos. En la tabla 6.1 se muestra un análisis comparativo de las funcionalidades que aporta Green In House frente a sus competidores actualmente existentes en el mercado.

| | Green In House | Click and Grow | AeroGarden | Planty Square | SproutsIO |
|-----------------------------------|-----------------------|-----------------------|-------------------|----------------------|------------------|
| Precio | Medio | Medio/Alto | Alto | Medio/Alto | Muy alto |
| Conectividad | Sí | Sí | Sí | Sí | Sí |
| Completamente automatizado | No | Sí | Sí | Sí | Sí |
| Educación Ambiental | Sí | No | No | No | No |
| Personalización | Sí | No | No | Parcial | No |
| Posibilidad de expansión | Sí | No | No | No | No |
| Open Source | Sí | No | No | No | No |

Tabla 6.1: Comparativa de Green In House con la competencia

Conclusiones y Líneas de trabajo futuras

A continuación se exponen las conclusiones obtenidas del desarrollo de Green In House y las posibles líneas de trabajo futuras para continuar mejorando Green In House.

7.1. Conclusiones

Actualmente Green In House se encuentra en una fase de desarrollo muy avanzada y se puede considerar un producto mínimo viable. Los objetivos iniciales del proyecto han sido cubiertos completamente, desarrollando un sistema capaz de recoger datos de diversos sensores electrónicos reales, almacenarlos en una base de datos, procesarlos y enviarlos como consultas de una API REST y generar nuevas instancias de elementos del sistema en la base de datos desde peticiones de la API REST. Además se ha diseñado con una estructura modular fácil de ampliar, con capacidad de generar nuevos sensores y plantas dinámicamente a través de servicios de la API REST. También cuenta con una aplicación gráfica que permite al usuario introducir las credenciales su red WiFi doméstica para que Green In House pueda comunicarse a través de red con la aplicación móvil desarrollada para visualizar gráficos de los registros del sistema en diferentes períodos de tiempo.

La realización de Green In House me ha permitido investigar diferentes tecnologías y metodologías actuales a un nivel de profundidad considerable, suficiente como para hacerme a la idea de las grandes posibilidades de diseño y desarrollo que otorgan. Aunque estoy especializado en la programación de

sistemas de automatización industrial mediante PLCs y redes de comunicación industrial para envío de datos entre dispositivos, este ha sido mi primer proyecto de automatización realizado en la plataforma Raspberry Pi. Esta primera aproximación la he realizado bastante en profundidad, teniendo que investigar cómo hacer uso de su interfaz de comunicación con sensores, sus lenguajes y protocolos soportados, cuales son las tensiones de trabajo que soporta y cómo realizar la conexión con los distintos dispositivos electrónicos que puede manejar.

Esta investigación me ha ayudado a descubrir que Raspberry Pi aporta un entorno de desarrollo ágil muy completo, al soportar Python para el desarrollo de sus aplicaciones. Además cuenta con numerosas librerías específicas de manejo de sensores aparte de las múltiples librerías soportadas por Python de carácter general para realizar tareas como despliegue de un gestor de base de datos ORM y servicios de comunicación HTML como API REST. Además, al ser un sistema con múltiples núcleos, permite la ejecución paralela de tareas mediante hilos, lo cuál es muy útil para poder realizar varias tareas simultáneamente. Raspberry Pi ha sido una gran elección para el desarrollo de Green In House, al tener todas las funcionalidades que necesitaba reunidas en una misma placa. Sin embargo, he podido descubrir, que no es el sistema más adecuado para realizar tareas críticas de lectura de sensores en tiempo real. Esto no ha afectado al proyecto ya que la lectura de los sensores de Green In House se realiza en un intervalo de minutos, pero si se realizase cada poco milisegundos, sería más óptimo utilizar Arduino para leer dichos sensores y después enviar esa información almacenada en paquetes a la Raspberry Pi.

La realización de este proyecto también me ha permitido investigar más en profundidad en el campo de las bases de datos, y los gestores ORM. Durante el grado había desarrollado alguna pequeña base de datos para algunos trabajos, pero siempre siguiendo algún esquema ya dado. Esta es la primera vez que he tenido que discernir por mi mismo cual es la mejor opción de diseño de datos para poder llevar a cabo el proyecto, cuál es la mejor forma de segmentar dichos datos en tablas individuales y cómo interrelacionarlas de manera óptima para tener todos los datos vinculados entre sí.

7.2. Líneas de trabajo futuras

La parte de *backend* está perfectamente diseñada y es completamente funcional en el estado actual, tanto para generar nuevas plantas, tipos de plantas, sensores, consejos, asociar plantas y sensores, dar de baja cualquier elemento del sistema, modificar los datos de elementos del sistema, etc., pero se pueden realizar ciertas mejoras, sobre todo en la parte del *frontend*:

- Diseñar la parte de frontend del tal manera que pueda aprovechar todas las funciones que ofrece el *backend*, de una manera intuitiva y llamativa para un niño. Por ello esta parte sería mejor desarrollarla en combinación con alguna persona que tenga un amplio conocimiento en el ámbito del trabajo con niños.
- Mejorar la aplicación de la pantalla táctil incorporada en la maceta para permitir visualizar en tiempo real los datos de los sensores.
- Incluir comprobación de contraseña de red WiFi introducida por el usuario para verificar que cumple el estándar WPA, ya que si no lo cumple, almacenar esta credencial en el sistema puede generar un problema permanente en la conectividad de la red WiFi.
- En la parte de backend se podría desarrollar la comunicación con nuevos modelos de sensores para ampliar la cantidad de factores físicos que es capaz de medir.
- Incluir seguridad en el servidor API REST desarrollando un sistema de verificación de *API KEY* o de *token*.
- Generar un sistema de notificaciones que avisase al usuario automáticamente cuando los valores de los sensores se saliesen de los rangos establecidos en los consejos de la planta.
- Intentar resolver en Flutter la dirección IP en la que está alojado el servidor API REST a través del hostname GreenInHouse, para poder habilitar el direccionamiento por DHCP y evitar posibles conflictos de red.
- Al poder generarse nuevos consejos para las plantas y modificar los existentes, también sería interesante crear una comunidad de usuarios donde pudieran compartir sus experiencias y consejos, para intentar aprender de otras personas. Esto sería muy interesante, ya que va

acorde con los objetivos sociales de Green In House y promueve una interacción activa con otras personas.

- Generar un plan de pruebas automatizado de pruebas unitarias y de pruebas de integración para poder validar más eficientemente que al añadir un nuevo módulo o funcionalidad, no se ve repercutido el correcto funcionamiento del código existente.
- Otra posible mejorar que se podría realizar sería dockerizar la aplicación de Green In House para hacer aún más sencillo su despliegue en nuevos entornos.
- Otro aspecto en el que se podría ampliar la funcionalidad de Green In House es generar otro modelo totalmente automatizado e incluir actuadores que permitan realizar el riego automáticos o dar luz a la planta. Esto desviaría el proyecto de su principal enfoque educativo y su intención de obligar a los niños a interactuar los máximo posible con la planta, para aprender de este proceso. Por ello, si se quiere desarrollar, lo mejor sería desarrollar otro modelo independiente enfocado al cultivo automatizado.

Bibliografía

- [1] Atlassian. Git — sistema de control de versiones, 2023. [Internet]. URL: <https://www.atlassian.com/es/git/tutorials/what-is-git>.
- [2] Junta de Andalucía. Patrones de diseño, 2023. [Internet]. URL: <https://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/13>.
- [3] MCI Electronics. Descarga sistema operativo raspbian, 2023. [Internet]. URL: <https://www.raspberrypi.com/software/>.
- [4] MCI Electronics. Información sobre raspberry pi, 2023. [Internet]. URL: <https://raspberrypi.cl/que-es-raspberry/>.
- [5] Flutter Flow. Flutter flow — generador de código flutter de aplicaciones móviles basado en diseño gráfico, 2023. [Internet]. URL: <https://docs.flutterflow.io/>.
- [6] Python Software Foundation. Tkinter — desarrollo de aplicaciones con entorno grafico para python, 2023. [Internet]. URL: <https://docs.python.org/es/3/library/tkinter.html>.
- [7] Python Software Foundation. venv — entorno virtual python, 2023. [Internet]. URL: <https://docs.python.org/es/3.8/library/venv.html>.
- [8] GitHub. Github — repositorio con control de versiones git, 2023. [Internet]. URL: <https://docs.github.com/es>.
- [9] Hostinger. Ssh — ejecución remota de shell segura, 2023. [Internet]. URL: <https://www.hostinger.es/tutoriales/que-es-ssh>.

- [10] Google I/O. Dart — lenguaje de programación de flutter, 2023. [Internet]. URL: <https://dart.dev/guides>.
- [11] Google I/O. Guia oficial de desarrollo flutter — framework de programación de apps multiplataforma, 2023. [Internet]. URL: <https://esflutter.dev/docs/get-started/codelab>.
- [12] Microsoft. vscode — ide ligero y potente, 2023. [Internet]. URL: <https://code.visualstudio.com/docs>.
- [13] Renee Miller. Factores que influyen en el crecimiento de las plantas, 2021. [Internet]. URL: https://www.ehowenespanol.com/principales-factores-afectan-crecimiento-plantas-info_263449/.
- [14] OpenApi. Openapi — swagger de api-rest para python, 2023. [Internet]. URL: <https://swagger.io/specification/>.
- [15] Performance. Metodología scrum, 2023. [Internet]. URL: <https://performance.io/metodologia-scrum-que-es-como-funciona/>.
- [16] Circuit Python. Adafruit circuit python — trabajo con sensores y actuadores en python, 2023. [Internet]. URL: <https://docs.circuitpython.org/en/latest/README.html>.
- [17] Seobility. Api rest — compartición sencilla de recursos por red, 2023. [Internet]. URL: https://www.seobility.net/es/wiki/API_REST.
- [18] SQLAlchemy. Sqlalchemy — sistema gestor de base de datos para python, 2023. [Internet]. URL: <https://www.sqlalchemy.org/>.
- [19] Oscar Valverde. Green in house repositorio app móvil, 2023. URL: https://github.com/ove1001/GreenInHouse_MobileApp.
- [20] Oscar Valverde. Green in house repositorio documentación, 2023. URL: https://github.com/ove1001/GreenInHouse_Doc.
- [21] Oscar Valverde. Green in house repositorio maceta, 2023. URL: https://github.com/ove1001/GreenInHouse_PlantPot.
- [22] Wikipedia. Bash — intérprete de comandos linux, 2023. [Internet]. URL: <https://es.wikipedia.org/wiki/Python>.
- [23] Wikipedia. Fase luminosa, 2023. [Internet]. URL: https://es.wikipedia.org/wiki/Fase_luminosa.

- [24] Wikipedia. Fase oscura, 2023. [Internet]. URL: https://es.wikipedia.org/wiki/Fase_oscura.
- [25] Wikipedia. Fotosíntesis, 2023. [Internet]. URL: <https://es.wikipedia.org/wiki/Fotos%C3%ADntesis>.
- [26] Wikipedia. Python — lenguaje de programación de alto nivel, 2023. [Internet]. URL: <https://es.wikipedia.org/wiki/Python>.



Este obra está bajo una licencia Creative Commons Reconocimiento 4.0 Internacional ([CC-BY-NC-4.0](#)).