



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
Tutor: Dr. Raúl Marticorena Sánchez
Dr. Antonio J. Canepa Oneto
D. Yeray Pescador Calleja (Técnico de
Emprendimiento UBU)

Índice general

Índice general	i
Índice de figuras	v
Índice de tablas	vii
Apéndice A Plan de Proyecto Software	1
A.1. Introducción	1
A.2. Planificación temporal	2
Sprint 0 (30/1/2023 - 12/2/2023)	3
Sprint 1 (13/2/2023 - 26/2/2023)	3
Sprint 2 (27/2/2023 - 12/3/2023)	4
Sprint 3 (13/3/2023 - 26/3/2023)	4
Sprint 4 (27/3/2023 - 9/4/2023)	5
SSprint 5 (10/4/2023 - 23/4/2023)	5
Sprint 6 (24/4/2023 - 14/5/2023)	6
Sprint 7 (15/5/2023 - 28/5/2023)	6
Sprint 8 (29/5/2023 - 4/6/2023)	7
Sprint 9 (5/6/2023 - 11/6/2023)	7
Sprint 10 (11/6/2023 - 18/6/2023)	8
Sprint 11 (19/6/2023 - 25/6/2023)	8
Sprint 12 (26/6/2023 - 5/7/2023)	8
A.3. Estudio de viabilidad	9
Viabilidad económica	9
Costes de desarrollo inicial	9
Costes de personal	9
Costes de hardware	10

Costes de software	10
Costes de producción de prototipo	10
Costes de documentación	11
Costes totales	11
Análisis de mercado	12
Potenciales clientes	12
Competidores	13
Plan de negocio	14
Costes de personal	15
Costes de hardware	15
Costes de software	16
Costes de servicios	16
Costes de producción por maceta	16
Costes totales	17
Conclusión de precio de la maceta educativa	18
Beneficios	18
Viabilidad legal	19
Apéndice B Especificación de Requisitos	21
B.1. Introducción	21
B.2. Objetivos generales	21
B.3. Catalogo de requisitos	22
Requisitos no funcionales	23
Requisitos funcionales	23
B.4. Especificación de requisitos	29
Apéndice C Especificación de diseño	47
C.1. Introducción	47
C.2. Diseño de datos	47
Tablas de la base de datos e interrelación entre ellas	48
Utilización de un sistema de fechas	50
C.3. Diseño procedimental	52
C.4. Diseño arquitectónico	56
Patrón de diseño MVC (Modelo-Vista-Controlador)	56
Modelo	56
Vista	57
Controlador	57
Patrón de diseño Adaptador:	58
Patrón de diseño Factoría	59
Patrón de diseño Estrategia	59
Patrón de diseño Plantilla	59

Partón de diseño Fachada	59
Partón de diseño Singleton	60
C.5. Diseño 3D	60
Apéndice D Documentación técnica de programación	62
D.1. Introducción	62
D.2. Estructura de directorios	62
D.3. Manual del programador	65
SQLAlchemy: Librería de base de datos en Python	65
Adafruit - CircuitPython: Librería de sensores y actuadores en Python	65
MCP3008: Conversor analógico a digital (ADC)	66
DHT11: Sensor de humedad y temperatura ambiente	66
FC28: Sensor de humedad de tierra de la maceta	66
LDR: Sensor de luminosidad ambiente	67
BH1750: Sensor de luminosidad ambiente	68
OpenAPI: Librería de API REST en Python	68
TKinter: Librería de interfaces gráficas en Python	69
D.4. Compilación, instalación y ejecución del proyecto	70
Instalación del sistema operativo Raspbian	70
Necesidad de instalar un adaptador WiFi USB	72
Instalación de Green In House en Raspberry Pi	73
Modificación de instalación de instalación de Green In House en Raspberry Pi	74
Scripts Bash de Green In House para automatizar funciones de desarrollo	75
Configuración del sistema operativo Raspbian	79
Utilización de Venv en Green In House	80
Configuración de SSH para desarrollo remoto en Raspberry Pi utilizando VSCode	80
Configuración de SSH en Raspberry Pi	81
Configuración de VSCode en ordenador	82
Desarrollo de aplicación móvil en Flutter	87
D.5. Pruebas del sistema	91
Apéndice E Documentación de usuario	92
E.1. Introducción	92
E.2. Requisitos de usuarios	92
E.3. Instalación	93
E.4. Manual del usuario	95

IV

Índice general

Bibliografía

100

Índice de figuras

C.1. captura de pantalla 1 del modelo 3D de Green In House.	60
C.2. captura de pantalla 2 del modelo 3D de Green In House.	61
C.3. captura de pantalla 3 del modelo 3D de Green In House.	61
D.1. captura de pantalla del Swagger de la API REST de Green In House desarrollada con OpenAPI	69
D.2. captura de pantalla de la app gráfica de Green In House desarrollada en TKinter	70
D.3. Captura de pantalla del instalador de Raspbian	71
D.4. captura de pantalla del escritorio de Raspbian	72
D.5. captura de pantalla 1 de VSCode configurando acceso por SSH.	82
D.6. captura de pantalla 2 de VSCode configurando acceso por SSH.	83
D.7. captura de pantalla 3 de VSCode configurando acceso por SSH.	83
D.8. captura de pantalla 4 de VSCode configurando acceso por SSH.	83
D.9. captura de pantalla 5 de VSCode configurando acceso por SSH.	84
D.10.captura de pantalla 6 de VSCode configurandn acceso por SSH.	84
D.11.captura de pantalla 7 de VSCode configurando acceso por SSH.	85
D.12.captura de pantalla 8 de VSCode configurando acceso por SSH.	85
D.13.captura de pantalla 9 de VSCode configurando acceso por SSH.	86
D.14.captura de pantalla 10 de VSCode configurando acceso por SSH.	86
D.15.captura de pantalla 11 de VSCode configurando acceso por SSH.	87
D.16.captura de pantalla 1 de Android Studio.	88
D.17.captura de pantalla 2 de Android Studio.	89
D.18.captura de pantalla 3 de Android Studio.	89
D.19.captura de pantalla 4 de Android Studio.	90
D.20.captura de pantalla 5 de Android Studio.	90
E.1. Conceder permisos de intalación en Android	94

E.2. Conceder permisos de instalación en Android	94
E.3. Imagen de App Android de monitorización de GreenInHouse	95
E.4. GreenInHouse no está conectado a la red WiFi.	96
E.5. Conectar GreenInHouse a red WiFi.	96
E.6. Teclado en pantalla de GreenInHouse.	97
E.7. GreenInHouse necesita reiniciar.	97
E.8. GreenInHouse conectado a red a red WiFi.	98
E.9. Gráfica de aplicación Android de GreenInHouse	98
E.10. Gráfica de aplicación Android de GreenInHouse	99

Índice de tablas

A.1. Costes de personal	9
A.2. Costes de hardware	10
A.3. Costes de software	10
A.4. Costes de producción por maceta	11
A.5. Costes de producción por maceta	11
A.6. Costes totales	12
A.7. Comparativa de Green In House con la competencia	14
A.8. Costes de personal	15
A.9. Costes de hardware	16
A.10. Costes de software	16
A.11. Costes varios	16
A.12. Costes de materiales por maceta	17
A.13. Costes de producción por maceta	17
A.14. Costes totales	18
A.15. Dependencias y versiones	19
B.1. CU-1 Creación de una entidad.	29
B.2. CU-2 Modificación de una entidad.	31
B.3. CU-3 Eliminación de una entidad.	34
B.4. CU-4 Obtención de una entidad.	37
B.5. CU-5 Obtención de varias instancias de una entidad filtrando por atributos.	40
B.6. CU-6 Lectura y registro de un sensor electrónico	42
B.7. CU-7 Introducir las credenciales de la red WiFi	43
B.8. CU-8 Graficar los datos de los sensores desde una aplicación móvil	45

Apéndice A

Plan de Proyecto Software

A continuación se procede a especificar detalladamente como se ha realizado el plan de desarrollo software de Green In House.

A.1. Introducción

La planificación es una etapa fundamental en el desarrollo de cualquier proyecto software. A continuación, se expone por qué es importante realizar una planificación adecuada:

- **Organización y estructura:** La planificación permite establecer una estructura clara y organizada para el proyecto. Define los objetivos, los recursos necesarios, los plazos de entrega y las tareas específicas que se deben realizar para llevar a cabo el proyecto. Esto ayuda a evitar confusiones y asegura que todos los miembros del equipo estén alineados en cuanto a las metas y el enfoque del proyecto.
- **Estimación de recursos y tiempos:** La planificación ayuda a determinar los recursos necesarios para llevar a cabo el proyecto, en términos de personal, de software y de presupuesto. Además, permite estimar los tiempos de ejecución de cada tarea y establecer un cronograma realista. Esto es esencial para gestionar eficientemente los recursos y evitar retrasos y sobre costes.
- **Identificación de riesgos y mitigación:** Durante la planificación, se pueden identificar posibles riesgos y obstáculos que podrían surgir durante el desarrollo del proyecto. Esto permite tomar medidas

preventivas y establecer estrategias para minimizar su impacto en el cronograma y en la calidad del proyecto. Al anticipar y abordar los riesgos de manera pro activa, se reducen las posibilidades de enfrentar problemas imprevistos en un futuro.

- **Asignación de tareas y responsabilidades:** La planificación ayuda a definir claramente las tareas específicas que deben ser realizadas y asignar responsabilidades a los miembros del equipo. Esto promueve la colaboración y la eficiencia, ya que cada persona conoce cuál es su rol en el proyecto y qué se espera de ella. Además, facilita la coordinación entre los miembros del equipo y permite realizar un seguimiento más preciso del progreso de cada tarea.
- **Control y seguimiento del progreso:** Una vez que el proyecto se ha iniciado, contar con una planificación adecuada facilita el control y seguimiento del progreso. Esto permite comparar el avance real con el planificado, identificar desviaciones y tomar medidas correctivas cuando sea necesario. Esto ayuda a garantizar que se cumplan los objetivos establecidos.

A.2. Planificación temporal

Al comenzar con el proyecto, se planteó utilizar una metodología ágil. En este caso se decidió utilizar Scrum. Aunque no se a seguido completamente en todos los aspectos que especifica esta metodología, ya que el equipo de desarrollo únicamente está formado por una persona, la cual hace todos los roles. Debido a eso, no se han realizado reuniones diarias. Las reuniones que se han realizado son las de seguimiento de los sprint, con los tutores del proyecto Raúl Martincorena y Jose Antonio Canepa y con Yeray Pescador. Durante estas reuniones se han definido los objetivos importantes a desarrollar durante el nuevo sprint, se ha hablado de los objetivos conseguidos del sprint anterior y los objetivos que no se han conseguido realizar, se han incluido de nuevo en el siguiente sprint. No se ha empleado ninguna herramienta de seguimiento de los sprint estilo ZenHUB, aunque habría sido una buena idea utilizarla para tener un mejor control del proceso iterativo del proyecto. la duración normal de los *sprints* ha sido de dos semanas, hasta llegar a las fases finales, en las que está duración ha sido modificada a una semana.

Sprint 0 (30/1/2023 - 12/2/2023)

En la reunión de este sprint inicial, lo que se realizó fue hacer una propuesta inicial de lo que sería el TFG, y hacer una lluvia de ideas sobre qué objetivos básicos se debería buscar cumplir, a qué público debería ir destinado, cuál sería la función principal del proyecto, como se haría la comunicación de los datos con sistemas externos y varios aspectos de carácter general

Como objetivos a cumplir en el primer sprint, se propusieron decidir las funcionalidades principales que se iban a implementar e investigar qué tecnologías podría utilizar para resolverlo.

Las tareas que se realizaron durante este sprint, fue la toma de decisión de que IDE utilizar, que tipo de microcontrolador utilizar (Arduino o Raspberry pi), qué factores de la planta habría que medir, cómo almacenar los datos (en archivos o en base de datos), investigar como realizar la comunicación de los datos con sistemas externos, decidir qué metodología utilizar, realizar un primer enfoque del diseño y empezar con la parte de desarrollo de la parte de qué datos utilizar.

Sprint 1 (13/2/2023 - 26/2/2023)

En la reunión de este *sprint*, lo que se hizo fue debatir entre todos, cuales de los objetivos que había pensado, sería buena idea implementar, definir una serie de funcionalidades básicas con las que empezar a trabajar y decidir el alcance inicial del proyecto para no abordar más aspectos de los necesarios. Se aceptó la utilización de Raspberry Pi como entorno de desarrollo a utilizar por su posibilidad de cubrir todos los aspectos necesarios del sistema y se decidió utilizar una base de datos como sistema de almacenamiento persistente e implementar toda la aplicación en Python al ser un lenguaje óptimo para desarrollar prototipos y tener ya conocimientos te he dicho lenguaje. Para realizar la comunicación de los datos con sistemas externos se decidió realizar a través de comunicación por red utilizando un servidor API REST

Como objetivos a cumplir en este *sprint*, se estableció realizar la instalación de el sistema operativo Raspbian en la Raspberry, iniciar con el desarrollo del modelo de datos, comenzar con la implementación de la base de datos del sistema e investigar como realizar la lectura de los sensores.

Las tareas realizadas durante este *sprint* fue la instalación de Raspbian, realizar el diseño inicial de las clases que se iban a utilizar en el sistema, definiendo sus atributos y sus jerarquías, e implementar la base de datos utilizando Python y la librería SQLAlchemy. También se comenzó con el desarrollo de los scripts de instalación del sistema para automatizar y simplificar la instalación y se investigó el uso de la librería AdaFruit para leer los sensores del sistema.

Sprint 2 (27/2/2023 - 12/3/2023)

En la reunión de este *sprint*, lo que se hizo fue analizar el modelo de datos que había diseñado para ver si cubría las necesidades de los objetivos propuestos, resolver dudas que tenía respecto al uso de SQLAlchemy para resolver las consultas a la base de datos utilizando el ORM y explicar el primer acercamiento con el trabajo con sensores en Raspberry. Se propusieron varias mejoras del modelo de datos y se estudiaron las posibilidades de las consultas mediante el ORM. Durante esta reunión se introdujo un problema que no se había contemplado anteriormente y era decidir cómo el usuario iba a introducir en el sistema las credenciales de su red WiFi.

Como objetivos a cumplir en este *sprint*, se estableció empezar con el desarrollo de la lectura de sensores utilizando AdaFruit, implementar las mejoras propuestas en el modelo de datos y en el gestor de la base de datos para realizar las consultas. También se propuso como objetivo empezar a investigar cómo desarrollar posteriormente el servidor API Rest.

Las tareas realizadas durante este *sprint* fue la implementación de las mejoras propuestas en el modelo de datos, la mejora de las consultas realizadas por el gestor de la base de datos utilizando el ORM e intentar comunicar con todos los sensores por separado, para ver como había que trabajar con cada sensor.

Sprint 3 (13/3/2023 - 26/3/2023)

En la reunión de este *sprint*, lo que se hizo fue analizar el estado actual del modelo de datos y del gestor de la base de datos, dando por buena la aproximación actual. Se debatió sobre los resultados que había obtenido con las primeras pruebas leyendo los sensores desde Raspberry Pi y los problema que había tenido para ponerlos en funcionamiento. También se

debatío sobre qué lirerías utilizar para implementar la API Rest y se decidió utilizar Swagger y Conexxon.

Como objetivos a cumplir en este *sprint*, se estableció definir el funcionamiento de los sensores y crear una interfaz general para todos ellos, la cual implemente cada sensor para sobre escribir el método de lectura en cada clase de manera específica para dicho sensor y poder trabajar con todos los sensores de la misma manera.

Las tareas que se llevaron a cabo en este *sprint* fueron las de crear una interfaz para los sensores electrónicos y una factoría que se encargase de mapear automáticamente los datos almacenados en la base de datos para determinar el objeto de sensor a instanciar en base al modelo declarado y utilizase los datos de las patillas para saber donde estaba conectado el sensor.

Sprint 4 (27/3/2023 - 9/4/2023)

En la reunión de este *sprint* se debatió la estructura que había realizado para trabajar con los sensores electrónicos, y se analizó su eficiencia y escalabilidad si necesitaba meter nuevos sensor durante el proyecto. También se debatió sobre la capa de servicio que estaba implementando para hacer uso del gestor de la base de datos y separar así la lógica del modelo de datos y se comenzó a indagar sobre qué lirerías utilizar para implementar la API Rest y se decidió utilizar Swagger y Conexxon.

Como objetivos a cumplir en este *sprint*, se estableció continuar desarrollando la capa de servicios y comenzar con el desarrollo de la API Rest, utilizando Swagger y diseñar un script en Bash para que realizase la lectura periódica de los sensores del sistema, gestionando el arranque de periódico mediante un hilo controlado por el propio sistema operativo.

Las tareas que se realizaron durante este *sprint* fueron las de terminar de crear la capa de servicios que utiliza al gestor de la base de datos y empezar a crear la API REST, encargada de gestionar la comunicación con servicios externos.

SSprint 5 (10/4/2023 - 23/4/2023)

En la reunión de este *sprint* se debatió los problemas encontrados al trabajar con sensores analógicoa, al no tener Raspberry un conversor analó-gico digital integrado y se estudió las posibles soluciones. También se habló

sobre el uso de conexión en remoto por SSH a la Raspberry ya que era muy ineficiente tener que desarrollar el código en la propia Raspberry.

Como objetivos de este *sprint* se propuso implementar una solución para leer los sensores que faltaban en el sistema, continuar con el desarrollo de la API REST e instalar SSH en la Raspberry para hacer trabajo en remoto sobre ella.

Las tareas que se realizaron durante este *sprint* fueron las de investigar e implementar alternativas para leer los sensores que estaban dando problemas, configurar la comunicación SSH, actualizar los scripts de instalación automática del sistema y continuar con el desarrollo de la API REST y la capa de servicios

Sprint 6 (24/4/2023 - 14/5/2023)

En la reunión de este *sprint* se hablo sobre como había resuelto el problema con la lectura de los sensores analógicos, si había podido establecer la comunicación por ssh. En esta reunión se planteó el problema que había tenido al realizar el cambio de red cableada a red wifi en la raspberry pi, ya que no mi Raspberry no tiene WiFi integrado, y la instalación de un USB wifi estaba dando problemas.

Como objetivos del *sprint* se propuso solucionar el problema con la conectividad WiFi, continuar con el desarrollo de la API REST y estudiar como desarrollar unos *scripts* que me permitiesen lanzar los trabajos de lecturas de sensores mediante hilos del sistema operativo.

Las tareas que se realizaron durante este *sprint* fueron las de conseguir instalar el USB WiFi para poder tener red WiFi en la Raspberri y desarrollar unos scripts que lanzasen durante el arranque del sistema operativo el servicio de lectura de sensores y el servidor API REST.

Sprint 7 (15/5/2023 - 28/5/2023)

En la reunión de este *sprint* se hablo sobre cómo había solucionado el problema con el USB WiFi y sobre el funcionamiento de los scripts de lectura y los problemas que había tenido a la hora de establecer que se lanzasen automáticamente durante el arranque. Se comenzó a debatir cómo realizar la aplicación móvil, como conectar la Raspberry a la red WiFi, y cómo enfocar la memoria.

Como objetivos del *sprint* se propuso comenzar con el desarrollo de la aplicación móvil e intentar comunicar por Bluetooth con la Raspberry para enviarle las credenciales del WiFi. También se decidió comenzar con la memoria.

Las tareas que se realizaron durante este *sprint* fueron comenzar con el desarrollo de la aplicación móvil e intentar comunicar por Bluetooth con la Raspberry y hacer uso del servidor API REST cuando tuviera conexión de red. también se comenzó con el desarrollo de la memoria.

Sprint 8 (29/5/2023 - 4/6/2023)

En la reunión de este *sprint* se habló sobre los problemas encontrados al hacer uso del Bluetooth desde Flutter Flow, y qué alternativa utilizar al no poder enviar las credenciales del WiFi por Bluetooth. se habló sobre el uso de los *endpoint* de la API Rest desde Flutter Flow y se revisó los primeros pasos de la memoria.

Como objetivos del *sprint* se propuso desarrollar una alternativa para introducir las credenciales WiFi en la Raspberry y continuar con el desarrollo de la aplicación móvil y de la memoria. También se decidió crear un sistema de consejos a seguir para el cuidado de la planta.

Las tareas que se realizaron durante este *sprint* fueron realizar una aplicación en una pantalla táctil conectada a la Raspberry que permitiera introducir las credenciales WiFi, crear el sistema de consejos de la planta, añadir más servicios de API REST y continuar con el desarrollo de la aplicación móvil y de la memoria.

S

Sprint 9 (5/6/2023 - 11/6/2023)

En la reunión de este *sprint* se habló sobre el estado de la API REST, la aplicación de la pantalla táctil para introducir las credenciales de la red WiFi, cómo graficar los datos de los registros en la aplicación móvil y los problemas que me estaba dando, cómo se había resuelto la parte de los consejos, y se continuó revisando la memoria

Como objetivos del *sprint* se propuso terminar la aplicación móvil, mostrando únicamente las gráficas de los sensores en intervalos temporales y continuar con el desarrollo de la memoria y los anexos.

Las tareas que se realizaron durante este *sprint* fueron terminar de desarrollar la aplicación móvil para que pudieran mostrar las gráficas de los sensores agrupados por intervalos temporales y continuar con la documentación de la memoria y los anexos.

Sprint 10 (11/6/2023 - 18/6/2023)

En la reunión de este *sprint* se hablo sobre el estado general de la aplicación de la maceta y del móvil, dando por terminadas estas partes. Se decidió centrar los esfuerzos en terminar la memoria y los anexos.

Como objetivos del *sprint* se propuso avanzar lo máximo posible en la documentación de la memoria y los anexos.

Las tareas que se realizaron durante este *sprint* fueron para desarrollar lo máximo posible la parte de la memoria y los anexos.

Sprint 11 (19/6/2023 - 25/6/2023)

En la reunión de este *sprint* se hablo sobre el estado actual de la memoria, se hizo una revisión de los puntos realizados y se planteó cómo enfocar los puntos faltantes.

Como objetivos del *sprint* se propuso terminar de desarrollar la memoria y avanzar lo máximo posible en la parte de anexos

Las tareas que se realizaron durante este *sprint* fueron para terminar la parte de la memoria y avanzar lo máximo posible en la parte de anexos.

Sprint 12 (26/6/2023 - 5/7/2023)

En la reunión de este ultimo *sprint* se revisó completamente de la memoria y los anexos, a falta de algunos apartados de anexos.

Como objetivos del *sprint* se propuso terminar la parte de anexos y revisar que el sistema se pudiera instalar correctamente de manera automática en un entorno limpio, utilizando los *scripts* de instalación.

Las tareas que se realizaron durante este *sprint* fueron terminar la parte de anexos y verificar que el sistema se pudiera instalar correctamente de manera automática en un entorno limpio, utilizando los *scripts* de instalación.

A.3. Estudio de viabilidad

Para poder determinar si fabricar a gran escala Green In House podría ser rentable, o no, es necesario realizar correctamente un estudio de viabilidad.

Viabilidad económica

El estudio de viabilidad económica es una etapa fundamental para determinar la rentabilidad y viabilidad financiera del proyecto Green In House. Este estudio permitirá evaluar los costos y beneficios asociados al desarrollo y funcionamiento de Green In House. Para llevar a cabo el análisis económico, es necesario considerar diferentes aspectos:

Costes de desarrollo inicial

Se deben estimar los costes asociados al desarrollo del software, la adquisición del hardware (ordenador, dispositivo móvil, impresoras 3D, Raspberry Pi, sensores y componentes electrónicos), así como los gastos de diseño y construcción de la maceta. El tiempo para esta estimación será de 6 meses.

Costes de personal

El proyecto lo realizará un desarrollador, a tiempo parcial (20 horas semanales) durante los 6 meses especificados, teniendo en cuenta el siguiente salario estimado. Estos gastos de personal se reflejan en la tabla A.1.

Tabla A.1: Costes de personal

Concepto	Coste
Salario mensual neto	606€
Retención IRPF (15 %)	165€
Seguridad Social (29.9 %)	329€
Salario mensual bruto	1100€
Total 6 meses	6600€

Costes de hardware

En este apartado se reflejan los costes que han sido necesarios, para desarrollar el dispositivo hardware que hemos necesitado para este proyecto. Estos gastos de hardware se reflejan en la tabla A.2.

Tabla A.2: Costes de hardware

Concepto	Coste	Cantidad	Coste total
Dispositivo móvil	400€	1	400€
Ordenador	1500€	1	1500€
Impresora 3D	1000€	1	1000€
Total			2900€

Costes de software

En este apartado se definen los costes de las licencias de software que no son gratuitos. Estos gastos de software se reflejan en la tabla A.3.

Tabla A.3: Costes de software

Concepto	Coste	Coste total
Windows 10 Pro	200€	200€
Flutter flow (suscripción)	30€ al mes	180€
Total		380€

Costes de producción de prototipo

En este apartado se definen los costes de materiales utilizados para realizar la producción del prototipo, sumando la amortización de gastos. Estos gastos de producción de la maceta se reflejan en la tabla A.4.

Tabla A.4: Costes de producción por maceta

Concepto	Coste	Cantidad	Coste total
Raspberry Pi	60€	1	60€
Tarjeta SD	7€	1	7€
Batería	4€	1	4€
Pantalla táctil	12€	1	12€
Cable HDMI	1,5€	1	1,5€
Cable USB	0,5€	1	0,5€
Sensor FC28	3€	1	3€
Sensor DHT11	3€	1	3€
Sensor BH1750	2€	2	4€
Cableado	0,3€ metro	5	1,5€
Plástico maceta	1€ kilo	2	2€
Impresión de maceta	0,3€ KWh	5	1,5€
Total			100,00€

Costes de documentación

En este apartado se definen los costes de los materiales de documentación. Estos gastos de documentación se reflejan en la tabla A.5.

Tabla A.5: Costes de producción por maceta

Concepto	Coste	Cantidad	Coste total
Impresión de memoria	20€	1	20€
Total			20,00€

Costes totales

En este apartado se realiza el sumatorio total de todos los gastos estimados asociados al desarrollo del Green In House. Este resumen de gastos totales se reflejan en la tabla A.6.

Tabla A.6: Costes totales

Concepto	Coste
Personal	6.600€
Hardware	2.900€
Software	380€
Fabricación prototipo	100€
Documentación	20€
Total	10.000€

Para desarrollar el proyecto el único gasto real que he tenido que afrontar son los gastos de materiales para la fabricación del prototipo, ya que he utilizado mi ordenador actual con Windows, las licencias de software han sido cedidas por el fabricante al ser un estudiante universitario, la impresora 3D ha sido facilitado su uso sin coste más allá del material gastado y no se ha tenido que pagar a ningún desarrollador, ya que esa tarea la he realizado yo.

Análisis de mercado

Para conocer la potencial demanda de Green In House hay que realizar un estudio de mercado. Esto implica analizar el tamaño del mercado, identificando a los potenciales clientes y evaluando la competencia existente.

Potenciales clientes

Green In House principalmente está diseñado y pensado para niños, pero también puede ser útil para los adultos y hay varias segmentaciones de clientes que podrían encontrar este producto interesante:

- **Jardineros aficionados:** Los adultos que disfrutan del jardín y quieren aprender más sobre el cuidado de las plantas podrían beneficiarse de las características interactivas y educativas de Green In House.
- **Amantes de la tecnología:** Aquellos que están interesados en la tecnología, especialmente la tecnología que se integra con la vida cotidiana de formas nuevas e interesantes, pueden ver el valor en un producto como Green In House.

- **Profesores y educadores:** Los profesores podrían utilizar Green In House como una herramienta educativa en las aulas para enseñar sobre ciencias naturales, ecología y biología, o para fomentar la responsabilidad y el cuidado del medio ambiente.
- **Padres:** Los padres que quieren proporcionar experiencias educativas prácticas para sus hijos en casa podrían estar interesados en Green In House.
- **Personas que viven en espacios reducidos:** Para aquellos que viven en apartamentos o casas sin jardín, Green In House ofrece una forma de tener y cuidar plantas en espacios interiores.
- **Empresas de bienestar en el trabajo:** Las empresas que buscan mejorar el bienestar de sus empleados pueden estar interesadas en utilizar Green In House para animar a los empleados a tomar descansos activos, cuidar una planta y desconectar de su trabajo por un tiempo.
- **Asistentes de terapia ocupacional y psicólogos:** Green In House podría ser utilizado como una herramienta en la terapia ocupacional o en el tratamiento de trastornos de atención, enseñando a los pacientes a concentrarse en tareas, a ser conscientes del momento presente y a cuidar de algo más.

Competidores

Actualmente en el mercado ya se ofrecen diversas alternativas a este producto, aunque todas tienen un enfoque más autónomo puramente dicho, lo cual 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. En la tabla A.7 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 A.7: Comparativa de Green In House con la competencia

Plan de negocio

Para poder definir un precio de venta al público he realizado un plan de negocio básico, sin entrar en demasiado detalle, asumiendo que el dinero de la inversión inicial se posee y no sería necesario pedir créditos. Debido a ello no se han tenido en cuenta intereses aplicados en el tiempo. Para realizar este plan he estimado los costes asociados al sueldo del desarrollador,

la adquisición del hardware (ordenador, dispositivo móvil, impresoras 3D, Raspberry Pi, sensores y componentes electrónicos), así como los gastos de diseño y construcción de las maceta. Además, se deben incluir los costos de personal (un desarrollador), los costes de alquiler de oficina, los costes de servicios como luz, gas e internet y ver como amortizar dichos gastos en el precio de las macetas educativas vendidas. Para calcular el sobrecoste derivado de la amortización de costes de fabricación que habría que aplicar al precio de los materiales de cada maceta, he estimado como tiempo de amortización de gastos iniciales total de 4 años y una producción y venta mensual de 100 macetas.

Costes de personal

El proyecto lo realizará un desarrollador, a tiempo completo, teniendo en cuenta el siguiente salario estimado. Estos gastos de personal se reflejan en la tabla A.8.

Tabla A.8: Costes de personal

Concepto	Coste	Coste amortizado
Salario mensual neto	1212€	1,01€
Retención IRPF (15 %)	330€	0,28€
Seguridad Social (29.9 %)	658€	0,55€
Salario mensual bruto	2200€	1,83€
Total 14 pagas	30800€	25,67

En el apartado de seguridad social se han sumado los costes en contingencias comunes ,desempleo, garantía salarial y la formación profesional.

Costes de hardware

En este apartado pondremos los costes que han sido necesarios, para desarrollar el dispositivo hardware que hemos necesitado para este proyecto. Estos gastos de hardware se reflejan en la tabla A.9.

Tabla A.9: Costes de hardware

Concepto	Coste	Cantidad	Coste total	Coste amortizado
Dispositivo móvil	400€	1	400€	0,08€
Ordenador	1500€	1	1500€	0,31€
Impresora 3D	1000€	5	5000€	1,04€
Total			6900€	1,44€

Costes de software

En este apartado se definen los costes de las licencias de software que no son gratuitos. Estos gastos de software se reflejan en la tabla A.10.

Tabla A.10: Costes de software

Concepto	Coste	Coste amortizado anual	Coste amortizado
Windows 10 Pro	200€	50€	0,04€
Flutter flow (suscripción)	30€ al mes	360€	0,30€
Total	230€	410€	0,34€

Costes de servicios

En este apartado se definen el resto de gastos del proyecto. Estos gastos de servicios se reflejan en la tabla A.11.

Tabla A.11: Costes varios

Concepto	Coste	Cantidad	Coste total	Coste amortizado
Alquiler de oficina	600€	12	7.200€	6,00€
Internet	40€	12	480€	0,40€
Electricidad	300€	12	3600€	3,00€
Gas	100€	12	1200€	1,00€
Total 1.040€			12.480€	10,40

Costes de producción por maceta

En este apartado se definen los costes de producción de una maceta. Primeramente se calculan los costes referentes a los gastos de materiales como se reflejan en la tabla A.12.

Tabla A.12: Costes de materiales por maceta

Concepto	Coste	Cantidad	Coste total
Raspberry Pi	60€	1	60€
Tarjeta SD	7€	1	7€
Batería	4€	1	4€
Pantalla táctil	12€	1	12€
Cable HDMI	1,5€	1	1,5€
Cable USB	0,5€	1	0,5€
Sensor FC28	3€	1	3€
Sensor DHT11	3€	1	3€
Sensor BH1750	2€	2	4€
Cableado	0,3€ metro	5	1,5€
Plástico maceta	1€ kilo	2	2€
Impresión maceta	0,3€ KWh	5	1,5€
Total			100,00€

A estos costes hay que sumarle las amortizaciones de gatos estimados, para calcular el precio de producción total asociado a cada maceta, como se refleja en la tabla A.13 .

Tabla A.13: Costes de producción por maceta

Concepto	Coste	Cantidad	Coste total
Costes de materiales	100,00€	1	100,00€
Amortización de personal	25,67€	1	25,67€
Amortización de hardware	1,44€	1	1,44€
Amortización de software	0,34€	1	0,34€
Amortización de servicios	10,40€	1	10,40€
Total			137,85€

Costes totales

En este apartado se realiza el sumatorio total de todos los gastos anuales estimados. Este resumen de gastos totales se reflejan en la tabla A.14.

Tabla A.14: Costes totales

Concepto	Coste
Personal	30.800€
Hardware	6.900€
Software	410€
Varios	12.480€
Producción 1200 macetas	120.000€
Total anual	170.590€

Conclusión de precio de la maceta educativa

Amortizando los costes de personal, de hardware, de software y de varios, en cada una de las macetas, sale que el precio para no perder dinero vendiendo las 1200 macetas estimadas al año, tendría que ser 137,85€. El objetivo de producir estas macetas, es obtener más beneficios a parte del salario que se va a cobrar como desarrollador para poder reinvertirlo en mejorar la empresa, por lo que habría que venderlas a un mayor coste. En estos costes, no están incluidos los gastos de envío, ya que dependerán de la zona a la que halla que realizar el envío y se facturarán aparte. El precio estimado de Green In House sería de 180€. Gracias a su versatilidad de software se podrían construir modelos que contengan varias plantas en un mismo macetero con sensores independientes para cada una, sin elevar demasiado los costes, ya que los mayores costes vienen del precio de la Raspberry Pi, la cual puede leer muchísimos más sensores de los que se están utilizando. Además, el precio actual de la Raspberry Pi está muy alto debido a los actuales problemas con la distribución de semiconductores, ya que su precio original era de 30€.

Beneficios

El beneficio estimado de la venta de Green in House vendiéndolo a 180€ y descontando los 137,85€ que costaría fabricar la maceta, da un beneficio de 42,15€ por unidad. Esto multiplicado por las 1200 unidades que se estiman vender, daría unos beneficios anuales de 50.585€.

Viabilidad legal

Para poder distribuir Green In House es necesario tener en cuenta que hay que cumplir con ciertos requisitos legales, entre ellos el más importante tiene que ver con el tema de la licencia con la que se suministra. Esta licencia no puede ser impuesta arbitrariamente, ya que al utilizar otras librerías es necesario adaptarla a la licencia con la que cuentan dichas librerías, teniendo que utilizar como mínimo la licencia más restrictiva de todas. Para poder realizar correctamente este análisis a continuación se expone una tabla A.15 con las dependencias que utiliza Green In House, sus versiones, su uso y su licencia de uso asociada.

Tabla A.15: Dependencias y versiones

Dependencia	Versión	Descripción	Licencia
Python	3.9	Lenguaje de programación general	PSFL
Flutter	3.7.12	Lenguaje de programación multiplataforma	BSD3-Clause
Venv	3.9	Entorno virtual Python	PSFL
SQLAlchemy	2.0.0b3	Librería Python para ORM de base de datos	MIT
SQLite3	3.34.1	Librería C de motor de base de datos SQL ligero	BSD
OpenAPI (swagger)	2.14.2	Libería Python para desarrollo de servidor API REST	MIT
Connexion	2.14.2	Libería Python para desarrollo de servidor API REST	MIT
Tkinter	0.1.0	Libería Python para desarrollo de aplicaciones gráficas	PSFL
AdaFruit Circuit Python DHT	4.0.2	Librería Python para trabajo con sensores DHT	MIT
AdaFruit Circuit Python BH1750	4.0.2	Librería Python para trabajo con sensores BH1750	MIT
AdaFruit Circuit Python MCP3XXX	4.0.2	Librería Python para trabajo con ADC MCP3XXX	MIT
Flask	2.2.5	Framework de microservicios Python	BSD3-Clause

Dependencia	Versión	Descripción	Licencia
PyYAML	6.0	Librería Python para archivos yaml	MIT
appdirs	1.4.4	Librería Python para gestión de directorios	MIT

Al revisar la tabla de dependencias utilizadas en Green In House, se aprecia que las licencias más comunes son MIT, PSFL, y BSD, las cuales son todas licencias permisivas. Estas licencias permiten la redistribución y modificación del código, incluso en software cerrado o comercial, siempre que se incluya una copia de la licencia original y los derechos de autor.

Por lo tanto, creo que la elección más acertada de licencia a utilizar para Green In House sería una licencia MIT. La razón por la que he escogido una licencia MIT es porque es simple, fácil de entender, y compatible con prácticamente todas las demás licencias. Además, permite la utilización en aplicaciones comerciales, lo cual abre la posibilidad de monetizar Green In House en el futuro si así lo decidiese. Además, aunque esta licencia permite el uso comercial de Green In House, no ofrecen ninguna garantía y no se responsabilizan de los posibles daños derivados de su uso.

Apéndice B

Especificación de Requisitos

En este apartado se recogen los requisitos especificados por el cliente para la realización de Green In House. Como es un proyecto propio, el cliente soy yo mismo, que también soy el desarrollador, pero a la vez he contado con la ayuda de mis tutores para especificar los requisitos de Green In House.

B.1. Introducción

Al principio la especificación de requisitos era bastante simple, pero a medida que fui construyendo el código, se me fueron ocurriendo nuevas funcionalidades que implementar y al final la especificación de requisitos se ha extendido bastante.

B.2. Objetivos generales

A continuación se describen los objetivos generales del proyecto:

- Leer diferentes sensores electrónicos que miden factores físicos como la temperatura, humedad y luminosidad de diferentes zonas como la maceta o el ambiente.
- Desarrollar una base de datos que almacene toda la información recolectada de los sensores y la interrelacione con la planta a la que dichos sensores están asociados a lo largo de diferentes períodos temporales

y a los consejos asociados a dicha planta. Esto permitirá 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.

- Generar una estructura modular en la que se puedan incorporar fácilmente y de manera dinámica y persistente, 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. Entre estas acciones se encuentra la de permitir al usuario introducir las credenciales de su red WiFi, para permitir a la maceta comunicarse por red.
- Implementar un sistema API REST capaz de comunicar datos por red entre el servidor alojado en la Raspberry Pi y aplicaciones externas que hagan uso de ella. Por medio de esta API REST se podrá crear en el sistema nuevos sensores, tipos de plantas, plantas y consejos asociados a ellas.
- Desarrollar una aplicación móvil 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. haciendo uso del servidor API REST.
- Generar una serie de *scripts* en Bash que permitan realizar fácilmente la instalación de la aplicación de Green In House en cualquier Raspberry Pi y su iniciación automática al encenderla.

B.3. Catalogo de requisitos

A continuación se enumeran los requisitos funcionales y no funcionales derivados de los objetivos generales del proyecto.

Requisitos no funcionales

- **RNF-1 Utilización de Raspberry:** Se utilizará una Raspberry Pi para leer los sensores, almacenar los valores y gestionar la comunicación y tratamiento de los datos.
- **RNF-2 Almacenamiento de los datos:** el almacenamiento de los datos de manera persistente se hará utilizando una base de datos y su correspondiente gestor.
 - **RNF-1.1 Borrado de los datos:** Para no violar la integridad referencial de los datos de la base de datos, no se permite borrar las instancias de los datos una vez grabadas. Para poder marcarlas como eliminadas se ha empleado un sistema de fechas, el cual determina cuando se creó la instancia y cuando se declaró como eliminada.
- **RNF-2 Utilización de Python** el código de la aplicación para la Raspberry Pi será desarrollado en Python para agilizar el desarrollo del prototipo.
- **RNF-3 Utilización de Flutter** el código de la aplicación móvil será desarrollado en Flutter al ser de desarrollo multiplataforma, para poder desplegar la misma aplicación en varios sistemas.
- **RNF-4 Comunicación de datos con otros sistemas** para que otros dispositivos puedan hacer uso de los datos recogidos por Green In House se utilizará conectividad red mediante un servidor API REST.
- **RNF-5 Conexión a red WiFi** será necesario suministrar al usuario una manera de introducir en el sistema las credenciales de la red WiFi.

Requisitos funcionales

- **RF-1 Gestión de sensores:** la aplicación tiene que ser capaz de gestionar los sensores del sistema.
 - **RF-1.1 Creación de sensores:** la aplicación tiene que ser capaz de crear nuevos sensores en el sistema, especificando la zona en la que estará ubicado el sensor, el tipo de medición que realizará, el modelo de sensor que se utilizará y los pines y dirección que se utilizarán para realizar su lectura. Para facilitar su posterior

identificación por parte del usuario se le asignará también un nombre y se almacenará la fecha en la que se dio de alta el sensor.

- **RF-1.2 Modificación de sensores:** la aplicación tiene que ser capaz de modificar los datos de los sensores existentes en el sistema.
- **RF-1.3 Eliminación de sensores:** la aplicación tiene que ser capaz de eliminar sensores existentes en el sistema.
- **RF-1.4 Obtención de sensores:** la aplicación tiene que ser capaz de recuperar los datos de los sensores existentes en el sistema.
 - **RF-1.4.1 Obtención de sensores activos:** la aplicación tiene que ser capaz de recuperar los datos de los sensores activos existentes en el sistema filtrándoles por su fecha de baja. Este filtrado se tiene que poder aplicar a todos los filtrados detallados continuación.
 - **RF-1.4.2 Obtención de sensores por tipo:** la aplicación tiene que ser capaz de recuperar los datos de los sensores existentes en el sistema filtrándoles por su tipo de medición.
 - **RF-1.4.3 Obtención de sensores por zona:** la aplicación tiene que ser capaz de recuperar los datos de los sensores existentes en el sistema filtrándoles por su zona de ubicación.
 - **RF-1.4.4 Obtención de sensores por tipo y zona:** la aplicación tiene que ser capaz de recuperar los datos de los sensores existentes en el sistema filtrándoles por su tipo de medición y su zona de ubicación.
 - **RF-1.4.5 Obtención de sensores por modelo:** la aplicación tiene que ser capaz de recuperar los datos de los sensores existentes en el sistema filtrándoles por su modelo.
- **RF-2 Gestión de plantas:** la aplicación tiene que ser capaz de gestionar las plantas del sistema.
 - **RF-2.1 Creación de plantas:** la aplicación tiene que ser capaz de crear nuevas plantas en el sistema, especificando el tipo de planta y el nombre que se asignará a dicha planta.
 - **RF-2.2 Modificación de plantas:** la aplicación tiene que ser capaz de modificar los datos de las plantas existentes en el sistema.
 - **RF-2.3 Eliminación de plantas:** la aplicación tiene que ser capaz de eliminar plantas existentes en el sistema.

- **RF-2.4 Obtención de plantas:** la aplicación tiene que ser capaz de recuperar los datos de las plantas existentes en el sistema.
 - **RF-2.4.1 Obtención de plantas activas:** la aplicación tiene que ser capaz de recuperar los datos de las plantas activas existentes en el sistema filtrándolas por su fecha de baja. Este filtrado se tiene que poder aplicar a todos los filtrados detallados continuación.
 - **RF-2.4.2 Obtención de plantas por tipo:** la aplicación tiene que ser capaz de recuperar los datos de las plantas existentes en el sistema filtrándolas por su tipo.
- **RF-3 Gestión de tipos de plantas:** la aplicación tiene que ser capaz de gestionar los tipos de plantas del sistema.
 - **RF-3.1 Creación de tipos de plantas:** la aplicación tiene que ser capaz de crear nuevas tipos de plantas en el sistema, especificando el tipo de planta y el nombre que se asignará a dicha planta.
 - **RF-3.2 Modificación de tipos de plantas:** la aplicación tiene que ser capaz de modificar los datos de los tipos de plantas existentes en el sistema.
 - **RF-3.3 Obtención de tipos de plantas:** la aplicación tiene que ser capaz de recuperar los datos de los tipos de plantas existentes en el sistema.
- **RF-4 Gestión de asociaciones entre sensores y plantas:** la aplicación tiene que ser capaz de gestionar las asociaciones entre los sensores y las plantas del sistema.
 - **RF-4.1 Creación de asociación sensor-planta:** la aplicación tiene que ser capaz de crear nuevas asociaciones entre los sensores y las plantas existentes en el sistema, especificando el sensor y la planta asociados que se quieren asociar.
 - **RF-4.2 Eliminación de asociación sensor-planta:** la aplicación tiene que ser capaz de eliminar las asociaciones existentes entre los sensores y las plantas del sistema, especificando el sensor y la planta asociados que se quieren desasociar.
 - **RF-4.3 Obtención de asociación sensor-planta:** la aplicación tiene que ser capaz de obtener las asociaciones existentes entre los sensores y las plantas del sistema, especificando el sensor y la planta asociados que se quiere modificar.

- **RF-4.3.1 Obtención de asociaciones sensor-planta activas:** la aplicación tiene que ser capaz de recuperar las asociaciones activas entre sensores y plantas existentes en el sistema filtrándolas por su fecha de baja. Este filtrado se tiene que poder aplicar a todos los filtrados detallados continuación.
 - **RF-4.3.2 Obtención de todos los sensores asociados a una planta:** la aplicación tiene que ser capaz de obtener todos los sensores asociados a una planta del sistema, especificando la planta de la cual se quieren obtener los sensores.
 - **RF-4.3.3 Obtención de todas las plantas asociadas a un sensor:** la aplicación tiene que ser capaz de obtener todas las plantas asociadas a un sensor del sistema, especificando el sensor de la cual se quieren obtener las plantas.
- **RF-5 Gestión de registros de sensores:** la aplicación tiene que ser capaz de gestionar los registros de sensores del sistema.
 - **RF-5.1 Creación de registros de sensores:** la aplicación tiene que ser capaz de crear nuevos registros de sensores en el sistema, especificando el sensor que generó el registro, el valor del sensor y la unidad de medida del registro.
 - **RF-5.2 Obtención de registros de sensores:** la aplicación tiene que ser capaz de recuperar los datos de los registros de sensores existentes en el sistema.
 - **RF-5.2.1 Obtención de registros de sensores generados entre determinadas fechas:** la aplicación tiene que ser capaz de recuperar los datos de los registros de sensores existentes en el sistema filtrándolos por su fecha de creación. Este filtrado se tiene que poder aplicar a todos los filtrados detallados continuación.
 - **RF-5.2.2 Obtención de registros de sensores en formato para graficar:** la aplicación tiene que ser capaz de recuperar los datos de los registros de sensores existentes en el sistema agrupándolos en dos listas (fechas para el eje X y valores para el eje Y) para poder graficarlos fácilmente. Este filtrado se tiene que poder aplicar a todos los filtrados detallados continuación.
 - **RF-5.2.3 Obtención de registros de sensores por sensor:** la aplicación tiene que ser capaz de recuperar los datos de

los registros de sensores existentes en el sistema filtrándoles por el sensor que generó dicho registro.

- **RF-5.2.4 Obtención de registros de sensores por planta:** la aplicación tiene que ser capaz de recuperar los datos de los registros de sensores existentes en el sistema filtrándoles por la planta a la que están asociados.
- **RF-6 Gestión de consejos de plantas:** la aplicación tiene que ser capaz de gestionar los consejos de plantas del sistema.
 - **RF-6.1 Creación de consejos de plantas:** la aplicación tiene que ser capaz de crear nuevos consejos de plantas en el sistema, el tipo de medida, la zona de ubicación, la unidad de medida y los valores mínimos y máximos.
 - **RF-6.2 Modificación de consejos de plantas:** la aplicación tiene que ser capaz de modificar los datos de los consejos de plantas existentes en el sistema.
 - **RF-6.3 Obtención de consejos de plantas:** la aplicación tiene que ser capaz de recuperar los datos de los consejos de plantas existentes en el sistema.
 - **RF-6.3.1 Obtención de consejos de plantas por planta:** la aplicación tiene que ser capaz de recuperar los datos de los consejos de plantas existentes en el sistema filtrándoles por la planta a la que están asociados. Este filtrado se tiene que poder aplicar a todos los filtrados detallados continuación.
 - **RF-6.3.2 Obtención de consejos de plantas por zona:** la aplicación tiene que ser capaz de recuperar los datos de los consejos de plantas existentes en el sistema filtrándoles por la zona a la que están asociados.
 - **RF-6.3.3 Obtención de consejos de plantas por tipo de medida:** la aplicación tiene que ser capaz de recuperar los datos de los consejos de plantas existentes en el sistema filtrándoles por el tipo de medida a la que están asociados.
- **RF-7 Gestión de consejos de tipos de plantas:** la aplicación tiene que ser capaz de gestionar los consejos de tipos de plantas del sistema.
 - **RF-7.1 Creación de consejos de tipos de plantas:** la aplicación tiene que ser capaz de crear nuevos consejos de tipos de plantas en el sistema, el tipo de medida, la zona de ubicación, la unidad de medida y los valores mínimos y máximos.

- **RF-7.2 Modificación de consejos de tipos de plantas:** la aplicación tiene que ser capaz de modificar los datos de los consejos de tipos de plantas existentes en el sistema.
- **RF-7.3 Obtención de consejos de tipos de plantas:** la aplicación tiene que ser capaz de recuperar los datos de los consejos de tipos de plantas existentes en el sistema.
 - **RF-7.3.1 Obtención de consejos de tipos de plantas por planta:** la aplicación tiene que ser capaz de recuperar los datos de los consejos de tipos de plantas existentes en el sistema filtrándoles por el tipo de planta a la que están asociados. Este filtrado se tiene que poder aplicar a todos los filtrados detallados continuación.
 - **RF-7.3.2 Obtención de consejos de tipos de plantas por zona:** la aplicación tiene que ser capaz de recuperar los datos de los consejos de tipos de plantas existentes en el sistema filtrándoles por la zona a la que están asociados.
 - **RF-7.3.3 Obtención de consejos de tipos de plantas por tipo de medida:** la aplicación tiene que ser capaz de recuperar los datos de los consejos de tipos de plantas existentes en el sistema filtrándoles por el tipo de medida a la que están asociados.
- **RF-8 Lectura de sensores electrónicos:** Se tienen que poder leer sensores electrónicos conectados al sistema, utilizando los datos almacenados de los sensores y generando registros de sesores con los valores leídos.
- **RF-9 Conexión a red WiFi:** Se tiene que poder introducir las credenciales de la red WiFi a conectar.
- **RF-10 Graficar los datos de los sensores desde una aplicación móvil:** Se tiene que poder graficar desde una aplicación móvil los registros de los sensores agrupándoles en diferentes periodos temporales.

B.4. Especificación de requisitos

Este caso de uso es aplicable a todas las entidades del sistema. Únicamente cambiarían los datos suministrados para realizar la creación de la instancia de la entidad respectiva y variaría el tipo de objeto generado, correspondiente al tipo de entidad creada, así como los datos del archivo JSON devuelto al cliente.

Tabla B.1: CU-1 Creación de una entidad.

CU-1	Creación de una instancia de una entidad
Versión	1.0
Autor	Oscar Valverde Escobar
Requisitos asociados	RF-1.1, RF-2.1, RF-3.1, RF-4.1, RF-5.1, RF-6.1, RF-7.1
Descripción	Operaciones realizadas para crear una nueva instancia de una entidad de manera persistente en la base de datos de la aplicación.
Precondición	<ul style="list-style-type: none"> ■ El sistema tiene que estar conectado a la red WiFi y tener el servidor API REST funcionando. ■ La instancia de la entidad existe en la base de datos. ■ Se dispone de todos los datos necesarios para eliminar la entidad. ■ Si contiene referencias a otras entidades, tienen que existir dichas instancias de las entidades en la base de datos. ■ El almacenamiento del sistema no está lleno

Sigue en la página siguiente.

CU-1	Creación de una instancia de una entidad
Acciones	<ol style="list-style-type: none">1. Suministrar al <i>endpoint</i> de la API REST los datos necesarios para crear la entidad correspondiente .2. Verificar que dichos datos son válidos.3. Comunicar la petición de la API REST a la capa de servicios para que resuelva como realizar la creación.4. Comunicar la petición de la capa de servicios al manejador de la base de datos para que resuelva como realizar la creación.5. Realizar la creación del objeto en la base de datos.6. Devolución del objeto creado por el manejador de la base de datos a la capa de servicios.7. Devolución del objeto creado por la capa de servicios a la API REST.8. Devolución al cliente desde la API REST de un archivo JSON con los atributos del objeto creado por la capa de servicios junto con un código HTML de estado de la operación.
Postcondición	Existirá una nueva instancia de entidad en la tabla correspondiente de la base de datos.
Excepciones	<ol style="list-style-type: none">1. EntidadExiste2. Código HTML de error
Importancia	Alta

Este caso de uso es aplicable a todas las entidades del sistema. Únicamente cambiarían los datos suministrados para realizar la modificación de la instancia de la entidad respectiva y variaría el tipo de objeto generado, correspondiente al tipo de entidad modificada, así como los datos del archivo JSON devuelto al cliente.

Tabla B.2: CU-2 Modificación de una entidad.

CU-2	Modificación de una instancia de una entidad
Versión	1.0
Autor	Oscar Valverde Escobar
Requisitos asociados	RF-1.2, RF-2.2, RF-3.2, RF-6.2, RF-7.2
Descripción	Operaciones realizadas para modificar una instancia de entidad de manera persistente en la base de datos de la aplicación.
Precondición	

- El sistema tiene que estar conectado a la red WiFi y tener el servidor API REST funcionando.
- La instancia de la entidad tiene que existir en la base de datos.
- Se dispone de todos los datos necesarios para obtener y modificar la entidad.
- Si contiene referencias a otras entidades, tienen que existir dichas instancias de las entidades en la base de datos.

Sigue en la página siguiente.

CU-2	Modificación de una instancia de una entidad
Acciones	

1. Suministrar al *endpoint* de la API REST los datos necesarios para recuperar y modificar la entidad correspondiente.
2. Verificar que dichos datos son válidos.
3. Comunicar la petición de la API REST a la capa de servicios para que resuelva como realizar la modificación de la instancia de la entidad.
4. Comunicar la petición de la capa de servicios al manejador de la base de datos para que resuelva como realizar la recuperación y modificación de la instancia de la entidad.
5. Realizar la obtención de la instancia de la entidad de la base de datos y generar su objeto correspondiente.
6. Realizar la modificación del objeto generado.
7. Realizar el grabado en la base de datos del objeto modificado.
8. Devolución del objeto modificado por el manejador de la base de datos a la capa de servicios.
9. Devolución del objeto modificado por la capa de servicios a la API REST.
10. Devolución al cliente desde la API REST de un archivo JSON con los atributos del objeto modificado por la capa de servicios junto con un código HTML de estado de la operación.

Postcondición

- Existirá el mismo número de instancias de entidad en la tabla correspondiente de la base de datos.
- Los datos de la instancia especificada de la entidad habrán sido modificados en la tabla correspondiente de la base de datos.

Sigue en la página siguiente.

CU-2 **Modificación de una instancia de una entidad**
Excepciones

1. EntidadNoExiste
2. Código HTML de error

Importancia Alta

Este caso de uso es aplicable a todas las entidades del sistema. Únicamente cambiarían los datos suministrados para realizar la eliminación de la instancia de la entidad respectiva y variaría el tipo de objeto generado, correspondiente al tipo de entidad eliminado, así como los datos del archivo JSON devuelto al cliente.

Tabla B.3: CU-3 Eliminación de una entidad.

CU-3	Eliminación de una instancia de una entidad
Versión	1.0
Autor	Oscar Valverde Escobar
Requisitos asociados	RF-1.3, RF-2.3, RF-4.2
Descripción	Operaciones realizadas para dar de baja una instancia de una entidad de manera persistente en la base de datos de la aplicación.
Precondición	<ul style="list-style-type: none"> ■ El sistema tiene que estar conectado a la red WiFi y tener el servidor API REST funcionando. ■ La instancia de la entidad tiene que existir en la base de datos. ■ Se dispone de todos los datos necesarios para dar de baja la entidad. ■ Si contiene referencias a otras entidades, tienen que existir dichas instancias de las entidades en la base de datos.

Sigue en la página siguiente.

CU-3	Eliminación de una instancia de una entidad
Acciones	

1. Suministrar al *endpoint* de la API REST los datos necesarios eliminar la entidad correspondiente.
2. Verificar que dichos datos son válidos.
3. Comunicar la petición de la API REST a la capa de servicios para que resuelva como realizar la eliminación de la instancia de la entidad.
4. Comunicar la petición de la capa de servicios al manejador de la base de datos para que resuelva como realizar la eliminación de la instancia de la entidad.
5. Realizar la obtención de la instancia de la entidad de la base de datos y generar su objeto correspondiente.
6. Dar de baja la instancia de la entidad en la base de datos asignando al objeto generado la fecha en la que se realizó esta operación.
7. Realizar el grabado en la base de datos del objeto modificado.
8. Devolución del objeto modificado por el manejador de la base de datos a la capa de servicios.
9. Devolución del objeto modificado por la capa de servicios a la API REST.
10. Devolución al cliente desde la API REST de un archivo JSON con los atributos del objeto dado de baja por la capa de servicios junto con un código HTML de estado de la operación.

Postcondición

- Existirá el mismo número de instancias de entidad en la tabla correspondiente de la base de datos.
- Los datos de fecha de baja de la instancia especificada de la entidad habrán sido modificados en la tabla correspondiente de la base de datos.

Sigue en la página siguiente.

CU-3	Eliminación de una instancia de una entidad
Excepciones	<ol style="list-style-type: none">1. EntidadNoExiste2. Código HTML de error
Importancia	Alta

Este caso de uso es aplicable a todas las entidades del sistema. Únicamente cambiarían los datos suministrados para realizar la modificación de la instancia de la entidad respectiva y variaría el tipo de objeto generado, correspondiente al tipo de entidad modificada, así como los datos del archivo JSON devuelto al cliente.

Tabla B.4: CU-4 Obtención de una entidad.

CU-4	Obtención de una entidad
Versión	1.0
Autor	Oscar Valverde Escobar
Requisitos asociados	RF-1.4, RF-2.4, RF-3.3, RF-4.3, RF-5.2, RF-6.3, RF-7.3
Descripción	Operaciones realizadas para obtener una entidad de la base de datos de la aplicación.
Precondición	<ul style="list-style-type: none"> ■ El sistema tiene que estar conectado a la red WiFi y tener el servidor API REST funcionando. ■ La instancia de la entidad existe en la base de datos. ■ Se dispone de todos los datos necesarios para eliminar la entidad. ■ Si contiene referencias a otras entidades, tienen que existir dichas entidades en la base de datos.

Sigue en la página siguiente.

CU-4	Obtención de una entidad
Acciones	

1. Suministrar al *endpoint* de la API REST los datos necesarios para obtener la instancia de la entidad correspondiente .
2. Verificar que dichos datos son válidos.
3. Comunicar la petición de la API REST a la capa de servicios para que resuelva como realizar la obtención de la instancia de la entidad.
4. Comunicar la petición de la capa de servicios al manejador de la base de datos para que resuelva como realizar la recuperación de la instancia.
5. Realizar la recuperación de la instancia de la entidad en la base de datos.
6. Devolución del objeto recuperado por el manejador de la base de datos a la capa de servicios.
7. Devolución del objeto recuperado por la capa de servicios a la API REST.
8. Devolución al cliente desde la API REST de un archivo JSON con los atributos del objeto obtenido por la capa de servicios junto con un código HTML de estado de la operación.

Postcondición

- Existirá el mismo número de instancias de entidad en la tabla correspondiente de la base de datos.
- Los datos de la instancia especificada de la entidad no habrán variado en la tabla correspondiente de la base de datos.
- Se habrá devuelto un archivo JSON con los datos del objeto correspondiente

Sigue en la página siguiente.

CU-4	Obtención de una entidad
Excepciones	<ol style="list-style-type: none">1. EntidadNoExiste2. Código HTML de error
Importancia	Alta

Este caso de uso es aplicable a todas las entidades del sistema. Únicamente cambiarían los datos suministrados para realizar la modificación de la instancia de la entidad respectiva y variaría el tipo de objeto generado, correspondiente al tipo de entidad modificada, así como los datos del archivo JSON devuelto al cliente. Este caso de uso es una extensión de la funcionalidad del caso de uso 4, permitiendo recuperar varias instancias de una entidad, en lugar de una única instancia, filtrando las instancias a recuperar por alguno de sus atributos.

Tabla B.5: CU-5 Obtención de varias instancias de una entidad filtrando por atributos.

CU-5	Obtención de varias instancias de una entidad filtrando por atributos
Versión	1.0
Autor	Oscar Valverde Escobar
Requisitos asociados	RF-1.4.1, RF-1.4.2, RF-1.4.3, RF-1.4.4, RF-1.4.5, RF-2.4.1, RF-2.4.2, RF-4.3.1, RF-4.3.2, RF-4.3.3, RF-5.2.1, RF-5.2.2, RF-5.2.3, RF-5.2.4, RF-6.3.1, RF-6.3.2, RF-6.3.3, RF-7.3.1, RF-7.3.2, RF-7.3.3
Descripción	Operaciones realizadas para obtener una entidad de la base de datos de la aplicación.
Precondición	<ul style="list-style-type: none"> ■ El sistema tiene que estar conectado a la red WiFi y tener el servidor API REST funcionando. ■ Existe en la base de datos instancias de las entidades que cumplen las cumplen los filtros establecidos. ■ Se dispone de todos los datos necesarios para obtener las entidades. ■ Si contiene referencias a otras entidades, tienen que existir dichas entidades en la base de datos.

Sigue en la página siguiente.

CU-5	Obtención de varias instancias de una entidad filtrando por atributos
Acciones	
	<ol style="list-style-type: none">1. Suministrar al <i>endpoint</i> de la API REST los datos necesarios para obtener las instancias de la entidad correspondiente (entidad a recuperar y filtros a aplicar).2. Verificar que dichos datos son válidos.3. Comunicar la petición de la API REST a la capa de servicios para que resuelva como realizar la obtención de las instancias de la entidad correspondiente a recuperar aplicando los filtros establecidos.4. Comunicar la petición de la capa de servicios al manejador de la base de datos para que resuelva como realizar la recuperación de las instancias de la entidad correspondiente aplicando los filtros establecidos.5. Realizar la recuperación de las instancias de la entidad correspondiente en la base de datos filtros establecidos.6. Devolución de la lista de objetos recuperados por el manejador de la base de datos a la capa de servicios.7. Devolución de la lista de objetos recuperados por la capa de servicios a la API REST.8. Devolución al cliente desde la API REST de un archivo JSON con los atributos de la lista de objetos obtenidos por la capa de servicios junto con un código HTML de estado de la operación.

Sigue en la página siguiente.

CU-5	Obtención de varias instancias de una entidad filtrando por atributos
-------------	--

Postcondición

- Existirá el mismo número de instancias de entidad en la tabla correspondiente de la base de datos.
- Los datos de la instancia especificada de la entidad no habrán variado en la tabla correspondiente de la base de datos.
- Se habrá devuelto un archivo JSON con los datos del objeto correspondiente

Excepciones

1. EntidadNoExiste
2. Código HTML de error

Importancia	Media
--------------------	-------

Tabla B.6: CU-6 Lectura y registro de un sensor electrónico

CU-6	Lectura y registro de un sensor electrónico
Versión	1.0
Autor	Oscar Valverde Escobar
Requisitos asociados	RF-1.4, RF-5.1, RF-8
Descripción	Operaciones realizadas para obtener un sensor de la base de datos, leer su valor real .

Sigue en la página siguiente.

CU-6	Lectura y registro de un sensor electrónico
-------------	--

Precondición

- El sistema tiene que tener los sensores conectados correctamente.
- El sistema tiene que estar encendido
- Existe en la base de datos instancias de los sensores conectado con todos sus datos.
- Se dispone de todos los datos necesarios para obtener el sensor.
- El almacenamiento del sistema no está lleno.

Acciones

1. Se solicita al sistema obtener los datos de un sensor.
2. Se realiza la lectura de dicho sensor
3. Se genera un registro del valor leído asociado al sensor
4. Se almacena de manera persistente el registro generado

Postcondición

- Existira una instancia nueva en la tabla de registros de sensores con los valores recogidos.

Excepciones

1. SensorNoExiste

Importancia Alta

Tabla B.7: CU-7 Introducir las credenciales de la red WiFi

CU-7	Introducir las credenciales de la red WiFi
Versión	1.0
Autor	Oscar Valverde Escobar

Sigue en la página siguiente.

CU-7	Introducir las credenciales de la red WiFi
Requisitos asociados	RF-9
Descripción	Operaciones realizadas para conectar el sistema a una red WiFi.
Precondición	<ul style="list-style-type: none"> ■ El sistema tiene que estar dentro del alcance de la red WiFi. ■ El sistema tiene que estar encendido ■ La pantalla táctil tiene que funcionar ■ La app gráfica tiene que estar lanzada
Acciones	<ol style="list-style-type: none"> 1. Mediante la pantalla tactil se accede a la configuración del wifi. 2. Se escribe el nombre de la red WiFi 3. Se escribe la contraseña de la red WiFi 4. Se pulsa en aceptar 5. Se guardan los datos de la red wifi y pide reiniciar 6. Se reinicia el sistema
Postcondición	<ul style="list-style-type: none"> ■ Al encender el sistema indicará que está conectado a la red WiFi
Excepciones	<ol style="list-style-type: none"> 1. Red no accesible
Importancia	Alta

Tabla B.8: CU-8 Graficar los datos de los sensores desde una aplicación móvil

CU-8	Graficar los datos de los sensores desde una aplicación móvil
Versión	1.0
Autor	Oscar Valverde Escobar
Requisitos asociados	RF-10
Descripción	Operaciones realizadas para obtener un sensor de la base de datos, leer su valor real y almacenarlo de manera persistente.
Precondición	<ul style="list-style-type: none"> ■ El sistema tiene que estar conectado a la misma red WiFi que el dispositivo móvil. ■ El sistema tiene que estar encendido ■ el Servidor API REST tiene que estar funcionando
Acciones	<ol style="list-style-type: none"> 1. La App móvil utiliza un <i>endpoint</i> de la API REST para solicitarle los datos de los sensores a graficar. 2. El sistema recupera todos los registros de datos solicitados. 3. El sistema procesa los registros de datos por intervalos para calcular su valor medio en dichos intervalos. 4. El sistema empaqueta en una lista de valores medios y otra lista de intervalos temporales. 5. El servidor API REST convierte el objeto generado con las listas en un archivo JSON y se lo envía a la App móvil 6. La app móvil recibe el archivo con las listas de datos 7. La app móvil grafica las listas de datos recibidas.

Sigue en la página siguiente.

CU-8 Graficar los datos de los sensores desde una aplicación móvil

Postcondición

- El número y valor de las instancias en la base de datos debe ser el mismo tras el procesado de valores medios.
- Se tiene que haber mostrado una gráfica en el móvil con los datos recibidos

Excepciones

1. Red no accesible

Importancia media

Apéndice C

Especificación de diseño

A continuación se detalla el diseño que se ha generado para realizar Green In House y se detalla en profundidad porque se ha decidido generar dicho diseño, que metodologías se han utilizado, que estructura de trabajos y de directorias se ha definido, etc.

C.1. Introducción

Realizar un buen diseño de las especificaciones del sistema supone una enorme diferencia a la hora de realizar la codificación y el posterior mantenimiento y crecimiento de la aplicación. Si se parte de un mal diseño, habrá que ir haciendo cambios continuamente para poder ir incluyendo funcionalidades. Por contra, si se parte de un buen diseño, que esté pensado para ser modular y crecer continuamente, la inclusión de nuevas funcionalidades se realizará de manera sencilla y orgánica, si necesidad de generar cambios extraños o refactorizar el código de la aplicación completa.

C.2. Diseño de datos

A continuación se detalla la estructura que tiene cada tabla de la base de datos y se especifica como se interrelacionan entre ellas.

Tablas de la base de datos e interrelación entre ellas

Las tablas que se han diseñado para utilizar en Green In House y las interrelaciones que se han creado entre ellas son las siguientes:

- **tipos_plantas**

- tipo_planta {String} **PK**: El tipo de planta.
- descripcion_planta {String}: Descripción del tipo de planta.

- **plantas**

- nombre_planta {String} **PK**: El nombre de la planta.
- tipo_planta {String} **FK** de *tipos_plantas.tipo_planta*: El tipo de la planta.
- fecha_plantacion {Timestamp}: La fecha de plantación de la planta.
- fecha_marchitacion {Timestamp}: La fecha de marchitación de la planta.

- **sensores**:

- tipo_sensor {Enum type} **PK**: El tipo de sensor.
- zona_sensor {Enum type} **PK**: La zona del sensor.
- numero_sensor {Integer} **PK**: El número del sensor.
- modelo_sensor {Enum type}: El modelo del sensor.
- nombre_sensor {String}: El nombre del sensor para reconocerle fácilmente.
- direccion_lectura {String}: La dirección de lectura del sensor.
- patilla_0_lectura {Integer}: Pin 0 de lectura del sensor.
- patilla_1_lectura {Integer}: Pin 1 de lectura del sensor.
- patilla_2_lectura {Integer}: Pin 2 de lectura del sensor.
- patilla_3_lectura {Integer}: Pin 3 de lectura del sensor.
- unidad_medida_0 {Enum type}: Unidad de medida 0 para los datos recogidos por el sensor.
- unidad_medida_1 {Enum type}: Unidad de medida 1 para los datos recogidos por el sensor.

- unidad_medida_2 {Enum type}: Unidad de medida 2 para los datos recogidos por el sensor.
- unidad_medida_3 {Enum type}: Unidad de medida 3 para los datos recogidos por el sensor.
- fecha_creacion {Timestamp}: Fecha de creación del sensor.
- fecha_eliminacion {Timestamp}: Fecha de eliminación del sensor.

- **sensores_plantas**

- id_ {Integer} **PK**: Un identificador del registro de unión de sensor y planta.
- tipo_sensor {Enum type} **FK** de *sensores.tipo_sensor*: El tipo de sensor.
- zona_sensor {Enum type} **FK** de *sensores.zona_sensor*: La zona del sensor.
- numero_sensor {Integer} **FK** de *sensores.numero_sensor*: El número del sensor.
- nombre_planta {String} **FK** de *plantas.nombre_planta*: El nombre de la planta asociada al sensor.
- fecha_asociacion {Timestamp}: Fecha de asociación del sensor a la planta.
- fecha_anulacion {Timestamp}: Fecha de anulación de la asociación del sensor a la planta.

- **registros_sensores**

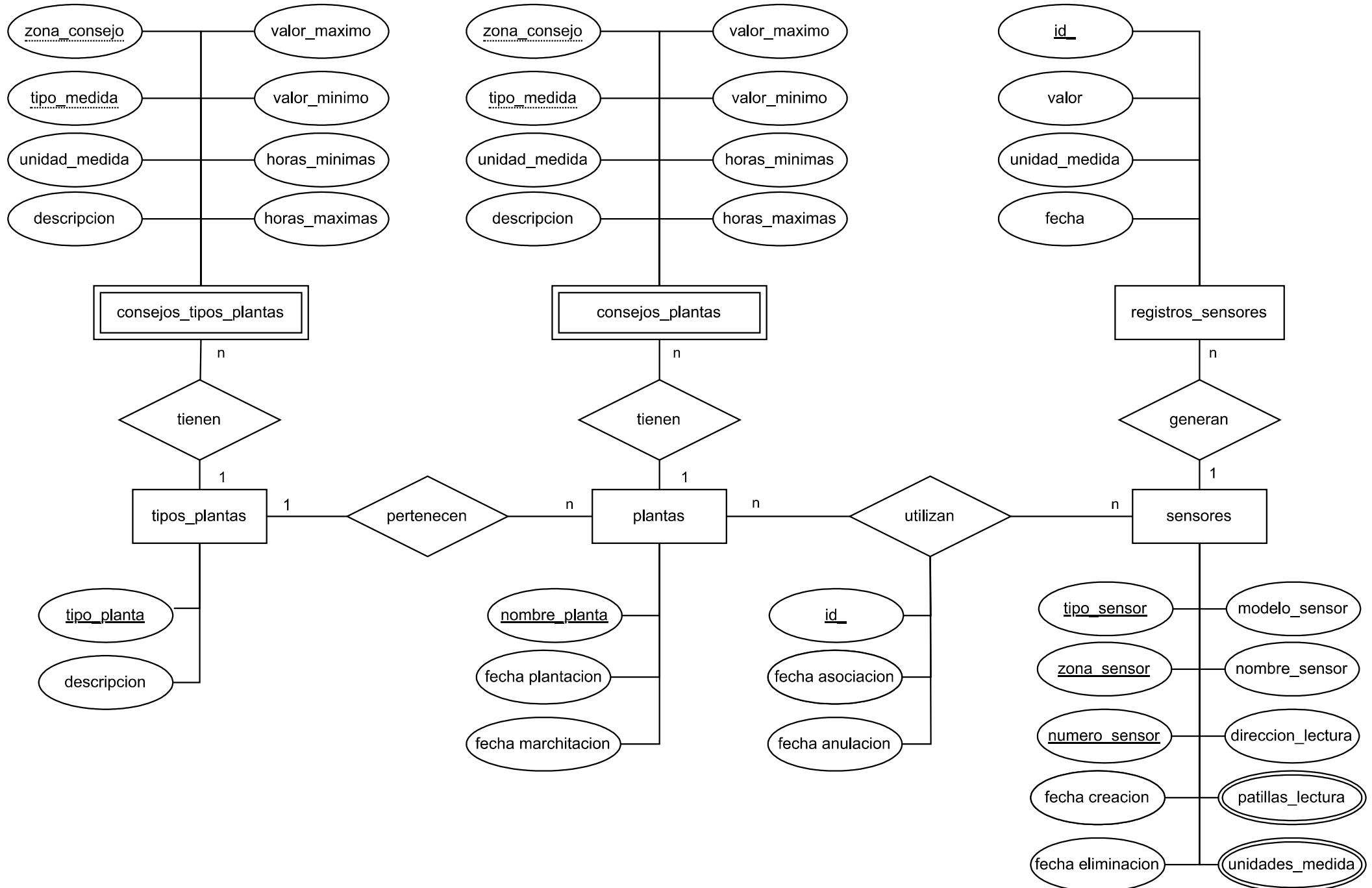
- id_ {Integer} **PK**: El identificador del registro del sensor.
- tipo_sensor {Enum type} **FK** de *sensores.tipo_sensor*: El tipo de sensor.
- zona_sensor {Enum type} **FK** de *sensores.zona_sensor*: La zona del sensor.
- numero_sensor {Integer} **FK** de *sensores.numero_sensor*: El número del sensor.
- valor {Float}: El valor registrado por el sensor.
- unidad_medida {Enum type}: La unidad de la medida.
- fecha {Timestamp}: La fecha del registro.

- **consejos_plantas**

- descripcion {String}: Una descripción del consejo.
 - nombre_elemento {String} **PK FK** de *plantas.nombre_planta*: El nombre del elemento que se refiere al consejo.
 - zona_consejo {Enum type} **PK**: La zona del sensor.
 - tipo_medida {Enum type} **PK**: El tipo de medida.
 - unidad_medida {Enum type}: La unidad de la medida.
 - valor_minimo {Float}: El valor mínimo de la medida.
 - valor_maximo {Float}: El valor máximo de la medida.
 - horas_minimas {Integer}: Las horas mínimas de la medida.
 - horas_maximas {Integer}: Las horas máximas de la medida.
- **consejos_tipos_plantas**
- descripcion {String}: Una descripción del consejo.
 - nombre_elemento {String} **PK FK** de *tipos_plantas.tipo_planta*: El nombre del elemento que se refiere al consejo.
 - zona_consejo {Enum type} **PK**: La zona del sensor.
 - tipo_medida {Enum type} **PK**: El tipo de medida.
 - unidad_medida {Enum type}: La unidad de la medida.
 - valor_minimo {Float}: El valor mínimo de la medida.
 - valor_maximo {Float}: El valor máximo de la medida.
 - horas_minimas {Integer}: Las horas mínimas de la medida.
 - horas_maximas {Integer}: Las horas máximas de la medida.

Utilización de un sistema de fechas

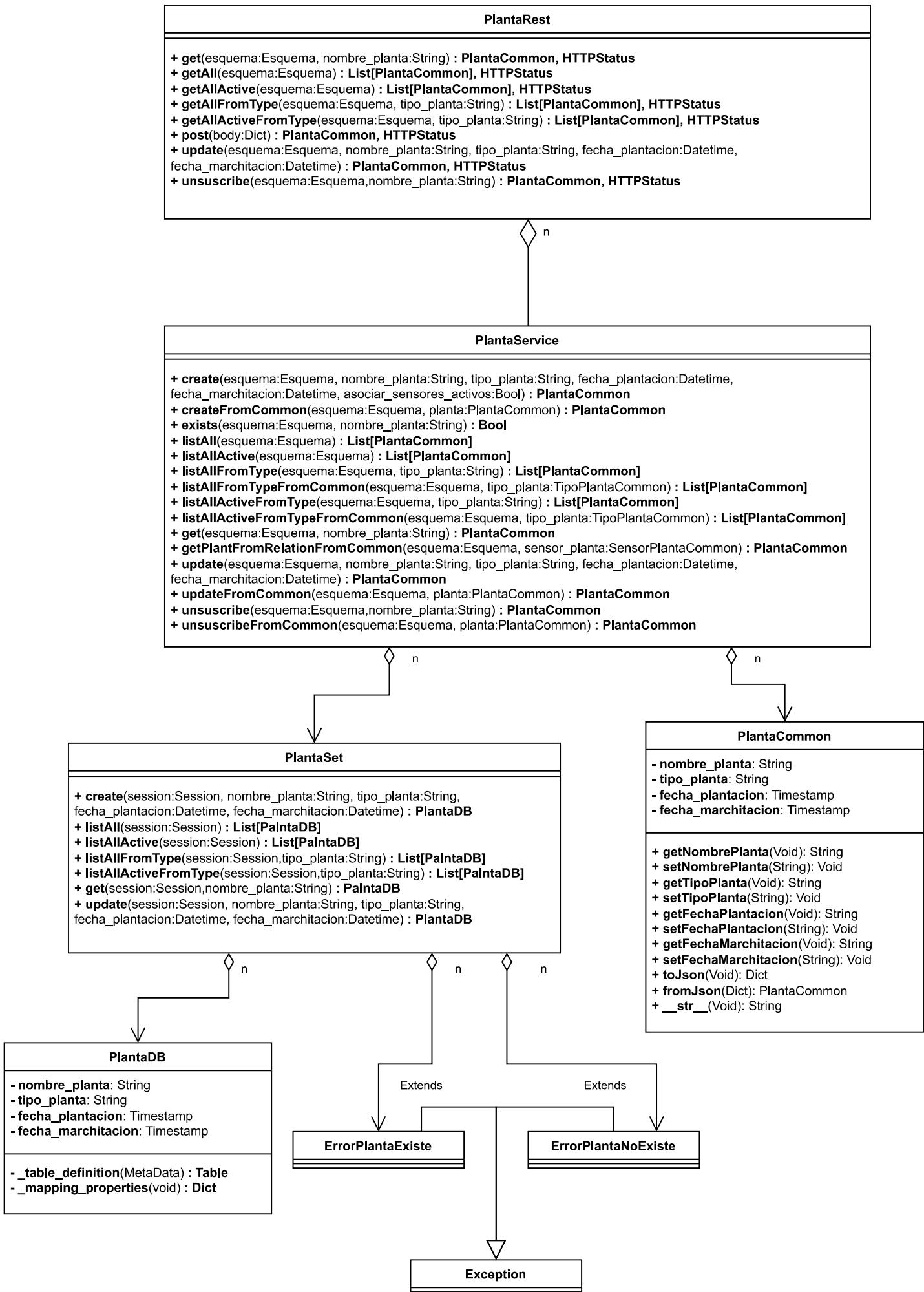
Para no violar la integridad referencial de los datos de la base de datos, no se permite borrar las instancias de los datos una vez grabadas. Para poder marcarlas como eliminadas se ha empleado un sistema de fechas, el cual determina cuando se creó la instancia y cuando se declaró como eliminada.



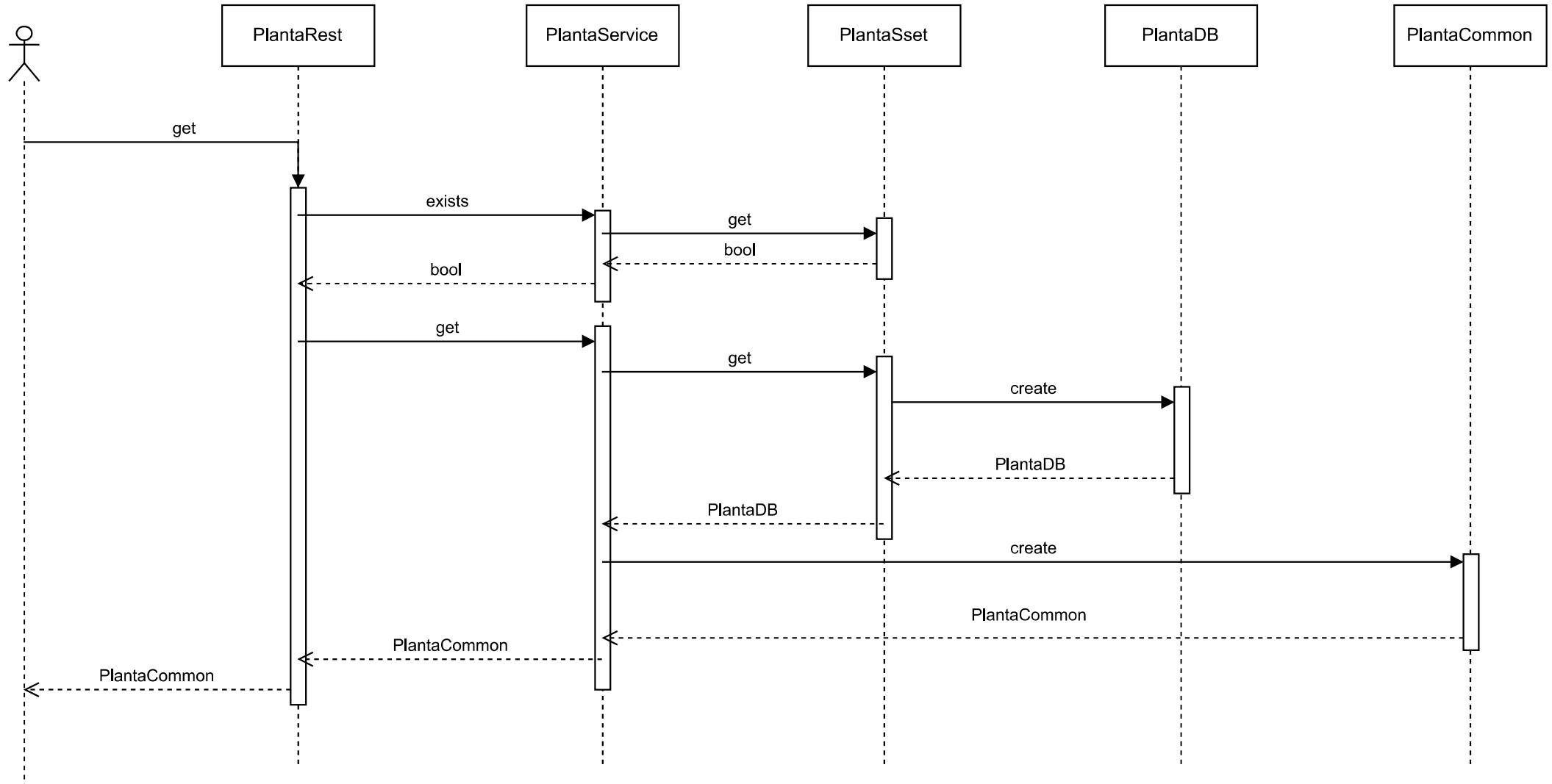
C.3. Diseño procedimental

Todas las siguientes entidades, las cuales son con las que trabaja Green In House para poder funcionar, siguen la misma estructura de clases, la cual se detalla a continuación. La única diferencia que hay entre estas entidades se debe a los atributos que contiene cada entidad, acorde a lo especificado en el diseño de datos, y a como se utilizan esos atributos para filtrar los objetos en los métodos get y list, utilizados para recuperar objetos de la base de datos. El método post y create se utiliza para crear los objetos en la base de datos. El método update se utiliza para actualizar los datos de los objetos en la base de dar y el método unsubscribe se utilizar para dar de baja dichos objetos, almacenando la fecha en que se dieron de baja.

- Sensores
- Registros Sensores
- Plantas
- Tipos Plantas
- Sensores Plantas
- Consejos Plantas
- Consejos Tipo Plantas



El usuario siempre va a interactuar con el sistema a través del API REST, ya sea desde la aplicación móvil haciendo consultas a la API REST, como desde un cliente API REST estilo postman o el propio Swager que recupere los objetos JSON para leerlos. Por lo tanto, la secuencia del flujo de consultas siempre va a ser muy similar al que se detalla en el ejemplo de a continuación. En este ejemplo se aprecia como el usuario utiliza un método de la API REST, el cual utiliza un método de la capa de servicios, el cual a su vez hace uso de un método del manejador de la base de datos para dicha clase. Esto devuelve una instancia de la base de datos en forma de objeto, del cual se extraen sus atributos para generar un objeto del tipo common, con el que poder trabajar en la aplicación de manera segura, sin miedo a que sea modificado por error y esto realice algún cambio en los datos de la instanciar al hacer el commit de alguna transacción en curso.



C.4. Diseño arquitectónico

Para poder desarrollar la arquitectura de Green In House de manera eficiente y con amplias posibilidades de crecimiento y reutilización, se ha recurrido a la utilización de patrones de diseño. 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 y modifiqué según mis necesidades, 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.

Patrón de diseño MVC (Modelo-Vista-Controlador)

El patrón de diseño Modelo-Vista-Controlador (MVC) se ha utilizado en Green In House para separar la lógica de la aplicación en tres componentes principales: el Modelo, la Vista y el Controlador.

Modelo

Gestiona los datos y las reglas de negocio de Green In House. Representa la información con la que trabaja y se encarga de acceder a los datos, almacenarlos y procesarlos. Estas clases están alojadas con la siguiente estructura:

- **/GreenInHouse/db/GreenInHouseBackend.sqlite3.db** : Archivo que almacena la base de datos SQLite3. Para manejar las consultas a este archivo desde Green In House se utiliza SQLAlchemy.
- **components/backend/backend/data** : En este directorio se almacenan las clases de datos del backend y sus clases manejadoras encargadas de recuperarlas de la base de datos y grabarlas de nuevo.
 - **./db/results** : clases de datos de las tablas almacenadas en la base de datos y de cada una de sus instancias.
 - **./db/resultsets** : clases de datos manejadoras de las tablas de la base de datos y de sus instancias.

- **./db/exc** : clases de excepción propias lanzadas durante problemas con la base de datos.
 - **./db/esquema.py** : clase que define el esquema principal de la base de datos.
 - **./electronic** : clases de datos utilizadas para generar objetos a través de los cuales interactuar con los sensores de la aplicación.
 - **./util** : clases de datos genéricos de los sensores electrónicos
- **components/common/common/data/util** : clases de datos genéricas de los objetos utilizados en Green In Hosue desligadas de su representación en la base de datos. Estas clases son utilizadas para poder compartir por la aplicación los objetos recuperados de la base de datos, sin riesgo a que sean modificados por error y eso repercuta en la base de datos.

Vista

Es la representación visual de los datos que maneja el modelo. Se encarga de mostrar la información al usuario de una forma entendible. Estas clases están alojadas con la siguiente estructura:

- **components/backend/backend/presentacion/rest** :clases que definen los *endpoints* a través de los que otros sistemas interactúan con Green In House mediante el uso de su servidor API REST.
- **components/backend/backend/openapi/spec.yaml** : clase definida según la especificación OpenAPI que define como tiene que funcionar internamente el servidor API REST
- **components/frontend/frontend/app/rpi** : clases que definen y gestionan la aplicación multiventana desplegada en la pantalla táctil de Green In House.
- **Aplicación Flutter** : aplicación multiplataforma que gestiona como interactuar con Green In House desde un dispositivo móvil.

Controlador

Es el intermediario entre la vista y el modelo. Gestiona la interacción del usuario con la vista, procesa las acciones del usuario y actualiza el modelo cuando es necesario. Estas clases están alojadas con la siguiente estructura:

- **components/backend/backend/service** : clases que definen los servicios a través de los cuales se utilizan de manera sencilla y cómoda los objetos de la aplicación sin necesidad de saber como está construida internamente ni como son las interrelaciones entre los objetos.
- **components/backend/bin** : scripts que gestionan el lanzamiento de los servicios de backend de Green In House, como el servidor API REST, la lectura de la base de datos, la lectura de los sensores y la lectura periódica de los sensores activos.
- **componentes/backend/backend/config** : clases de configuración de como lanzar los servicios de backend.
- **components/common/common/service** : clases que definen y gestionan servicios comunes como la gestión de la conectividad WiFi.
- **components/frontend/bin** : scripts que gestionan el lanzamiento de los servicios de frontend de Green In House, como la aplicación de interacción de la pantalla táctil de la maceta.
- **config** : archivos de configuración de como lanzar los servicios de backend.
- **init** : archivos de configuración de como lanzar durante el arranque de la raspberry los servicios de frontend de Green In House.
- **scripts** : scripts que gestionan la instalación de Green In House y sus entornos virtuales y dependencias, la configuración del sistema y el lanzamiento y la parada de los servicios de backend y frontend.

Patrón de diseño Adaptador:

Se implementa la clase SensorElectronico, la cual utiliza el patrón de diseño Adaptador, implementando una interfaz común de los métodos para realizar la lectura de los sensores. De esta interfaz heredan el resto de clases de sensores electrónicos del sistema (SensorElectronicoDHT11, SensorElectronicoMCP3008, etc), los cuales implementan los métodos de lectura suministrados, acorde a lo necesario para leer su correspondiente sensor electrónico del mundo real. A su vez, la clase SensorElectronicoMCP3008 se implementa como un adaptador de todos los módulos de sensores que necesiten hacer uso del módulo de conversión analógico digital MCP3008 para ser leídos por las Raspberry Pi, definiendo su método de comunicación y sus patillas a través de las cuales realizar las lecturas.

Patrón de diseño Factoría

En la clase FactoriaSensorElectronico se utiliza el patrón de diseño Factoría. Esta clase recibe el modelo de sensor que se quiere leer y los datos de las patillas que se van a utilizar para leerle. Con esos datos se encarga de generar un objeto SensorElectronico con la implementación correspondiente para poder leer dicho sensor.

Patrón de diseño Estrategia

En la clase SensorBackend se utiliza el patrón de diseño Estrategia para adaptarse al contexto suministrado, el cual son los datos almacenados de un sensor en la base de datos. Mediante el uso de dicho contexto, se llama a la factoría de sensores electrónicos para que devuelva un objeto SensorElectronico capaz de interactuar con el sensor real del modelo especificado a través de las patillas de comunicación especificadas.

Patrón de diseño Plantilla

En la clase SensorBackend se utiliza el patrón de diseño Plantilla al definir de manera general la lógica que permite leer dicho sensor electrónico, generar un registro del valor leído por el sensor y almacenar dicho registro en la base de datos. Para ello se utilizan los métodos suministrados por el adaptador SensorElectronico, los cuales son sobreescritos en su clase específica por su correspondiente implementación específica.

Partón de diseño Fachada

Se implementa una fachada mediante una capa de servicios que permite manejar de manera simple y cómoda las clases de Green In House y su interacción con la base de datos, sin necesidad de conocer como está programado y diseñado internamente. Se implementa una fachada mediante una capa manejadora de la API REST, la cual permite a aplicaciones externas hacer un uso sencillo del sistema y los datos almacenados en él, sin la necesidad de conocer como está programado y diseñado internamente.

Partón de diseño Singleton

Durante el lanzamiento del servidor API REST se define una única instancia de aplicación, la cual se almacena de manera global mediante el uso de Flask. Esta instancia de aplicación Flask se recupera cuando se hace uso de los métodos de la API REST para recuperar el directorio de la base de datos y pasárselo a la capa de servicios para que trabaje con él.

C.5. Diseño 3D

Para la realización del prototipo se ha realizado un diseño de modelado 3D de la maceta de Green In House preparada para albergar la planta, la raspberry y la pantalla tactil. Estos archivos han sido diseñados por Yeray Pescador Calleja y se encuentran accesibles en el repositorio de Green In House [15], en el apartado `/cad3D/maceta`. En las imágenes C.1, C.2 y C.3 puede apreciarse como es el diseño 3D de la maceta educativa de Green In House..



Figura C.1: captura de pantalla 1 del modelo 3D de Green In House.



Figura C.2: captura de pantalla 2 del modelo 3D de Green In House.

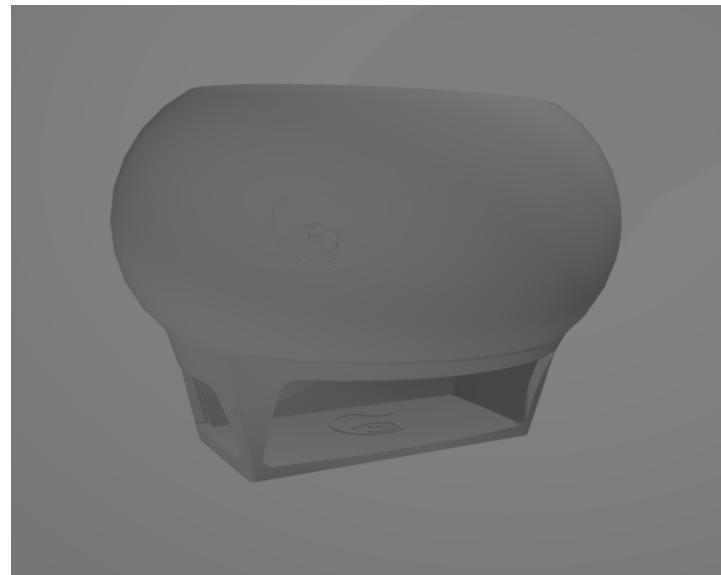


Figura C.3: captura de pantalla 3 del modelo 3D de Green In House.

Apéndice D

Documentación técnica de programación

Este apartado sirve para especificar y resumir la documentación técnica referente a la parte de desarrollo del código fuente de la aplicación, la instalación de Green In House realizada de cero y la ejecución de pruebas de sistema.

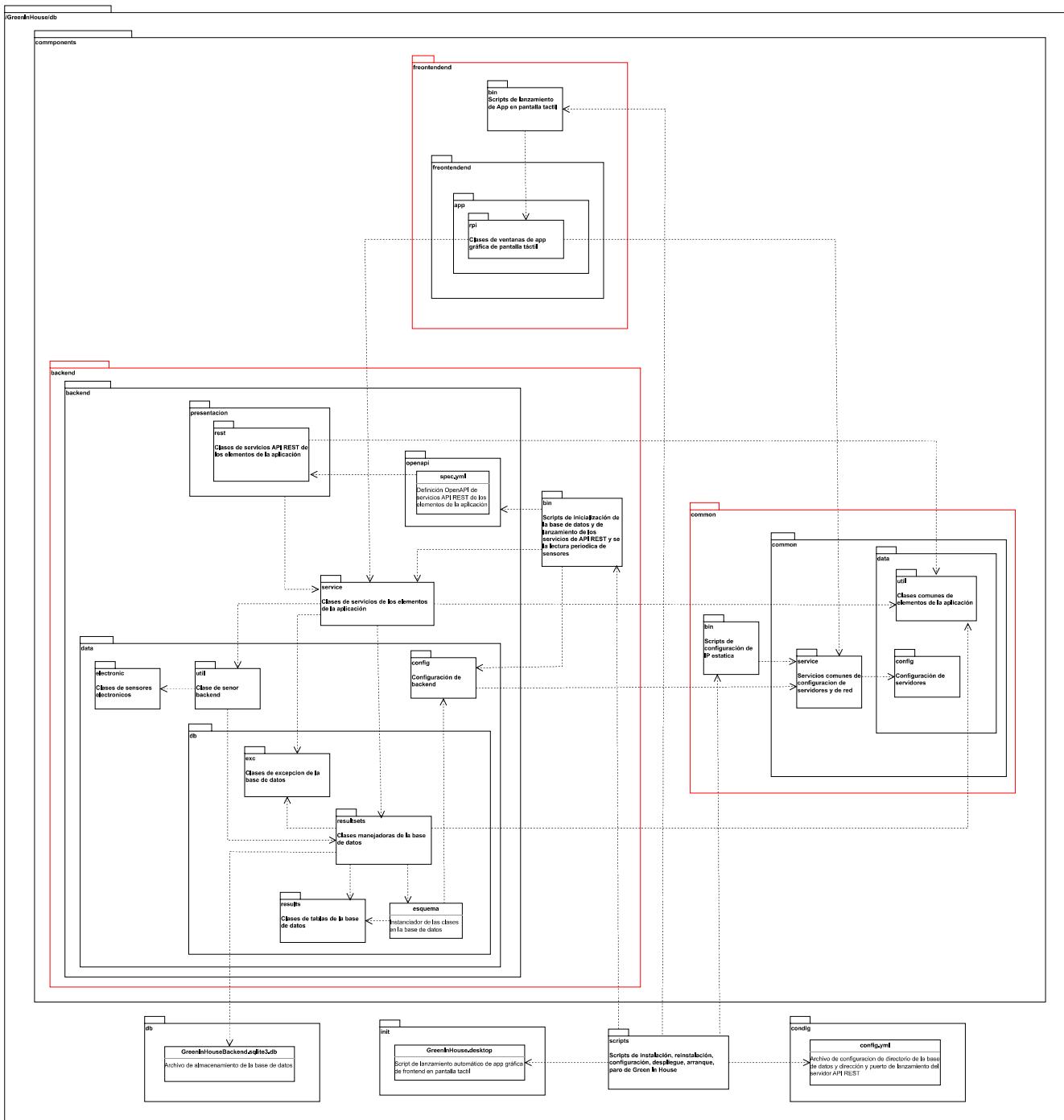
D.1. Introducción

En los siguientes apartados queda recogida la documentación respecto a las librerías que se utilizan en Green In House, las tecnologías que se utilizan y un manual de como habría que realizar de cero la instalación para poder replicar Green In Hosue. también está incluido un resumen de las pruebas de sistema que se han realizado.

D.2. Estructura de directorios

Este apartado ha sido descrito anteriormente en el apartado **C_Diseño** en la sección **Diseño arquitectónico** en la subsección **Patrón de diseño MVC (Modelo-Vista-Controlador)**. Se ha decidido redactar en dicho apartado, ya que la estructura de directorios ha sido diseñada haciendo uso de las recomendaciones de segmentación de funciones que propone dicha metodología.

La estructura de los módulos que conforman Green In House está recogida en el siguiente diagrama de paquetes.



D.3. Manual del programador

Para desarrollar el código de Green In House se han utilizando las librerías SQLAlchemy, Adafruit CircuitPython, OpenAPI, y Thinter las cuales detalla a continuación como han sido utilizadas.

SQLAlchemy: Librería de base de datos en Python

Green In House utiliza SQLAlchemy [13] para interactuar con una base de datos SQLite3. En esta base de datos están alojadas las tablas con las que trabaja Green In House, las cuales son:

- Sensores
- Registros Sensores
- Plantas
- Tipos Plantas
- Sensores Plantas
- Consejos Plantas
- Consejos Tipo Plantas

Cada una de estas tablas tiene implementada una clase intermediaria propia que se encarga de gestionar las transacciones con la base de datos y de construir el objeto Python con los datos de las filas de su tabla correspondiente, y de almacenar los datos en las tablas correspondientes de los objetos Python creados por la aplicación. Para poder manejar los datos de la base de datos de manera cómoda, Green In House tiene una serie de servicios implementados que se encarga de gestionar las conexiones con la base de datos, delegando las transacciones en las clases manejadoras anteriormente mencionadas.

Adafruit - CircuitPython: Librería de sensores y actuadores en Python

Green In House utiliza la librería CircuitPython de Adafruit [11] para interactuar con los sensores electrónicos del sistema y leer de manera cómoda

y sencillas sus valores. Green In House está preparada para comunicarse con los siguientes tipos de sensores:

MCP3008: Conversor analógico a digital (ADC)

El MCP3008 [2] es un conversor analógico-digital (ADC) de 10 bits y 8 canales. Esto significa que puede tomar hasta 8 señales analógicas diferentes y convertirlas en valores digitales. La resolución de 10 bits del MCP3008 indica que puede representar la señal analógica en valores digitales de 0 a 1023. Esto proporciona una precisión suficiente para muchas aplicaciones, incluyendo la lectura de la mayoría de los sensores analógicos. Uno de los aspectos más atractivos del MCP3008 es que se comunica con la Raspberry Pi (o cualquier otro microcontrolador) a través del protocolo de interfaz periférica serial (SPI). Esto hace que sea relativamente sencillo de conectar y programar, y más sencillo aun si se utiliza una librería que implementa ya la comunicación con dicho módulo como hace CircuitPython de Adafruit. [12]

DHT11: Sensor de humedad y temperatura ambiente

El DHT11 [1] es un sensor de temperatura y humedad que permite realizar lecturas de dichos parámetros mediante un microcontrolador, como Raspberry Pi. Este módulo es de bajo costo y de fácil uso, por lo es una opción muy popular para proyectos de monitoreo ambiental. Además la librería CircuitPython de Adafruit [12] cuenta con módulos ya programados para trabajar de manera muy sencilla con este sensor. El DHT11 es capaz de medir temperaturas entre 0 y 50 grados Celsius con una precisión de ± 2 grados, y humedad relativa entre 20 y 80 % con una precisión de ± 5 %. El módulo DHT11 utiliza un sensor capacitivo para la medición de la humedad y un termistor para la medición de la temperatura. Este sensor entrega una señal digital que puede ser leída directamente por la Raspberry Pi sin necesidad de convertirla mediante un ADC.

FC28: Sensor de humedad de tierra de la maceta

El módulo FC28 es un sensor de humedad del suelo muy utilizado en proyectos relacionados con la jardinería automatizada y el monitoreo del suelo. Este dispositivo tiene la capacidad de medir la cantidad de agua presente en el suelo, permitiendo tener un conocimiento preciso de las

condiciones de humedad de la tierra en la que está sembrada la planta. Este sensor entrega una señal analógica, por lo que se necesita convertirla mediante ADC a una señal digital para que pueda ser leída por la Raspberry Pi.

LDR: Sensor de luminosidad ambiente

Un LDR (*Light Dependant Resistor*), o fotoresistor es un dispositivo que varía su resistencia en función de la cantidad de luz que incide sobre él. Sin embargo, es importante tener en cuenta que un LDR no mide directamente los lúmenes, la cuál es una unidad de flujo luminoso. Un LDR mide la intensidad de luz de una manera no lineal y no opera en una frecuencia en particular. Para obtener una estimación de la luz en términos de lúmenes, se ha seguido el siguiente proceso:

1. **Configuración del circuito:** Conexión de un LDR y una resistencia fija en un circuito divisor de voltaje. Ambos extremos del LDR y la resistencia se conectan a una fuente de alimentación (en este caso a los 5V de la Raspberry Pi), y el otro extremo del LDR se conecta al otro extremo de la resistencia y a tierra (GND). De esta forma, el voltaje entre el LDR y la resistencia se puede medir y estará relacionado con la resistencia del LDR.
2. **Medición del voltaje:** Este voltaje se mide con un ADC en la Raspberry Pi. El modelo de Raspberry Pi utilizado o dispone de un ADC incorporado, por lo que se utiliza el ADC externo MCP3008, explicado anteriormente.
3. **Conversión del voltaje a resistencia:** Conociendo el voltaje de la fuente de alimentación (V), el voltaje medido (Vout) y la resistencia seleccionada (R), se calcula la resistencia del LDR (Rldr) utilizando la fórmula: $R_{ldr} = R * ((V / V_{out}) - 1)$.
4. **Calibración del sensor:** Para convertir la resistencia del LDR a lúmenes, hay que calibrar el sensor. Para ello, es necesario utilizar una fuente de luz con una salida conocida en lúmenes y medir la resistencia del LDR a diferentes niveles de luz. De esta forma, se puede crear una tabla de correspondencia entre la resistencia del LDR y los lúmenes.
5. **Conversión de resistencia a lúmenes:** Con la tabla de correspondencia, se puede convertir las medidas de resistencia del LDR a

lúmenes, aunque la fiabilidad de la medida depende mucho de lo bien que se haya realizado el proceso de calibración.

Es importante tener en cuenta que este método proporciona una medida aproximada y tiene sus limitaciones. Los LDR son sensibles a diferentes longitudes de onda de luz de diferentes maneras, por lo que diferentes tipos de luz (como la luz del sol, la luz fluorescente, etc.) darán diferentes lecturas.

BH1750: Sensor de luminosidad ambiente.

El módulo BH1750 es un sensor de luz ambiental de alta precisión muy utilizado en aplicaciones donde se requiere medir la intensidad de luz del entorno. Sirve para convertir la intensidad de luz que recibe en una señal digital que puede ser leída por un microcontrolador como Raspberry Pi. Este módulo tiene una alta precisión y resolución, pudiendo medir la intensidad de la luz en un rango desde 1 a 65535 lux. Cuenta interfaz de comunicación I2C, que permite una integración sencilla con la Raspberry Pi y la posibilidad de conectar varios sensores en serie al mismo bus I2C, lo cuál reduce la cantidad de pines utilizados en el microcontrolador. Este módulo puede tener 2 direcciones I2C distintas, dependiendo de si su patilla `Dir` se conecta a `Vcc` o a `GND`, por lo que esto permite tener 2 de estos módulos conectados en el mismo bus I2C.

OpenAPI: Librería de API REST en Python

Green In House utiliza el Microframework Connexion, el cuál mapea el API REST especificado en el archivo `spec.yml` (mirar el apartado *estructura de directorios* para conocer su ruta en el sistema) con OpenAPI [10] en Python, facilitando la creación de los *endpoints* para que otras aplicaciones puedan conectar con el servicio, el cuál es lanzado en el puerto 5000. Para poder hacer uso de estos *endpoints* hay que hacerlo utilizando de base una de las siguientes URLs:

- `http://192.168.1.240:5000/api/v1/`
- `http://greeninhouse:5000/api/v1/`

Para poder ver el Swager, la estructura de los *endpoints* y ejemplos de uso, hay que hacerlo utilizando de base una de las siguientes URLs:

- <http://192.168.1.240:5000/api/v1/ui/>
- <http://greeninhouse:5000/api/v1/ui/>

En la figura D.1 se muestra el aspecto del Swagger de Green In House

La captura de pantalla muestra la interfaz de Swagger para la API REST de Green In House. En la parte superior, se ve el logo de Swagger y la URL /api/v1/openapi.json. A la derecha, hay un botón verde que dice "Explore".

El título de la sección es "GIH backend service REST API 1.0 OAS3". Debajo de él, se indica que es la "REST API del servicio de backend" y que el autor es Oscar Valverde Escobar. Se menciona que es para el "Green In House. Grado en Ingeniería Informática, Universidad de Burgos, 2022-2023".

En la parte superior izquierda, hay un menú desplegable "Servers" con la opción "/api/v1" seleccionada.

La sección principal muestra los endpoints y sus operaciones:

- Servidor:** Operaciones sobre el propio servidor (query de estado del servidor). Incluye una operación "HEAD / Health test for the service".
- Registros Sensores:** Operaciones relacionadas con los sensores.
- Sensores:** Operaciones relacionadas con los sensores.
- Plantas:** Operaciones relacionadas con las plantas.
 - 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 D.1: captura de pantalla del Swagger de la API REST de Green In House desarrollada con OpenAPI

TKinter: Librería de interfaces gráficas en Python

Green In House utiliza la librería TKinter [4] para definir una aplicación gráfica multiventana que permite introducir en el sistema el nombre de la red WiFi del usuario y la contraseña de dicha red WiFi, así como reiniciar y apagar la maceta. En la figura D.2 se muestra como se ve realmente la interfaz gráfica de la aplicación.



Figura D.2: captura de pantalla de la app gráfica de Green In House desarrollada en TKinter

D.4. Compilación, instalación y ejecución del proyecto

Para facilitar la instalación de Green In House en nuevos sistemas, cuenta con una gran variedad de scripts ya programados en Bash para realizar de manera automatizada la instalación de librerías, la generación y configuración de entornos virtuales, la instalación de dependencias en sus correctas versiones, la creación de la base de datos, la configuración de dirección IP estática del WiFi, el despliegue del código del programa en los entornos virtuales y la configuración de autoarranque del servidor API REST, la lectura cíclica de sensores y el lanzamiento de la App gráfica en la pantalla táctil.

El directorio de despliegue en el que se almacenará toda la aplicación de Green In House tras su instalación por medio de los scripts facilitados se encuentra en la raíz del sistema y concretamente es **/GreenInHouse**. La estructura de directorios dentro de este directorio cambia ligeramente respecto a la del código fuente, pero se mantiene la misma estructura de paquetes para el funcionamiento de la aplicación. Se recomienda no modificar directamente los archivos del directorio **/GreenInHouse**, ya que al realizar un nuevo despliegue o instalación, las modificaciones en este directorio serán eliminadas. Si se quiere modificar el funcionamiento de la aplicación, habría que realizar dichas modificaciones en el código fuente descargado y posteriormente ejecutar el script de despliegue o instalación correspondiente.

Instalación del sistema operativo Raspbian

La instalación de Raspbian en la Raspberry Pi es un proceso muy sencillo. Solamente hay que seguir los siguientes pasos:

- Descargar de la página web oficial de Raspberry el software Raspberry Pi Manager <https://www.raspberrypi.com/software/>
- Instalar el software Raspberry Pi Manager en un ordenador.
- Ejecutar el software Raspberry Pi Manager. Se mostrará el menú de la figura D.3.

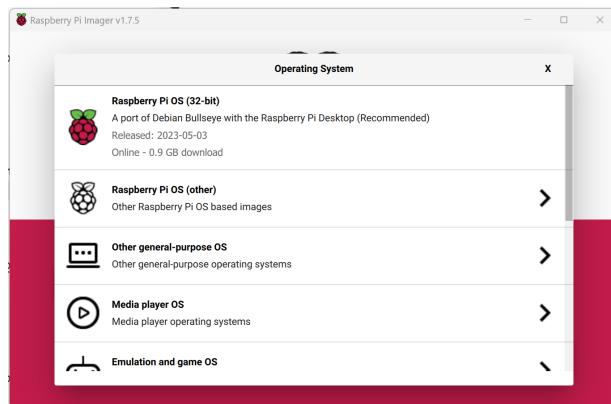


Figura D.3: Captura de pantalla del instalador de Raspbian

- Elegir la versión de sistema operativo que deseamos instalar. En mi caso he elegido Raspbian.
- Elegir el dispositivo de almacenamiento en el que instalarlo. En mi caso una tarjeta micro SD de 32 Gb.
- Una vez terminado el proceso de instalación, se introduce la tarjeta micro SD en la Raspberry, se le proporciona alimentación y la Raspberry comenzará a funcionar.
- La primera vez que arranque Raspbian nos pedirá configurar el país al que pertenecemos, el lenguaje que queremos utilizar y la zona horaria en la que vivimos.
- A continuación nos pide introducir un nombre de usuario y la co

En la figura D.4 se muestra como se visualiza el escritorio de Raspbian en mi Raspberry.

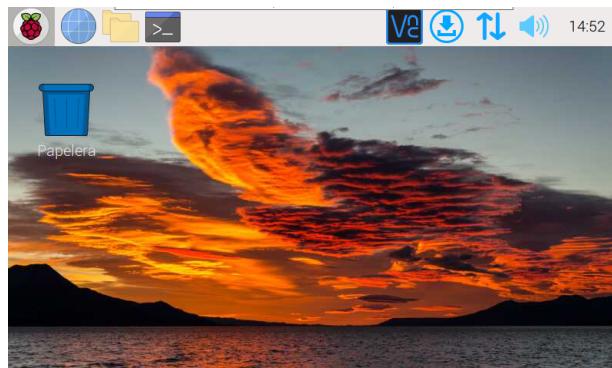


Figura D.4: captura de pantalla del escritorio de Raspbian

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. Como adaptador WiFi USB utilicé el modelo 8812BU el cuál dispone de conectividad a redes WiFi AC de alta velocidad. La instalación de este adaptador no está automatizada mediante un *script* ya que no es estrictamente necesario realizar en todos los modelos de Raspberry Pi, además necesita reiniciar el sistema varias veces a lo largo de la instalación, por lo que no es posible ejecutarlo en un único *script* de manera seguida. Debido a ello, su instalación manual queda reflejada a continuación. Es necesario contar con acceso a internet en la Raspberry para poder descargar el fichero, por lo que inicialmente será necesario conectarla por cable Ethernet:

- Los siguientes comandos sirven actualizar el sistema operativo a la última versión:
 - `sudo apt-get update`
 - `sudo apt-get install -f`
 - `sudo apt-get dist-upgrade`
- Ejecución del siguiente comando para realizar la actualización de las *headers* del sistema y de los módulos necesarios:
 - `sudo apt install -y linux-headers raspberrypi-kernel-headers build-essential bc dkms git`
- Reiniciar el sistema.

- Clonado del repositorio donde está almacenado su driver (para ello realicé un *fork* de la versión de los drivers que funcionaron en mi sistema, ya que probé muchos hasta encontrar unos que funcionasen correctamente):
 - `git clone https://github.com/ove1001/88x2bu-20210702`
- Entrar en el directorio del repositorio descargado y ejecutar el script de instalación:
 - `cd 88x2bu-20210702`
 - `sudo ./install-driver.sh`
- Reiniciar el sistema.
- Comprobar que el adaptador WiFi USB se reconoce correctamente:
 - `lsusb`
- Editar el fichero de sistema que almacena las interfaces Ethernet:
 - `sudo nano /etc/network/interfaces`
- Añadir en dicho fichero las siguientes líneas:
 - `allow-hotplug wlan0`
 - `iface wlan0 inet manual`
 - `wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf`
- Reiniciar el sistema.

Tras este proceso el adaptador WiFi estará instalado y configurado para poder conectarse a una red WiFi mediante una dirección IP estática. Debido a algún tipo de bug en la interfaz gráfica, es posible que no se muestren las redes WiFi de alrededor, pero el adaptador WiFi funcionará correctamente. Para introducir las credenciales de la red WiFi se utilizará la App desarrollada en la pantalla táctil.

Instalación de Green In House en Raspberry Pi

Una vez instalado el sistema operativo Raspbian, lo que hay que hacer a continuación es encender la Raspberry Pi y descargar el código del proyecto del repositorio oficial de Green In House [15]. Para ello se puede emplear el siguiente comando:

- Descargar del repositorio de Green In House utilizando Git.
`git clone https://github.com/ove1001/GreenInHouse_PlantPot`

Una vez descargado el código fuente, dentro de la carpeta `scripts` se encuentran varios `scripts` programados para realizar todas las tareas necesarias. Para realizar la instalación de Green In House únicamente es necesario realizar la siguiente acción:

- Entrar al directorio scripts del repositorio descargado
`cd GreenInHouse_PlantPot/scripts/`
- Ejecutar como superusuario el script `GIH-install.sh`
`sudo ./GIH-install.sh`

Este *script* tarda bastante en ejecutarse y requiere que la Raspberry tenga conexión a internet para descargar sus dependencias. Es posible que durante su ejecución pida la confirmación del usuario para instalar alguna librería, cuando eso suceda, escribir *yes* para conceder el permiso. Al terminar de ejecutar el *script*, reiniciar el sistema para que tome efecto la nueva configuración. Al encender de nuevo la Raspberry pi, se arrancarán de manera automática todos los servicios de Green In House.

Para poder hacer uso del servidor de la API REST será necesario tener conectado Green In House a una red Lan por Ethernet o por WiFi. Para introducir las credenciales de la WiFi se puede utilizar la aplicación gráfica desarrollada para la pantalla táctil.

Modificación de instalación de instalación de Green In House en Raspberry Pi

Entre los directorios del código fuente se encuentra la carpeta `config`, la cual contiene un archivo llamado `config.yml`. Este archivo contiene la información de la ruta donde se almacenará el archivo de la base de datos en el sistema y la dirección IP estática, máscara de red y puerta de enlace que se configurará en la Raspberry Pi durante la instalación. Se recomienda no cambiar los parámetros de este archivo, ya que inicialmente se definió con la idea de hacerlo modificable y que la aplicación se ejecutase acorde a lo establecido en dicho archivo, pero finalmente no ha sido posible por limitaciones técnicas. Estas limitaciones conciernen a la parte de la app móvil, ya que Android no es capaz de resolver las direcciones ip en base al

hostname. Debido a esto ha sido necesario establecer una IP estática en la Raspberry para el servidor API REST, y definir en el código Flutter de la app móvil, que el *endpoint* base al que tiene que dirigir todas las *API Calls* es 192.168.1.240, por lo que si se cambia la dirección IP de la Raspberry, la app móvil no será capaz de encontrar el servidor API REST y quedará inservible esta funcionalidad

Scripts Bash de Green In House para automatizar funciones de desarrollo

Green In House cuenta con varios scripts Bash desarrollados para facilitar las tareas de instalación, despliegue, ejecución y parada de la aplicación. El nombre de todos los *scripts* comienza por GIH-, haciendo referencia a que son tareas lanzadas para el funcionamiento de Green In House. Esto se ha determinado así para poder identificar fácilmente estas tareas en el programador de tareas de Raspbian y poder detener dichos procesos cuando sea necesario. Antes de lanzar una tarea, se realiza un intento de parada de la misma. Esto sirve para asegurar que no se lancen múltiples tareas para desarrollar el mismo trabajo, ya que eso ocasionaría inestabilidad en el sistema y pérdida de recursos.

- Instalación de la aplicación (tanto la parte de *backend* como de *front-end*), generación de entornos virtuales de Python con la instalación de sus correspondientes dependencias, establecimiento de IP estática, configuración de los archivos de sistema pertinentes y configuración del arranque automático de los servicios de Green In House durante el arranque del sistema operativo. Estos scripts están escritos en Bash y es necesario ejecutarlos como superusuario, ya que necesitan generar y modificar archivos dentro de los directorios del sistema. Estos *scripts* tardan bastante en ejecutarse y es necesario contar con conexión a internet. Es necesario reiniciar la Raspberry tras la instalación para que tengan efectos los cambios realizados en el sistema operativo. Todos ellos comprobarán durante la instalación del sistema si existe el archivo de la base de datos /GreenInHouse/db/GreenInHouseBackend.sqlite3.db y si no está vacío. En caso contrario, se crea un nuevo archivo de base de datos y se ejecuta el *script* Python GIH-backend-create-initial almacenado en el directorio backend/bin, instanciando los tipos de plantas, la planta y los sensores establecidos por defecto en dicho script.
 - GIH-install.sh : instalar todo el entorno de Green In House.

- `GIH-install_frontend.sh` : instalar el entorno de frontend de Green In House.
 - `GIH-install_backend.sh` : instalar el entorno de backend de Green In House.
 - `GIH-reinstall.sh` : renistalar todo el entorno de Green In House.
 - `GIH-reinstall_clean.sh` : reinstalar todo el entorno de Green In House y borrar el archivo actual de la base de datos. (**CUIDADO: el borrado es definitivo, no es posible recuperarlo después**).
 - `GIH-reinstall_frontend.sh` : reinstalar el entorno de frontend de Green In House.
 - `GIH-reinstall_backend.sh` : reinstalar el entorno de backend de Green In House.
 - `GIH-reinstall_clean_backend.sh` : renistalar el entorno de backend de Green In House y borrar el archivo actual de la base de datos. (**CUIDADO: el borrado es definitivo, no es posible recuperarlo después**).
- Despliegue de cambios realizados en el código de la aplicación.
- `GIH-deploy_all.sh` : desplegar todo el código fuente de Green In House a los entornos virtuales y actualizar sus dependencias.
 - `GIH-deploy_clean_all.sh` : desplegar todo el código fuente de Green In House a los entornos virtuales, actualizar sus dependencias y borrar el archivo actual de la base de datos. (**CUIDADO: el borrado es definitivo, no es posible recuperarlo después**).
 - `GIH-deploy_frontend.sh` : desplegar todo el código fuente del *frontend* al entorno virtual de *frontend* y actualizar sus dependencias.
 - `GIH-deploy_backend.sh` : desplegar todo el código fuente del *backend* al entorno virtual de *backend* y actualizar sus dependencias.
 - `GIH-deploy_clean_backend.sh` : desplegar todo el código fuente del *backend* al entorno virtual de *backend*, actualizar sus dependencias y borrar el archivo actual de la base de datos. (**CUIDADO: el borrado es definitivo, no es posible recuperarlo después**).

- **GIH-deploy_spec_yml_backend.sh** : desplegar la especificación de la API REST al entorno virtual de *backend* y actualizar sus dependencias.
- Configurar la dirección IP de la interfaz WiFi acorde a la que establece el archivo de configuración **config.yml** del directorio **/GreenInHouse/cfg/**
 - **GIH-configure_static_ip.sh** : modificar la dirección IP estática de la interfaz WiFi. Ejecuta internamente el *script* Python **GIH-common-configure-static-ip** almacenado en el directorio **common/bin**.
- Lectura de los sensores activos en el sistema y su correspondiente generación de líneas en la base de datos
 - **GIH-read_sensors.sh** : leer los sensores activos del sistema y grabar sus registros en la base de datos. Ejecuta internamente el *script* Python **GIH-backend-read-sensors** almacenado en el directorio **backend/bin**.
- Lectura de la base de datos para mostrarla por pantalla (principalmente utilizado para pruebas)
 - **GIH-read_db.sh** : mostrar por pantalla o por terminar el contenido de la base de datos de manera legible para las personas (frases redactadas en base a sus atributos). Ejecuta internamente el *script* Python **GIH-backend-read-db** almacenado en el directorio **backend/bin**.
- Parada y arranque de los diferentes procesos en ejecución que controlan el funcionamiento de Green In House.
- Arranque y parada de tareas para la ejecución de servicios de Green In House:
 - **GIH-run_api_rest.sh** : arranca el servidor API REST en la dirección y puerto configurados (por defecto en **192.168.1.240:5000:/api/v1/** o **greeninhouse:5000:/api/v1/**). Ejecuta internamente el *script* Python **GIH-backend-api-rest** almacenado en el directorio **backend/bin**.
 - **GIH-stop_api_rest.sh** : detiene el servidor API REST.

- `GIH-run_read_sensors_periodically.sh` : arranca el servicio cíclico de lectura de sensores activos. Por defecto este servicio lee los sensores y almacena sus registros en la base de datos cada 10 minutos. Ejecuta internamente el *script* Python `GIH-backend-read-sensors` almacenado en el directorio `backend/bin`.
- `GIH-stop_read_sensors_periodically.sh` : detiene el servicio cíclico de lectura de sensores activos.
- `GIH-run_app_rpi.sh` : arranca la aplicación gráfica de la pantalla tactil. Ejecuta internamente el *script* Python `GIH-frontend-app-rpi` almacenado en el directorio `frontend/bin`.
- `GIH-stop_app_rpi.sh` : detiene la aplicación gráfica de la pantalla tactil.
- Arranque de servicios de Green In House:
 - `GIH-start_all.sh` : arranca los servicios de frontend y backend.
 - `GIH-stop_all.sh` : detiene TODOS los servicios de Green In House (frontend, backend, instalación en curso y despliegue en curso).
 - `GIH-stop_except.sh` : detiene todos los servicios de Green In House (frontend, backend, instalación en curso y despliegue en curso) excepto los servicios que se pasen como argumentos de entrada al *script*. Admite hasta tres argumentos de entrada. Un argumento de entrada puede ser una tarea en concreto (`GIH-read_db.sh`) o un conjunto de tareas (`GIH-deploy` : todos los *scripts* de despliegue).
 - `GIH-stop_process.sh` : detiene el servicio especificado. Admite hasta tres argumentos de entrada. Un argumento de entrada solo puede ser una tarea en concreto (`GIH-read_db.sh`) o un conjunto de tareas (`GIH-deploy` : todos los *scripts* de despliegue).
 - `GIH-start_frontend.sh` : arranca los servicios de frontend.
 - `GIH-stop_frontend.sh` : detiene los servicios de frontend.
 - `GIH-start_backend.sh` : arranca los servicios de backend.
 - `GIH-stop_backend.sh` : detiene los servicios de backend.
 - `GIH-stop_backend.sh` : detiene los servicios de backend.

Estos *scripts* están alojados en el directorio `scripts` del código fuente de Green In House. Tras la instalación de Green In House quedan alojados en el directorio `/GreenInHouse/script`

Configuración del sistema operativo Raspbian

La configuración de estos archivos se realiza automáticamente durante la ejecución de los *scripts* de instalación y reinstalación. Para poder desplegar la aplicación de Green In House de manera automática al encender las Raspberry y poder configurar los parámetros del adaptador WiFi, ha sido necesario modificar los siguientes ficheros de sistema:

- **Cron y Crontab :** Estos archivos se encargan de la ejecución periódica de tareas por parte del sistema operativo. Se ha utilizado esta funcionalidad para desplegar el backend de la aplicación durante el arranque del sistema operativo, de tal manera que aunque no se inicie el entorno gráfico, los servicios de backend de Green In House se lanzarán igualmente.
- **/etc/xdg/autostart :** En este directorio se almacena el archivo de configuración `GreenInHouse.desktop`. Este archivo gestiona el arranque automático de la aplicación gráfica de la pantalla táctil de Green In House durante el arranque del entorno gráfico del sistema operativo. Este archivo de configuración es copiado a este directorio durante la instalación de Green In House desde el directorio `init` del código fuente. Se recomienda no modificar este archivo para evitar corromper el lanzamiento automático de la aplicación gráfica de Green In House.
- **/etc/network/interfaces :** en este archivo se almacena la configuración de las interfaces de red habilitadas en la Raspberry Pi. En mi caso, la interfaz WiFi está etiquetada como `Wlan0`.
- **/etc/dhcpcd.conf :** en este archivo se establece el funcionamiento del DHCP o la especificación de la IP estática definida para cada interfaz de red. En el caso de Green In House, se ha definido la IP estática `192.168.1.240/24` para la interfaz `Wlan0`.
- **/etc/wpa_supplicant/wpa_supplicant.conf :** en este archivo se almacenan las credenciales de las redes WiFi conocidas (los datos del SSID y la clave de la red WiFi que introduzca el usuario) con un formato determinado para que pueda ser utilizado por el sistema operativo.

La modificación de cualquiera de estos archivos requiere que se reinicie el sistema para que tengan efecto los cambios.

Utilización de Venv en Green In House

Green In House cuenta con 3 venvs [5] diferentes, de los cuales 2 están destinados al backend y 1 al frontend.

- **venv de backend para lectura de sensores:** Este entorno virtual está destinado principalmente a gestionar el uso de la librería Adafruit para leer los sensores electrónicos y la librería SQLAlchemy para almacenar los datos de los registros en la base de datos.
- **venv de backend para API REST:** Este entorno virtual está destinado principalmente a gestionar el uso de la librería OpenAPI, desplegando un servidor API REST, el cual gestiona las peticiones de datos externas e interactúa con la base de datos almacenando y recogiendo información mediante el uso de la librería SQLAlchemy.
- **venv de frontend para App gráfica de pantalla táctil:** Este entorno virtual está destinado principalmente a gestionar el uso de la librería Tkinter, la cual se utiliza para desplegar una aplicación gráfica que permite al usuario interactuar con la Raspberry Pi e introducir los datos de la red WiFi de su hogar, para conectar el servidor Green In House a su red. También permite apagar y reiniciar el sistema.

Configuración de SSH para desarrollo remoto en Raspberry Pi utilizando VSCode

A la hora de realizar el proyecto, la manera más cómoda de trabajar que he encontrado, ha sido desarrollar en remoto desde mi ordenador portátil utilizando una conexión SSH [6] y el IDE de programación VSCode [9]. Esto me ha permitido desplegar el directorio de trabajo de mi Raspberry Pi en el VSCode de mi ordenador, como si fuera parte de mi sistema local. Este apartado es solo para facilitar el desarrollo, no es necesario realizarle para ejecutar el programa de Green In House en la Raspberry Pi. La configuración de este apartado requiere hacerse de manera interactiva, ya que hay que generar las claves privadas y públicas dentro de la Raspberry Pi, y esto requiere suministrar una clave de cifrado propia del desarrollador. Este sistema de conexión remota segura no puede automatizarse debido a esto, ya que si se queda *hardcodeada* la clave de comunicación dentro del propio código de la aplicación, puede suponer un gran riesgo de seguridad. Por ello, este apartado es necesario realizarle a mano, aunque durante la instalación

Green In House habilita todos los puertos de comunicación para asegurar poder comunicarse correctamente con todos los sensores (SSH, I2C, SPI, SERIAL, GPIO). Para realizar la configuración de SSH y VSCode se ha seguido una guía de internet [8], pero se deja reflejado en este apartado los pasos a seguir.

Configuración de SSH en Raspberry Pi

A continuación se detalla como se configuran las claves de acceso por SSH en la Raspberry Pi

- Crear el directorio \$HOME/.ssh ejecutando el siguiente comando:
`mkdir -p $HOME/.ssh`
- **Generar par de llaves SSH**
`ssh-keygen -t ed25519 -f $HOME/.ssh/GIH`
- Introducir nuestra frase clave utilizada para generar el par de claves privada-publica
- Volver a introducir la misma frase clave.
- **Añadir la llave privada a SSH Agent**
`ssh-add $HOME/.ssh/GIH`
- Introducir nuestra frase clave utilizada para generar el par de claves privada-publica
- **Copiar la llave pública a RPi**
 - En el siguiente comando sustituir green-in-house por el nombre de usuario que se haya configurado en Raspbian para el usuario principal con el que arranca el sistema operativo por defecto.
`ssh-copy-id -i $HOME/.ssh/GIH.pub green-in-house@GreenInHouse`
 - Introducir la clave del usuario especificado en el comando anterior
 - Reiniciar el sistema

Configuración de VSCode en ordenador

A continuación se detalla como se configura en VSCode el acceso por SSH a la Raspberry Pi.

- Abrir VSCode.
- Abrir una terminal de VSCode.
- **Instalar la extensión Remote - SSH**
- `code -install-extension ms-vscode-remote.remote-ssh`
- Cerrar y abrir VScode.
- **Configurar la conexión SSH**
- Pinchar en el icono de la tele con un circulo (señalado en la imagen D.5) para acceder a las conexiones SSH y pulsar sobre el + (señalado en la imagen D.5)).

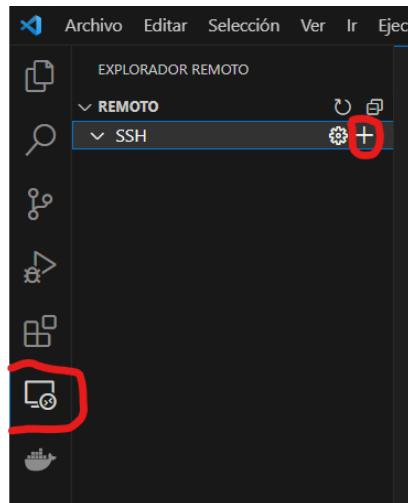


Figura D.5: captura de pantalla 1 de VSCode configurando acceso por SSH.

- En el cuadro que aparecerá escribir el usuario de la raspberry @ seguido del hostname o la dirección IP como en la imagen D.6. En el caso de Green In House es:
green-in-house@GreenInHouse

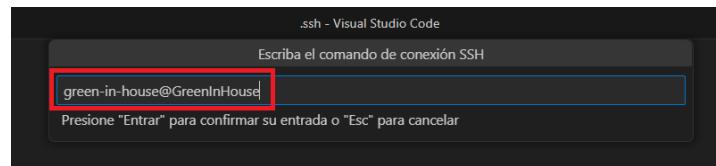


Figura D.6: captura de pantalla 2 de VSCode configurando acceso por SSH.

- Si experimentamos problemas intentando acceder por SSH utilizando usuario de la raspberry @ seguido del hostname, probar a acceder mediante usuario de la raspberry @ dirección IP. En el caso de Green In House es:
green-in-house@192.168.1.240
- En el cuadro que aparecerá seleccionar el directorio en el que se almacenará la configuración SSH como en la imagen D.7. Utilizar el directorio de nuestro usuario actual del ordenador. En mi caso es:
- C:/Users/UX581/.ssh/config

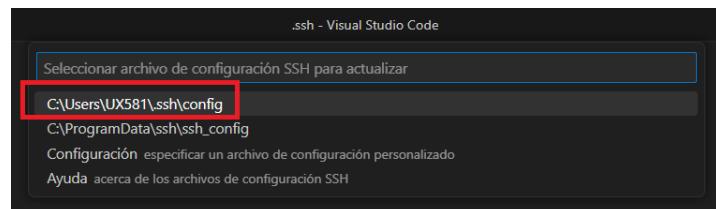


Figura D.7: captura de pantalla 3 de VSCode configurando acceso por SSH.

- Abajo a la derecha aparecerá un mensaje de que se ha agregado correctamente el host SSH, tal como se muestra en la imagen D.8.

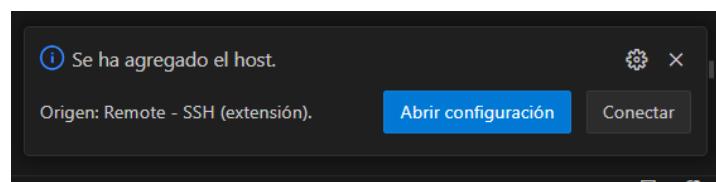


Figura D.8: captura de pantalla 4 de VSCode configurando acceso por SSH.

- Si pulsamos en abrir configuración se mostrará un archivo con las siguientes líneas:

```
Host GreenInHouse
  HostName GreenInHouse
  User green-in-house
```
- Pinchar de nuevo en el icono de la tele con un circulo (señalado en la imagen D.9) para acceder a las conexiones SSH y pulsar sobre la flecha ->(señalada en la imagen D.9).

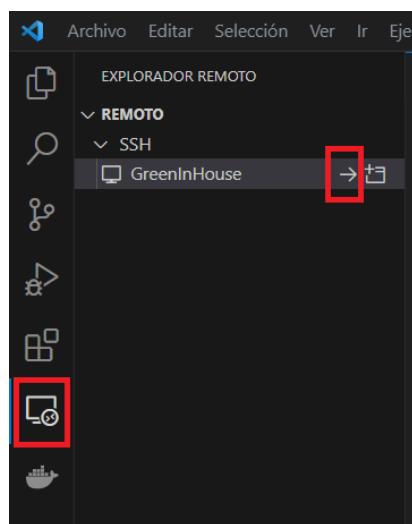


Figura D.9: captura de pantalla 5 de VSCode configurando acceso por SSH.

- En el cuadro que aparecerá introducir la contraseña del usuario de la raspberry en el que estamos intentando loguearnos, como se muestra en la imagen D.10:

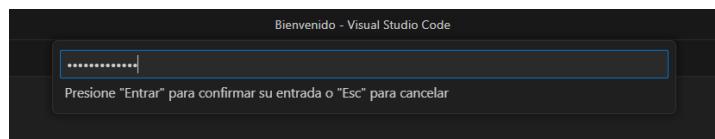


Figura D.10: captura de pantalla 6 de VSCode configurando acceso por SSH.

- Abajo a la izquierda aparecerá un cuadrado azul que indica que estamos conectados por SSH al equipo remoto como en la imagen D.11:

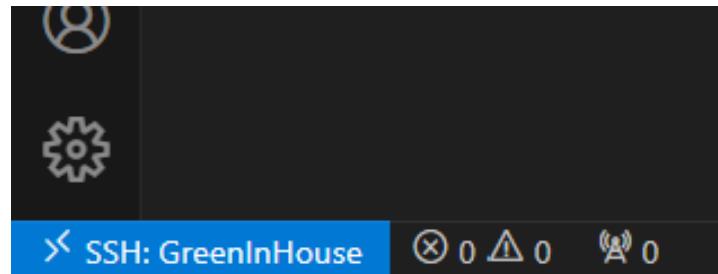


Figura D.11: captura de pantalla 7 de VSCode configurando acceso por SSH.

- Pulsando en archivo y en abrir carpeta nos permitirá abrir un directorio remoto y trabajar con él como de costumbre, como muestra la imagen D.12:

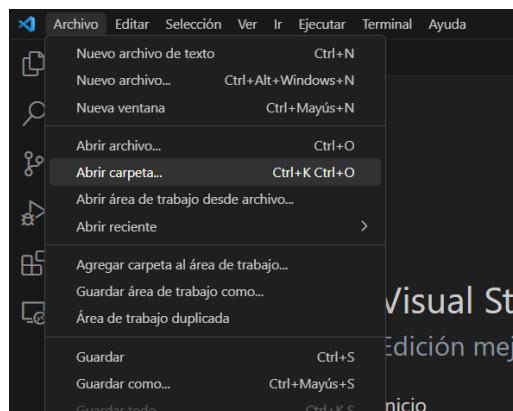


Figura D.12: captura de pantalla 8 de VSCode configurando acceso por SSH.

- En el cuadro que aparecerá introducir la ruta del directorio que queremos abrir, como se muestra en la imagen D.13. En mi caso es:
`/home/green-in-house/TFG/GreenInHouse_PlantPot`

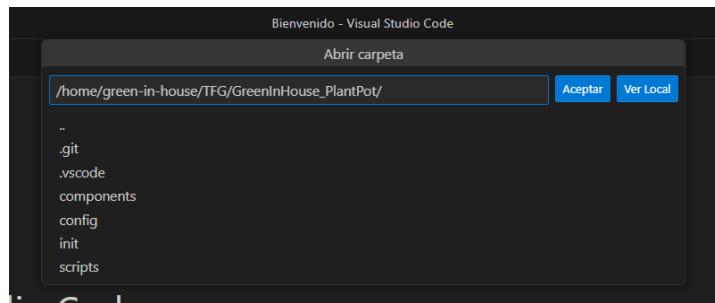


Figura D.13: captura de pantalla 9 de VSCode configurando acceso por SSH.

- Nos volverá a pedir introducir la contraseña del usuario de la Raspberry utilizado. Tras hacerlo se desplegará el directorio y sus archivos como se muestra en la imagen D.14.

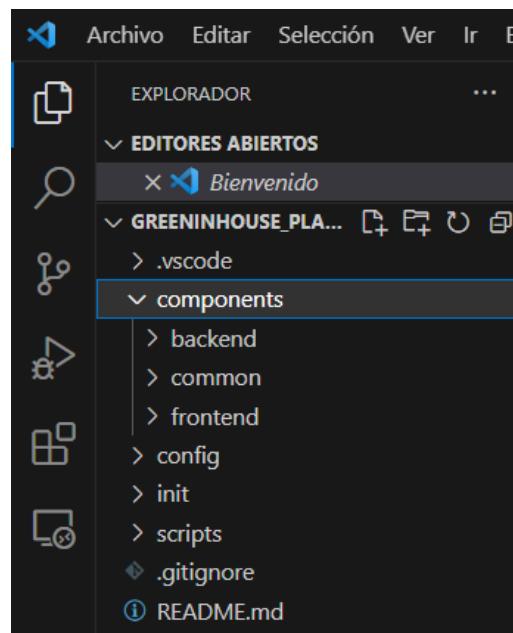


Figura D.14: captura de pantalla 10 de VSCode configurando acceso por SSH.

- Una vez que accedamos a un directorio, al pulsar en el ícono de la tele con el círculo para conectarnos a un servidor remoto, nos mostrará la dirección de los últimos directorios utilizados de dicho servidor remoto,

como se muestra en la imagen D.15:

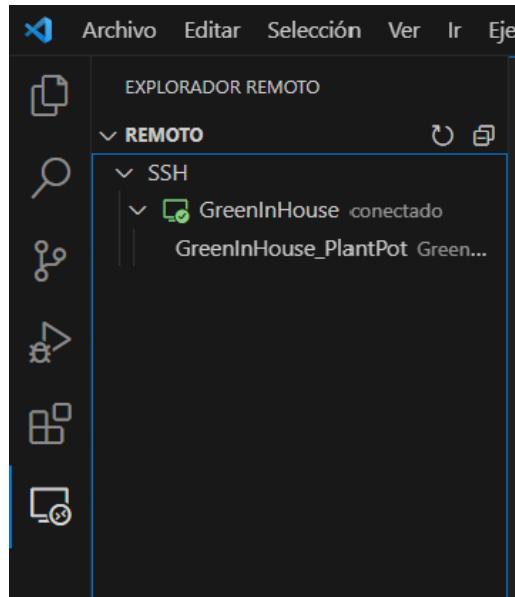


Figura D.15: captura de pantalla 11 de VSCode configurando acceso por SSH.

Desarrollo de aplicación móvil en Flutter

Para desarrollar la aplicación móvil en Flutter [7] se ha recurrido a la utilización de Flutter Flow [3], el cuál permite realizar el desarrollo de aplicaciones móviles basándose en el diseño de la interfaz. A medida que se va construyendo la interfaz gráfica, Flutter Flow va generando el código de la aplicación Flutter correspondiente. Flutter Flow es una plataforma privada y cerrada, por lo que no permite compartir públicamente mi proyecto de Flutter Flow para que cualquier pueda continuar desarrollándole. Por suerte la versión especial que tengo concedida gracias a ser estudiante universitario, me concede acceso a sincronizar la App desarrollada en Flutter Flow con un repositorio GitHub. Gracias a eso todos los avances en la aplicación móvil son accesibles públicamente a través de su repositorio [14]. Esto tiene el impedimento de que únicamente de deja realizar *push* de Flutter Flow al repositorio. En ningún caso Flutter Flow permite hacer *Pulls* de los cambios hechos desde otro IDE, por lo que obliga a desarrollar todo el proyecto dentro de su plataforma. Para poder continuar desarrollando este proyecto

Flutter en un IDE como por ejemplo Android Studio, hay que seguir los siguientes pasos:

- hacer un fork del repositorio de GitHub de Green In House Mobile App, para poder tener una versión propia del código de la aplicación.
- Realizar un *gitclone* desde Android estudio del repositorio creado. Esto se realiza desde la pestaña superior de git, como se muestra en la imagen D.16.

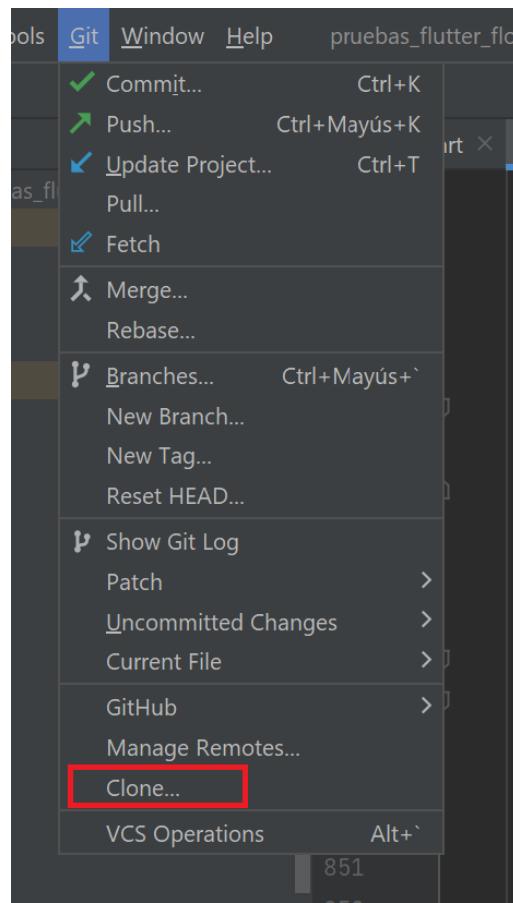


Figura D.16: captura de pantalla 1 de Android Studio.

- Especificar el repositorio que se quiere clonar, la ruta local donde queremos clonarlo y generar el proyecto y damos al botón *clone* como se muestra en la imagen D.17 .

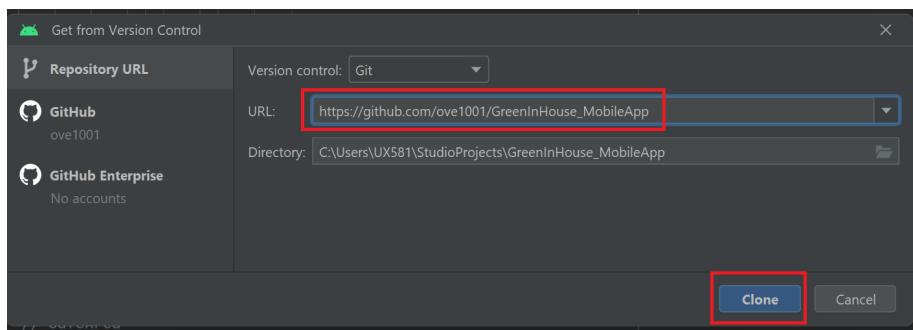


Figura D.17: captura de pantalla 2 de Android Studio.

- Una vez hecho el paso anterior, aparecerá en el navegador de Android Studio la aplicación Flutter como se muestra en la imagen D.18 .

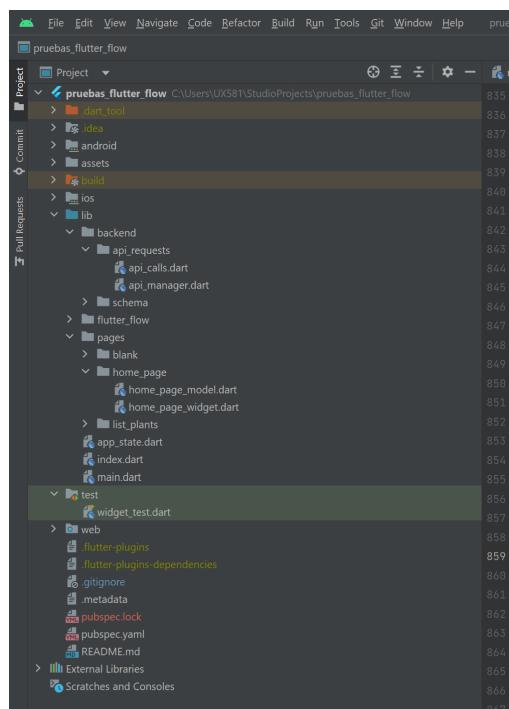


Figura D.18: captura de pantalla 3 de Android Studio.

- Para que la aplicación Flutter se pueda ejecutar, es necesario instalar sus dependencias. Para ello hay que pulsar sobre el botón Tools, y picchar en Flutter Pub Get como se muestra en la imgn imagen D.19. Para limpiar las dependencias pulsar sobre Flutter Clean.

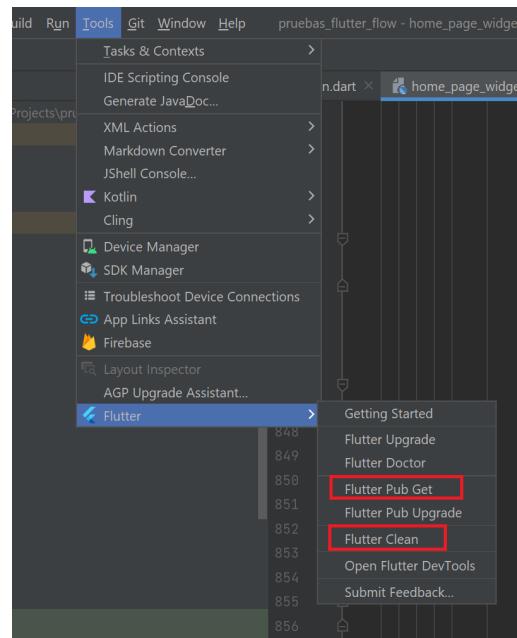


Figura D.19: captura de pantalla 4 de Android Studio.

- Una vez instalas las dependencias, arriba a la derecha aparecerá un botón de play y un cuadro donde seleccionar el dispositivo a desplegar la App como se aprecia en la imagen imagen D.20. Si se quiere, también se puede generar un dispositivo virtual, pero esa explicación queda fuera del alcance de este manual.

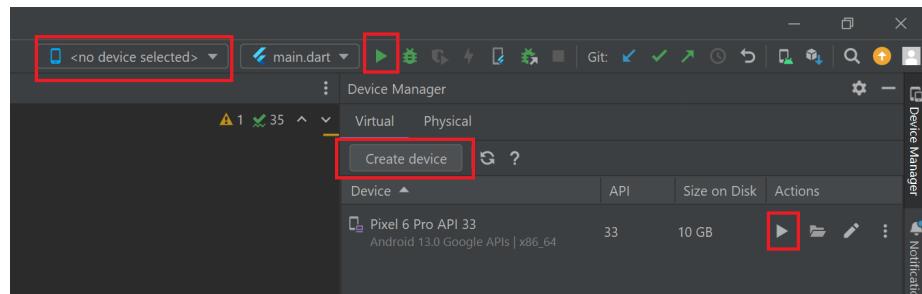


Figura D.20: captura de pantalla 5 de Android Studio.

D.5. Pruebas del sistema

No ha sido posible crear un sistema automatizado de pruebas para verificar la integridad del código, por lo que todas las pruebas de funcionamiento han sido realizadas a mano. Principalmente las pruebas que han sido realizadas son pruebas de integración, las cuales permiten validar el funcionamiento correcto de los módulos por separado, al comprobar que su funcionamiento en conjunto es correcto. Lo más óptimo habría sido diseñar un plan de pruebas automatizado de pruebas unitarias y otro de pruebas de integración, pero por motivos de falta de tiempo no ha sido posible realizar dichos planes de pruebas. Esto se ha incluido como una de las propuestas de líneas futuras.

Todas las pruebas se han realizado tanto utilizando los servicios del propio backend por terminal, como haciendo uso de los *endpoints* de la API REST, ya que esto es lo realmente interesante, debido a que será la forma habitual de operar de los usuarios. También se han realizado prueba de correcto funcionamiento de los scripts de Green In House. Las pruebas que se han realizado son de creación, obtención, modificación y borrado de todas y cada una de las entidades que se utilizan en el sistema. También se ha probado que todos los filtros de obtención de datos funcionan correctamente.

Apéndice E

Documentación de usuario

En este apartado se expone como un usuario que ha adquirido una maceta educativa Green In House tiene que realizar el arranque inicial del sistema para ponerlo en funcionamiento y cómo utilizarle una vez está funcionando.

E.1. Introducción

Para poder facilitar al usuario su iniciación en el uso de Green In House se ha redactado la siguiente guía. En ella se exponen los pasos necesarios a realizar para poner Green In House a funcionar y como hacer uso del sistema posteriormente.

E.2. Requisitos de usuarios

Es necesario que el usuario cumpla con los siguientes requisitos para poder hacer uso de Green In House.

- Para poder poner el sistema a funcionar, el usuario tendrá que disponer en casa de una red WiFi a la que conectar Green In House. Si no dispone de dicha red, podrá realizar lecturas de los sensores, pero no podrá consultar los registros de los datos desde el móvil.
- Para poder utilizar la aplicación móvil para visualizar los registros de Green In House, el usuario tendrá que disponer de un dispositivo Android en el que instalar la aplicación móvil.

- El usuario tendrá que conectar regularmente Green In House a la corriente eléctrica mediante su cargador para recargar la batería de la maceta.
- Este primer prototipo no es completamente impermeable, por lo que el usuario tendrá que tener un mínimo cuidado durante la operación de regado de la planta para evitar inundarla, ya que podría dañar la electrónica del sistema.

E.3. Instalación

A continuación se detallan los pasos necesarios a seguir para poder realizar la instalación de la aplicación de Green In House en un dispositivo móvil Android.

1. Descargar el instalador .apk de la App móvil de Green In House desde el repositorio oficial https://github.com/ove1001/GreenInHouse_MobileApp [14], ya que actualmente no se encuentra aún disponible en la tienda de aplicaciones Android.
2. Ejecutar el instalador .apk descargado.
3. Como la App no está siendo instalada desde la tienda oficial de aplicaciones, es necesario dar permiso de instalación desde fuentes desconocidas a la aplicación desde la que se esté realizando la instalación como se puede ver en las imágenes E.1 y E.2



Figura E.1: Conceder permisos de instalación en Android



Figura E.2: Conceder permisos de instalación en Android

4. Una vez terminado el proceso de instalación, se podrá abrir la App y se conectará automáticamente al servidor de Green In House, siempre que ambos dispositivos (móvil y maceta) estén conectados a la misma red WiFi como se muestra en la imagen E.3.

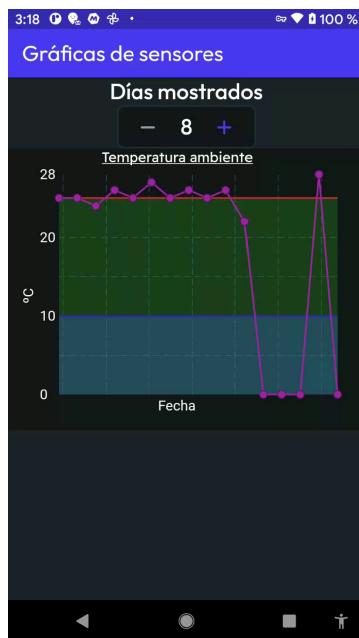


Figura E.3: Imagen de App Android de monitorización de GreenInHouse

E.4. Manual del usuario

Para poner en funcionamiento su nueva maceta educativa Green In House, lo primero que tiene que hacer es encender la maceta pulsando el interruptor de encendido. Cuando arranque el proceso de encendido se iluminará la pantalla táctil y a los pocos segundos aparecerá la ventana de la aplicación de Green In House, en la que se muestra la red Wi-Fi a la que estamos conectados. Al arrancar el sistema dirá que no estamos conectados a ninguna red WiFi como se muestra en la imagen E.4.

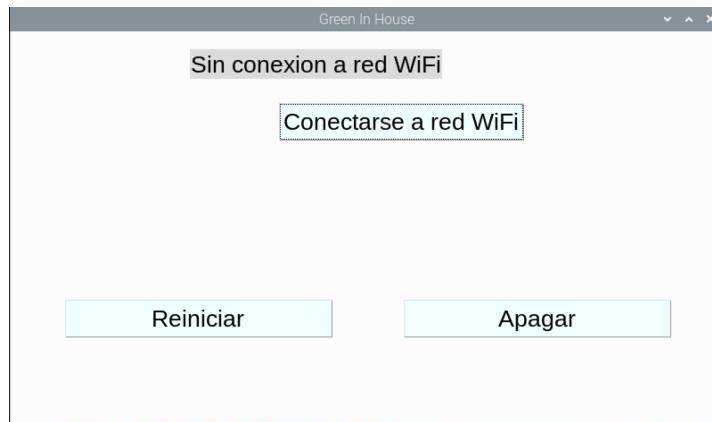


Figura E.4: GreenInHouse no esta conectado a la red WiFi.

Para conectarle a la red WiFi lo que hay que hacer es pulsar sobre el botón **Conectarse a red WiFi** y se mostrará una nueva pantalla en la que pide que introduzcamos el nombre de nuestra red WiFi y su contraseña como se puede ver en la imagen E.5.



Figura E.5: Conectar GreenInHouse a red WiFi.

Para introducir estos datos hay que pulsar sobre los botones azules que dicen **Nombre WiFi** y **Contraseña WiFi**. Al pulsar sobre ellos aparecerá un teclado en la pantalla como se muestra en la imagen E.6 que permitirá introducir dichos datos.



Figura E.6: Teclado en pantalla de GreenInHouse.

Tras introducirlos, hay que pulsar en aceptar y se volverá a la ventana principal. En esta venta nos pedirá que reiniciemos el sistema como se puede ver en la imagen E.7.

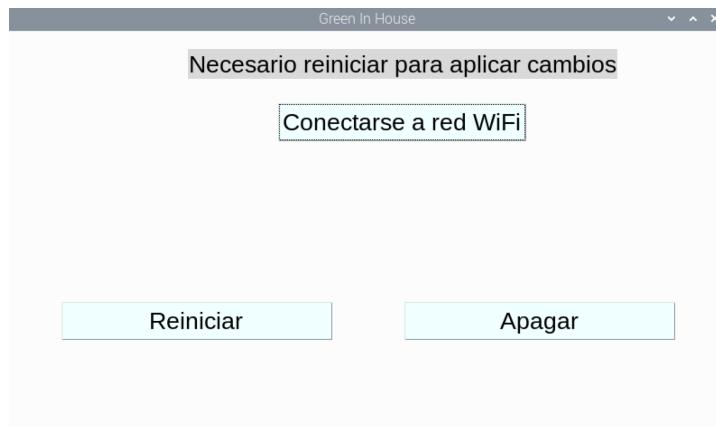


Figura E.7: GreenInHouse necesita reiniciar.

Pulsar sobre el botón reiniciar y el sistema se reiniciará y arrancará de nuevo. Si las credenciales de la red WiFi eran correctas y estamos dentro de su rango de conexión, al encenderse GreenInHouse nos mostrará el nombre de la red WiFi a la que estamos conectados como se muestra en la imagen E.8.



Figura E.8: GreenInHouse conectado a red a red WiFi.

Tras conectar GreenInHouse a la red WiFi, ya podemos utilizar la app móvil de GreenInHouse para ver los registros de los sensores siempre que el dispositivo móvil esté conectado a la misma red WiFi. Al abrir la app se mostrará un gráfico como el de la imagen E.9.



Figura E.9: Gráfica de aplicación Android de GreenInHouse

Para cambiar el número de días a mostrar en la gráfica, utilizar los botones - y + de al lado del valor de días mostrados. Desplazando el gráfico

hacia los lados podemos ver los gráficos de los diferentes sensores del sistema agrupados por tipo de medición y lugar de ubicación del sensor como se puede ver en la imagen E.10.

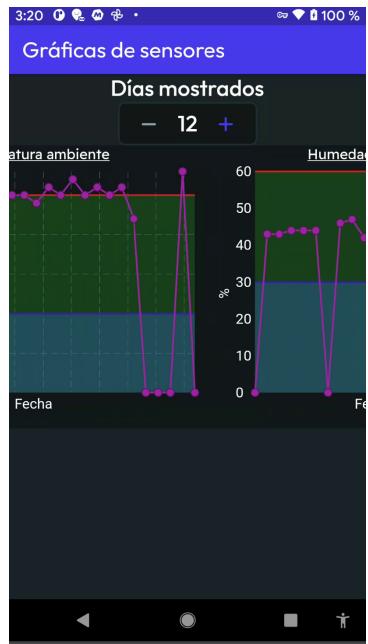


Figura E.10: Gráfica de aplicación Android de GreenInHouse

Bibliografía

- [1] Raspberry EA7TB. Explicación de conexión y funcionamiento de dht11, 2023. [Internet]. URL: <https://www.internetdelascosas.cl/2017/05/19/raspberry-pi-conectando-un-sensor-de-temperatura-y-humedad-dht11/>.
- [2] Raspberry EA7TB. Explicación de conexión y funcionamiento de mcp3008, 2023. [Internet]. URL: <http://raspberry.ea7tb.com/2017/01/15/mcp3008/>.
- [3] Flutter Flow. Flutter flow — generador de código flutter de aplicaciones móviles basado en diseño gráfico, 2023. [Internet]. URL: <https://docs.flutterflow.io/>.
- [4] Python Software Foundation. Tkinter — desarrollo de aplicaciones con entorno grafico para python, 2023. [Internet]. URL: <https://docs.python.org/es/3/library/tkinter.html>.
- [5] Python Software Foundation. venv — entorno virtual python, 2023. [Internet]. URL: <https://docs.python.org/es/3.8/library/venv.html>.
- [6] Hostinger. Ssh — ejecución remota de shell segura, 2023. [Internet]. URL: <https://www.hostinger.es/tutoriales/que-es-ssh>.
- [7] Google I/O. Guia oficial de desarrollo flutter — framework de programación de apps multiplataforma, 2023. [Internet]. URL: <https://esflutter.dev/docs/get-started/codelab>.
- [8] Jhordyess. Configuración de ssh en raspberry para desarrollo remoto desde vscode, 2023. [Internet]. URL: <https://>

- //blog.jhordyess.com/programacion-remota-en-raspberry-pi-con-ssh-y-visual-studio-code.
- [9] Microsoft. vscode — ide ligero y potente, 2023. [Internet]. URL: <https://code.visualstudio.com/docs>.
 - [10] OpenApi. Openapi — swagger de api-rest para python, 2023. [Internet]. URL: <https://swagger.io/specification/>.
 - [11] Circuit Python. Adafruit circuit python — trabajo con sensores y actuadores en python, 2023. [Internet]. URL: <https://docs.circuitpython.org/en/latest/README.html>.
 - [12] Circuit Python. Documentación de uso de mcp3008, 2023. [Internet]. URL: https://github.com/adafruit/Adafruit_CircuitPython_MCP3xxx.
 - [13] SQLAlchemy. Sqlalchemy — sistema gestor de base de datos para python, 2023. [Internet]. URL: <https://www.sqlalchemy.org/>.
 - [14] Oscar Valverde. Green in house repositorio app movil, 2023. URL: https://github.com/ove1001/GreenInHouse_MobileApp.
 - [15] Oscar Valverde. Green in house repositorio maceta, 2023. URL: https://github.com/ove1001/GreenInHouse_PlantPot.



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