

Sistema de Irrigación Inteligente con IoT y Predicción para Agricultura Sostenible

Intelligent Irrigation System with IoT and Prediction for Sustainable Agriculture

Trabajo de Fin de Grado de Ingeniería Informática



**VNiVERSiDAD
D SALAMANCA**

Julio de 2024

Autor:

Diego Plata Klingler

Tutores:

Sergio Alonso Rollán

Javier Prieto Tejedor

Certificado de los tutores TFG

D. Sergio Alonso Rollán y D. Javier Prieto Tejedor profesor/a del
Departamento de Informática y Automática de la Universidad de
Salamanca,

HACE/N CONSTAR:

Que el trabajo titulado “Sistema de Irrigación Inteligente con IoT y Predicción para
Agricultura Sostenible”, que se presenta, ha sido realizado por Diego Plata
Klingler, con DNI *****8 3 8 6 F y constituye la memoria del trabajo realizado
para la superación de la asignatura Trabajo de Fin de Grado en Ingeniería
Informática en esta Universidad.

Salamanca, 4 de Julio de 2024

Fdo.: _____

RESUMEN

El uso de agua para riego en España ha mostrado en los últimos años un aumento importante. Según datos del Instituto Nacional de Estadística (INE)[1], en 2018, el volumen total de agua utilizada en el sector agrario aumentó un 3.7% respecto a 2016, alcanzando los 15,495 hectómetros cúbicos.

Además, el cambio climático está alterando los patrones de lluvia, resultando en un aumento de sequías y déficits de agua que representan un riesgo significativo para la vida y la nutrición. Esto aumenta la necesidad de usar agua en la agricultura, haciendo que se apliquen cantidades excesivas de agua debido a la poca eficiencia de los sistemas de riego. [2]

En respuesta a este problema, este Trabajo de Fin de Grado se centra en el desarrollo de un prototipo de sistema de riego inteligente basado en tecnologías de Internet de las Cosas (IoT). El objetivo principal es diseñar una solución que optimice el uso del agua, aumentando la gestión eficiente de los recursos y a la sostenibilidad. Para ello se utilizarán sensores que recopilarán datos del entorno, como la temperatura y humedad ambiente, o la humedad del suelo. Esto monitorizará la zona de estudio y, mediante una planificación de riego, aumentar la eficiencia en el uso del agua.

Tras la realización de este proyecto se ha conseguido diseñar y construir un dispositivo IoT que permita la monitorización de las variables más importantes en lo que al crecimiento de la planta respecta. Además, se ha desarrollado una plataforma web que permite la visualización de los datos de manera eficiente y sencilla para el usuario. Por último, realiza una planificación de riego que evita el desperdicio de agua.

Palabras clave: IoT, Agricultura inteligente, sensores y monitorización

ABSTRACT

The use of water for irrigation in Spain has shown a significant increase in recent years. According to data from the National Institute of Statistics (INE) and the Ministry for the Ecological Transition and the Demographic Challenge (MITECO), in 2018, the total volume of water used in the agricultural sector increased by 3.7% compared to 2016, reaching 15,495 cubic hectometers.

Also, climate change is altering rain patterns, resulting in increased droughts and lack of water, which means a big risk to life and nutrition. This increases the need to use water in agriculture, leading to excessive amounts being applied due to the low efficiency of irrigation systems.

To solve this problem, this Final Degree Project focuses on developing a prototype of an intelligent irrigation system based on Internet of Things (IoT) technologies. The main objective is to design a solution that optimizes water use, improving resource management and sustainability. Sensors will be used to collect environmental data, such as temperature, humidity, and soil moisture. This will monitor the study area and, with an irrigation planning, increase water use efficiency.

As a result of this project, an IoT device has been designed and built that allows the monitoring of the most important variables for plant growth. Additionally, a web platform has been created that allows efficient and simple visualization of the data for the user. Finally, it provides an irrigation plan that prevents water waste.

Keywords: IoT, Intelligent Agriculture, sensors y monitoring

Contenido

1.	Introducción	13
2.	Estado del arte.....	15
2.1	Unidades estándar de las variables monitorizadas, así como su valor óptimo ...	16
	Investigación de técnicas de monitorización y cuidado automático de las plantas ...	17
3.	Análisis del mercado.....	18
3.1.	Rachio Smart Sprinkler Controller:.....	18
3.2.	Orbit B-hyve Smart Hose Faucet Timer:	19
3.3.	Datalogger Novagric SYNION:	19
3.4.	Plantae Sensor Inteligente de Humedad, Conductividad y Temperatura:	19
4.	Objetivos	21
4.1.	Objetivos funcionales.	21
4.2.	Objetivos Personales.	22
5.	Conceptos teóricos.....	23
5.1.	Frontend	23
5.2.	Backend.....	23
5.3.	Arquitectura Cliente-Servidor	23
5.4.	MQTT	24
5.5.	MONGODB	24
5.6.	Docker.....	25
5.7.	Internet of Things	26
5.8.	Impresión 3D.....	27
6.	Técnicas y herramientas	28
6.1.	Dispositivo	28
6.1.1.	Arduino IDE.....	28
6.1.2.	Draw.io	29
6.1.3.	Prusa Slicer	29
6.1.4.	Soldador	30
6.1.5.	MongoDB Compass	30
6.2.	Servidor	31
6.2.1.	Visual Studio Code.....	31
6.2.2.	Python	31
6.2.3.	Docker	32
6.2.4.	Flask.....	32
6.2.5.	API OpenWeatherMap	33
6.2.6.	Plotly.....	33

6.2.7.	Mosquitto	34
6.3.	Herramientas CASE	34
6.3.1.	UML	34
6.3.2.	Visual Paradigm	35
6.3.3.	EZEstimate	35
6.3.4.	Microsoft Project.....	35
6.4.	Control de versiones.....	36
6.4.1.	GitHub	36
7.	Aspectos relevantes del desarrollo.....	37
7.1.	Metodología	37
7.2.	Estimación del Esfuerzo	39
7.3.	Planificación temporal	43
7.4.	Especificación de requisitos	46
7.4.1.	Participantes:.....	46
7.4.2.	Objetivos del sistema:	46
7.4.3.	Requisitos de información.....	48
7.4.4.	Requisitos de restricción de la información	49
7.4.5.	Requisitos funcionales.....	50
7.4.6.	Requisitos no funcionales.....	53
7.5.	Análisis de requisitos.....	54
7.5.1.	Modelo de dominio	54
7.5.2.	Realización de casos de uso	55
7.5.3.	Clase de análisis.....	56
7.6.	Diseño del sistema.....	57
7.6.1.	Diseño de la interfaz.....	57
7.6.2.	Modelo de diseño.....	61
7.6.3.	Patrones arquitectónicos	62
7.6.4.	Subsistemas de diseño	63
7.6.5.	Clases de diseño	64
7.6.6.	Vista arquitectónica.....	65
7.6.7.	Realización de casos de uso	66
7.6.8.	Diseño de la base de datos.....	67
7.6.9.	Modelo de despliegue	70
7.7.	Implementación	71
7.8.	Pruebas	73
7.9.	Funcionalidad del sistema	74

7.9.1.	Página inicial	74
7.9.2.	Registro.....	75
7.9.3.	Iniciar sesión.....	76
7.9.4.	Página principal	77
7.9.5.	Visualizar datos.....	78
7.9.6.	Perfil	79
7.9.7.	Eliminar cuenta.....	80
7.9.8.	Recuperar contraseña	80
8.	Instalación y puesta en marcha	81
8.1.	Instalación	81
8.2.	Puesta en marcha.....	82
9.	Conclusiones y líneas de trabajo futuras.....	84
9.1.	Conclusiones	84
9.2.	Líneas de trabajo futuras	85
10.	Referencias	86

Índice de tablas

Tabla 1: Complejidad de los actores	39
Tabla 2: Complejidad de los casos de uso.....	39
Tabla 3: Factores de complejidad técnica	40
Tabla 4: Factores de complejidad del entorno	42
Tabla 5: Participante Diego Plata Klingler	46
Tabla 6: OBJ-0001 Gestión de usuarios	47
Tabla 7: IRQ-0001 Información sobre los usuarios	48
Tabla 8: CRO-0001 Correo electrónico.....	49
Tabla 9: UC-0008 Eliminar cuenta	51
Tabla 10: ACT-0004 Dispositivo	52
Tabla 11: NFR-0001 Disponibilidad	53

Índice de figuras

Figura 1: Gráfica de humedad del suelo óptima.....	17
Figura 2: Rachio Smart Sprinkler Controller.....	18
Figura 3: Orbit B-hyve Smart Hose Faucet Timer	19
Figura 4: Datalogger Novagric SYNION.....	19
Figura 5: Plantae Sensor Inteligente.....	20
Figura 6: Arquitectura Cliente-Servidor.....	24
Figura 7: MQTT	24
Figura 8: MongoDB.....	25
Figura 9: Docker	25
Figura 10: Internet of Things.....	26
Figura 11: Impresión 3D	27
Figura 12: Arduino IDE	28
Figura 13: Draw.io	29
Figura 14: Prusa Slicer	29
Figura 15: MongoDB Compass.....	30
Figura 16: Visual Studio Code	31
Figura 17: Python	31
Figura 18: Docker	32
Figura 19: Flask.....	32
Figura 20: OpenWeather	33
Figura 21: Plotly	33
Figura 22: Mosquitto	34
Figura 23: Unified Modeling Language	34
Figura 24: Visual Paradigm.....	35
Figura 25: EZEstimate	35
Figura 26: Microsoft Office Project.....	35
Figura 27: GitHub	36
Figura 28: Proceso Unificado.....	38
Figura 29: Resultados de EzEstimate.....	43
Figura 30: Microsoft Project. Inicio.....	44
Figura 31: Diagrama de Gantt 1.....	45
Figura 32: Paquete Gestión de usuarios.....	50
Figura 33: Modelo de dominio del sistema.....	54
Figura 34: Diagr secuencia CU-0004 Modificar dispositivos registrados	55
Figura 35: Diagrama de comunicación Gestión de usuarios	56
Figura 36: Código de colores seleccionados.....	57

Figura 37: Prototipado digital. Ventana Registro. Tema claro	58
Figura 38: Prototipado digital. Ventana Iniciar Sesión. Tema claro	58
Figura 39: Prototipado digital. Ventana Iniciar Sesión. Tema oscuro	59
Figura 40: Prototipado digital. Ventana Principal	59
Figura 41: Prototipado digital. Barra lateral	60
Figura 42: Prototipado digital. Ventana modificar perfil	60
Figura 43: Prototipado digital. Ver datos de dispositivo.....	61
Figura 44: Diagrama MVC	62
Figura 45: Subsistemas de diseño	63
Figura 46: Subsistema Controlador	64
Figura 47: Vista arquitectónica MVC.....	65
Figura 48: Diagrama de secuencia CU-0003 Ver perfil.....	66
Figura 49: Colección Usuarios. BD SQL	68
Figura 50: Colección Dispositivos. BD SQL	68
Figura 51: Mediciones. BD NoSQL	69
Figura 52: Modelo de despliegue.....	70
Figura 53: Ejemplo de endpoint	71
Figura 54: Página principal	74
Figura 55: Registrar cuenta	75
Figura 56: Mensaje de error de registro.....	75
Figura 57: Mensaje de error en iniciar sesión	76
Figura 58: Comprobación de campos. Tema oscuro.....	76
Figura 59: Página principal	77
Figura 60: Barra lateral	77
Figura 61: Gráficas de mediciones	78
Figura 62: Ampliar zona de la gráfica.....	78
Figura 63: Página perfil de usuario	79
Figura 64: Ventana modal de eliminar cuenta.....	80
Figura 65: Ventana modal de recuperar contraseña	80
Figura 66: Diagrama de conexión	81
Figura 67: Prototipo. Componentes instalados	82
Figura 68: Cuadro de diálogo Ejecutar.....	83

1. Introducción

El concepto de Internet of Things [3], de forma acortada IoT, se puede definir como una red global de dispositivos físicos que están integrados con sensores, software y conectividad de red, lo que les permite recopilar y compartir datos. Esta tecnología ha revolucionado el mundo de internet, ya que entre sus aplicaciones se encuentra conocer en tiempo real el estado de dichos dispositivos físicos, junto con la posibilidad de intervenir de forma remota en caso de ser necesario. Esto ofrece un sinfín de oportunidades y posibles mejoras en muchos ámbitos. Uno de ellos, que es el que se intenta lograr con este proyecto, es el ahorro de recursos y la mejora de la eficiencia.

Uno de los recursos más malgastados es el agua, y aunque se intenta concienciar cada vez más a la población de la urgencia de no desperdiciarla, sigue siendo necesario tomar medidas. Entre los sectores que más agua consumen a nivel global nos encontramos la agricultura o la jardinería.

Además, el precio del agua ha subido recientemente, haciendo que todas las actividades que impliquen su uso en grandes cantidades se vean afectadas negativamente. El sector de la agricultura, además de depender totalmente del agua corriente ya que normalmente el agua de las precipitaciones no suele ser suficiente, es un sector imprescindible en la sociedad al que no se le ha atribuido la importancia que merece, como hemos podido apreciar en las recientes manifestaciones. La necesidad de optimizar los métodos de cultivo y gestión de recursos se vuelve necesaria para abaratar los costes y reducir el impacto medioambiental. [4]

Al igual que la agricultura, la jardinería necesita del agua corriente para mantenerse, especialmente en las zonas urbanas donde las opciones de riego natural son limitadas. Esto, unido a las subidas del precio del agua, ha llevado a

En este contexto, este proyecto tiene como objetivo la creación de un dispositivo de riego inteligente basado en IoT que se pueda convertir en una herramienta que ofrezca facilidades al sector de la agricultura, aunque se puede escalar a cualquier ámbito que implique riego ya que se busca aumentar la eficiencia en el uso del agua.

En este documento se van a explicar los puntos más destacados del desarrollo del dispositivo. Su estructura sigue los siguientes apartados.

- **Análisis de mercado:** Se hace un estudio del mercado con los dispositivos disponibles que pertenecen al mismo campo de desarrollo.
- **Objetivos:** Se detallan los objetivos que se desean lograr con el desarrollo del dispositivo planteado en el proyecto.
- **Conceptos teóricos:** Se explican los conceptos teóricos necesarios para comprender correctamente el proyecto. Esto servirá como guía de conceptos técnicos para que se pueda entender de forma más clara el proyecto aún sin tener experiencia o conocimientos en este campo.
- **Técnicas y herramientas:** Se detallarán las técnicas y herramientas empleadas en la realización del proyecto.
- **Aspectos relevantes del desarrollo:** Se comentarán los aspectos más relevantes del desarrollo e implementación del dispositivo.
- **Conclusiones y líneas de trabajo futuras:** Una vez acabado el proyecto se comentan las conclusiones a las que se ha llegado, y las futuras líneas de trabajo.
- **Referencias:** Se incluyen las citas utilizadas durante el desarrollo del proyecto.

Además, se adjuntan el código y el resto de la documentación técnica que está compuesta por los *datasheets* de los componentes, y siguientes anexos:

- **ANEXO I. Plan de Proyecto:** En este anexo se presenta una estimación de la duración, coste y esfuerzo del proyecto.
- **ANEXO II. Especificación de Requisitos Software:** En este apartado se comenta el sistema, describiendo sus funcionalidades, la especificación de requisitos, actores...
- **ANEXO III. Análisis del Sistema Software:** Se realiza análisis, refinamiento y estructuración del proyecto usando la información recogida en el Anexo II.
- **ANEXO IV. Diseño del Sistema Software:** En este anexo se realiza la documentación relacionada con el diseño del sistema.
- **ANEXO V. Manual de Usuario:** Este anexo sirve para facilitarle el uso del dispositivo al usuario.
- **ANEXO VI. Manual del Programador:** Recoge toda la documentación referente al código desarrollado, facilitando su comprensión.
- **ANEXO VII. Esquemas electrónicos, diseño y montaje:** Recoge la documentación referente al Hardware del sistema.

2. Estado del arte

El proyecto da comienzo realizando una investigación de la literatura científica y estudios ya realizados acerca de la irrigación inteligente enfocada sobre todo a la agricultura sostenible.

El primer punto que se investigó fue la elección de las variables a monitorizar en nuestro sistema. Se ha hecho un repaso de las diferentes variables que más se monitorizan, descartando algunas de ellas por su impracticabilidad, como por ejemplo, los nutrientes del suelo. Aún así, se han detectado los parámetros más importantes para conseguir un buen sistema de riego inteligente. [5]

La humedad del suelo es una de las variables más importantes, ya que indica la disponibilidad de agua de las plantas. Su monitorización proporciona información vital para optimizar el riego y evitar el exceso o déficit de agua. Además, mantener los valores de esta variable en el intervalo correcto reduce considerablemente el desperdicio de agua [6]. Otro estudio realizado [7] muestra como el anegamiento o inundación del suelo hace que se desplace el oxígeno para las raíces de las plantas, provocando condiciones de anaerobia (falta de oxígeno). Este es otro motivo por el que mantener el nivel de humedad del suelo controlado.

La humedad ambiente es esencial para el cuidado de las plantas, ya que permite tener un panorama más amplio de las condiciones ambientales que afectan la disponibilidad del agua para las plantas. La humedad ambiente influye en la evaporación del agua del suelo y en la transpiración de las plantas. Conociendo esta variable se puede realizar una planificación de riego ajustada a estas variables. Además, hay otro factor en el que influye la humedad ambiente y es el estrés hídrico, una condición en la cual las plantas experimentan una falta de agua disponible en el suelo. Cuando la humedad ambiental es baja, el aire tiene una capacidad de absorber más agua, favoreciendo la evaporación del agua del suelo. [8]

La temperatura ambiente es otro factor crítico que considerar, ya que afecta directamente a la tasa de evaporación y transpiración de las plantas. Monitorizar de forma constante la temperatura ambiente permite ajustar los programas de riego según los valores recogidos. Además, la temperatura ambiente influye en la absorción de los nutrientes por parte de las plantas ya que, por ejemplo, temperaturas altas pueden aumentar la velocidad de transporte, mientras que temperaturas extremas ralentizan este proceso, afectando al crecimiento.

Por otro lado, una variable que también es importante monitorizar es la radiación solar, ya que es básica para la fotosíntesis de las plantas. Sin embargo, no se incluirá en este proyecto ya que, entre otras cosas, el equipamiento es costoso y el procesamiento de datos requiere conocimientos que no se disponen. En este proyecto se monitorizarán las tres primeras variables mencionadas. La combinación de estas tres ofrece suficiente información para realizar un sistema de riego preciso y eficaz, suponiendo un cuidado mayor de la planta, y un ahorro de agua elevado. Además, para maximizar el ahorro de agua mediante futuros cambios de estas variables, se obtiene un pronóstico meteorológico de las próximas horas en un determinado lugar mediante una API.

2.1 Unidades estándar de las variables monitorizadas, así como su valor óptimo

Se han mencionado las variables a monitorizar para mantener un estado correcto de la planta. En este apartado se definirán las unidades estándar de dichas variables, así como su valor óptimo. Para ello se usarán como referencia distintos estudios y publicaciones. Los valores óptimos dependen de la planta, pero usaremos un intervalo que abarque la mayor cantidad de plantas de interior posible.

La temperatura óptima a la que debe estar una planta varía mucho según el tipo, pero para plantas de interior comunes un rango adecuado es entre 15°C y 24°C. Una temperatura superior a lo adecuado puede producir una mayor deshidratación de la planta, mientras que una temperatura menor a ese rango puede producir un crecimiento lento ya que ralentiza su crecimiento, además de que pueden tener problemas con la absorción del agua.

La humedad ambiente del entorno oscila entre el 40% y el 60%. Este porcentaje es la cantidad de vapor que hay en el aire respecto al máximo que podría haber cuando ese aire se satura (que ya no puede acumular más agua en sí). Si se alcanza el 100%, empezaría a condensarse y a aparecer gotas de agua líquida. Este rango de humedad puede variar dependiendo de la etapa en la que se encuentre la planta, ya que, por ejemplo, en etapas de floración se recomienda un nivel más bajo para evitar hongos.

La humedad del suelo óptima de una planta depende del suelo, ya que varía su capacidad de campo; y del punto de marchitamiento de la planta. La capacidad de campo se refiere a la cantidad de agua que contiene un suelo saturado después de 48h de drenaje. Un rango entre un 20% y un 40% de humedad suele ser lo adecuado.

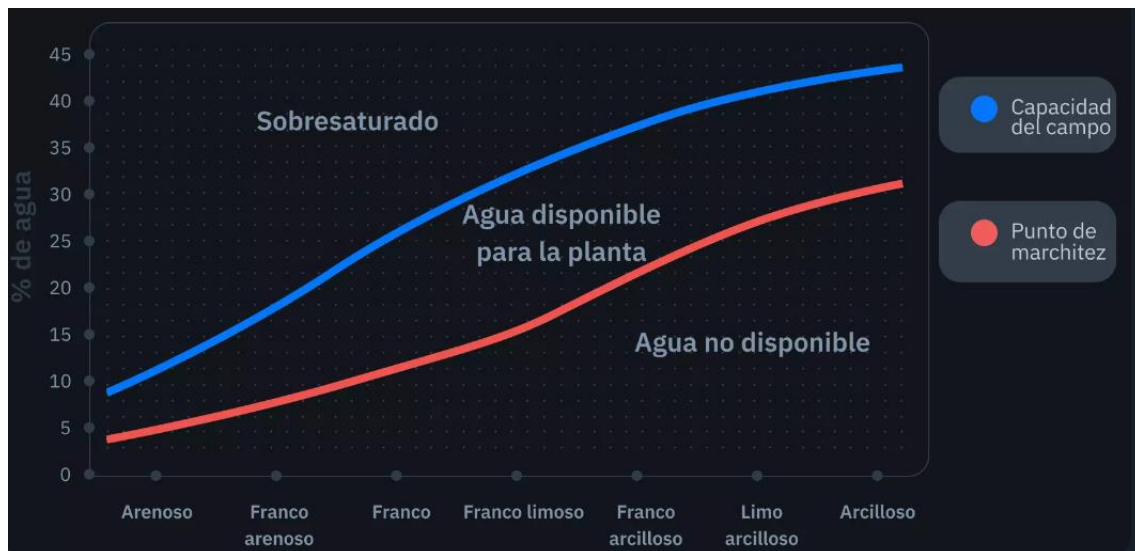


Figura 1: Gráfica de humedad del suelo óptima

Investigación de técnicas de monitorización y cuidado automático de las plantas

En este apartado, se ha investigado metodologías de análisis y monitorización de las plantas. Se han analizado y comparado los métodos ya existentes con el propuesto en este proyecto, buscando similitudes y diferencias. La mayoría de los estudios realizan una monitorización de la planta mediante sensores que recogen los datos directamente del ambiente, igual que la esta propuesta. Veremos otras propuestas que han sido descartadas, indicando los motivos.

La primera propuesta planteada fue la monitorización con drones. Los drones son dispositivos voladores que pueden ser controlados de forma remota, o volar de manera autónoma. Si a un dron se le equipa una cámara de alta resolución y otros sensores, se convierte en una técnica de análisis muy precisa ya que obtiene imágenes en tiempo real del estado de la planta de forma remota. Se ha descartado esta técnica ya que no se tienen los recursos para crear u obtener un dron. [9]

La segunda técnica valorada fue una red neuronal convolucional. Una red neuronal convolucional es un tipo de red neuronal artificial diseñada para procesar y analizar datos de imágenes. Esta técnica fue descartada ya que necesitaba demasiadas imágenes para ser entrenada, además de realizar un estudio biológico detallado acerca de una planta en concreto.[10]

La última técnica estudiada fue el uso de un controlador PID para mantener la planta en unos valores muy concretos. Un controlador PID es un método de dirigir un sistema

hacia una posición o nivel determinado. Se ha descartado esta idea ya que solamente serviría para plantas que necesitan un cuidado intensivo. [11]

En este proyecto se propone una forma de monitorización y riego mediante sensores y actuadores. Para ello se usa un sensor de temperatura y humedad ambiente, el DHT22; un sensor capacitivo de humedad del suelo; un caudalímetro; y una bomba de agua. Hay que añadir que estos sensores no suelen ser utilizados en entornos profesionales, pero teniendo en cuenta el ámbito del proyecto (Trabajo de Fin de Grado) y el presupuesto del que se dispone, han sido las mejores opciones encontradas.

A pesar de haber descartado las técnicas anteriores por falta de recursos, la combinación de la técnica propuesta en este proyecto con la red neuronal convolucional es una posibilidad que cubre todas las posibles necesidades de un sistema así. Se deja planteada la idea como línea de trabajo futura.

3. Análisis del mercado

Antes de empezar con el desarrollo del dispositivo realicé un análisis del mercado con el objetivo de encontrar dispositivos que realicen funciones similares al dispositivo que voy a desarrollar. Gracias a este estudio pude ver que el proyecto cuenta con funcionalidades parecidas a algunos dispositivos ya existentes, pero siempre encontramos algunas diferencias.

3.1. Rachio Smart Sprinkler Controller:

Se trata de un dispositivo que se conecta a los aspersores inteligentes y usa los datos meteorológicos en tiempo real para ajustar los tiempos de riego y así evitar desperdiciar agua. La principal desventaja es que únicamente obtiene los datos mediante pronósticos de tiempo, siendo estos imperfectos y pudiendo llegar a perjudicar a las plantas. [12]



Figura 2: Rachio Smart Sprinkler Controller

3.2. Orbit B-hyve Smart Hose Faucet Timer:

Este dispositivo se conecta al grifo de la manguera y permite programar horarios de riego precisos. También tiene funciones inteligentes como la suspensión automática de riego en caso de lluvia. La principal desventaja es que los horarios de riego se programan de manera manual, pudiendo dar esto lugar a fallos, y las funciones inteligentes también funcionan con pronósticos de tiempo. [13]

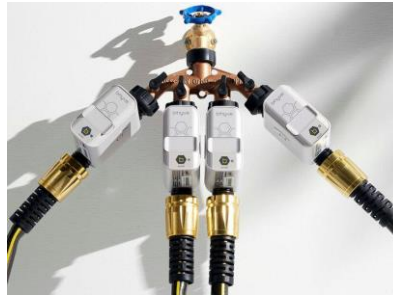


Figura 3: Orbit B-hyve Smart Hose Faucet Timer

3.3. Datalogger Novagric SYNION:

Este dispositivo es una herramienta capaz de recopilar y analizar datos de cultivos con el objetivo de maximizar el control y ahorrar agua y fertilizantes. La principal desventaja que tiene es que está pensado para grandes cultivos, además de que solamente obtiene los datos con los sensores sin tener en cuenta pronósticos, pudiendo ser esto perjudicial para los cultivos. [14]



Figura 4: Datalogger Novagric SYNION

3.4. Plantae Sensor Inteligente de Humedad, Conductividad y Temperatura:

Este dispositivo proporciona sensores útiles para el control y estudio de las plantas, midiendo la humedad de la tierra la temperatura superficial y del suelo entre otras cosas. Su principal desventaja es que está pensado para riego por goteo, reduciendo bastante las aplicaciones del dispositivo. [15]



Figura 5: Plantae Sensor Inteligente

4. Objetivos

Este apartado trata de representar los objetivos a cumplir en la realización del proyecto de fin de grado. Se van a desarrollar los objetivos marcados por los requisitos del software y los objetivos técnicos. Este punto también tratará los objetivos personales que pretendo alcanzar con la realización de este proyecto.

4.1. Objetivos funcionales.

Procedemos a definir los objetivos funcionales del sistema:

Medir datos mediante sensores ambientales: El sistema debe ser capaz de recopilar datos de los sensores de humedad del suelo, temperatura y humedad ambiente, de forma precisa y convertirlos a unidades métricas adecuadas, siguiendo las especificaciones técnicas del fabricante.

Enviar y recibir los datos: El sistema debe enviar y recibir los datos desde los sensores hasta la base de datos utilizando un protocolo de comunicación

Almacenar datos: Los datos recopilados deben ser almacenados en una base de datos de manera estructurada y coherente, facilitando el acceso y consulta de los mismos de manera eficiente.

Aplicar operaciones a los datos: El sistema debe aplicar las operaciones y expresiones necesarias a los datos para crear una óptima planificación de riego.

Poner en uso la planificación anterior: El sistema debe automatizar el proceso de riego utilizando la planificación creada anteriormente.

Visualizar los datos: El sistema debe proporcionar al usuario una interfaz en la que consultar todos el contenido de la base de datos, así como el de la predicción de riego.

4.2. Objetivos Personales.

El principal objetivo de este proyecto es la aplicación de los conocimientos adquiridos durante el Grado en Ingeniería Informática, además de trabajar nuevos lenguajes y tecnologías con los que no se ha trabajado hasta la fecha. Esta oportunidad representa no solo un desafío personal de superación y aprendizaje continuo, sino también una oportunidad para desarrollar mi creatividad en el ámbito tecnológico. La realización de este Trabajo de Fin de Grado (TFG) es una forma de relacionar la teoría aprendida en las clases del grado con la práctica real, permitiéndome así demostrar mi capacidad para resolver problemas complejos y desarrollar soluciones tecnológicas viables. Además, este proyecto me permitirá especializarme en áreas específicas de mi interés, abriendo puertas a futuras oportunidades de carrera y desarrollo profesional. En resumen, este TFG no solo es el final de mi etapa universitaria, sino también una forma de enfrentarme a los problemas tecnológicos de manera más realista y cotidiana.

5. Conceptos teóricos

Como vamos a usar varios términos algo más técnicos cuya comprensión es necesaria para conocer correctamente el funcionamiento del proyecto, vamos a explicarlos en este apartado:

5.1. Frontend

El Frontend de un Software es la parte a la que un usuario puede acceder directamente. Se encarga de hacer que funcione todo correctamente, y recoger las acciones de los usuarios. Una buena Interfaz Gráfica tiene que cumplir con los principios de usabilidad de Nielsen para crear una experiencia de usuario positiva. [16]

5.2. Backend

El Backend de un Software es la parte que gestiona la lógica del servidor, la base de datos y la comunicación entre el servidor y el cliente. Se encarga de procesar las solicitudes del frontend, manejar la lógica, almacenar y recuperar datos, y asegurar la integridad de la información. Un Backend eficiente debe garantizar el correcto funcionamiento de la aplicación. [17]

5.3. Arquitectura Cliente-Servidor

La arquitectura cliente-servidor es un modelo de diseño de software que permite la distribución de tareas dentro de una red de ordenadores. En esta arquitectura distinguimos dos roles claros: el cliente y el servidor.[18]

- El cliente es el que se encarga de interactuar directamente con los usuarios mediante un Frontend, realizando las solicitudes y peticiones al servidor.
- El servidor es el que recibe dichas peticiones, las procesa y responde a las mismas. A menudo, se hace la analogía con el Backend, ya que el servidor maneja la lógica de negocio, el acceso a bases de datos y asegura la seguridad e integridad de la información, haciendo que el Frontend funcione correctamente.

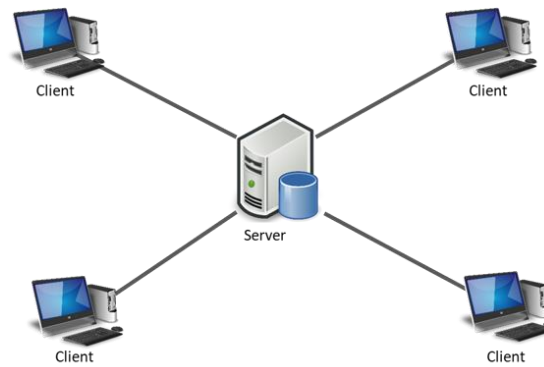


Figura 6: Arquitectura Cliente-Servidor [18]

5.4. MQTT

MQTT es un protocolo de mensajería que se utiliza para la comunicación de un equipo a otro. Los sensores inteligentes, los dispositivos portátiles y otros dispositivos de Internet de las cosas (IoT) generalmente tienen que transmitir y recibir datos a través de una red con recursos restringidos y un ancho de banda limitado. Estos dispositivos IoT utilizan MQTT para la transmisión de datos, ya que resulta fácil de implementar y puede comunicar datos IoT de manera eficiente.



Figura 7: MQTT

En este proyecto se utilizará para comunicar el dispositivo con el servidor, y viceversa. El bróker elegido en este caso ha sido Mosquitto, desarrollado Open Source desarrollado por la fundación Eclipse. Está programado en C, y es multiplataforma. Es un bróker ligero y adecuado para uso en servidores de baja potencia. [19]

5.5. MONGODB

MongoDB es una base de datos no relacional (NoSQL) orientada a documentos. Utiliza un formato de almacenamiento basado en JSON, en lugar de tablas y filas como las bases de datos relacionales. Esto permite mayor flexibilidad y esquemas dinámicos.

Se ha elegido esta base de datos para este sistema ya que MongoDB es especialmente adecuada para almacenar datos de sensores de dispositivos IoT gracias a su capacidad de manejar grandes cantidades de datos. Además, la flexibilidad de mongo permite

insertar los datos en formato JSON, que coincide con el mensaje enviado por MQTT. Esto simplifica el procesamiento del sistema debido a que no hace falta realizar ningún formateo de los datos. [20]



Figura 8: MongoDB

5.6. Docker

Docker es una plataforma de código abierto que permite la virtualización a nivel de sistema operativo, usando contenedores para empaquetar el software con sus dependencias. [21]



Figura 9: Docker

En este proyecto se ha decidido usar Docker para el almacenamiento de datos de los sensores y para el control de los mensajes que llegan del dispositivo, es decir, para el bróker MQTT.

Para el almacenamiento de los datos de los sensores se ha optado por usar un contenedor Docker por diversos motivos:

- **Portabilidad y consistencia:** Usar Docker para la base de datos de las mediciones de los sensores asegura que funcione en cualquier entorno, independientemente del sistema operativo. Esto elimina los problemas de configuración que puedan aparecer.
- **Aislamiento:** Con Docker podemos aislar la base de datos en el contenedor, reduciendo el riesgo de conflictos de dependencia con otras bases de datos que tengamos en nuestro dispositivo.

- **Facilidad en el despliegue:** Usar Docker nos permite un despliegue muy sencillo, necesitando únicamente tener instalado Docker. Evita tener que instalar todas las dependencias y bibliotecas necesarias para que funcione.

Por otro lado, se ha decidido mantener las bases de datos de los usuarios y de los dispositivos en local, ya que vamos a estar accediendo constantemente a ella y por temas de rendimiento y latencia tenerla en local puede mejorar estos aspectos.

La inclusión del bróker MQTT en el contenedor Docker es por motivos similares. Para empezar, mantener el bróker MQTT dentro del Docker permite el procesamiento e ingesta de los datos en la base de datos MongoDB sin salir del contenedor, mejorando el rendimiento. Además, para volver a lanzar un bróker MQTT con Docker es tan sencillo como ejecutar el contenedor de nuevo, permitiendo un despliegue muy sencillo.

5.7. Internet of Things

El concepto de Internet of Things (IoT), también conocido como Internet de las Cosas, se refiere a la red global de dispositivos físicos integrados con sensores, software y conectividad de red, que les permite recopilar y compartir datos. Esta tecnología ha revolucionado múltiples sectores, incluyendo la agricultura, donde puede ayudar al ahorro de recursos y a la mejora de la eficiencia. [3]

En el contexto de la agricultura sostenible, el IoT puede desempeñar un papel muy importante en la optimización del uso del agua, un recurso cada vez más escaso. Por ejemplo, este sistema de riego inteligente basado en IoT puede ajustar automáticamente y la cantidad de riego en función de los datos recopilados por sensores de humedad del suelo y ambiente, y condiciones meteorológicas, reduciendo así el desperdicio de agua y asegurando que las plantas reciban la cantidad exacta que necesitan.



Figura 10: Internet of Things

5.8. Impresión 3D

La impresión 3D es un proceso de fabricación aditiva que permite crear objetos a partir de un modelo digital. La impresión 3D utiliza una serie de capas sucesivas de material, que pueden ser de diferentes tipos, como plástico, resina o cerámica, para formar el objeto. Esta tecnología se usa en numerosos campos, como la medicina (para crear prótesis, implantes u órganos artificiales), la arquitectura (para construir edificios o maquetas), o la ingeniería (para fabricar prototipos o componentes). [22]

Esto ofrece una serie de ventajas para personalizar, tales como la rapidez o la eficiencia. Además, al utilizar solo el material necesario para construir un objeto, la fabricación aditiva reduce mucho el desperdicio de material en comparación con los métodos de fabricación tradicionales.

Para este proyecto se ha utilizado la impresión 3D para crear una carcasa protectora del dispositivo, y darle una presentación al dispositivo más estética.



Figura 11: Impresión 3D

6. Técnicas y herramientas

En este apartado se van a comentar y describir las técnicas empleadas para el desarrollo del sistema, no solo para la parte del servidor o la parte del dispositivo, sino que también se mencionarán el resto de las herramientas empleadas.

6.1. Dispositivo

6.1.1. Arduino IDE

Para el desarrollo del dispositivo se ha utilizado Arduino IDE, un entorno de desarrollo que facilita la programación de placas de microcontroladores. Ofrece una interfaz de usuario sencilla, ideal para programar en Arduino ya que incluye un administrador de placas que nos permite seleccionar el modelo de nuestra placa y cargar el código directamente.

El lenguaje Arduino es un lenguaje de programación visto en la asignatura de Periféricos. Se utiliza para escribir programas y cargarlos en placas de desarrollo. Además, es un lenguaje basado en C y C++, visto en asignaturas de programación del primer curso del grado.



Figura 12: Arduino IDE

Para poder programar y utilizar la ESP32 desde Arduino, es necesario descargar un software que incluya bibliotecas, definición de pines, configuraciones y otros recursos específicos para esta plataforma. En este caso, se ha utilizado "arduino-esp32", desarrollado por Espressif Systems, que proporciona el núcleo de Arduino adaptado para el microcontrolador ESP32, facilitando la creación de proyectos y la interacción con sus funciones de conectividad WiFi.

6.1.2. Draw.io

Para el diagrama de conexión del dispositivo se utilizó la herramienta online draw.io. El principal motivo de elegir esta herramienta frente a otras es que, además de ser gratuita, permite crear diagramas de manera intuitiva y sin necesidad de tener demasiada experiencia previa, y exportarlos a cualquier formato. Este diagrama facilita la visualización de la disposición física de los componentes electrónicos.



Figura 13: Draw.io

6.1.3. Prusa Slicer

Prusa Slicer es un software diseñado para la preparación y el “slicing” de modelos 3D para impresoras 3D. El slicing es el proceso mediante el cual se convierte un modelo tridimensional digital, como un archivo STL o 3MF, en instrucciones en formato G-code. Estas instrucciones son necesarias para que la impresora 3D pueda construir el objeto capa por capa de manera precisa. Otra ventaja que tiene PrusaSlicer es que permite configurar parámetros de impresión como la velocidad, la temperatura y el tipo de relleno. También permite generar soportes automáticos. La siguiente foto muestra una impresora de filamento del centro de emprendimiento Tormes Plus, en Salamanca. [23]

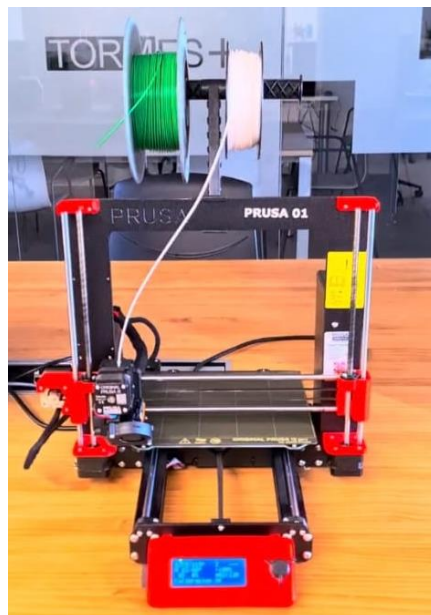


Figura 14: Prusa Slicer

Se ha elegido este proyecto ya que fue con este software con el que se aprendió a imprimir en 3D, además de ser el más extendido dentro de las impresoras Prusa.

6.1.4. Soldador

Por último, aunque no sea una herramienta software, hay que mencionar el soldador de estaño usado para realizar las conexiones eléctricas permanentes del sistema.

6.1.5. MongoDB Compass

MongoDB Compass es una interfaz gráfica para MongoDB [24]. Su uso es exclusivamente para facilitar la interacción con MongoDB, permitiendo realizar las tareas de gestión de datos de manera más intuitiva y rápida. Se utiliza esta herramienta ya que simplifica la exploración de los datos, permitiendo modificaciones de manera muy sencilla sin tener que usar la terminal. Además, permite las conexiones a bases de datos con autenticación de manera muy intuitiva.

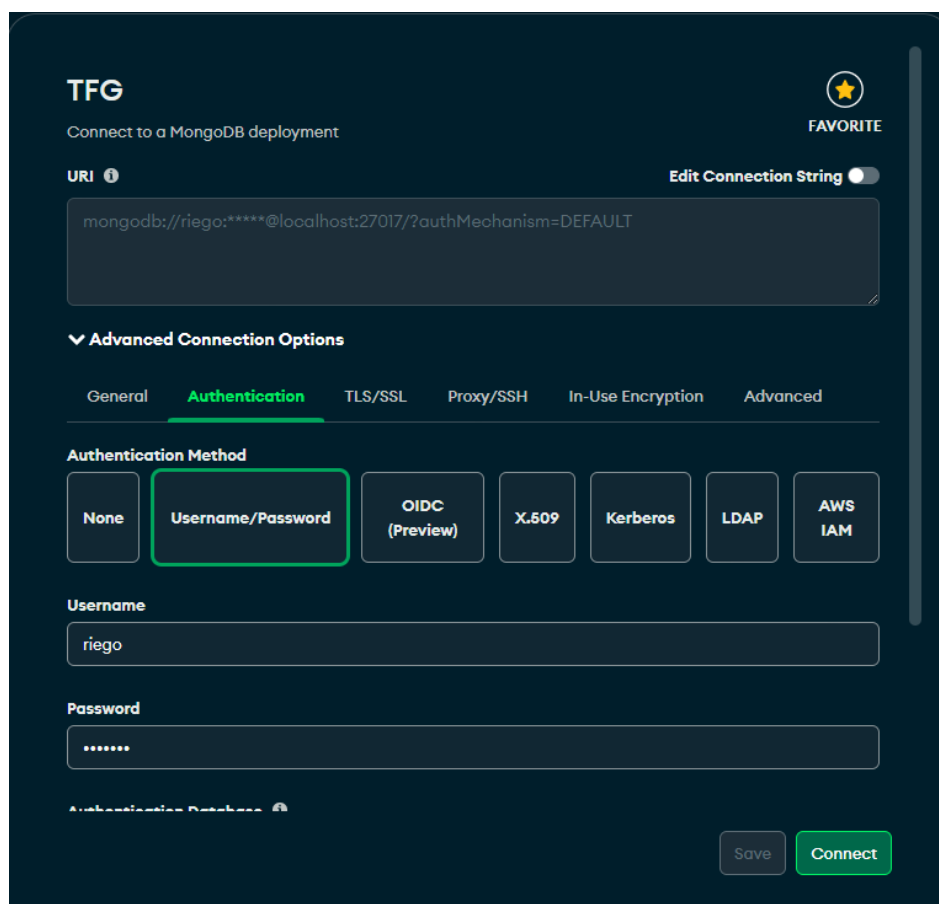


Figura 15: MongoDB Compass

6.2. Servidor

6.2.1. Visual Studio Code

Visual Studio Code es un editor de código desarrollado por Microsoft, Es multiplataforma y gratuito. Esto, junto a su amplia gama de extensiones que mejoran la experiencia de usuario y la familiaridad de uso, han hecho que sea la opción elegida como herramienta en la que se pasará la mayor parte del tiempo de desarrollo. [25]



Figura 16: Visual Studio Code

6.2.2. Python

El lenguaje de desarrollo elegido para la parte de aplicación web es Python. Python es un lenguaje que destaca por su simplicidad. Además, cuenta con muchas bibliotecas y frameworks (como Flask, del que se hablará más adelante), que permiten crear aplicaciones web robustas.

Por otro lado, al ser un lenguaje tan utilizado tiene una gran comunidad detrás, haciendo que haya muchos recursos en línea.

Estas ventajas sumadas a que se ha estado trabajando con Python durante los últimos meses, ha hecho que sea el lenguaje escogido para el Trabajo de Fin de Grado.



Figura 17: Python

6.2.3. Docker

Docker es una plataforma de virtualización a nivel de sistema operativo que permite crear, probar e implementar aplicaciones rápidamente. Docker empaqueta software en unidades estandarizadas llamadas contenedores que incluyen todo lo necesario para que el software se ejecute, incluidas bibliotecas, herramientas de sistema, código y tiempo de ejecución. Los componentes principales de Docker son las imágenes y los contenedores. Para crear una imagen se usa un archivo llamado Dockerfile. Este es un archivo de texto que contiene una serie de instrucciones sobre cómo construir una imagen Docker. Define el entorno y los pasos necesarios para instalar las dependencias y la aplicación.

En este proyecto se ha usado un Docker para usar una base de datos MongoDB para guardar las mediciones de los sensores, y un bróker MQTT para la comunicación entre dispositivo y servidor, ya que Docker nos permite crear entornos portátiles de manera eficiente.

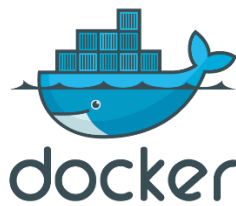


Figura 18: Docker

6.2.4. Flask

Flask es un framework de código abierto para aplicaciones web en Python. Es conocido por ser ligero y fácil de usar, permitiendo crear aplicaciones web robustas y escalables. Flask proporciona las herramientas y bibliotecas necesarias para construir una aplicación web, incluyendo soporte para solicitudes, respuestas y sesiones.

En términos de arquitectura, Flask sigue el patrón de diseño MVC (Modelo-Vista-Controlador), lo que facilita organizar el código. Esto hace que el mantenimiento y la expansión de las aplicaciones sean más sencillas.



Figura 19: Flask

6.2.5. API OpenWeatherMap

OpenWeatherMap es una plataforma en línea que ofrece datos meteorológicos. Su API permite acceder a información actual y pronosticada sobre el clima, el viento, la presión, la humedad, la lluvia y otros parámetros meteorológicos de cualquier lugar del mundo. La API devuelve los datos en formato JSON.

En este proyecto, se usa la API de OpenWeatherMap para obtener el pronóstico meteorológico de la ciudad en la que está registrado un dispositivo. Esto permite ajustar la planificación de riego en función del pronóstico obtenido. [26]



Figura 20: OpenWeather

6.2.6. Plotly

Aunque no se vayan a mencionar todas las bibliotecas utilizadas en el desarrollo del TFG, sí hay que hacer una mención a Plotly.

Plotly es una biblioteca de visualización de datos que crea gráficos interactivos para representar datos de manera efectiva. En este caso, se ha decidido representar los datos en forma de gráfico de líneas.

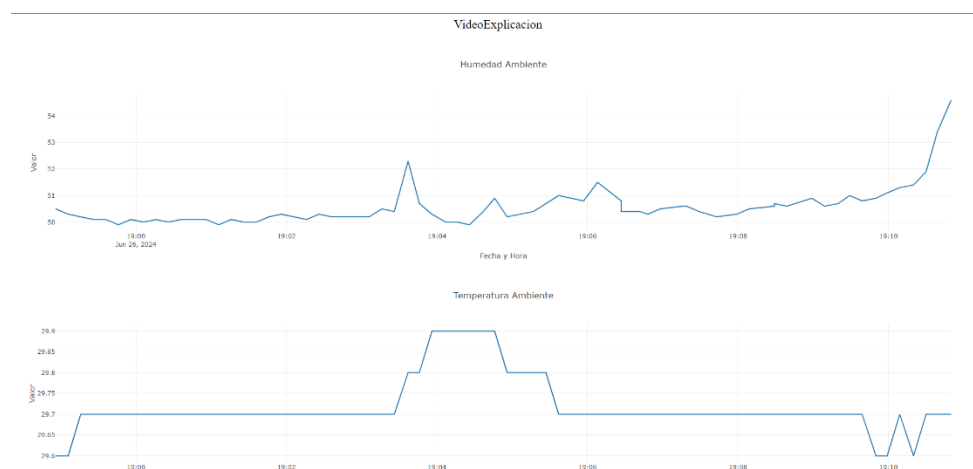


Figura 21: Plotly

6.2.7. Mosquitto

Mosquitto es un bróker de mensajes MQTT de código abierto que implementa el protocolo MQTT. Este protocolo es usado en la comunicación de dispositivos IoT para enviar mensajes entre dispositivos y servidores de manera eficiente y ligera.

Se ha elegido este bróker ya que es el más reconocido y utilizado en el ámbito del Internet de las Cosas. Gracias a esto pude aprender y trabajar con él en las prácticas externas del Grado.



Figura 22: Mosquitto

6.3. Herramientas CASE

Una herramienta CASE es un software que facilita el desarrollo de sistemas de software. Estas herramientas ayudan a planificar, diseñar y probar un sistema software de manera eficiente. Entre sus funcionalidades podemos destacar el modelado gráfico (crear diagramas de casos de uso o de clases), o la gestión de los requisitos (ayudan a documentar y gestionar los requisitos del sistema). Estas herramientas se han utilizado para la realización de los anexos adjuntos a la memoria. [27]

6.3.1. UML

UML es un lenguaje de modelado visual que permite representar conceptos, diseños y decisiones relacionadas con el software. Proporciona diferentes tipos de diagramas (como diagramas de clases, diagramas de secuencia, diagramas de actividad, entre otros) que sirven para modelar la estructura y el comportamiento de un sistema.

Se ha elegido ya que se ha usado este modelo a lo largo del Grado, y porque es el modelado de sistemas más popular.



Figura 23: Unified Modeling Language

6.3.2. Visual Paradigm

Herramienta que sirve para capturar requisitos y transformarlos en diagramas que permitan un desarrollo más sencillo. Se ha utilizado para realizar los diferentes diagramas UML de los anexos del proyecto.

Se ha elegido, igual que UML, porque ya se había utilizado en otras asignaturas.



Figura 24: Visual Paradigm

6.3.3. EZEstimate

Herramienta que permite calcular la estimación de coste y esfuerzo del proyecto para productos software. Se basa en la técnica de estimación de Puntos de Caso de Uso, para los que hay que indicar actores, casos de uso, y factores de complejidad de entorno y técnica. Con ella obtenemos una estimación de la duración del proyecto en horas.



Figura 25: EZEstimate

6.3.4. Microsoft Project

Microsoft Project es una herramienta de gestión de proyectos que ayuda a planificar, ejecutar y supervisar los proyectos y mejorar el rendimiento del equipo. Con ella se ha creado el Diagrama de Gantt para la estimación. [28]

También es una herramienta que se ha usado previamente en la asignatura de Gestión de Proyectos, haciendo que sea la elegida.



Figura 26: Microsoft Office Project

6.4. Control de versiones

6.4.1. GitHub

GitHub es una plataforma de desarrollo colaborativo basado en Git. Es la más utilizada entre desarrolladores y equipos para compartir código, ya que tiene integrado un control de versiones haciendo que deshacer los cambios sea muy sencillo. [29]



Figura 27: GitHub

7. Aspectos relevantes del desarrollo

En este punto se van a comentar los aspectos más relevantes del desarrollo del proyecto en su totalidad.

7.1. Metodología

Para el desarrollo de este trabajo hemos seguido el marco de trabajo del Proceso Unificado.

“El Proceso Unificado es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas software, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de aptitud y diferentes tamaños de proyectos”. [4]

Sus principales características son las siguientes:

- **Basado en componentes:** Se enfoca en la definición y uso de componentes reusables, lo que facilita la gestión de la complejidad del software y promueve la reutilización de código.
- **Utiliza UML:** El lenguaje UML es una herramienta que proporciona una notación estándar para la especificación, visualización, construcción y documentación de los artefactos de sistemas de software, facilitando la comprensión entre los miembros del equipo.
- **Es un proceso conducido por casos de uso:** Guía el desarrollo del software a centrarse en satisfacer los requisitos funcionales desde la perspectiva del usuario.
- **Está centrado en la arquitectura:** Define una arquitectura de software robusta, que sirve como esqueleto para el sistema y ayuda a garantizar que el producto final sea escalable, mantenible y que cumpla los requisitos definidos.
- **Es iterativo e incremental:** El desarrollo se realiza en ciclos iterativos, donde cada iteración produce una versión incrementada y más completa del software. Este enfoque permite manejar cambios en los requisitos de manera más efectiva, así como identificar y solucionar problemas en etapas tempranas del desarrollo.

Como hemos mencionado anteriormente, el Proceso Unificado se realiza en ciclos de desarrollo que constituyen la vida del sistema. Cada ciclo cuenta con cuatro fases: [4]

- **Inicio:** En esta fase se define el alcance del proyecto, definiendo los requisitos y casos de negocio de este.

- **Elaboración:** Se planifica el proyecto, especificando en detalle la mayoría de los casos de uso. También se diseña la arquitectura del proyecto.
- **Construcción:** En esta fase se construye el proyecto, implementando los requisitos establecidos para conseguir el producto final. Es la fase más larga del proyecto.
- **Transición:** En esta fase el producto tiene que estar disponible para ser probado y utilizado por el cliente. Cuando esta fase termine se empezarán a plantear futuras mejoras que hayan surgido en las revisiones.

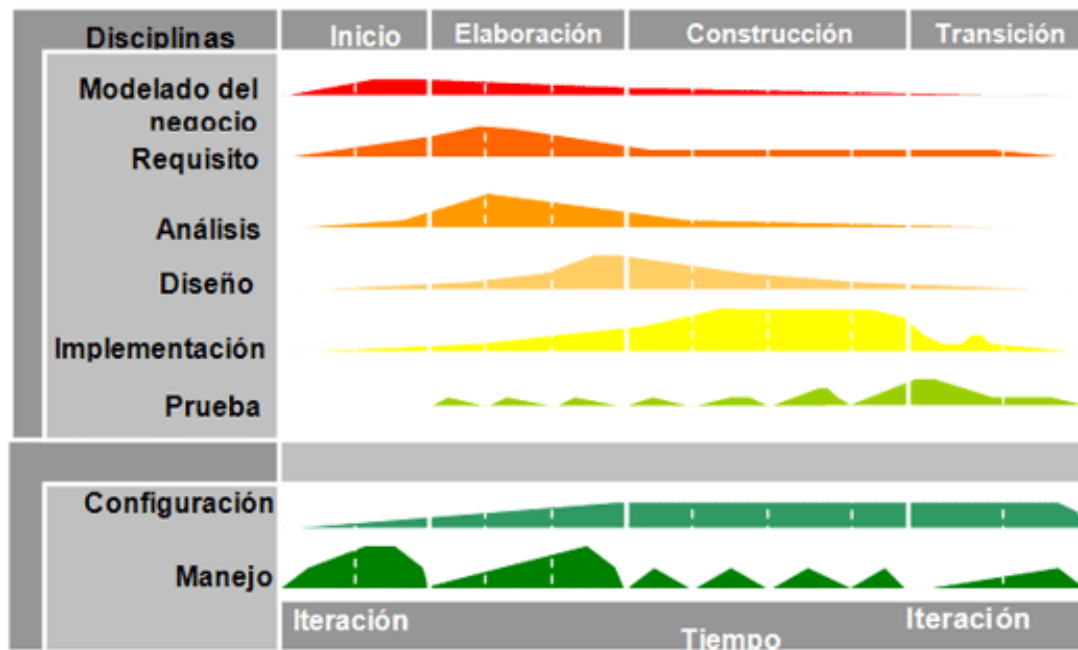


Figura 28: Proceso Unificado

7.2. Estimación del Esfuerzo

Tal y como se recoge en el anexo 1, la estimación del esfuerzo del proyecto trata de calcular cuánto tiempo se necesita para completar el proyecto en su totalidad. Para llevarla a cabo usaremos la métrica basada en puntos de casos de uso (UCP o Use Case Points). Esta métrica se basa en escenarios, actores y factores técnicos y de entorno para realizar el cálculo.

El primer paso era asignar la complejidad de los actores siguiendo la siguiente tabla:

Tabla 1: Complejidad de los actores

Complejidad	Descripción	Peso
Simple	Otro sistema que interactúa con el sistema mediante una API.	1
Media	Otro sistema que interactúa con el sistema mediante un protocolo, o persona que interactúa a través de una interfaz en modo texto	2
Compleja	Una persona que interactúa con el sistema mediante una interfaz gráfica.	3

Con esto hecho, se ha determinado la complejidad de los casos de uso mediante la siguiente tabla:

Tabla 2: Complejidad de los casos de uso

Complejidad	Descripción	Peso
Simple	<ul style="list-style-type: none">• Si el caso de uso tiene tres transacciones o menos• Si el caso de uso tiene cuatro clases o menos	5
Medio	<ul style="list-style-type: none">• Si el caso de uso tiene entre cuatro y siete transacciones• Si el caso de uso tiene entre cinco y diez clases	10
Complejo	<ul style="list-style-type: none">• Si el caso de uso tiene más de siete transacciones• Si el caso de uso tiene más de diez clases	15

El siguiente paso es establecer los valores de complejidad técnica. Para ello se ha seguido la siguiente tabla:

Tabla 3: Factores de complejidad técnica

Descripción	Complejidad	Explicación
Sistemas distribuidos	4	Se ha asignado un factor de complejidad técnica de cuatro, ya que el sistema incluye múltiples dispositivos IoT que operan de manera independiente del servidor central, lo que requiere una gestión fiable de la comunicación, sincronización de datos y gestión remota de dispositivos.
Rendimiento	2	Se ha asignado un factor de complejidad técnica de dos, ya que, aunque no es el punto más crítico, se requiere una cierta optimización para asegurar que las respuestas sean suficientemente rápidas y eficientes, especialmente para manejar datos de sensores en tiempo real.
Eficiencia del usuario final	1	Se ha asignado un factor de complejidad técnica de uno, ya que uno de los objetivos prioritarios es que el usuario se sienta cómodo utilizando el sistema, por lo que se utilizan pantallas familiares para los usuarios más inexpertos.
Procesamiento interno complejo	3	Se ha asignado un factor de complejidad técnica de tres, ya que el sistema tendrá que calcular la predicción de riego en base a los datos almacenados de los sensores.
Reusabilidad	2	Se ha asignado un factor de complejidad técnica de dos, ya que se espera una alta reusabilidad de componentes del sistema relativamente alta.
Facilidad de instalación	1	Se ha asignado un factor de complejidad técnica de uno, dado que la instalación del sistema es sencilla.
Facilidad de uso	2	Se ha asignado un factor de complejidad técnica de dos, ya que es importante que los usuarios, independientemente de su nivel de experiencia, puedan utilizar el sistema con facilidad y sin necesidad de formación extensa

Portabilidad	2	Se ha asignado un factor de complejidad técnica de dos, ya que el sistema debe ser compatible con múltiples plataformas y dispositivos.
Facilidad de cambio	2	Se ha asignado un factor de complejidad técnica de dos, ya que, aunque no es el objetivo principal, se busca que el sistema sea fácil de mantener y extender, permitiendo la adición de nuevas funcionalidades y mejoras sin grandes esfuerzos.
Concurrencia	3	Se ha asignado un factor de complejidad técnica de tres, ya que se espera que múltiples usuarios y dispositivos estén conectados simultáneamente al sistema, lo que requiere una arquitectura capaz de manejar esos niveles de concurrencia y asegurar la disponibilidad y rendimiento del servicio.
Características especiales de seguridad	1	Se ha asignado un factor de complejidad técnica de uno, ya que, aunque se deben proteger los datos de los usuarios y asegurar la autenticación, no se manejan datos altamente sensibles que requieran medidas de seguridad extremas.
Provee acceso directo a terceras partes	0	Se ha asignado un factor de complejidad técnica de cero, ya que el sistema no necesita integrarse directamente con servicios de terceros ni proporcionar APIs para acceso externo, simplificando la arquitectura y el desarrollo.
Se requiere entrenamiento especial del usuario	0	Se ha asignado un factor de complejidad técnica de cero, dado que el sistema está diseñado para ser intuitivo y familiar para los usuarios, sin necesidad de formación especial para utilizar sus funcionalidades básicas.

En cuanto a los factores de complejidad técnica se ha asignado la complejidad de estos factores en la siguiente tabla:

Tabla 4: Factores de complejidad del entorno

Factor	Complejidad	Motivo
Familiaridad con UML	3	Existe cierta familiaridad ya que se ha estudiado a lo largo de la carrera de Ingeniería Informática.
Trabajadores a tiempo parcial	4	El desarrollo del proyecto se ha de compaginar con el último año del grado y con una beca de formación.
Capacidad de los analistas	2	Existe cierta familiaridad con el análisis, pero al no haberla podido aplicar en proyectos reales falta experiencia
Experiencia en la aplicación	3	Se ha asignado un valor de tres ya que durante las prácticas de empresa he podido formarme en las tecnologías usadas para el desarrollo del proyecto
Experiencia en la orientación a objetos	3	Durante el grado se han utilizado lenguajes orientados a objetos.
Motivación	5	Hay una gran motivación detrás de este proyecto, ya que es una forma de aplicar los conocimientos adquiridos en los cuatro años de grado, además de ser el Trabajo Final de Grado.
Dificultad del lenguaje de programación	2	Se tiene cierta experiencia usando lenguajes como Python y Arduino gracias a las prácticas académicas y a asignaturas como Periféricos.
Estabilidad de los requisitos	4	Los requisitos se han establecido con anterioridad por lo que no se esperan cambios muy significativos.

El resultado final se ha calculado con la herramienta mencionada anteriormente, EZEstimate. El tiempo de desarrollo del proyecto se muestra en la siguiente figura:

Estimation Summary	
UAW	9
UUCW	80
UUPC = UAW + UUCW	89
TFactor	29
EFactor	17
TCF = 0.6 + (.01*TFactor)	0.89
EF = 1.4 + (-0.03*EFactor)	0.89
UCP = UUCP*TCT*EF	70,4969
Total Effort@ 7 Hrs/UCP	493,4783

Figura 29: Resultados de EzEstimate

7.3. Planificación temporal

Tal y como se recoge en el anexo 1, la planificación temporal permite organizar las tareas, asignar recursos y prever posibles retrasos. Para ello hemos seguido la referencia del Proceso Unificado. Esta divide el proyecto en cuatro fases:

- **Inicio:** Esta fase indica el comienzo del proyecto. En ella se identifican los requisitos y objetivos. Sirve para tener una idea general del proyecto.
- **Elaboración:** Es la segunda fase del proyecto. En ella se siguen buscando y definiendo requisitos, además de realizar algunos diagramas. También sirve para revisar las acciones realizadas en la fase anterior.
- **Construcción:** Esta es la fase en la que se realiza el proyecto, por lo que suele tener una mayor carga de trabajo. Aunque también sirve para revisar acciones en fases anteriores, se centra en la implementación de subsistemas.
- **Transición:** Es la fase final del proyecto. Se evalúan tanto las partes individuales como el conjunto para asegurarse de que todo funcione correctamente. También es la fase en la que se realiza la documentación.

Es una metodología iterativa e incremental ya que dentro de cada fase se pueden producir varias iteraciones.

Para la realización de la planificación se ha utilizado Microsoft Project, usado anteriormente para la asignatura de Gestión de Proyectos. En la siguiente figura se puede ver la fase de Inicio el calendario realizado.

Nombre de tarea ▼	Duración ▼	Comienzo ▼	Fin ▼	Predecesoras ▼
▣ Inicio	15,5 días	mié 14/02/24	vie 08/03/24	
▣ Iteración 1	15,5 días	mié 14/02/24	vie 08/03/24	
▣ Modelado de negocio	0,5 días	mié 14/02/24	mié 14/02/24	
Reunión de equipo	0,5 días	mié 14/02/24	mié 14/02/24	
▣ Requisitos	7 días	mié 14/02/24	dom 25/02/24	3
Establecer objetivos del sistema	2 días	mié 14/02/24	sáb 17/02/24	
Definir actores	1 día	sáb 17/02/24	lun 19/02/24	6
Establecer los requisitos de información	1 día	lun 19/02/24	mar 20/02/24	7
Establecer los requisitos funcionales	2 días	mar 20/02/24	vie 23/02/24	8
Establecer los requisitos no funcionales	1 día	sáb 24/02/24	dom 25/02/24	9
▣ Análisis	3 días	dom 25/02/24	vie 01/03/24	5
Análisis de tecnologías disponibles	1 día	dom 25/02/24	mar 27/02/24	
Análisis del mercado	2 días	mar 27/02/24	vie 01/03/24	15
▣ Diseño	5 días	vie 01/03/24	vie 08/03/24	14
Diseño de prototipo con Figma	5 días	vie 01/03/24	vie 08/03/24	
Fin Iteración 1	0 días	vie 08/03/24	vie 08/03/24	18
Fin Inicio	0 días	vie 08/03/24	vie 08/03/24	19
▣ Elaboración	27,5 días	vie 08/03/24	sáb 20/04/24	1
▣ Iteración 1	19 días	vie 08/03/24	dom 07/04/24	
▣ Modelo de negocio	2,5 días	vie 08/03/24	mar 12/03/24	
Reunión de equipo	0,5 días	vie 08/03/24	sáb 09/03/24	
Estimación temporal	2 días	sáb 09/03/24	mar 12/03/24	27
Estimación del esfuerzo	1 día	sáb 09/03/24	lun 11/03/24	27
▣ Requisitos	5 días	mar 12/03/24	mié 20/03/24	26
Diagrama de casos de uso	3 días	mar 12/03/24	dom 17/03/24	
Revisión de los requisitos	2 días	dom 17/03/24	mié 20/03/24	31
▣ Análisis	3 días	mié 20/03/24	dom 24/03/24	30
Paquete de análisis	2 días	mié 20/03/24	sáb 23/03/24	
Realización de diagramas de secuencia del paquete: Gestión de usuarios	1 día	sáb 23/03/24	dom 24/03/24	34

Figura 30: Microsoft Project. Inicio

En la siguiente figura se muestra el Diagrama de Gantt formado a partir del calendario realizado en Microsoft Project.

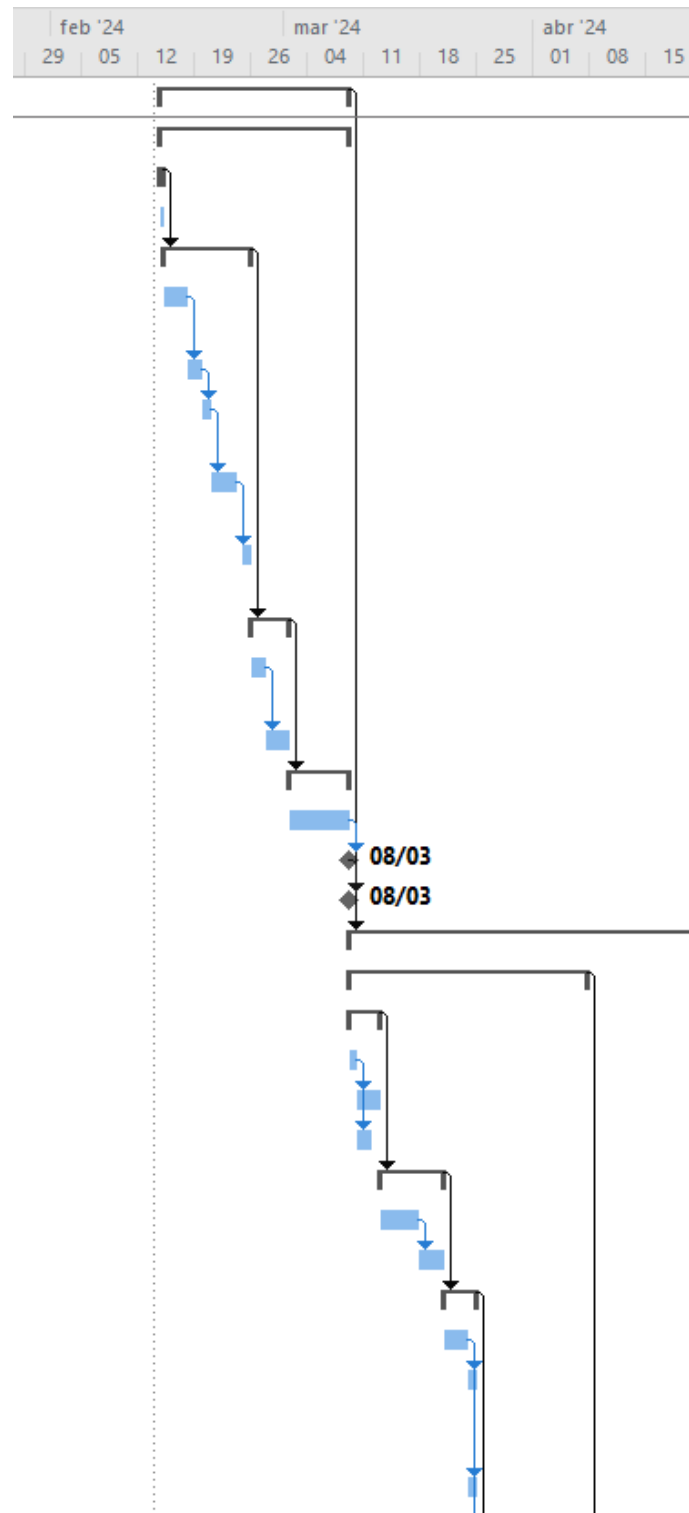


Figura 31: Diagrama de Gantt 1

Toda la estimación del esfuerzo y la planificación temporal se puede ver en el Anexo 1.

7.4. Especificación de requisitos

A especificación de requisitos sigue la metodología de Durán y Bernárdez, y se encuentra definida en el Anexo II, ya que en este apartado se han incluido ejemplos de la estructura que sigue el anexo.

7.4.1. Participantes:

El proyecto cuenta con tres participantes, dos tutores y un alumno:

- Diego Plata Klingler
- Sergio Alonso Rollán
- Javier Prieto Tejedor

Como ya se ha mencionado anteriormente, no se incluirán todas las tablas ni figuras, pero se podrán consultar en el Anexo II.

Tabla 5: Participante Diego Plata Klingler

Participante	Diego Plata Klingler
Organización	Universidad de Salamanca
Rol	Desarrollador
Es desarrollador	Sí
Es cliente	No
Es usuario	No
Comentarios	Ninguno

7.4.2. Objetivos del sistema:

Los objetivos que el sistema debe cumplir son:

- **Gestión de usuarios:** El sistema debe manejar el registro de usuarios que deseen inscribirse, así como permitir el inicio de sesión para aquellos que ya tienen una cuenta. Además, los usuarios registrados deben poder modificar sus datos personales y el contenido de su perfil.
- **Gestión de dispositivos:** El sistema deberá gestionar la conexión a los dispositivos IoT y garantizar que los datos enviados por éstos se almacenen correctamente en la base de datos.

- **Gestión de la planificación:** El sistema deberá ser capaz de realizar planificaciones y tomar decisiones automáticas de riego basándose en los datos recogidos y analizados, además del pronóstico meteorológico.

En la siguiente tabla se puede ver un ejemplo de las tablas de objetivos del sistema:

Tabla 6: OBJ-0001 Gestión de usuarios

OBJ-0001	Gestión de usuarios
Versión	1.0
Autor	Diego Plata Klingler
Fuentes	
Descripción	El sistema debe manejar el registro de usuarios que deseen inscribirse, así como permitir el inicio de sesión para aquellos que ya tienen una cuenta. Además, los usuarios registrados deben poder modificar sus datos personales y el contenido de su perfil.
Importancia	Alta
Urgencia	Alta
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

7.4.3. Requisitos de información

Este tipo de requisitos especifican qué datos son necesarios, cómo hay que obtenerlos, y cómo se almacenan. En nuestro sistema tenemos los siguientes:

- Información sobre los usuarios
- Información sobre los dispositivos
- Información de sensores

En la siguiente tabla vemos un ejemplo de las tablas de los requisitos de información que se encuentran en el Anexo II:

Tabla 7: IRQ-0001 Información sobre los usuarios

IRQ-0001	Información sobre los usuarios
Versión	1.0
Autor	Diego Plata Klingler
Fuentes	
Dependencias	[OBJ-001] Gestión de usuarios
Descripción	El sistema debe almacenar la información relevante de los usuarios para garantizar el acceso y correcto funcionamiento del mismo.
Datos específicos	<ul style="list-style-type: none">• Identificador• Nombre de usuario• Correo electrónico• Contraseña• Fecha de creación• Dispositivos registrados
Importancia	Alta
Urgencia	Alta
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno.

7.4.4. Requisitos de restricción de la información

Los requisitos de restricción de información establecen las limitaciones y controles necesarios para garantizar la correcta gestión y seguridad de los datos dentro del sistema.

En la siguiente tabla se puede ver la estructura de las tablas de requisitos de restricción de la información, encontradas en el Anexo II.

Tabla 8: CRO-0001 Correo electrónico

CRO-0001	Correo electrónico
Versión	1.0
Autor	Diego Plata Klingler
Fuentes	
Dependencias	[OBJ-0001] Gestión de usuarios
Descripción	El sistema no permitirá a los usuarios registrarse con una dirección de correo electrónico no válida.
Importancia	Alta
Urgencia	Alta
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

7.4.5. Requisitos funcionales

Los requisitos funcionales describen los comportamientos y capacidades que el sistema debe tener para cumplir con los objetivos propuestos. Estos requisitos se han agrupado en paquetes para tener una mejor organización del sistema:

- Gestión de usuarios
- Gestión de dispositivos
- Gestión de la planificación

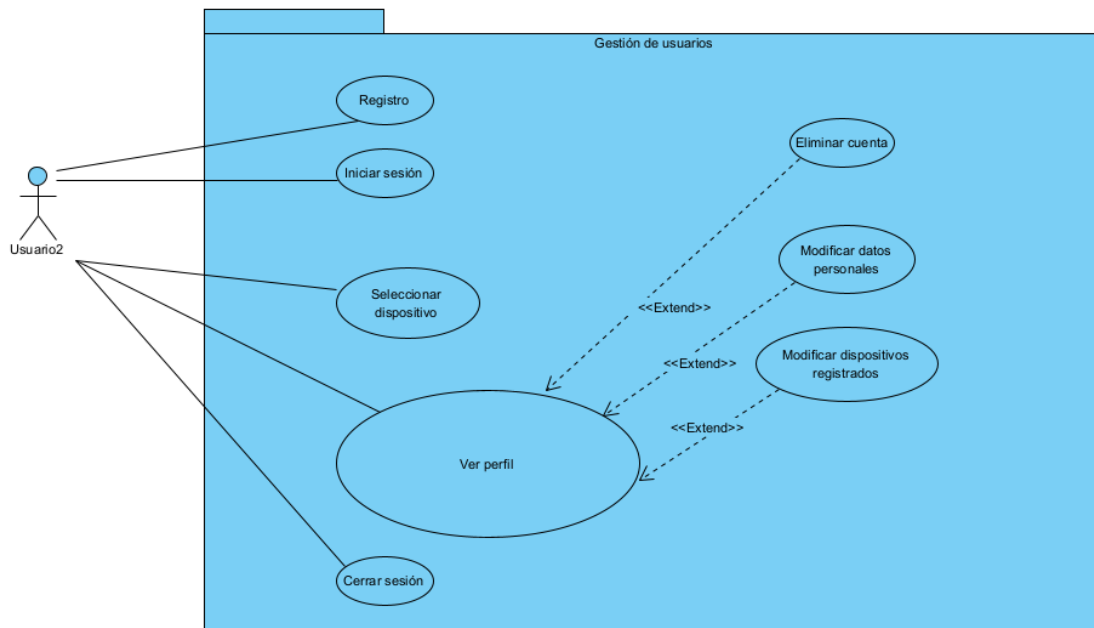


Figura 32: Paquete Gestión de usuarios

Dentro de cada paquete se encuentran varios requisitos funcionales. Para cada uno de ellos se han realizado tablas de casos de uso que los explican. En la siguiente tabla veremos un ejemplo de tabla de caso de uso:

Tabla 9: UC-0008 Eliminar cuenta

UC-0008	Eliminar cuenta	
Versión	1.0	
Autor	Diego Plata Klingler	
Fuentes		
Dependencias	[OBJ-0001] Gestión de los usuarios [IRQ-0001] Información sobre los usuarios	
Descripción	El sistema deberá comportarse tal como se describe en el siguiente caso de uso cuando el usuario solicite eliminar su cuenta del sistema.	
Precondiciones	El usuario tiene que estar registrado en el sistema	
Secuencia normal	Paso	Acción
	1	El usuario solicita al sistema eliminar su cuenta.
	2	El sistema le pide al usuario que introduzca su contraseña.
	3	El usuario introduce su contraseña correctamente
	4	El sistema borra todos los datos del usuario
Postcondiciones	El usuario registrado [ACT-0002] pasa a ser un usuario no registrado [ACT-0001]	
Excepciones	3	El usuario introduce una contraseña incorrecta
	4	El sistema cancela la operación de eliminar cuenta
Frecuencia	1 vez al mes	
Importancia	Alta	
Urgencia	Alta	
Estado	Validado	
Estabilidad	Alta	
Comentarios	Ninguno	

Los actores que van a interactuar con el sistema son los siguientes:

- **Usuario sin registrar:** Este actor representa a las personas que no tengan cuenta creada en el sistema. Únicamente podrán acceder a la pantalla de inicio.
- **Usuario registrado:** Este actor representa a las personas con una cuenta creada en el sistema. Podrán acceder al sistema en su totalidad.

- **Sistema:** Este actor representa al propio sistema, encargado de realizar las planificaciones y cálculos necesarios.
- **Dispositivo:** Este actor representa los dispositivos IoT existentes, englobando los sensores y actuadores de estos.

A continuación, se muestra una tabla de ejemplo de las tablas actores:

Tabla 10: ACT-0004 Dispositivo

ACT-0004	Dispositivo
Versión	1.0
Autor	Diego Plata Klingler
Fuentes	
Descripción	Este actor representa los dispositivos IoT existentes, englobando los sensores y actuadores de estos.
Comentarios	Ninguno

7.4.6. Requisitos no funcionales

Los requisitos no funcionales describen las características y restricciones de calidad que el sistema debe cumplir.

En nuestro sistema se han establecido los siguientes:

- **Disponibilidad:** El sistema debe ofrecer servicio la mayor cantidad de tiempo posible, excluyendo los períodos de mantenimiento.
- **Portabilidad:** El sistema debe ser capaz de ejecutarse en múltiples sistemas operativos sin necesidad de grandes cambios.
- **Mantenimiento:** Se deben asegurar revisiones cada cierto tiempo, tanto del servidor como de los sensores y sus conexiones, para garantizar su funcionamiento.

En la siguiente tabla se puede ver un ejemplo de tabla de requisitos no funcionales:

Tabla 11: NFR-0001 Disponibilidad

NFR-0001	Disponibilidad
Versión	1.0
Autor	Diego Plata Klingler
Fuentes	
Dependencias	Ninguno
Descripción	El sistema debe ofrecer servicio la mayor cantidad de tiempo posible, excluyendo los períodos de mantenimiento.
Importancia	Alta
Urgencia	Alta
Estado	Validado
Estabilidad	Alta
Comentarios	Ninguno

El resto de las tablas de las que se han mostrado solamente un ejemplo se encuentran disponibles en el Anexo II.

7.5. Análisis de requisitos

En este apartado se va a documentar la etapa de análisis de los requisitos del sistema mencionados en el apartado anterior. Se ha seguido la metodología de Durán y Bernárdez para obtener información acerca del dominio. Para la realización de los diagramas del anexo se ha usado la herramienta UML Visual Paradigm.

En esta memoria solo se incluirá un ejemplo de cada sección. Para ver toda la documentación completa, consultar el Anexo III.

7.5.1. Modelo de dominio

El modelo de dominio tiene como objetivo representar los conceptos clave del mundo real para el dominio del problema, incluyendo los datos relevantes o atributos y las relaciones entre ellos. Este modelo conceptualiza las clases de datos utilizadas por la aplicación, mostrando sus atributos y las interrelaciones.

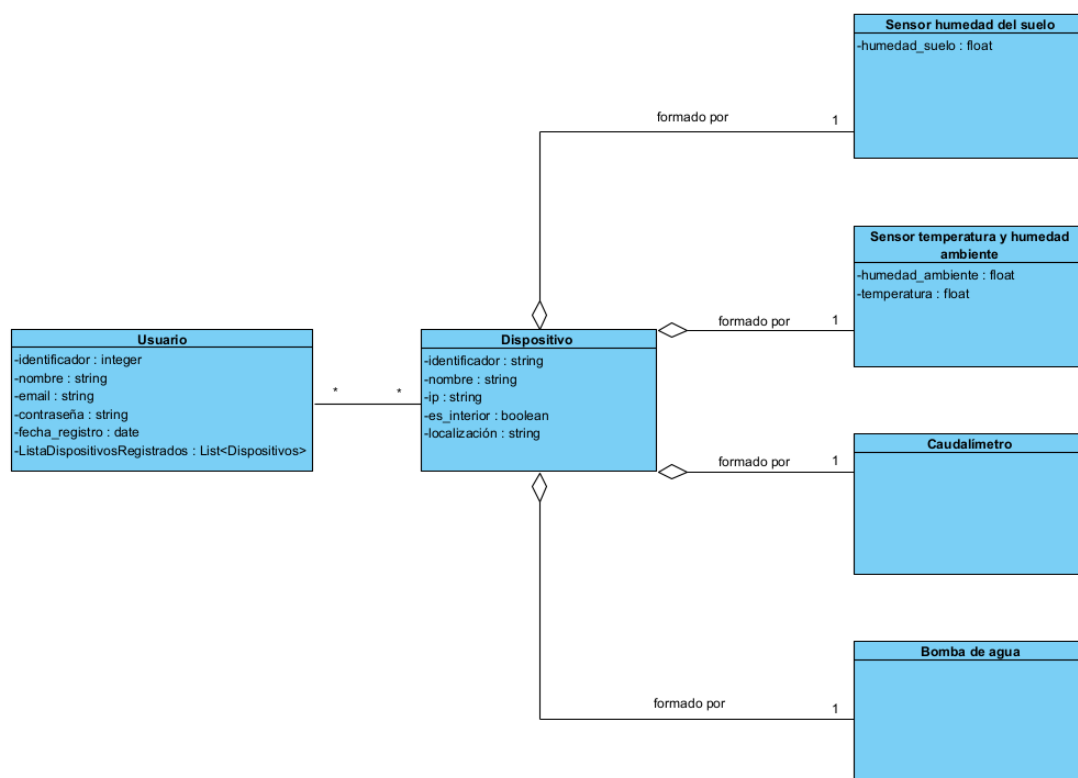


Figura 33: Modelo de dominio del sistema

7.5.2. Realización de casos de uso

Como ya se ha comentado, no se añadirán todos los diagramas de secuencia de los casos de uso, sino que solamente se mostrará un ejemplo para mostrar su funcionamiento.

Los casos de uso los dividimos en tres paquetes:

- Gestión de usuarios
- Gestión de dispositivos
- Gestión de la planificación

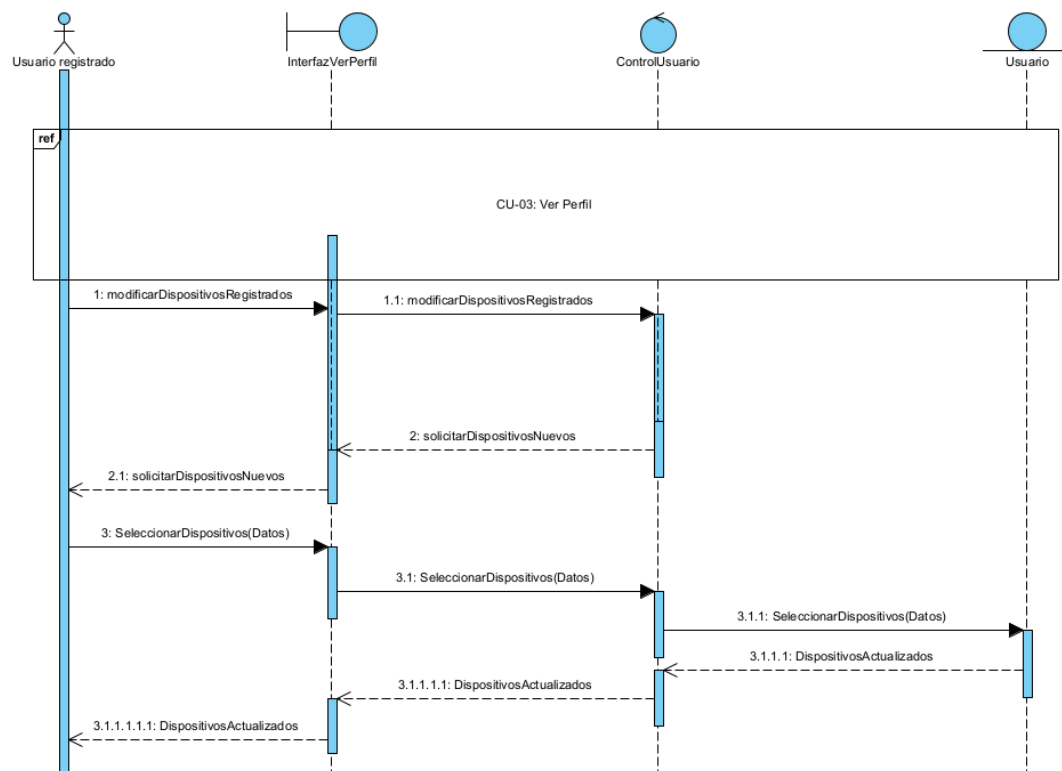


Figura 34: Diagr secuencia CU-0004 Modificar dispositivos registrados

7.5.3. Clase de análisis

Este apartado busca documentar la comunicación y relación entre los módulos del sistema mencionados anteriormente. La siguiente figura muestra el esqueleto que siguen todos los diagramas de comunicación.

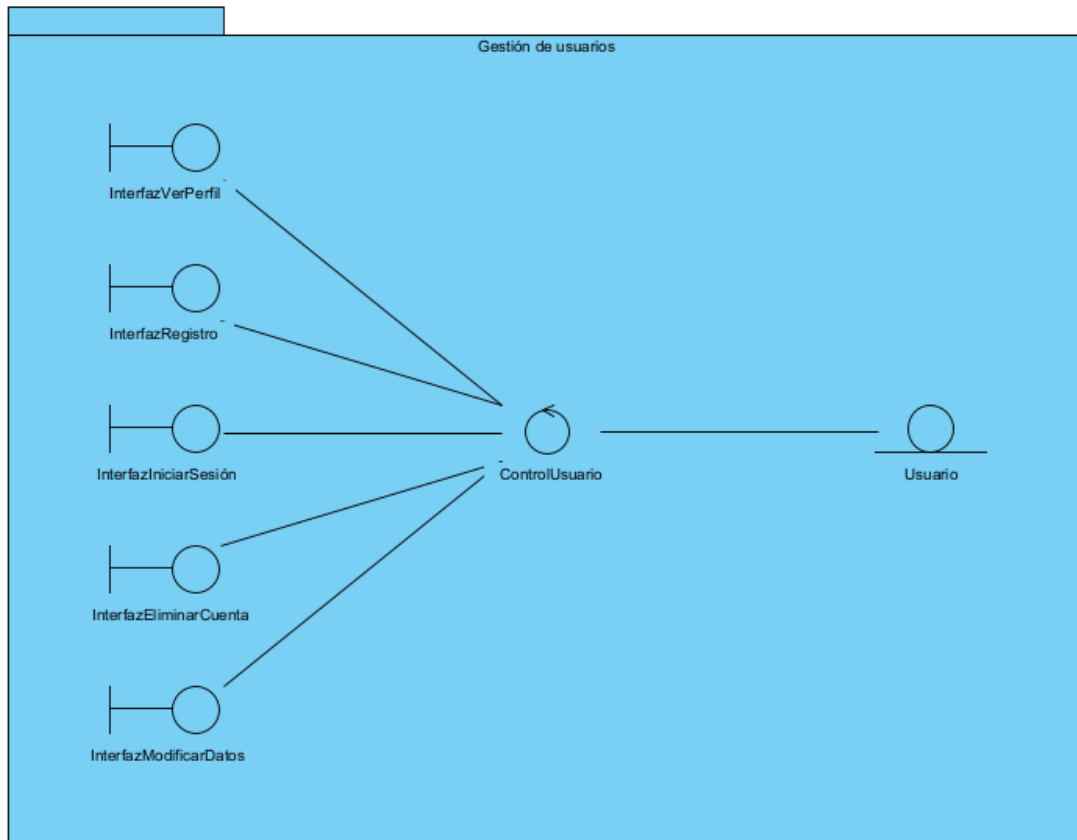


Figura 35: Diagrama de comunicación Gestión de usuarios

7.6. Diseño del sistema

Con el análisis de requisitos de la fase anterior se realiza el diseño del sistema. Esto incluye el diseño de la interfaz, el diseño de la base de datos, y el modelo de despliegue.

Como en apartados anteriores, en este apartado solo se hará un resumen para mostrar la estructura que se ha seguido. Para encontrar todo el proceso con detalle, consultar el Anexo IV.

7.6.1. Diseño de la interfaz

En este apartado se va a explicar el procedimiento para crear la interfaz gráfica de usuario, es decir, los HTML con los que va a interactuar el usuario.

El primer paso fue escoger los colores que iba a tener el sistema. Al tratarse de un dispositivo de irrigación, se ha intentado mantener la temática relacionada con los principales elementos de riego, que son el agua y las plantas. Por ello, se ha elegido una gama de colores azules y verdes:



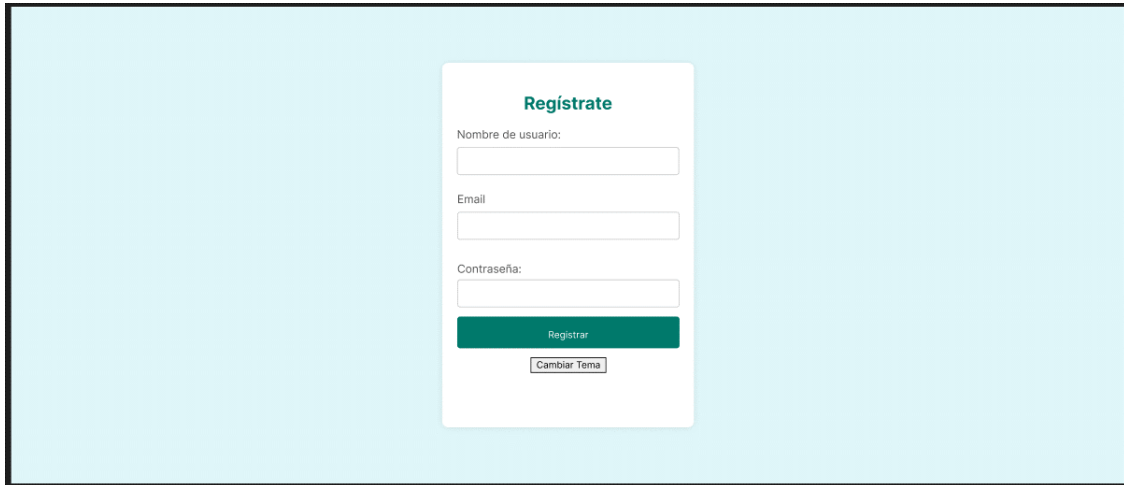
Figura 36: Código de colores seleccionados

El azul cian se seleccionó para el fondo, transmitiendo una sensación de limpieza y frescura. El verde oscuro se eligió ya que proporciona el contraste suficiente con el azul cian y con letras blancas, además de que se relaciona con el verde de las hojas de las plantas.

Con los colores elegidos, se procede con la creación de un prototipo digital usando la aplicación Figma vista en la asignatura de Interacción Persona-Ordenador. Con ella

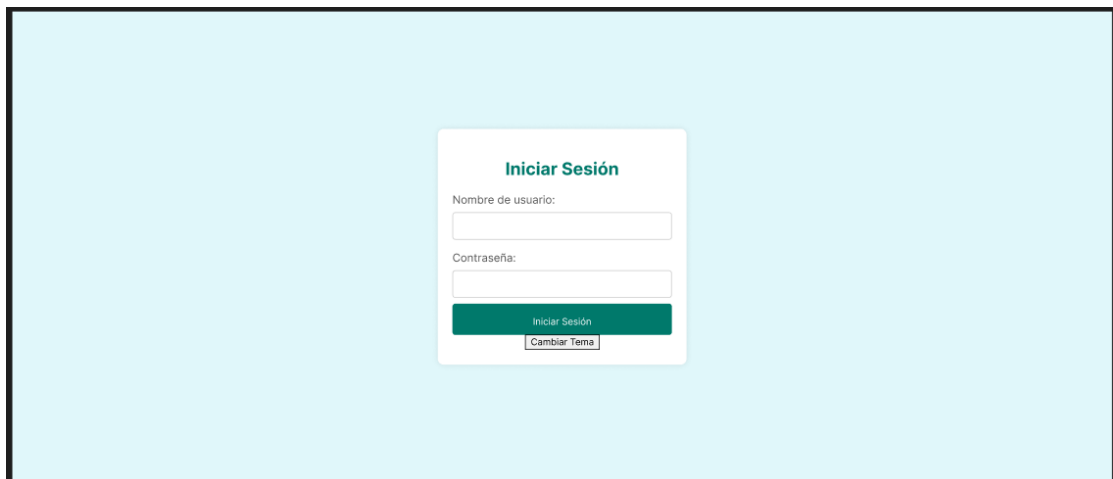
creamos un boceto de las pantallas de la aplicación web, que nos servirá para comprobar cómo se interactúa con el sistema antes de empezar la programación.

En las siguientes imágenes se adjuntan capturas del prototipado digital realizado mediante Figma:



El prototipo muestra una ventana de registro centrada en una pantalla de fondo azul claro. El formulario tiene un título 'Regístrate' en verde. Incluye tres campos de entrada: 'Nombre de usuario:', 'Email' y 'Contraseña:'. Debajo de los campos hay un botón verde 'Registrar' y un enlace 'Cambiar Tema' en un botón gris.

Figura 37: Prototipado digital. Ventana Registro. Tema claro



El prototipo muestra una ventana de inicio de sesión centrada en una pantalla de fondo azul claro. El formulario tiene un título 'Iniciar Sesión' en verde. Incluye dos campos de entrada: 'Nombre de usuario:' y 'Contraseña:'. Debajo de los campos hay un botón verde 'Iniciar Sesión' y un enlace 'Cambiar Tema' en un botón gris.

Figura 38: Prototipado digital. Ventana Iniciar Sesión. Tema claro

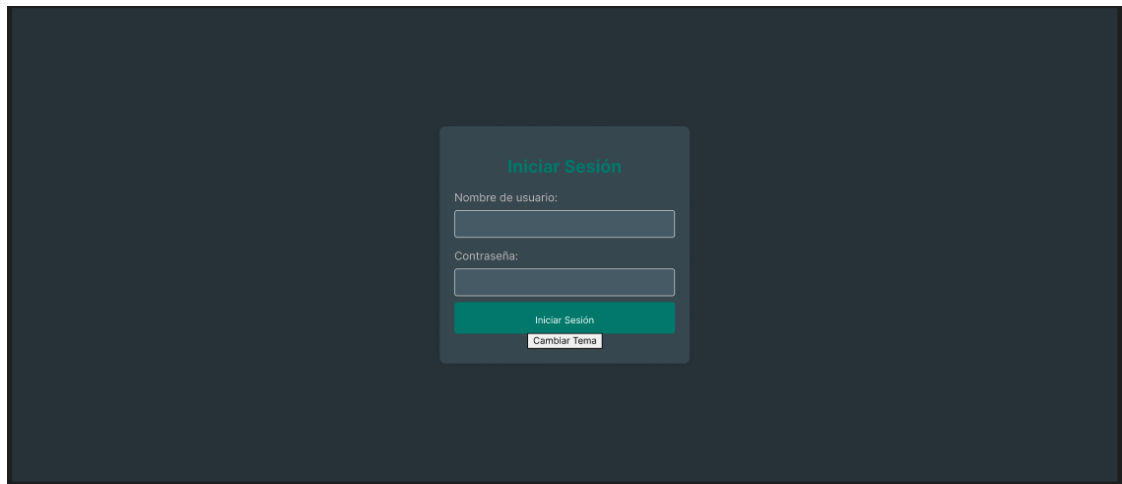


Figura 39: Prototipado digital. Ventana Iniciar Sesión. Tema oscuro

Se ha añadido también la funcionalidad de cambiar el tema a cada una de las ventanas, ya que para muchos usuarios es más cómodo trabajar con un fondo oscuro. No se agregarán visualizaciones de todas, pero están implementadas.

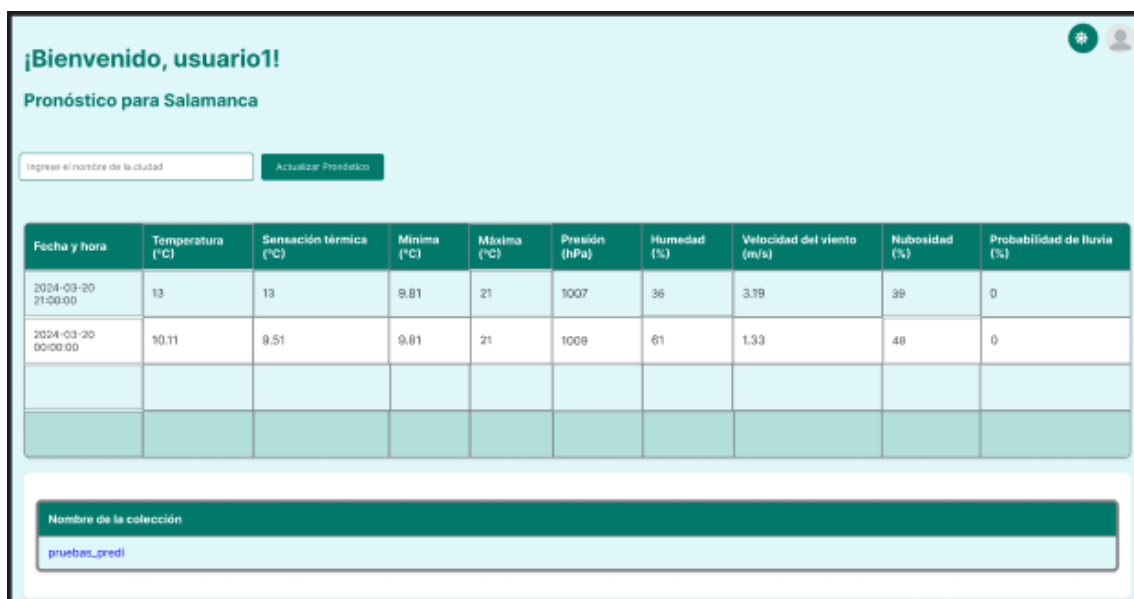


Figura 40: Prototipado digital. Ventana Principal

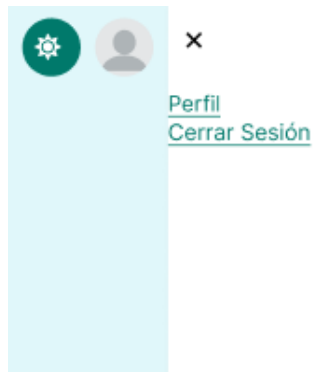


Figura 41: Prototipado digital. Barra lateral

Perfil de Usuario

Datos Actuales del Usuario:

Nombre de Usuario: usuario1
Email: usuario1@gmail.com

Modificar Datos del Perfil:

Nuevo Nombre:

Nuevo Email:

Nueva Contraseña:

Actualizar Perfil

Dispositivos Registrados:

pruebas_predi

Dispositivos Disponibles. Selecciona todos a los que quieras registrarte:

☐ Isaac

☐ Prueba2

☐ maceta4

☐ prueba

☐ pruebas_predi

Guardar Cambios

Figura 42: Prototipado digital. Ventana modificar perfil



Figura 43: Protipado digital. Ver datos de dispositivo

7.6.2. Modelo de diseño

En este apartado se explicarán los patrones arquitectónicos que se han usado para facilitar el desarrollo, mientras que se asegura que se utilizan buenas prácticas de programación. También se van a explicar las clases de diseño, la vista arquitectónica y la realización de casos de uso.

Las tecnologías empleadas para el desarrollo de software son las siguientes:

- **Flask:** Flask es un framework web escrito en Python que permite desarrollar aplicaciones web. Se ha usado para el desarrollo del Backend del servidor web.
- **HTML:** Se ha utilizado para desarrollar las interfaces de usuario (Frontend) con las que interactuará el usuario en el servidor
- **MariaDB:** Base de datos relacional que guardará los datos de los usuarios y los dispositivos.
- **MongoDB:** Base de datos no relacional que guardará las mediciones de los sensores de los dispositivos.
- **Arduino:** Utilizado para el desarrollo del código del dispositivo. Se encarga de recoger las mediciones de los sensores y mandarlas al servidor por MQTT.

7.6.3. Patrones arquitectónicos

El patrón arquitectónico empleado en este proyecto ha sido el MVC (Modelo-Vista-Controlador), que sirve para separar las capas que componen la aplicación.

El **modelo** gestiona la lógica de negocio y la interacción con las bases de datos.

La **vista** se encarga de la presentación de la interfaz de usuario mediante los ficheros HTML.

El **controlador** actúa como intermediario, procesando las solicitudes del usuario, coordinando las interacciones entre el modelo y la vista, y gestionando la lógica de programación.

El uso de este patrón facilita la escalabilidad y mantenibilidad del código a medida que se añaden funcionalidades y mantenimiento.

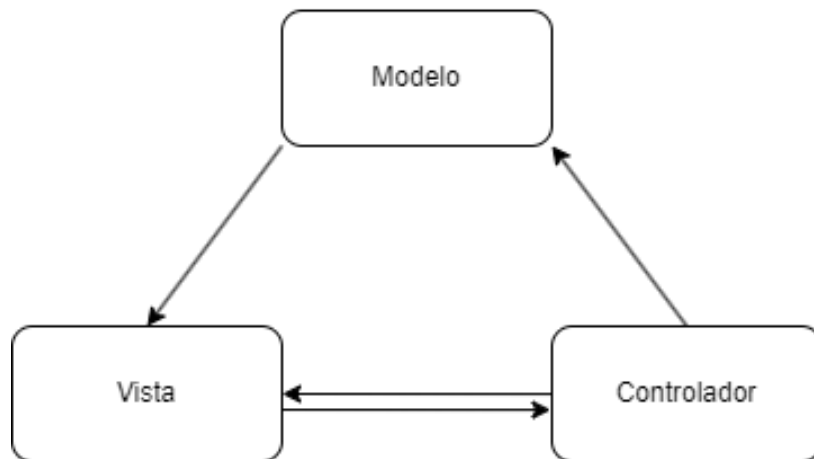


Figura 44: Diagrama MVC

7.6.4. Subsistemas de diseño

Siguiendo en el patrón MCV del apartado anterior, se divide el sistema en diferentes subpaquetes más específicos para tener elementos más manejables. Tendremos los siguientes paquetes:

- **Paquete Servidor:** Contiene la lógica de negocio y la infraestructura de gestión de datos del servidor Flask. Se encarga de almacenar los datos de la base de datos de usuarios y dispositivos, y la lógica para mostrar los datos al usuario. Además, el paquete servidor incluye la gestión de autenticación y autorización de usuarios.
- **Paquete Dispositivo:** Incluye la lógica de control del dispositivo IoT, que comprende la configuración de los sensores, la recogida de datos, y la comunicación con el servidor Flask a través del protocolo MQTT. Este paquete también gestiona la transmisión de datos al servidor.

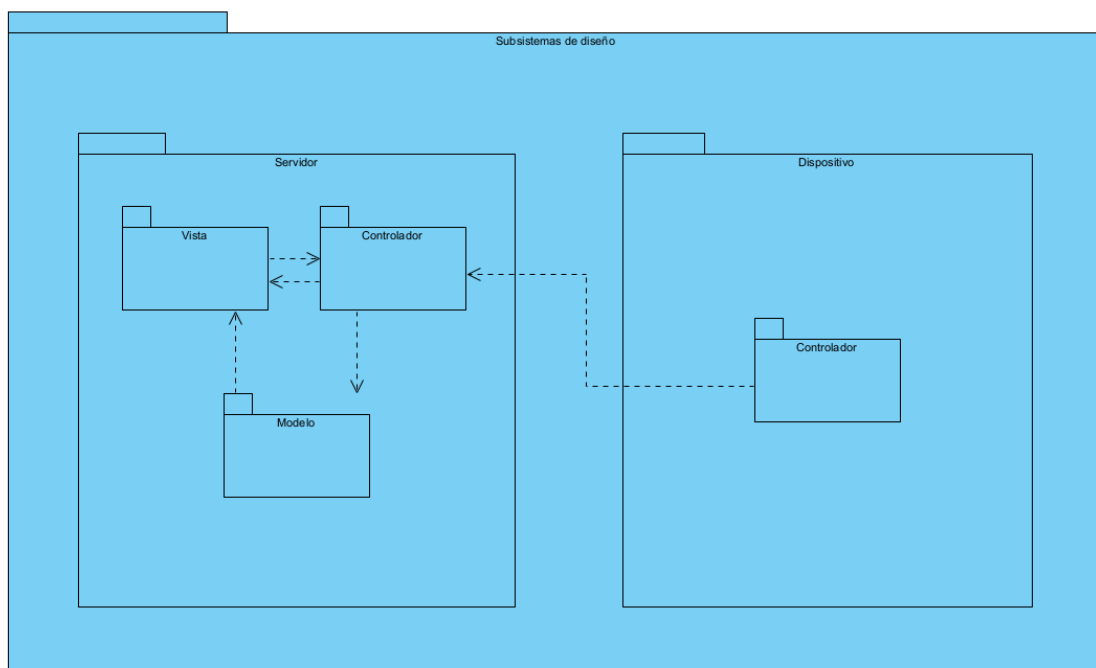


Figura 45: Subsistemas de diseño

7.6.5. Clases de diseño

En las clases de diseño se van a especificar las clases de diseño de los subpaquetes que forman el patrón MVC. Como buscamos un sistema con bajo acoplamiento y alta cohesión, dividiremos cada paquete en subsistemas más pequeños.

Dentro del paquete servidor tenemos tres subsistemas, que son el Modelo, la Vista y el Controlador. En la siguiente figura vamos a ver el diagrama de uno de ellos. El resto se encuentran especificados en el Anexo IV.

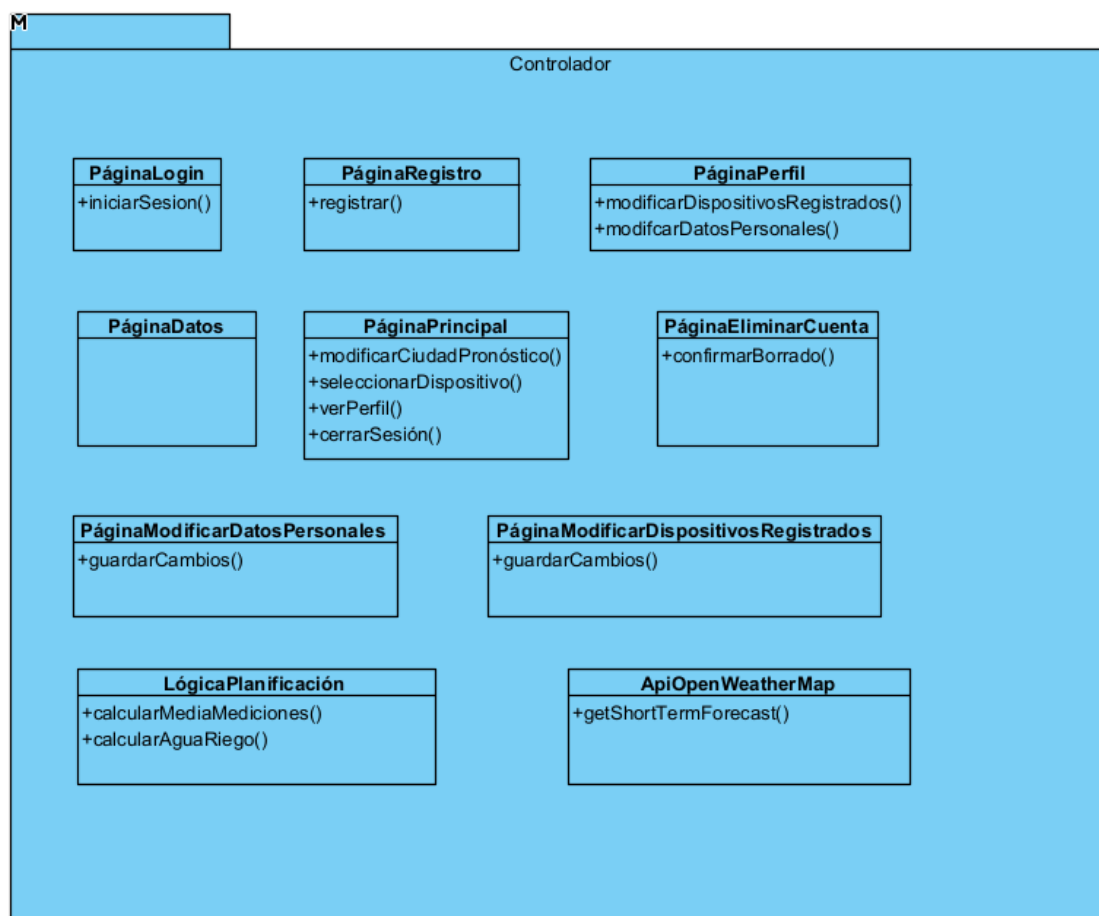


Figura 46: Subsistema Controlador

Por otro lado, el sistema cuenta también con el paquete de dispositivo. Este, aunque un poco más simple que el servidor, también cuenta un subpaquete controlador detallado en el Anexo IV.

7.6.6. Vista arquitectónica

En este apartado se muestra el patrón MVC del sistema de forma más detallada.

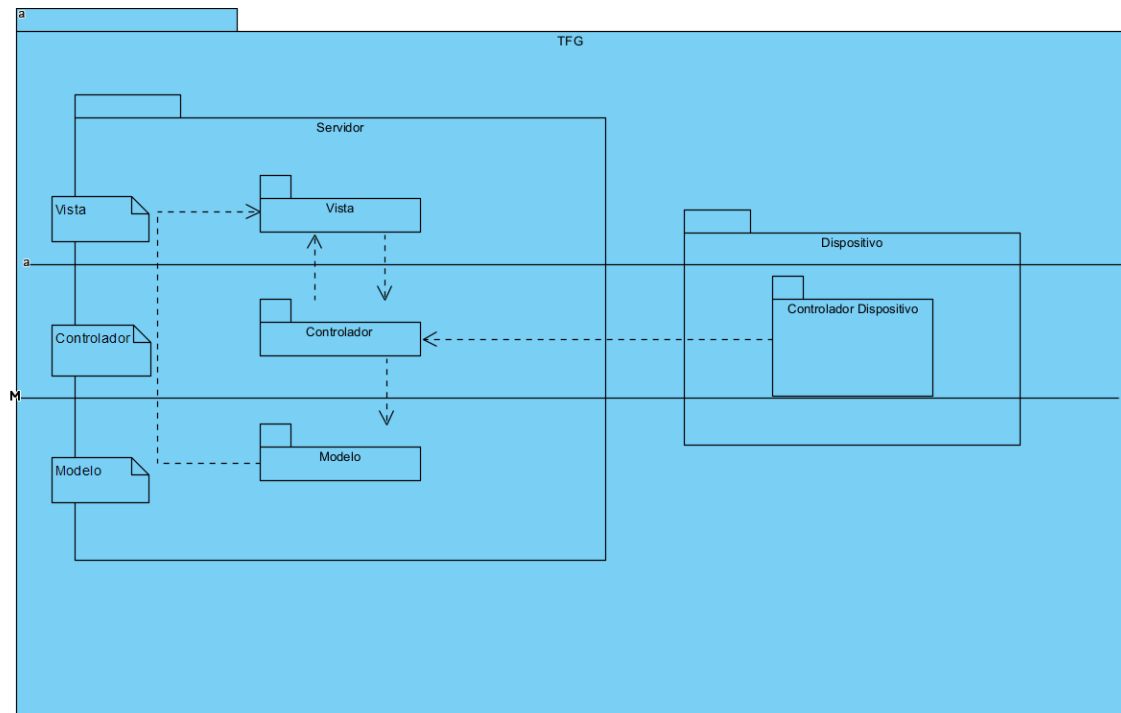


Figura 47: Vista arquitectónica MVC

7.6.7. Realización de casos de uso

En este apartado se han mejorado los diagramas de secuencia realizados en el análisis de requisitos. Se ha partido de las clases de diseño de los apartados anteriores hasta llegar a una aproximación final de los métodos que se van a emplear.

En la siguiente figura se plantea un ejemplo de la estructura de estos diagramas:

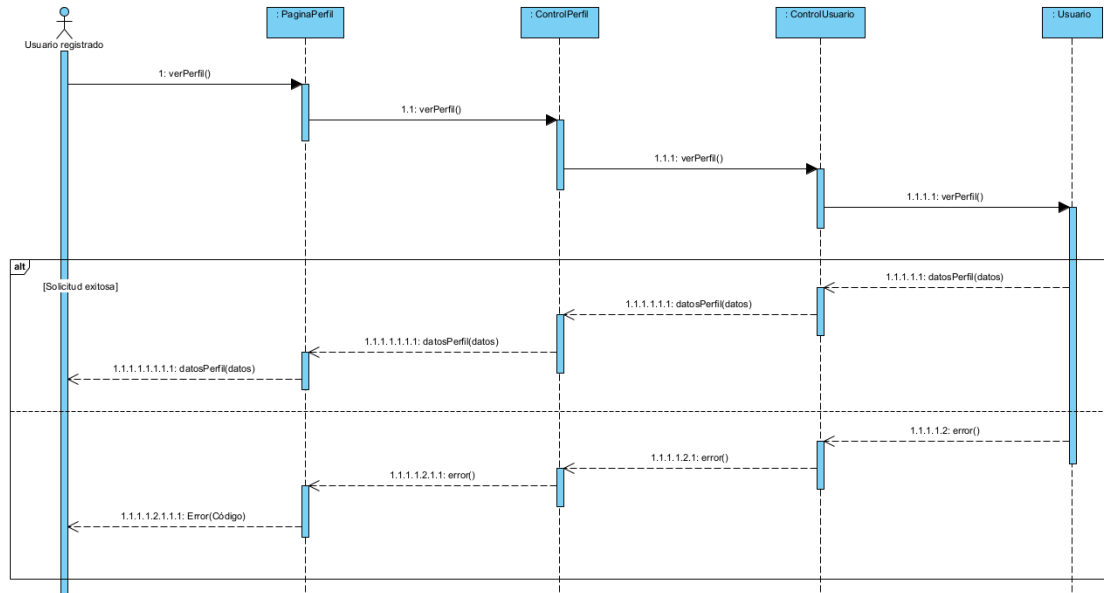


Figura 48: Diagrama de secuencia CU-0003 Ver perfil

7.6.8. Diseño de la base de datos

Para que el sistema sea capaz de almacenar correctamente toda la información se usarán dos tipos de bases de datos, una base de datos NoSQL como MongoDB, y una base de datos SQL como MariaDB. Para ello vamos a explicar previamente las diferencias entre ambos tipos:

- **Base de datos relacional (SQL):** Las bases de datos relacionales organizan los datos en tablas. Cada tabla tiene una estructura predefinida que define la estructura de los datos, incluyendo tipos de datos y relaciones entre tablas mediante claves (primarias y externas). Este tipo de base de datos requieren soporte para transacciones ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad).
- **Base de datos no relacional (NoSQL):** Por otro lado, las bases de datos no relacionales son más flexibles en cuanto al almacenamiento de datos. No usan tablas con modelos fijos, sino que usan modelos de datos como documentos, columnas o pares clave-valor. Esto permite manejar datos sin estructurar de manera eficiente. Este tipo de datos son idóneas para aplicaciones que manejan grandes cantidades de datos.

Con los conceptos clave explicados, procedemos a justificar la elección de las bases de datos para cada dato almacenado en nuestro sistema.

- **MariaDB:** En nuestro sistema tenemos una base de datos MariaDB con dos tablas diferentes, ya que en una almacenamos la información sobre los usuarios y en otra almacenamos la información sobre los dispositivos. Se ha elegido esta base de datos por varios motivos. En primer lugar, MariaDB ofrece un modelo relacional que permite organizar los datos en tablas con modelos predefinidos. Esto viene bien para almacenar dos tipos distintos de datos en nuestro sistema, como los usuarios y los dispositivos. Por otro lado, MariaDB proporciona un sistema de seguridad avanzado, asegurando la confidencialidad de los datos. Por último, como tiene soporte para las transacciones ACID, permite garantizar la consistencia y fiabilidad de los datos.
- **MongoDB:** En cuanto a la base de datos NoSQL, usamos MongoDB para almacenar las mediciones de los sensores de cada dispositivo ya que permite mayor flexibilidad a los cambios. Además, MongoDB maneja muy bien grandes cantidades de datos, y las operaciones de escritura son rápidas, lo cual es ideal para un dispositivo IoT como el de este proyecto ya que las escrituras de las

mediciones son muy frecuentes. Por último, MongoDB permite guardar los datos en formato JSON, que es como lo manda el propio dispositivo.

A continuación, se muestran las bases de datos para apreciar mejor la estructura:

```
MariaDB [usuarios_tfg]> select * from dispositivo;
```

identificador	nombre	ip	es_interior	ciudad
123456789	Interior1	192.168.1.1	1	New York
150481616	maceta4	192.168.137.12	1	Zurich
208286571	prueba	192.168.137.12	1	prueba
222222222	Exterior2	192.168.1.4	0	Tokyo
248987701	PruebaTFG	192.168.137.85	1	Madrid
277693621	VideoExplicacion	192.168.137.85	0	Salamanca
347737362	Prueba2	192.168.137.12	0	Salamanca
398104427	Explicacion	192.168.137.85	0	Salamanca
499507726	pruebas_predi	192.168.137.173	0	Rio de Janeiro
673021246	Video	192.168.137.85	0	Salamanca
987654321	Interior2	192.168.1.2	1	London

```
11 rows in set (0.000 sec)
```

```
MariaDB [usuarios_tfg]> describe dispositivo;
```

Field	Type	Null	Key	Default	Extra
identificador	int(11)	NO	PRI	NULL	
nombre	varchar(50)	NO	UNI	NULL	
ip	varchar(15)	NO		NULL	
es_interior	tinyint(1)	NO		NULL	
ciudad	varchar(15)	NO		NULL	

```
5 rows in set (0.022 sec)
```

Figura 49: Colección Usuarios. BD SQL

```
MariaDB [usuarios_tfg]> select * from usuario;
```

id	username	password	mail	fecha_registro	dispositivos_registrados_id
28	usuariol	\$2b\$12\$a6HooxSPUkhbkDnC0TkGKe3lld6j.uziMchg2G3e84ewhlsQQ.QLi	usuariol@gmail.com	2024-07-02 15:43:24	VideoExplicacion,maceta4

```
1 row in set (0.001 sec)
```

```
MariaDB [usuarios_tfg]> describe usuario;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
username	varchar(50)	NO	UNI	NULL	
password	varchar(100)	NO		NULL	
mail	varchar(120)	NO	UNI	NULL	
fecha_registro	datetime	YES		current_timestamp()	
dispositivos_registrados_id	varchar(255)	YES		NULL	

```
6 rows in set (0.014 sec)
```

Figura 50: Colección Dispositivos. BD SQL

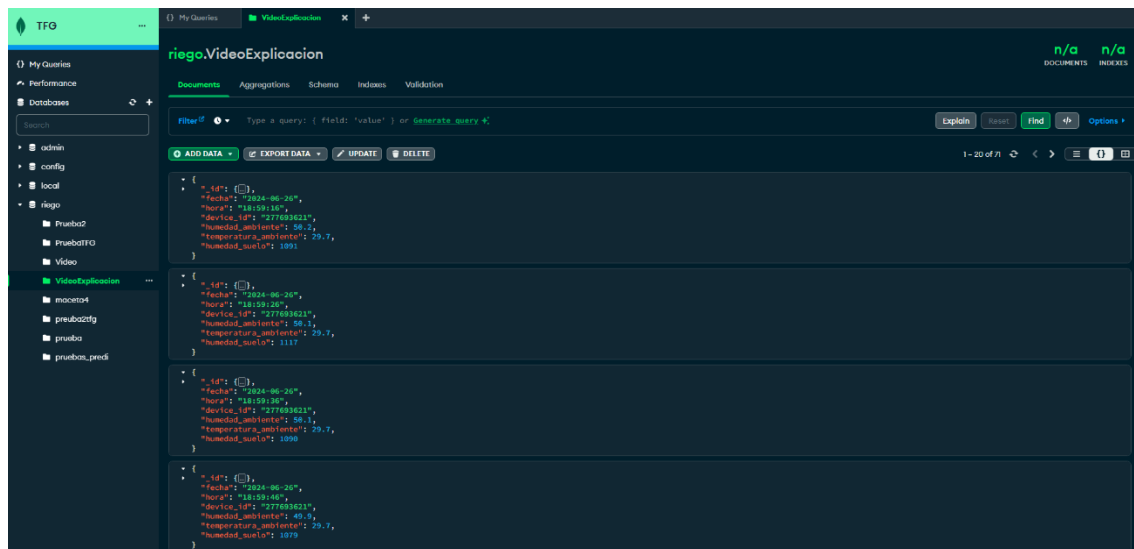


Figura 51: Mediciones. BD NoSQL

En las dos primeras fotografías se muestra MariaDB, la base de datos relacional. Tiene una tabla para los usuarios y una tabla para los dispositivos.

En la tercera foto se muestra la base de datos no relacional MongoDB. En la base de datos TFG se crea una colección para cada dispositivo, que almacenará los datos del dispositivo que recoja los datos.

7.6.9. Modelo de despliegue

Para representar el despliegue de los componentes se ha elaborado un diagrama de despliegue que muestra la arquitectura y relación física entre los distintos elementos.

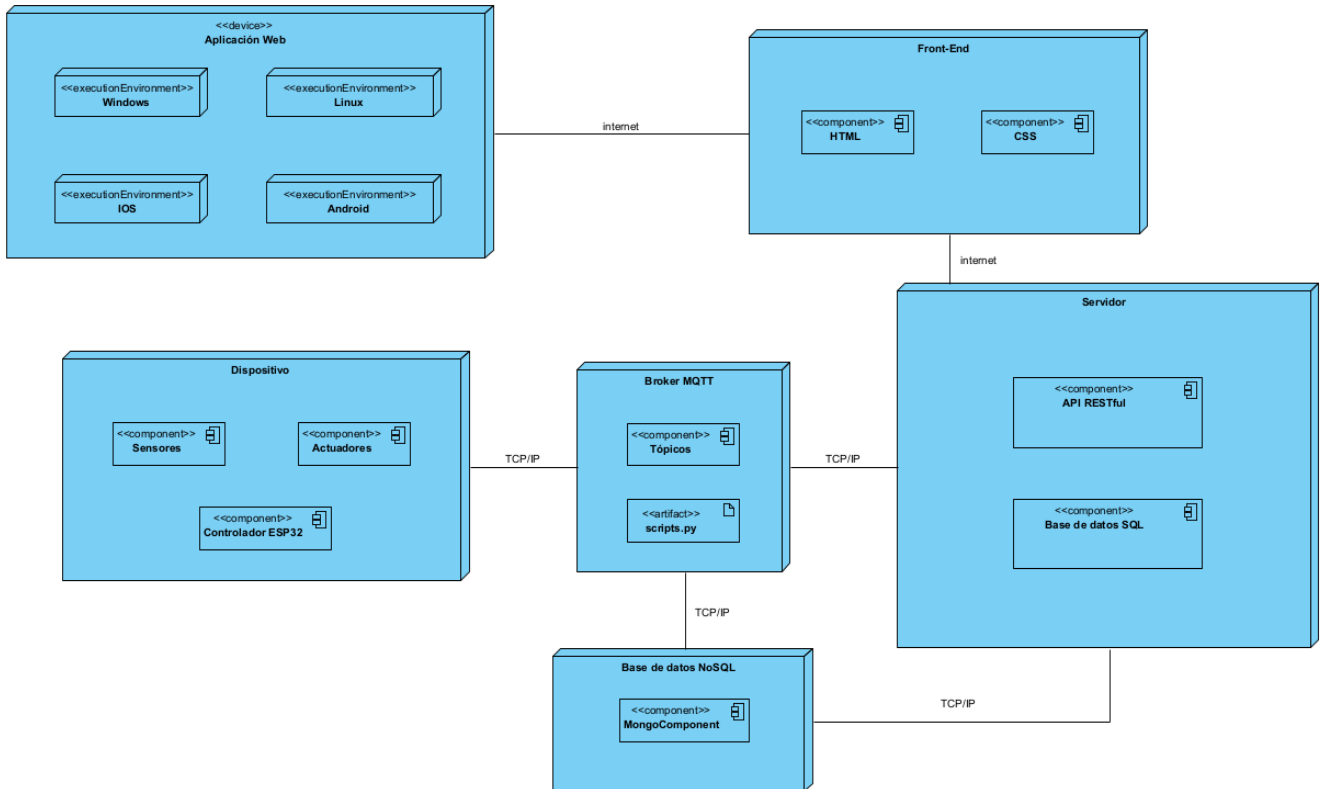


Figura 52: Modelo de despliegue

7.7. Implementación

En este apartado se describen los detalles sobre el desarrollo del software, tanto la parte del Backend como la parte de Frontend. Dentro del **Backend** distinguimos tres subsistemas distintos: Servidor, Dispositivo y Docker:

- **Servidor:** se encarga de realizar ciertos cálculos necesarios para el correcto funcionamiento del sistema, entre los que se encuentran los cálculos para las predicciones o la validación de los datos. La estructura del servidor es bastante sencilla, ya que se ha dividido en paquetes, cada uno con su funcionalidad. Dentro de estos paquetes tenemos un fichero con los endpoints del servidor. Un endpoint es una url a la que se hacen peticiones para interactuar con el servidor. Un ejemplo puede ser el endpoint de /login, que inicia sesión en una cuenta con los datos introducidos en el formulario.

```
@usuarios_bp.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form.get('username')
        password = request.form.get('password')

        usuario = Usuario.query.filter_by(username=username).first()

        if usuario and bcrypt.checkpw(password.encode('utf-8'), usuario.password.encode('utf-8')):
            login_user(usuario)
            flash('Inicio de sesión exitoso', 'success')
            return redirect(url_for('usuarios.home'))
        else:
            flash('Nombre de usuario o contraseña incorrectos', 'danger')
            return render_template('login.html', error=True)

    return render_template('login.html', error=False)
```

Figura 53: Ejemplo de endpoint

- **Dispositivo:** La estructura del dispositivo es muy sencilla, ya que únicamente se trata de un código escrito en Arduino, y que realiza toda la parte de recogida de datos de los sensores, envío mediante MQTT, y recepción de mensajes con las órdenes de riego. El código de Arduino configura varios componentes: un caudalímetro para medir la cantidad de agua regada, un sensor capacitivo para medir la humedad del suelo, un sensor de humedad y temperatura DHT22, y un relé para controlar la bomba de agua. La conexión a la red WiFi permite al dispositivo comunicarse con un broker MQTT, enviando periódicamente los datos recopilados y recibiendo mensajes para iniciar el riego.
- **Docker:** En este proyecto se ha usado un Docker para usar una base de datos MongoDB para guardar las mediciones de los sensores, y un bróker MQTT para la comunicación entre dispositivo y servidor, ya que Docker permite crear entornos portátiles de manera eficiente. Para la creación de la imagen se ha usado un Dockerfile con las instrucciones necesarias para que funcionen ambas cosas.

Primero se ha seleccionado una imagen de Ubuntu que soporte tanto MongoDB como Mosquitto-MQTT. Sobre esta, se han realizado las instalaciones necesarias para hacer funcionar los scripts Python que tenemos, mediante el gestor de paquetes pip. Por último, se han copiado los ficheros desarrollados por nosotros y se configuran para que, cada vez que se arranque el contenedor, estén funcionando.

Por otro lado, se encuentra la parte **Frontend**, en la que solo se tiene la parte del servidor, donde se realizan la mayoría de las interacciones entre el usuario y el sistema. Para el desarrollo del frontend del servidor, se han utilizado principalmente HTML y CSS. HTML (HyperText Markup Language) es el estándar para estructurar el contenido en la web, mientras que CSS (Cascading Style Sheets) se utiliza para controlar el estilo y la presentación del contenido.

7.8. Pruebas

Las pruebas se han convertido en un componente imprescindible para asegurar la calidad de los productos software. En este apartado se detalla la metodología de pruebas, identificando los tipos de pruebas. Para comprobar el funcionamiento del sistema realizaremos tres tipos de pruebas:

- **Pruebas unitarias de componentes:** Estas pruebas se enfocan en evaluar el funcionamiento de un componente específico de la aplicación de manera aislada, sin considerar los otros elementos de la interfaz. Se llevaron a cabo desde el inicio del desarrollo del sistema.
- **Pruebas finales de integración:** Son aquellas que verifican cómo dos entidades se comunican e interactúan entre sí. Estas pruebas se realizaron una vez que se había completado la mayor parte del desarrollo del sistema.
- **Pruebas de interfaz de usuario:** Pruebas en las que se simula de qué forma interactúa el usuario con la aplicación.

7.9. Funcionalidad del sistema

Este apartado realiza un resumen del funcionamiento del sistema, descrito de forma más exhaustiva en el Anexo IV.

El servidor no se encuentra desplegado en ningún entorno de producción, sino que para acceder al servidor hay que encontrarse en la misma red que el ordenador que aloja el servidor. Únicamente habría que escribir en el buscador <http://localhost:5000/>

7.9.1. Página inicial

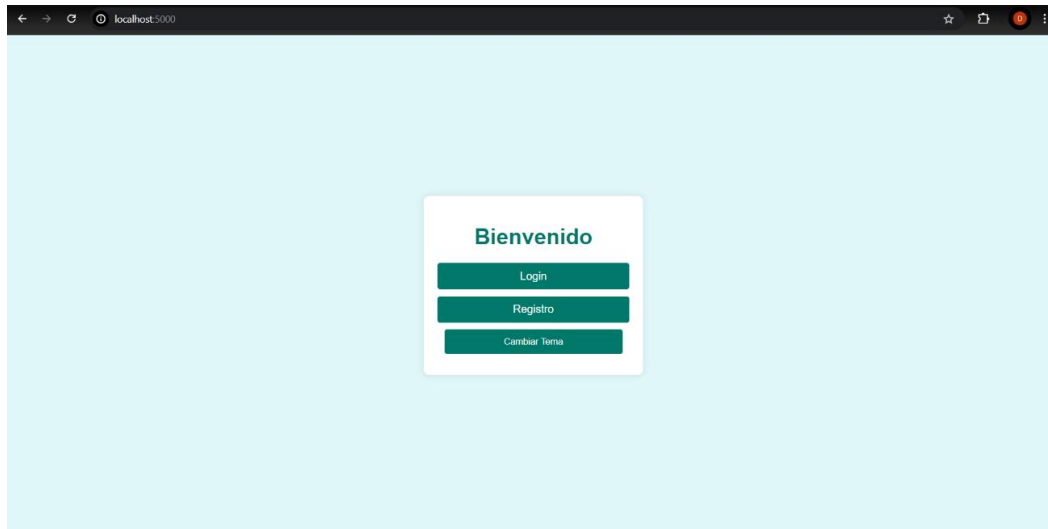
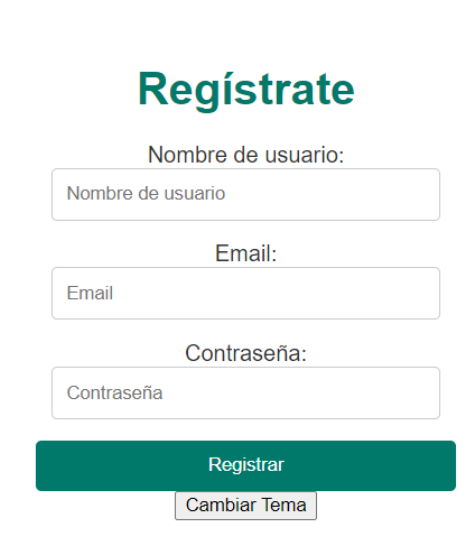


Figura 54: Página principal

En la ventana vemos que tenemos tres botones, dos de los cuales sirven registrarse e iniciar sesión.

7.9.2. Registro

Si pulsamos en el botón de registro nos llevará a una página con un formulario que el usuario tendrá que completar para crear una cuenta. Tener una cuenta creada es necesario para usar el sistema en su totalidad.



Regístrate

Nombre de usuario:

Nombre de usuario

Email:

Email

Contraseña:

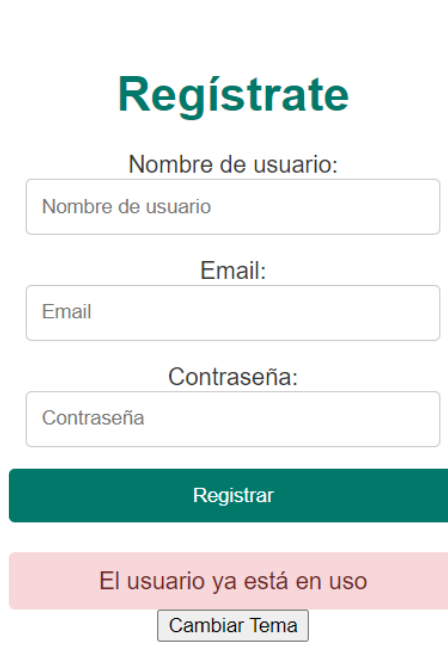
Contraseña

Registrar

Cambiar Tema

Figura 55: Registrar cuenta

En el caso de que se intentase crear una cuenta con unas credenciales ya usadas, saldría el siguiente aviso:



Regístrate

Nombre de usuario:

Nombre de usuario

Email:

Email

Contraseña:

Contraseña

Registrar

El usuario ya está en uso

Cambiar Tema

Figura 56: Mensaje de error de registro

7.9.3. Iniciar sesión

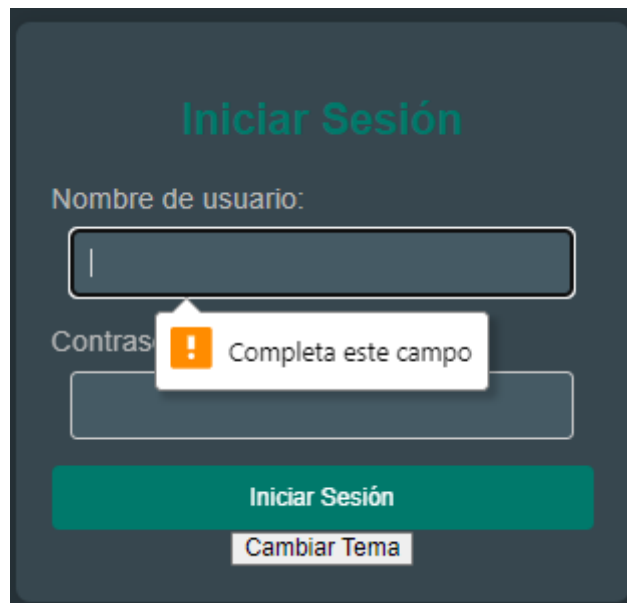
Tras crear una cuenta, ya se puede acceder al sistema desde la pantalla de inicio de sesión. En ella saldrá un formulario en el que tendremos que introducir las credenciales creadas anteriormente. En caso de que las credenciales introducidas no sean correctas saldrá el siguiente mensaje:



The screenshot shows a login form titled "Iniciar Sesión" in green. It has two input fields: "Nombre de usuario:" and "Contraseña:". Below the password field is a green "Iniciar Sesión" button. Below the button is a pink error message box that says "Inicio de sesión incorrecto". At the bottom is a small button labeled "Cambiar Tema".

Figura 57: Mensaje de error en iniciar sesión

En todas las pantallas aparece un botón que permite cambiar el tema, ya que muchos usuarios prefieren trabajar con un fondo oscuro.



The screenshot shows the same login form as Figure 57, but in a dark theme. The title "Iniciar Sesión" is green. The "Nombre de usuario:" field is empty. The "Contraseña:" field has a validation error: a white tooltip with an orange exclamation mark icon and the text "Completa este campo". Below the password field is a green "Iniciar Sesión" button. At the bottom is a button labeled "Cambiar Tema".

Figura 58: Comprobación de campos. Tema oscuro

7.9.4. Página principal

En la página principal del sistema tenemos la predicción meteorológica con un buscador para cambiar la ciudad, una lista de los dispositivos registrados, y el icono de usuario que abre un desplegable con varias opciones.

¡Bienvenido, usuario2!

Pronóstico para Salamanca

Actualizar Pronóstico

Fecha y hora	Temperatura (°C)	Sensación térmica (°C)	Mínima (°C)	Máxima (°C)	Presión (hPa)	Humedad (%)	Velocidad del viento (m/s)	Nubosidad (%)	Probabilidad de lluvia (%)
2024-06-27 12:00:00	24.21	23.78	24.21	27.02	1011	42	1.64	97	20
2024-06-27 15:00:00	28	27.3	28	30.6	1010	34	2.42	80	0
2024-06-27 18:00:00	28.76	27.99	28.76	28.76	1009	35	3.39	82	0
2024-06-27 21:00:00	20.51	20.42	20.51	20.51	1012	69	3.89	95	20
2024-06-28 00:00:00	17.74	17.71	17.74	17.74	1013	82	1.78	75	0
2024-06-28 03:00:00	16.88	16.79	16.88	16.88	1013	83	2.61	92	0
2024-06-28 06:00:00	17.17	17.11	17.17	17.17	1014	83	1.68	80	20

Nombre de la colección

[VideoExplicación](#)

Figura 59: Página principal

¡Bienvenido, usuario2!

Pronóstico para Salamanca

Actualizar Pronóstico

Fecha y hora	Temperatura (°C)	Sensación térmica (°C)	Mínima (°C)	Máxima (°C)	Presión (hPa)	Humedad (%)	Velocidad del viento (m/s)	Nubosidad (%)
2024-06-27 12:00:00	24.21	23.78	24.21	27.02	1011	42	1.64	97
2024-06-27 15:00:00	28	27.3	28	30.6	1010	34	2.42	80
2024-06-27 18:00:00	28.76	27.99	28.76	28.76	1009	35	3.39	82

x

Perfil

Cerrar Sesión

Figura 60: Barra lateral

7.9.5. Visualizar datos

Cuando pulsamos en uno de los dispositivos que tenemos registrados, el sistema nos lleva a una página que permite la visualización de estos de forma sencilla.

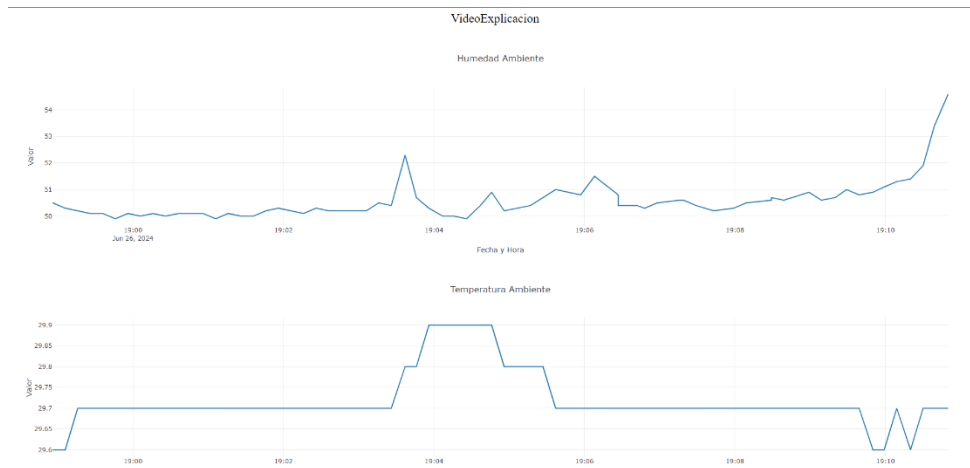


Figura 61: Gráficas de mediciones

Se muestran tres gráficas, y si hace un cuadro con el ratón encima de una de ellas se amplía para ver más en detalle los datos.

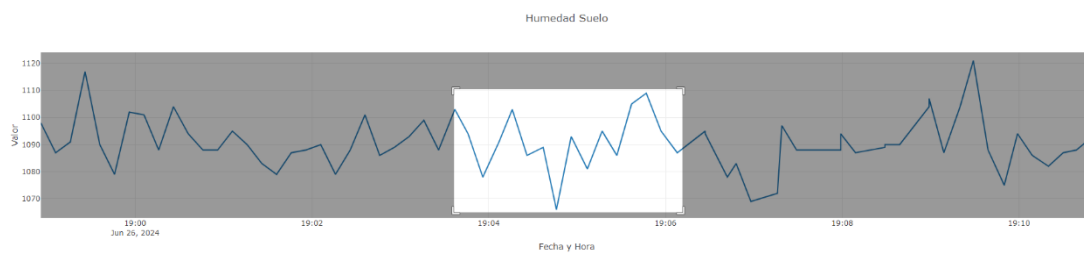


Figura 62: Ampliar zona de la gráfica

7.9.6. Perfil

Cuando un usuario pulsa en el botón “Perfil” del desplegable de la página principal, se muestra una ventana con lo siguiente.

Perfil de Usuario

Datos Actuales del Usuario:

Nombre de Usuario: usuario2
Email: usuario2@gmail.com

Modificar Datos del Perfil:

Nuevo Nombre:

Nuevo Email:

Nueva Contraseña:

Actualizar Perfil

Eliminar Cuenta

Dispositivos Registrados:

VideoExplicacion

Dispositivos Disponibles. Selecciona todos a los que quieras registrarte:

☐ Isaac

☐ Prueba2

☐ PruebaTFG

☐ Video

☐ VideoExplicacion

☐ maceta4

☐ preuba2tfg

☐ prueba

☐ pruebas_predi

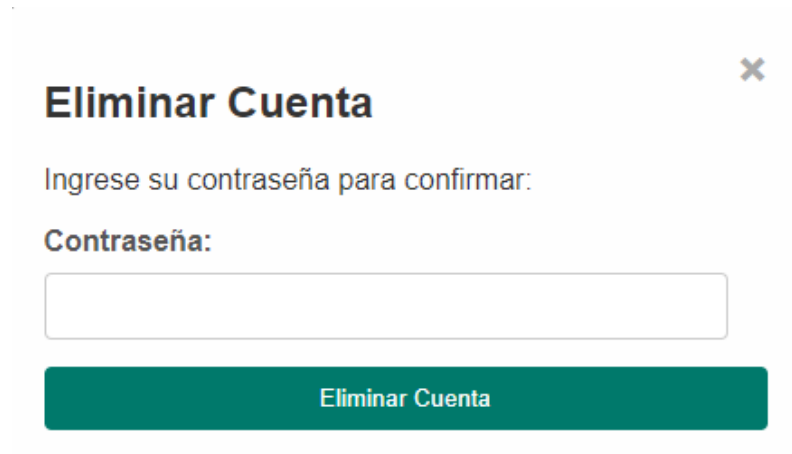
Guardar Cambios

Figura 63: Página perfil de usuario

En esta página podemos ver los datos actuales y los dispositivos que el usuario tiene registrados, y varios formularios para modificar los datos del perfil. Además, aparece una lista con los dispositivos disponibles en el sistema que, cuando se seleccionan, serán mostrados en la pantalla principal.

7.9.7. Eliminar cuenta

También aparece un botón de eliminar cuenta que, si lo pulsamos, nos abrirá una ventana modal pidiendo la contraseña de la cuenta por seguridad.

A modal window titled "Eliminar Cuenta" with a close button (X) in the top right corner. Below the title, it says "Ingrese su contraseña para confirmar:". There is a label "Contraseña:" followed by a text input field. At the bottom, there is a green button labeled "Eliminar Cuenta".

Eliminar Cuenta

Ingrese su contraseña para confirmar:

Contraseña:

Eliminar Cuenta

Figura 64: Ventana modal de eliminar cuenta

7.9.8. Recuperar contraseña

Si el usuario ha olvidado la contraseña, puede obtener una nueva pulsando en el botón de recordar contraseña en la pantalla de inicio. Se abrirá una nueva ventana modal en la que el usuario introducirá su correo, y recibirá una nueva contraseña que podrá modificar en cuanto acceda al servidor.

A modal window titled "Recuperar Contraseña" with a close button (X) in the top right corner. Below the title, it says "Correo electrónico:". There is a text input field. At the bottom, there is a green button labeled "Enviar".

Recuperar Contraseña

Correo electrónico:

Enviar

Figura 65: Ventana modal de recuperar contraseña

8. Instalación y puesta en marcha

Por último, en este apartado se va a comentar cómo se ha realizado la instalación de los sensores, las conexiones que forman el circuito, y la puesta en marcha de los dispositivos.

En este apartado se comentará de forma breve la instalación, puesta en marcha y conexiones. Se encontrará de forma detallada en el Anexo 7.

8.1. Instalación

Para el montaje del dispositivo, el usuario final no va a tener que realizar todas las conexiones eléctricas. Esto se realizará previamente siguiendo el siguiente diseño:

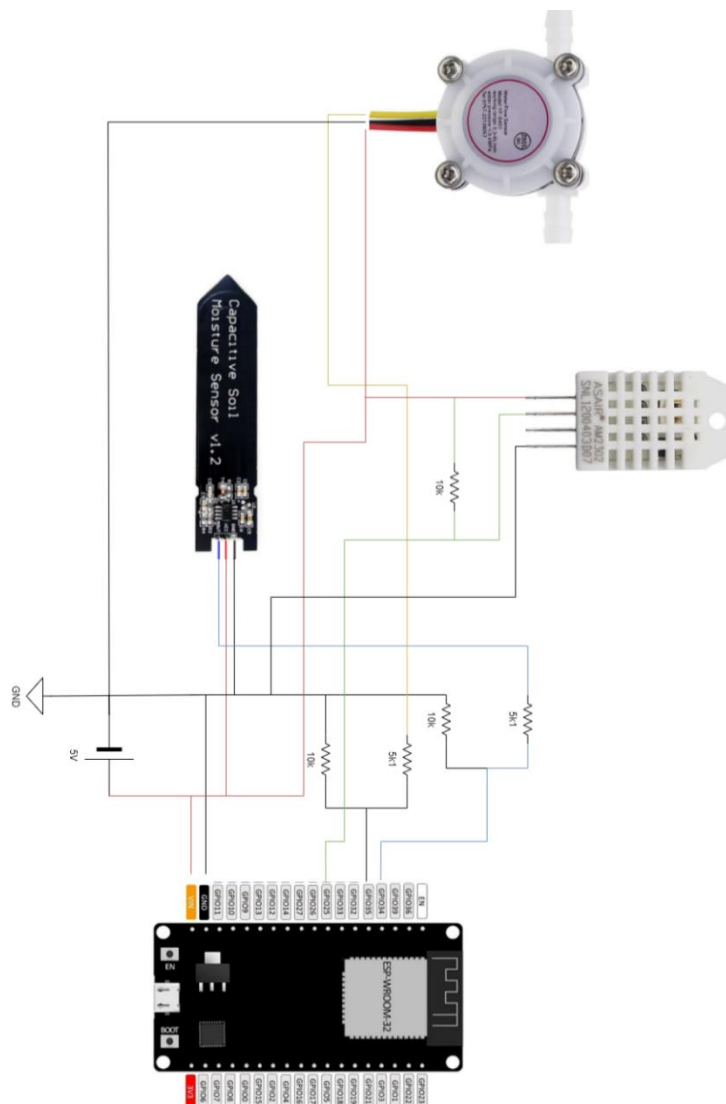


Figura 66: Diagrama de conexión

El usuario final recibirá el dispositivo ya montado, teniendo que conectar únicamente los sensores. Para facilitar esto, se han indicado en los encabezados de pines las conexiones necesarias:

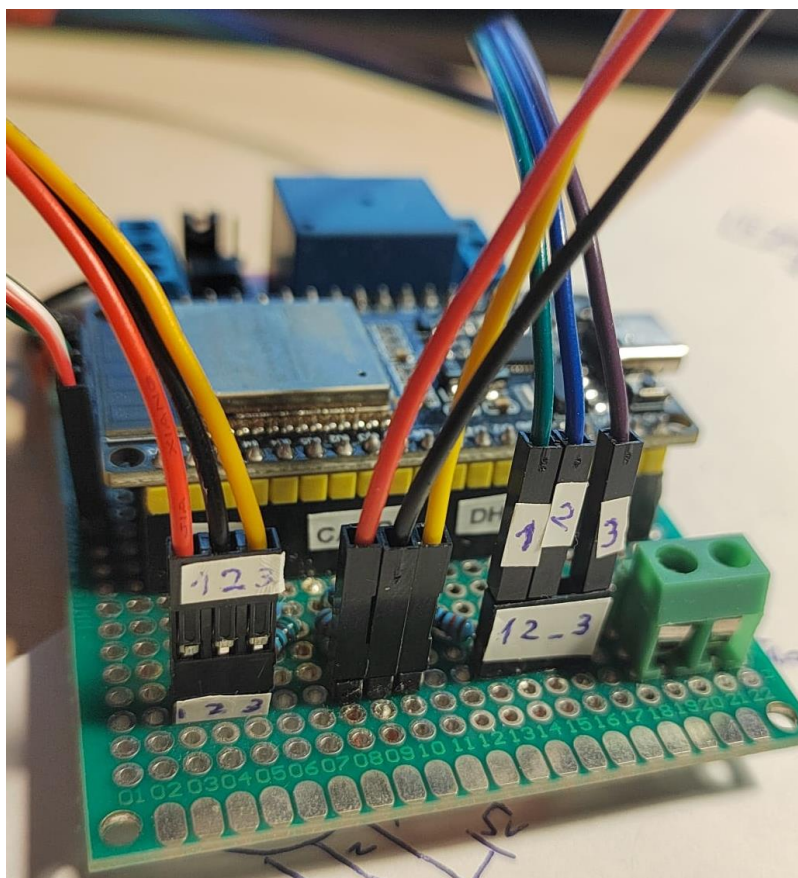


Figura 67: Prototipo. Componentes instalados

8.2. Puesta en marcha

Para comenzar a usar el dispositivo habría que modificar las direcciones del bróker, y las credenciales de la conexión WiFi. Esto se realiza de forma muy sencilla ya que se ha parametrizado el código del dispositivo.

Con el código modificado, habría que cargarlo en la ESP32 y ésta estaría funcional. Cada vez que se pulsa el botón de reinicio de la ESP se puede simular el arranque de un nuevo dispositivo.

Respecto a la parte del servidor, la puesta en marcha es más sencilla. Para los distintos ficheros Python que gestionan el registro de un dispositivo y guardan las mediciones en la base de datos, se creará un ejecutable que se guardará en la carpeta: "C:\Users\dplat\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup".

Esta carpeta indica al sistema operativo qué aplicaciones hay que arrancar al iniciar el dispositivo.

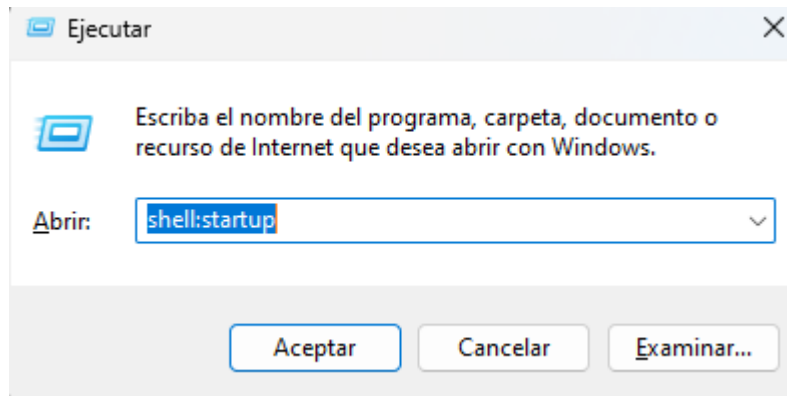


Figura 68: Cuadro de diálogo Ejecutar

Para el contenedor Docker se creará un script que abra Docker y escriba por terminal los comandos necesarios para arrancar el contenedor.

En resumen, se hará una puesta en marcha totalmente automatizada para el servidor para facilitar al usuario el uso del sistema. Para el dispositivo habría que cambiar las direcciones y las credenciales para permitir la conexión, y estaría funcionando también.

9. Conclusiones y líneas de trabajo futuras

9.1. Conclusiones

Con el desarrollo del sistema terminado, se muestran las conclusiones sacadas a lo largo del desarrollo. También se mencionan líneas de trabajo futuras para mejorar las funcionalidades o capacidades.

En primer lugar, comentar que se han cumplido con todos los objetivos y requisitos propuestos:

- Desarrollo de un dispositivo IoT conectado a distintos sensores, como son el sensor capacitivo de humedad del suelo, el sensor DHT22, y un caudalímetro.
- Desarrollo de un modo de comunicación inalámbrico como MQTT, que permite la ingesta de los datos de los sensores en la base de datos.
- Creación de una planificación mediante los datos del entorno y el pronóstico meteorológico de la ciudad en la que se encuentra el dispositivo.
- Desarrollo de una plataforma web que permita a los usuarios la visualización de los datos de forma más clara
- Registro de nuevos dispositivos de manera sencilla.
- Obtención del pronóstico meteorológico mediante una API.
- Automatización de la puesta en marcha del sistema.
- Se ha aprendido a desarrollar una plataforma web mediante el framework Flask.
- Se han podido aplicar los conocimientos adquiridos a lo largo de estos cuatro años de Grado en un proyecto útil e interesante.
- Se ha conseguido solucionar de forma autónoma los problemas surgidos a lo largo del desarrollo del sistema, investigando posibles soluciones y alternativas.

El desarrollo del proyecto ha sido sencillo gracias a las tecnologías usadas, muchas de ellas recomendadas por los tutores. El uso del framework de Flask ha derivado en un desarrollo relativamente sencillo de la plataforma web, ya que es bastante intuitivo. Además, el uso del protocolo MQTT ha hecho que sea muy fácil la comunicación entre dispositivo y servidor, ya que su funcionamiento con tópicos es muy sencillo de entender y permite dividir las funcionalidades implementadas en tópicos distintos. Esto ahorra una gran cantidad de esfuerzo, ya que facilita la organización de las distintas partes del sistema.

Otro punto que comentar son las bases de datos elegidas, MariaDB para los usuarios y dispositivos, y MongoDB para las mediciones de los sensores. La elección de estas bases de datos ha hecho que no sea complicado el desarrollo del sistema ya que una está optimizada para diferentes tipos de datos y operaciones. MariaDB, siendo una base de datos relacional, ofrece una estructura robusta y segura para manejar información estructurada, garantizando la integridad y consistencia de los datos de usuarios y dispositivos. Por otro lado, MongoDB, con su esquema flexible y capacidad para almacenar grandes volúmenes de datos no estructurados, es ideal para manejar las mediciones de los sensores, permitiendo un almacenamiento y recuperación eficiente de datos de manera escalable. Esta combinación ha permitido un desarrollo más fluido y organizado, facilitando la implementación de funcionalidades.

9.2. Líneas de trabajo futuras

Con el desarrollo del sistema terminado, se proponen nuevas características que se podrían añadir a la aplicación:

- Realizar un mantenimiento del sistema, ofreciendo soporte continuado al usuario solucionando cualquier tipo de problema. Esto incluye la corrección de errores,
- Desarrollo de una aplicación móvil para permitir al usuario acceder desde dispositivos móviles. Esto aumentaría la accesibilidad del sistema, permitiendo a usuarios monitorear el sistema en tiempo real y recibir notificaciones desde cualquier lugar.
- Implementación de un estudio botánico para cada planta que se vaya a registrar. Esto sería útil ya que cada planta necesita estar en un intervalo de humedad específico. Se podría crear un catálogo de plantas disponibles para que el usuario pueda elegir a qué planta coloca el dispositivo.
- Implementación de un controlador PID para plantas que necesiten un cuidado muy intensivo. Un controlador PID es un método de dirigir un sistema hacia una posición o nivel determinado. No serviría para todas las plantas ya que el cambio de humedad del suelo tras regar no es inmediato ni uniforme, pero para cierto tipo de plantas que necesiten un cuidado intensivo podría funcionar, como por ejemplo plantas de interior sensibles.
- Desarrollo de una red neuronal que prediga con mayor antelación cuándo va a ser necesario regar, pudiendo así crear planificaciones más complejas y precisas.

10. Referencias

- [1] «INEbase / Agricultura y medio ambiente / Agua / Estadísticas sobre el uso del agua / Últimos datos», INE. Accedido: 4 de julio de 2024. [En línea]. Disponible en: https://ine.es/dyngs/INEbase/es/operacion.htm?c=Estadistica_C&cid=1254736176839&menu=ultiDatos&idp=1254735976602
- [2] uniqoders, «Consumo del agua en la agricultura», Aquacorp. Accedido: 4 de julio de 2024. [En línea]. Disponible en: <https://aquacorp.es/consumo-del-agua-en-la-agricultura/>
- [3] «What is the Internet of Things (IoT)? | IBM». Accedido: 4 de julio de 2024. [En línea]. Disponible en: <https://www.ibm.com/topics/internet-of-things>
- [4] ngarcia, «Las nuevas tarifas del agua entrarán en vigor parcialmente el 10 de mayo», Emasesa. Accedido: 4 de julio de 2024. [En línea]. Disponible en: <https://www.emasesa.com/las-nuevas-tarifas-del-agua-entraran-en-vigor-parcialmente-el-10-de-mayo/>
- [5] E. Goins, «House and Indoor Plants Temperature Guide», House Plants Expert. Accedido: 4 de julio de 2024. [En línea]. Disponible en: <https://houseplantsexpert.com/indoor-plants-temperature-guide.html>
- [6] «Humedad Del Suelo: Métodos E Instrumentos De Medición». Accedido: 4 de julio de 2024. [En línea]. Disponible en: <https://eos.com/es/blog/humedad-del-suelo/>
- [7] J. A. Pardos, «Respuestas de las plantas al anegamiento del suelo», 2004.
- [8] «Understanding optimum Temperature and Humidity for Plants», London Grow. Accedido: 4 de julio de 2024. [En línea]. Disponible en: <https://www.londongrow.com/blogs/grow-tips/understanding-optimum-temperature-and-humidity-for-plants>
- [9] Lorena, «Drones y medio ambiente», IDC. Accedido: 4 de julio de 2024. [En línea]. Disponible en: <https://idc.apddrones.com/drones/drones-y-medio-ambiente/>
- [10] A. Dhakal y S. Shakya, «Image-Based Plant Disease Detection with Deep Learning», *Int. J. Comput. Trends Technol.*, vol. 61, pp. 2231-2803, jul. 2018, doi: 10.14445/22312803/IJCTT-V61P105.
- [11] N. O. Godoy y D. G. A. Diaz, «Control de Humedad de suelo implementando control por PID», ene. 2018, Accedido: 4 de julio de 2024. [En línea]. Disponible en: https://www.academia.edu/36796556/Control_de_Humedad_de_suelo_implementando_control_por_PID
- [12] «Rachio 3 Smart Sprinkler Controller», Rachio. Accedido: 4 de julio de 2024. [En línea]. Disponible en: <https://rachio.com/products/rachio-3/>
- [13] «B-hyve Smart Hose Watering Timer», OrbitOnline. Accedido: 4 de julio de 2024. [En línea]. Disponible en: <https://www.orbitonline.com/products/b-hyve-smart-hose-watering-timer>
- [14] «Sensores de Precisión Controlar tu Riego | Datalogger Novagric SYNION». Accedido: 4 de julio de 2024. [En línea]. Disponible en: <https://novagric.com/es/tecnologia/sensor-riego-synion>
- [15] «Plantae - sensores agrícolas de humedad y sondas para optimizar el riego», Plantae®. Accedido: 4 de julio de 2024. [En línea]. Disponible en: <https://plantae.garden/>
- [16] J. Nielsen, *Usability engineering*. Morgan Kaufmann, 1994. Accedido: 4 de julio de 2024. [En línea]. Disponible en: https://books.google.com/books?hl=en&lr=&id=95As2OF67f0C&oi=fnd&pg=PR9&dq=info:H6_RbkR2_6AJ:scholar.google.com&ots=3dAxxp9vUw&sig=6G9a4Dv0jEy4fp4qqZUTj-VJiMg
- [17] «Front End frente a back-end: diferencia entre el desarrollo de aplicaciones - AWS», Amazon Web Services, Inc. Accedido: 4 de julio de 2024. [En línea]. Disponible en: <https://aws.amazon.com/es/compare/the-difference-between-frontend-and-backend/>

- [18] «Arquitectura Cliente-Servidor». Accedido: 4 de julio de 2024. [En línea]. Disponible en: <https://reactiveprogramming.io/blog/es/estilos-arquitectonicos/cliente-servidor>
- [19] M. A. V. Acosta, «Pasarela de comunicación entre MQTT y DDS».
- [20] «MongoDB: The Developer Data Platform», MongoDB. Accedido: 4 de julio de 2024. [En línea]. Disponible en: <https://www.mongodb.com>
- [21] «Docker: Accelerated Container Application Development». Accedido: 4 de julio de 2024. [En línea]. Disponible en: <https://www.docker.com/>
- [22] «¿Qué es la impresión 3D? | Software de impresión 3D | Autodesk». Accedido: 4 de julio de 2024. [En línea]. Disponible en: <https://www.autodesk.com/es/solutions/3d-printing>
- [23] «Tormes+ – Tormes+ | Centro de formación y emprendimiento de Salamanca». Accedido: 4 de julio de 2024. [En línea]. Disponible en: <https://tormesplus.com/>
- [24] «MongoDB Compass», MongoDB. Accedido: 4 de julio de 2024. [En línea]. Disponible en: <https://www.mongodb.com/products/tools/compass>
- [25] «Visual Studio Code - Code Editing. Redefined». Accedido: 4 de julio de 2024. [En línea]. Disponible en: <https://code.visualstudio.com/>
- [26] «Weather API - OpenWeatherMap». Accedido: 3 de julio de 2024. [En línea]. Disponible en: <https://openweathermap.org/api>
- [27] «Herramientas CASE. Ingeniería del software. Informática Aplicada a la gestión Pública. Universidad de Murcia». Accedido: 4 de julio de 2024. [En línea]. Disponible en: https://www.um.es/docencia/barzana/IAGP/Enlaces/CASE_principales.html
- [28] «Project». Accedido: 4 de julio de 2024. [En línea]. Disponible en: <https://project.microsoft.com/usales.onmicrosoft.com/es-ES>
- [29] «Build software better, together», GitHub. Accedido: 4 de julio de 2024. [En línea]. Disponible en: <https://github.com>