Anexo IV: Diseño del sistema software

Trabajo de Fin de Grado de Ingeniería Informática

Julio de 2024

Autor:

Diego Plata Klingler

Tutores:

Sergio Alonso Rollán

Javier Prieto Tejedor

Tabla de contenidos

1. Introducción 1
2. Modelo de diseño 2
   2.1 Patrones de diseño 2
3. Subsistemas de diseño 3
4. Clases de diseño 4
   4.1 Servidor 4
      4.1.1 Modelo 5
      4.1.2 Vista 6
      4.1.3 Controlador 6
      4.1.4 Docker 7
   4.2 Dispositivo 8
5. Vista arquitectónica de diseño 9
6. Diagrama de casos de uso de diseño 10
   6.1 Diagramas de secuencia del paquete Gestión de usuarios 10
   6.2 Diagramas de secuencia del paquete Gestión de dispositivos 16
   6.3 Diagramas de secuencia del paquete Gestión de la planificación 18
8. Diseño de la base de datos 21
9. Modelo de despliegue 23
11. Diseño de la interfaz 25
12. Referencias 30

Índice de figuras

Figura 1: Diagrama MVC 3
Figura 2: Subsistemas de diseño 4
Figura 3: Subsistema Modelo 5
Figura 4: Subsistema Vista 6
Figura 5: Subsistema Controlador 7
Figura 6: Subsistema controlador del dispositivo 8
Figura 7: Vista arquitectónica MVC 9
Figura 8: Diagrama de secuencia CU-0001 Registro 10
Figura 9: Diagrama de secuencia CU-0002 Iniciar sesión 11
Figura 10: Diagrama de secuencia CU-0003 Ver perfil 11
Figura 11: Diagrama de secuencia CU-0004 Modificar dispositivos registrados 12
Figura 12: Diagrama de secuencia CU-0005 Seleccionar dispositivo para visualizar .. 12
Figura 13: Diagrama de secuencia CU-0006 Cerrar sesión 13
Figura 14: Diagrama de secuencia CU-0007 Modificar perfil 13
Figura 15:. Diagrama de secuencia CU-0008 Eliminar cuenta 14
Figura 16: Diagrama de secuencia CU-0009 Recuperar contraseña 15
Figura 17: Diagrama de secuencia CU-0010 Completar datos del dispositivo 16
Figura 18: Diagrama de secuencia CU-0011 Guardar mediciones de sensores 16
Figura 19: Diagrama de secuencia CU-0012 Mandar mensaje de registro 17
Figura 20: Diagrama de secuencia CU-0013 Calcular planificación de riego 18
Figura 21: Diagrama de secuencia CU-0014 Obtener últimas medidas 18
Figura 22: Diagrama de secuencia CU-0015 Obtener datos del dispositivo 19
Figura 23: Diagrama de secuencia CU-0016 Obtener pronóstico meteorológico 19
Figura 24: Diagrama de secuencia CU-0017 Mandar orden de riego 20
Figura 25: Colección Usuarios. BD SQL 22
Figura 26: Colección Dispositivos. BD SQL 22
Figura 27: Mediciones. BD NoSQL 23
Figura 28: Modelo de despliegue 24
Figura 29: Código de colores seleccionados 25
Figura 30: Prototipado digital. Ventana Registro. Tema claro 26
Figura 31: Prototipado digital. Ventana Iniciar Sesión. Tema oscuro 26
Figura 32: Prototipado digita. Ventana Iniciar Sesión. Tema claro 27
Figura 33: Prototipado digita. Ventana Principal 27
Figura 34: Prototipado digita. Barra lateral 28

Figura 35: Prototipado digital. Ventana modificar perfil 28
Figura 36: Prototipado digita. Ver datos de dispositivo..... 29

1. Introducción

Este anexo documenta la etapa de diseño del sistema, enfocándose en una aproximación detallada de cómo será la implementación final de la aplicación. Se cubrirán aspectos relacionados con el dominio de la solución, describiendo entidades, atributos, papeles y relaciones, basándonos en el modelo del dominio. Esta fase se centra en ofrecer una visión clara de la arquitectura y la estructura del sistema una vez implementado.

Para el desarrollo de este anexo, se consultaron diversos conceptos de Ingeniería del Software utilizando apuntes de UML de Ingeniería de Software [1]. La herramienta Visual Paradigm [2]se empleó para la creación de los diagramas.

El anexo está estructurado de la siguiente manera:

Modelo de diseño: Explicación del modelo de diseño que se seguirá en la implementación.

> **-Patrones arquitectónicos**: Detalle de los patrones arquitectónicos utilizados en el diseño del sistema.
>
> **-Subsistemas de diseño**: Organización de los diferentes sistemas de paquetes y las relaciones entre ellos.
>
> **-Clases de diseño**: Descripción detallada de las clases de diseño, incluyendo métodos y atributos, y sus relaciones.
>
> **-Vista arquitectónica**: Detalle de las clases que formarán las distintas capas del patrón MVP (Modelo-Vista-Presentador) de la arquitectura diseñada.

Realización de casos de uso: Diagramas de secuencia de diseño de los casos de uso del sistema, mostrando el intercambio de mensajes entre los distintos componentes.

Diseño de la base de datos: Visión de las clases en un diagrama de tipo entidad-relación que representará la información de la base de datos.

Modelo de despliegue: Descripción del despliegue de la funcionalidad de nodos y artefactos que componen el sistema.

Diseño de la interfaz: Descripción del proceso de diseño de los elementos gráficos de la aplicación, como la paleta de colores, el logo y el prototipado digital.

A lo largo de este documento se mostrarán tanto los diagramas de casos de uso-diseño como los diagramas de secuencia que se realizaron en el Anexo III – Análisis del sistema software, adaptándolos a los requerimientos del diseño.

2. Modelo de diseño

El modelo de diseño es una representación detallada de cómo se va a implementar un sistema a nivel de código. Este modelo traduce los requisitos descritos en anteriores anexos en una estructura que el desarrollador puede seguir para construir el sistema.

2.1 Patrones de diseño

Los patrones de diseño son soluciones habituales a problemas que ocurren con frecuencia en el diseño de software. Son como planos prefabricados que se pueden personalizar para resolver un problema de diseño recurrente en el código.

El patrón de diseño empleado en este proyecto ha sido el MVC (Modelo-Vista-Controlador), que sirve para estructurar las capas que componen la aplicación, cada una con sus responsabilidades. Separa los datos de las interfaces de datos, y el controlador es el encargado de relacionar estos dos. [3]

El **modelo** es una representación de la información del mundo real que el sistema debe gestionar. El diseño del modelo se centra solamente en la estructura de los datos, sin preocuparse por su presentación al usuario final. Esta separación permite que el modelo funcione independientemente y no dependa de otras partes de la aplicación, como la interfaz de usuario.

La **vista** se encarga de la presentación de la interfaz de usuario, utilizando ficheros HTML para mostrar los datos. Su principal función es traducir los datos del modelo de forma atractiva y comprensible para el usuario. Una vista obtiene del modelo solamente la información que necesita desplegar, y se actualiza cada vez que el modelo cambia.

El **controlador** actúa como intermediario entre el usuario y el sistema. Procesa solicitudes y coordina las interacciones entre el modelo y la vista. Tiene en cuenta las entradas del usuario, ejecuta la lógica de programación, actualiza el modelo y, con éste, la vista. [4]

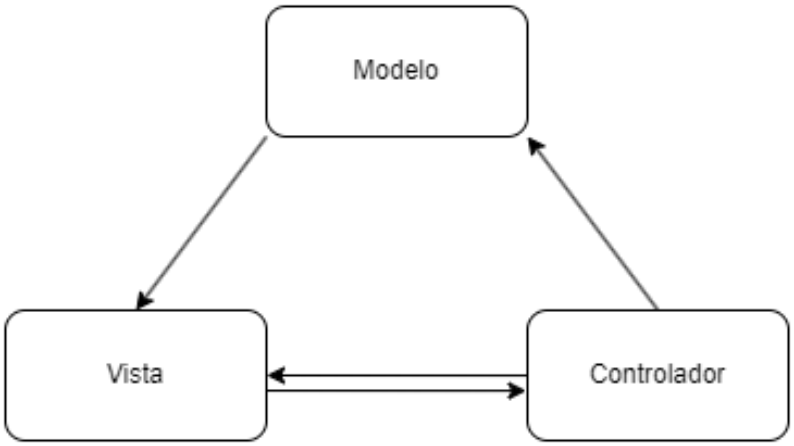

Figura 1: Diagrama MVC

3. Subsistemas de diseño

Apoyándonos en el patrón MCV del apartado anterior, se divide el sistema en diferentes subpaquetes más específicos. Con esto lo que buscamos es tener elementos más manejables, y tener los atributos y relaciones entre ellos mejor definidos. Tendremos los siguientes paquetes:

-**Paquete Servidor**: Contiene la lógica de negocio y la infraestructura de gestión de datos del servidor Flask. Esto incluye la definición de rutas y controladores para manejar los datos del dispositivo IoT, y almacenamiento de datos de la base de datos de usuarios y dispositivos, y la lógica para procesar y presentar estos datos a través de los ficheros HTML. 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 [5]. Este paquete también gestiona la transmisión de datos al servidor.

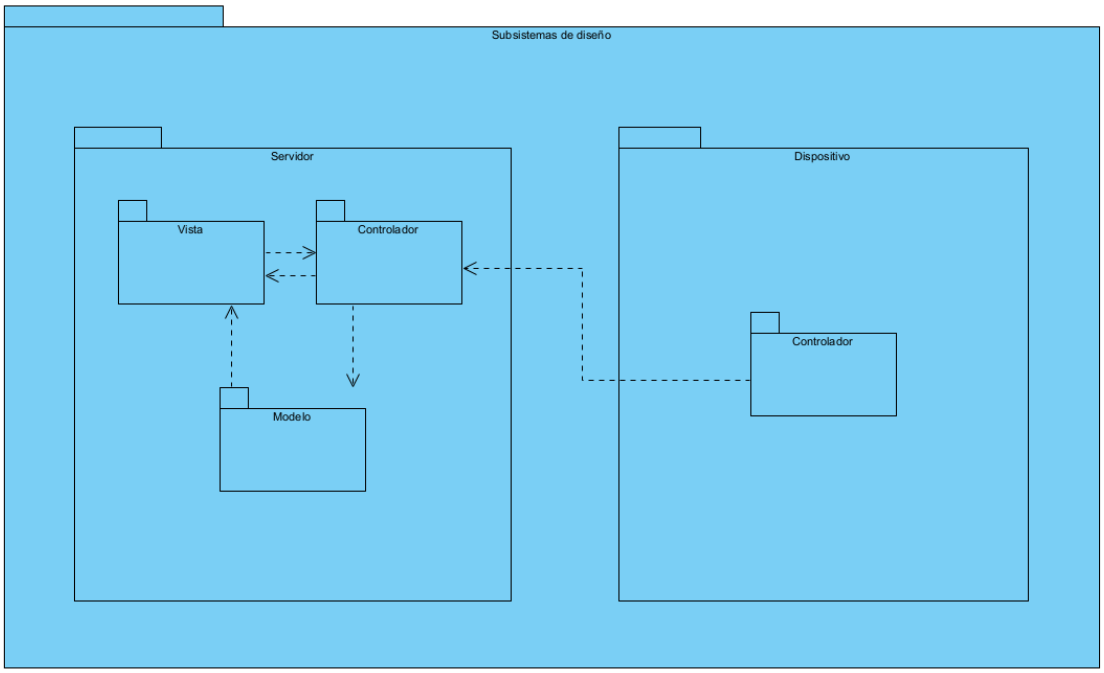

Figura 2: Subsistemas de diseño

4. Clases de diseño

En este apartado se explicarán los subsistemas de diseño, divididos por paquetes.

4.1 Servidor

En este apartado se detallan los subsistemas del paquete Servidor. Como buscamos un sistema con bajo acoplamiento y alta cohesión, dividiremos cada paquete en subsistemas más pequeños.

En el caso del servidor tenemos tres subsistemas, que son el Modelo, la Vista y el Controlador. En las siguientes figuras veremos el diagrama de cada uno:

4.1.1 Modelo

En la figura siguiente se puede ver el diagrama para el modelo:

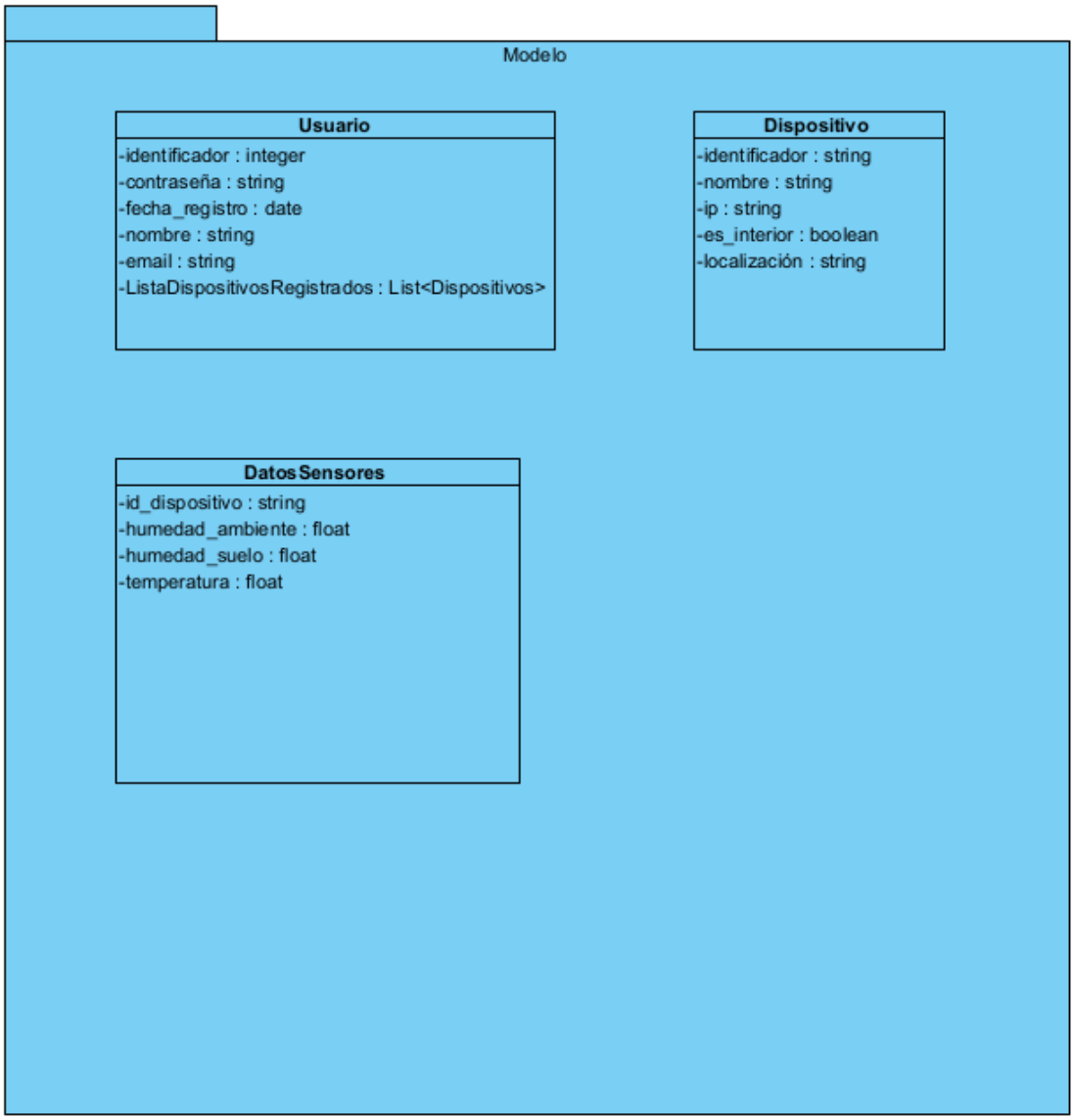

Figura 3: Subsistema Modelo

4.1.2 Vista

En este subsistema tenemos todas las funcionalidades relacionadas con la representación de elementos en la vista del usuario. Esto incluye todas las pantallas a las que el usuario puede acceder durante el uso del sistema.

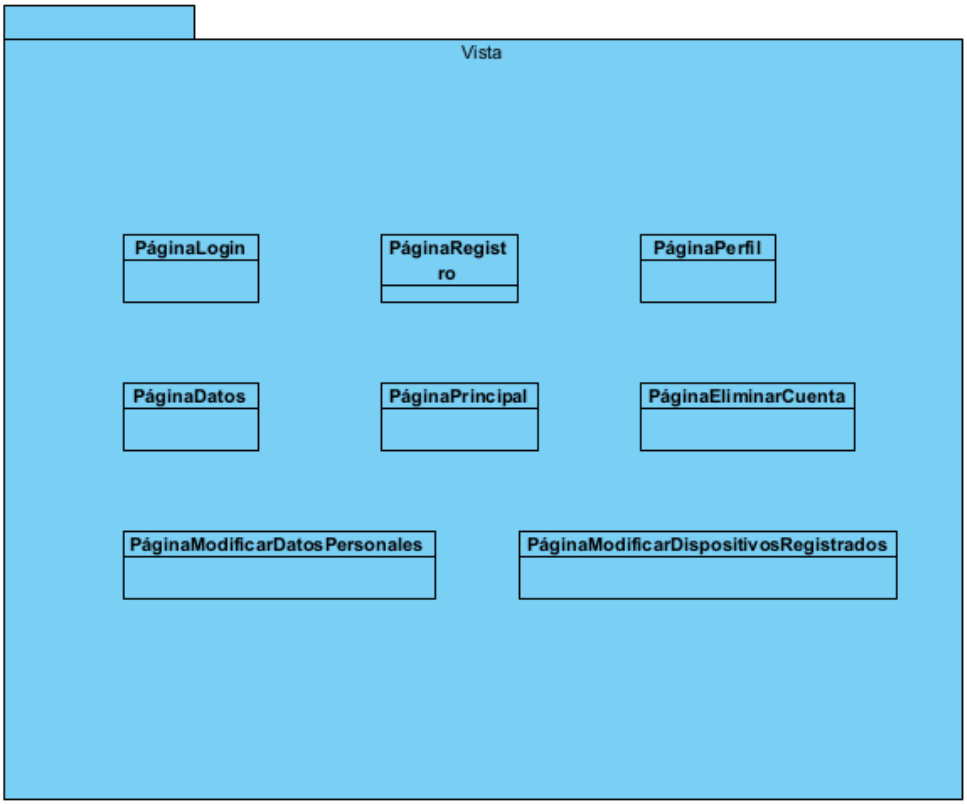

Figura 4: Subsistema Vista

4.1.3 Controlador

En este subsistema tenemos todas las funcionalidades relacionadas con la gestión y procesamiento de las solicitudes del usuario. Esto incluye la lógica para manejar la interacción entre la vista y el modelo, y coordinar las operaciones de lectura y escritura en la base de datos. El controlador actúa como intermediario, garantizando que los datos solicitados sean recuperados y presentados correctamente. Además, realiza todos los cálculos necesarios para el correcto funcionamiento del sistema, como por ejemplo los cálculos para la predicción de riego, o las peticiones a la API del pronóstico del tiempo. [5]

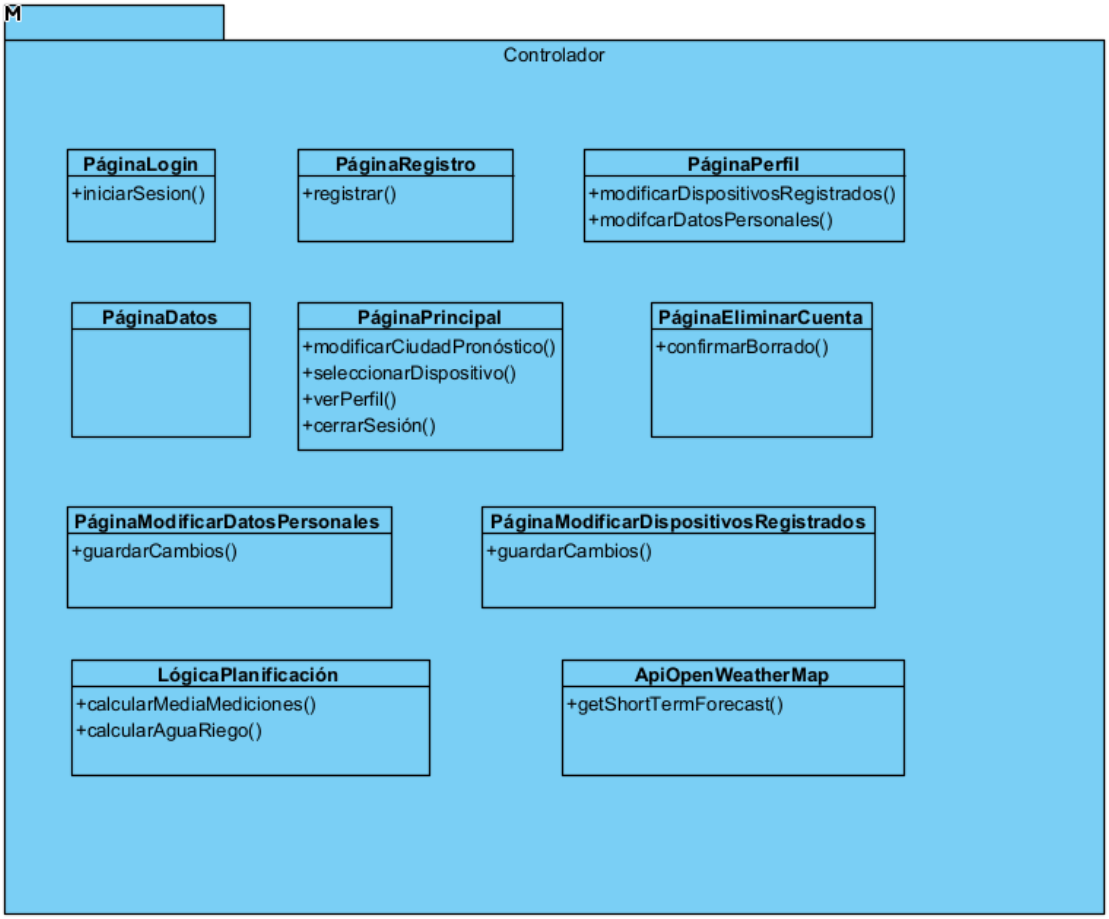

Figura 5: Subsistema Controlador

4.1.4 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. Para lograr esto se “dockerizará” [6] la base de datos de los sensores, y 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, rediciendo 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 librerías 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.

4.2 Dispositivo

En cuanto al dispositivo, únicamente tenemos un controlador, ya que no almacena ni muestra datos directamente.

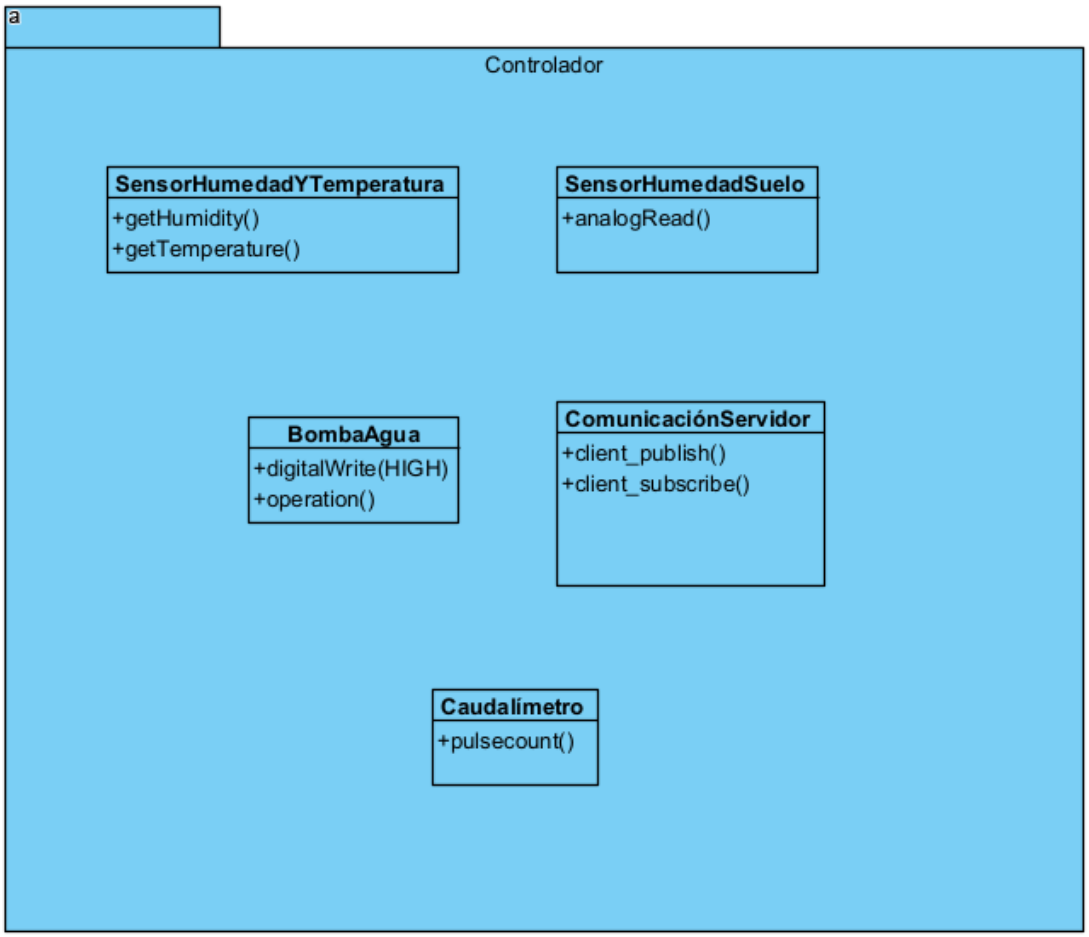

Figura 6: Subsistema controlador del dispositivo

5. Vista arquitectónica de diseño

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

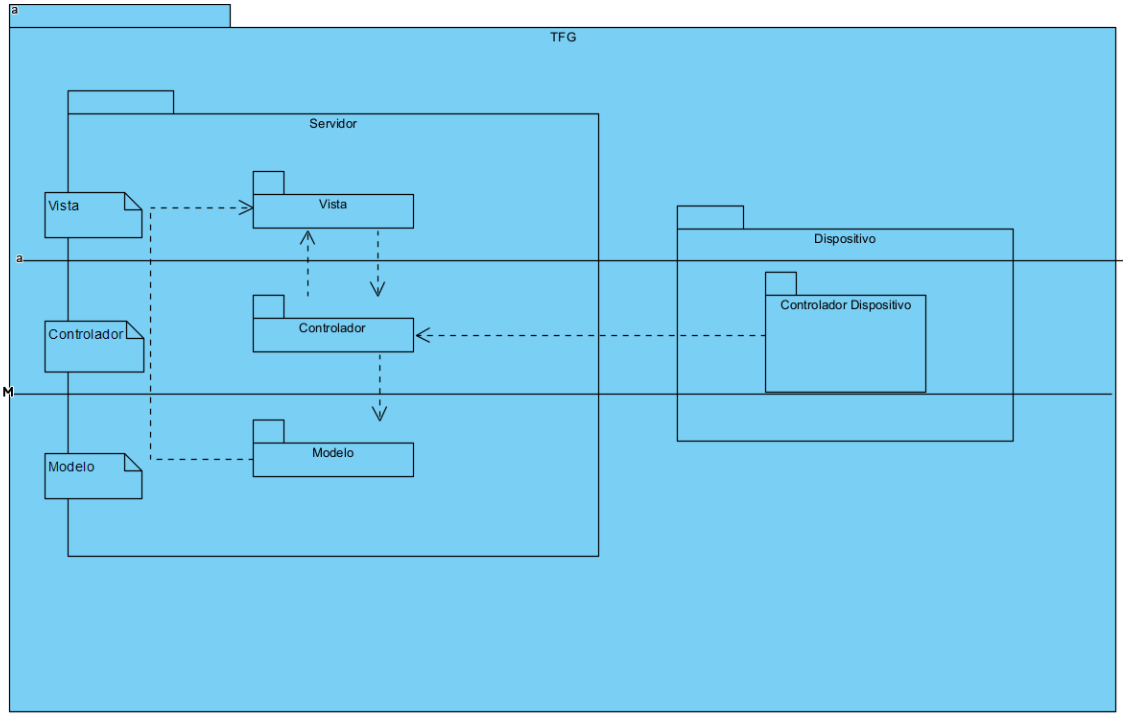

Figura 7: Vista arquitectónica MVC

6. Diagrama de casos de uso de diseño

6.1 Diagramas de secuencia del paquete Gestión de usuarios

CU-0001 Registro:

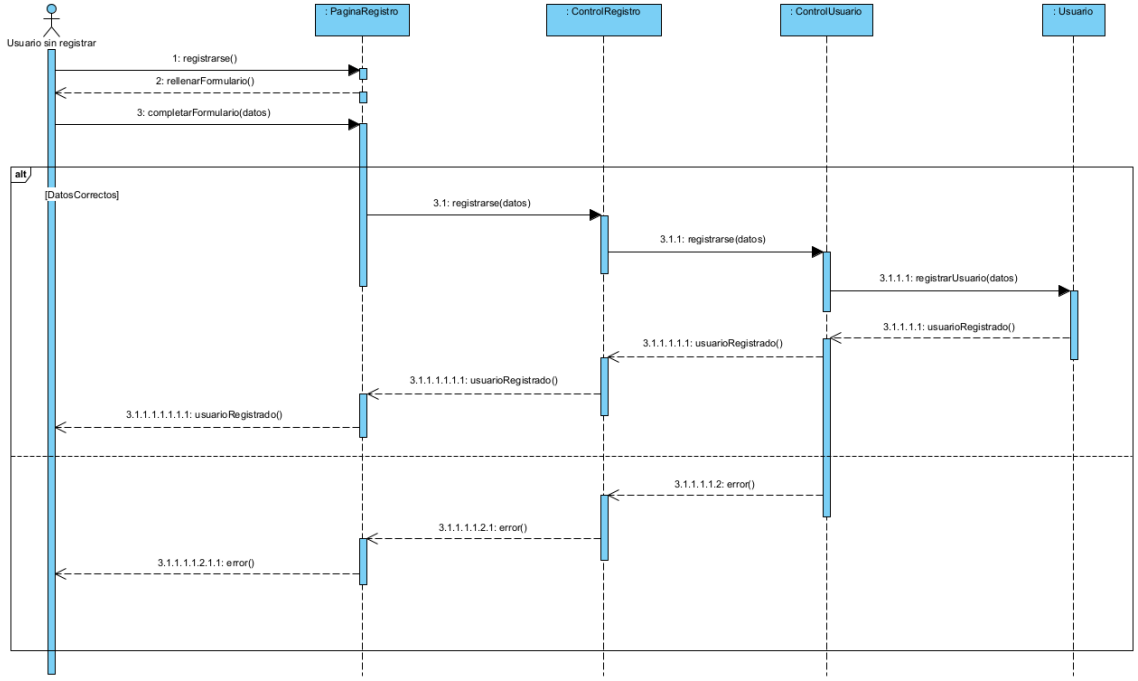

Figura 8: Diagrama de secuencia CU-0001 Registro

CU-0002 Iniciar sesión:

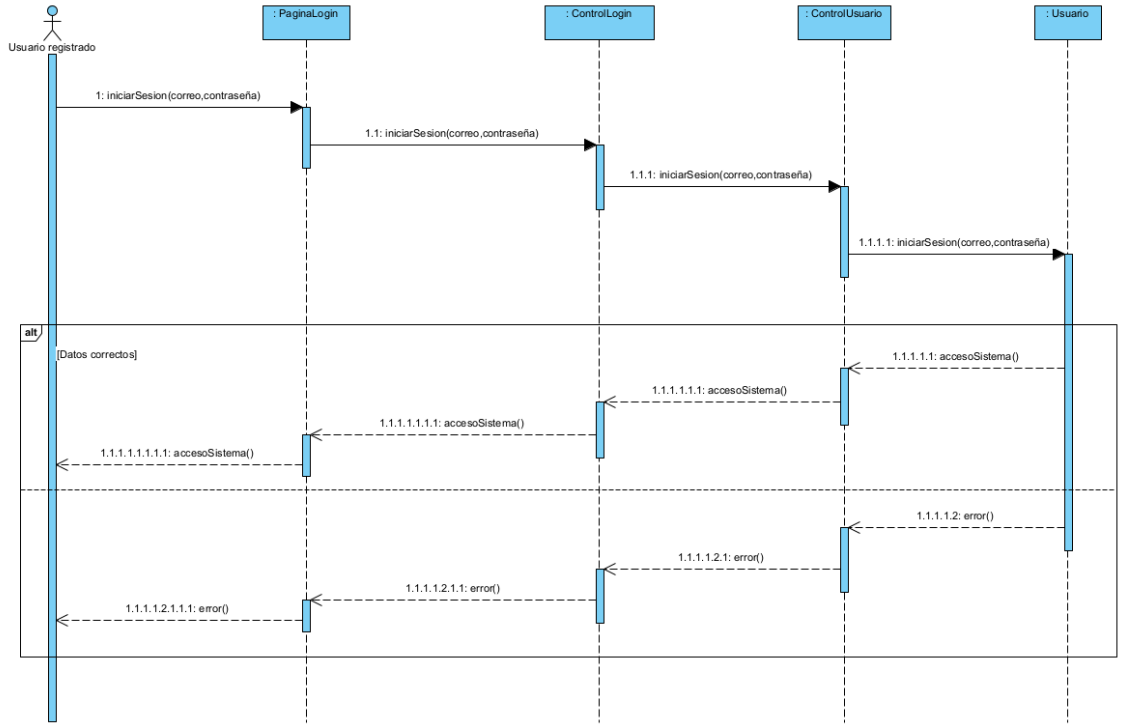

Figura 9: Diagrama de secuencia CU-0002 Iniciar sesión

CU-0003 Ver perfil:

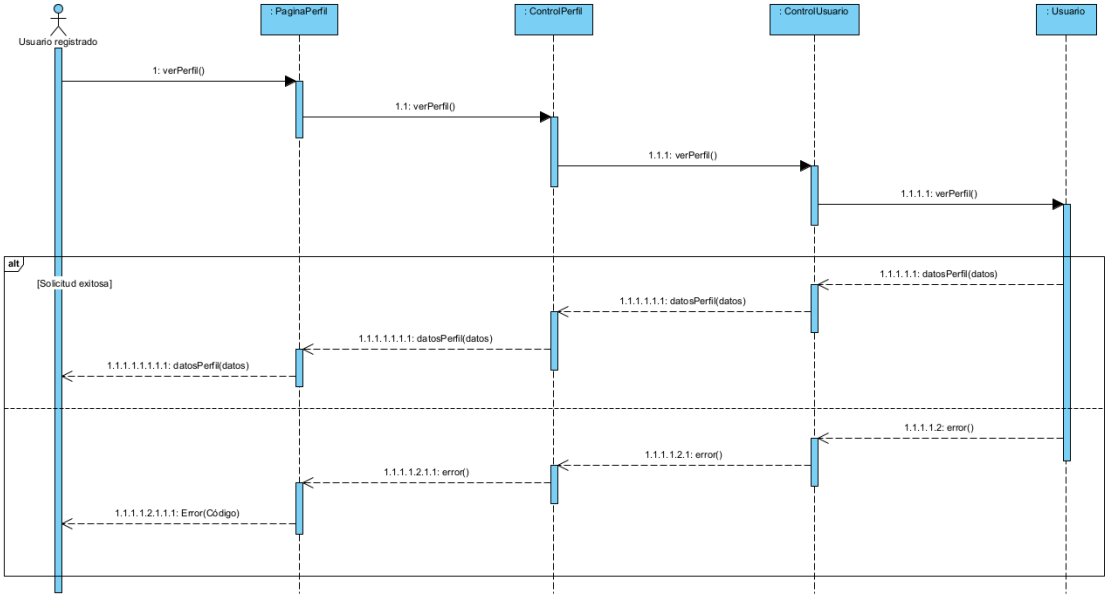

Figura 10: Diagrama de secuencia CU-0003 Ver perfil

CU-0004 Modificar dispositivos registrados:

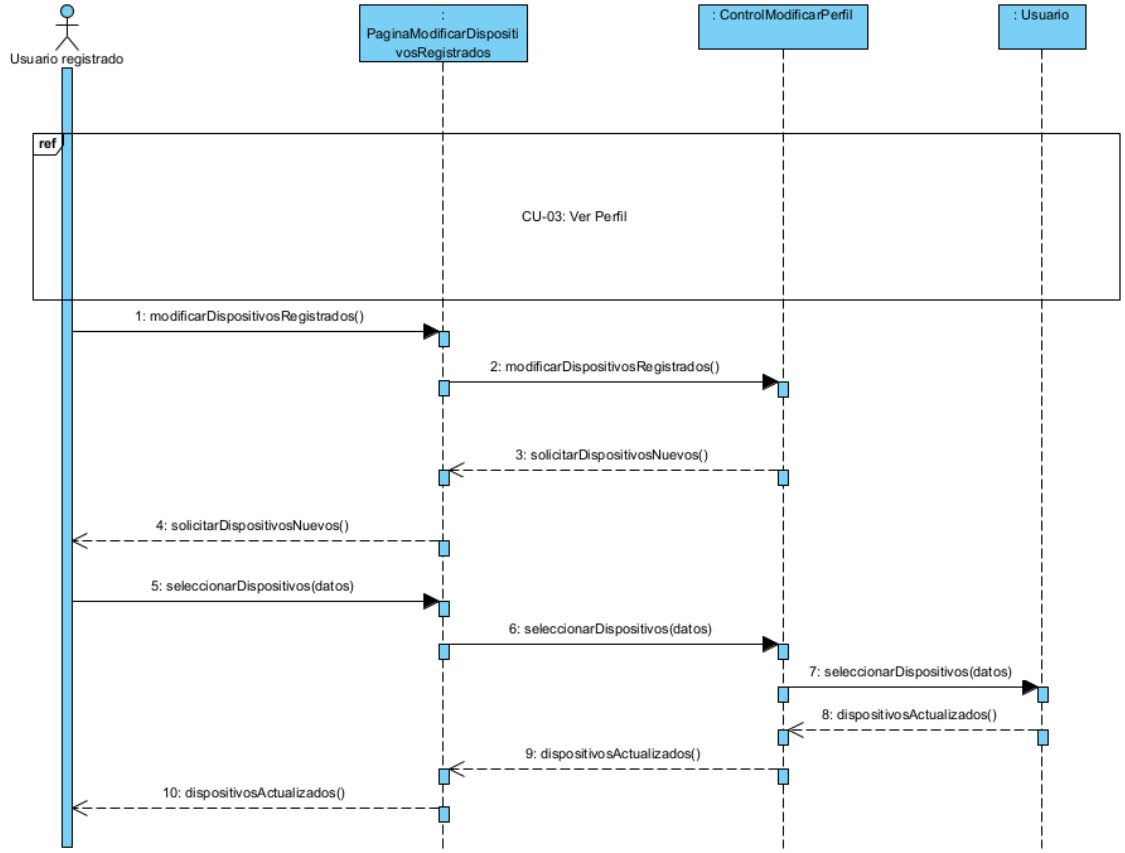

Figura 11: Diagrama de secuencia CU-0004 Modificar dispositivos registrados

CU-0005 Seleccionar dispositivos para visualizar:

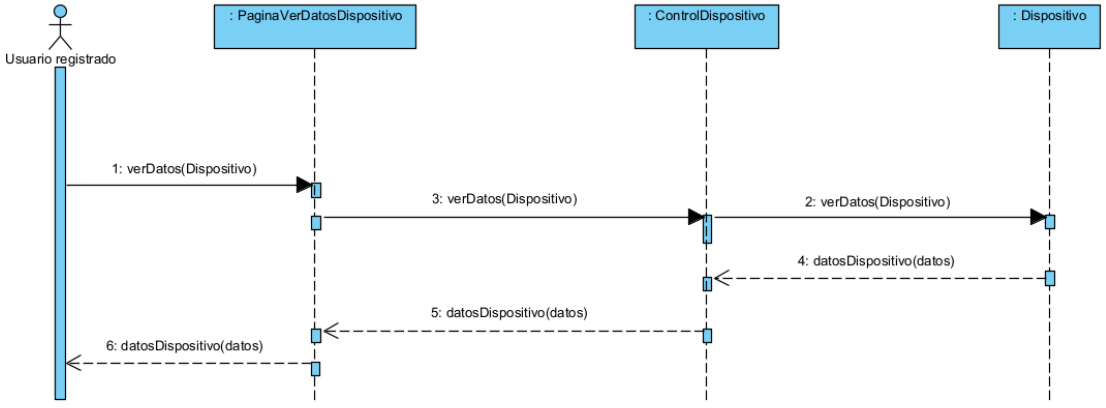

Figura 12: Diagrama de secuencia CU-0005 Seleccionar dispositivo para visualizar

CU-0006 Cerrar sesión:

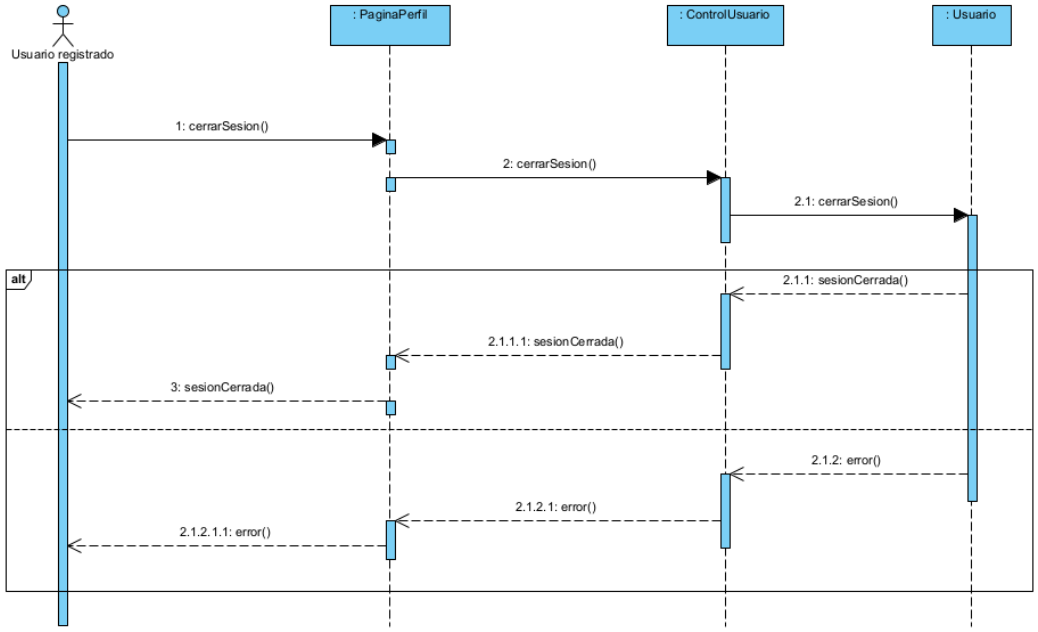

Figura 13: Diagrama de secuencia CU-0006 Cerrar sesión

CU-0007 Modificar perfil:

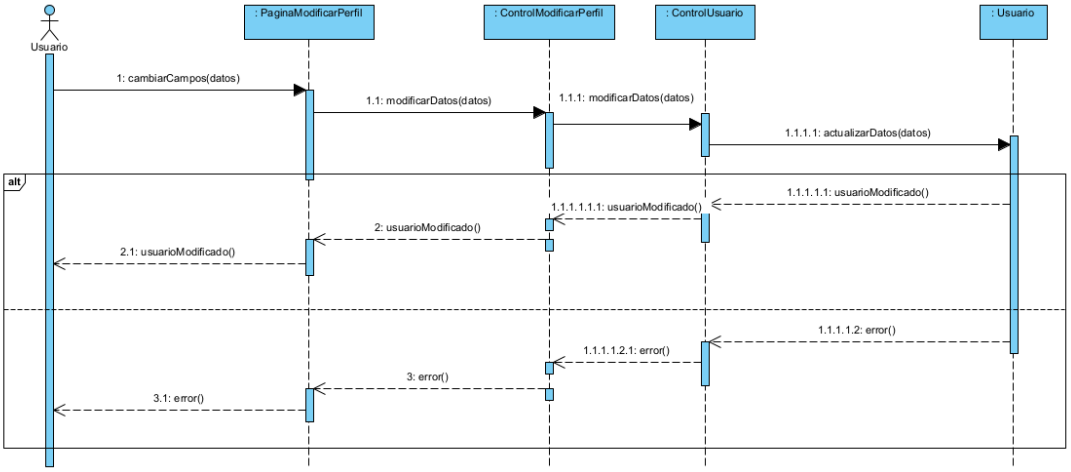

Figura 14: Diagrama de secuencia CU-0007 Modificar perfil

CU-0008 Eliminar cuenta:

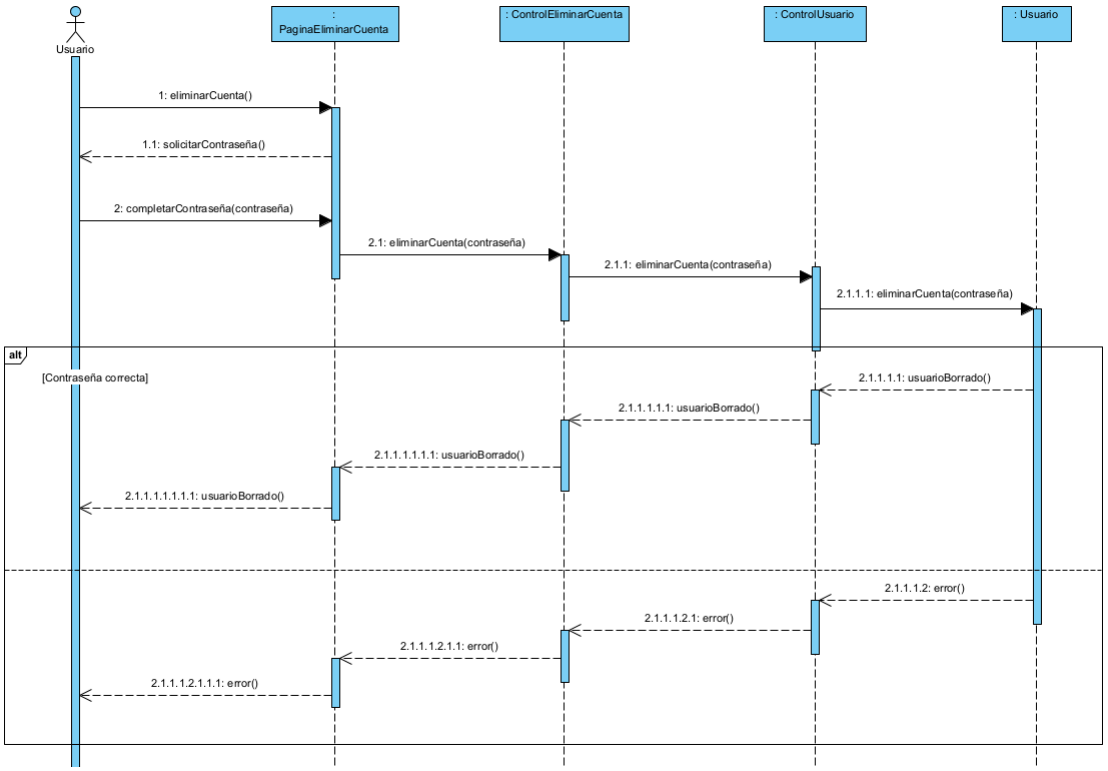

Figura 15:. Diagrama de secuencia CU-0008 Eliminar cuenta

CU-0009 Recuperar contraseña

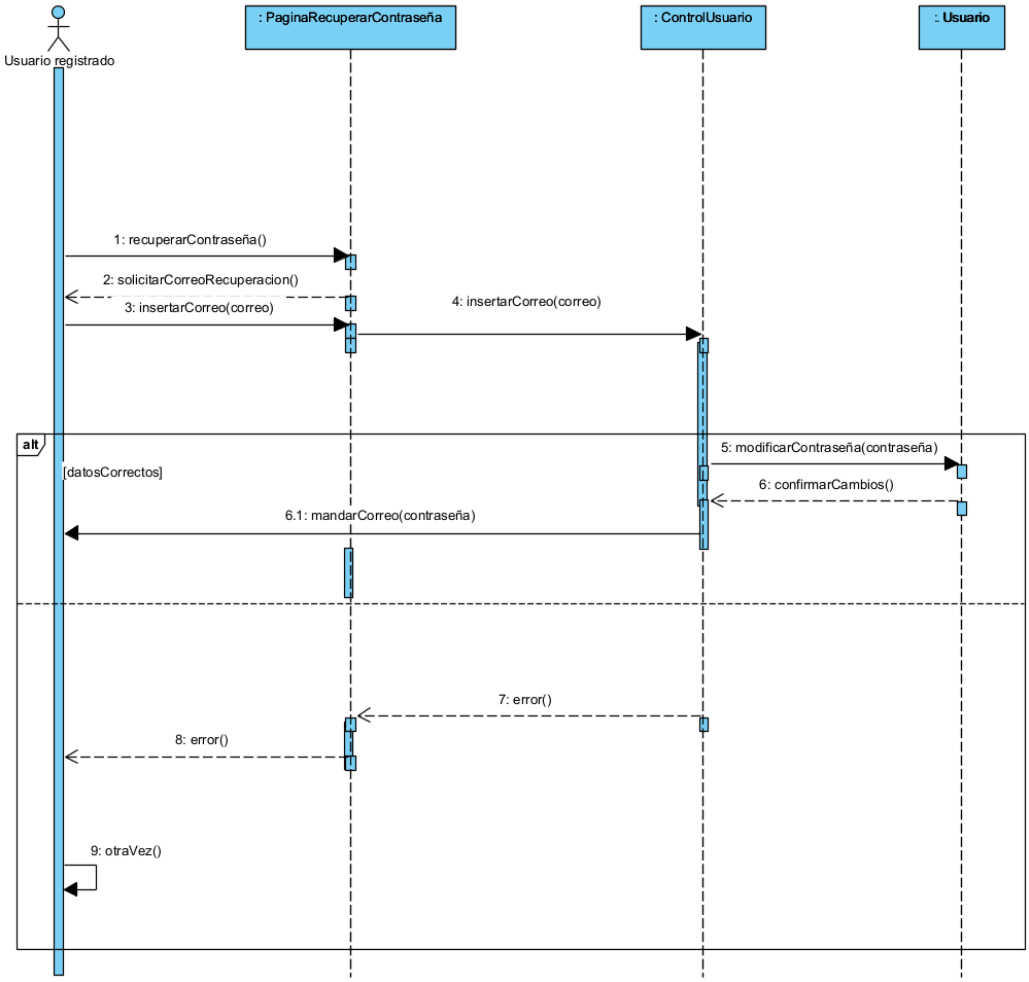

Figura 16: Diagrama de secuencia CU-0009 Recuperar contraseña

6.2 Diagramas de secuencia del paquete Gestión de dispositivos

Se mostrarán los diagramas de secuencia del paquete Gestión de dispositivos:

CU-00010 Completar datos del dispositivo:

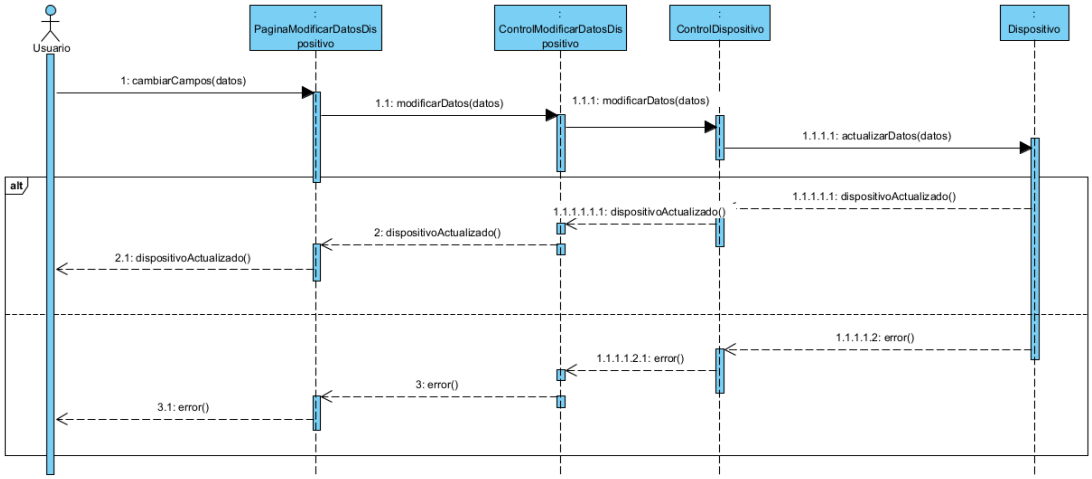

Figura 17: Diagrama de secuencia CU-0010 Completar datos del dispositivo

CU-0011 Guardar mediciones de sensores:

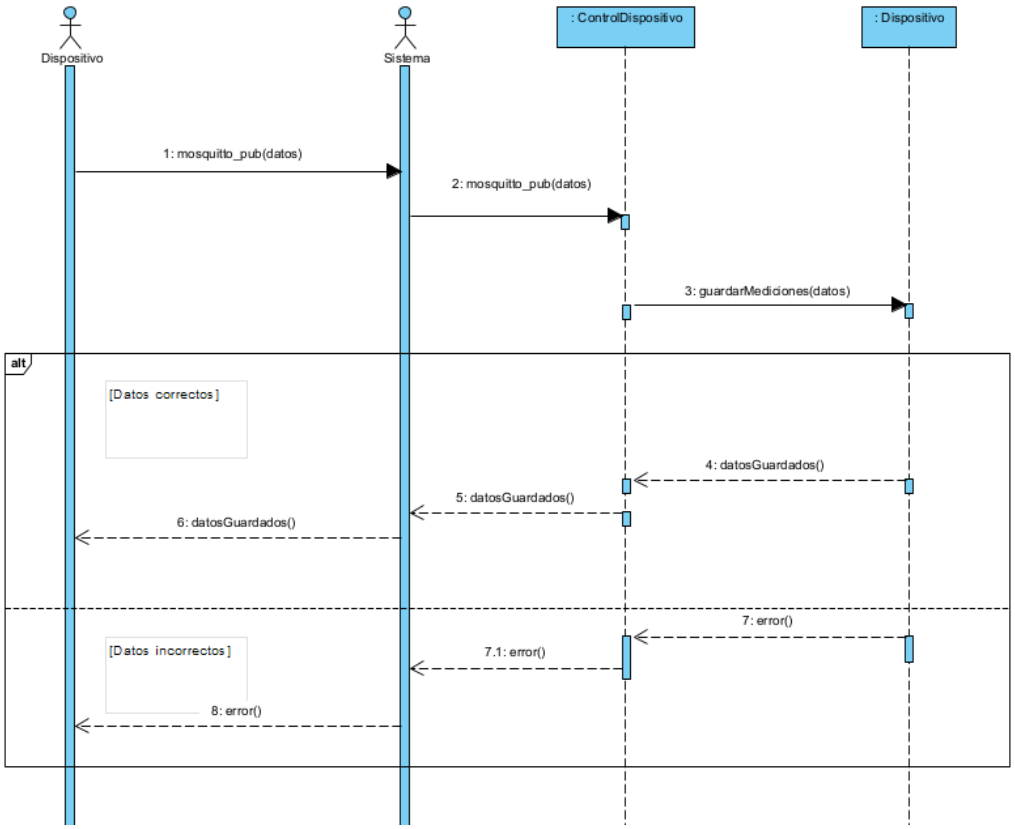

Figura 18: Diagrama de secuencia CU-0011 Guardar mediciones de sensores

CU-0012 Mandar mensaje de registro:

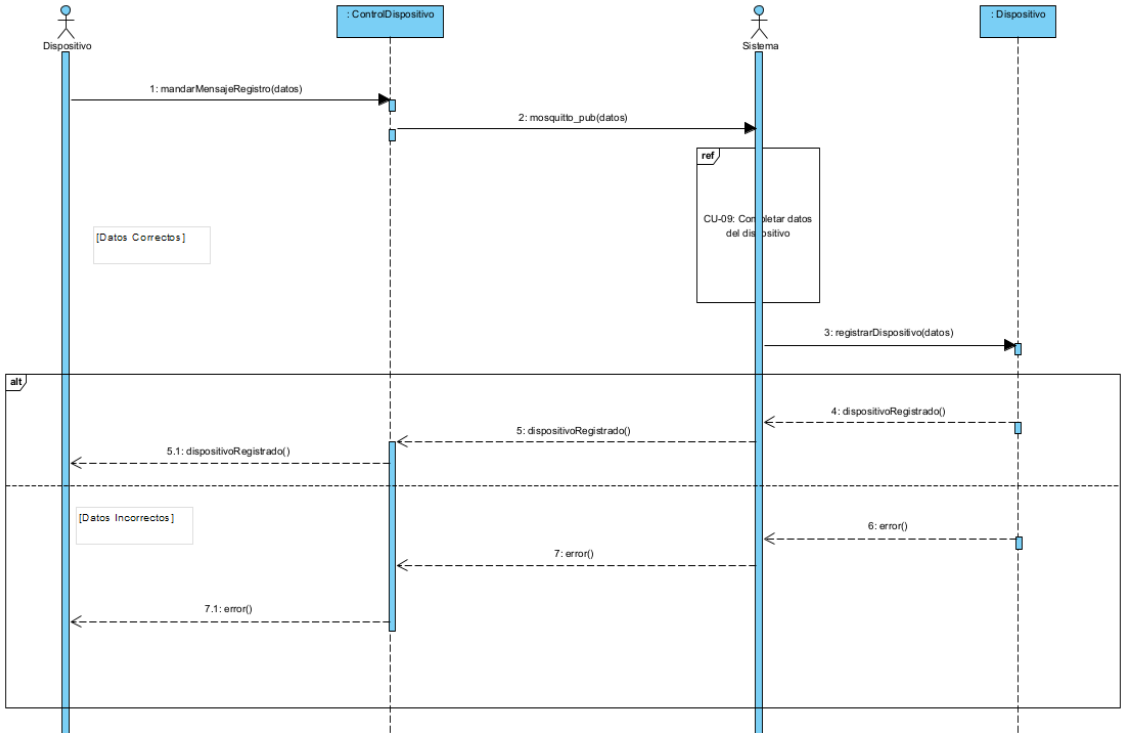

Figura 19: Diagrama de secuencia CU-0012 Mandar mensaje de registro

6.3 Diagramas de secuencia del paquete Gestión de la planificación

7. Se mostrarán los diagramas de secuencia del paquete Gestión de la planificación:

CU-0013 Calcular planificación de riego

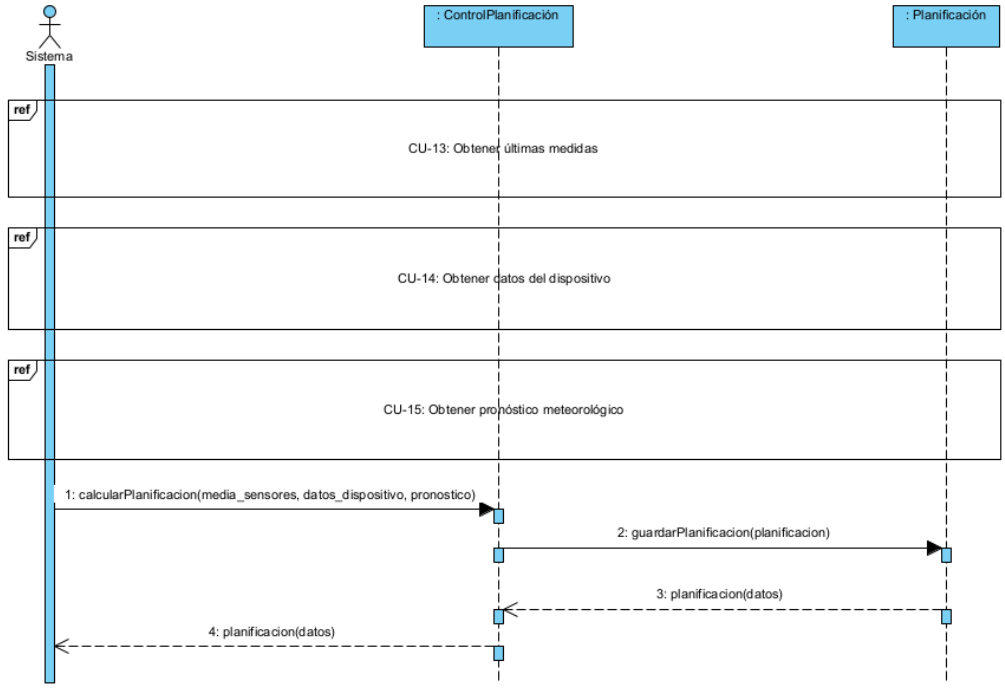

Figura 20: Diagrama de secuencia CU-0013 Calcular planificación de riego

CU-0014 Obtener últimas medidas:

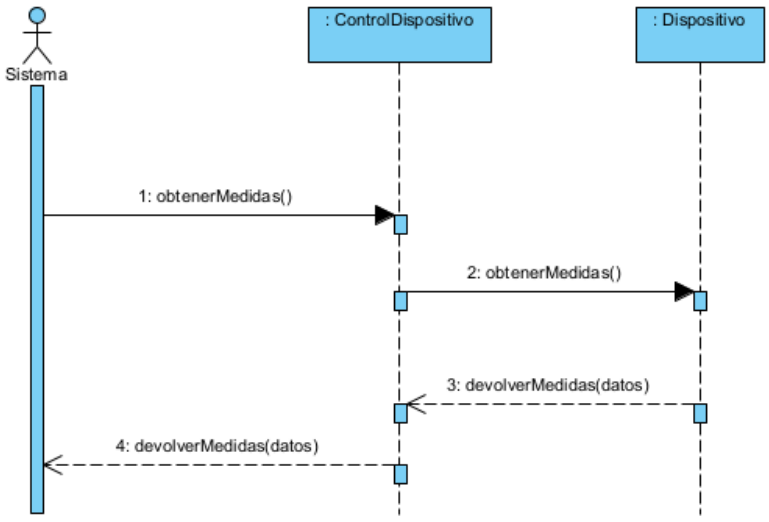

Figura 21: Diagrama de secuencia CU-0014 Obtener últimas medidas

CU-0015 Obtener datos del dispositivo:

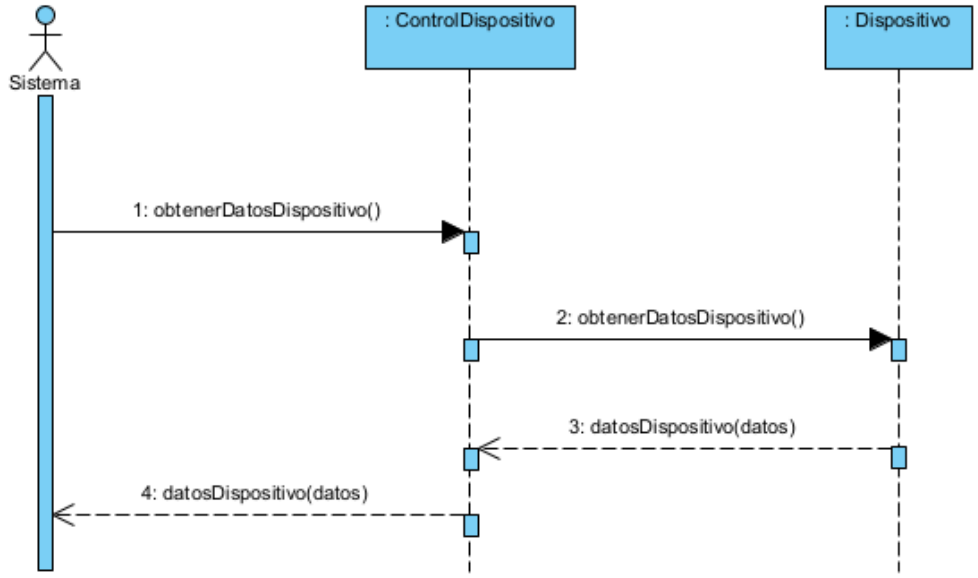

Figura 22: Diagrama de secuencia CU-0015 Obtener datos del dispositivo

CU-0016 Obtener pronóstico meteorológico:

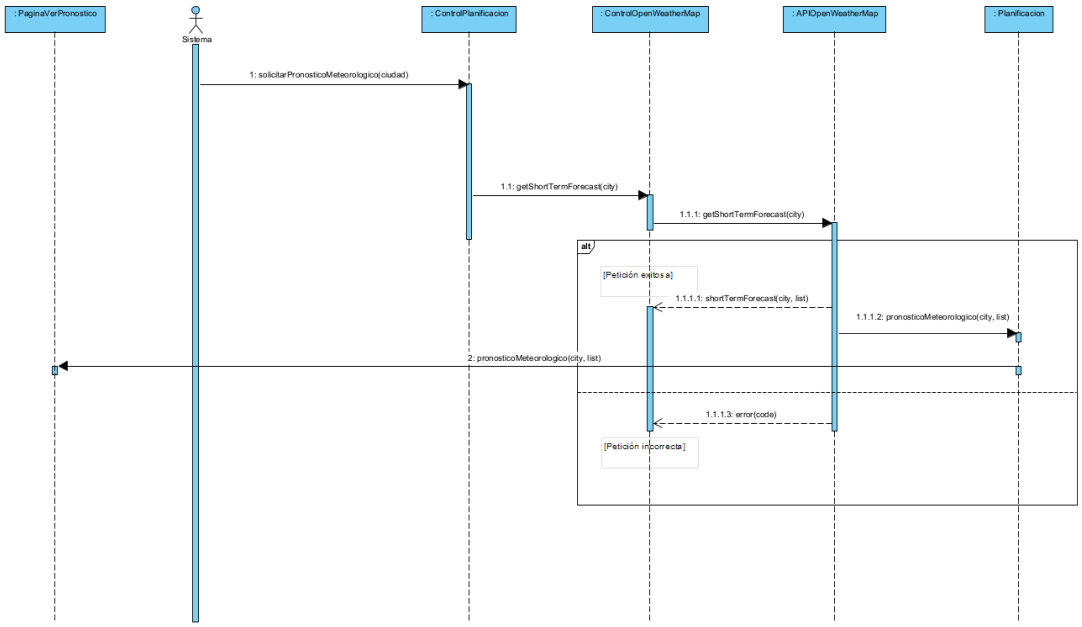

Figura 23: Diagrama de secuencia CU-0016 Obtener pronóstico meteorológico

CU-0017 Mandar orden de riego:

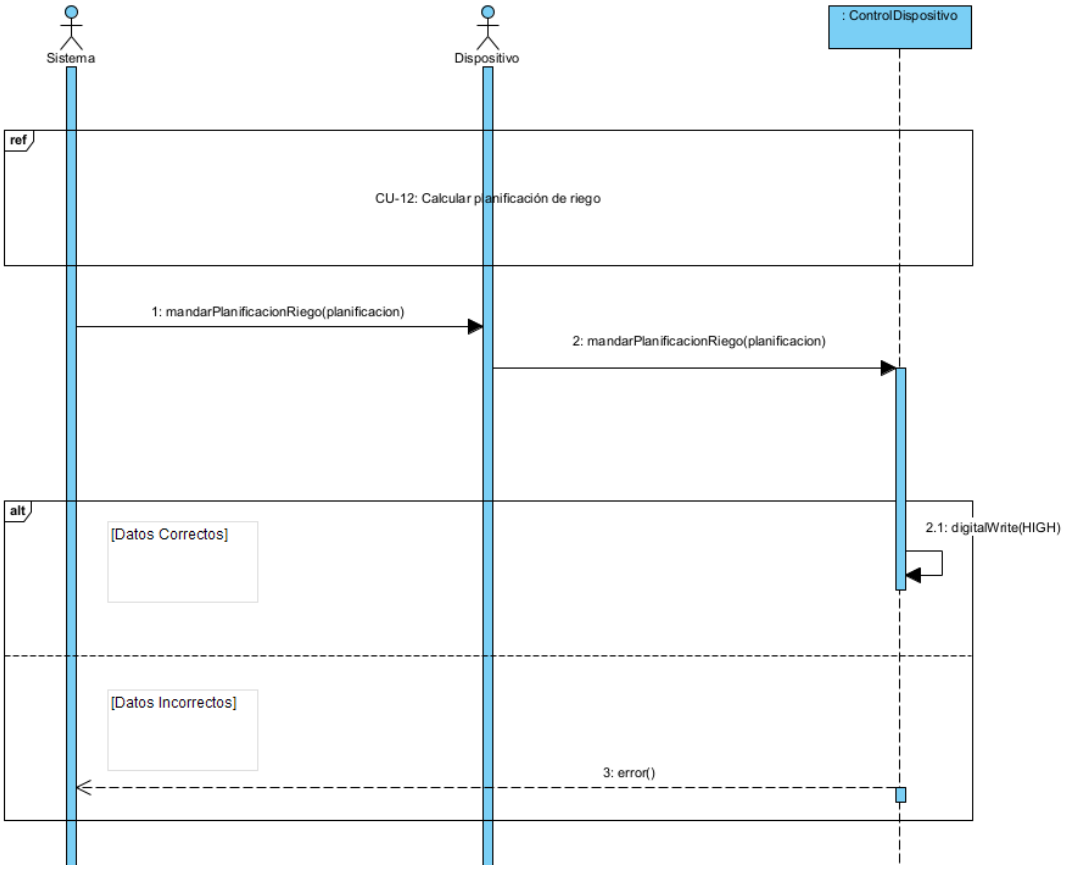

Figura 24: Diagrama de secuencia CU-0017 Mandar orden de riego

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 se va a explicar previamente las diferencias entre ambos tipos: [7]

-**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). [8]

-**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. [9]

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. En lo que respecta a los datos, al no existir relación entre ellos más allá de la secuencia temporal,

no hace falta tablas. Por último, MongoDB permite guardar los datos en formato JSON, que es como lo manda el propio dispositivo.

Combinar ambos tipos de bases de datos permite aprovechar lo mejor de cada una de ellas. Las bases de datos relacionales permitirán manejar las relaciones complejas de los datos, mientras que las bases de datos no relacionales manejan grandes cantidades de datos sin estructurar de manera eficiente. En el caso de este sistema, la base de datos relacional se usa como base de datos sobre usuarios y dispositivos; mientras que la no relacional se usa para datos procedentes de dispositivos IoT.

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 25: Colección Usuarios. BD SQL

```
MariaDB [usuarios_tfg]> select * from usuario;
+----+----------+--------------------------------------------------------------+-----------------+---------------------+----------------------------+
| id | username | password                                                     | mail            | fecha_registro      | dispositivos_registrados_id |
+----+----------+--------------------------------------------------------------+-----------------+---------------------+----------------------------+
| 28 | usuario1 | $2b$12$a6HooxSPUkhbkDnC0TkGKe3lld6j.uziMchg2G3e84ewh1sQQ.QLi | usuario1@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 26: Colección Dispositivos. BD SQL

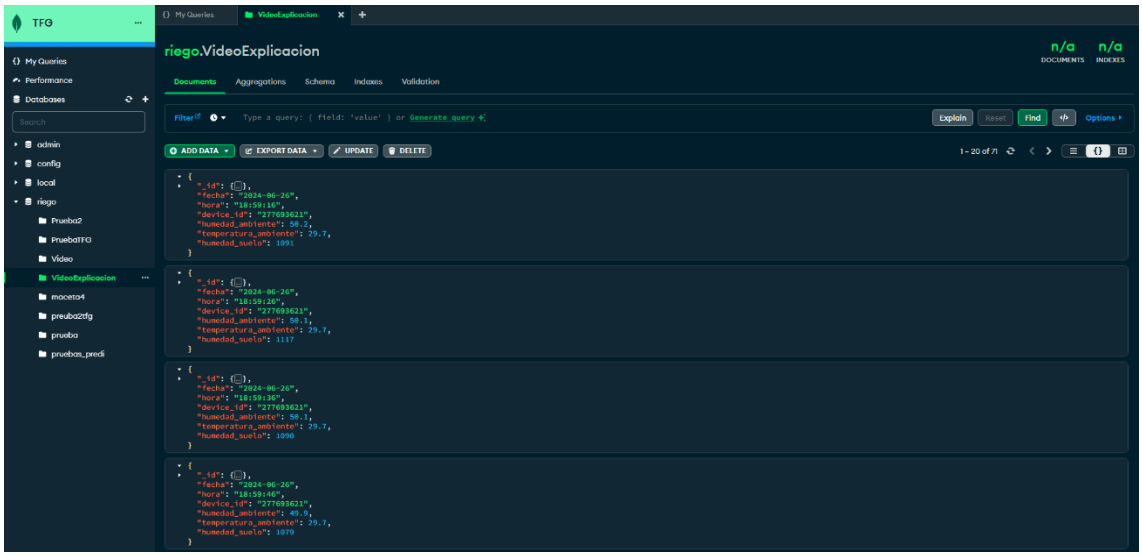

Figura 27: Mediciones. BD NoSQL

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.

9. Modelo de despliegue

10. 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.

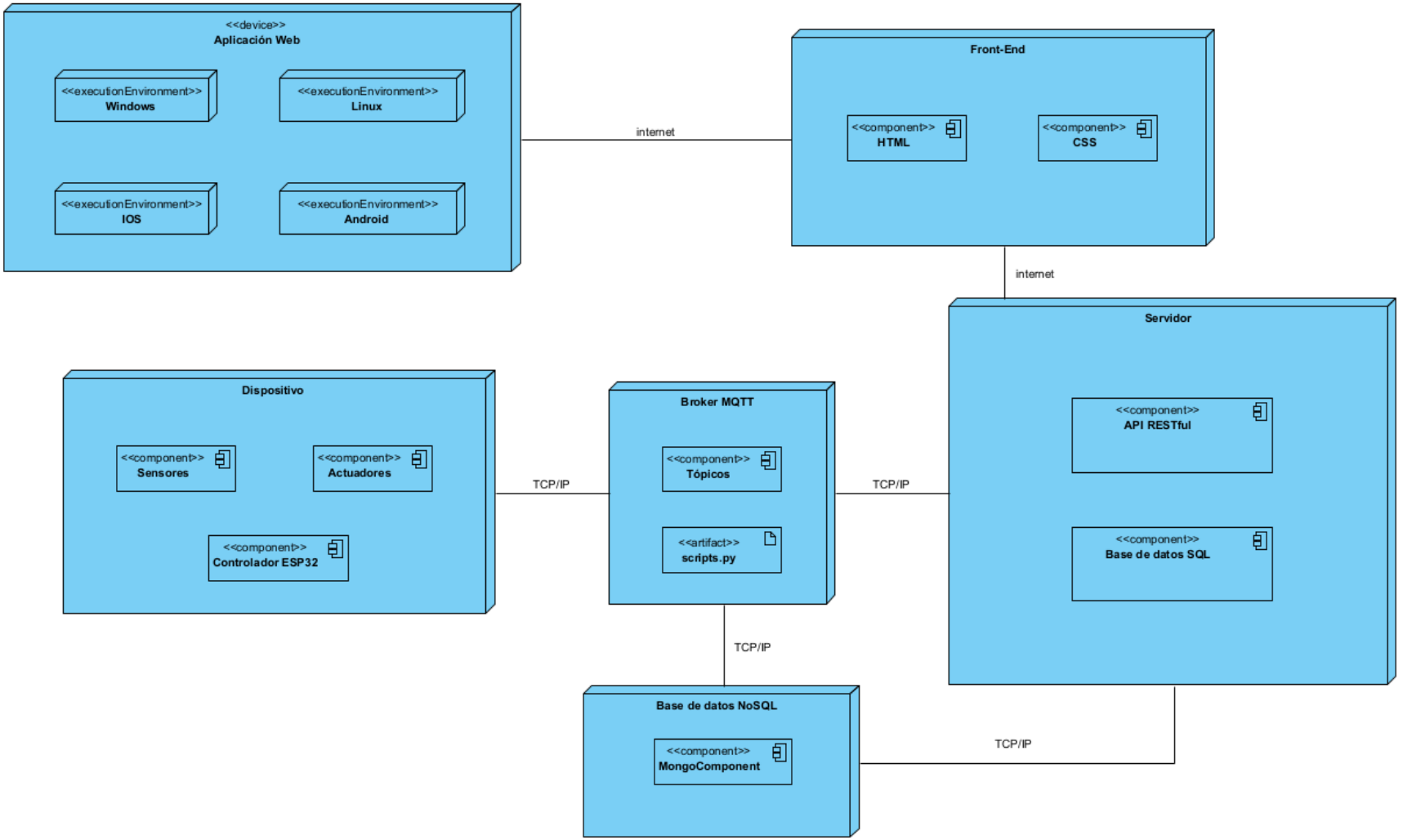

Figura 28: Modelo de despliegue

11. 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 [10]:

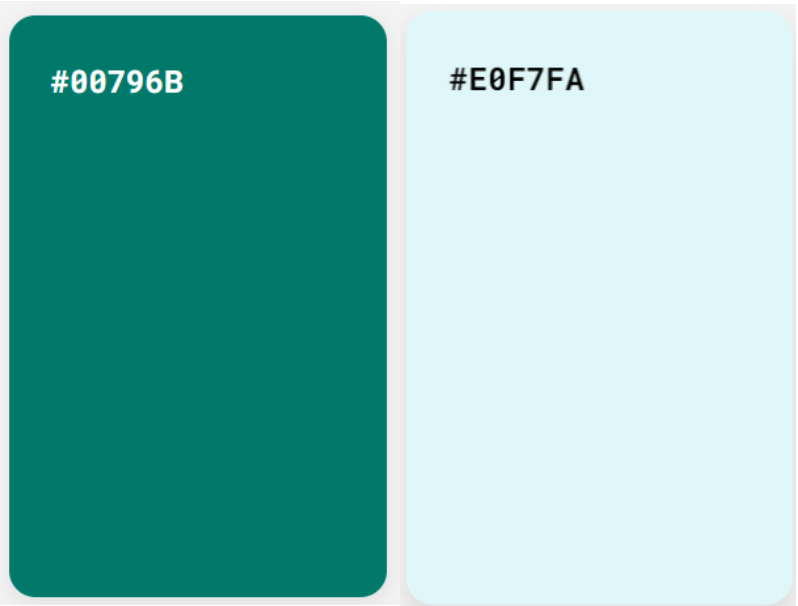

Figura 29: Código de colores seleccionados

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:

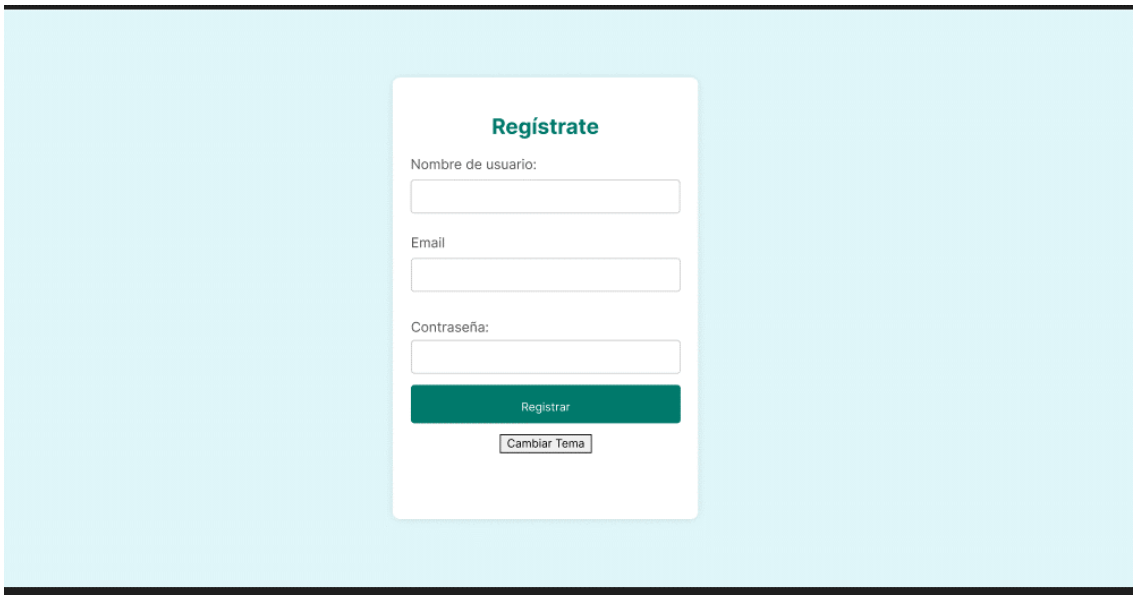

Figura 30: Prototipado digital. Ventana Registro. Tema claro

Figura 31: Prototipado digital. Ventana Iniciar Sesión. Tema oscuro

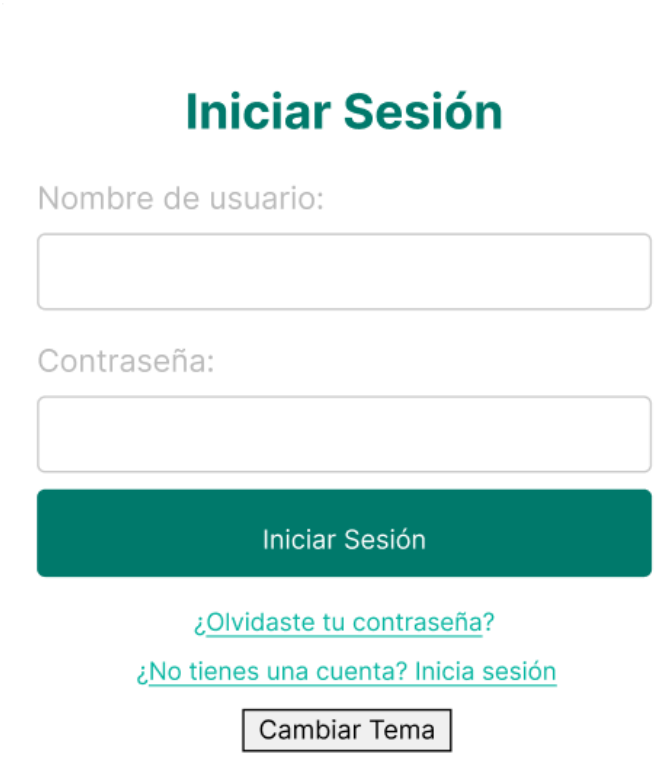

Figura 32: Prototipado digita. Ventana Iniciar Sesión. Tema claro

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 ambos temas de todas las ventanas, pero están implementadas.

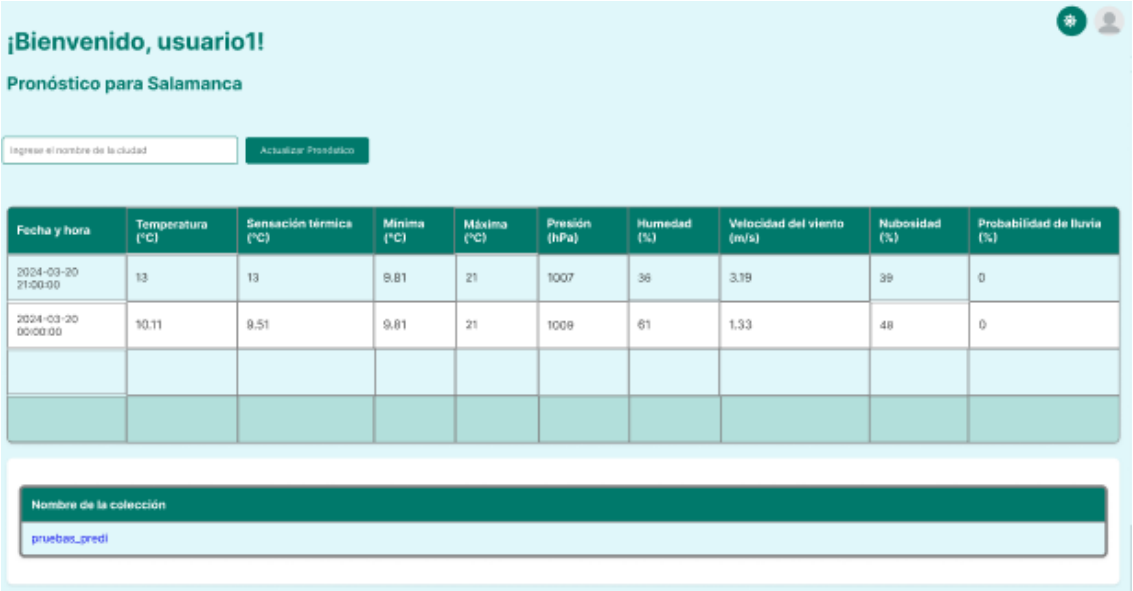

Figura 33: Prototipado digita. Ventana Principal

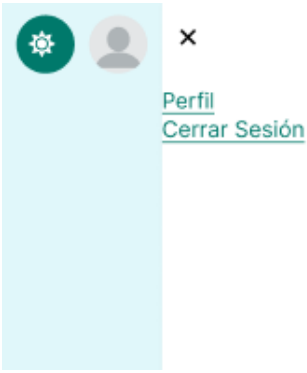

Figura 34: Prototipado digita. Barra lateral

Figura 35: Prototipado digital. Ventana modificar perfil

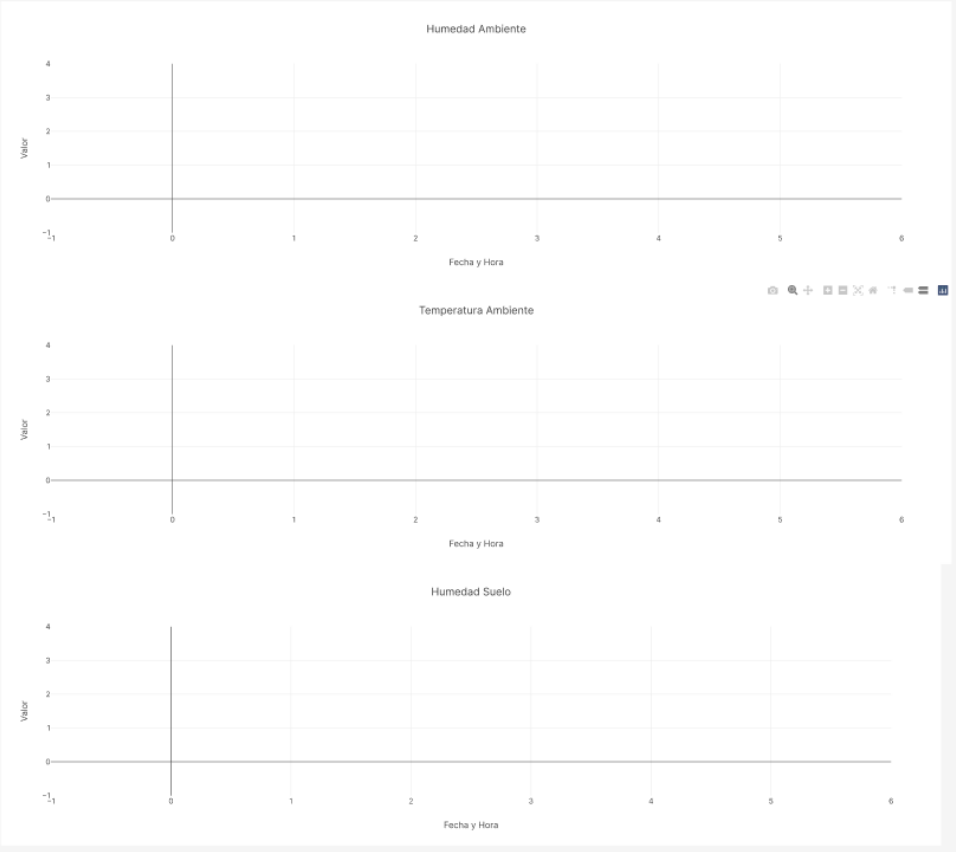

Figura 36: Prototipado digita. Ver datos de dispositivo

12. Referencias

[1] «IS II Tema 2 - UML - VISTA INTERACCIÓN.pdf». Accedido: 3 de julio de 2024. [En línea]. Disponible en: https://studium21.usal.es/pluginfile.php/852292/mod_resource/content/4/IS%20II%20Tema%202%20-%20UML%20-%20VISTA%20INTERACCI%C3%93N.pdf

[2] «Ideal Modeling & Diagramming Tool for Agile Team Collaboration». Accedido: 3 de julio de 2024. [En línea]. Disponible en: https://www.visual-paradigm.com/

[3] J. G. C. Sagredo, A. T. Espinosa, y M. M. Reyes, «patrón modelo vista controlador (mvc)».

[4] E. Bascón Pantoja, «El patrón de diseño Modelo-Vista-Controlador (MVC) y su implementación en Java Swing», *Acta Nova*, vol. 2, n.º 4, pp. 493-507, dic. 2004.

[5] «MQTT - The Standard for IoT Messaging». Accedido: 3 de julio de 2024. [En línea]. Disponible en: https://mqtt.org/

[6] J. A. Castañeda, C. I. O. Álvarez, y J. C. I. G. Ibarra, «DOCKERIZANDO UN LABORATORIO VIRTUAL DE PROGRAMACIÓN (VPL) Y MOODLE EN GOOGLE CLOUD», *Encuentro Int. Educ. En Ing.*, ago. 2019, doi: 10.26507/ponencia.212.

[7] M.-L. E. Chang y H. N. Chua, «SQL and NoSQL Database Comparison: From Performance Perspective in Supporting Semi-structured Data», en *Advances in Information and Communication Networks*, vol. 886, K. Arai, S. Kapoor, y R. Bhatia, Eds., en Advances in Intelligent Systems and Computing, vol. 886. , Cham: Springer International Publishing, 2019, pp. 294-310. doi: 10.1007/978-3-030-03402-3_20.

[8] «What is a relational database?» Accedido: 3 de julio de 2024. [En línea]. Disponible en: https://www.oracle.com/database/what-is-a-relational-database/

[9] «What Is NoSQL? NoSQL Databases Explained», MongoDB. Accedido: 3 de julio de 2024. [En línea]. Disponible en: https://www.mongodb.com/resources/basics/databases/nosql-explained

[10] «Información del color: Códigos de color, colores HTML, tonos y paletas, HEX y RGB», gradients.app. Accedido: 3 de julio de 2024. [En línea]. Disponible en: https://gradients.app/es