



**UNIVERSIDAD DE CASTILLA-LA MANCHA
ESCUELA SUPERIOR DE INFORMÁTICA**

GRADO EN INGENIERÍA INFORMÁTICA

TECNOLOGÍA ESPECÍFICA DE TECNOLOGÍAS DE LA INFORMACIÓN

TRABAJO FIN DE GRADO

Arquitectura IoT basada en Arduino y SigFox para industrias agrícolas

Antonio Rubio Menchero

julio, 2021



UNIVERSIDAD DE CASTILLA-LA MANCHA ESCUELA SUPERIOR DE INFORMÁTICA

TECNOLOGÍA ESPECÍFICA DE TECNOLOGÍAS DE LA INFORMACIÓN

TRABAJO FIN DE GRADO

**Arquitectura IoT basada en Arduino y SigFox para
industrias agrícolas**

Autor: Antonio Rubio Menchero

Tutor Académico: Jesús Barba Romero

Tutor en Pulsia: José Antonio Aranda Almansa

julio, 2021

Arquitectura IoT basada en Arduino y SigFox para industrias agrícolas
© Antonio Rubio Menchero, 2021

Este documento se distribuye con licencia CC BY-NC-SA 4.0. El texto completo de la licencia puede obtenerse en <https://creativecommons.org/licenses/by-nc-sa/4.0/>.

La copia y distribución de esta obra está permitida en todo el mundo, sin regalías y por cualquier medio, siempre que esta nota sea preservada. Se concede permiso para copiar y distribuir traducciones de este libro desde el español original a otro idioma, siempre que la traducción sea aprobada por el autor del libro y tanto el aviso de copyright como esta nota de permiso, sean preservados en todas las copias.

Este texto ha sido preparado con la plantilla L^AT_EX de TFG para la UCLM publicada por [Jesús Salido](#) en GitHub¹ y Overleaf² como parte del curso «[L^AT_EX esencial para preparación de TFG, Tesis y otros documentos académicos](#) impartido en la Escuela Superior de Informática de la Universidad de Castilla-La Mancha.



¹https://github.com/JesusSalido/TFG_ESI_UCLM

²<https://www.overleaf.com/latex/templates/plantilla-de-tfg-escuela-superior-de-informatica-uclm/phjgscmfqtsw>

TRIBUNAL:

Presidente: _____

Vocal: _____

Secretario: _____

FECHA DE DEFENSA: _____

CALIFICACIÓN: _____

PRESIDENTE

VOCAL

SECRETARIO

Fdo.:

Fdo.:

Fdo.:

Resumen

Ante la resistencia y dejadez por parte del sector agrícola ante la digitalización y la gran cantidad de recursos que son destinados a la modernización de la agricultura por parte de la Unión Europea para proyectos que conlleven modernización a nivel digital del sector que no se están pudiendo utilizar debido a la falta de proyectos. Se plantea un proyecto sobre la creación de Arquitectura IoT en tres capas mediante la utilización de un dispositivo Arduino y una serie de sensores para captar variables influyentes sobre el cultivo para posterior transmisión y almacenamiento de estos valores a una base de datos. Con el objetivo de ocupar este hueco en el mercado y dotar a los agricultores que con las nuevas generaciones van perdiendo esa reticencia a la digitalización a mejorar sus proceso de toma de decisiones.

Para la transmisión de datos se utilizará una red LPWAN (Low-Power Wide-Area Network) proporcionada por la plataforma SigFox. Esta red de bajo consumo y largo alcance que se utiliza como canal de comunicación entre el dispositivo Arduino y nuestro servidor de Drupal que almacenará las lecturas realizadas en una base de datos MySQL.

Los parámetros que transmite la placa Arduino se codifican utilizando la librería aportada para envío de mensajes hacia la plataforma backend de Sigfox que nos ayudará a la transmisión de mensajes hacia nuestro endpoint para almacenar los datos de manera correcta en la base de datos. Estos datos serán corroborados con las mediciones realizadas por la AEMET gracias a la API REST proporcionada por AEMET.

Los datos que son almacenados en nuestra base de datos serán utilizados para ofrecer al agricultor información que pueda ayudarles en la toma de decisiones sobre la gestión de sus cultivos, por ejemplo, cambio de sistema de riego para obtener mayor humedad en sustrato, cambio de estructura para obtener un mayor número de horas de radiación solar o detectar posibles fallos en el sistema de riego o ventilación.

Abstract

Given the resistance and laziness on the part of the agricultural sector in the face of digitalisation and the large amount of resources allocated to the modernisation of agriculture by the European Union for projects involving digital modernisation of the sector that cannot be used due to the lack of projects, a project is proposed on the creation of a three-layer IoT architecture using an Arduino device and a series of sensors to capture influential variables on crops for subsequent transmission and storage of these values in a database. A project is proposed on the creation of IoT architecture in three layers by using an Arduino device and a series of sensors to capture influential variables on the crop for subsequent transmission and storage of these values to a database. With the aim of filling this gap in the market and providing farmers who, with the new generations, are losing their reluctance to digitalisation to improve their decision-making process.

For data transmission we will use a LPWAN (Low-Power Wide-Area Network) provided by the SigFox platform. This low-power, long-range network is used as a communication channel between the Arduino device and our Drupal server, which will store the readings in a MySQL database.

The parameters transmitted by the Arduino board are encoded using the library provided for sending messages to the Sigfox backend platform, which will help us to transmit messages to our endpoint to store the data correctly in the database. transmission of messages to our endpoint to store the data correctly in the database. database. This data will be corroborated with the measurements taken by AEMET thanks to the API REST provided by AEMET.

The data that is stored in our database will be used to provide the farmer with information that can help them in making decisions on the management of their information that can help them to make decisions on the management of their crops, e.g. change of irrigation system to obtain more moisture in the substrate for example, change of irrigation system to obtain more moisture in the substrate, change of structure to obtain more hours of solar radiation or detect possible failures, change of structure to obtain a greater number of hours of solar radiation or detect possible faults in the irrigation or ventilation system.

Agradecimientos

Haciendo una referencia al ciclismo ahora que nos encontramos sumergidos en estas fechas tan señaladas me pregunto ¿La vida es una carrera, no? Pues esta es una etapa de esa carrera a la que llamamos vida. Esta etapa en la que estamos a punto de llegar a la meta ha sido dura pero la he disfrutado al máximo. Ahora que ya queda tan poco es el momento de agradecer a todas las personas que nos hemos encontrado y que han hecho posible que llegase hasta este punto.

Para empezar, quería agradecer a la empresa Pulsia Technology, y a la Escuela Superior de Informática la oportunidad que me han brindado de poder realizarme de manera tanto laboral como académica.

A mis amigos de esta etapa universitaria, a todos y cada uno de vosotros que me habéis acompañado dentro y fuera de la escuela. Sin vosotros esta etapa no habría sido lo mismo y me llevo unos grandes amigos para el resto de la vida. Me habéis dado a conocer esos fines de semana que comenzaban los miércoles por la noche y acaban el lunes por la mañana. Me habéis hecho vivir momentos inolvidables.

A mis amigos de los fines de semana, los del pueblo, los que siempre estaban ahí aunque no contestasen por los grupos de whatsapp y que aunque sigan sin contestar, sé que seguirán acompañándome. Gracias, por ayudarme en mis etapas más difíciles y hacerme llegar a esta que he vivido. Por los momentos que hemos vivido y los que viviremos juntos.

A mis compañeros de la oficina, que tanto me habéis ayudado durante este tiempo y que me tendrán que seguir soportando. A mi tutor en la empresa, José Antonio, que sin él nada de esto habría sido posible. A mi tutor en la escuela, Jesús, por hacerme ver que no fue buena idea copiar en Lengua y Literatura.

A mi familia, tanto a los que están como los que ya no están con nosotros. A mis padres, por todo el apoyo durante todos estos años. A mi hermana, que aunque me pide dinero, yo le tengo cariño. A mi abuela, que ha sido la que más me ha sufrido durante la escritura de estas líneas. A mis tíos y primos, por preocuparos de mí en todo momento.

Muchas gracias a TODOS.

*Antonio Rubio Menchero
Ciudad Real, 2021*

Acrónimos

LISTA DE ACRÓNIMOS

TFG	: Trabajo Fin de Grado
IoT	: Internet of Things
LPWAN	: Low-Power Wide-Area Network
AEMET	: Agencia Estatal de Meteorología.
API	: Application Programming Interface.
REST	: REpresentational State Transfer
ESI	: Escuela Superior de Informática
TIC	: Tecnologías de la Información y Comunicaciones
NFC	: Near Field Communication
RAMI 4.0	: Reference Architectural Model Industrie 4.0
HTTP	: Hypertext Transfer Protocol
URI	: Uniform Resource Identifier
JSON	: JavaScript Object Notation
XML	: Extensible Markup Language
HTML	: HyperText Markup Language
CSS	: Cascading Style Sheets
URL	: Uniform Resource Locator
HTTPS	: HyperText Transfer Protocol Secure
UI	: User Interface
GPL	: General Public License
IDE	: Integrated Development Environment
CMS	: Content Management System
SPA	: Single Page Application
MVC	: Modelo-Vista-Controlador
SDK	: Software Development Kit
PWA	: Aplicaciones Web Progresivas
SQL	: Structured Query Language
PWM	: Pulse-Width Modulation
UART	: Universal Asynchronous Receiver-Transmitter
SPI	: Analog Reference
SRAM	: Static Random Access Memory
EEPROM	: Electrically Erasable Programmable Read-Only Memory
PCB	: Printed Circuit Board
UV	: Ultra-Violeta
UVI	: Índice Ultra-Violeta
UML	: Unified Modeling Language
ETL	: Extract, Transform and Load
UTF-8	: 8-bit Unicode Transformation Format
WWW	: World-Wide-Web
USB	: Universal Serial Bus
SSL	: Secure Sockets Layer

Índice general

Resumen	VII
Abstract	IX
Agradecimientos	XI
Acrónimos	XIV
Índice de figuras	XIX
Índice de tablas	XXI
Índice de listados	XXIII
1. Introducción	1
1.1. Contexto	1
1.2. Descripción del Proyecto	1
1.3. Estructura del documento	3
1.4. Digitalización del mundo Agrícola	4
1.4.1. Agricultura de precisión	5
1.4.2. Industria transformadora	6
2. Objetivo	7
2.1. Objetivo Principal	7
2.2. Objetivos Específicos	7
3. Antecedentes	9
3.1. Internet of Things	9
3.1.1. Tipos de redes IoT	10
3.1.2. Low-Power Wide-Area Network	11
3.1.3. Ventajas de las LPWAN	12
3.1.4. Desventajas de las LPWAN	13
3.1.5. RAMI 4.0	14
3.2. API REST	15
3.2.1. ¿Qué es REST?	16
3.2.2. ¿Qué es una API REST?	16
3.2.3. Ventajas de API REST	17
4. Herramientas y Metodología de Trabajo	19
4.1. Metodología	19
4.1.1. SCRUM	20
4.1.2. Equipos Scrum	20

4.1.3.	Eventos Scrum	22
4.1.4.	Artefactos Scrum	23
4.1.5.	Metodología de desarrollo en cascada	24
4.1.6.	Adaptación al proyecto	26
4.2.	Planificación y costes	27
4.2.1.	Costes Suscripción SigFox	27
4.2.2.	Costes Dispositivo Siabox	28
4.2.3.	Diagrama de Gantt	28
4.3.	Herramientas Software utilizadas	28
4.3.1.	Redmine	28
4.3.2.	SigFox Backend	29
4.3.3.	Postman	29
4.3.4.	SWAGGER	30
4.3.5.	Drupal 8	30
4.3.6.	Arduino IDE	31
4.3.7.	Angular	31
4.3.8.	Ionic	32
4.3.9.	Capacitor	32
4.3.10.	Base de Datos	33
4.3.11.	Control de Versiones	33
4.3.12.	Documentación	33
4.4.	Dispositivos Hardware utilizados	34
4.4.1.	Arduino MKRFOX 1200	34
4.4.2.	Sensor YL-69 y módulo YL-39	35
4.4.3.	Sensor TSL2591	36
4.4.4.	Sensor DHT11	37
4.4.5.	Sensor UVM-30A	38
5.	Resultados	41
5.1.	Sprint 1.Configuración de Base de Datos	41
5.1.1.	Planificación Sprint 1	41
5.1.2.	Ejecución Sprint 1	41
5.1.3.	Conclusión Sprint 1	44
5.2.	Sprint 2.Desarrollo de Módulo Drupal Dedicado	44
5.2.1.	Planificación Sprint 2	44
5.2.2.	Ejecución Sprint 2	45
5.2.3.	Conclusión Sprint 2	52
5.3.	Sprint 3.Visualización de Datos	53
5.3.1.	Planificación Sprint 3	53
5.3.2.	Ejecución Sprint 3	54
5.3.3.	Conclusión Sprint 3	64
5.4.	Sprint 4.Montaje y Configuración Hardware	65
5.4.1.	Planificación Sprint 4	65
5.4.2.	Ejecución Sprint 4	65
5.4.3.	Conclusión Sprint 4	69
5.5.	Sprint 5.Comunicación Hardware-Plataforma	70
5.5.1.	Planificación Sprint 5	70
5.5.2.	Ejecución Sprint 5	70
5.5.3.	Conclusión Sprint 5	72
5.6.	Sprint 6.Documentación del Proyecto	73
5.6.1.	Planificación Sprint 6	73
5.6.2.	Ejecución Sprint 6	73

5.6.3. Conclusión Sprint 6	74
6. Conclusiones	75
6.1. Justificación de competencias adquiridas	75
Bibliografía	77
A. Manual de Montaje	81
B. Manual de Despliegue	85
C. Estructura de Módulos	87
C.1. Modulo SigFoxREST	87
C.2. Modulo IonicREST	89
D. Diagrama de Gantt	93
D.0.1. Tablas Estimación/Duración	94
E. Diagramas UML de Clases	95
F. Dirección Pruebas App Móvil	101

Índice de figuras

1.1. Esquema general de la Arquitectura	2
1.2. Ilustración de digitalización de la agricultura	4
3.1. Situación como tecnología inalámbrica	10
3.2. Ciclo de vida del Producto RAMI 4.0	14
3.3. Arquitectura RAMI 4.0	15
3.4. Descripción API REST	15
4.1. Diagrama de Metodología en Cascada	25
4.2. Arduino MKRFOX1200	34
4.4. Sensor Luminico TSL2591	37
4.5. Sensor Ambiental DH11	37
4.6. Indice UV	38
4.7. Sensor indice Ultravioleta UVM-30A	39
5.1. Planificación Sprint 1.Configuración Base de Datos	41
5.2. Requisitos de Almacenamiento	42
5.3. Diagrama Base de Datos	42
5.4. Estructura de tipo de contenido Dispositivo	43
5.5. Estructura de tipo de contenido Cultivo	43
5.6. Estructura de tipo de contenido Valor Sensor	43
5.7. Estructura de tipo de contenido Alarma Sensor	44
5.8. Conclusión Sprint 1.Configuración Base de Datos	44
5.9. Planificación Sprint 2.Desarrollo de módulo dedicado	45
5.10. Requisitos Funcionales de módulo Configuración Routing	45
5.11. Diagrama UML de Casos de Uso de módulo Configuración Routing	46
5.12. Petición HTTP con Postman para registro de valores de sensor	47
5.13. Datos del contenido registrado Valor Sensor	47
5.14. Registro almacenado en el contenido Valor Sensor	47
5.15. Diagrama UML de secuencia de Registro de datos ambientales	48
5.16. Requisitos Funcionales Comprobación AEMET	48
5.17. Diagrama UML Casos de Uso Comprobación AEMET	49
5.18. Información requerida en la cabecera petición AEMET	49
5.19. Respuesta recibida a la petición GET	49
5.20. Simulación generación de alarmas con Postman	50
5.21. Evidencia de generación de registro y alarma correspondiente	51
5.22. Valores de Alarma Sensor registrada	51
5.23. Diagrama UML de Secuencia Sprint 2 completo	52
5.24. Conclusión Sprint 2.Desarrollo de módulo dedicado	53
5.25. Planificación Sprint 3.Visualización de Datos	54
5.26. Requisitos Funcionales Sprint 3	54

5.27. Diagrama UML de Casos de Uso Sprint 3	55
5.28. Bocetos aplicación móvil	56
5.29. Simulación de Login con Postman	58
5.30. Simulación de obtención de token de sesión con Postman	59
5.31. Diagrama UML de secuencia de Login	59
5.32. Bocetos aplicación móvil 1	60
5.33. Bocetos aplicación móvil 1	61
5.34. Diagrama UML de secuencia visualización de luminosidad	62
5.35. Diagrama UML de secuencia visualización de alarma	62
5.36. Diagrama UML de secuencia cerrar de sesión	63
5.37. Conclusión Sprint 3. Visualización de datos	64
5.38. Planificación Sprint 4.Montaje y Configuración Hardware	65
5.39. Código Arduino para obtener credenciales	66
5.40. Resultado de ejecución código de obtención de credenciales	66
5.41. Activación de dispositivo en plataforma SigFox	66
5.42. Esquema de Conexiones del dispositivo Siabox	67
5.43. Resultado Montaje dispositivo Siabox	68
5.44. Requisitos funcionales definidos para el Sprint 4	68
5.45. Diagrama de de Casos de Uso Sprint 4	68
5.46. Captura de salida por monitor serie	69
5.47. Conclusión Sprint 4.Montaje y Configuración Hardware	70
5.48. Planificación Sprint 5.Comunicación Hardware-Plataformas	70
5.49. Creación de Callback para comunicación con servidor	71
5.50. Configuración de Callback para envio de información al servidor	71
5.51. Espectro de cobertura de red SigFox	72
5.52. Confirmación de envío de Callback de forma adecuada	72
5.53. Conclusión Sprint 5.Configuración Base de Datos	72
5.54. Planificación Sprint 6.Documentación del Proyecto	73
5.55. Conclusión Sprint 6.Documentación del Proyecto	74
A.1. Distribución de elementos sobre PCB	81
A.2. Distribución de Orificios en Caja de Seguridad	82
A.3. Esquema de Conexiones del dispositivo Siabox	83
A.4. Ubicación de sensores	83
B.1. Despliegue Dispositivo	85
B.2. Despliegue Sensor Humedad	85
C.1. Estructura de Ficheros de módulo personalizado SigFox_REST	87
C.2. Fichero de información módulo SigFox_REST	88
C.3. Fichero de configuración módulo SigFox_REST	88
C.4. Estructura de módulo Ionic REST	89
C.5. Archivo Info Ionic REST	90
C.6. Archivo Routing Ionic REST	90
D.1. Diagrama de Gantt	93
D.2. Diagrama de Gantt	93
E.1. Diagrama UML de clases del módulo Sigfox_REST Prueba de configuración de routing	96
E.2. Diagrama UML de clases Comprobación de datos con AEMET	97
E.3. Diagrama UML de clases Ionic REST (Parte Servidor)	98
E.4. Diagrama UML de clases de Login	99
E.5. Diagrama UML de Clases App Movil	100

Índice de tablas

3.1. Comparativa de Tecnologías LPWAN	12
4.1. Costes de Suscripción SigFox	28
4.2. Costes de Dispositivo Siabox	28
4.3. Características Técnicas Arduino MKRFOX1200	35
4.4. Especificación YL-69 y YL-39	36
4.5. Especificación TSL2591	37
4.6. Especificaciones Sensor Humedad y Temperatura DHT11	38
4.7. Especificaciones Sensor UVM-30A	39
D.1. Tabla de Estimación vs Duración Real	94

Índice de listados

5.1. Instrucción para Instalación de librerías	63
5.2. Instrucción para Creación de Capacitor a proyecto existente	63
5.3. Instrucción para Creación de proyecto WWW	63
5.4. Instrucción para Creación de Proyecto Android	64
5.5. Instrucción para Apertura de Android Studio	64

CAPÍTULO 1

Introducción

En el siguiente capítulo, se muestra una perspectiva general de todo lo relativo a este Trabajo Fin de Grado (TFG). En primer lugar, se expondrá motivación y contexto general. Después se seguirá con la explicación del funcionamiento de la arquitectura a desarrollar. Para finalizar, se detallará la estructura de este documento.

1.1. CONTEXTO

El motivo del nacimiento del presente Trabajo Fin de Grado (TFG, de ahora en adelante) surge del convenio Forte. Siguiendo una de las líneas de acción del programa profESIonalízate lanzado por la Escuela Superior de Informática (ESI) de Ciudad Real, este programa permite el fortalecimiento de las competencias a nivel profesional de los egresados de la ESI.

El principal objetivo del programa FORTE es permitir a los estudiantes que se encuentran próximos a finalizar del grado e idealmente, sólo les falte por realizar las prácticas y el TFG para finalizar el grado. Tengan la posibilidad de adentrarse en el desarrollo de un proyecto real que se esté llevando a cabo en alguna empresa y realizar el TFG dentro de esta. Al estudiante, se le brinda la oportunidad de utilizar su tiempo dedicado a prácticas en empresa para realizar su TFG dentro de esta. El estudiante será tutorizado por un profesor parte de la ESI y un ingeniero por parte de la empresa seleccionada. El tutor de empresa, tendrá una visión más práctica sobre el TFG. Por su parte, el tutor de la ESI podrá acometer tanto labores prácticas siempre que no interfieran con la visión de la empresa pero, salvaguardando el cumplimiento de los objetivos desde el punto de vista académico, como tareas más formales y académicas, comprobando que se cumple la normativa existente, estándares de calidad y competencias adscritas a la intensificación del estudiante.

Yo, como estudiante del último curso de Grado de Ingeniería Informática me embarco en este TFG debido a la siguiente razon, la posibilidad de integrar un gran grupo de tecnologías para generar una arquitectura IoT gracias a la utilización de API REST que influya sobre el sector que más arraigo tiene en nuestra zona.

1.2. DESCRIPCIÓN DEL PROYECTO

Nuestra empresa se localiza en una comarca donde el sector agrícola se sitúa entre los más punteros a nivel mundial. Las empresas pertenecientes a este sector empiezan a buscar en la informática la posibilidad de mejorar el rendimiento de las producciones. Esto es debido a la escasez de recursos

hídricos en la zona y la búsqueda de un cultivo más ecológico mediante la utilización de una menor cantidad de sustancias químicas como pueden ser fertilizantes e insecticidas. En el caso concreto de este TFG se va a abordar desde dos perspectivas diferentes, pero con una finalidad ligada entre sí. La primera parte tratará sobre la creación de la arquitectura IoT basada en SigFox y Arduino para la recolección de datos sobre parámetros ambientales y creación de una sencilla aplicación móvil para la posible visualización de información generada por nuestro sistema. Como finalidad se busca que esta arquitectura obtenga información con suficiente valor como para influir en la toma de decisiones de los agricultores.

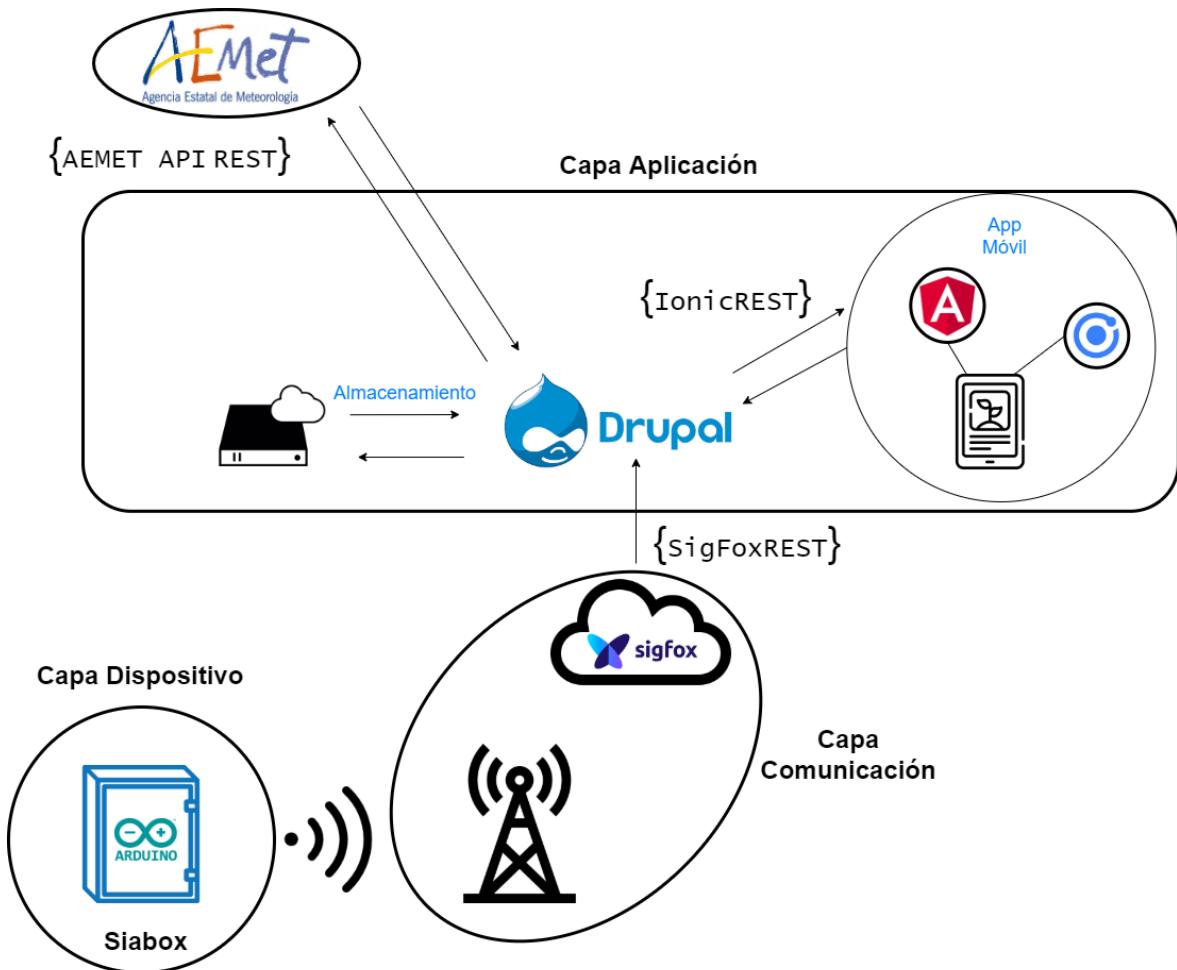


Figura 1.1: Esquema general de la Arquitectura

Para la creación de la Arquitectura IoT basada en SigFox y Arduino se realizará el montaje de un pequeño dispositivo donde se integrarán todos los componentes necesarios para la captación de información del entorno y posterior envío de esta a la plataforma de SigFox Backend mediante una LPWAN. Este dispositivo será denominado Siabox. Los componentes se pueden dividir en dos grupos según función. Un primer grupo, donde situaremos los sensores ambientales (humedad ambiente, humedad en sustrato, temperatura ambiente, luminosidad y radiación ultravioleta) que se encargarán de obtener información del medio. Un segundo grupo, donde se encontrarán tanto la placa Arduino con el módulo SigFox integrado y una antena para transmitir la información. La mayoría de las zonas donde se cultiva se encuentra deslocalizadas, por no decir aisladas. Esto hace que nos hayamos

decantado por la utilización de una tecnología LPWAN o 0G como es SigFox.

Los datos recogidos por los sensores cada un cierto periodo de tiempo que oscilará entre 30 y 35 minutos debido a que existe una restricción por parte de nuestro proveedor. Los datos son recogidos, empaquetados y enviados hacia el proveedor de servicio SigFox que registrará el mensaje. Posteriormente, esta información será transmitida desde el SigFox Backend hacia un endpoint anteriormente configurado en nuestro sistema. Primero, los datos transmitidos serán almacenados y más tarde ser procesados para generar posibles alarmas y así el usuario final utilizar esta información como referencia a la hora de tomar decisiones.

La aplicación móvil que será desarrollada utilizando tecnologías web como Angular e Ionic para su posterior conversión mediante la utilización de Capacitor. Esta aplicación contará con una venta de login para controlar el acceso a la información del sistema y recopilará la información correspondiente a los dispositivos de este usuario. El usuario tendrá la posibilidad de visualizar la información de cada uno de los dispositivos de manera gráfica y revisar las alarmas generadas con respecto a la información que obtendremos de la AEMET mediante su API REST.

1.3. ESTRUCTURA DEL DOCUMENTO

Este documento se estructura de la siguiente manera:

- Capítulo 1: Introducción
 - En el capítulo actual se expone la motivación del proyecto, así como, el contexto general del TFG. También se da una visión de cómo está estructurado el documento.
- Capítulo 2: Objetivos
 - El capítulo 2 se expone cuál es el objetivo principal de este TFG, además de los objetivos específicos que llevarán a alcanzar el objetivo principal propuesto.
- Capítulo 3: Antecedentes
 - En el capítulo 3 se describe la tecnología y su aplicabilidad a distintos sectores, se realiza una comparación con respecto al resto de tecnologías de red y se enumeran ventajas y desventajas de esta.
- Capítulo 4: Herramientas y Metodología de Trabajo
 - En el capítulo 4 se exponen los dispositivos hardware y herramientas software utilizadas para el desarrollo del TFG y las metodologías de trabajo que se han usado, además de la explicación de cómo las hemos aplicado a este TFG en concreto.

- Capítulo 5: Resultados

- En el capítulo 5 se explica detalladamente cómo se ha desarrollado el proyecto. Para la gestión del proyecto se utilizará la metología SCRUM que nos ayudará en dividir el proyecto en 6 partes. En cada uno de las partes se realiza una descripción detallada de la planificación, ejecución y conclusión de las etapas.

- Capítulo 6: Conclusiones

- En el capítulo 6 se expondrán las conclusiones obtenidas tras el desarrollo final del proyecto. También las competencias y destrezas adquiridas por el desarrollador tras la finalización del proyecto. Así como, las ideas futuras posibles para la aplicación.

1.4. DIGITALIZACIÓN DEL MUNDO AGRÍCOLA

La importancia de la implantación de las TIC (Tecnologías de la Información y Comunicaciones) en el sector agrario, y más en la agricultura, cada vez está más presente en el mercado, ya que supone un gran avance para un sector que no ha crecido al mismo ritmo que otros sectores productivos teniendo menor impacto en la economía de un país como España.

El proyecto nace de la necesidad de incorporar mecanismos de prevención para una agricultura sostenible, dónde no existen medios diferenciadores que permitan evolucionar en el ahorro de elementos químicos contaminantes y el uso de recursos hídricos.



Figura 1.2: Ilustración de digitalización de la agricultura Fuente:[14]

Según [10], actualmente, los sistemas tradicionales de producción y transformación de la industria agroalimentaria ya han empezado a trabajar codo con codo con las tecnologías digitales, esta 'revolución verde' tiene el objetivo de dar un salto de calidad para encarar de forma exitosa hacia el

camino de sostenibilidad a largo plazo.

Las tecnologías digitales están redefiniendo a gran velocidad las cadenas de valor y los procesos de negocio. Esto hace que exista una gran dependencia y se empiece a hablar de transformación digital de la economía. Existen dos bases principales de este fenómeno:

- **La importancia de los datos.** Como punto de inflexión a la hora de tomar decisiones y optimizar los procesos de negocio. Esto ha dado lugar al nacimiento del concepto 'data economy', que incluye todas las herramientas, productos y servicios que proporcionan los datos y el procesamiento adecuado de los mismos para la toma de decisiones.
- **Las tecnologías habilitadoras.** Estas son necesarias para liberar todo el potencial de la economía digital. Hablamos de big data y data analytics, tecnologías cloud, ciberseguridad, Internet of Things (IoT), robótica, servicios móviles, y social media.

El sector agroalimentario también se encuentra en proceso de digitalización. Para ser más concretos, se encuentra en el inicio de una gran transformación, que afecta tanto a la mejora de los procesos como a la redefinición del modelo de negocio. En las siguientes secciones se expondrán las tecnologías más relevantes al igual que sus beneficios en las principales fases de la cadena de valor agroalimentario.

1.4.1. Agricultura de precisión

La agricultura de precisión, también conocida como 'agricultura 4.0' se basa en dos principios: sensorización y robótica.

- **Sensorización.** Las tecnologías IoT permiten monitorizar con granularidad fina todos los parámetros que intervienen en la producción (uso de los recursos, fertilidad del suelo, aparición de plagas y enfermedades). La combinación de los datos sensorizados con modelos agronómicos y fuentes de datos externas (datos meteorológicos, satélites) permite tomar, a través de algoritmos de análisis de datos, decisiones óptimas a lo largo de todo el proceso de producción. Estas tecnologías IoT pueden influir de manera positiva y reducir el consumo de recursos insumos (agua, pesticidas, nutrientes), en la reducción del impacto de plagas. Además, de las tecnologías relacionadas con IoT y análisis de datos, los servicios en la nube y los servicios serán grandes apoyos del mundo agrícola en el futuro.
- **Robótica agraria.** Los avances en robótica e inteligencia artificial posibilitan la automatización de tareas que se pensaba que era imposible que las realizaran máquinas en vez de humanos. Se está viendo en las explotaciones agrícolas tractores autónomos que realizan tareas de arado, plantación y recolección del fruto. Al tiempo que generan una enorme cantidad de datos que más tarde pueden ser analizados e integrados en la toma de decisiones, no sólo sobre el proceso productivo, sino también con fines de mantenimiento predictivo de la maquinaria.

1.4.2. Industria transformadora

La integración de sistemas IoT y ciberfísicos con procesado avanzado de datos permite integrar no solo todos los procesos operativos internos sino también los de proveedores y clientes en la misma cadena de valor, y conseguir así unos niveles de eficiencia y flexibilidad en los procesos industriales nunca logrados anteriormente, en entornos de fabricación cada vez más autónomos, una tendencia a la que la industria agroalimentaria no escapará.

CAPÍTULO 2

Objetivo

En este capítulo se describe el objetivo principal que se pretende alcanzar con el desarrollo del presente TFG, destacando el problema específico que trata de resolver. Además, se enumeran una serie de subobjetivos que se corresponden con las diferentes partes que constituyen el sistema final a desarrollar.

2.1. OBJETIVO PRINCIPAL

El objetivo **principal** de este TFG consiste en la creación de una arquitectura IoT de tres capas para la recopilación de datos ambientales que afecta a un cultivo y brindarle la posibilidad al usuario de este servicio, agricultor, de visualizar las condiciones ambientales a los que esta expuesto su cultivo y generar información detallada para ayudar al usuario a la hora de tomar decisiones sobre las acciones que debe tomar sobre ese cultivo.

2.2. OBJETIVOS ESPECÍFICOS

En este apartado, se enumeran y describen los objetivos específicos mediante los cuales se busca la consecución del objetivo principal del proyecto.

- **Producto para obtención de parámetros ambientales como objetivo.** Se procederá al diseño y montaje de un dispositivo que utilizando un conjunto de sensores de captación de parámetros ambientales y una placa programable se obtenga un dispositivo con la funcionalidad de extraer, transformar y cargar (en este caso transmitir) los valores leídos por esa serie de sensores que integran el dispositivo.
- **Proporcionar conectividad o la posibilidad de transmitir valores desde zonas reconditas.** Utilización del módulo de comunicación incorporado en la placa programable para realizar envío de valores ambientales desde zonas deslocalizadas y mediante mensajes de tamaño pequeño utilizando LPWAN.
- **Creación de un servicio de almacenamiento.** Se procederá un servicio orientado al almacenamiento de valores enviados desde el dispositivo de recepción de datos ambientales mediante una pasarela perteneciente al proveedor de LPWAN.

- **Asegurar la precisión de valores leídos por los sensores ambientales.** Para ello se desarrollará software para realizar una comparación entre los valores almacenados en la base de datos y las predicciones realizadas por la AEMET que serán obtenidas mediante su API REST con la finalidad de testar la precisión de los sensores utilizados en el dispositivo y la veracidad de las predicciones suministradas por AEMET.
- **Visualización de información con valor por parte de los usuarios.** Desarrollo de una aplicación móvil con la finalidad de ofrecer a los usuarios la posibilidad de monitorizar las mediciones ambientales que realiza el dispositivo que tienen asignado de una manera visual y estructurada. Además, se incorpora un sistema de alarmas con el objetivo de advertir sobre la diferencia observada en las comprobaciones.

CAPÍTULO 3

Antecedentes

Este capítulo tiene como propósito describir en qué consiste la tecnología elegida para este proyecto, IoT y las diferentes alternativas de redes que incorpora la tecnología IoT, proveedores del tipo específico utilizado para este proyecto e de identificar las ventajas y desventajas de esta tecnología. Además, se expondrá un ejemplo de intento de estandarizar los proyectos de IoT. Para finalizar, se dará a conocer la tecnología principal que en la que nos apoyaremos para integrar las diversas partes de la arquitectura propuesta.

3.1. INTERNET OF THINGS

Lo primero y más importante es acotar la definición de lo que representan el término Internet of Things. IoT podría definirse como la agrupación de interconexiones entre dispositivos y objetos a través de una red, donde puedan comunicarse y ser visibles entre ellos. Cada vez se van encontrando más tipos de dispositivos que incluyen esta tecnología. Podemos encontrar desde simples sensores hasta objetos de la vida cotidiana como pueden ser frigoríficos o prendas de ropa. Todo objeto en el que podamos llegar a pensar, podría llegar a tener su versión con la tecnología IoT incorporada.

En los últimos años, el término Internet of Things (IoT) se está volviendo cada vez más popular. Debido a que lo que en el pasado se creía que era sueño, se está convirtiendo en una realidad. La fama de esta tecnología radica en la mejora de las condiciones de la vida cotidiana de las personas y en las mejoras que ha acarreado en el sector empresarial.

Las aplicaciones de IoT son casi infinitas, pero en el siguiente listado se enuncian algunos de los ejemplos más representativos.

- El ejemplo del frigorífico de una casa donde los alimentos que se conservan en él, tienen una fecha de caducidad. En este caso particular, se podría conectar el frigorífico a Internet para que el usuario recibiese una alerta en el caso de que alguno de los productos esté cerca de cumplir su fecha de caducidad.
- Un ejemplo en el sector ganadero, puede ser aplicación de IoT para la geolocalización y monitoreo biométrico como factor de ayuda a los ganaderos a que sus animales siempre estén controlados e inferir sobre la toma decisiones en sus cuidados.

- Si pensamos a nivel industrial, IoT está siendo utilizado en muchas plantas de producción donde sensores y dispositivos están conectados a la red, con el fin de realizar un análisis de datos y una generación de alertas y mensajes. Los usuarios responsables del funcionamiento de esas plantas, recibirán estos mensajes y alertas para tomar las acciones necesarias.

Gracias a los ejemplos anteriores, podemos apreciar que IoT es algo muy amplio con una variedad infinita de aplicaciones, sin embargo, que los dispositivos estén conectados o que tengan un cierto grado de inteligencia, no es algo nuevo, la clave de IoT es la capacidad de que los dispositivos estén conectados a Internet, a ser posible, de manera directa. Que tengan la capacidad de recopilar información y transmitirla a otros dispositivos, y que esta información pueda ser guardada y procesada para mejora del funcionamiento del dispositivo u otros.

3.1.1. Tipos de redes IoT

El tipo de conectividad que se utiliza como parte del IoT viene condicionado por el dispositivo, su función y los usuarios. Normalmente, la distancia a la que los datos deben viajar (corta o larga) determinan el tipo de red IoT necesaria para satisfacer las necesidades de la aplicación. Para verlo de una manera más clara, utilizamos la siguiente figura 3.1

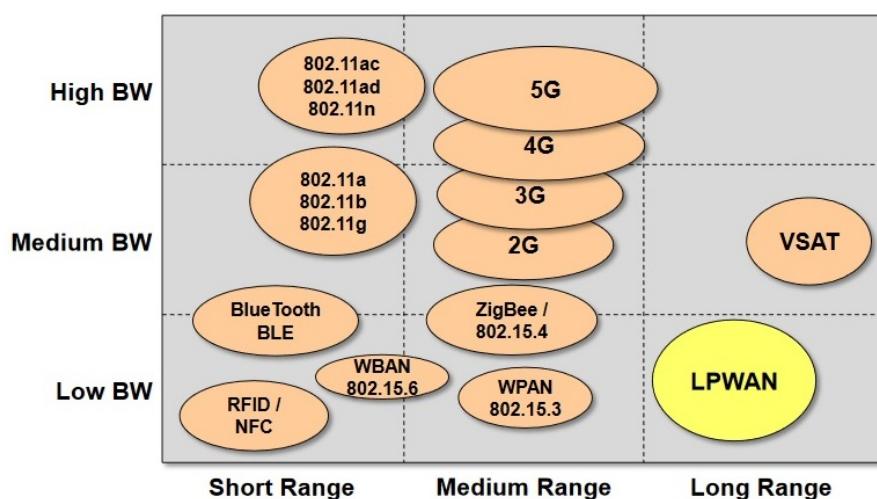


Figura 3.1: Comparativa de tipos de red por ancho de banda y alcance Fuente:[7]

■ Redes de corto alcance y bajo consumo

Las redes de baja potencia y corto alcance están destinadas para su uso en hogares, oficinas y otros espacios de tamaño reducido. Los dispositivos necesitarán de baterías pequeñas debido a la posibilidad de no existir suministro eléctrico y su uso suele ser económico. A continuación, se listan unos ejemplos comunes:

Bluetooth: La tecnología Bluetooth envía señales de voz y datos hasta una distancia de 10 metros y es ideal para la transferencia de datos de alta velocidad.

NFC: Conjunto de protocolos para la comunicación entre dos dispositivos electrónicos que se encuentren a una distancia de 4 cm o menos. NFC ofrece una conexión de baja velocidad

con una configuración sencilla que se puede usar para iniciar conexiones inalámbricas de más capacidad.

Wi-Fi: El bajo costo del uso del Wi-Fi lo convierte en un estándar en hogares y oficinas. Sin embargo, es posible que no sea la opción adecuada para todos los escenarios, por su alcance limitado y el consumo energético ininterrumpido.

■ **Redes de área extensa de bajo consumo (LPWAN)**

Las redes LPWAN permiten la comunicación en un radio mínimo de 500 metros, tienen un consumo de energía mínimo y se usan para la mayoría de los dispositivos IoT. En el siguiente apartado se realizará una descripción extensa de este tipo de redes que será el utilizado en el proyecto. En el siguiente listado se enumeran y describen brevemente algunos ejemplos comunes:

LoRaWan: Las redes de área extensa de largo alcance (LoRaWAN) conectan dispositivos móviles, seguros y con batería bidireccional.

SigFox: Este proveedor internacional de redes de IoT ofrece redes inalámbricas para conectar objetos de baja potencia que emiten datos continuamente. En los siguientes apartados, esta sección será ampliada debido a que ha sido la seleccionada para la creación de la arquitectura propuesta.

NB-IoT: Tecnología LPWAN de operador ofrecida por las mismas compañías de telecomunicaciones que ofrecen las redes de telefonía móvil, es decir, es la tecnología que las Telecom ofrecen al ecosistema IoT. Actualmente, es una muy desarrollada pero se cree que avanzará en los próximos años.

Wi-Sun: Estas redes basadas en LTE son la opción menos costosa. Sientan las bases para Cat-M, una tecnología que reemplazará al 2G.

3.1.2. Low-Power Wide-Area Network

Las redes LPWAN son las tecnologías de comunicación que permiten transmitir datos entre dispositivos y estación base que se encuentran separados por grandes distancias (centenares de metros e incluso kilómetros), con un bajo consumo de energía. Las tres características técnicas que hacen a las redes LPWAN excelentes para su uso en IoT son:

- **Bajo consumo eléctrico:** El protocolo se fundamenta en el uso de dispositivos cuyas baterías permiten una duración de años en lugar de semanas y meses.
- **Cantidad de datos transmitidos:** La idea de este tipo de red es regular el transporte no constante de paquetes de datos con tamaños pequeños.
- **Alcance geográfico:** LPWAN está diseñado para realizar transmisiones de datos de manera inalámbrica entre dispositivos separados por distancias en el rango de kilómetros.

Estas características además diferencian a LPWAN de otras tecnologías inalámbricas como Wi-Fi, Bluetooth, 3GPP y Zigbee.

Un punto técnico a destacar sobre las LPWAN es la banda de radiofrecuencia en la que suelen trabajar. Los diferentes productos que incorporan LPWAN que se encuentran en Europa, trabajan en la banda ISM (Industrial, Scientific and Medical), la que representa un espectro intencionalmente que se reserva al uso no comercial asociado con la industria, ciencia y servicios médicos. Esta banda es muy importante porque este grupo de frecuencias son utilizadas para realizar un pago por licencia, siempre y cuando se respeten las restricciones en niveles de potencia transmitida. Esto representa otra diferencia de LPWAN y las tecnologías celulares, por ejemplo, en las cuales es necesario pagar por el derecho a transmitir en una frecuencia específica.

Hay que tener en cuenta que para LPWAN existe una diferencia de frecuencia utilizada dependiendo de la región en el que se encuentre; en Europa se utilizan las frecuencias entre 867-869 MHz, en Estados Unidos se utiliza la franja 902-928MHz. Esto hace que los dispositivos de hardware puedan utilizarse en varios países siempre y cuando realicen ajustes en lo relativo a software que permitan poder trabajar en una frecuencia en particular.

Existen diversas alternativas como hemos visto en la enumeración anterior. Para poder compararlas de manera rápida y clara se hace uso de una tabla 3.1 con tecnologías y los valores de cada uno de los factores claves.

Factor	SigFox	NB-IoT	LoRaWAN	Wi-SUN
Cobertura(Km)	10(urbano),40(rural)	1(urbano),10(rural)	5(urban),20(rural)	2-3 LoS ¹
Transferencia(Kbps)	0.1	200	0.3-27	50-300
Max.Payload(bytes)	12(UP),4(DOWN)	1600	243	Tamaño de payload
Max.Mensajes/dia	140(UP),8(DOWN)	Sin límite	Sin límite	Sin límite
Escalabilidad	10^6	$55 * 10^3 / \text{cell}$	10^4	$5 * 10^3$
Coste/Espectro	<2€/Gratis	>20/>500M€/Mhz	3-5€/Gratis	NA/Gratis

Tabla 3.1: Comparativa de Tecnologías LPWAN Fuente:[8]

3.1.3. Ventajas de las LPWAN

Para poder realizar una comparativa debemos de conocer tanto los beneficios como los inconvenientes que nos aporta las LPWAN.

- El bajo consumo por parte de los dispositivos que permite les permite funcionar durante cerca de un año con una sola carga y en algunos casos la vida útil de un dispositivo puede llegar a ser de 10 años.
- La escalabilidad de la red para adaptarse a escenarios que requieran una mayor capacidad.
- Los precios de cada módulo de conexión están sufriendo una bajada progresiva de precio que desembocará en la posibilidad de comprar un módulo por tan solo unos pocos euros.
- Las LPWAN incorporan una conectividad segura de End to End y servicios de autenticación eficaces que son completamente compatibles con los requisitos de las aplicaciones IoT.
- Se basan en una transferencia de datos optimizada que es compatible con bloques de tamaño por lo general pequeños y enviados de manera intermitente.

3.1.4. Desventajas de las LPWAN

Al igual que hay que conocer las ventajas hay que hacer lo mismo con las desventajas, LPWAN aunque no lo parezca, también tiene desventajas. Se puede decir que, sus propias ventajas, hace que tenga desventajas.

- La baja velocidad de transmisión hace que sea imposible manejar grandes cantidades de información como elementos multimedia. Aunque, LPWAN nos permita crear redes de sensores y dispositivos esto no implica que la telemetría que requieran transportar no debe ser muy grande. Esto hace que se descarten los elementos que requieran una gran cantidad de información.
- Al igual que en otro tipo de red, existen registros sobre problemas de atenuación de señales cuando se ubican dispositivos en edificios y zonas cubiertas. Por lo tanto, En espacios abiertos funciona de una manera eficiente.
- La conectividad entre dispositivos y aplicación servidora no es constante y puede ser unidireccional (del dispositivo a un elemento de captura), lo que dificulta actividades como el control de movimiento de objetos en tiempo real. Aunque, hay que decir que la mayoría de las implementaciones de LPWAN permiten comunicaciones bidireccionales.

3.1.5. RAMI 4.0

El Internet of Things se encuentra en un estado de crecimiento y por lo tanto, no es un concepto que se haya madurado de manera correcta todavía. Existe un gran esfuerzo por parte de la comunidad que está relacionada con las tecnologías IoT o SMART aplicadas a distintos sectores en conseguir unos estándares. En caso, el proyecto está encuadrado dentro de un sector que sería la *Agricultura 4.0* y el estándar que se va a tratar en este apartado RAMI 4.0 está relacionado con la *Industria 4.0* pero, trata la problemática relacionada con la arquitectura y existen una gran cantidad de puntos en común.

RAMI 4.0 (Reference Architectural Model Industrie 4.0) es un modelo que ha sido diseñado por Platform Industrie 4.0 para tener un modelo de referencia para el concepto de Industria SMART o industria conectada. Este modelo, desecha el anterior que estaba basado en una estructura piramidal donde las funcionalidades estaban definidas en el nivel se encontraban los distintos componentes de la red. Esta nueva arquitectura opta por no utilizar jerarquías definidas en su modelo. Esto quiere decir que todos los dispositivos forman parte de la red y las comunicaciones se dan dentro de esta de manera libre.

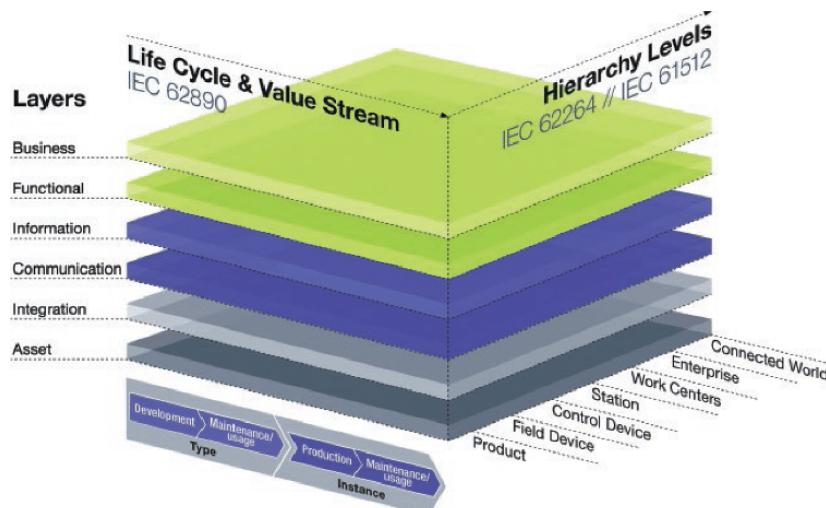


Figura 3.2: Ciclo de vida del Producto RAMI 4.0 Fuente:[17])

Tridimensional: este modelo apuesta por una visión tridimensional que engloba varios aspectos del proceso de fabricación.

- Jerarquía: propone un sistema distribuido de los elementos que forman la empresa. Se define que tanto los sistemas como los componentes deben tener la capacidad de adaptarse a los cambios de la red y el proceso. Las distintas acciones se llevan a cabo de manera distribuida y las conexiones entre componentes no se reducen solo a un nivel concreto sino que todos forman parte la red, incluido el propio producto.
- Ciclo de Vida del Producto: se tienen en cuenta las fases del ciclo de vida para la obtención producto y su postventa. Se referencia la fase del producto de la siguiente manera si está en fase de desarrollo, se le considera un tipo y si está en fase de producción y al realizar una actualización cambiará de una fase a otra, implementando una ideología de desarrollo y buscando la mejora continua.
- Capas: se definen los diferentes a tener en cuenta para la fabricación de un producto en capas.

Activos, Integración y Comunicación hacen referencia a componentes físicos de la planta. Información, como su nombre indica abarca todos los datos relativos al producto y fabricación. Esta capa alimenta al resto del modelo. Las últimas capas Función y de Negocio se encargarán de gestionar el proceso de fabricación y la parte de logística y marketing.

IoT y Servicios: Se centra en la digitalización de la empresa, incluyendo todos los dispositivos que forman la red en una capa denominada física. Se prioriza la digitalización de los procesos creando objetos virtuales de cada dispositivo físico para poder gestionar la información correspondiente que es importante en el proceso de fabricación. Dado que el aspecto digital tiene gran importancia en el papel de la Industria 4.0, en este modelo se incluye el concepto de la seguridad y la privacidad de los datos como un aspecto más a la hora de diseñar la red y los componentes que formarán la empresa.

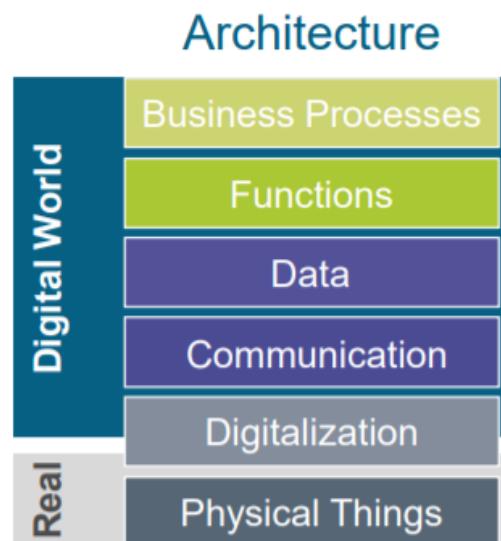


Figura 3.3: Arquitectura RAMI 4.0 Fuente:[17]

3.2. API REST

En esta sección nos dedicaremos a explicar de qué es una API REST y conociendo cómo cada una de las palabras le da significado al conjunto. Nos encontramos realizando un TFG para la mención Tecnologías de Información y Comunicaciones donde la tarea de integrar ha sido algo que se ha recalcado de manera repetida a lo largo de nuestra formación por ello hemos pensado en ella para realizar la integración de componentes de la arquitectura IoT [9].

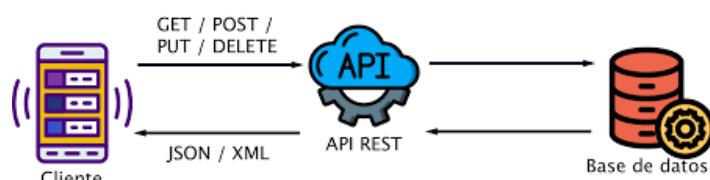


Figura 3.4: Descripción API REST Fuente:[11]

3.2.1. ¿Qué es REST?

El término REST significa REpresentation State Transfer (transferencia de estado representacional) lo que quiere decir que entre dos llamadas cualquiera , el servicio no guarda los datos. Para completar esta pequeña definición debemos añadir que es considerado un estilo de arquitectura software que utiliza para describir cualquier interfaz entre sistemas que utilice el protocolo HTTP para comunicarse entre ellas. Dentro de nuestro propio proyecto encontramos un buen ejemplo que será la autenticación de un usuario utilizando su username y contraseña en el servidor Drupal.

El cliente de un API REST puede ser muy variado desde una aplicación Android o iOS hasta un navegador web, incluso cualquier nuevo dispositivo inteligente como puede ser un Alexa o un frigorífico inteligente.

Las características que ha de tener una arquitectura REST son:

- El cliente y el servidor deben estar acoplados de una manera débil, esto quiere decir, el cliente no debe de conocer detalles de implementación del servidor y el servidor no debe preocuparse de la utilización de los datos.
- Se utilizan los verbos HTTP GET, POST, PUT, DELETE para acceso, creación, actualización y borrado. Estos son los verbos más relevantes pero existen más.
- La interfaz tiene que ser uniforme, es decir, cada recurso del servicio REST debe tener una única dirección URI.
- No existen estados por lo tanto, cada petición que recibe el servidor se trata de manera independiente.
- Las llamadas son cacheables para así evitar pedir varias veces un mismo recurso.

3.2.2. ¿Qué es una API REST?

En el apartado anterior le hemos dado sentido a palabra REST , por lo tanto, en esta sección debemos completar el significado de la palabra API REST. API (Application Programming Interface) que su traducción al castellano sería Interfaz de programación de aplicaciones. En el ámbito de web, se podría decir que una API es un servicio que tiene implementada en el backend para conectar dos aplicaciones.

Se definen los métodos HTTP a los que nosotros aportaremos la funcionalidad que queramos como por ejemplo, autenticar usuarios, obtener información de la base de datos, introducir información en la base de datos. Gracias a los verbos que se definen en la arquitectura REST se nos brinda la posibilidad que desde cualquier dispositivo que pueda usar HTTP pueda consumir una API REST.

Una API REST es un backend capaz de contestar a las llamadas a una serie de URLs en formato JSON y que también es capaz de recibir JSON u otros formatos como www-urlencoded para gestionar la información que le enviamos.

Hemos mencionado JSON (JavaScript Object Notation) pero no sabemos qué es. JSON es un formato de texto sencillo orientado al intercambio de datos. Este formato le ha ganado terreno a la otra alternativa que era XML (Extensible Markup Language) debido a la simplicidad de su formato. Es un formato de texto completamente independiente al lenguaje pero utiliza convenciones que son conocidas de manera clara por la familia de programadores de los grandes lenguajes de programación.

JSON está constituido por dos estructuras que son:

- Un conjunto de pares clave/valor. En varios lenguajes es conocido como tabla hash, objeto, estructura ...
- Lista ordenada de valores que en la mayoría de los lenguajes se implementaran mediante vectores, listas, matrices o secuencias.

3.2.3. Ventajas de API REST

En este apartado damos a conocer las ventajas que nos aporta API REST con respecto a otras alternativas que hacen que nos decantemos por esta. Estas ventajas son las siguientes:

1. La principal ventaja de la API REST es la separación que existe entre cliente y servidor. Esto hace que con un solo servicio pueda ser consumido por varios clientes que no tienen porque ser del mismo tipo.
2. La independencia entre tecnologías y lenguajes lo que hace que si se sigue el contrato "hace que".
3. El bajo consumo de recursos que afirmamos basandonos en que no se mantiene el estado, no requiere memoria, se pueden atender más peticiones y que no requiere reescribir la parte HTML.

CAPÍTULO 4

Herramientas y Metodología de Trabajo

En este capítulo se mostrarán las metodologías que se han utilizado para la gestión del proyecto y para el desarrollo del software. Además, se incluirán los componentes que darán vida al dispositivo de captación a nuestra pequeña estación meteorológica y las herramientas software sobre las que nos apoyaremos para llevar a cabo el proyecto.

4.1. METODOLOGÍA

De poco sirve utilizar una gran cantidad de herramientas y tecnologías de buena calidad si no se cuenta con una metodología de trabajo adecuada. Es necesario aplicar una metodología de trabajo robusta que planifique, guíe y estructure el proceso de desarrollo de un proyecto.

Las metodologías de desarrollo de software tradicionales se basan en un control estricto del proceso mediante una meticulosa definición de roles y actividades, así como una documentación con un gran grado de detalle. Estas metodologías han sido eficientes como muestran sus resultados en proyectos estables, en cuanto a tiempo y recursos, desarrollados por grandes grupos de trabajo.

En los últimos años, han surgido las metodologías ágiles como alternativa a estas metodologías tradicionales. Estas nuevas metodologías comparten los mismos principios que permiten desarrollar software de manera más flexible, aportando la posibilidad de adaptarse a cualquier cambio que surja a lo largo del proyecto. Por esto, están orientadas especialmente a proyectos pequeños en los que exige reducir el tiempo de desarrollo, sin renunciar de ninguna manera a las prácticas fundamentales que garantizan la calidad del producto final [15].

Para trabajar dentro de algunas partes en particular del proyecto denominadas iteración que serán más tarde explicadas se utilizará una metodología de desarrollo secuencial para poder realizar el trabajo de manera organizada debido a que este proyecto es individual.

Para la gestión de este proyecto se ha seleccionado la utilización de una metodología SCRUM debido a la gran cantidad de tecnologías incorporadas y la posibilidad de que esto produzca cambios que abordar de manera ágil y además, no todas las partes del proyecto se abordarán con la misma metodología de desarrollo. Por lo tanto, utilizaremos SCRUM como pegamento para todas estas partes

En el entorno de prácticas se selecciona esta metodología de gestión de proyectos con la intención de que este TFG es introducir al alumno en un entorno de prácticas real.

4.1.1. SCRUM

Scrum es el marco de trabajo que utiliza Pulsia. SCRUM se puede definir de la siguiente manera: un marco de trabajo en el que las personas afrontan diversos problemas que pueden ser adaptados, mientras que entregan productos con el máximo valor posible de forma creativa.

Scrum tiene las siguientes características:

- Es Ágil
- Es Fácil de comprender
- Es Difícil de dominar

Según [13], El marco de trabajo Scrum consiste en equipos y sus roles, eventos, artefactos y reglas asociadas. Cada componente dentro del marco de trabajo sirve a un propósito específico y es esencial para el éxito de Scrum y para su uso.

Las reglas de Scrum relacionan los eventos, roles y artefactos, gobernando las relaciones e interacciones entre ellos.

4.1.2. Equipos Scrum

El equipo Scrum consiste en tres roles diferentes **Product Owner, Development Team y Scrum Master**. Los equipos Scrum son autoorganizados y multifuncionales. Los equipos autoorganizados eligen la mejor manera de llevar a cabo el trabajo sin ser inferidas sus decisiones por personas del exterior. Los equipos autoorganizados eligen la mejor forma de llevar a cabo su trabajo y no son dirigidos por personas externas al equipo. Los equipos multifuncionales tienen todas las competencias necesarias para llevar a cabo el trabajo sin depender de otras personas que no son parte del equipo. El modelo de equipo en Scrum está diseñado para optimizar la flexibilidad, la creatividad y la productividad.

Los Equipos Scrum entregan productos de forma iterativa e incremental, maximizando las oportunidades de obtener retroalimentación.

- **Product Owner** es el responsable de maximizar el valor del producto y el trabajo del Development Team. El cómo se lleva a cabo esto podría variar ampliamente entre distintas organizaciones, Equipos Scrum e individuos.

El Product Manager es la única persona responsable de gestionar la Lista del Producto (Product Backlog). La gestión de la Lista del Producto incluye:

- Expresar de manera clara, los elementos que se muestran en la Lista del Producto.
- Ordenar los elementos en la Lista del Producto para alcanzar los objetivos y misiones de la mejor manera posible.

- Asegurar que la Lista del Producto es visible, transparente y clara para todos y que muestra aquello en lo que el equipo trabajará a continuación
- Asegurar que el Equipo de Desarrollo entiende los elementos de la Lista del Producto al nivel necesario

Para que el Product Owner pueda hacer bien su trabajo, toda la organización debe respetar sus decisiones. Las decisiones del Product Owner se reflejan en el contenido y en la priorización de la Lista del Producto. No se permita que nadie pida al Development Team que trabaje con base en un conjunto diferente de requisitos y el Development Team no debe obedecer las órdenes de cualquier otra persona.

- El **Development Team** consiste en los profesionales que realizan el trabajo de entregar un Incremento de producto “Terminado” que potencialmente se pueda poner en producción al final de cada Sprint. Solo los miembros del Development Team participan en la creación del Incremento.

La organización es la encargada de estructurar y empoderar a los Development Team para que estos organicen y gestionen su propio trabajo. La sinergia resultante optimiza la eficiencia y efectividad del Development Team.

Los integrantes del Development Team tienen las siguientes características:

- Los miembros individuales pueden tener habilidades especializadas y áreas en las que estén más enfocados, pero la responsabilidad recae en equipo como un todo.
- Los equipos son multifuncionales, esto es, como equipo cuentan con todas las habilidades necesarias para crear un Incremento de producto.
- Scrum no reconoce títulos para los miembros de un equipo, todos son desarrolladores, independientemente del trabajo que realice cada persona; no hay excepciones a esta regla.
- Scrum no reconoce subequipos dentro de los propios equipos bajo ninguna condición.

El tamaño óptimo del equipo debe ser lo suficientemente pequeño como para ser ligero y lo suficientemente grande como para completar una cantidad de trabajo significativa.

- El **Scrum Master** es el responsable que la metodología Scrum se lleve a cabo de manera correcta. Ayuda a las personas externas al equipo Scrum a entender que interacciones pueden ser útiles y cuáles no. El Scrum Master ayuda a todos a modificar estas interacciones para maximizar el valor creado por el Equipo Scrum.

El Scrum Master da los siguientes servicios a los componentes del equipo Scrum:

- Product Owner
 - Facilitar eventos Scrum cuando sea necesario.
 - Entender y practicar la agilidad.
 - Darle a conocer la manera de ordenar el Product Backlog para maximizar el valor.
 - Encontrar técnicas para gestionar el Product Backlog de manera mas efectiva.
- Development Team
 - Apoyar al equipo para ser autoorganizado y multifuncional.
 - Eliminar impedimentos para el progreso del Development Team.
 - Ayudar al Equipo de Desarrollo a crear productos de alto valor.
 - Facilitar los eventos de Scrum según se requiera o necesite.
- Organización
 - Liderar y guiar a la organización en la adopción de Scrum.
 - Ayudar a los empleados e interesados a entender y llevar a cabo Scrum.
 - Motivar cambios que incrementen la productividad del Equipo Scrum.
 - Planificar las implementaciones de Scrum en la organización.

4.1.3. Eventos Scrum

Existen eventos predefinidos con el fin de crear regularidad y minimizar la necesidad de reuniones no definidas en Scrum. Todos los eventos son bloques de tiempo.

Una vez que comienza un Sprint, su duración es fija y no puede acortarse o alargarse. Los demás eventos pueden terminar siempre que se alcance el objetivo del evento, asegurando que se emplee una cantidad apropiada de tiempo sin permitir desperdicio en el proceso.

- **Sprint** es el corazón de Scrum, es un bloque de tiempo (time-box) de un mes o menos durante el cual se crea un incremento de producto “Terminado” utilizable y potencialmente desplegable. Es más conveniente si la duración de los Sprints es consistente a lo largo del esfuerzo de desarrollo

Los Sprints estan formados por los componentes que se muestran en las siguientes entradas de la lista.

Los Sprints están limitados a un mes calendario. Cuando el horizonte de un Sprint es demasiado grande la definición de lo que se está construyendo podría cambiar, la complejidad podría incrementarse y el riesgo podría aumentar. Los Sprints habilitan la predictibilidad al asegurar la inspección y adaptación del progreso al menos en cada mes calendario. Los Sprints también limitan el riesgo al costo de un mes calendario.

- En el **Sprint Planning** se planifica el trabajo que se va a realizar durante el Sprint. Este plan se crea mediante el trabajo colaborativo del Equipo Scrum completo. Tiene un máximo de duración de ocho horas para un Sprint de un mes. Para Sprints más cortos el evento es usualmente más corto. El Scrum Master se asegura de que el evento se lleve a cabo y que los asistentes entiendan su propósito.

- **Sprint Goal** es la meta establecida para el Sprint que puede lograrse mediante la implementación del Product Backlog. Proporciona una guía al Development Team acerca de por qué está construyendo el incremento. Se crea durante la Sprint Planning. El objetivo del Sprint brinda al equipo de desarrollo cierta flexibilidad con respecto a la funcionalidad implementada en el Sprint.

Con el fin de satisfacer el objetivo del Sprint se implementa la funcionalidad y la tecnología. Si el trabajo resulta ser diferente de lo que el Equipo de Desarrollo espera, ellos colaboran con el Product Owner para negociar el alcance de la Lista de pendientes del Sprint. Se usa el Daily para evaluar el progreso hacia el Objetivo del Sprint y para evaluar qué tendencia sigue este progreso hacia la finalización del trabajo contenido en la Sprint Backlog. El Daily Scrum optimiza las posibilidades de que el Equipo de Desarrollo cumpla el Sprint Goal.

- **Daily Scrum** es una reunión con un bloque de tiempo de 15 minutos para que el Development Team sincronice sus actividades y cree un plan para las siguientes 24 horas. Esto se lleva a cabo inspeccionando el trabajo avanzado desde el último Daily Scrum.
- **Sprint Review** sirve para inspeccionar el Incremento y adaptar la Lista de Producto si fuese necesario. Durante la Revisión de Sprint, el Equipo Scrum y los interesados colaboran acerca de lo que se hizo durante el Sprint. Basándose en esto y en cualquier cambio a la Lista de Producto durante el Sprint, los asistentes colaboran para determinar las siguientes cosas que podrían hacerse para optimizar el valor. Se trata de una reunión informal, no una reunión de seguimiento.
- **Sprint Retrospective.** Oportunidad para el equipo Scrum de inspeccionarse a sí mismo y de crear un plan de mejoras que sean abordadas durante el siguiente Sprint. Tiene lugar después de la Sprint Review y antes del siguiente Sprint Planning. Se trata de una reunión restringida a un bloque de tiempo de tres horas para Sprints de un mes. Para Sprints más cortos se reserva un tiempo usualmente más corto. El Scrum Master se asegura de que el evento se lleve a cabo y que los asistentes entiendan su propósito. El Scrum Master enseña a todos a mantener el evento dentro del bloque de tiempo fijado.

4.1.4. Artefactos Scrum

Los artefactos Scrum representan trabajo o valor en diversas formas que son útiles para proporcionar transparencia y oportunidades para la inspección y adaptación. Los artefactos definidos por Scrum están diseñados específicamente para maximizar la transparencia de la información clave, necesaria para asegurar que todos tengan el mismo entendimiento del artefacto. Dentro de los artefactos vamos a encontrar los términos de **Product Backlog**, **Sprint Backlog**, **Incremento**.

- **Product Backlog** es una lista ordenada de todo lo que podría ser necesario en el producto y es la única fuente de requisitos para cualquier cambio a realizarse en el producto. El responsable de esta lista será el Product Owner, incluyendo su contenido disponibilidad y ordenación.

La Lista de Producto evoluciona a medida de que el producto y el entorno en el que se usará también lo hacen. La lista en la parte temprana del desarrollo solo contendrá los requisitos conocidos. Por lo tanto, podemos decir que es dinámica. Mientras exista producto, el Product Backlog continuará.

La Lista de Producto enumera todas las características, funcionalidades, requisitos, mejoras y correcciones que constituyen cambios a realizarse sobre el producto para entregas futuras. Los elementos de la Lista de Producto tienen como atributos la descripción, el orden, la estimación y el valor. A medida que va avanzando el producto este va ganando valor.

- **Sprint Backlog** es el conjunto de elementos de la Lista de Producto seleccionados para el Sprint, más un plan para entregar el Incremento de producto y conseguir el Objetivo del Sprint. Sprint Backlog es una predicción hecha por el Development Team acerca de qué funcionalidad formará parte del próximo Incremento y del trabajo necesario para entregar esa funcionalidad

Sprint Backlog es un plan con un nivel de detalle suficiente como para que los cambios en el progreso se puedan entender en el Daily Scrum . Cuando se identifica un nuevo trabajo, el Development Team lo añade al Sprint Backlog. A medida que se va avanzado el trabajo se va actualizando el tiempo estimado. Esta lista pertenece únicamente al Development Team.

- **Incremento** es la suma de todos los elementos de la Lista de Producto completados durante un Sprint y el valor de los incrementos de todos los Sprints anteriores. Al final de un Sprint el nuevo Incremento debe estar "Terminado", lo cual significa que está en condiciones de ser utilizado y que cumple la Definición de "Terminado" del Equipo Scrum.

4.1.5. Metodología de desarrollo en cascada

Esta metodología es considerada la más antigua, también es denominada como secuencial. Se le denomina así por las posiciones que ocupan las diferentes fases que componen el proyecto, colocadas una encima de otra, y siguiendo un flujo de ejecución de arriba hacia abajo, como una cascada. El estilo de esta metodología es la imposibilidad de avanzar a la siguiente fase hasta que no se encuentre realizada de una manera completa debido a que no existe una vuelta atrás. En la siguiente figura 4.1 se muestran las fases del desarrollo en cascada.

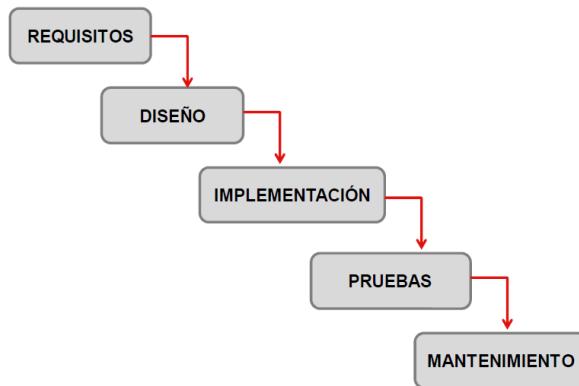


Figura 4.1: Diagrama de Metodología en Cascada Fuente:[6]

■ Análisis de Requisitos

En esta fase se hace un análisis de las necesidades del cliente para determinar las características del software a desarrollar, y se especifica todo lo que debe hacer el sistema sin entrar en detalles técnicos. Es la fase más importante debido a que influirá sobre el resto del desarrollo.

Por lo tanto, esta es la etapa en la que se lleva a cabo una descripción de los requisitos del software, y se acuerda entre el cliente y la entidad a cargo del desarrollo lo que el producto deberá hacer. Disponer de una buena especificación de requisitos permite estimar de forma precisa las necesidades del software antes de su realizar el diseño. El equipo de desarrollo tiene que comprender de manera clara el producto a desarrollar.

- **Diseño** En esta fase de desarrollo se describe la estructura del software y las relaciones que existen entre los componentes que forman esta estructura. Se descompone y organiza el sistema en elementos que puedan realizarse de forma individual, aprovechando el desarrollo en equipo. De aquí nace el documento del diseño software que contiene una descripción de la estructura relacional del sistema y la especificación de cada una de las partes y la manera en la que se combinan estas. Para plasmar el diseño se hace uso de una serie de diagramas como pueden ser
- **Implementación** En esta fase se lleva a cabo de la codificación los requisitos especificados y posteriormente, plasmados en la estructura diseñada en la fase anterior. La programación es el proceso que lleva de la formulación de un problema a nivel de computación a una solución para el problema propuesto.
- **Pruebas** Como su propio nombre indica, una vez se termina la fase de implementación se verifica que todos los componentes del sistema funcionen correctamente y cumplen con los requisitos. El objetivo de esta fase es obtener información sobre la calidad del software desarrollado y sirve para encontrar fallos, aumentar la calidad del producto software y refinar el código.
- **Mantenimiento** Una vez se han desarrollado todas las funcionalidades del software y se ha comprobado que funcionan correctamente, se inicia la fase de instalación y mantenimiento. Se instala la aplicación en el sistema y se comprueba que funcione correctamente en el entorno en que se va a utilizar.

A partir de ahora hay que asegurarse del correcto funcionamiento del software y hay que destinar recursos a mantenerlo. El mantenimiento del software consiste en la modificación del producto después de haber sido entregado al cliente, ya sea para corregir errores o para

mejorar el rendimiento o las características.

4.1.6. Adaptación al proyecto

Para trabajar la realización de este proyecto, hemos optado por utilizar una metodología de desarrollo combinada utilizando la metodología SCRUM para gestionar el proyecto y una metodología de cascada adaptada para trabajar sobre los Sprint más dedicados al desarrollo de software.

Para gestionar el proyecto se utilizará una metodología SCRUM que ayudará a dividir el proyecto en tareas cada vez más cortas y simples. Para ello, se hará uso de la plataforma Redmine se ha adaptado el uso habitual de esta para generar 6 Sprints en mismo tablero para así agilizar el trabajo con la plataforma. Debido a que la intención de este proyecto es involucrar al alumno con la metodología implantada en la empresa y así en caso de que el alumno continúe en la empresa ya conocerá el procedimiento.

El proyecto, se ha realizado de manera individual, por lo tanto, Además, de una pequeña reunión diaria al inicio de la jornada para simular las Daily Scrum. Para supervisar como avanza el proyecto se ha programado una reunión semanal con el tutor de prácticas en empresa para mantener el contacto con el proyecto y conocer la evolución del alumno dentro del proyecto

Los Sprints se han configurado de la siguiente manera:

- Configuración de Base de Datos
 - Creación Esqueleto Tablas.
Se ha cumplido creando la estructura de tablas de base de datos en la instalación de Drupal basándonos los datos que vamos a recibir de los sensores.
- Desarrollo de Módulo Dedicado
 - Prueba de configuración de routing.
Se ha logrado este hito creando el módulo e implementando la recepción y almacenado de valores ambientales.
 - Comprobación de datos con AEMET.
Se consume el servicio elegido para obtener las predicciones de AEMET, se implementa la comprobación y registro de alarmas.
- Visualización de Datos
 - Creación de Bocetos.
Se crean los bocetos de la app para guiar el trabajo a lo largo del sprint.
 - Creación Login.
Se crean implementa la funcionalidad de acceso a la instalación de Drupal desde la app.
 - Creación de Módulo Drupal de muestra de datos.
Se implementa el recurso para enviar información hacia el dispositivo móvil.
 - Desarrollo de Gráficas.
Se implementan las peticiones para recibir información y mostrarla en la aplicación móvil.
- Configuración Hardware
 - Instalación de Placa Arduino.
Se monta el dispositivo para recolección de valores y se registra en la plataforma del

proveedor.

- Desarrollo de recolección de datos.

Desarrollamos el software para recolectar valores leídos por los sensores , estandarizar y trasmisir.

- Comunicación Hardware-Plataforma

- Configuración Device en SigFox.

Se configura en la plataforma del proveedor el reenvío de los valores emitidos por la placa Arduino. El reenvío se realizará hacia el recurso del API REST para inserción de información en la plataforma.

- Documentación del Proyecto

- Manual de Montaje.

Se redacta un manual para plasmar el conocimiento sobre el montaje del dispositivo Siabox para futuros proyectos.

- Manual de Despliegue.

Se redacta un manual para estandarizar el despliegue del producto en las zonas de instalación

- Estructura del Documento - Memoria TFG.

Se realiza la memoria del Trabajo Fin de Grado asociado a este proyecto.

Dentro de los Sprints más dedicados a desarrollo que son *Desarrollo de Módulo Dedicado*, *Visualización de Datos* y *Configuración Hardware* se apoyara en una metodología de desarrollo de software secuencial adaptada debido a que algunas de las fases son imposibles de llegar mostrar en esta memoria. Por ello, esta metodología tradicional adaptada se basara en las tres primeras fases **Análisis, Diseño e Implementación**.

4.2. PLANIFICACIÓN Y COSTES

En esta sección se van a incluir toda la información para justificar el trabajo realizado durante este proyecto. Para justificar el trabajo se utilizará un diagrama de Gantt y el coste de los componentes para el montaje del dispositivo Siabox.

Previo a la realización del proyecto se ha realizado un periodo de formación en algunas de las tecnologías con las que se ha desarrollado el proyecto como han sido Drupal ¹ y Arduino ² que se ha realizado desde la primera semana de Marzo 2021 hasta la segunda semana de Abril 2021 para continuar con el desarrollo del proyecto.

4.2.1. Costes Suscripción SigFox

Adjuntamos una tabla con los precios de suscripción que se pueden encontrar en el portal web del operador SigFox. ³.

¹<https://www.forcontu.com/>

²<https://programarfacil.com/blog/arduino-blog/arduino-mkrfox1200-sigfox-lpw/>

³<https://buy.sigfox.com/buy/offers/ES>

Articulo	Precio
Con la compra de un dispositivo	2 AÑOS GRATIS
Tárrifa Discovery 1 Año	19.52€/Dispositivo
Tárrifa Discovery con geolocalización 1 Año	22.45€/Dispositivo
Tárrifa Enterprise	Contactar con operador Sigfox

Tabla 4.1: Costes de Suscripción SigFox

4.2.2. Costes Dispositivo Siabox

Para justificar costes hemos realizado un presupuesto del montaje y despliegue del dispositivo Siabox para captación de datos.

Articulo	Precio
Placa Arduino MKRFOX1200	51€
Placa PCB	2,18€
Sensor UVM30A	3€
Sensor DHT11	5,38€
Sensor YL69/YL39	7,99€
Sensor TSL2591	2,99€
Caja Estanca exterior IP65	7,99€
Lámina de plástico cristalizado	0,99
Antena GSM compatible con SigFox	7,99€
Fuente de alimentación USB Completa	5,49€
Suscripción	2 AÑOS GRATIS
Total	95,00€

Tabla 4.2: Costes de Dispositivo Siabox

4.2.3. Diagrama de Gantt

En este apartado se incorpora un diagrama de *Gantt* para representar gráficamente el tiempo estimado para las diferentes actividades del proyecto y el tiempo dedicado. Que serán desarrolladas en el capítulo de [5](#). El diagrama lo adjuntamos en el ANEXO D.

4.3. HERRAMIENTAS SOFTWARE UTILIZADAS

En esta sección se mostrarán todas las herramientas software relevantes utilizadas para la consecución del proyecto. Podemos encontrar desde entornos de desarrollo integrado a frameworks de creación de aplicaciones web.

4.3.1. Redmine

La herramienta Redmine⁴ es utilizada para gestión de proyectos, gracias a la diversidad de funcionalidades permite a los usuarios de diferentes proyectos realizar un seguimientos y organización de los mismos. Es muy útil, debido a que tiene la posibilidad optimizar su funcionamiento incorporando funcionalidades.

Destacan las siguientes funcionalidades:

⁴<https://es.wikipedia.org/wiki/Redmine>

- Sistema de seguimiento de incidencias con seguimiento de errores.
- Calendario de Actividades.
- Wiki.
- Control de flujo de trabajo.
- Foro.

Esta herramienta se encuentra disponible bajo Licencia Pública General de GNU debido a que es software libre y de código abierto.

4.3.2. SigFox Backend

SigFox Backend es un servicio web perteneciente al proveedor SigFox utilizada para la gestión de dispositivos y configuración de integración de datos. Además, cuenta con una API para automatizar las acciones que pueden ser realizadas.

Dentro de este servicio, se pueden diferenciar 4 grandes apartados:

- Dispositivo
- Tipo de Dispositivo
- Grupo
- Usuario

Desde la sección **Dispositivo**, se podrá gestionar los dispositivos registrados en la plataforma e incorporar nuevos a esta.

En **Tipo de Dispositivo** se tendrá la posibilidad de crear configuración de CallBacks para reenviar los mensajes utilizando la API de la plataforma para enviar información hacia URL externas con el formato que se desee. Además, desde esta opción se puede gestionar la asignación de suscripciones a los dispositivos.

En la sección **Grupo** se podrán crear grupos de trabajo donde agrupar diferentes usuarios y poder trabajar sobre los mismos dispositivos y tipos de estos. Desde esta sección se podrán visualizar los contratos que pertenecen al grupo.

Para finalizar, en la sección **Usuario** se podrá revisar el registro de acciones del usuario que esté en sesión.

4.3.3. Postman

Herramienta que permite realizar solicitudes HTTP a cualquier endpoint. Postman es muy útil tanto a nivel de programación como a nivel de pruebas debido a que nos permite verificar el funcionamiento del desarrollo.

El usuario de Postman ⁵ en la mayoría de los casos será un desarrollador que verifica el funcio-

⁵<https://www.postman.com/>

namiento de un recurso web para poder desarrollar sobre el o en su defecto será un operador que realiza tareas monitorización de esa API en concreto. Uno de los puntos fuertes de Postman es la posibilidad de generar documentación de manera sencilla, así como ejemplos y fragmentos de código que muestran el funcionamiento de una API de una manera clara y sencilla.

Postman nos brinda la posibilidad de agrupar y guardar solicitudes en un conjunto de solicitudes denominado colecciones, es decir, directorios en diferentes niveles que organizarán nuestros conjuntos de solicitudes HTTP y HTTPS. En nuestro proyecto, Postman es usado para realizar peticiones de forma local y remota a diferentes módulos personalizados de Drupal 8 para lectura y escritura de datos. Además, se ha utilizado a modo de prueba para revisar el esquema de datos de los JSON enviados por parte de la API REST de AEMET.

4.3.4. SWAGGER

Swagger [20] es un estándar compuesto por una serie de reglas, especificaciones y herramientas que nos permiten diseñar, construir, documentar y utilizar de manera adecuada nuestras APIs. De este modo, realizar documentación sobre estas es mucho más útil para futuros desarrolladores. Swagger ayuda a crear documentación entendible para todos. Además, incluye la herramienta Swagger UI que ayuda a entender el desarrollo y utilización de estas APIs. Con Swagger la documentación puede utilizarse directamente para automatizar procesos dependientes de APIs.

Swagger es la versión más extendida de este tipo de frameworks pero sin su herramienta Swagger UI no ocuparía la cima. Debido a que nos ofrece la documentación útil de una manera que sea navegable entre ella y este organizada para su uso fácil. Utilizando el Swagger UI para exponer la documentación de nuestra API, podemos ahorrarnos mucho tiempo. Con el Swagger UI podremos organizar nuestros métodos e incluso poner los ejemplos que necesitemos.

4.3.5. Drupal 8

Drupal es un Sistema de Gestión de Contenidos (CMS) que se distribuye como software libre bajo licencia GNU GPL (General Public License) versión 2 o superior. Este CMS nos aporta la posibilidad de modificar y distribuir libremente, siempre que haga bajo la misma licencia.

El software está desarrollado con el lenguaje de programación PHP y permite la utilización de base de datos MySQL o MariaDB, aunque permite su instalación entre motores como PostgreSQL o MongoDB. Para el aspecto visual hace uso de plantillas y hojas de estilo CSS⁶ (Cascading Style Sheets) que hacen posible la construcción de cualquier tipo de sitio web. Uno de los puntos fuertes de esta herramienta es lo extendido que está la integración en los hosting de un instalador de Drupal.

Como ventajas en la utilización de este software debemos destacar las siguientes:

- **Escalabilidad:** Actualmente, es compatible con los sitios más activos a nivel mundial. Drupal es capaz de manejar los picos de tráfico regulares o un gran volumen de visitantes.

⁶CSS en español Hojas de estilo en cascada es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado.

- **Modular:** Funciona con módulos creados por la comunidad y que se agregan a medida de las necesidades existentes. Por lo tanto, existen una infinidad de posibilidades para este CMS.
- **Adaptabilidad:** Drupal permite adaptarse en todo momento de una manera rápida a los continuos cambios del mercado donde proliferan las formas de gestionar de manera mas productiva y rentables para alcanzar el mayor rendimiento en el menor tiempo posible.
- **Amplia comunidad de desarrolladores:** Existe una amplia comunidad de desarrolladores que gracias a su trabajo constante tenemos la posibilidad de disponer de una gran cantidad de funcionalidades para incorporar a nuestros sitios web o servidor como nosotros lo hemos utilizado en nuestro proyecto.
- **Capacidad de Integración:** No necesita la instalación de extensiones o módulos para conseguir crear cualquier tipo de web. Al igual que por ejemplo WordPress está orientado inicialmente a blogs (aunque se pueda adaptar), Drupal se puede adaptar para hacer prácticamente cualquier cosa con él (tienda, blog, foro de discusión...)

A nivel particular, dentro de la empresa la utilización del CMS Drupal está muy extendido debido a que una de las principales actividades es el desarrollo de portales web.

4.3.6. Arduino IDE

Entorno de desarrollo integrado perteneciente al fabricante Arduino que se utiliza para escribir y cargar programas en placas compatibles con Arduino, pero también, con la ayuda de núcleos de terceros, se puede usar con placas de desarrollo de otros proveedores. Desde esta herramienta se nos brinda la posibilidad de añadir librerías para aumentar la funcionalidad de nuestros equipos como en nuestro caso realizaremos con los sensores ambientales. Además, será el entorno principal para el compilado, cargado de información y monitoreo de nuestros dispositivos.

Esta herramienta se puede encontrar en la Microsoft Store de manera totalmente gratuita.

4.3.7. Angular

Angular es un framework de código abierto para desarrollo de aplicaciones web móviles y de escritorio. Actualmente, su mantenimiento esta acargo de Google. El objetivo de Angular es facilitar la creación y mantenimiento de aplicaciones web en una página SPA (Single Page Application). Para conseguirlo Angular ofrece un sistema de routing entre sus herramientas. Es altamente configurable y muy potente, qué se puede configurar fácilmente con una estrategia de *Lazy loading*. Así se obtienen aplicaciones muy optimizadas.

Angular separa completamente el front-end y el back-end en la aplicación, esto hace que se evite la escritura de código de manera repetitiva y que existan orden en su desarrollo. Esto, es gracias a su modelo MVC (Modelo-Vista-Controlador) que garantiza un desarrollo rápido al mismo tiempo que permite modificar y actualizar el contenido.

Las ventajas a reseñar de este framework, son modularidad y escalabilidad que aportan a los proyectos. Ademas, se basa en un estándar de componentes web y un conjunto de APIs que permite

la creación nuevos elementos HTML personalizados que pueden reutilizarse.

Las páginas web SPA se caracterizan por tener un tiempo de carga elevado en la apertura, debido a que se realiza una carga completa del sitio web. La navegación posterior, será casi instantánea. El servidor, se encargar de enviar rutas y Angular realiza cambios en la vista durante la navegación, para presentar la apariencia de un sitio web normal de una manera dinámica.

El lenguaje de programación principal de Angular es Typescript, por lo que toda la sintaxis y procesamiento en el código es el mismo. Esto hace que relación y consistencia de la información sea muy alta.

4.3.8. Ionic

Ionic es un framework⁷ de código libre destinado a la creación de interfaces de usuario para aplicaciones móviles y de escritorio utilizando tecnologías web (HTML,CSS,Javascript) con la integración de frameworks populares como Angular, React y Vue.

Este framework nos permite desarrollar aplicaciones para iOS nativo, Android y web utilizando un único código base. Esto, es posible gracias a la utilización de **Capacitor** que se mostrará más en profundidad en el siguiente apartado.

Listamos de manera breve las principales características que hacen Ionic tan usado:

- Está desarrollado bajo una base confiable de tecnologías web y API estandarizadas.
- Ágil y con un alto rendimiento.
- Limpieza en el diseño, atractivo y sencillo de entender para desarrolladores habituales en otras plataformas.
- Sus principios de diseño se basan en un diseño web responsive lo que nos facilita el trabajo entre diferentes tamaños de pantalla y píxeles de densidad.
- Muy recomendable para desarrollar Apps robustas y para cualquier dispositivo o plataforma.

4.3.9. Capacitor

Capacitor es un puente que nos permite lanzar aplicaciones multiplataforma nativas en tiempo real, lo que facilita la construcción de aplicaciones móviles modernas. Esto representa la evolución de las aplicaciones híbridas. Crea aplicaciones nativas de la web, proporcionando un enfoque de contenedor nativo moderno para los equipos que quieren construir la web primero sin sacrificar el acceso completo a los SDKs nativos cuando lo necesitan.

Capacitor nos aporta con respecto a sus predecesores las siguientes ventajas:

- Compatibilidad con PWA (Aplicaciones Web Progresivas).
- Facilidad para integrar controles nativos.

⁷Un framework o entorno de trabajo, puede entenderse como un conjunto estandarizado de conceptos, criterios y herramientas software utilizadas el desarrollo de software dentro de una problemática particular.

- Instalación localizada en proyectos individuales.
- Soporte para plugins
- Disponibilidad de funcionalidad nativa de manera inmediata.

4.3.10. Base de Datos

Ante la variedad de posibilidades que Drupal 8 nos ofrece para la gestión de almacenamiento de información hemos seleccionado **MySQL**. Por lo tanto, nuestra instalación de Drupal contará con una base de datos relacional (SQL).

Para la gestión de la persistencia del sistema no haremos consultas directamente sobre la base de datos. Utilizaremos un librerie que incorpora Drupal para la gestión de contenidos almacenados en la base de datos. Por lo tanto, nuestro tratamiento de la base de datos se realizará de una manera un poco peculiar.

Para la creación de la base de datos hemos utilizado los tipos de contenido de Drupal como tabla y las referencias a entidad como relación entre tablas. Por lo tanto, así trabajaremos con las librerías que Drupal nos ofrece para el tratamiento de la persistencia pero sin perder la esencia de SQL. Esto hace que las consultas que realicemos sobre la persistencia sean mucho más sencillas.

4.3.11. Control de Versiones

Para el mantenimiento y control de los cambios realizados durante el desarrollo del proyecto software, se ha utilizado **Git** como sistema de control de versiones y **Github** como servicio web para almacenamiento en repositorio remoto.

4.3.12. Documentación

- **Draw.io** Esta herramienta web utilizada para crear y modificar diagramas. Se ha utilizado para la creación de algunas de las figuras incluidas en la memoria como diagramas UML de clases , secuencia y casos de uso.
- **Balsamiq Mockup** Herramienta utilizada para creación de bocetos y prototipos de la interfaz de usuario de la aplicación web.
- **LATEX** Herramienta orientada a la creación de documentos escritos, utilizada para desarrollar la memoria de este TFG.
- **GanttProject** Herramienta orientada a la creación de diagramas de Gantt para exponer el tiempo de dedicación previsto para diferentes tareas o actividades a lo largo de un tiempo total determinado.

4.4. DISPOSITIVOS HARDWARE UTILIZADOS

Las herramientas software mencionadas en el apartado anterior, han tomado valor gracias a una serie de dispositivos hardware que le han dado vida a nuestro dispositivo Siabox.

4.4.1. Arduino MKRFOX 1200

De la colaboración entre Arduino y SigFox, nace la Arduino MKRFOX1200. Esta placa es perfecta para proyectos de IoT donde hay movilidad debido a que utiliza una red celular y de bajo consumo. Al comprar este dispositivo, se obtiene tanto una suscripción gratuita durante 2 años al plan Discovery⁸ de SigFox. Destaca debido a que integra el microcontrolador **SAMD21** y el módulo de SigFox **ATA8500** ambos fabricados por ATMEL.



Figura 4.2: Imagen de placa Arduino MKRFOX1200 Fuente:[5]

Este placa distribuye las funcionalidades entre los elementos que la forman. El microcontrolador se encarga de gestionar la interfaz de los GPIO y donde se carga el programa y el módulo de SigFox se encargará de la conexión.

⁸<https://buy.sigfox.com/buy/offers/ES>

Característica	Información
Microcontrolador	SAMD21 Cortex-M0+ 32bit low power ARM MCU
Fuente de alimentación (USB/VIN)	5V
Baterías admitidas	2x AA o AAA
Tensión de funcionamiento del circuito	3.3
Pines Digitales I/O	8
Pines PWM	12 (0, 1, 2, 3, 4, 5, 6, 7, 8, 10, A3 - o 18 -, A4 -o 19)
UART	1
SPI	1
I2C	1
Pines Analógicos de Entrada	7 (ADC 8/10/12 bit)
Pines Analógicos de Salida	1 (DAC 10 bit)
Interrupciones Externas	8 (0, 1, 4, 5, 6, 7, 8, A1 -o 16-, A2 - o 17)
Corriente por Pin I/O	7mA
Memoria Flash	256KB
SRAM	32KB
EEPROM	no
Velocidad de Reloj	no
Dispositivo USB de alta velocidad y host integrado	
LEDBUILTIN	6
Potencia de Antena	2dB
Frecuencia	868MHz
Región	EU
Longitud	67.64mm
Anchura	25mm
Peso	32 gr.

Tabla 4.3: Características Técnicas Arduino MKRFOX1200

4.4.2. Sensor YL-69 y módulo YL-39

Este sensor esta formado por dos componentes, la sonda **YL-69**⁹ con dos terminales separados adecuadamente y un módulo **YL-39** que está formado por un comparador estable, un led de encendido y otro de activación de salida digital.

Este sensor tiene la capacidad de medir la humedad del suelo. Esto es posible aplicando una pequeña tensión entre los 'dientes' de la sonda hace pasar corriente que depende básicamente de la resistencia que genere el suelo, esto depende en gran medida de la humedad. Por lo tanto, cuanto menor sea la resistencia generada, mayor será el porcentaje de humedad del sustrato.

⁹<https://create.arduino.cc/projecthub/nekhbet/using-the-yl-39-yl-69-soil-humidity-sensor-with-arduino-968268>

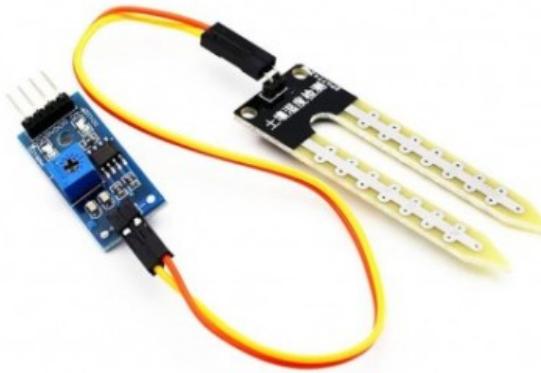


Figura 4.3: Módulo YL-39 (Derecha) y Sensor de humedad en tierra YL-69 (Izquierda) Fuente:[18]

Especificación	Información
Voltaje de Entrada	3.3 - 5 VCD
Voltaje de salida	0 - 4.2V
Corriente	35mA
VCC	Tensión de alimentación
GND	Tierra
A0	Salida analógica que entrega una tensión proporcional a la humedad.
D0	Salida digital. Permite ajustar cuándo el nivel lógico en esta salida pasa de bajo a alto mediante el potenciómetro
Dimensiones YL-39	30 x 16 mm
Dimensiones YL-69	60 x 30 mm
Peso	7gr.

Tabla 4.4: Especificación YL-69 y YL-39

4.4.3. Sensor TSL2591

Sensor de luminosidad fabricado por Adafruit. Incorpora un ADC, lo que significa que este sensor se puede usar con cualquier microcontrolador, incluso si no tiene entradas analógicas. Tiene una alta precisión permitiendo obtener cálculos de Lux¹⁰ con exactitud, además puede ser configurado para diferentes ganancias para detectar rangos de entre 188 μ Lux hasta 88000 Lux. Además, incorpora un diodo infrarrojo y un full spectrum que permite obtener mediciones separadas de luz infrarroja y luz visible por el ojo humano. Una de las mayores ventajas de la utilización de este sensor es la incorporación de un ADC, lo que significa que este sensor se puede usar con cualquier microcontrolador, incluso si el microcontrolador carece de entradas analógicas.

Este sensor tiene un consumo de corriente bajo, por lo que es ideal para sistemas de registro de datos de baja potencia. Aproximadamente 0.4 mA cuando se detecta activamente y menos de 5 μ A cuando está en modo de apagado.

¹⁰Se usa en la fotometría como medida de la iluminancia, tomando en cuenta las diferentes longitudes de onda según la función de luminosidad, un modelo estándar de la sensibilidad del ojo humano a la luz.

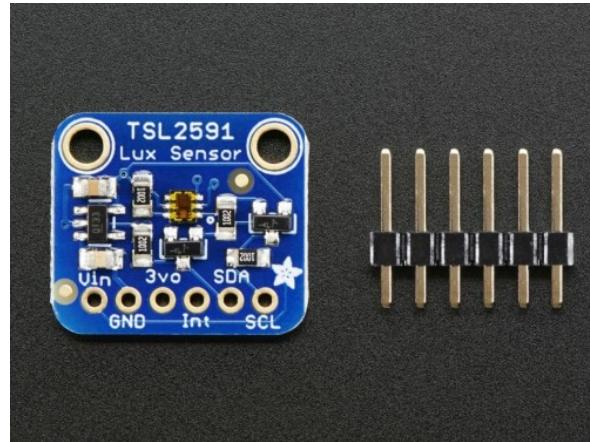


Figura 4.4: Imagen del Sensor Lumínico TSL2591 Fuente:[19]

Especificación	Información
Voltaje de Funcionamiento	3.3 - 5V
Rango dinámico	Extremadamente amplio 1 a 600,000,000 Counts
Rango de lux	Sensibilidad de 188 μ Lux, hasta medidas de entrada de 88,000 Lux.
Interfaz	I2C
I2C Dirección	7 bits 0x29 (fijo)
Dimensiones	19 mm x 16 mm x 1 mm
Peso	11gr

Tabla 4.5: Especificación TSL2591

4.4.4. Sensor DHT11

El sensor DHT11 dedicado para medir temperatura y humedad ambiente ofrece una alta fiabilidad y estabilidad debido a su señal digital calibrada. Podemos encontrarlo solo el sensor o insertado en una PCB. En el proyecto se monta en una PCB.

La versión con PCB aporta una resistencia pull-up de 5 k ω y un LED que nos avisa de su funcionamiento.

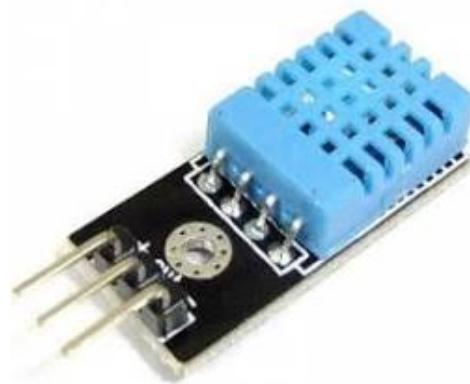


Figura 4.5: Imagen del Sensor de Temperatura y Humedad ambiente DH11 Fuente:[16]

El funcionamiento de este sensor se basa en elemento sensor de humedad capacitivo y un termistor

para detectar la temperatura. El condensador sensor de humedad tiene dos electrodos con un sustrato que retiene la humedad como un dieléctrico entre ellos. El cambio en el valor de la capacitancia se produce con el cambio en los niveles de humedad. El IC mide, procesa estos valores de resistencia cambiados y los cambia en forma digital.

Por otro lado, el funcionamiento del sensor para detectar la temperatura utiliza un termistor de coeficiente de temperatura negativa, que provoca una disminución de su valor de resistencia con el aumento de la temperatura.

Especificación	Información
Alimentación	3,5 - 5 V
Consumo	2,5mA
Señal de salida	Digital
Temperatura	
Rango	0°C a 50°C
Precisión	25°C ± 2°C
Resolución	1°C (8-bit)
Humedad	
Rango	20 % RH a 90 % RH
Precisión	0°C y 50°C ± 5 % RH
Resolución	1 % RH

Tabla 4.6: Especificaciones Sensor Humedad y Temperatura DHT11

4.4.5. Sensor UVM-30A

Los sensores ultravioleta son utilizados para detectar el índice de intensidad ultravioleta, donde la radiación electromagnética tiene longitudes de onda más cortas que las radiaciones visibles al ojo humano. Por lo tanto, el sensor es capaz de detectar la longitud de onda. Este sensor en particular cuenta con una amplia gama espectral.

Lo primero que debemos saber es el funcionamiento del Índice UV ¹¹. El índice tiene un valor mínimo teórico de 0 y no tiene un valor máximo. Este índice estándar permite emitir predicciones de UVI comparables en todo el mundo. Los colores utilizados son el verde para UVI bajo (entre 0 y 2), el amarillo para UVI moderado (entre 3 y 5), el naranja con un riesgo alto (entre 6 y 7), el rojo para UVI muy alto (entre 8 y 10) y el morado para UVI extremo (superior a 11).



Figura 4.6: Indice UV. Extraido de [12]

¹¹Explicar el Indice UV

Este módulo se basa en el sensor UVM-30A, que tiene una amplia gama espectral de 200 hasta 370nm. La señal eléctrica de salida del módulo, es de tipo analógica, que varía respecto a la intensidad de los rayos UV.

Especificación	Información
Voltaje de funcionamiento	3 a 5 VDC
Voltaje de salida	0 a 1 VDC
Longitud de onda UV	200 a 370nm
Precisión	1UV
Corriente estándar	0.06mA
Temperatura de trabajo	-20°C a 85°C

Tabla 4.7: Especificaciones Sensor UVM-30A



Figura 4.7: Imagen del Sensor UVM-30A Fuente:[3]

CAPÍTULO 5

Resultados

En esta sección se describe en detalle los procesos que se han llevado a cabo para la consecución del proyecto. Desde la configuración de la base datos de nuestro Drupal pasando por la creación de módulos personalizados para la inserción de datos en el almacenamiento y posterior visualización de estos datos por parte de la App. Además, el montaje y configuración del dispositivo mediante este proyecto sería inviable (SiaBox) que nos proveerá de información al servidor así como la plataforma del proveedor de LPWAN.

Los Sprint se abordarán de la siguiente manera, una planificación del Sprint y la ejecución de este.

5.1. SPRINT 1.CONFIGURACIÓN DE BASE DE DATOS

Este Sprint se centra en la creación y configuración de la base de datos del sistema. Para ello se utilizará el sistema de referencias y entidades que Drupal ofrece para la creación de contenido.

5.1.1. Planificación Sprint 1



Figura 5.1: Planificación Sprint 1. Configuración Base de Datos

Como resultado de la reunión realizada para la planificación del "Sprint 1. Configuración de Base de Datos" se ha determinado que este Sprint cuente con un uno item con una estimación de 16 horas.

5.1.2. Ejecución Sprint 1

Para la creación de la estructura de la base de datos se realiza un análisis de los requisitos de almacenamiento definidos con anterioridad al inicio del proyecto con el fin de diseñar de una manera guiada las tablas de la base de datos y la elección del tipo que será asignada a los campos de estas. En la siguiente figura 5.2 se presentan los requisitos de almacenamiento sobre los que trabajaremos en este Sprint.

Req1 Dispositivo	Req3 Usuario	Req5 Alarma Sensor
nid:int Cultivo:node idSiabox::String Usuario:user	id:int nombre:string contraseña:string correo electronico:string	nid:int Siabox:Node Valor Registrado:Node Humedad:float Error Humedad:boolean Ultravioleta:float Error Ultravioleta:boolean Temperatura:float Error Temperatura:boolean Comprobada:boolean
Req2 Cultivo	Req4 Valor Registrado	
nid:int idCultivo:string Localización AEMET:string Nombre:string	nid:int Siabox::String Humedad Ambiente:float Humedad Tierra:float Luminosidad:float Ultravioleta:float Temperatura:float	

Figura 5.2: Requisitos de Almacenamiento

Para la creación de cada una de las tablas necesarias para el funcionamiento correcto de nuestra arquitectura. Hemos utilizado el sistema de creación de tipo de contenido que el CMS Drupal nos ofrece, como ya explicamos en el apartado [4.3.10](#).

Este es el resultado de las tablas necesarias para el trabajo de nuestra Arquitectura IoT, hay que especificar que la tabla de usuarios es creada de manera automática por el propio Drupal. Al igual, que los identificadores de cada uno de los Tipos de Contenido. Posteriormente, se explicará de manera más detallada la función y estructura de cada una de las tablas creadas.

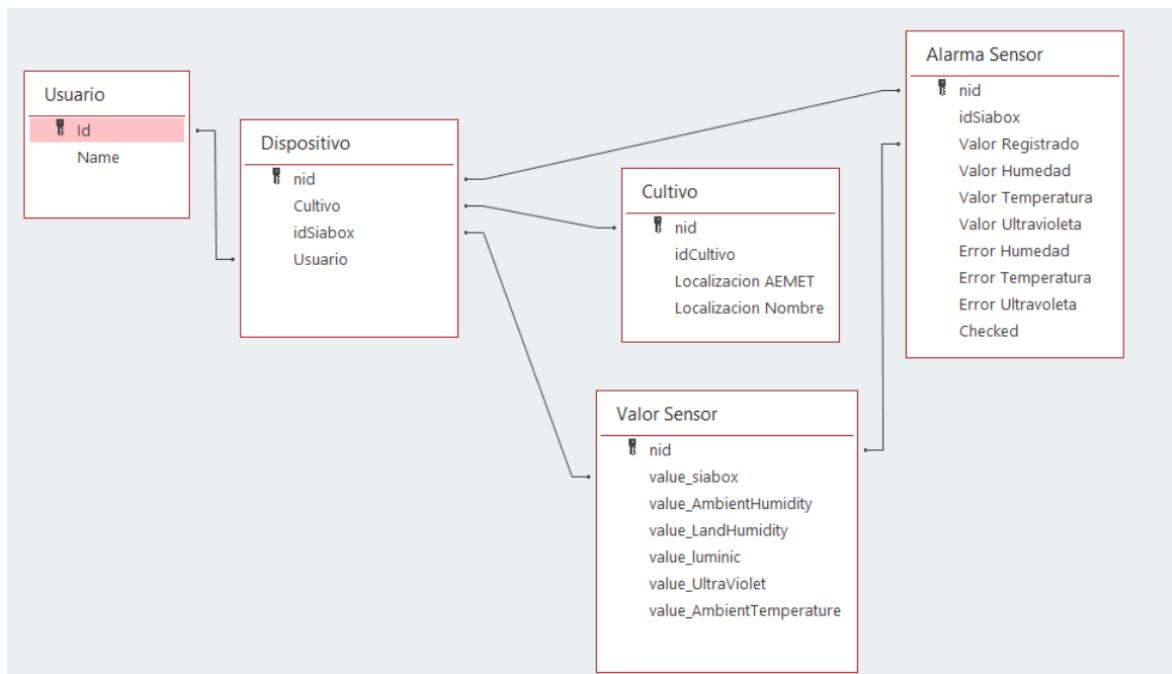


Figura 5.3: Diagrama Base de Datos

Dispositivo

En esta tabla recogemos la información de cada uno de los dispositivos SiaBox, al usuario al que se encuentra ligado y cultivo donde se encuentra instalado el dispositivo. Por ello utilizamos los valores

de **Referencia de entidad**. También, almacenaremos el código que nuestro dispositivo tendrá en la plataforma de SigFox Backend.

ETIQUETA	NOMBRE DE SISTEMA	TIPO DE CAMPO	OPERACIONES
Cultivo	field_dispositivo_cultivo	Referencia de entidad	<button>Editar</button>
idsiabox	field_dispositivo_idsiabox	Texto (sin formato)	<button>Editar</button>
Usuario	field_dispositivo_usuario	Referencia de entidad	<button>Editar</button>

Figura 5.4: Estructura de tipo de contenido Dispositivo

Cultivo

En esta tabla se va a almacenar la información relativa al cultivo donde se encuentra emplazado el dispositivo de captación de parámetros ambientales. Además, incluiremos un alias de cultivo y un código de localización de zona proporcionado por la AEMET para la posterior comprobación de valores con respecto a las predicciones.

ETIQUETA	NOMBRE DE SISTEMA	TIPO DE CAMPO	OPERACIONES
idCultivo	field_idcultivo	Texto (sin formato)	<button>Editar</button>
Localización AEMET	field_localizacion_aemet	Número (entero)	<button>Editar</button>
Localización Nombre	field_cultivo_localizacion_nombr	Texto (sin formato)	<button>Editar</button>

Figura 5.5: Estructura de tipo de contenido Cultivo

Valor Sensor

En esta tabla se va a almacenar la información que nuestro Dispositivo nos reenvía mediante el Backend de SigFox.

ETIQUETA	NOMBRE DE SISTEMA	TIPO DE CAMPO	OPERACIONES
value_AmbientHumidity	field_value_ambient_humidity	Número (flotante)	<button>Editar</button>
value_AmbientTemperature	field_value_ambient_temp	Número (flotante)	<button>Editar</button>
value_LandHumidity	field_value_land_humidity	Número (flotante)	<button>Editar</button>
value_luminic	field_value_luminic	Número (flotante)	<button>Editar</button>
value_siabox	field_valor_idsiabox	Referencia de entidad	<button>Editar</button>
value_UltraViolet	field_value_ultraviolet	Número (flotante)	<button>Editar</button>

Figura 5.6: Estructura de tipo de contenido Valor Sensor

Alarma Sensor

Este contenido se generará y almacenará los valores siempre y cuando no se cumplan las condiciones entre los valores leídos y los valores predecidos por AEMET. Para el control de visualización de alarmas, añadimos un campo checked que nos ayudara a la hora de mostrar la información a los usuarios.

ETIQUETA	NOMBRE DE SISTEMA	TIPO DE CAMPO	OPERACIONES
checked	field_alarma_sensor_checked	Booleano	<button>Editar</button>
Error Humedad	field_alarma_error_humedad	Booleano	<button>Editar</button>
Error Temperatura	field_alarma_error_temperatura	Booleano	<button>Editar</button>
Error Ultravioleta	field_alarma_error_ultravioleta	Booleano	<button>Editar</button>
idsiabox	field_alarma_idsiabox	Referencia de entidad	<button>Editar</button>
Valor Humedad	field_alarma_valor_humedad	Número (flotante)	<button>Editar</button>
Valor Registrado	field_alarma_valor_sensor	Referencia de entidad	<button>Editar</button>
Valor Temperatura	field_alarma_valor_temperatura	Número (flotante)	<button>Editar</button>
Valor Ultravioleta	field_alarma_valor_ultravioleta	Número (flotante)	<button>Editar</button>

Figura 5.7: Estructura de tipo de contenido Alarma Sensor

5.1.3. Conclusión Sprint 1

Se realiza una reunión con el product owner una vez finalizado el periodo del Sprint para revisar si el sprint backlog se había realizado de forma correcta. El product owner revisa la configuración con respecto a los requisitos de almacenamiento y da su visto bueno.

El sprint list se han realizado de manera satisfactoria y no hay que realizar modificaciones. El tiempo dedicado para este sprint han sido 16 horas. Por lo tanto, se realiza un incremento sobre el producto del proyecto.



Figura 5.8: Conclusión Sprint 1. Configuración Base de Datos

5.2. SPRINT 2.DESARROLLO DE MÓDULO DRUPAL DEDICADO

Este Sprint se centrará en la creación de un módulo que será utilizado como parte de la API REST que utilizaremos para comunicación entre elementos de la arquitectura. Este recurso se basará en la obtención de información enviada desde el SigFox Backend para su posterior almacenado en la base de datos de nuestro Drupal 8 y realizar una comprobación de estos datos recibidos con las predicciones realizadas por la AEMET que obtendremos mediante la consumición del servicio que nos proporciona su API REST.

5.2.1. Planificación Sprint 2

Se realiza una reunión con el product owner para la planificación del Sprint 2 Desarrollo de Módulo Dedicado, que como resultado de esta reunión se ha determinado que la realización de este Sprint se dividirá en dos productos del backlog.

Cabe destacar que para la realización de las tareas de este Sprint se aplicará una metodología de modelo en cascada centrándonos en las primeras fases del desarrollo de software (Análisis de Requisitos, Diseño e Implementación) con la finalidad de realizar una implementación de manera adecuada y con cimientos.

Un primer ítem denominado **Prueba de configuración de routing** en el que se realizará la creación del propio módulo y desarrollo del recurso de la API REST que será el encargado de registrar en la base de datos los valores leídos por el sensor que recibimos gracias a SigFox Backend. Este ítem se ha planificado para su realización en 10 horas.

El segundo ítem que se ha denominado **Comprobación de datos con AEMET** se realizará una petición a la API REST de AEMET para obtener la predicción de valores ambientales en el territorio donde se encuentra el dispositivo Siabox instalado, con estas predicciones se realizará una comprobación con los valores registrados en el ítem anterior y en caso de que no cumplan las condiciones establecidas se registrará una alarma en el sistema. Para desarrollo de este ítem se le ha asignado una duración de 30 horas.



Figura 5.9: Planificación Sprint 2. Desarrollo de módulo dedicado [Extraído de Redmine]

5.2.2. Ejecución Sprint 2

- **Prueba de configuración de routing**

Para la realización de este ítem partimos de una serie de requisitos funcionales que se han definido con anterioridad con el fin de recopilar las necesidades que existen por parte de los solicitantes y se presentan en la siguiente figura 5.10.



Figura 5.10: Requisitos Funcionales de módulo Configuración Routing

A partir de estos requisitos funcionales se van a diseñar el diagrama UML de casos de uso que forman el sistema y el comportamiento que tienen estos con entidades del sistema. Con la utilización de este diagrama se busca que de manera gráfica tengamos la posibilidad de

mostrar el comportamiento del sistema que se implementará para comenzar la codificación de las llamadas de registro del API REST utilizado en el servidor Drupal.

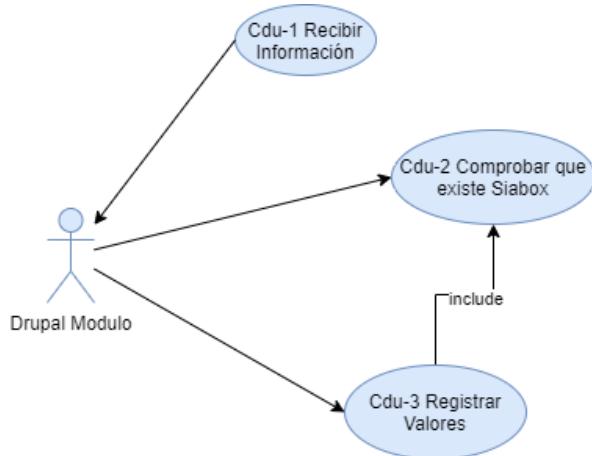


Figura 5.11: Diagrama UML de Casos de Uso de módulo Configuración Routing

El API REST que se ha definido en este apartado tiene la siguiente estructura:

Recurso: El recurso para el registro de valores ambientales será `/postSigFoxData` que se configurará en archivo de configuración de Drupal [C.2](#). Este recurso recibirá en el body de la petición HTTP un contenido de tipo `x-www-urlencoded` con todos los valores de la lectura realizada y el dispositivo que la realiza.

Método: El método que utilizará este recurso de nuestra API REST será el POST que es el verbo estandarizado para escribir información al igual que el recurso que se especifica en el archivo de configuración [C.2](#) se hace lo mismo con el método.

Significado: Utilizamos un nombre significativo para denominar al recurso. Habiendo estudiado con anterioridad la herramienta Sigfox backend que da formato a los mensajes que envía el dispositivo Sigfox optamos por www-urlencoded como tipo de contenido para realizar la petición y enviamos los valores con respecto a la creación de base de datos del [5.1](#). Utilizamos el método POST debido a que la intención de este recurso es introducir datos en la base de datos.

En nuestro servidor de pruebas en local realizamos la creación de un módulo personalizado de Drupal con la estructura que se muestra en el anexo [C.1](#). Utilizamos un servidor de pruebas en local debido a que no genera errores la ejecución en local y existe la posibilidad de realizar una tarea de depuración más eficiente y controlada.

La estructura y la relación que existe entre las clases de este módulo de Drupal se representa en el diagrama UML de clases que encontramos en el Anexo [E](#).

Utilizando la herramienta Postman se realizan pruebas de registro de valores en la plataforma, simulando el envío de valores ambientales desde la plataforma del proveedor de 0G SigFox hacia el servidor en local de Drupal.

Se envía un petición HTTP de tipo POST hacia el servidor de pruebas en local con un contenido como el que se muestra en la siguiente figura 5.12 se utiliza un identificador de dispositivo inventado que se encuentra registrado en la instalación de Drupal para realizar este tipo de comprobaciones.

KEY	VALUE
<input checked="" type="checkbox"/> idsiabox	12345
<input checked="" type="checkbox"/> Luminic	60
<input checked="" type="checkbox"/> AmbientTemp	32
<input checked="" type="checkbox"/> AmbientHum	40
<input checked="" type="checkbox"/> LandHum	70
<input checked="" type="checkbox"/> Ultraviolet	8

Figura 5.12: Petición HTTP con Postman para registro de valores de sensor

En la parte inferior derecha de la figura 5.12 podemos apreciar que la respuesta a nuestra petición nos reenvía un mensaje satisfactorio 200 OK para comprobar que el funcionamiento de nuestro código es óptimo, se revisa los contenidos creados en Drupal y el contenido de estos.

Figura 5.13: Datos del contenido registrado Valor Sensor

```

value_siabox
12345
value_luminic
60.00
value_AmbientTemperature
32.00
value_AmbientHumidity
40.00
value_LandHumidity
70.00
value_UltraViolet
8.00

```

Figura 5.14: Registro almacenado en el contenido Valor Sensor

En la figura 5.13 se puede apreciar que se registra un contenido de tipo *Valor Sensores* y visualizando la información que se encuentra registrada en ese contenido apreciable en la figura 5.14 concuerda con el cuerpo de la petición HTTP enviada.

Por lo tanto, el funcionamiento del trabajo realizado en el Sprint hasta el momento se plasma en el siguiente diagrama UML de secuencia para mostrar la secuencia de acciones 5.15.

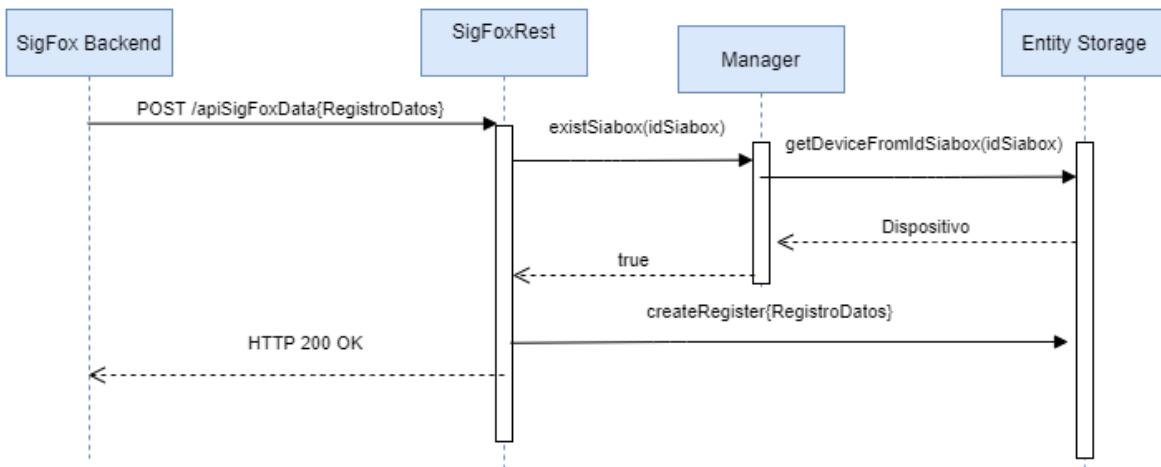


Figura 5.15: Diagrama UML de secuencia de Registro de datos ambientales

■ Comprobación de datos con AEMET

Al igual que en la tarea anterior 5.2.2 existen unos requisitos funcionales adquiridos con anterioridad que tienen la finalidad de establecer unas necesidades del proyecto para un buen funcionamiento. Estos requisitos complementan a los representados en la figura 5.10 y se utilizarán como base de desarrollo en la finalización de este Sprint.

Comprobación con Datos AEMET
RF4 - Solicitar Previsiones
RF5 - Procesar Previsiones
RF6 - Realizar Comprobacion
RF7 - Registrar Alarma

Figura 5.16: Requisitos Funcionales Comprobación AEMET

Una vez realizado el análisis de los requisitos funcionales capturados para la realización de esta tarea. Se genera como resultado un diagrama UML de casos de uso para reflejar el comportamiento del sistema y las entidades de este. Este diagrama como se ha mencionado en párrafo anterior, se hace como ampliación del diagrama de la figura 5.11.

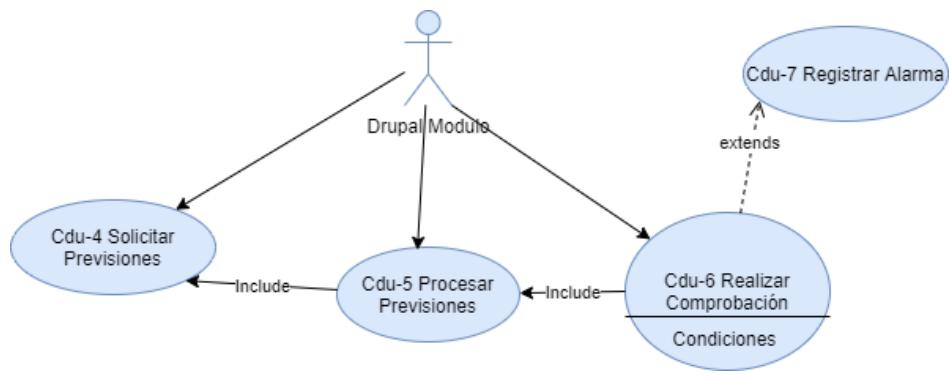


Figura 5.17: Diagrama UML Casos de Uso Comprobación AEMET

Cuando vamos a utilizar información que proviene de otra entidad, lo primero que debemos de hacer es realizar un análisis y elección de los recursos que se necesitan para dotar de la funcionalidad establecida a nuestra arquitectura. Para solucionar este problema. Se ha hecho uso de la documentación que AEMET pone a nuestra disposición en la plataforma Swagger¹ para realizar de manera visual el análisis de información.

Se decide utilizar la opción que nos afecta a más cantidad de valores de nuestras lecturas. Nos adentramos en la descripción para conocer los requerimientos de nuestras peticiones HTTP para obtener la información.

Para realizar una petición GET que debe incluir en la cabecera tanto el formato en el que deseamos la respuesta que será JSON como la clave que AEMET nos ha proporcionado para poder hacer uso de esta funcionalidad.

```
--header 'Accept: application/json' --header 'api_key: eyJhbGciOiJIUzI1NiJ9.eyJ
```

Figura 5.18: Información requerida en la cabecera petición AEMET

Al realizar la petición sobre esta URL debemos añadir un parámetro que se obtendrá de manera manual previa al registro del cultivo en la nuestro sistema que encontraremos en documento alojado la web² que representará el código de población sobre la que queremos obtener datos. Una vez realizada la petición como se muestra en la siguiente figura, adjuntando los valores de cabecera anteriormente mencionados se obtiene como respuesta 5.19.

Request URL	<code>https://opendata.aemet.es/opendata/api/prediccion/especifica/municipio/diaria/13023</code>
Response Body	<pre>{ "descripcion": "exito", "estado": 200, "datos": "https://opendata.aemet.es/opendata/sh/3d149e35", "metadatos": "https://opendata.aemet.es/opendata/sh/dfd88b22" }</pre>

Figura 5.19: Respuesta recibida a la petición GET

Trabajando sobre la respuesta obtenida se observa que realiza la petición de manera adecuada y obtenemos 2 direcciones la correspondiente a la clave *datos* que contiene el enlace a los datos de

¹<https://opendata.aemet.es/dist/index.html#/prediccion-especificas>

²<https://www.ine.es/daco/daco42/codmun/codmunmapa.htm>

provisiones meteorológicas y *metadatos* documento donde se especifica la estructura en la que se ofrecen las predicciones. Además, se contará con los períodos de tiempo válidos, unidades de medida en la que se expresan los valores y descripciones de los campos. En nuestro caso, se utilizarán las previsiones para comparar valores de **humedad ambiente, temperatura y radiación ultravioleta** que solo nos ofrecerá el máximo diario. Debemos advertir que existe una restricción temporal de 50 segundos entre usos de la clave de API de AEMET que en nuestro caso no afectará al desarrollo del proyecto.

Una vez realizado este estudio sobre la API REST de AEMET se procede a la explicación sobre la implementación de este ítem. La estructura de directorios de este módulo de Drupal la encontramos en C.2. Incorporamos un diagrama UML para mostrar las relaciones que tienen las distintas clases que componen este módulo que se puede encontrar en el Anexo E.

Según [1], .^{ETL} es un tipo de integración de datos que hace referencia a los tres pasos (extraer, transformar, cargar) que se utilizan para mezclar datos de múltiples fuentes. Se utiliza a menudo para construir un almacén de datos. Durante este proceso, los datos se toman (extraen) de un sistema de origen, se convierten (transforman) en un formato que se puede almacenar y se almacenan (cargan) en un data warehouse u otro sistema".

Para comprobar la generación de alarmas, se simula una lectura de valores ambientales muy extremos comparados a los que se encuentran en las predicciones. Para poder realizarlo se hará uso de la herramienta para envío de peticiones HTTP Postman.

KEY	VALUE
IdsLabox	12345
Luminic	53
AmbientTemp	40
AmbientHum	80
LandHum	70
Ultraviolet	11

Figura 5.20: Simulación generación de alarmas con Postman

Durante esta comprobación, desde el módulo personalizado instalado en nuestro Drupal se realizará una petición HTTP gracias a la librería de PHP *Guzzle* hacia el endpoint de AEMET indicado 5.19, una vez obtenido el JSON con los datos sobre la predicción meteorológica se aplica un proceso ETL (Extraer, Transformar y Cargar). Para empezar, **Extraemos** todos los datos que forman parte de la respuesta, realizamos una **transformación** mediante un filtrado de información para no utilizar parámetros ambientales que no aporten valor y normalizamos estos valores que pasan el filtro para poder utilizarlos de manera correcta en la comparación. Para finalizar, se **cargarán** estos datos en la comprobación y como se explica en el siguiente párrafo si se cumplen las condiciones pasaran a nuestra base de datos.

Si durante las comprobaciones se detecta que las lecturas recibidas por parte del dispositivo Siabox no están dentro de los rangos establecidos se registrará Alarma en nuestra base de datos. Para poder comprobar el registro de datos en nuestra base de datos se debe acceder a la instalación de Drupal como se muestra en la figura 5.21.

	TÍTULO	TIPO DE CONTENIDO	AUTOR	ESTADO	ACTUALIZADO	OPERACIONES
<input type="checkbox"/>	2021-06-22 18:55	Alarma Sensor	Anónimo (no verificado)	Publicado	22/06/2021 - 18:55	<button>Editar</button>
<input type="checkbox"/>	Registro	Valor Sensores	Anónimo (no verificado)	Publicado	22/06/2021 - 18:55	<button>Editar</button>

Figura 5.21: Evidencia de generación de registro y alarma correspondiente

Se realiza una comprobación de para verificar que los registros han generado los contenidos correspondientes en el entorno Drupal y se procede a verificar que los valores registrados en esos contenidos se estructuran de manera adecuada y con los valores adecuados.

idsiabox
12345

Error Temperatura
Activado
Valor Temperatura
40.00
Error Humedad
Activado
Valor Humedad
80.00

(a) Parte 1

Error Ultravioleta
Desactivado
Valor Ultravioleta
11.00
Valor Registrado
[Registro](#)
checked
Desactivado

(b) Parte 2

Figura 5.22: Valores de Alarma Sensor registrada

Para poder mostrar de manera más clara la ampliación de funcionalidad que se ha desarrollado mediante el trabajo realizado en este Backlog Product sobre el producto anterior. Se adjunta el diagrama UML de secuencia 5.23 que muestra la funcionalidad del incremento realizado sobre la arquitectura que ha obtenido en este Sprint.

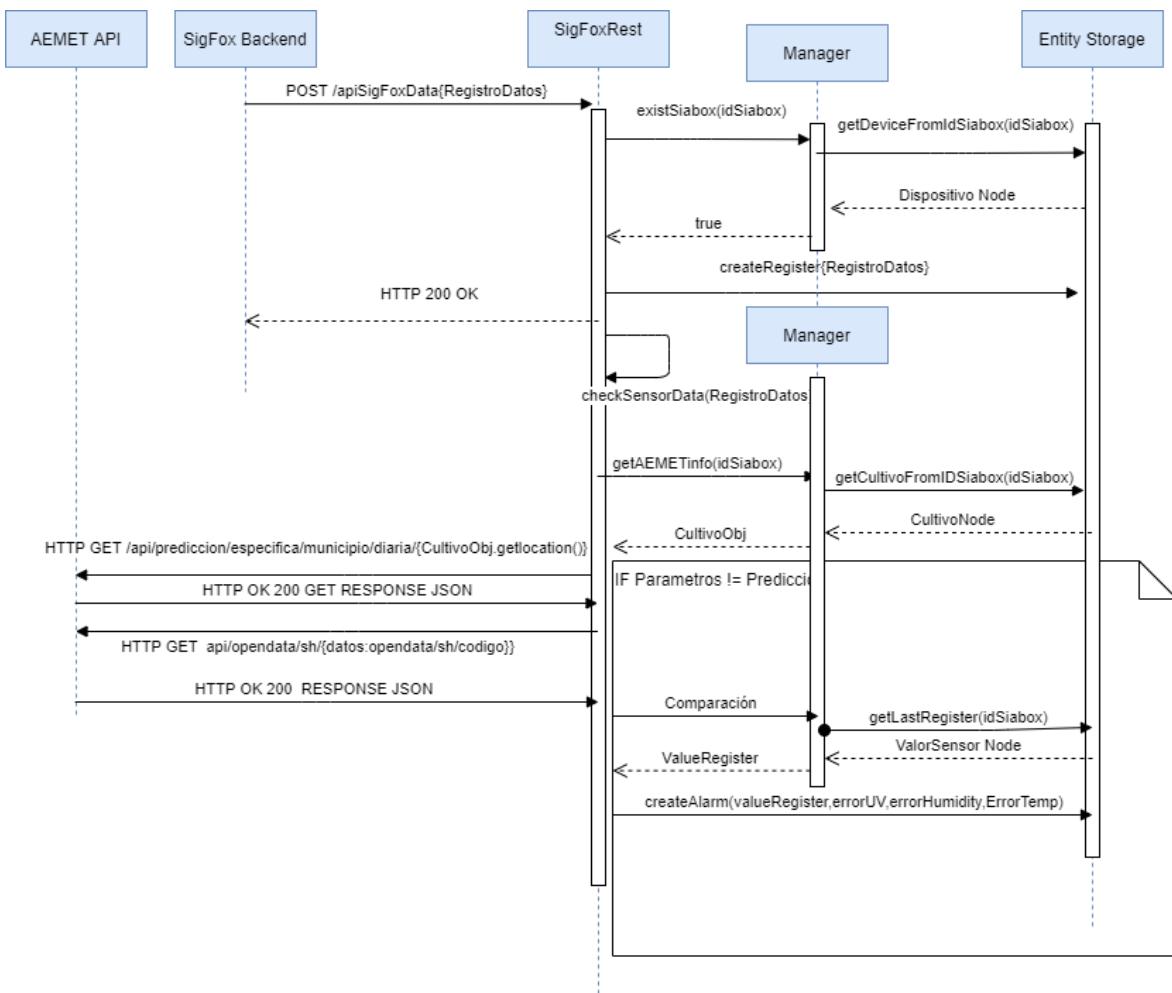


Figura 5.23: Diagrama UML de Secuencia Sprint 2 completo

5.2.3. Conclusión Sprint 2

En la realización de este Sprint se han encontrado varios problemas que han hecho que se haya tenido que dedicar muchas más horas que las estimadas en la planificación. Como se puede apreciar en la imagen 5.24.

Al encontrar este problema en el Sprint, se le comunica al Product Owner que la duración planificada para ambos items es escaso y que se van a generar desviaciones. Después de la comunicar esta situación se continua con la ejecución del Sprint.

En el primer item, para la realización de manera satisfactoria de los objetivos buscados se ha sufrido con una variación de horas 6 veces mayor que las horas planificadas, debido a problemas entorno al funcionamiento de la versión de Drupal y el formato de envío de datos por parte de SigFox Backend. Por otro lado, en el Item 2 se ha encontrado una variación del doble del tiempo establecido para lograr el objetivo propuesto en la planificación debido a los problemas encontrados en la realización de la fase de extracción de valores del proceso ETL. Esto es debido al tamaño de la respuesta recibida y la codificación en la que estabamos trabajando UTF-8³

³UTF-8 es un formato de codificación de caracteres Unicode e ISO 10646 que utiliza símbolos con longitud variable.

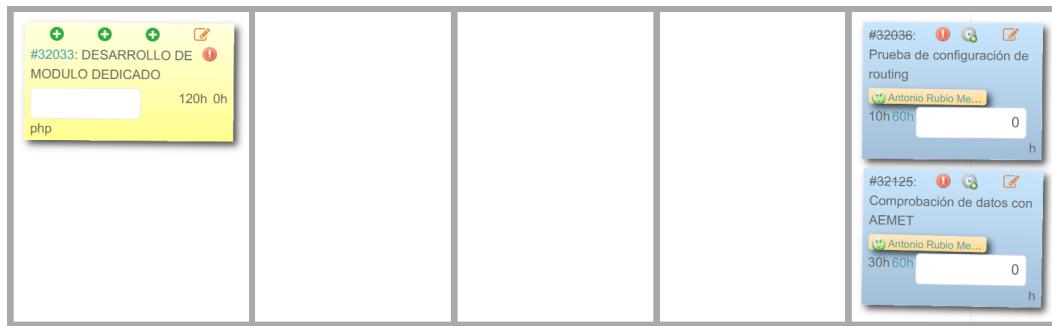


Figura 5.24: Conclusión Sprint 2. Desarrollo de módulo dedicado

5.3. SPRINT 3.VISUALIZACIÓN DE DATOS

En este Sprint se va a realizar el desarrollo de una aplicación móvil para la visualización de la información que genera el Sprint 5.2 . Además, para poder llevar esto acabo se creará un módulo para creación de una API REST que ofrezca la comunicación de la app móvil y la instalación de Drupal.

5.3.1. Planificación Sprint 3

Se programa una reunión con el product owner para planificar el tercer Sprint, con el objetivo de no repetir el fallo de planificación del Sprint anterior, se creará un product backlog con 4 item backlog definidos y un tiempo estimado de realización más amplio para cada uno de esos items.

- **Creación de Bocetos** para el que destinaremos 30 horas de trabajo, en este item nos centraremos en crear una primera idea de las pantallas de nuestra aplicación móvil. Para ello contaremos con la herramienta de creación de bocetos Balsamiq Mockup.
- **Creación de Modulo Drupal para Muestra de datos** para poder comunicarnos con nuestro servidor donde se encuentra la información y así poder realizar peticiones de información desde nuestra App. Se desarrollará un módulo específico de Drupal con la intención de crear un API REST para proveer de información a la App. Esta tarea tendrá una designación de 40 horas para su realización.
- **Creación Login** con motivo de securizar el acceso a la información y facilitar el trabajo a la hora de controlar la pertenencia de la información, se utilizará una funcionalidad integrada de Drupal llamada REST UI⁴. Se le ha asignado un tiempo de 10 horas para su realización.
- **Desarrollo de Gráficas** la tarea que mas importancia tiene con respecto a los usuarios. La App se desarrollara con las tecnologías Angular e Ionic con la intención de adaptarla a dispositivos móviles utilizando Capacitor. Para la muestra de datos se utilizaran gráficas creadas con la librería *Chart.js*. Esto hace que esta parte tenga mucho valor debido a que sin usuarios la Arquitectura IoT no tendría sentido. Poder visualizar la información de una manera clara. Para el desarrollo de este item se ha designado un tiempo de 16 horas.

Como en el periodo de formación en las tecnologías se ha detectado la incompatibilidad de los servidores en local con el desarrollo de aplicaciones de Ionic debido a esto la realización de este apartado debemos de utilizar un servidor remoto que no este ubicado en local para evitar problemas

⁴<https://www.drupal.org/project/restui>

de CORS (Cross-origin resource sharing)⁵.



Figura 5.25: Planificación Sprint 3. Visualización de Datos

5.3.2. Ejecución Sprint 3

Antes de comenzar con la ejecución de los items que forman el Sprint Backlog se obtiene los requisitos para la consecución de una manera satisfactoria y estandarizada del objetivo de este sprint. En la siguiente figura 5.26 se representan los requisitos funcionales definidos para la realización de nuestra App.



Figura 5.26: Requisitos Funcionales Sprint 3

Una vez definidos los requisitos funcionales que debe satisfacer nuestra aplicación móvil se genera un diagrama UML de casos uso con la finalidad de representar el entorno del sistema y el comportamiento que tendrá cada uno de los elementos del sistema con los casos de uso.

⁵CORS: mecanismo que permite que se puedan solicitar recursos restringidos en una página web desde un dominio diferente del dominio que sirvió el primer recurso.

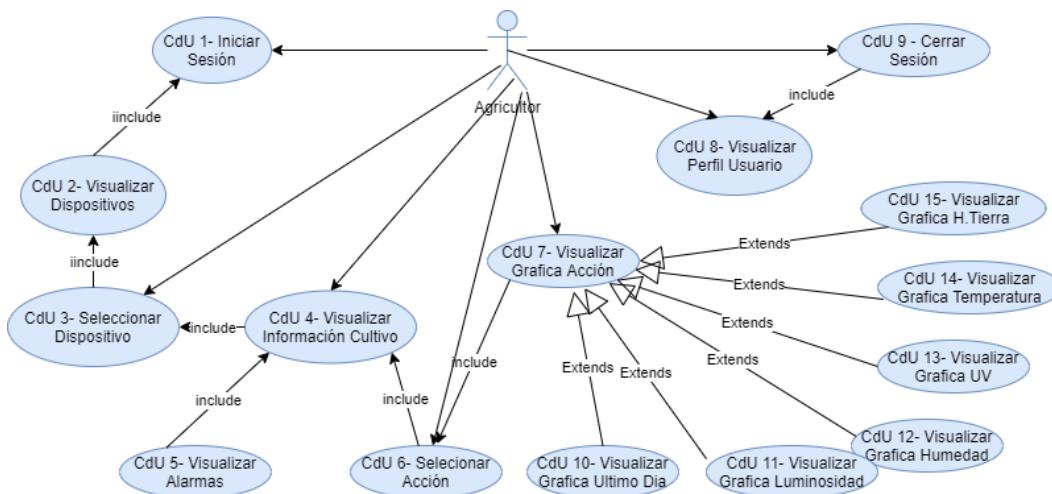


Figura 5.27: Diagrama UML de Casos de Uso Sprint 3

■ Creación de Bocetos

Utilizando la herramienta Balsamiq Mockup se han creado los prototipos de las pantallas que formarán parte de nuestra App. Estos prototipos vienen motivados por fase de análisis de requisitos y diseño de software que se ha realizado de manera previa al comienzo del sprint y que además influirán a lo largo del mismo.

Login

Será la ventana inicial de nuestra aplicación mediante la que controlaremos el acceso. Se representa en la figura 5.28a.

Home

Ventana donde accederemos si la autentificación se realiza de manera satisfactoria, desde esta ventana se podrán visualizar el listado de dispositivos asignados al usuario con el que hemos iniciado sesión. Se muestra un boceto en la imagen 5.28b.

Perfil

El acceso a esta ventana se realiza mediante el botón situado en la parte superior derecha de todas las ventanas. Desde esta ventana se pueden visualizar los datos del usuario y cerrar sesión. El boceto se muestra en la figura 5.28c.

Acciones

La figura 5.28d se encuentra la información del dispositivo y cultivo. Además, se incluyen unos botones para acceder a la ventana de visualización de los valores registrados por el dispositivo SiaBox. En la parte inferior, se incorpora un listado de Alarms donde se muestra el tipo de alarma representado y la fecha en la que se produce.

Gráficos

La figura 5.28e muestra en una gráfica de áreas donde se mostrarán la información de los seis botones localizados en la ventana 5.28d. En esta ventana incorporamos un apartado para comprensión de los datos mostrados con un apartado donde se explican las unidades en las que están representadas los datos.

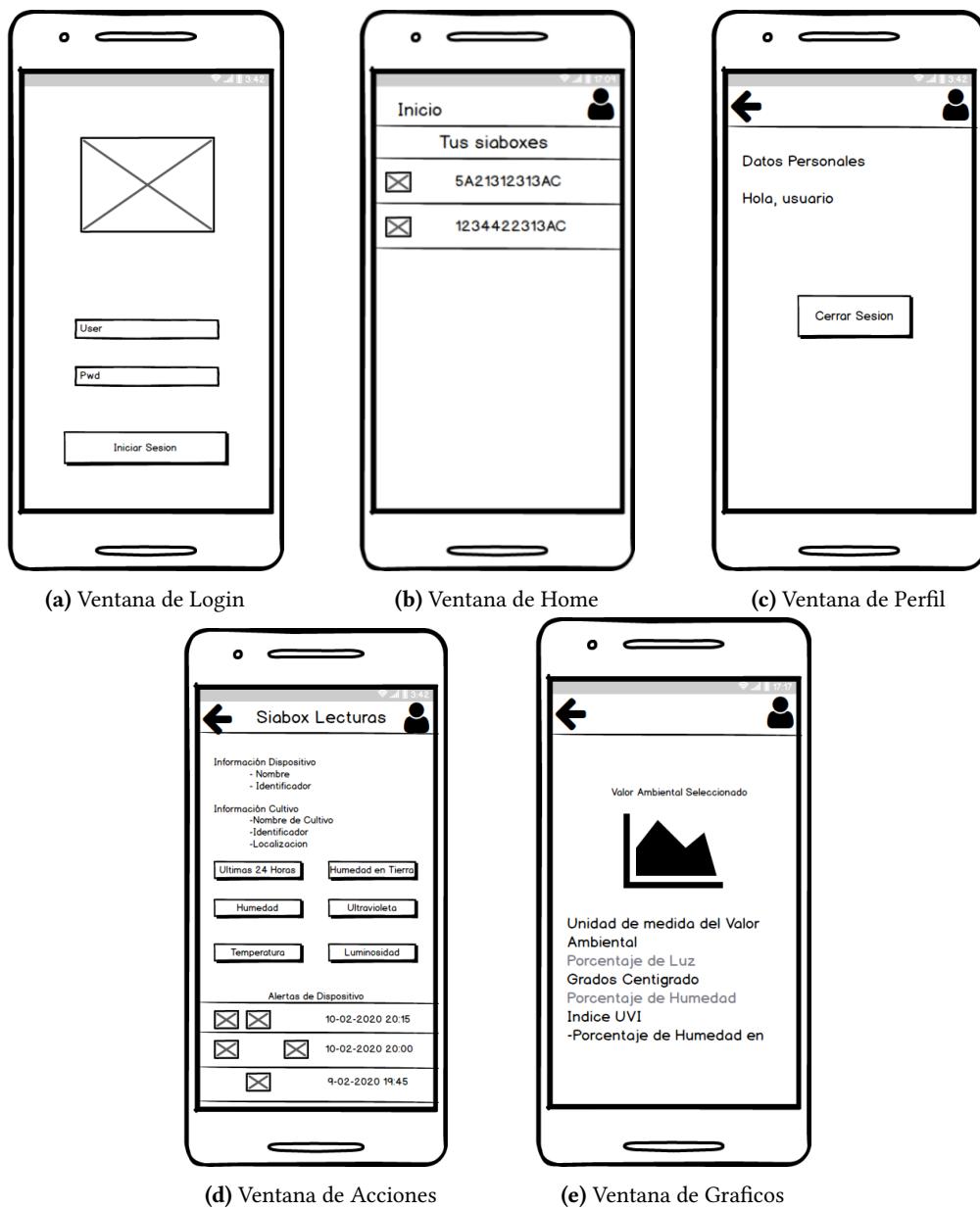


Figura 5.28: Bocetos de aplicación móvil

■ Creación de Modulo Drupal para Muestra de datos

En la tarea de este ítem se va a crear un módulo personalizado de Drupal con la intención de tener una API REST para que pueda ser consumida desde nuestra app móvil y así poder realizar envíos de información solicitada por nuestros usuarios.

El API REST que se va a utilizar en este caso solo tiene un único recurso se ha definido en este apartado tiene la siguiente estructura:

Recurso: El recurso para la extracción de información de nuestro servidor Drupal `/ionicRest` que se configurará en archivo de configuración de Drupal C.6. Este recurso recibirá en los parámetros de la URI de la petición HTTP el tipo de información que quiere del servidor y en su cabecera incluiremos el código de Basic Auth⁶ para autenticar en el servidor.

⁶la autentificación de acceso básica es un método diseñado para permitir a un navegador web, proveer credenciales en la forma de usuario y contraseña cuando se le solicita una página al servidor.

Método: El método que utilizará este recurso de nuestra API REST será el GET que es el verbo estandarizado para extraer información. Al igual que el recurso que se especifica en el archivo de configuración C.6 se hace lo mismo con el método.

Significado: Utilizamos un nombre significativo para denominar al recurso. Basándonos en los bocetos 5.28 y diagramas UML de casos de uso 5.27 creamos nuestro propio estándar para extraer información que será *uri/ionicREST?type=variable&uid=id* donde *variable* tomará un valor distinto dependiendo de la información que queramos extraer y *uid* el identificador del elemento a solicitar. Variable puede tomar los siguientes valores.

- **dispositivo** esta petición enviará una respuesta JSON con un listado de los dispositivos Siabox pertenecientes al usuario que ha iniciado sesión.
- **dispositivoOnly** esta petición enviará una respuesta JSON con la información detallada del dispositivo Siabox que concuerde con el identificador incluido en la URI.
- **lastDay** cuando el parámetro type toma el valor lastDay se enviará una respuesta JSON con los valores ambientales registrados en el último dia del que tengan como dispositivo el nid igual al valor de uid.
- **historialLuminic** cuando el parámetro type toma el valor historialLuminic se enviará una respuesta JSON con el historial de luminosidad en porcentaje de luz del cultivo que tenga como dispositivo asignado el nid igual al valor de uid.
- **historialAmbientTemp** esta petición con el parámetro type tomando el valor historialAmbientTemp se enviará una respuesta JSON con el historial de temperatura en
 - ambiente del cultivo que tenga como dispositivo asignado el nid igual al valor del parámetro uid.
- **historialAmbientHumidity** cuando el parámetro type toma el valor historialLandHumidity se enviará una respuesta JSON con el historial de humedad ambiente en porcentaje del cultivo que tenga como dispositivo asignado el nid igual al valor de uid.
- **historialLandHumidity** cuando el parámetro type toma el valor historialLandHumidity se enviará una respuesta JSON con el historial de humedad en tierra en porcentaje del cultivo que tenga como dispositivo asignado el nid igual al valor de uid.
- **historialUltraviolet** cuando el parámetro type toma el valor historialLandHumidity se enviará una respuesta JSON con el historial de radiación ultravioleta en UVI que ha sido del cultivo que tenga como dispositivo asignado el nid igual al valor de uid.
- **updateAlarm** esta petición se utilizará para actualizar la alarma seleccionada tendrá un parámetro adicional que incluirá la fecha en la que se genera la alarma.
- **alarms** esta petición enviará en la respuesta JSON todo el listado de alarmas asignado a un dispositivo.
- **newAlarms** esta petición se utilizará para revisar si existe alguna nueva alarma generada por parte del sistema y no ha sido revisada por el usuario. Enviará al cliente una respuesta JSON con el número de alarmas nuevas generadas.

Sobre las librerías que han incluido en el desarrollo de esta API REST se hace una mención especial al objeto JsonResponse de Symphony que permite preparar y manipular las respuestas antes de que sean devueltas al cliente y en este caso, convertir el tipo de contenido en formato JSON.

Una vez definida la API REST, se da a conocer la estructura del módulo instalado en el servidor de Drupal que se encuentra en el Anexo C.2 y el diagrama UML de clases en el Anexo E.

■ Creación Login

Para poder utilizar la aplicación móvil de forma segura debemos asegurarnos que podemos aislar el público que puede acceder a ella y usuario que acceda a esta lo haga identificándose de una manera correcta. Por ello, se ha desarrollado esta funcionalidad utilizando REST UI de Drupal y peticiones HTTP desde nuestra aplicación Angular+Ionic. La creación del Login seguirá a nivel visual del boceto 5.28a.

En la siguiente figura E.4 que se encuentra en el Anexo E se representa la estructura de la funcionalidad Login mediante la utilización de un diagrama de clases UML. Como se puede apreciar a la parte visual se la ha representado a escala de paquete.

Se puede identificar la clase *Login.Page* en el paquete Model y al servicio *Login.Service* en el paquete Controller.

Una vez descrita la estructura del trabajo realizado en este item, hay que explicar el funcionamiento del desarrollo producido en este item.

Primero, se va a realizar unas peticiones HTTP modo de prueba con la herramienta Postman para comprobar el funcionamiento del endpoint que Drupal ofrece a los desarrolladores para el inicio de sesión en su portal.

Se crea una petición POST hacia el endpoint indicado en el módulo de Drupal REST UI. El formato de envío será JSON y en el cuerpo de la petición se incluirá un usuario y una contraseña ya registrados en la base de datos de Drupal.

La respuesta a esta petición será en formato JSON y contendrá información sobre el usuario actual, identificador dentro de Drupal y nombre. Además, se incluirán 2 tokens. El primero será de sesión que según las recomendaciones encontradas en la formación se desecha y un segundo será utilizado para cerrar la sesión en Drupal.

```

POST https://arubio.es/user/login?_format=json
Status: 200 OK

Body
  raw JSON Beautify
  1 {
  2   "name": "Antonio Rubio",
  3   "pass": "████████"
  4 }

  1 {
  2   "current_user": {
  3     "uid": "26",
  4     "name": "Antonio Rubio"
  5   },
  6   "csrf_token": "BTPuIAsB7p7L3BN2aYrPkxX5dpsk66X5RdViid6VtaA",
  7   "logout_token": "z7_q10I4molaaum0llq7dTLOvyrykYPx8L68mgis_yQ"
  8 }
  
```

Figura 5.29: Simulación de Login con Postman

Una vez revisado el correcto funcionamiento del endpoint para inicio de sesión, Se procede a revisar la obtención del identificador de sesión para el inicio de sesión. En la siguiente imagen [5.30](#) puede comprobar la petición de envío y la respuesta.

The screenshot shows a Postman interface with the following details:

- Method:** GET
- URL:** https://arubio.es/session/token
- Headers:** Params, Auth, Headers (7), Body, Pre-req., Tests, Settings
- Body:** Body ▾
- Response Status:** 200 OK
- Response Body:**

```
1   UCIInnS13IO5ui80RsNQh7rPltr4iBOZLVU8TwOLGFEI
```

Figura 5.30: Simulación de obtención de token de sesión con Postman

Para mostrar el funcionamiento del este item se realiza un diagrama de secuencia con una situación convencional de login en la aplicación web. Desde el desencadenamiento del click en el botón iniciar sesión pasando por la petición de información hacia el servidor Drupal. Hasta la validación de credenciales y re-direccionamiento a la ventana de inicio. En la próxima tarea del sprint se mostrará el funcionamiento del caso de uso Logout que se ha desarrollado en este backlog item.

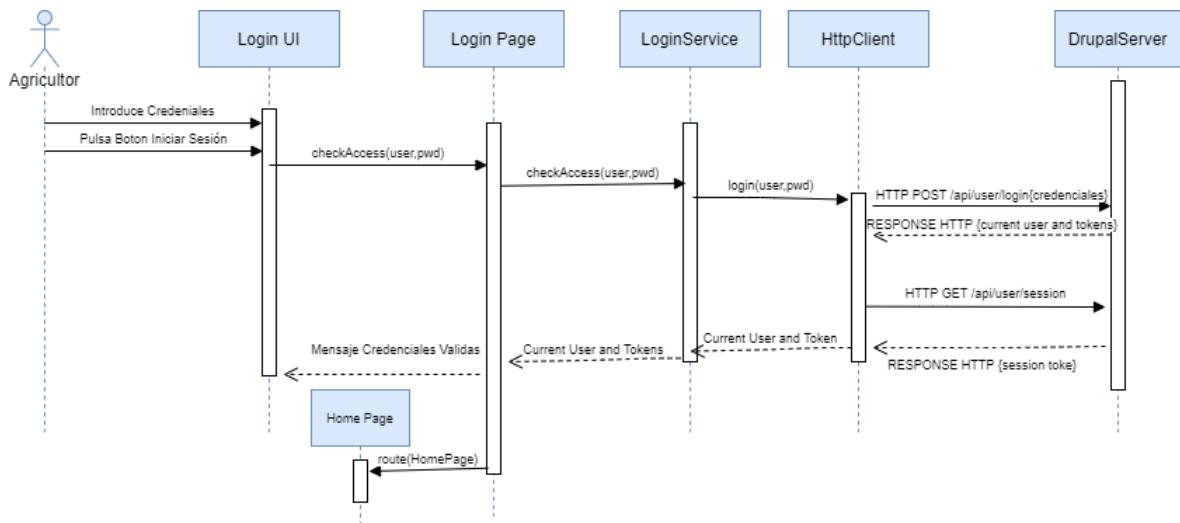


Figura 5.31: Diagrama UML de secuencia de Login

■ Desarrollo de Gráficas

Este será el ítem del Sprint donde se desarrollará la navegación entre pantallas y las funcionalidades de estas para mostrar información a los usuarios. Después de crear las ventanas con alguna modificación con respecto a los bocetos creados en al comienzo del Sprint 5.28.

Para la realización de item ha sido necesario la utilización de manera agrupada de una gran diversidad de tecnologías. Para el aspecto visual se ha utilizado una fusión de tecnologías Angular + Ionic para la creación de la aplicación web. Por otro lado, la lógica de la aplicación móvil ha requerido la configuración del direccionamiento de cada una de las ventanas y la utilización de diversas librerías como han sido:

1. **HttpClient**. Utilizada para realizar peticiones HTTP sobre la API creada en el Sprint 2 para obtener información almacenada en el lado Servidor.
2. **Router**. Esta librería nos ha facilitado la navegación entre ventanas haciendo que nuestro código sea muchas más sencillo e intuitivo.
3. La libreria **AlertController** ha sido utilizada para asegurar que el usuario tiene la intención de realizar una acción (e.j Cerrar Sesión).
4. **ActivatedRoute**. Gracias a la utilización de esta librería se ha podido obtener información que estaba introducida en la URL con la que se estaba trabajando.
5. **Chart.js** ha proporcionado al proyecto la capacidad de mostrar la información destinada al usuario de una manera clara gracias a unos diagramas de áreas dinámicos. Representando desde un único campo por gráfica hasta la representación de todos los campos en una misma gráfica.

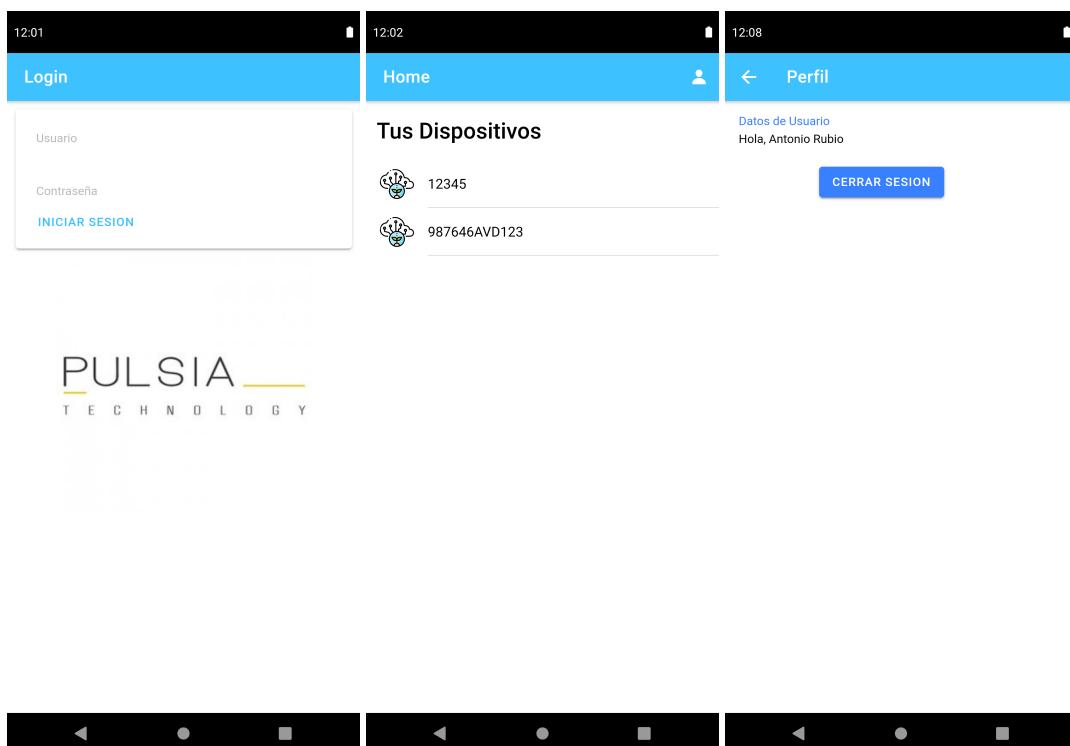


Figura 5.32: Bocetos de aplicación móvil

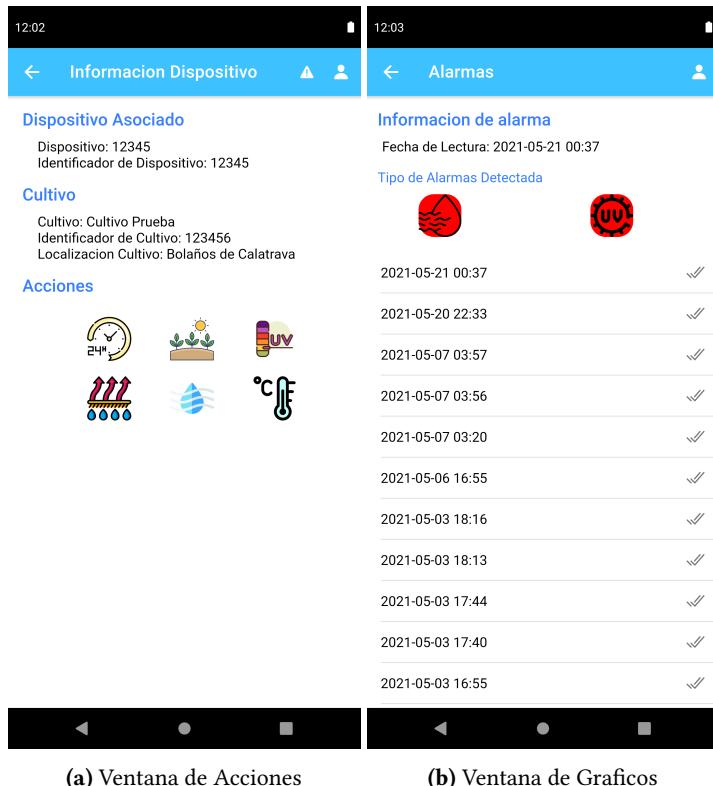


Figura 5.33: Bocetos de aplicación móvil 2

Como se puede apreciar se realizó un cambio con respecto a los bocetos 5.28 realizados al principio del Sprint. Dándole una mayor importancia a las alertas con la implementación de una ventana exclusivamente para ellas e incluyendo los datos gráficos en una ventana sin necesidad de re-direcciónamiento.

Debido a la gran extensión de archivos que se encuentran en el proyecto. Para mostrar el trabajo realizado en este ítem se ha optado por mostrar su estructura, clases que lo forman y relaciones que existen entre ellas utilizando un diagrama de clases UML E.5 que puedes encontrar en el Anexo E.

En el diagrama se crea un bloque HTML + CSS para representar la parte perteneciente al paquete de *Vista* de cada una de las ventanas que van a formar parte de la aplicación móvil.

Por otro lado, en el paquete Controller se encontrarán los servicios y objetos que se han utilizado para llevar a cabo la petición de información al servidor mediante peticiones HTTP. Se pueden observar dos servicios bien diferencias uno *LoginService* orientado a como bien indica su nombre relación de acceso a la aplicación y cierre de sesión y *DeviceService* encargado de la petición relacionada con la Arquitectura.

En el paquete de *Model* se encuentran las clases que comunican la parte visual de cada una de las ventanas con el servicio utilizado. Existirá una clase por cada una de las pantallas que forme parte de nuestra App. Además, incluiremos enumeraciones e interfaces para facilitar el desarrollo de esta aplicación web.

Para representar el funcionamiento de la App desarrollada se van a incluir una serie de escenarios de uso mediante la utilización de diagramas UML de secuencia. Además, se incluirá una imagen de la App con el estado final de cada uno de los escenarios para una mejor interpretación del funcionamiento de este.

Primero, vamos a comenzar con la selección de Dispositivo y muestra de cualquier tipo de valor ambiental. Para ello hay que recalcar que el inicio de sesión se habría realizado con anterioridad y de manera satisfactoria conforme a la figura 5.31.

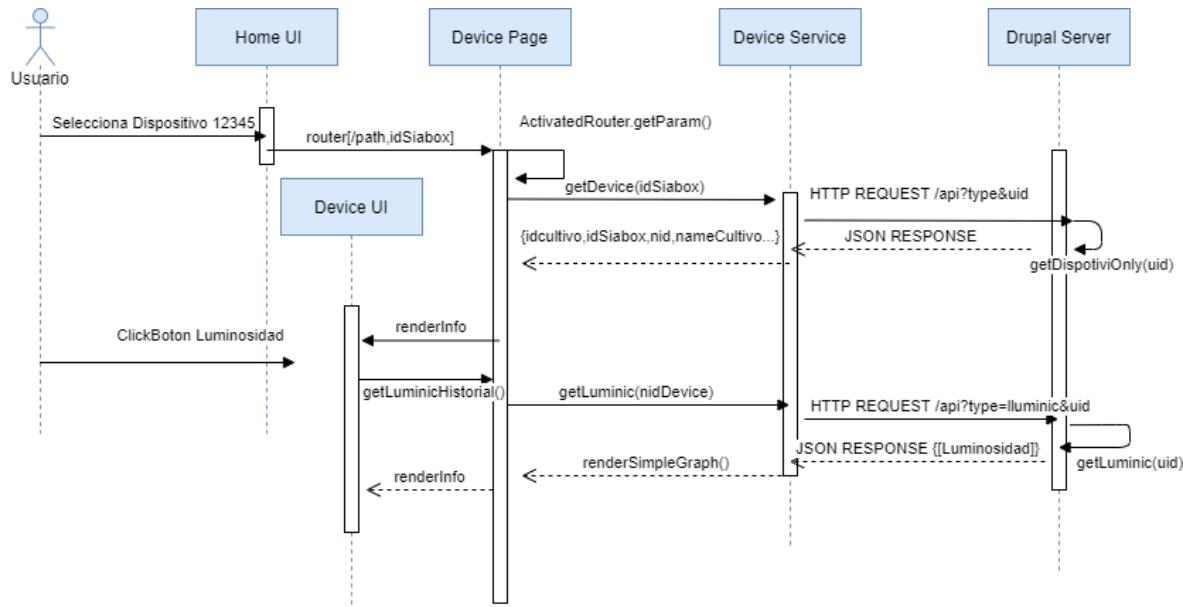


Figura 5.34: Diagrama UML de secuencia visualización de luminosidad

Para el siguiente escenario, hay que situarse en la finalización del escenario anterior, el usuario se sitúa en la ventana de 5.33a habiendo solicitado información sobre uno de los parámetros ambientales que se ofrecen. Desde aquí el usuario realizará una revisión de las alarmas que su dispositivo ha registrado.

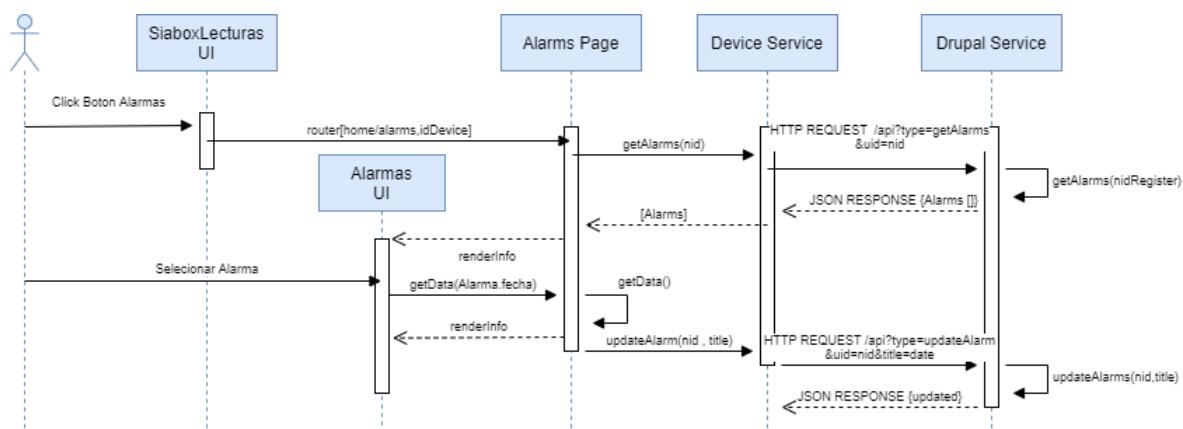


Figura 5.35: Diagrama UML de secuencia visualización de alarma

Todo lo que empieza tiene que acabar y este caso no iba a ser menos. Para ello, se crea el escenario en el que un usuario desea cerrar sesión. Esto lo podrá realizar gracias a la pantalla *Profile* la que tendrá acceso desde cualquier ventana de la aplicación.

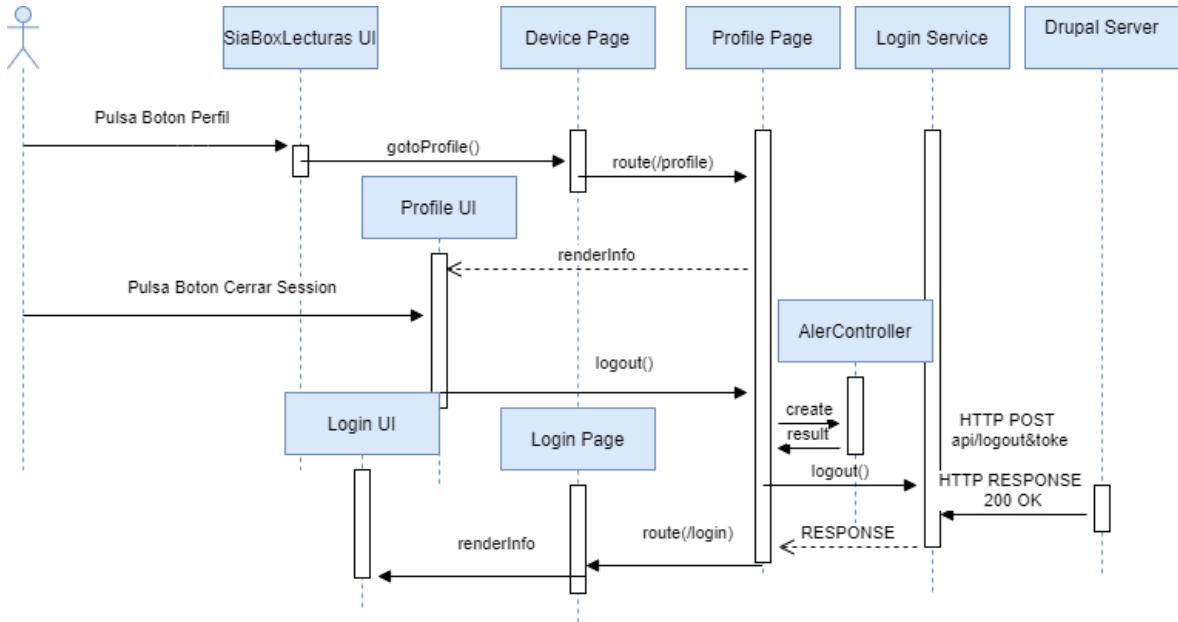


Figura 5.36: Diagrama UML de secuencia cerrar sesión

En este punto de la ejecución del sprint tendríamos una aplicación web responsive para dispositivos móviles. Sin embargo, nuestro objetivo creación de una app nativa en su totalidad, se hace uso [4.3.9](#) para reescribir la versión web como una aplicación móvil Android nativa. Para ello incorporamos una demostración de transformación de la app.

Para realizar la conversión de aplicación web a aplicación Android nativa debemos seguir esta secuencia de ejecución de comandos en el shell.

Primero, nos ubicamos en el directorio raíz de proyecto Ionic. En este momento instalaremos todos los módulos necesarios de node. Forzamos la descarga de la versión 2.4.7 del core debido a que las versiones futuras pueden llegar a generar errores de transformación.

Listado 5.1: Instrucción para Instalación de librerías

```
npm install --save @capacitor/core~2.4.7 @capacitor/cli
```

Una vez instalado Capacitor en nuestro proyecto debemos de iniciararlo con el siguiente comando.

Listado 5.2: Instrucción para Creación de Capacitor a proyecto existente

```
npx cap init
```

Al iniciar debemos introducir la información del proyecto y generará un archivo JSON con la configuración de nuestra app. A continuación generaremos nuestro archivo WWW que contendrá la información a convertir en Android.

Listado 5.3: Instrucción para Creación de proyecto WWW

```
ionic build
```

Para convertir en Android u otro sistema operativo generamos con Capacitor el directorio donde se ubica la aplicación.

Listado 5.4: Instrucción para Creación de Proyecto Android

```
npx cap add android
```

En el caso de que tengamos instalado Android Studio en nuestro dispositivo podremos realizar la apertura y prueba de nuestro proyecto de la siguiente manera.

Listado 5.5: Instrucción para Apertura de Android Studio

```
npx cap open android
```

5.3.3. Conclusión Sprint 3

Tras una reunión con el tutor de prácticas para certificar que los objetivos del tercer se habían cumplido de manera correcta. Se detecta una pequeña desviación en la realización de la segunda tarea (8 horas) con respecto a la cantidad de tiempo estimada. Por lo tanto, el tiempo dedicado total a la realización de este Sprint 3 es de 104 horas aunque este estaba planificado realizarlo en 96.

El desglose de la dedicación por ítem del product backlog de este ítem se detalla en la siguiente figura 5.37.

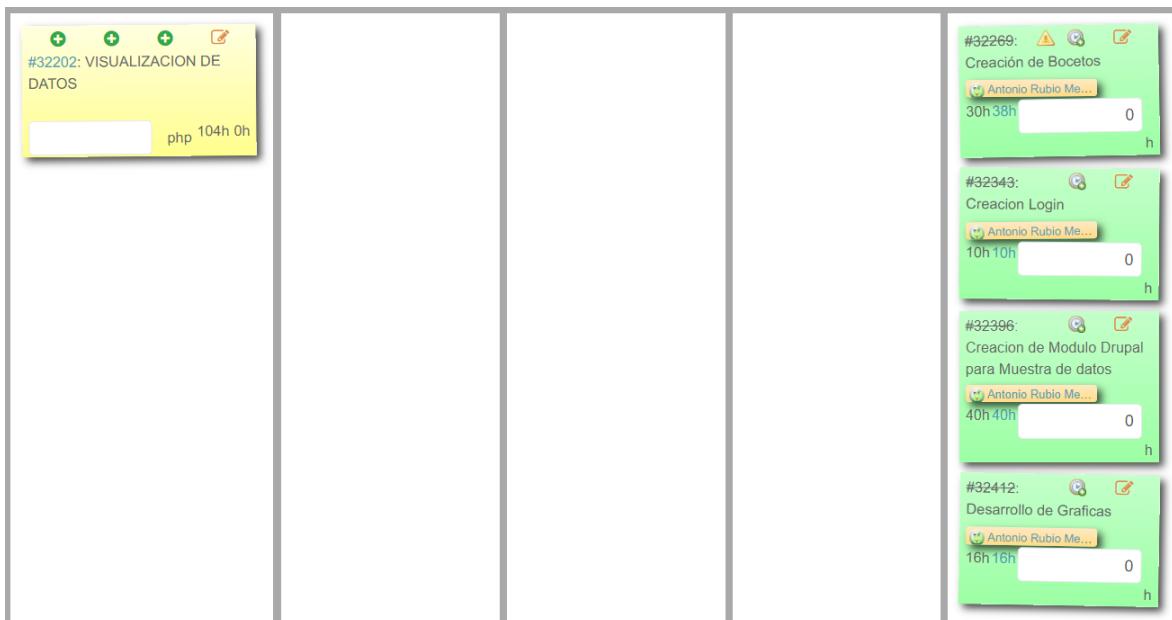


Figura 5.37: Conclusión Sprint 3. Visualización de datos

5.4. SPRINT 4.MONTAJE Y CONFIGURACIÓN HARDWARE

En este Sprint se va a realizar el montaje y ensamblado de los componentes (Placa Arduino y Sensores) que forman parte del dispositivo Siabox y su posterior configuración para captar la información de los sensores, temporalidad de lecturas, procesamiento de valores y realizar envíos de datos mediante la red LPWAN de SigFox hacia su backend.

5.4.1. Planificación Sprint 4

Se convoca una reunión con el tutor de prácticas para planificar el desarrollo del Sprint 4. En esta reunión se decide el desglose de sprint backlog para llegar a obtener un incremento para la Arquitectura. En este caso se ha decidido la creación de 2 items que serán:

- **Instalación de Placa Arduino**

Se realiza el registro de la placa Arduino MKRFOX 1200 en la plataforma para posteriormente integrarla junto al resto de sensores que forman el dispositivo Siabox. Esto, es necesario debido a que sin realizar el registro de los dispositivos en la plataforma no pueden tener acceso a la red de SigFox. Se le ha asignado una duración de 20 horas.

- **Desarrollo de recolección de datos**

En este item se desarrollara el código para obtención de los valores captan los sensores, procesar esos valores para darles un formato legible y enviarlos al SigFox Backend. La duración estimada para la realización de esta tarea será de 25 horas.



Figura 5.38: Planificación Sprint 4. Montaje y Configuración Hardware

5.4.2. Ejecución Sprint 4

- **Instalación de Placa Arduino**

Para comenzar con la instalación de una placa Arduino MKRFOX 1200 totalmente nueva. Primero, se utiliza el código 5.39 para obtener las credenciales de esa placa. Que están compuestas por una tupla de valores separados en ID/PAC. El *ID* es un identificador único de cada dispositivo SigFox. Sin embargo, el contrato *PAC* cambia al realizarse el primer registro en la plataforma de activación.

```

void loop() {
    // put your main code here, to run repeatedly:
    SigFox.begin();
    delay(100);
    SigFox.debug();

    String version = SigFox.SigVersion();
    String ID = SigFox.ID();
    String PAC = SigFox.PAC();

    Serial.println("MKRFox1200 Sigfox first configuration");
    Serial.println("SigFox FW version " + version);
    Serial.println("ID = " + ID);
    Serial.println("PAC = " + PAC);
}

```

Figura 5.39: Código Arduino para obtener credenciales Fuente:[4]

Como resultado de la compilación, subida y ejecución de este programa a la placa Arduino se mostrará por la salida correspondiente al monitor un mensaje similar al siguiente 5.40.

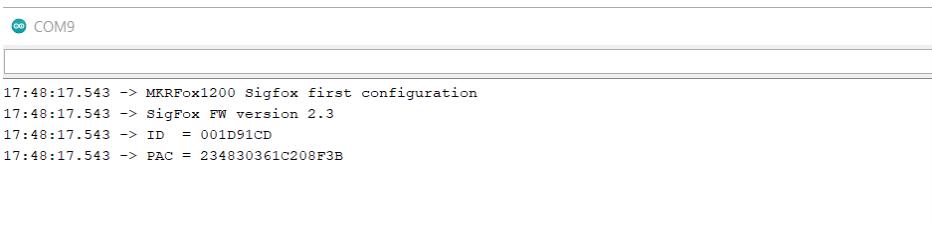


Figura 5.40: Resultado de ejecución código de obtención de credenciales

Utilizando la tupla de valores ID/PAC mostrada en la figura 5.40 se registra el dispositivo en la plataforma de SigFox⁷.

Provide your DevKit's details for identification

Device ID *

Up to 8 numbers and letters (from A to F)

PAC *

Exactly 16 numbers and letters (from A to F)

Figura 5.41: Activación de dispositivo en plataforma SigFox

En este momento, ya tendríamos registrado nuestro dispositivo en la plataforma SigFox con una suscripción de 2 años de envío de datos. A partir de este momento, se procede al montaje y cableado de los componentes del dispositivo Siabox.

Los sensores seleccionados han sido presentados en apartado 4.4 para conocer una descripción de estos y además un listado de características para que conocer cómo está formado el dispositivo Siabox sea mucho más sencillo y transparente. Para representar de manera adecuada

⁷<https://buy.sigfox.com/activate>

la distribución y conexiones de los componentes se ha realizado el siguiente esquema de conexiones.

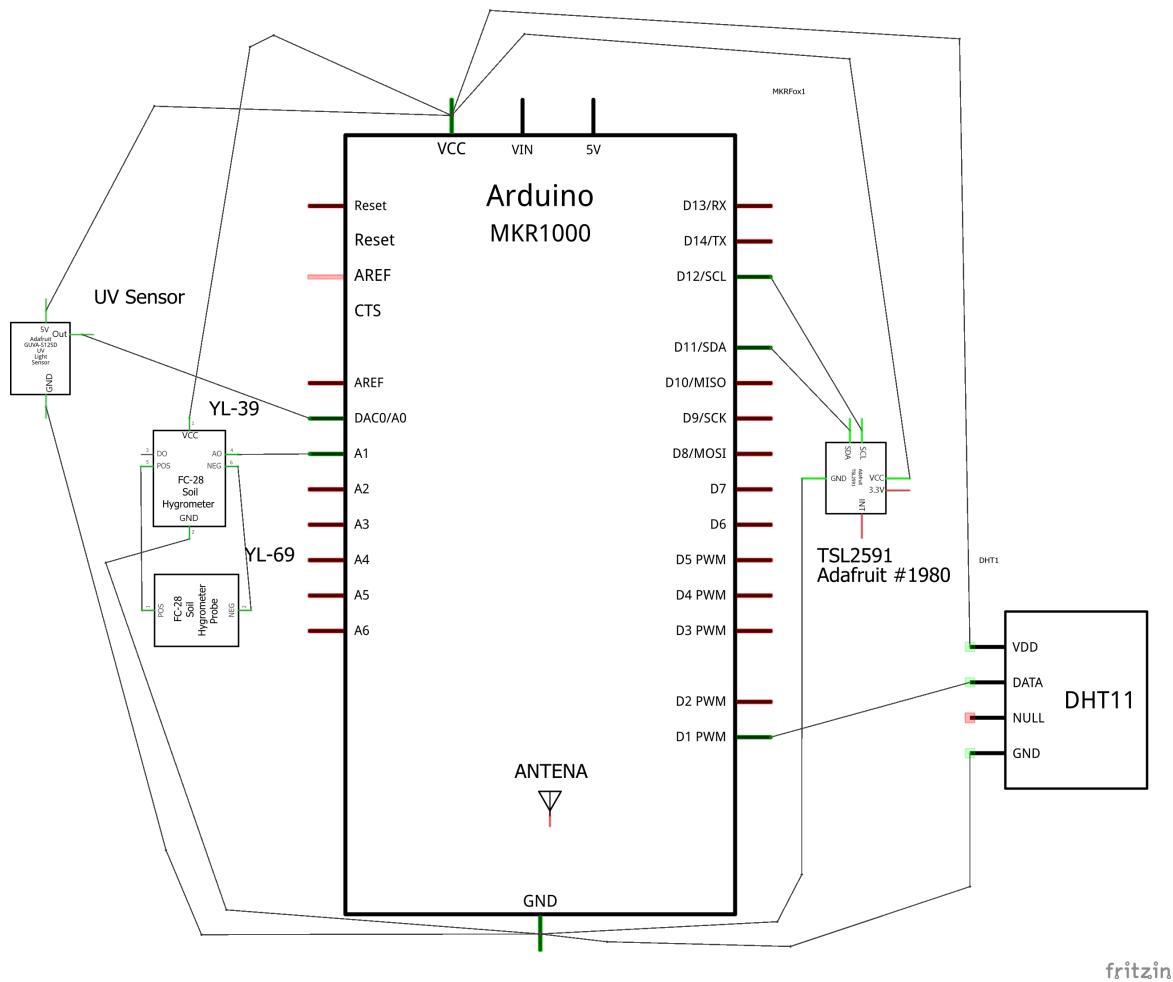


Figura 5.42: Esquema de Conexiones del dispositivo Siabox

Una vez realizadas las conexiones que se muestran en el esquema anterior 5.42 se procede a enclustrar los componentes delicados dentro de una caja de seguridad. Se dota al dispositivo de una fuente de alimentación USB de 5V para un correcto funcionamiento.

Se utiliza una lámina de plástico duro para cubrir la parte superior del dispositivo y así aislar de manera completa los componentes del dispositivo y no interferir en las lecturas realizadas por los sensores de luminosidad y rayos ultravioleta.

■ Desarrollo de recolección de datos

En esta tarea se va a desarrollar el código que se ejecutará en el dispositivo Arduino para obtener los valores leídos por parte de los sensores ambientales y procesar de manera adecuada estos datos leídos para obtener una información en un formato que pueda ser comprendido para cualquier usuario y finalizar con el envío de información mediante la utilización del módulo SigFox que incorpora la placa MKRFOX 1200.

Para seguir una buena metodología de desarrollo se extraen los requisitos funcionales que esta

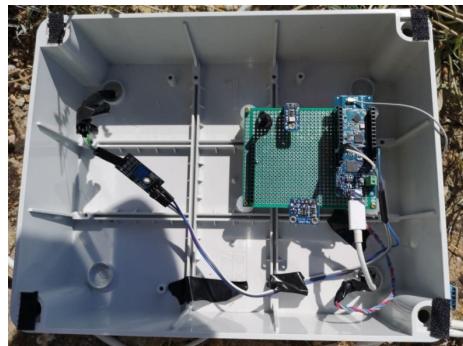


Figura 5.43: Resultado Montaje dispositivo Siabox

parte de la arquitectura necesita y se analizan para establecer las bases del software que se ejecutará en la placa Arduino. En la siguiente figura 5.44 mostramos los requisitos funcionales de este backlog item.



Figura 5.44: Requisitos funcionales definidos para el Sprint 4

A partir del análisis de los requisitos que se ha realizado se realiza un diagrama UML de casos de uso 5.45 con la intención de mostrar estos requisitos y el comportamiento de estos tendrán con el resto de componentes del sistema. Este diagrama nos ayudará a desarrollar el programa para la recolección de valores meteorológicos.

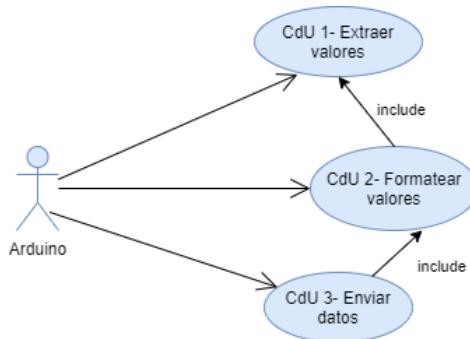


Figura 5.45: Diagrama de Casos de Uso Sprint 4

Observando el diagrama de casos de uso hemos optado por utilizar para el desarrollo de este programa en 3 pasos con la motivación de realizar la obtención de valores de manera ordenada. Primero, en el paso de **extracción** se obtienen las lecturas realizadas en bruto por los sensores ambientales para almacenarlas de manera temporal en la placa Arduino. Estos valores serán **transformados** se le aplicará una normalización para obtener el formato deseado (unidades de medida) de los elementos que forman el mensaje SigFox y se trasmiten mediante la utilización de la pasarela SigFox y la API REST creada en el 5.2 para realizar la **carga** en la base de datos instalada con el servidor Drupal.

Para poder desarrollar este código se ha utilizado el IDE de Arduino y las siguientes librerías que en su mayoría pertenecen a los sensores incorporados.

- **SigFox** utilizada para la manejar el módulo SigFox incorporado en la placa Arduino y así poder crear y enviar mensajes hacia la plataforma SigFox Backend.
- La librería **ArduinoLowPower** permite utilizar las características de bajo consumo del microcontrolador SAMD21 y así poder reducir el consumo de los dispositivos y en el caso de llegar a utilizar baterías alargar la vida de estas.
- **Adafruit_Sensor** perteneciente al fabricante con este mismo nombre Adafruit⁸ que se utilizará para comunicarse con el sensor de luminosidad TSL2591 [4.4.3](#).
- **DHT** como indica su nombre corresponde al sensor ambiental de temperatura y humedad [4.4.4](#) que facilitará la obtención de datos leídos por este sensor.
- **conversions** librería creada para este proyecto con el motivo de convertir valores de tipo Float a Int.

Como resultado de la ejecución del programa implementado en esta tarea se adjunta una captura de salida del monitor serie mostrando la información que ha extraído nuestra placa Arduino de los sensores ambientales, la transformación de parte ella y la carga mediante la pasarela SigFox.

```
COM9
18:16:22.447 -> Humedad Tierra: 467
18:16:26.477 -> 6.45
18:16:26.570 -> 6.45
18:16:26.665 -> 6.45
18:16:26.806 -> 6.45
18:16:26.900 -> 6.45
18:16:26.993 -> 6.45
18:16:27.151 -> 6.45
18:16:27.240 -> 6.45
18:16:27.322 -> 6.45
18:16:27.461 -> 6.45
18:16:27.556 -> Radiación UV (mW/cm^2): 6
18:16:37.021 -> LUMINOSIDAD: 65535
18:16:39.038 -> T: 56.50
18:16:39.038 -> H: 0.00
18:16:39.038 -> Temperatura °C: 56
18:16:39.038 -> Humedad : 0
18:16:41.046 -> Fin
```

Figura 5.46: Captura de salida por monitor serie

5.4.3. Conclusión Sprint 4

Al concluir el Sprint 4 se realiza una reunión con el Product Owner que será representado por el tutor de prácticas para comprobar si el trabajo obtenido con la realización de las tareas ha generado el incremento sobre el producto que se había planificado.

⁸<https://www.adafruit.com/>

Para la realización del primer item del Sprint se estimó la utilización de 20 horas, este plazo ha sido cumplido de manera correcta y no ha generado ninguna desviación.

El segundo item se ha completado en 20 horas lo que implica que se han necesitado 5 horas de las estimadas y por lo tanto, se ha generado un adelanto.

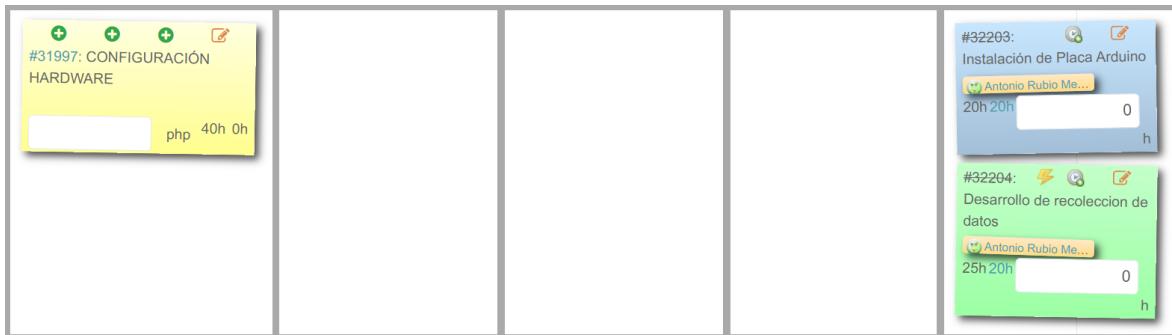


Figura 5.47: Conclusión Sprint 4.Montaje y Configuración Hardware

Al satisfacer el contenido del Sprint 4 se genera un producto que es representado por el dispositivo Siabox y el software que incorpora.

5.5. SPRINT 5.COMUNICACIÓN HARDWARE-PLATAFORMA

En Sprint se utilizará para integrar el componente que capta variables ambientales para su posterior envío a la arquitectura propuesta. Para ello, es necesario utilizar la plataforma SigFox Backend.

5.5.1. Planificación Sprint 5

Se mantiene una reunión con el Product Owner para generar el Product Backlog. De esta reunión se obtiene como resultado la creación de un único item denominado **Configuración Device en SigFox** donde se realizará la configuración del envío de peticiones HTTP hacia el endpoint de recepción de datos de nuestro servidor Drupal desarrollado en el Sprint 5.2.

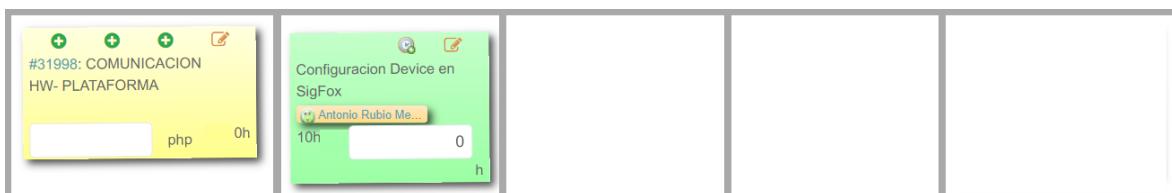


Figura 5.48: Planificación Sprint 5.Comunicación Hardware-Plataforma

5.5.2. Ejecución Sprint 5

Teniendo en cuenta la parte de API REST generada en el Sprint 2 5.2 se configura la comunicación entre el dispositivo hardware correspondiente al Sprint 5.4 y así se logra la integración total de los componentes que forman la arquitectura propuesta. Para ello, desde la sección de *Device Type* de SigFox Backend se utilizará la funcionalidad denominada Callback. Esta funcionalidad se basa en el envío de peticiones HTTP personalizadas desde la plataforma que SigFox ofrece para la monitorización de sus dispositivos hacia un servidor propio.

La configuración se muestra en las siguientes figuras.

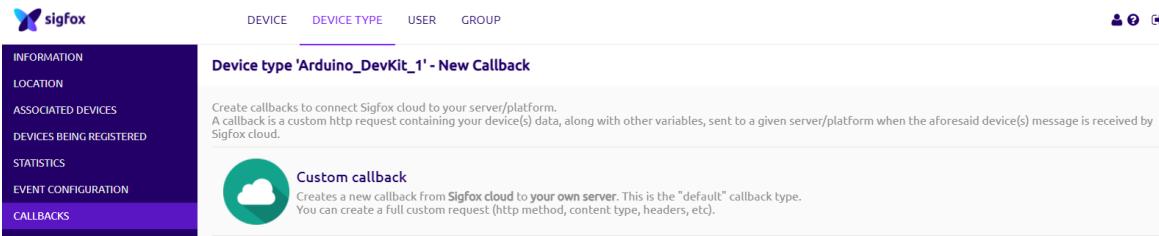


Figura 5.49: Creación de Callback para comunicación con servidor

En la figura 5.50 se selecciona el tipo de callback a utilizar, en este caso será para envío de datos por ello se han seleccionado las opciones *DATA* y *UPLINK*. Para el canal se selecciona la opción *URL* con la intención de enviar el mensaje hacia una dirección web.

En la siguiente sección se declaran una serie de variables que corresponderán con la información del mensaje recibido desde el dispositivo Siabox junto a las variables ofrecidas por la propia plataforma. Además, añadimos el endpoint de la API REST hacia donde se realizará la petición HTTP de tipo *POST* incluyendo en el payload la estructura de información que se desea enviar.

EL payload del contenido de la petición HTTP se ha determinado que application/x-www-urlencoded y contenga el id del dispositivo y los valores ambientales definido.

Al utilizar un servidor que utiliza un certificado SSL para proporcionar una comunicación segura a nuestra arquitectura gracias a la estandarización de cifrado de tráfico entre navegador web y sitio web debido a esto debemos activar la sección de *SSL/TLS*

Figura 5.50: Configuración de Callback para envío de información al servidor

Al finalizar la configuración, toca realizar una de las tareas más complicadas. Ubicar el dispositivo en un entorno donde la cobertura de la red SigFox sea suficiente para establecer conexión. Para ello, hacemos uso del espectro proporcionado por el propio proveedor de servicio en su web⁹.

⁹<https://www.sigfox.com/en/coverage>

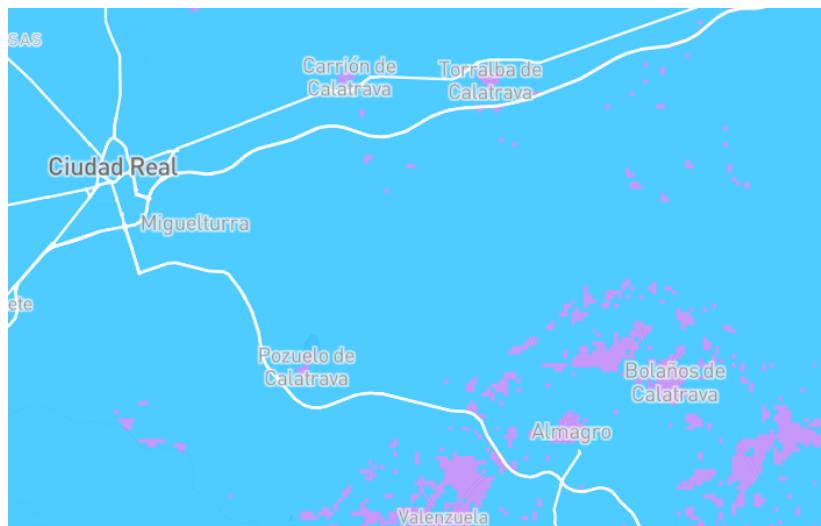


Figura 5.51: Espectro de cobertura de red SigFox Fuente:[2]

5.5.3. Conclusión Sprint 5

La duración del Sprint 5 se ha visto desviada del tiempo estimado en la planificación del mismo debido a los problemas de cobertura encontrados en el taller de montaje. Debido a esta incertidumbre se realizaron cambios de elementos del dispositivo pensando que se trataba de una avería. El dispositivo, una vez encontrada una zona controlada con cobertura completa la integración total de todos los elementos de la arquitectura propuestas. Para evidenciar esto, se adjunta una captura del envío satisfactorio de mensajes hacia la plataforma. Por lo tanto, generamos un incremento sobre el producto aumentando el valor de este.

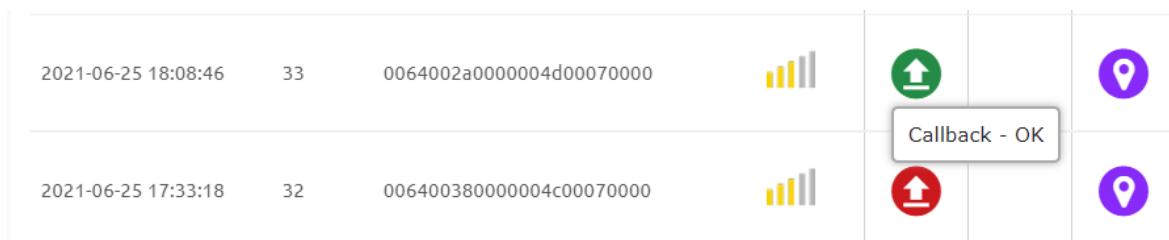


Figura 5.52: Confirmación de envío de Callback de forma adecuada

Después de tener una reunión con el Product Owner se concluye que el Sprint 5 se completa de manera satisfactorio con una desviación de 3 veces el tiempo estimado. Con un tiempo final dedicado de 40 horas de trabajo como se puede apreciar en el estado final del tablero SCRUM 5.53

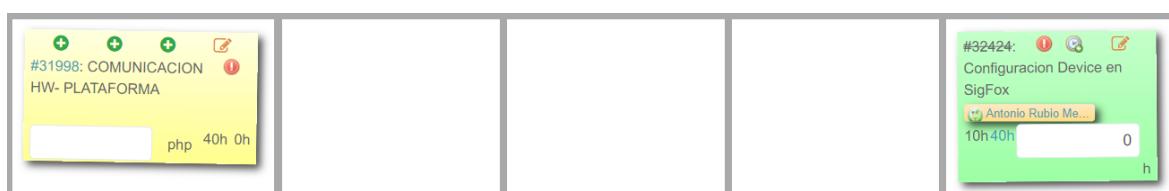


Figura 5.53: Conclusión Sprint 5.Comunicación Hardware-Plataforma

5.6. SPRINT 6.DOCUMENTACIÓN DEL PROYECTO

Para realizar un buen proyecto de ingeniería siempre es necesaria la documentación que ayudará en el futuro a los individuos a enfrentarse a problemas similares a los resultados. Por lo tanto, se genera la documentación para el proyecto de una manera muy peculiar, incluyendo dentro de esta la propia documentación que se entregará como memoria de Trabajo de Fin de Grado adjuntándole un manual de montaje de dispositivo y un manual de despliegue del producto hardware.

5.6.1. Planificación Sprint 6

Se mantiene una reunión con el tutor de prácticas para planificar la generación de la documentación necesaria para abordar la entrega del Trabajo Fin de Grado y a su vez una serie de manuales que sirvan como conocimiento para las posibles mejoras de este proyecto. Para la ejecución de este Sprint de documentación del proyecto se ha generado un Product Backlog de compuesto por 3 items:

- **Manual de Montaje**

Se realiza un manual para recoger el conocimiento sobre el montaje del dispositivo Siabox para una futura utilización. La realización de item se ha estimado en 8 horas de trabajo.

- **Manual de Despliegue**

El propósito de la realización de este manual es estandarizar el procedimiento de despliegue del producto. Para la realización de este item se ha estimado un periodo de 8 horas de trabajo.

- **Estructura del Documento - Memoria TFG**

En este item se realizará la memoria de Trabajo de Fin de Grado para su posterior entrega en la ESI. Para completar la redacción del documento a realizar en esta tarea se ha estimado una duración de 40 horas de trabajo.



Figura 5.54: Planificación Sprint 6. Documentación del Proyecto

5.6.2. Ejecución Sprint 6

- **Manual de Montaje**

Se recoge todo el conocimiento sobre el montaje del dispositivo Siabox en un manual para poder producir estos dispositivos de una manera rápida y efectiva. Para realizar este manual se seguirá el esquema de conexiones 5.3. Esta sección la encontraremos en ANEXOS A.

- **Manual de Despliegue**

Este manual se incluye la metodología para realizar el despliegue del dispositivo Siabox en los entornos de clientes. Una vez completado el manual de montaje del dispositivo Siabox 5.6.2. Esta sección la encontraremos en ANEXOS A.

■ Estructura del Documento - Memoria TFG

La documentación de este TFG se ha dividido en las siguientes partes.

- **Introducción**
- **Objetivos**
- **Antecedentes**
- **Metodologías y Herramientas Utilizadas**
- **Resultados**
- **Conclusión**

5.6.3. Conclusión Sprint 6

Se realiza una reunión con el product owner para aprobar los resultados obtenidos en este Sprint y dar por concluido el trabajo en el proyecto.

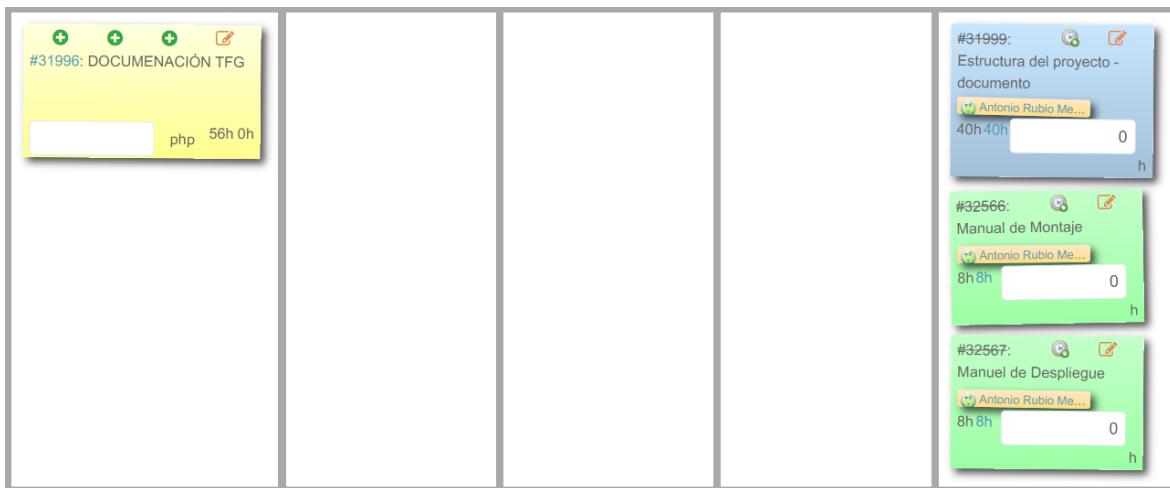


Figura 5.55: Conclusión Sprint 6.Documentación del Proyecto

CAPÍTULO 6

Conclusiones

6.1. JUSTIFICACIÓN DE COMPETENCIAS ADQUIRIDAS

En este apartado se verán las competencias específicas de la intensificación de Tecnologías de la Información y Comunicaciones. Las competencias que se muestran a continuación se han podido cumplir gracias a los conocimientos adquiridos en la intensificación anteriormente nombrada, así como la ayuda recibida por parte de los mentores a académico y laboral.

- **TI1:** *Capacidad para comprender el entorno de una organización y sus necesidades en el ámbito de las tecnologías de la información y las comunicaciones.* Esta competencia se refleja en el hecho de que cuando un estudiante se incorpora como equipo de desarrollo de una organización real, ya puede conocer por sí mismo las necesidades que se experimentan.
- **TI3** *Capacidad para emplear metodologías centradas en el usuario y la organización para el desarrollo, evaluación y gestión de aplicaciones y sistemas basados en tecnologías de la información que aseguren la accesibilidad, ergonomía y usabilidad de los sistemas.* Esta competencia se ve plasmada con la utilización de SCRUM para la gestión del proyecto y un modelo en cascada para trabajar sobre Sprints en particular. .
- **TI6** *Capacidad de concebir sistemas, aplicaciones y servicios basados en tecnologías de red, incluyendo Internet, web, comercio electrónico, multimedia, servicios interactivos y computación móvil.* Desarrollo de una aplicación móvil híbrida soportada por una API REST para la comunicación de esta con el servidor .
- **TI7** *Capacidad para comprender, aplicar y gestionar la garantía y seguridad de los sistemas informáticos.* Utilizando protocolos seguros para la comunicación entre los distintos elementos de la arquitectura y utilizando el protocolo Basic_auth para comunicarnos de manera segura con el servidor Drupal.

Bibliografía

- [1] (), dirección: https://www.sas/es_es/insights/data-management/what-is-etl.html.
- [2] (), dirección: <https://www.sigfox.com/en/coverage>.
- [3] «5pcs DC 3.3-5V 0.1mA UV Prueba Sensor Módulo de interruptor Rayo ultravioleta Sensor Módulo 200-370nm Geekcreit para Arduino - productos que funcionan con placas oficiales Arduino». (), dirección: https://www.banggood.com/es/5pcs-DC-3_3-5V-0_1mA-UV-Test-Sensor-Switch-Module-Ultraviolet-Ray-Sensor-Module-200-370nm-p-1218096.html?utm_source=googleshopping&utm_medium=cpc_organic&gmcCountry=ES&utm_content=minha&utm_campaign=minha-es-es-pc¤cy=EUR&cur_warehouse=CN&createTmp=1&utm_source=googleshopping&utm_medium=cpc_bgs&utm_content=dcr&utm_campaign=dcr-ssc-es-all-0316&ad_id=425803135184&gclid=CjwKCAjw_o-HBhAsEiwANqYhp0zPCBEs9ij83P7ai-6ReBg1GEDYi6uvXK4ETdN-68oBAuPCP-16nBoCQVwQAvD_BwE
- [4] A. Arduino. (2018), dirección: <https://www.aprendiendoarduino.com/2018/03/05/arduino-mkrfox1200/>.
- [5] «Arduino MKR FOX 1200». (), dirección: <https://tienda.bricogEEK.com/arduino-mkr/1128-arduino-mkr-fox-1200.html>.
- [6] «Ciclo de vida de desarrollo de Software». (), dirección: <https://www.efectedigital.online/single-post/2018/04/23/ciclo-de-vida-de-desarrollo-de-software>.
- [7] «Comparativa de tipos de redes en función de ancho de banda y alcance». (), dirección: <https://telos.fundaciontelefonica.com/la-cofa/las-tecnologias-ipwan-un-internet-de-las-cosas-low-cost/>.
- [8] S. Escolar y col., «The PLATINO Experience: A LoRa-based Network of Energy-Harvesting Devices for Smart Farming», en *2019 XXXIV Conference on Design of Circuits and Integrated Systems (DCIS)*, 2019, págs. 1-6. doi: [10.1109/DCIS201949030.2019.8959848](https://doi.org/10.1109/DCIS201949030.2019.8959848).
- [9] M. P. Esteso. «Qué es una API REST y para qué se utiliza». G. Theory, ed. (2020), dirección: <https://geekytheory.com/que-es-una-api-rest-y-para-que-se-utiliza>.
- [10] L. P. Frerire. «¿Hacia dónde lleva la digitalización a la agricultura española?» Interempresas, ed. (2018), dirección: <https://www.interempresas.net/Agricola/Articulos/223249-Hacia-donde-lleva-la-digitalizacion-a-la-agricultura-espanola.html> (visitado 15-06-2021).
- [11] U. Infante. «CONECTANDO UNA APPLICACIÓN DE ANGULAR CON UNA API REST». (2020), dirección: <https://rd.rocktech.mx/publicaciones/entrada/conectando-una-aplicacion-de-angular-con-una-api-rest>.
- [12] «Interpretación: Radiación Ultravioleta (UVI)». (), dirección: <http://www.aemet.es/es/eltiempo/prediccion/radiacionuv/ayuda>.
- [13] K. S. y Jeff Sutherland. «La Guía Definitiva de Scrum: Las Reglas del Juego». (2016), dirección: <https://scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-Spanish.pdf#zoom=100> (visitado 16-06-2021).

- [14] «La digitalización en la agricultura». (2020), dirección: <https://infoagro.com.ar/la-digitalizacion-en-la-agricultura/> (visitado 01-07-2021).
- [15] J. P. y M. d. C. Penadés, *Metodologías ágiles en el desarrollo de software*. Universidad Politécnica de Valencia, 2012.
- [16] «MODULO H1 DHT11 TEMPERATURA Y HUMEDAD PARA ARDUINO». (), dirección: https://www.tiendatec.es/arduino/modulos/627-modulo-h1-dht11-temperatura-y-humedad-para-arduino-8406271480015.html?kk=a4c6365-17a7ec6c71d-8f46e&gclid=CjwKCAjw_o-HBhAsEiwANqYhpVG8YI7M4IFtxMqL_vsA2oVu74zfvACPhYZ6ZTb0bv5eT3vL6Si8xoCP3MQAvD_BwE&utm_source=kelkooes&utm_medium=cpc&utm_campaign=kelkooclick&utm_term=tiendatec+MODULO+H1+DHT11+TEMPERATURA+Y+&from=kelkooes.
- [17] D. K. Schweichhart. «Reference Architectural Model Industrie 4.0 (RAMI 4.0)». (2020), dirección: https://ec.europa.eu/futurium/en/system/files/ged/a2-schweichhart-reference_architectural_model_industrie_4.0_rami_4.0.pdf (visitado 01-07-2021).
- [18] «Sensor de Humedad de Suelo Modelo YL-38 y Sonda YL-69». (), dirección: <https://maxelectronica.cl/temperatura-y-humedad/44-sensor-de-humedad-de-suelo-modelo-yl-38-y-sonda-yl-69.html>.
- [19] «Sensor de luz digital de alto rango dinámico TSL2591 de Adafruit». (), dirección: <https://www.melopero.com/es/tienda/sensori/luz-y-color/sensor-de-luz-digital-de-alto-rango-din%C3%A1mico-adafruit-tsl2591/>.
- [20] C. Team. (2020), dirección: <https://www.chakray.com/es/swagger-y-swagger-ui-por-que-es-imprescindible-para-tus-apis/>.

ANEXOS

ANEXO A

Manual de Montaje

En este Anexo se incluye un manual para el montaje de los dispositivos Siabox.

1. Ubicación de los componentes sobre PCB

Se plantea la ubicación de los sensores que no tienen contacto directo con el ambiente y que se ubicarán dentro del dispositivo Siabox.

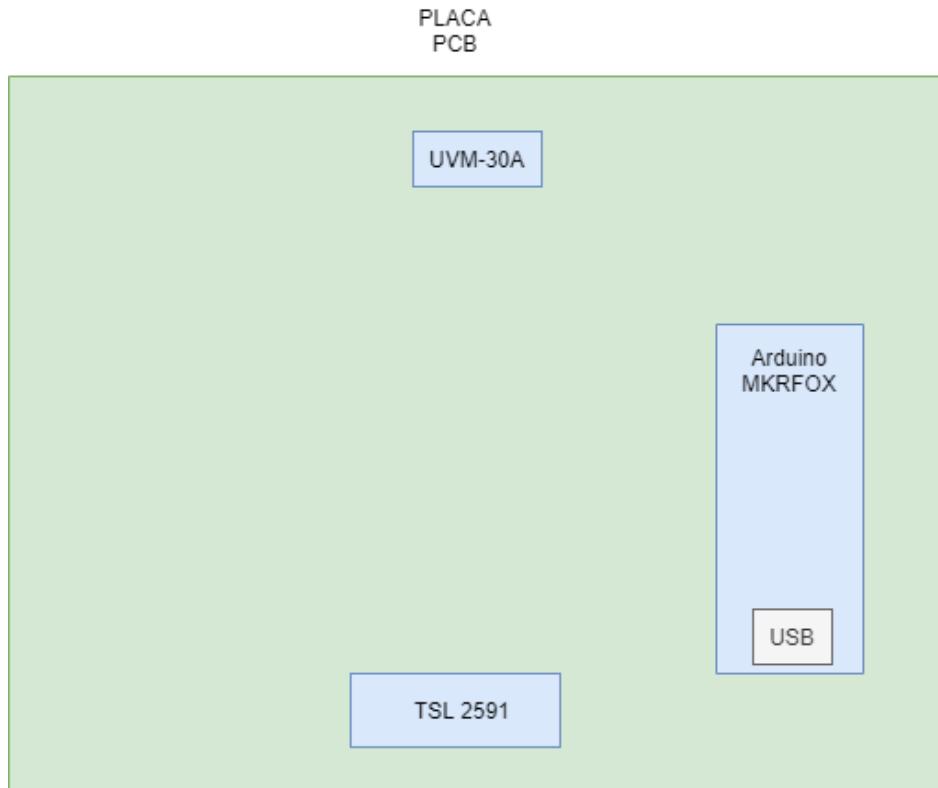


Figura A.1: Distribución de elementos sobre PCB

2. Preparación de la caja para cableado exterior

Se realizan 2 agujeros con un taladro en ambos extremos de la caja de seguridad estanca de exterior IP65 que nos servirán para pasar los cables del exterior a la parte de abajo del PCB.

3. Conexión de componentes de interior del dispositivo

Se realiza la conexión de los componentes a la placa PCB y el cableado entre estos componentes. Estas conexiones se realizaran siguiendo el esquema de referencia.

Para conectar sensor YL69/YL39 es necesario que la parte que va anclada a tierra este fuera de



Figura A.2: Distribución de Orificios en Caja de Seguridad

la caja de seguridad. Se pasarían los cables por el orificio 1 de esquema de distribución A.2 y se realizaría dentro la conexión. Además, se recomienda que los pines que estan en contacto con la tierra se cubriesen con pegamento termo-fusible para asegurar su durabilidad.

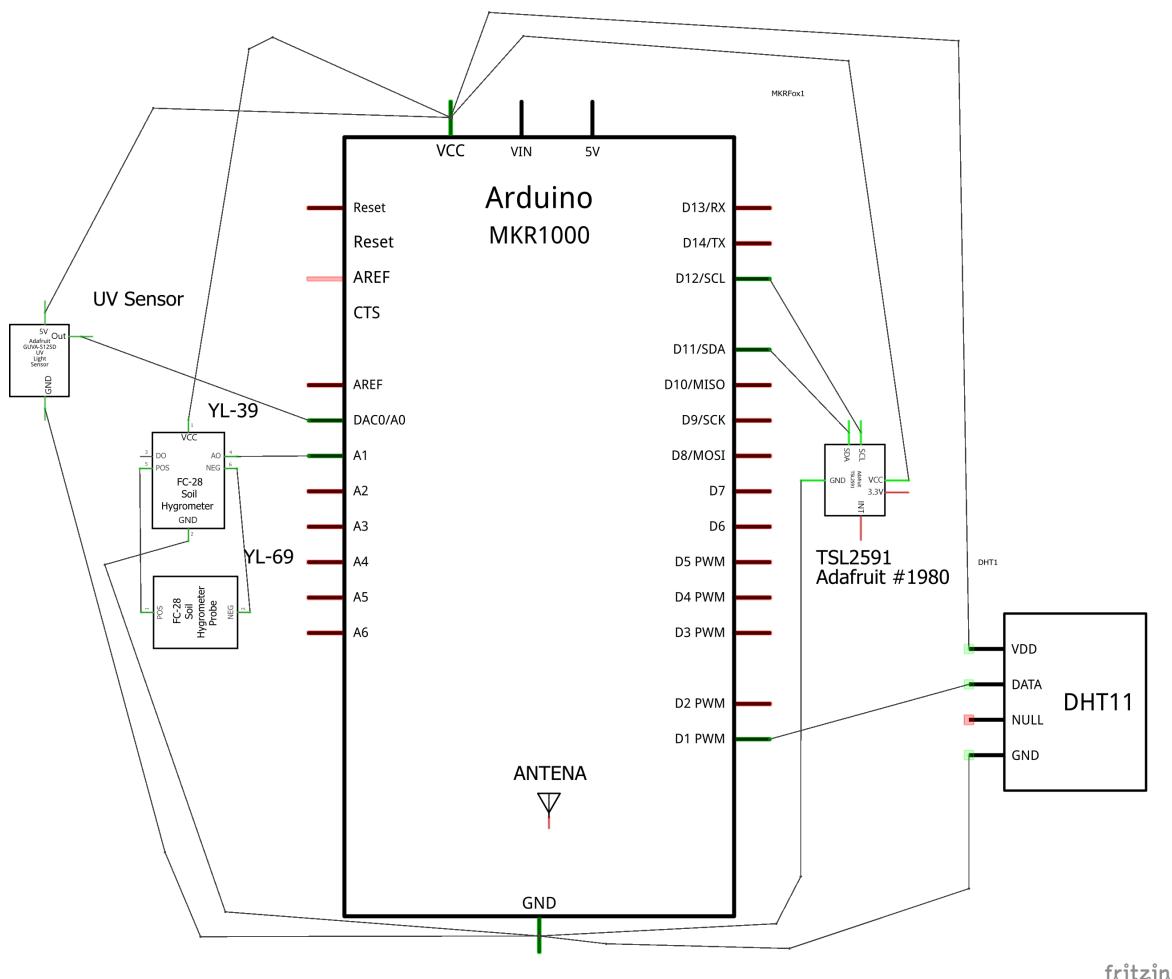


Figura A.3: Esquema de Conexiones del dispositivo Siabox

4. Enclaustrar la estación de procesamiento en caja estanca

Se colocan los componentes dentro de la caja de protección exterior IP65 para su protección para evitar que haya contacto con la caja colocamos unos tacos de goma y con unos tornillos anclamos la PCB. Debemos de asegurarnos que los cables se canalizan por los orificios que de la imagen A.2 para los sensores que se colocan fuera de la caja estanca. Para finalizar, colocamos en la parte superior de la caja la lamina de plástico cristalizado para "sellár" el dispositivo.

5. Colocación Antena y sensor DHT11

El sensor DHT11 utilizará el orificio 2 de A.2 para salir al exterior y se pegará a la caja estanca como en la imagen A.4. Por otro lado, la antena GSM se conectara al dispositivo Arduino y el cable se pondrá entre el cristal y la caja. Para anclar la antena a la caja se hará uso de la cinta adhesiva que esta incorpora.



Figura A.4: Ubicación de sensores

ANEXO B

Manual de Despliegue

En este anexo se incluirá un manual de despliegue de producto Siabox en un entorno.

Para poder desplegar el dispositivo Siabox necesitamos seguir los siguientes pasos:

1. Incorporar fuente de alimentación con dos alternativas adaptador USB o incorporar una batería.

Conectamos el dispositivo Siabox a la fuente de alimentación que en este caso será mediante un adaptador USB de 5V. En este caso, esperamos al despliegue total para conectar a la red eléctrica nuestro dispositivo.

2. Colocación de caja estanca.

Buscamos una ubicación donde no interceda ninguna elemento en la iluminación que reciba nuestro dispositivo para no generar datos engañosos y dejamos en este lugar la caja del dispositivo. Siempre recordar que tenemos que tener la zona de humedad a la distancia adecuada.



Figura B.1: Despliegue Dispositivo

3. Despliegue de sensor de humedad en tierra.



Figura B.2: Despliegue Sensor Humedad

Seleccionamos una zona de riego donde se pueda controlar la humedad en sustrato y clavamos el sensor YL69 lo protegemos de posibles roturas por factor humano enterrando un poco este

sensor como se muestra en la imagen B.2. Recordar que tenemos una distancia máxima que viene limitada por el tamaño del cable que tiene el sensor. Por lo tanto, en caso de tener que mover algún componente, será la caja.

4. Activamos corriente eléctrica.

Activamos entrada de corriente eléctrica al dispositivo para que comience su funcionamiento.

ANEXO C

Estructura de Módulos

En este anexo incluimos las estructuras de directorio que tendrán los módulos Drupal.

C.1. MODULO SIGFOXREST

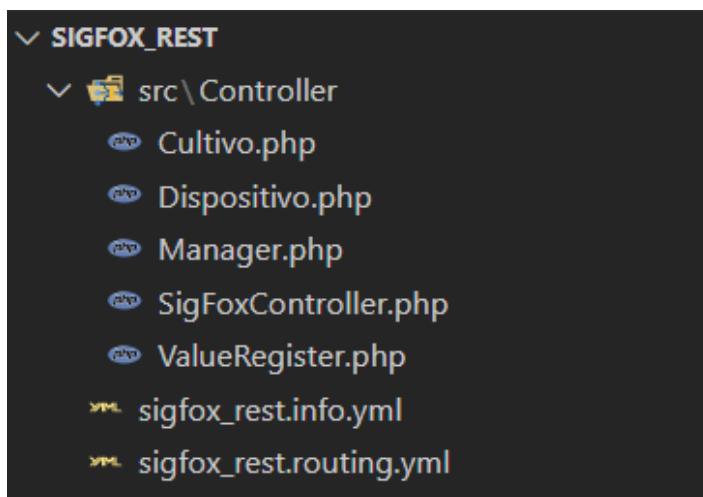


Figura C.1: Estructura de Ficheros de módulo personalizado SigFox_REST

Para describir la estructura que tendrá el directorio donde ubicaremos nuestro módulo. Comenzaremos por los archivos con extensión .yml que están destinados a la descripción y configuración del módulo y finalizaremos con la estructura de clases que forman parte del directorio **src/Controller** donde se encuentra el código fuente que le aporta funcionalidad nuestra API REST en la dirección especificada.

- **sigfox_rest.info.yml**

Este fichero contiene la información que Drupal en su gestor de instalación de módulos interpretará para darlo a conocer. Podemos apreciar en la figura C.2 que se incorpora el nombre, descripción y paquete al que pertenece el modulo. Además, el tipo y la versión de Drupal con las que es compatible.

```

name: SigFox Rest
description: Modulo para la obtención de datos de SigFox.
package: PulsiaForte

type: module
core: 8.x

```

Figura C.2: Fichero de información módulo SigFox_REST

- **sigfox_rest.routing.yml**

En este fichero se configura el comportamiento que va a tener el módulo con respecto a la instalación de Drupal. En la siguiente figura C.3 se puede apreciar el endpoint donde serán tratadas las peticiones, el controlador y su ubicación dentro de la estructura de directorios. Por otro lado, se encuentran los tipos de peticiones HTTP que serán aceptados por el módulo generar respuesta y el sistema de seguridad que se utilizará.

```

sigfox_rest.content:
  path: '/postSigFoxData'
  defaults:
    _controller: '\Drupal\sigfox_rest\Controller\SigFoxController::apiSigFoxData'
    _title: 'sigFox REST'
  methods:
    - POST
  requirements:
    _permission: 'access content'
    _access: 'TRUE'

```

Figura C.3: Fichero de configuración módulo SigFox_REST

- **Directorio src/Controller**

En este directorio se ubica el código fuente del módulo de Drupal SigFox REST dedicado al almacenamiento de registros, comprobación de predicciones y generación de alarmas. Este directorio se irá ampliando a medida que se vayan completando los Items que forman el Backlog del Sprint.

ControllerBase

Esta clase es un controlador que tiene acceso a una gran cantidad de métodos de utilidad y al contenido. Esta clase está destinada a un uso sea exclusivo por parte de clases controlador que funcionan como pegamento para las clases con código más triviales.

SigFoxRestController

En este fichero se desarrollará el controlador específico que heredará del Controlador Base para comunicarnos con la instalación de Drupal en el momento. En esta clase gracias a librería **Symphony** tendremos la posibilidad de recibir la información que es enviada en la petición que se realiza sobre el endpoint `/postSigFoxData` y a partir de ahí realizar la comprobación de autenticidad del dispositivo y almacenamiento de esta información.

Manager

Esta clase, será utilizada para extraer información de la base de datos, con esto, se buscará reducir tanto la complejidad como la limpieza del módulo principal. Buscando crear un código lo más limpio y escalable para posibles mejoras y ampliaciones de funcionalidad.

Dispositivo

Clase se destinará a albergar el conocimiento que utilizaremos para tratar con la información referente a los Dispositivos Siabox y así poder realizar comprobaciones y extracción de información de nuestra base de datos de manera más limpia.

ValueRegister

Esta clase conocimiento se utilizará para tratar con la información capturada por parte del dispositivo Siabox que es enviada con Callbacks de SigFox Backend hacia nuestro servidor para su almacenamiento y posterior almacenamiento, así evitamos el uso abusivo de lista de elementos primitivos.

En el segundo item del Sprint Backlog del [5.2](#) se amplía la información de la estructura del directorio.

Comenzamos hablando de la estructura de archivos que será una ampliación de la realizada en el item anterior [C.1](#). Por lo tanto, se muestra de manera detallada cuales han sido los nuevos componentes añadidos a esta solución [5.2.2](#).

- **Directorio src/Controller**

En este directorio encontramos el código fuente del módulo personalizado de Drupal SigFox REST dedicado al almacenamiento de registros, comprobación de predicciones y generación de alarmas. Este directorio se amplia en el segundo Item del Sprint 2 y como resultado de este trabajo podemos apreciar la figura [E.2](#).

SigFoxRestController

Continuando con el desarrollo de este controlador incorporaremos las funciones necesarias para realizar una petición de datos al servicio de AEMET, depurar esa información para un procesarla y sacar resultados mediante la comparación de esta información almacenada en Valor Sensores. En caso de detectarse una anomalía fuera de los parámetros establecidos, se procederá la creación de un contenido *Alarma Sensor* correspondiente a esa lectura.

Cultivo

Esta nueva clase se utilizará para albergar el conocimiento que existe en nuestra plataforma sobre los cultivos donde están emplazados nuestros dispositivos Siabox. Este conocimiento nos ayudará a la obtención de información de AEMET.

Manager

Sobre esta clase se realizará un incremento de funcionalidad incorporando varias funciones para extraer información almacenada en nuestra base de datos de contenidos.

C.2. MODULO IONICREST

La estructura del módulo [C.4](#) donde se define e implementa la segunda parte de nuestra API REST

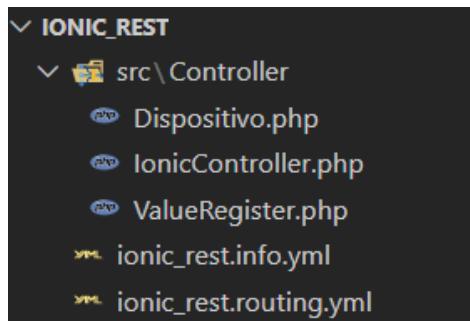


Figura C.4: Estructura de módulo Ionic REST

- **ionic_rest.info.yml**

Fichero donde se especifica la información del módulo. Esta información será interpretada por la instalación de Drupal en la sección de *Módulos*. De los datos mostrados en la siguiente imagen C.5 caben destacar el nombre, la descripción del módulo, paquete al que pertenece y la versión de Drupal compatible que podemos apreciar en la etiqueta *core*.

```
name: Ionic REST
description: Modulo para proveer de informacion a la aplicacion movil.
package: PulsiaForte

type: module
core: 8.x
```

Figura C.5: Fichero de información de Ionic REST

- **ionic_rest.routing.yml**

Este archivo contiene la información de configuración del módulo personalizado Ionic REST. Para una mejor comprensión se añade la imagen C.6, donde se muestra la dirección donde se puede acceder al módulo, el tipo de autentificación, la configuración del controlador y el role necesario por usuario para poder solicitar la información.

```
ionicRest.content:
  path: '/ionicRest'
  options:
    _auth: [ 'basic_auth' ]
  defaults:
    _controller: '\Drupal\ionic_rest\Controller\IonicController::apiIonicData'
    _title: 'Ionic REST'
  methods:
    [GET]
  requirements:
    _role: 'authenticated'
```

Figura C.6: Fichero de configuración de Ionic REST

- **Directorio src/Controller**

En este directorio encontraremos todo el código relacionado con la lógica del módulo personalizado. En los siguientes apartados se desmenuzará la funcionalidad que aporta cada una de las clases y funciones a nuestra API REST.

IonicController

En esta clase se desarrolla un controlador que heredará las funcionalidades del ControllerBase que proporciona Drupal. En esta clase se incluye la función que recibe la información de la petición HTTP gracias a *Symphony*, interpretará el usuario que está solicitando esta información y el tipo de información. Una vez que la información sea recuperada y formateada de manera adecuada, se responderá a la petición que ha desencadenado esta ejecución. Para enviar la información se utilizará *Symphony* y en particular el objeto JsonResponse.

Dispositivo

Esta clase será utilizada como una clase conocimiento, los objetos de esta clase apoyarán en el desarrollo de la clase principal IonicController. Los objetos de esta clase contendrán la información relativa a los dispositivos SiaBox que de los que se solicite información.

ValueRegister

El contenido de esta clase se utiliza para crear una clase que albergue el conocimiento de los valores leídos por los dispositivos Siabox que se recuperaren de la persistencia del sis-

tema. Esto apoyará al desarrollo de la clase más relevante de este módulo que es IonicController.

ANEXO D

Diagrama de Gantt

En este anexo incluimos el diagrama de Gantt y una tabla con el tiempo estimado y tiempo dedicado.

En esta planificación se ha considerado que se trabajará en una jornada de 8 horas de manera flexible de lunes a viernes. La duración total de la realización del proyecto se extiende desde la segunda semana de Abril de 2021 hasta la tercera semana de Junio 2021.

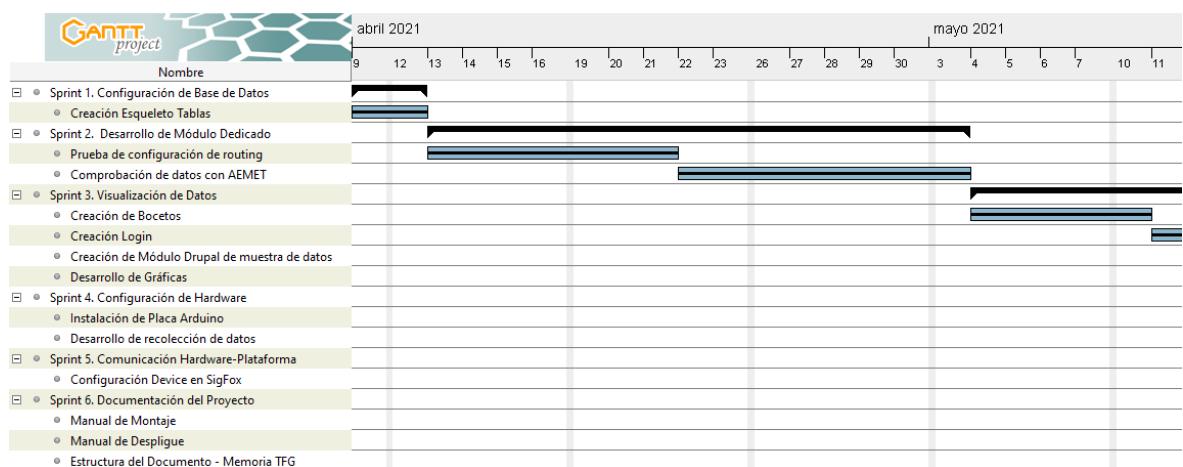


Figura D.1: Diagrama de Gantt

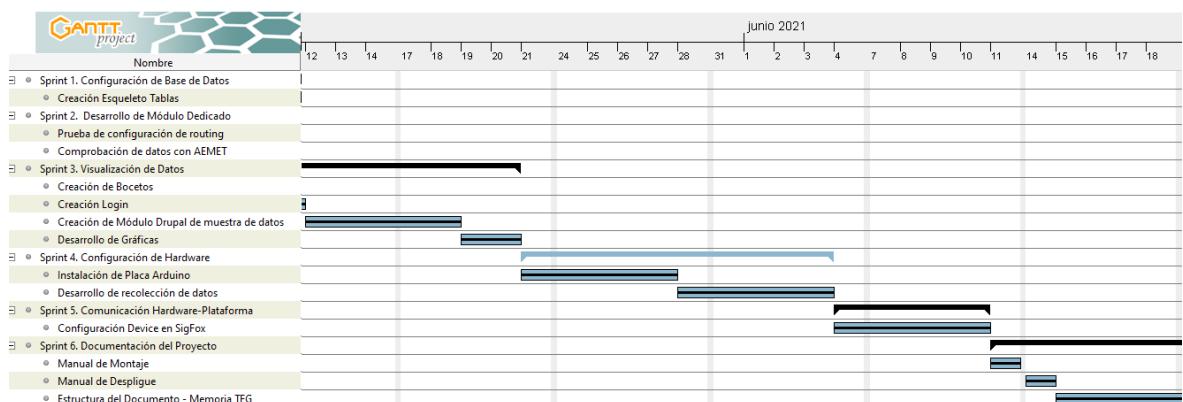


Figura D.2: Diagrama de Gantt

D.0.1. Tablas Estimación/Duración

Tarea	Tiempo Estimado	Tiempo Dedicado
Sprint 1. Configuración base de datos	16h	16h
S1 Creación Esqueleto Tablas	16h	16h
Sprint 2. Desarrollo de módulo dedicado	40h	120h
S2 Prueba de configuración de routing	10h	60h
S2 Comprobación de datos con AEMET	30h	60h
Sprint 3. Visualización de Datos	96h	104h
S3 Creación Bocetos	38h	30h
S3 Creación Login	10h	10h
S3 Creación de Modulo Drupal para muestra de datos	40h	40h
S3 Desarrollo de Gráficas	16h	16h
Sprint 4. Configuración Hardware	45h	40h
S4 Instalación de Placa Arduino	20h	20h
S4 Desarrollo de recolección de datos	25h	20h
Sprint 5. Comunicación HW-Plataforma	10h	40h
S5 Configuración Device en SigFox		
Sprint 6. Documentación del proyecto	56h	56h
S6 Manual de Montaje	8h	8h
S6 Manual de Despliegue	8h	8h
S6 Estructura del Documento - Memoria TFG	40h	40h
TOTAL	263h	376h

Tabla D.1: Tabla de Estimación vs Duración Real

ANEXO E

Diagramas UML de Clases

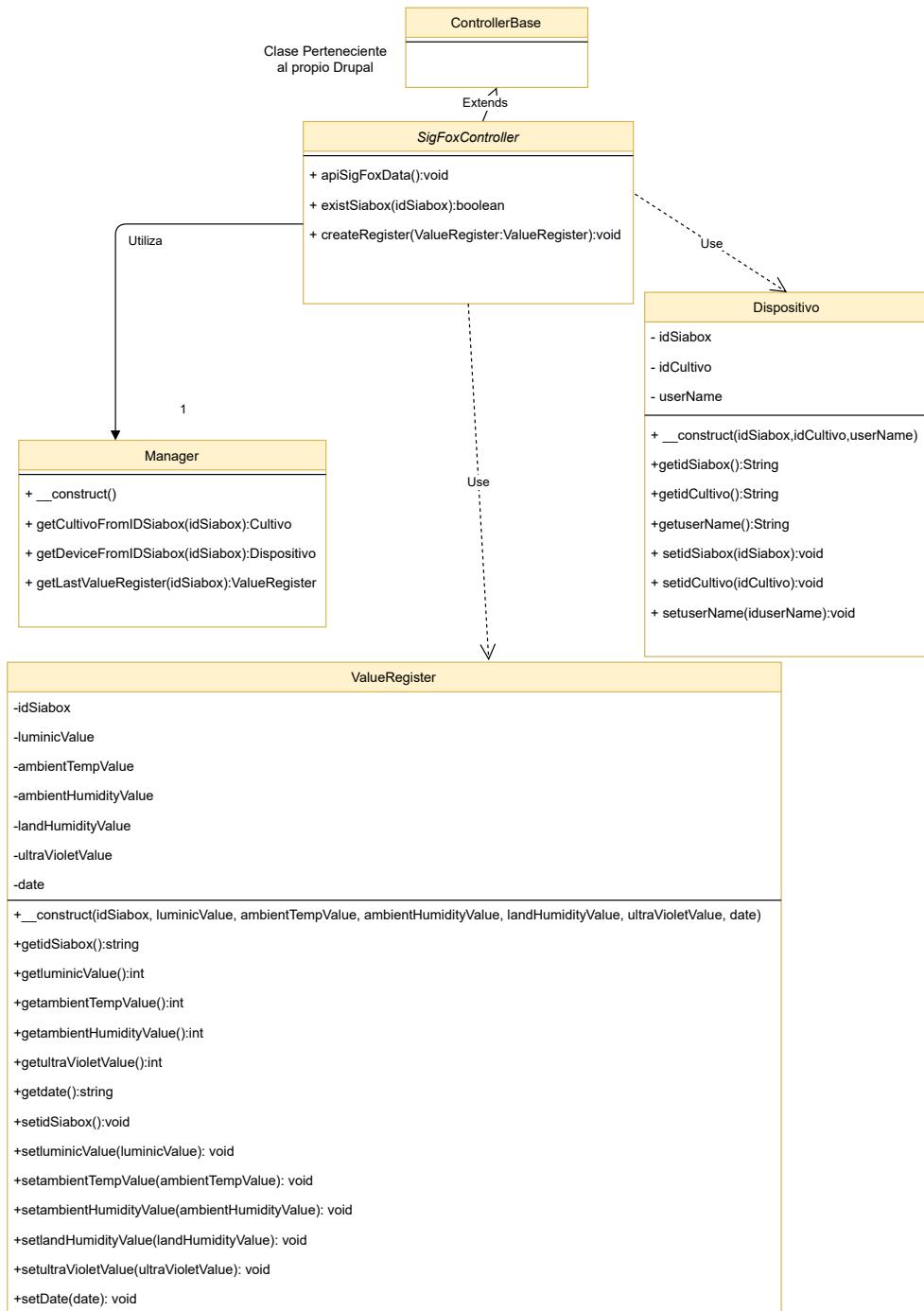


Figura E.1: Diagrama UML de clases del módulo Sigfox_REST Prueba de configuración de routing

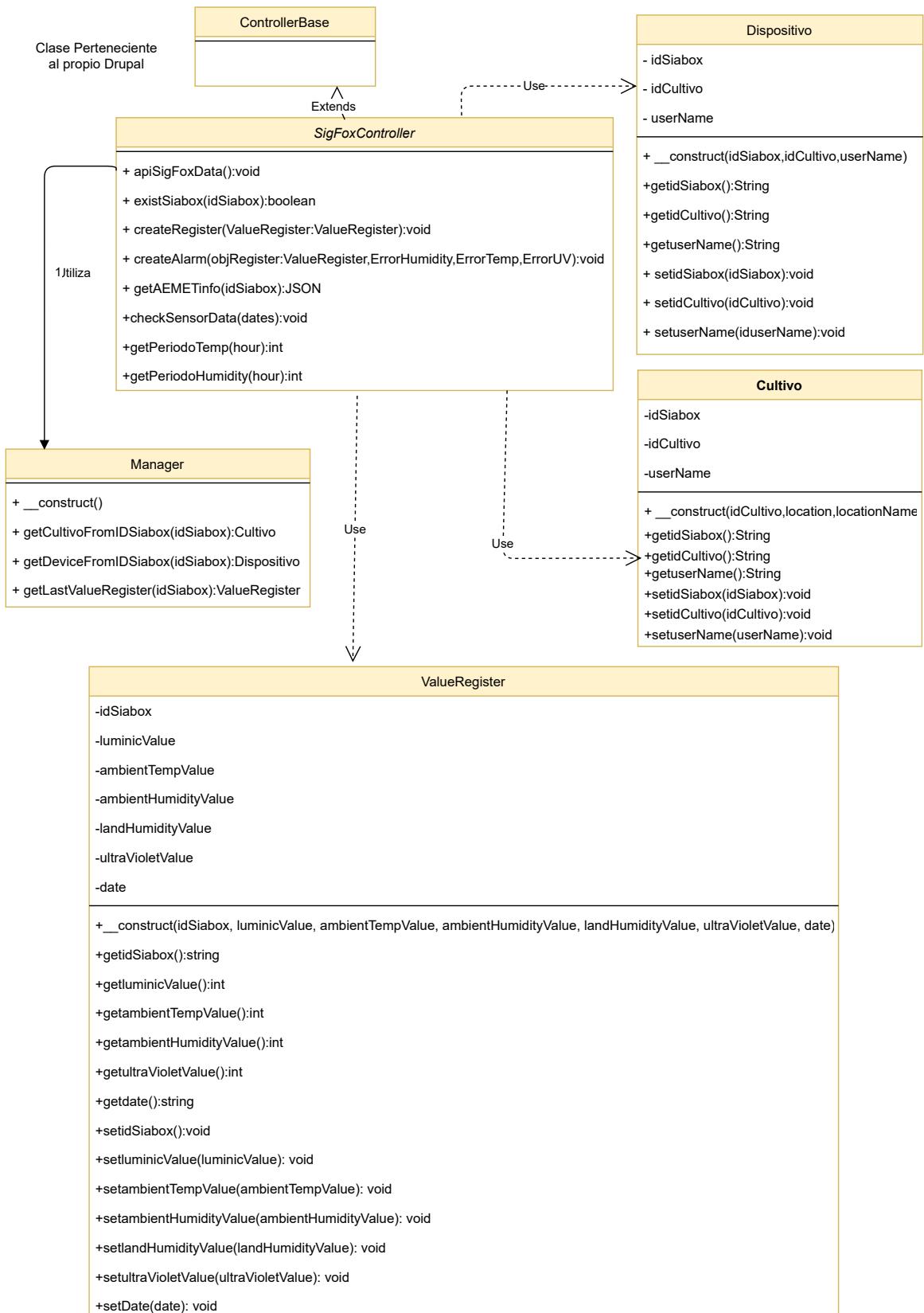


Figura E.2: Diagrama UML de clases Comprobación de datos con AEMET

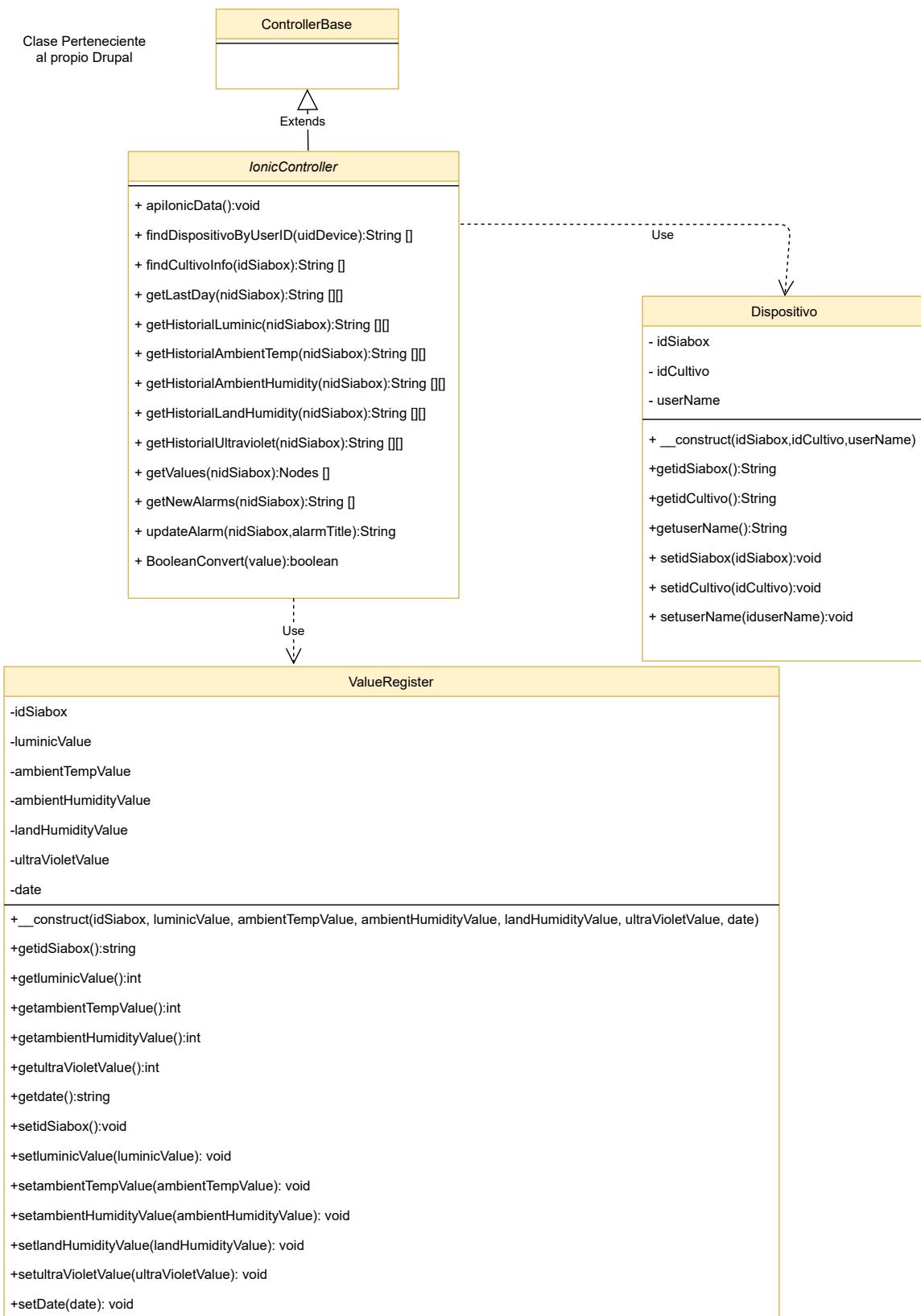


Figura E.3: Diagrama UML de clases Ionic REST (Parte Servidor)

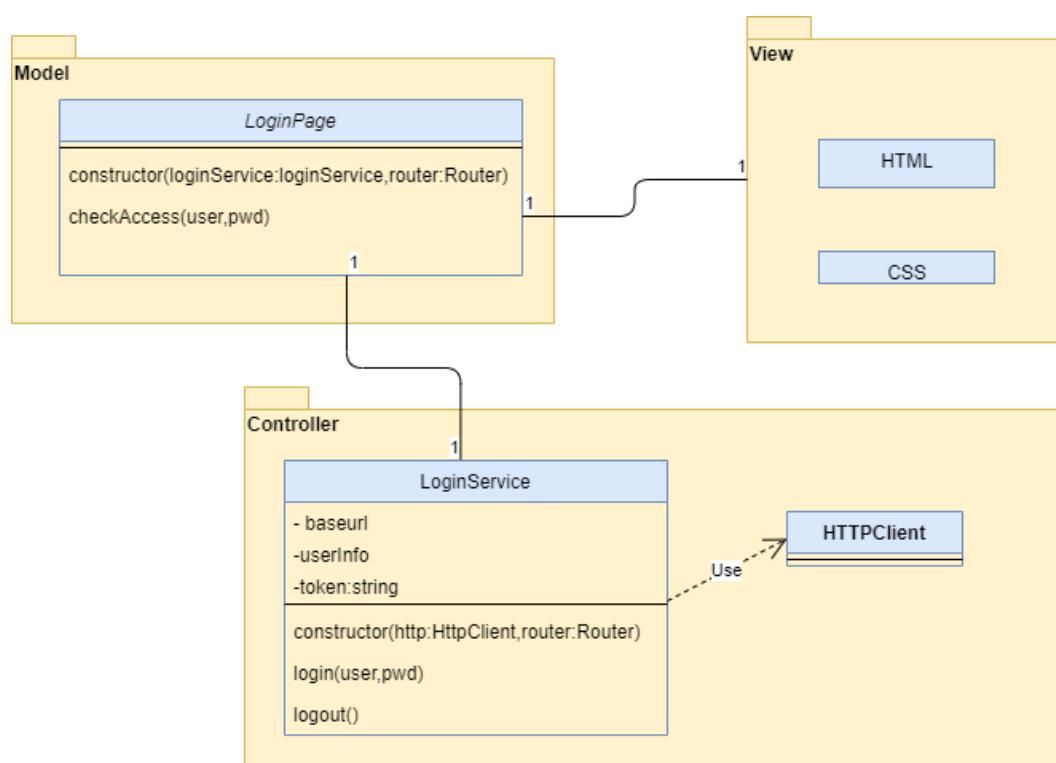


Figura E.4: Diagrama UML de clases de Login (Parte Cliente)

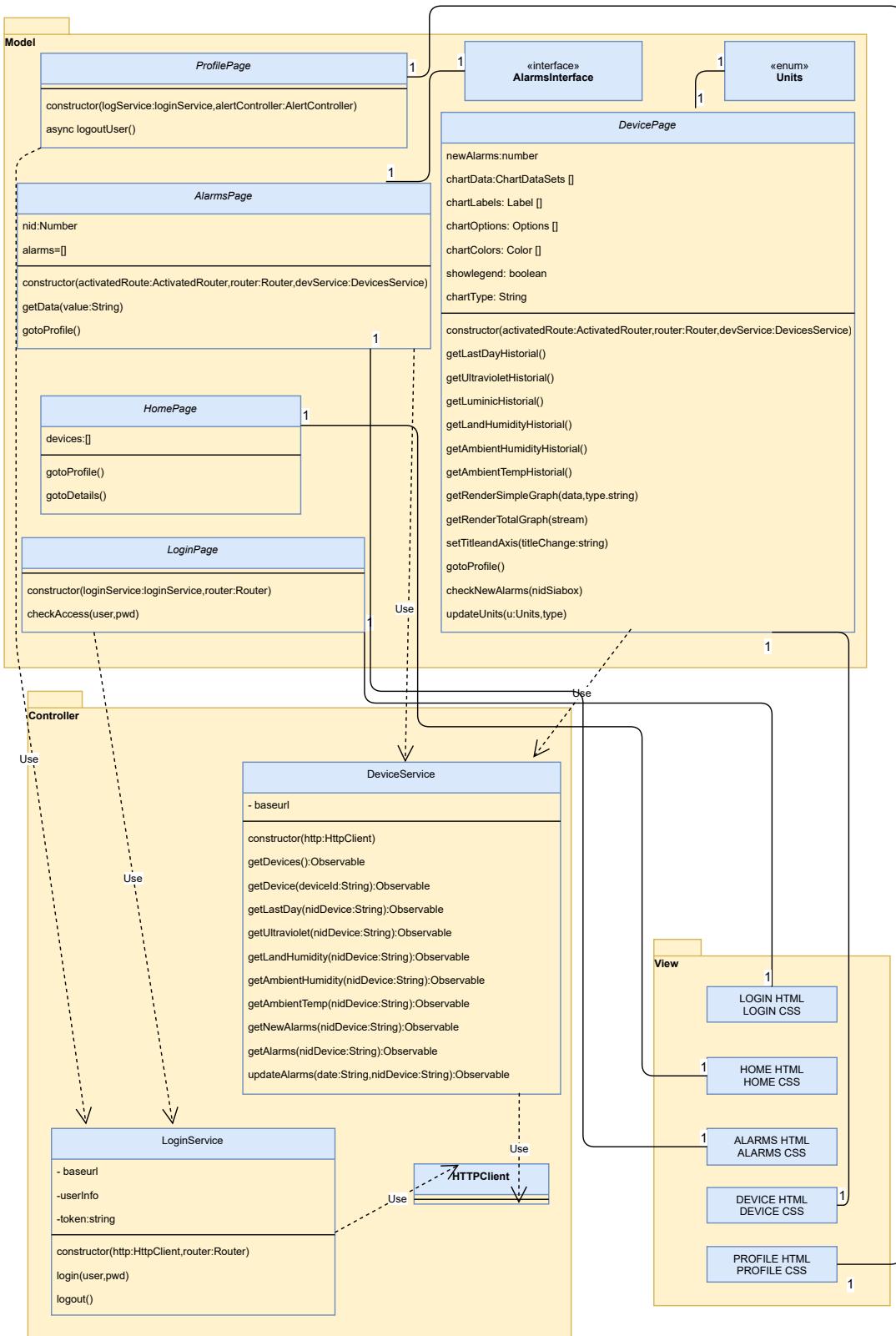


Figura E.5: Diagrama UML de Clases App Movil

ANEXO F

Dirección Pruebas App Móvil

En este anexo incluimos la dirección de la ubicación donde se encontrará la aplicación móvil en caso de querer probar sus funcionalidades.

https://pruebasaluuclm-my.sharepoint.com/personal/antonio_rubio5_alu_uclm_es/Documents/TFGPrueba