Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA INFORMÁTICA
Mención en Ingeniería de Software

Adquisición y visualización de datos radiométricos del GOA-UVa para su control

---

Alumno:
Jaime Saiz Losada

Tutores:
Yania Crespo González-Carvajal
Ramiro González Catón

A todas esas personas que me han permitido mostrarme tal y como soy

Agradecimientos

A mis padres, por haberme soportado todos estos años de carrera.

A mis amigos, en especial a Hector por ser como un hermano para mi y a Raquel por ser una de las mejores cosas que me han pasado en la vida.

A Paula, por ayudarme durante todos estos años a superar los obstáculos que se me han ido presentado.

Y finalmente, gracias a Yania, a Ramiro, a David y a los miembros del GOA-UVa por darme la oportunidad de hacer este proyecto y ayudarme durante su desarrollo.

Resumen

El Grupo de Óptica Atmosférica de la Universidad de Valladolid (GOA-UVa) cuenta con una serie de sensores destinados a medir la radiación solar, en sus distintos componentes, a nivel de superficie. Estos sensores almacenan las medidas en ficheros de texto de difícil interpretación, para su posterior procesado.

En este trabajo se propone la creación de dos programas de escritorio que permitan el almacenamiento de los datos obtenidos en una base de datos y la visualización de los mismos mediante una interfaz gráfica, mejorando así el control del GOA-UVa sobre los instrumentos radiométricos y la calidad de las medidas obtenidas.

Para el desarrollo de dichos programas, emplearemos el lenguaje de programación Python y la tecnología de bases de datos relacionales MySQL.

Abstract

The Grupo de Óptica Atmosférica de la Universidad de Valladolid (GOA-UVa) has a series of sensors designed to measure solar radiation, in its different components, at the surface level. These sensors store the measurements in text files that are difficult to interpret, for later processing.

In this work we propose the creation of two desktop programs that allow the storage of the data obtained in a database and their visualization through a graphical user interface (GUI), thus improving the control of the GOA-UVa over the radiometric instruments and the quality of the measurements obtained.

For the development of these programs, we will use the Python programming language and MySQL relational database technology.

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

1.1. Contexto

El Grupo de Óptica Atmosférica de la Universidad de Valladolid (GOA-UVa) [10] es un grupo de investigación creado en 1996, cuya principal línea de investigación es el estudio de los aerosoles atmosféricos a través de métodos ópticos y los distintos componentes de la radiación solar. Para entender mejor la utilidad de su trabajo, vamos a explicar brevemente cómo distintos factores, como por ejemplo los aerosoles, pueden llegar a afectar al clima de la Tierra.

El sistema climático [12] se compone de distintos elementos como la atmósfera, la superficie terrestre, la nieve y el hielo, los océanos junto con otras superficies de agua y los seres vivos; siendo la atmósfera el que más influye en el clima. El sistema climático va cambiando con el paso del tiempo por la influencia de su funcionamiento interno y por factores externos, como las erupciones volcánicas, las alteraciones a la composición atmosférica causadas por el ser humano y, en especial para nuestro caso, la variaciones en la radiación solar que llegan al planeta Tierra.

Teniendo en cuenta esto, el GOA-UVa pretende, mediante el uso de radiómetros, medir los distintos componentes de la radiación solar que recibimos, para estudiar cómo va cambiando el clima. Para ello, utiliza distintos tipos de sensores que se encargan de medir diferentes tipos de radiaciones, como por ejemplo:

- **Albedómetro CM 7B:** este sensor es empleado para medir la radiación solar global y difusa.
- **Pirheliómetro CHP 1:** este sensor es empleado para medir la radiación solar directa.
- **Piranómetro ultravioleta UVB-1:** este sensor es empleado para medir la radiación solar ultravioleta.

Para ayudarles en sus investigaciones, se propone el desarrollo de varios programas para visualizar y almacenar los datos que reciben en sus distintos sensores, facilitándoles así el estudio del comportamiento de las distintas radiaciones a lo largo del tiempo.

1.2. Motivación

La idea de este proyecto surgió principalmente por parte del GOA-UVa, dado que este fue uno de los Trabajos de Fin de Grado ofrecidos en convenio con la Escuela de Ingeniería Informática, con el objetivo de desarrollar un software para apoyar sus investigaciones y mantener controlados sus datos.

Por mi parte, no tenía claro qué hacer como TFG, y decidí asistir a la charla en la que se presentaron los distintos proyectos ofrecidos por las empresas y otras entidades que tenían convenio con la Escuela, entre las cuales se encontraba el que nos atañe. Ahí fue donde despertaron mi interés y tras varias reuniones con ellos me decidí a llevar a cabo este proyecto como TFG.

1.3. Objetivos

El objetivo principal de este trabajo es el facilitar al GOA-UVa la visualización y manejo de los datos obtenidos por sus radiómetros.

Siendo más específicos, los objetivos del proyecto son:

- Desarrollar un programa que procese los medidas almacenadas en archivos y las almacene en una base de datos.
- Desarrollar un programa que permita visualizar los datos almacenados de forma gráfica, además de poder añadir nuevos elementos a la base de datos para mantenerla actualizada.
- Asegurarse de que todo el proyecto funcione en un sistema Linux.

1.4. Objetivos de formación

Adicionalmente, el Trabajo de Fin de Grado complementa la formación obtenida en los estudios por lo que al objetivo principal del proyecto lo acompañan algunos objetivos de formación personal poniendo el foco en ampliar mi formación de cara a mi futura carrera profesional.

Siendo más específicos, mis objetivos de formación son:

- Obtener una mejor formación en programación en Python de cara a mi futuro laboral, dado que es un lenguaje con el que prácticamente no se trabaja en la carrera.
- Recordar y afianzar los conceptos y buenas prácticas de las bases de datos relacionales y en particular con el sistema gestor MySQL, desarrollando un programa que emplee esta tecnología.

1.5. Estructura de la memoria

Este documento se estructura de la siguiente forma:

Capítulo 1 Introducción: Describe el contexto en el que se ha creado la aplicación, y la motivación y objetivos del proyecto.

Capítulo 2 Requisitos y planificación: Describe las historias de usuario obtenidas durante las reuniones con el equipo de investigación GOA-UVa y la planificación seguida durante el desarrollo, además de la explicación de las metodologías seguidas y de las estimaciones de costes.

Capítulo 3 Análisis: Documenta el resultado del análisis de los requitos.

Capítulo 4 Tecnologías utilizadas: Describe las herramientas software empleadas para el desarrollo de la memoria y del sistema.

Capítulo 5 Diseño: Explicación de las decisiones de diseño tomadas durante el desarrollo, tales como los patrones arquitectónicos y de diseño adoptados, además de los bocetos de las interfaces de usuario del sistema y los diagramas que explican la arquitectura del sistema.

Capítulo 6 Implementación y pruebas: Explicación de la estructura del código, de los problemas de desarrollo que surgieron durante la implementación de los programas y de las pruebas realizadas durante el proceso de desarrollo para asegurar el correcto funcionamiento del sistema.

Capítulo 7 Seguimiento del proyecto: Explicación de como ha ido avanzando el proyecto a lo largo de todo el proceso de desarrollo, haciendo una análisis sprint a sprint.

Capítulo 8 Conclusiones: Pensamientos finales tras la realización del trabajo, además de mejoras que se le pueden aplicar en un futuro a los programas.

Bibliografía: Fuentes bibliográficas y recursos empleados para la realización del trabajo.

Anexo A Manuales: Incluye manuales de mantenimiento, de instalación, despliegue, y de uso.

Anexo B Resumen de enlaces adicionales: Incluye enlaces de interés sobre el proyecto: el repositorio de código y de descarga de Python.

Capítulo 2

Requisitos y Planificación

2.1. Metodología

Para este trabajo hemos decidido una metodología de planificación dinámica basada en Scrum, dado que los requisitos deseados por el cliente no estaban fijos desde el principio del proyecto, aunque en este caso emplearemos una versión modificada de la misma (véase Sección 2.1.6).

2.1.1. ¿Qué es Scrum?

Scrum [30] [29] es una forma de trabajo ágil creada en la década de los 90, con el objetivo de coordinar equipos de trabajo a la hora de resolver problemas complejos, todo ello mediante un proceso iterativo e incremental, con el objetivo de mantener un control sobre dicho proyecto y sus riesgos.

Esta forma de trabajo cuenta con tres pilares fundamentales:

- **Transparencia:** los resultados del trabajo deben de ser perceptibles, ya que las decisiones que se toman durante el desarrollo se basan en el estado de sus tres artefactos (véase Sección 2.1.4), con lo cual ante un artefacto poco transparente se puede llegar a tomar una decisión perjudicial.
- **Inspección:** se realizará una inspección frecuente de los artefactos (véase Sección 2.1.4) y del proceso de trabajo con el objetivo de prevenir y detectar cualquier problema. Este pilar se cumple gracias a sus cinco eventos (véase Sección 2.1.3).
- **Adaptación:** se deberán ajustar los procesos de trabajo lo antes posible, en caso de que se note un desvío de lo planeado o se obtenga algún resultado no deseado.

Gracias a estos pilares, Scrum es el marco de trabajo ideal para la realización de un proyecto software del que no se tenga total certeza sobre sus requisitos, o de que estemos seguros que estos van a ir cambiando a lo largo del desarrollo.

2.1.2. Equipo Scrum

Un equipo Scrum [29] está formado por un número reducido de personas, siendo la media 10 miembros, las cuales cuentan con las suficientes habilidades como para poder realizar las distintas tareas que se presentarán a lo largo del desarrollo y generar valor en cada sprint. Podemos dividir el equipo Scrum en tres roles:

- **Product Owner:** representante de los stakeholders y de sus intereses, mediante la gestión del Product Backlog (véase Sección 2.1.4), que pretende maximizar el valor resultante del proyecto.
- **Scrum Master:** persona encargada de asesorar a los distintos miembros del equipo y la organización en la metodología Scrum, asegurándose de que esta se mantenga y liderando así el proyecto.
- **Development Team:** desarrolladores encargados de generar valor para crear los incrementos (véase Sección 2.1.4) de cada sprint (véase Sección 2.1.3), todo ello siendo previamente planeado en el Sprint Backlog (véase Sección 2.1.4).

2.1.3. Eventos Scrum

Como dijimos anteriormente, estos eventos [29] aseguran que se cumpla el pilar de Inspección (véase Sección 2.1.1). Estos eventos ofrecen la oportunidad de revisar y adaptar los artefactos Scrum (véase Sección 2.1.4) de forma regular evitando así cualquier tipo de evento no planeado.

Scrum cuenta con cinco eventos distintos:

- **Sprint:** son la parte fundamental de Scrum, ya que su objetivo es la de generar valor y con ello incrementos (véase Sección 2.1.4). Son eventos de una duración fija de dos a cuatro semanas que, cuando finaliza uno, el siguiente empieza inmediatamente, además de que al principio de cada uno de ellos se congelan las historias de usuario (véase Sección 2.1.5) y no se pueden modificar hasta el final del sprint. Este evento abarca al resto de los eventos Scrum: el Sprint Planning, el Daily Scrum, el Sprint Review y el Sprint Retrospective.
- **Sprint Planning:** evento al principio de cada sprint en el que se discute qué trabajo se va a realizar en éste, creando así el Sprint Backlog (véase Sección 2.1.4) a partir del Sprint Goal, el método a seguir para realizarlo y los items del Product Backlog elegidos durante la misma (véase Sección 2.1.4). Tiene una duración máxima de unas ocho horas para un sprint de un mes.

- **Daily Scrum:** reunión de unos 15 minutos en la que el Development Team (véase Sección 2.1.2) habla sobre su progreso de cara al Sprint Goal, modificando el Sprint Backlog (véase Sección 2.1.4) en caso necesario para así ajustar el trabajo futuro.
- **Sprint Review:** evento al final del sprint, siendo éste el penúltimo, en el cual se revisan los resultados obtenidos en el sprint y determinando así las futuras acciones del equipo. Tiene una duración máxima de unas cuatro horas para un sprint de un mes.
- **Sprint Retrospective:** evento final del sprint en el cual el equipo Scrum revisa como se desarrolló, haciendo especial énfasis en qué se hizo bien, qué problemas se encontraron y en cómo fueron solucionados o no, e identificando los cambios más beneficiosos; mejorando así la eficiencia y la calidad.

2.1.4. Artefacto Scrum

Como dijimos en un sección anterior, los artefactos Scrum [29] fueron diseñados para maximizar la transparencia de la información más importante (véase Sección 2.1.1), siendo estos una representación de trabajo o valor.

En este caso contamos con tres artefactos:

- **Product Backlog:** lista ordenada de items que definen cómo el equipo Scrum puede mejorar el producto. Estos items pueden ir siendo divididos en unos más pequeños y precisos durante el desarrollo, siendo esto el refinamiento del Product Backlog. Este contiene el Product Goal, que es el estado del producto que el equipo Scrum pretende alcanzar a largo plazo.
- **Sprint Backlog:** está compuesto por el Sprint Goal, siendo este el objetivo del sprint (véase Sección 2.1.3); por los items seleccionados del Product Backlog por el Development Team (véase Sección 2.1.2) y por un plan para cumplir el incremento. Es un plan creado por y para los desarrolladores, con el objetivo de poder comprobar fácilmente su progreso y el trabajo restante de cara a alcanzar el Sprint Goal, además de ser actualizado durante el sprint a medida que se va adquiriendo conocimiento.
- **Incremento:** es el valor obtenido durante un sprint (véase Sección 2.1.3), siendo este creado cuando uno de los items del Product Backlog cumple su Definition of Done, la cual es una definición formal de las cotas de calidad que debe cumplir su correspondiente item para ser añadido como un incremento.

2.1.5. Items de backlog: Épicas e historias de usuario

Ya hemos mencionado que tanto el Product Backlog como el Sprint Backlog (véase Sección 2.1.4) están formados por unos 'items', pero aun no hemos mencionado de qué se tratan.

Hay dos tipos de items:

- **Épicas [3]:** es un conjunto de tareas generales que se pueden ir refinando en tareas más específicas, denominadas historias de usuario. Estas épicas e historias de usuario se suelen escribir en un lenguaje informal, de tal forma que sean entendibles para cualquier persona.

- **Historias de usuario [4]:** tarea específica obtenida a partir de refinar las épicas del Product Backlog, siendo esta la unidad más pequeña de trabajo; que son descritas de manera informal y general desde la perspectiva del usuario final.

2.1.6. Versión adaptada

En nuestro caso emplearemos una versión modificada del marco de trabajo ágil Scrum, dado que no contamos con un equipo tan grande y este no deja de ser un trabajo académico.

Nuestro equipo Scrum estará formado por Ramiro González Catón como Product Owner, Yania Crespo González-Carvajal como Scrum Master y el alumno como único miembro del Development Team. Además, en vez de realizar Daily Scrums, realizaremos un versión semanal, es decir, una Weekly Scrum.

2.2. Planificación inicial

Se realizó la planificación inicial contando con hacer una media de cinco horas de trabajo al día durante los cinco días de diario, dejando los fines de semana para documentar o realizar la cantidad de trabajo necesario antes de la próxima reunión, sumando en total las 300 horas de la asignatura [33].

El trabajo se organizará en sprints (véase Sección 2.1.3) de dos semanas, dejando así un total de 6 sprints para completar la asignatura, aunque dado que el Development Team (véase Sección 2.1.2) está formado por una persona, añadiremos un sprint al final del desarrollo con el objetivo de cubrir posibles retrasos y documentar el trabajo. Además, se hará previamente un 'Sprint 0', que se empleará para planificar el trabajo y la formación del alumno.

Con toda esta información, se estima que el trabajo durará unas **350 horas** en caso de que se cumplan los plazos, pero, en caso de que se tenga que emplear el sprint de refuerzo, se sumarían otras 50 horas, dando un total de **400 horas**.

Al ser las reuniones semanales y los sprints de dos semanas, una de las reuniones semanales se dedicará a los eventos mencionados: Sprint Review, Sprint Retrospective y Sprint Planning.

En la Tabla 2.1 podemos ver la calendarización de los sprints y sus eventos.

Sprint	Inicio	Fin
Sprint 0	27/02/2023	13/03/2023
Sprint Planning	27/02/2023	
Weekly Scrum	06/03/2023	
Sprint Review y Sprint Retrospective	13/03/2023	
Sprint 1	13/03/2023	27/03/2023
Sprint Planning	13/03/2023	
Weekly Scrum	20/03/2023	
Sprint Review y Sprint Retrospective	27/03/2023	
Sprint 2	27/03/2023	10/04/2023
Sprint Planning	27/03/2023	
Weekly Scrum	10/04/2023	
Sprint Review y Sprint Retrospective	10/04/2023	
Sprint 3	10/04/2023	24/04/2023
Sprint Planning	10/04/2023	
Weekly Scrum	17/04/2023	
Sprint Review y Sprint Retrospective	24/04/2023	
Sprint 4	24/04/2023	08/05/2023
Sprint Planning	24/04/2023	
Weekly Scrum	02/05/2023	
Sprint Review y Sprint Retrospective	08/05/2023	
Sprint 5	08/05/2023	22/05/2023
Sprint Planning	08/05/2023	
Weekly Scrum	15/05/2023	
Sprint Review y Sprint Retrospective	22/05/2023	
Sprint 6	22/05/2023	05/05/2023
Sprint Planning	22/05/2023	
Weekly Scrum	29/05/2023	
Sprint Review y Sprint Retrospective	05/06/2023	
Sprint de refuerzo 1	05/06/2023	19/06/2023
Sprint Planning	05/06/2023	
Weekly Scrum	12/06/2023	
Sprint Review y Sprint Retrospective	19/06/2023	
Limite de solicitud de defensa primera convocatoria	23/06/2023	
Limite de solicitud de defensa segunda convocatoria	11/07/2023	

Tabla 2.1: Calendarización de los sprints

2.3. Análisis de riesgos

Todo proyecto plantea una serie de riesgos en mayor o en menor medida que pueden llegar a afectar al desarrollo de este, pudiendo llegar hasta el punto de hacer que fracase. Por esto, conviene realizar un análisis de riesgos con antelación para crear planes de mitigación y de contingencia para cada uno de ellos (véase Sección 2.3.2).

2.3.1. ¿Qué es un riesgo?

Según la guía PM-BOOK [19], un riesgo es un evento o condición incierta, que si ocurre, puede tener un impacto negativo o positivo en los objetivos de un proyecto; siendo clave en esta definición el factor de incertidumbre de las consecuencias de estos, pudiendo ser positivas o negativas.

2.3.2. ¿Qué pasos hay que seguir?

La gestión de riesgos incluye los siguientes pasos [6]:

1. **Identificación de riesgos:** consiste en enumerar los distintos riesgos que se pueden presentar. Esto se puede conseguir por varios métodos, pero los principales son: el uso de listas de riesgos de proyectos previos con sus correspondientes contramedidas; y el *brainstorming*, que consiste en reunir a las distintas partes implicadas del proyecto para que compartan sus inquietudes, y así elaborar una lista de riesgos a partir de dicha reunión.
2. **Análisis y priorización de riesgos:** identificar los riesgos que son más graves, y por lo tanto cuáles son los que merecen una mayor atención del equipo. Esto se consigue estableciendo la probabilidad de que dicho riesgo se materialice y el impacto que tendría en caso de hacerlo, dando como resultado la prioridad de dicho riesgo (véase Sección 2.3.3).
3. **Planificación de riesgos:** establecer planes de mitigación; para establecer qué acciones tomar para reducir la probabilidad de materialización del riesgo; y planes de contingencia; para establecer qué acciones tomar para reducir el impacto del riesgo en el caso de que ocurra; para cada uno de los riesgos.
4. **Seguimiento del proyecto:** comprobar los riesgos con el objetivo de conocer su estado y actualizarlos.

2.3.3. Gestión de riesgos inicial

En esta sección vamos a mostrar los riesgos identificados y analizados inicialmente, aunque dado que vamos a desarrollar un proceso basado en Scrum (véase Sección 2.1.1), a lo largo

Probabilidad/ Impacto	Baja	Media	Alta
Bajo	Bajo	Bajo	Medio
Medio	Bajo	Medio	Alto
Alto	Medio	Alto	Alto

Tabla 2.2: Cálculo de la prioridad de un riesgo

Impacto	Acciones a tomar
Bajo	Ninguna
Medio	Acciones de mitigación
Alto	Acciones de contingencia

Tabla 2.3: Acciones a tomar según la prioridad de un riesgo

del desarrollo pueden llegar a presentarse nuevos riesgos; los cuales iremos mostrando en el Capítulo 7; con lo cual estos tienen que pasar por el mismo proceso que el resto (véase Sección 2.3.2).

Durante el análisis de los riesgos, debemos establecer una prioridad para cada uno de ellos en base a su probabilidad de aparición e impacto en caso de que ocurra, y así saber en cuales debemos fijarnos con mayor atención. Para saber como la establecemos tenemos la Tabla 2.2.

En función de la prioridad asignada, habrá que tomar unas acciones u otras y esto lo podemos ver de forma generalizada en la Tabla 2.3.

Podemos ver los riesgos identificados inicialmente desde la Tabla 2.4 hasta la 2.15.

ID	R001
Nombre	Falta de tiempo debido a las prácticas en empresa
Descripción	Se corre el riesgo de que las prácticas de empresa reduzcan mucho el tiempo de trabajo dedicado al TFG
Categoría	Programación
Vulnerabilidad	La cantidad de tiempo que requieran las prácticas de empresa es variable
Amenaza	No poder cumplir el plazo por no dedicar el tiempo suficiente al TFG debido a sprints o trabajos excesivos en las prácticas de empresa
Probabilidad	Baja
Impacto	Medio
Prioridad	Baja
Acciones de mitigación	1. Seleccionar las prácticas de empresa para minimizar el impacto del trabajo en el TFG 2. Dedicar un tiempo estable diariamente a avanzar el TFG
Acciones de contingencia	1. Pedir una ampliación de plazo de entrega 2. Replanificar el proyecto adecuándome a los nuevos tiempos

Tabla 2.4: Riesgo 01: Falta de tiempo debido a las prácticas en empresa

ID	R002
Nombre	Falta de tiempo debido a los exámenes de Sistemas Empotrados
Descripción	Se corre el riesgo de que la ultima asignatura optativa, que se cursa a la vez que se realiza el TFG, interfiera con el desarrollo del proyecto
Categoría	Programación
Vulnerabilidad	Los exámenes de Sistemas Empotrados requieren un tiempo de estudio dedicado
Amenaza	No poder cumplir el plazo por no dedicar el tiempo suficiente al TFG debido a los exámenes de Sistemas Empotrados
Probabilidad	Media
Impacto	Alto
Prioridad	Alta
Acciones de mitigación	1. Avanzar la parte teórica de la asignatura leyendo la bibliografía recomendada pronto en el cuatrimestre 2. Realizar tutorías con los profesores para reducir el tiempo de dedicación necesario 3. Planificar el proyecto teniendo en cuenta la asignatura
Acciones de contingencia	1. Replanificar el proyecto adecuándome a los nuevos tiempos 2. Pedir una ampliación de plazo de entrega

Tabla 2.5: Riesgo 02: Falta de tiempo debido a los exámenes de Sistemas Empotrados

ID	R003
Nombre	Recursos insuficientes para la realización del TFG
Descripción	Se corre el riesgo de que la realización del TFG requiera acceso a hardware especifico y que el acceso al mismo no esté garantizado
Categoría	Hardware
Vulnerabilidad	No sabemos que recursos se pueden necesitar para la realización del TFG
Amenaza	No poder realizar las acciones necesarias a tiempo
Probabilidad	Baja
Impacto	Medio
Prioridad	Baja
Acciones de mitigación	1. Contactar con los coordinadores de los TFG para buscar posibilidades 2. Buscar alternativas viables 3. Preparar algún recurso backup por mi cuenta que pueda simular las pruebas
Acciones de contingencia	1. Replanificar el proyecto, reduciendo la importancia del recurso al que no se puede acceder en el mismo 2. Pedir una ampliación del plazo de entrega

Tabla 2.6: Riesgo 03: Recursos insuficientes para la realización del TFG

ID	R004
Nombre	Incapacitación
Descripción	Se corre el riesgo de que el alumno esté incapacitado y no pueda realizar el TFG en el tiempo previsto
Categoría	Personal
Vulnerabilidad	Pueden darse circunstancias, como caer enfermo, que me imposibiliten hacer el TFG
Amenaza	No tener el TFG terminado a tiempo para la entrega
Probabilidad	Baja
Impacto	Medio
Prioridad	Baja
Acciones de mitigación	1. Realizar la planificación del TFG, de tal forma que sobre cierto tiempo antes de la defensa, al que podemos denominar "horas de refuerzo"
Acciones de contingencia	1. Realizar el tratamiento necesario para curarme lo antes posible 2. Replanificar el proyecto adecuándome a los nuevos tiempos

Tabla 2.7: Riesgo 04: Incapacitación

ID	R005
Nombre	Perdida del TFG
Descripción	Se corre el riesgo de que pierda el documento de TFG durante su realización
Categoría	Proyecto
Vulnerabilidad	Puede haber errores que ocasionen la pérdida del TFG, ya sea por un error informático o físico, perdiendo así todo o parte de mi progreso
Amenaza	No tener el TFG a tiempo para la entrega
Probabilidad	Baja
Impacto	Alto
Prioridad	Media
Acciones de mitigación	1. Realizar copias de seguridad semanales tanto en la nube, como en dispositivos físicos de almacenamiento extraíbles 2. Llevar a cabo un control de versiones del trabajo con GitLab
Acciones de contingencia	1. Replanificar el proyecto adecuándome a los nuevos tiempos 2. Utilizar un software de recuperación de archivos

Tabla 2.8: Riesgo 05: Perdida del TFG

ID	R006
Nombre	Falta de formación en Python
Descripción	Se corre el riesgo de que no cuente con suficiente formación de Python para realizar el TFG
Categoría	Personal
Vulnerabilidad	Puedo no poseer los suficientes conocimientos de Python como para completar con éxito el TFG
Amenaza	No realizar las partes practicas del TFG a tiempo
Probabilidad	Baja
Impacto	Alto
Prioridad	Media
Acciones de mitigación	1. Conseguir suficientes conocimientos de Python antes de la realización del TFG
Acciones de contingencia	1. Conseguir los conocimientos necesarios de Python para el TFG durante su edición 2. Replanificar el proyecto adecuándome a los nuevos tiempos

Tabla 2.9: Riesgo 06: Falta de formación

ID	R007
Nombre	Mala planificación
Descripción	Se corre el riesgo de que no planifique correctamente la realización del TFG
Categoría	Planificación y control
Vulnerabilidad	Una mala planificación para la realización de mi TFG puede llegar a tener un gran impacto
Amenaza	No tener el TFG a tiempo para la entrega
Probabilidad	Baja
Impacto	Alto
Prioridad	Media
Acciones de mitigación	1. Conseguir formación en planificación de proyectos 2. Corroborar que se tiene una buena planificación con mi tutora académica
Acciones de contingencia	1. Pedir ayuda a mi tutora y a otras personas que tengan conocimientos sobre planificación 2. Replanificar el proyecto adecuándome a los nuevos tiempos

Tabla 2.10: Riesgo 07: Mala planificación

ID	R008
Nombre	Caída del medio de comunicación
Descripción	Se corre el riesgo que el medio de comunicación con mi tutor académico no se encuentre disponible
Categoría	Comunicación
Vulnerabilidad	Para realizar el TFG necesitare usar el servicio de correo electrónico de la UVa, pudiendo, en caso de que se caiga, producir retrasos
Amenaza	No poder realizar la entrega del TFG a tiempo
Probabilidad	Media
Impacto	Bajo
Prioridad	Baja
Acciones de mitigación	1. Exploración de otras opciones de comunicación distintas al e-mail 2. Proposición y acuerdo con la tutora para llevar a cabo la comunicación por otras plataformas, como por ejemplo Rocket.Chat
Acciones de contingencia	1. Establecer el contacto con la tutora presencialmente en su despacho o al acabar ella alguna de sus clases 2. Establecer como canal de comunicación la llamada telefónica hasta que estén resueltos los problemas con los servidores de correo electrónico 3. Replanificar el proyecto adecuándome a los nuevos tiempos

Tabla 2.11: Riesgo 08: Caída del medio de comunicación

ID	R009
Nombre	Abandono de tutora
Descripción	Se corre el riesgo de que la tutora académica abandone el proyecto, dejándome sin tutorizar
Categoría	Personal
Vulnerabilidad	Para realizar el TFG necesitare un tutor, que en caso de que abandone el proyecto, se producirían retrasos en el mismo
Amenaza	No poder entregar el TFG
Probabilidad	Baja
Impacto	Alto
Prioridad	Media
Acciones de mitigación	1. Ponerse en contacto con más de un profesor de la facultad y procurar tener un profesor sustituto. 2. Asegurarse de que el profesor elegido tiene verdadero interés por el tema a realizar para que no abandone por desinterés.
Acciones de contingencia	1. Buscarse a otro tutor de TFG 2. Cambiar de tema de TFG y encontrar uno que tenga tutor 3. Replanificar el proyecto adecuándome a los nuevos tiempos

Tabla 2.12: Riesgo 09: Abandono de tutora

ID	R010
Nombre	Compatibilidad
Descripción	Se corre el riesgo de que el sistema final no sea compatible con la plataforma final en la que se despliegue
Categoría	Hardware
Vulnerabilidad	El sistema puede no funcionar correctamente en el servidor Linux
Amenaza	No poder entregar el producto final siguiendo lo planeado y tener que extender su desarrollo
Probabilidad	Baja
Impacto	Alto
Prioridad	Media
Acciones de mitigación	1. Desarrollar el sistema en una maquina con sistema operativo Linux 2. Probar los incrementos en el servidor final a medida que van saliendo
Acciones de contingencia	1. Emplear el siguiente sprint para arreglar dichas incompatibilidades

Tabla 2.13: Riesgo 10: Compatibilidad

ID	R011
Nombre	Cumplir requisitos
Descripción	Se corre el riesgo de que el sistema no cumpla los requisitos del cliente
Categoría	Planificación y control
Vulnerabilidad	El Development Team no tiene mucha experiencia, pudiendo hacer que no se lleguen a elicitar o cumplir todos los requisitos
Amenaza	No poder entregar el producto final siguiendo lo planeado y tener que extender su desarrollo
Probabilidad	Baja
Impacto	Alto
Prioridad	Media
Acciones de mitigación	1. Asegurarnos de que se han elicitado todos los requisitos correctamente en reuniones con el Product Owner
Acciones de contingencia	1. Reunirnos con el Product Owner para añadir a la lista de historias de usuario que faltan

Tabla 2.14: Riesgo 11: Cumplir Requisitos

ID	R012
Nombre	Respuestas de interesados
Descripción	Se corre el riesgo de que las partes interesadas tarden en responder a las dudas del Development Team
Categoría	Comunicación
Vulnerabilidad	Las partes interesadas del proyecto pueden tardar en responder a las dudas del Development Team, dados sus horarios
Amenaza	Tardar en introducir un cambio o función, pudiendo así no cumplir lo planeado
Probabilidad	Media
Impacto	Medio
Prioridad	Media
Acciones de mitigación	1. El Development Team debe realizar las preguntas con suficiente antelación para que no se produzca ningún retraso
Acciones de contingencia	1. Replanificar el proyecto adecuándome a los nuevos tiempos

Tabla 2.15: Riesgo 12: Respuestas de interesados

2.4. Estimación de presupuesto

En esta sección estimaremos los costes de hardware, software, de mano de obra del proyecto y de local. Estas estimaciones se harán tanto desde un punto de vista simulado; siendo este en el caso de que el proyecto lo realizase una empresa; como realista; teniendo en cuenta que en realidad este es un proyecto académico.

2.4.1. Estimación de costes simulada

Según la web *indeed.com* [11], el sueldo medio a fecha de 2 de marzo de 2023 de un desarrollador software en España es de 30.900 € al año, por lo tanto, teniendo en cuenta que el Convenio Colectivo Estatal de Empresas de Consultoría de Sistemas de Información [7] estipula que la jornada laboral de los desarrolladores software es 1800 horas anuales, serian unos 17,17 € la hora y teniendo en cuenta las horas estimadas (véase Sección 2.2) que vamos a dedicar al proyecto, el coste de personal para todo el proyecto sería de 17,17 €/hora * 350 horas = **6.009,50 €** en caso de no usar el sprint de refuerzo, pero en el caso de necesitarlo sería de 17,17 €/hora * 400 horas = **6.866 €**. Además, a lo anterior hay que añadirle el 30 % de costes que cotiza la empresa a la Seguridad Social, dando un total de **7.812,35 €**.

En cuanto a los costes de hardware, para el desarrollo del sistema empleamos un portátil Lenovo IdeaPad 3 15/TL6, valorado en 649 € y teniendo en cuenta el tiempo medio de vida útil es de 4 años, con lo cual si calculamos su coste al mes, serian de 649 € / 48 meses = 13,52 €/mes, dando lugar a un coste total durante el proyecto de 13,52 €/mes * 4 meses = **54,08 €**.

En cuanto a los costes de software, utilizamos los programas con las licencias económicas más asequibles. Para el desarrollo de los programas empleamos Visual Studio Code (véase Sección 4.1.1), cuya licencia es gratuita; para la creación de los diagramas utilizamos Astah Professional (véase Sección 4.1.3), cuya licencia cuesta 7,50 €/mes, dando un total de 7,50 €/mes * 4 meses = 30 €; para el dibujado de bocetos utilizamos Balsamiq (véase Sección 4.1.6), cuya licencia cuesta 129 € y para la comunicación con la tutora utilizamos la versión gratuita de Rocket.chat (véase Sección 4.2.3). Con todo esto, los costes software serian 30 € + 129 € = **159 €**.

En cuanto a los costes de local, tendríamos unos costes de 38,72 €/mes de luz; dando unos costes totales de 38,72 €/mes * 4 meses = 154,88 € de luz; y unos costes de 46 €/mes de Internet; dando unos costes totales de 46 €/mes * 4 meses = 184 €. Con todo esto, los costes de local serían 154,88 € + 184 € = **338,88 €**.

Por lo tanto el coste total asciende a 7.812,35 € + 54,08 € + 159 € + 338,88 € = **8.364,31 €**, sin embargo, para contemplar los posibles sobrecostes, añadimos un 20 % a los costes totales, dando un coste total de **10.037,17 €**. Podemos ver todo este desglose de costes de forma resumida en la Tabla 2.16.

Finalmente, debemos destacar que esta es una estimación interna de costes, y que este no se pretende utilizar como un presupuesto para el cliente.

Concepto	Cantidad	Precio	Total
Horas de trabajo de desarrollador software	350 horas	17,17 €/hora	6.009,50 €
Pago a la Seguridad Social	350 horas	5,15 €/hora	1.802,85 €
Licencia Astah Professional	4 meses	7,50 €/mes	30,00 €
Licencia Balsamiq	1 unidad	129 €/unidad	129,00 €
Lenovo IdeaPad 3 15/TL6	4 meses	13,52 €/mes	54,08 €
Luz	4 meses	38,72 €/mes	154,88 €
Internet	4 meses	46 €/mes	184,00 €
Total			**8.364,31 €**
Total +20 % para sobrecostes			**10.037.17 €**

Tabla 2.16: Estimación de costes simulado

2.4.2. Estimación de costes real

Siguiendo con la sección anterior, tanto los costes de horas de trabajo del desarrollador software como los costes de licencia de software, realmente no se tienen en cuenta, dado que el alumno no cuenta con ningún tipo de sueldo y todas las licencias de software o son gratuitas, o están pagadas por la universidad, con lo cual al final los costes económicos se reducen a los de el ordenador personal del alumno y de local. Podemos ver los costes reales en la Tabla 2.17.

Concepto	Cantidad	Precio	Total
Lenovo IdeaPad 3 15/TL6	4 meses	13,52 €/mes	54,08 €
Luz	4 meses	38,72 €/mes	154,88 €
Internet	4 meses	46 €/mes	184,00 €
Total			**392,96 €**
Total +20 % para sobrecostes			**471,55 €**

Tabla 2.17: Estimación de costes real

2.5. Product Backlog inicial

Como dijimos en la Sección 2.1.5, tanto el Product Backlog como el Sprint Backlog están compuestos por épicas e historias de usuario, siendo estas las análogas a los requisitos que se elicitan en los enfoques más tradicionales, solo que en este caso se escriben de una forma más informal siguiendo una estructura parecida a la siguiente: 'Como <*stakeholder*> quiero <*algo*> para <*propósito*>'.

Inicialmente, el Product Backlog esta compuesto por épicas para luego dividirlas en historias de usuario (véase Sección 2.1.5) ya que estas son más generales; lo cual viene bien para las fases iniciales del proyecto. En la Tabla 2.18 podemos ver el Product Backlog inicial.

ID	Épica
E01	Como usuario quiero almacenar las medidas de los radiómetros para mantenerlas seguras y visualizarlas posteriormente
E02	Como usuario quiero visualizar los datos almacenados en la base de datos para realizar un control de los mismos

Tabla 2.18: Product Backlog inicial

2.6. Product Backlog final

El Product Backlog ha ido evolucionando a lo largo del desarrollo del proyecto, desglosando las épicas mencionadas en la sección anterior en historias de usuario, desglosando las propias historias de usuario en otras nuevas o incluso creándolas a partir de las distintas reuniones del Scrum Team. Esta evolución la podemos ver en mayor detalle en la Sección 7.

En la Tabla 2.19 podemos ver el Product Backlog final, con el identificador, la descripción, el origen y el peso de cada historia de usuario. En cuanto al peso de las historias de usuario, debemos destacar que se han dado asumiendo que cada punto se corresponde a 5 horas de trabajo.

Como último aspecto a destacar, en la Tabla 2.19 podemos encontrarnos historias de usuario con cero puntos de historia asignados y sin ninguna hora invertida, las cuales se deberán asumir como restricciones o más bien requisitos no funcionales si pensamos en un enfoque más tradicional; que los programas deben cumplir.

ID	Historia de usuario	Origen	Puntos de historia	Horas Invertidas	Estado
HU01	Como Product Owner quiero que los archivos procesados sean guardados en otra carpeta para saber cuales archivos han sido procesados	E01	0		
HU02	Como Product Owner quiero que solo se procesen los archivos '.dat' para evitar errores humanos	E01	0		
HU03	Como Product Owner quiero visualizar las medidas de los radiómetros para realizar un control de las mismas	E02	4	5,5 horas	Completada

Continúa en la siguiente página ...

... viene de la página anterior

ID	Historia de usuario	Origen	Puntos de historia	Horas Invertidas	Estado
HU04	Como Product Owner quiero poder aplicar filtros a como se muestran las medidas para poder verlas según me resulte conveniente	E02	3	5 horas	Completada
HU05	Como Product Owner quiero visualizar la información de los sensores para poder controlarlos	E02	2	6 horas	Completada
HU06	Como Product Owner quiero poder añadir nuevos sensores para poder mantener el sistema actualizado	E02	2	5 horas	Completada
HU07	Como Product Owner quiero poder añadir nuevos factores de calibración para poder mantener el sistema actualizado	E02	2	5,5 horas	Completada
HU08	Como Product Owner quiero poder configurar los archivos y la base de datos sobre el que trabajar	E01 y E02	1	2 horas	Completada
HU09	Como Product Owner quiero que el sistema procese los datos en los archivos para almacenarlos en la base de datos	E01	3	14 horas	Completada
HU10	Como Product Owner quiero que el sistema funcione en un sistema Linux para poder alojarlo en un servidor Linux	E01 y E02	0		

Continúa en la siguiente página ...

... viene de la página anterior

ID	Historia de usuario	Origen	Puntos de historia	Horas Invertidas	Estado
HU11	Como Product Owner quiero poder llevar un seguimiento de las estaciones de los radiómetros para poder saber cuando y donde han estado instalados	E01	3		
HU12	Como Product Owner quiero que el sistema compruebe cada hora la carpeta de origen para mantener actualizada la base de datos	E01	0		
HU13	Como Development Team quiero que el sistema ignore las medidas cuyos sensores no existan para evitar errores	E01	0		
HU14	Como Development Team quiero que el sistema genere la estructura de la base de datos de forma automática para reducir el trabajo necesario para iniciar el sistema	HU09	2	7 horas	Completada
HU15	Como Development Team quiero que el sistema sea capaz de poblar la base de datos automáticamente para reducir la intervención del usuario en el proceso	HU09	2	8 horas	Completada
HU16	Como Development Team quiero guardar las distintas configuraciones del sistema en archivos YAML para facilitar su lectura por parte del mismo	HU08	0		

Continúa en la siguiente página ...

... viene de la página anterior

ID	Historia de usuario	Origen	Puntos de historia	Horas Invertidas	Estado
HU17	Como Development Team quiero mantener todas las acciones documentadas para llevar un seguimiento de las mismas	E01 y E02	0		
HU18	Como Product Owner quiero que el sistema se me entregue en forma de ejecutable para facilitar su uso	E01 y E02	0		
HU19	Como Development Team quiero que el sistema pueble la base de datos al crearla para poder procesar los datos de los archivos	HU09	1	1 hora	Completada
HU20	Como Product Owner quiero poder especificar con que carpetas trabaja el sistema para que este se adapte a mis necesidades	HU08	1	1 hora	Completada
HU21	Como Product Owner quiero que el sistema lea los archivos para guardar sus datos	HU09	1	1,5 horas	Completada
HU22	Como Product Owner quiero que el sistema genere logs mensuales para comprobar las acciones que ha ido realizando		2	4 horas	Completada
HU23	Como Development Team quiero que el sistema pueda procesar grandes cantidades de datos para evitar que este no funcione correctamente	HU09	2	2 horas	Completada

Continúa en la siguiente página ...

... viene de la página anterior

ID	Historia de usuario	Origen	Puntos de historia	Horas Invertidas	Estado
HU24	Como Development Team quiero que el sistema sea paralelizado para reducir los tiempos de ejecución		3	8 horas	Completada
HU25	Como Product Owner quiero poder especificar con que base de datos trabaja el sistema para que este se adapte a mis necesidades	HU08	1	1 hora	Completada
HU26	Como Development Team quiero que el sistema pueda realizar distintas consultas sobre la base de datos para poder acceder y guardar la información		0		
HU27	Como Development Team quiero que el código se mantenga lo más limpio y documentado posible para que sea claro y fácil de entender		0		Completada
HU28	Como Development Team quiero contar una interfaz de usuario principal para poder acceder al resto de funcionalidades del sistema		1	8 horas	Completada
HU29	Como Development Team quiero que el segundo programa siga un patrón MVC para poder separar sus distintas lógicas		0		Completada

Continúa en la siguiente página ...

... viene de la página anterior

ID	Historia de usuario	Origen	Puntos de historia	Horas Invertidas	Estado
HU30	Como Development Team quiero que el segundo programa muestre mensajes de error para informar al usuario		0		Completada
HU31	Como Development Team quiero que el sistema muestre las medidas en un gráfico de lineas para ver su evolución en el tiempo	HU03	3	5,5 horas	Completada
HU32	Como Development Team quiero que el sistema pueble dinámicamente los filtros para mantenerlos actualizados con todas las opciones disponibles	HU04	2	3 horas	Completada
HU33	Como Development Team quiero que se ofrezca un filtro con los identificadores de los sensores y otro con el tipo de medida para poder personalizar la visualización de las medidas	HU04	1	2 horas	Completada
HU34	Como Product Owner quiero poder añadir estaciones a la base de datos para poder mantener actualizada la base de datos		2	5,25 horas	Completada
HU35	Como Product Owner quiero poder editar las estaciones para poder mantenerlas actualizadas		2	5,25 horas	Completada

Continúa en la siguiente página ...

... viene de la página anterior

ID	Historia de usuario	Origen	Puntos de historia	Horas Invertidas	Estado
HU36	Como Product Owner quiero poder asignar sensores a las estaciones para poder mantener actualizada la base de datos		2	5 horas	Completada
HU37	Como Product Owner quiero poder filtrar las medidas según los sensores que hay en una estación en un cierto periodo para poder personalizar la visualización de las medidas		2	5 horas	Completada

Tabla 2.19: Product Backlog final

Capítulo 3

Análisis

En este capítulo, se muestran los resultados obtenidos durante la fase de análisis de los distintos sprints (véase Sección 2.2).

3.1. Modelo de dominio

Durante el sprint 0, se creó el modelo de dominio de los programas a partir de las historias de usuario especificadas en esta primera fase de planificación con Astah (véase Sección 4.1.3). Podemos ver este modelo de dominio en la Figura 3.1.

Pasamos a explicar cada una de las clases:

- **Rad_Radiometer:** esta es la clase central del modelo, la cual representa los distintos radiómetros que posee el GOA-UVa.
- **Rad_Calibration_Coef:** esta clase representa los distintos coeficientes de calibración asignados a cada sensor.
- **Rad_FileMeassured:** esta clase representa los distintos archivos en los que se han encontrado medidas de un sensor concreto y posteriormente han sido almacenadas.
- **Rad_Meassured:** esta clase representa las distintas medidas contenidas en un archivo.
- **Rad_Instalation:** esta clase representa la instalación virtual de un determinado sensor a una determinada estación.
- **Rad_Site:** esta clase representa las distintas estaciones del sistema.

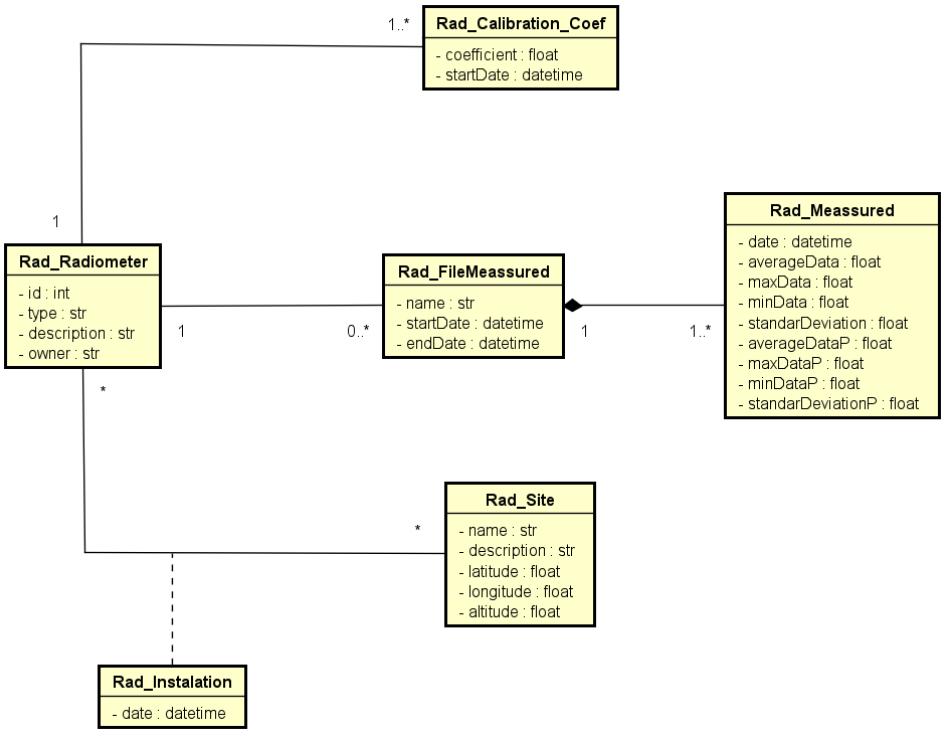

Figura 3.1: Modelo de dominio

3.2. Modelo de Casos de Uso

En este modelo de casos de uso podemos ver las distintas funcionalidades que se han ido presentando para GOAVisualRadiometer (véase Sección 6.1.3) durante el desarrollo de una forma abstracta, dado que GOASaveMeassure (véase Sección 6.1.2) es un programa que funciona en batch sin ninguna interacción con el usuario, y cuya ejecución se programa mediante el programador de tareas del sistema operativo. Podemos verlo en la Figura 3.2.

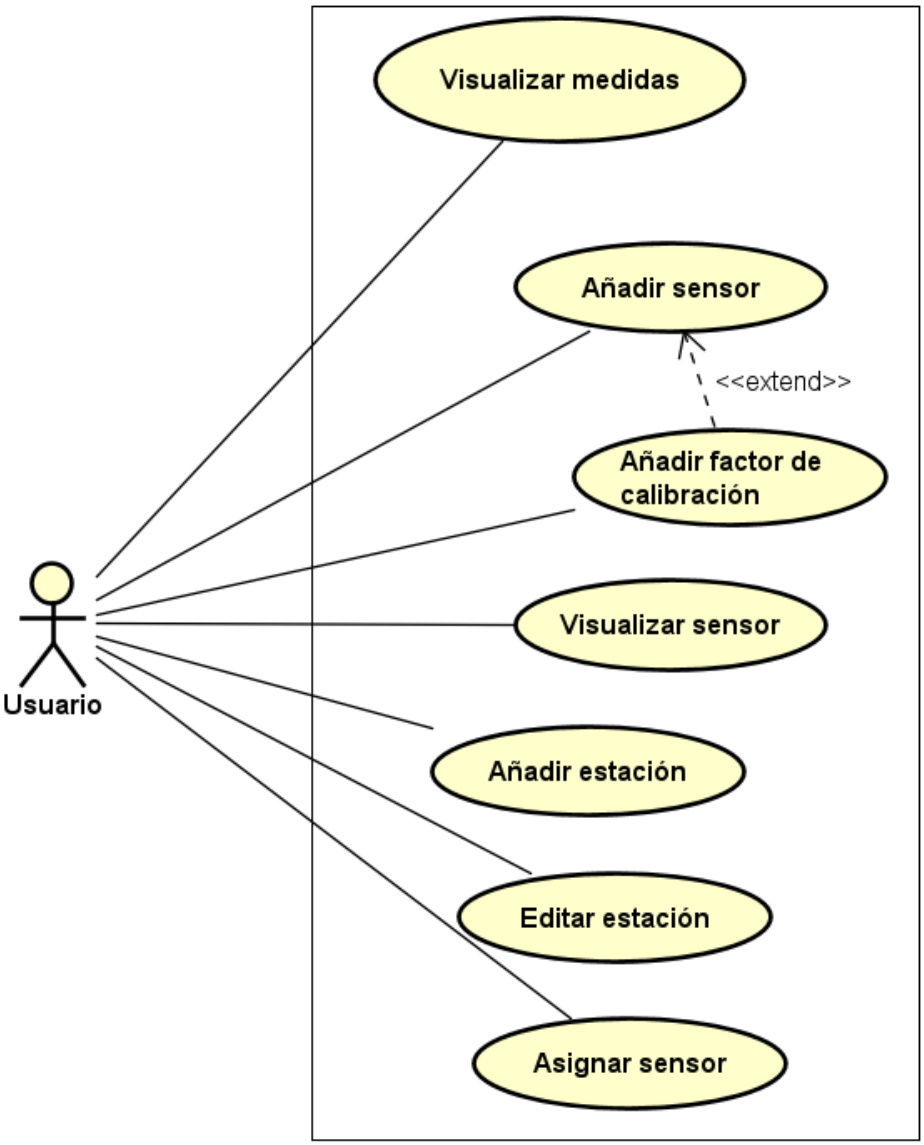

Figura 3.2: Modelo de Casos de Uso de GOAVisualRadiometer

Capítulo 4

Tecnologías utilizadas

4.1. Herramientas de desarrollo

En esta sección vamos a explicar brevemente las distintas herramientas utilizadas durante el desarrollo, tales como herramientas de modelado, bibliotecas y programas.

4.1.1. Visual Studio Code

Visual Studio Code [34] es un entorno de desarrollo creado por Microsoft para los sistemas operativos Windows, Linux y macOS, que permite programar en una amplia variedad de lenguajes como Python, Java, C etc.

Este editor ha sido empleado para crear el código de los programas, no solo por su comodidad, sino también por la capacidad de añadir extensiones al mismo con el fin de facilitar más el trabajo de desarrollo al usuario.

4.1.2. Python

Python [22] es un lenguaje de programación empleado para el desarrollo de software, el aprendizaje automático y la ciencia de datos, ampliamente utilizado por su facilidad de aprendizaje y comprensión de código, siendo este muy cercano al lenguaje humano; además de su simplicidad, ya que emplea menos líneas que otros lenguajes. Por otra parte, permite trabajar empleando distintos paradigmas, y cuenta con una comunidad y documentación muy amplia, teniendo así una ayuda accesible ante distintos problemas.

En nuestro caso, es el lenguaje en el que hemos desarrollado los programas, empleando los paradigmas de Programación Orientada a Objetos, de Programación Funcional y de Programación Orientada a Eventos.

4.1.3. Astah Professional

Astah [2] es una herramienta de modelado UML, utilizada en las actividades previas a la implementación del software para la creación de los distintos diagramas vistos en el Capítulo 3 y el Capítulo 5.

Se decidió utilizar esta herramienta, dado que la Escuela cuenta con un licencia para su uso y que el alumno ya contaba con experiencia previa en su uso, con lo cual esto reducía el tiempo de formación necesario.

4.1.4. Servidor MySQL

MySQL es un sistema gestor de bases de datos relacional. El grupo GOA-UVa cuenta con una instancia de MySQL desplegada en el servidor Linux del grupo. El programa desarrollado transforma y almacena los datos de los ficheros en una base de datos MySQL. Esta base de datos se emplea para realizar las consultas necesarias que permiten visualizar la información en la interfaz gráfica desarrollada.

A pesar de que el GOA-UVa cuenta con esta tecnología, durante la mayor parte del desarrollo del sistema, el alumno empleará una base de datos igual localizada en su propio portátil personal, reduciendo así las complicaciones que puede suponer emplear la del servidor, y a su vez, desarrollar el sistema teniendo ya en mente su despliegue final para producción.

4.1.5. MySQL Workbench

MySQL Workbench [16] es una herramienta que permite el diseño, modelado y administración de bases de datos relacionales MySQL de forma sencilla mediante una interfaz gráfica, que esta disponible para sistemas operativos Windows, Linux y Mac OS X.

Se ha utilizado en este proyecto, para facilitar al alumno la gestión de la base de datos del sistema, y así permitirle realizar distintas pruebas y consultas durante el desarrollo.

4.1.6. Balsamiq

Balsamiq [5] es un programa que permite realizar el bocetado de GUIs de forma rápida, sencilla e intuitiva, dando como resultado unos dibujos simples con los que transmitir la estructura y contenido de la interfaz olvidándonos de los aspectos más estéticos. Cuenta tanto con una versión de escritorio como una versión en la nube con una prueba gratuita de un mes, que posteriormente se puede ampliar pagando su licencia.

Se decidió emplear esta herramienta, dado que el alumno contaba ya con experiencia previa respecto a su uso, reduciendo así el tiempo necesario de formación, además de que la Escuela cuenta con una licencia educativa.

4.1.7. SQLAlchemy

SQLAlchemy [31] es una librería de Python que permite la conexión con bases de datos y la realización de consultas sobre la misma siguiendo un patrón ORM de tipo Data-Mapper (véase Sección 5.4.1).

Se decidió emplear esta librería a consejo del tutor del GOA-UVa, ya que este contaba con experiencia previa con ella, lo cual podía resultar útil en caso de que el alumno no consiguiese que funcionase.

4.1.8. PySide y Qt Designer

PySide [21] es una librería de Python que nos permite el diseño de interfaces gráficas de forma programática, actuando como un API que nos comunica con el toolkit de desarrollo de GUIs de Python: Qt, siendo este desarrollado por la Qt Company. En nuestro caso utilizaremos la versión PySide6.

A medida que vamos diseñando interfaces más complejas, puede llegar a ser muy tedioso el crearlas mediante programación, pero para estos casos contamos con Qt Designer [26], una herramienta gráfica que nos permite el diseño de interfaces en archivos de extensión '.ui', que luego podemos convertir a código Python mediante la herramienta de comando 'pyside6-uic' [27] para integrarlas en nuestros programas.

4.1.9. Matplotlib

Matplotlib [14] es una librería de Python que nos permite crear gráficos tanto animados como estáticos de forma sencilla y gratuita; abarcando muchos tipos de diagramas entre los que encontramos gráficos de líneas e histogramas, por ejemplo.

Esta librería la emplearemos en el sistema para la visualización de las medidas de los radiómetros en forma de gráficos de líneas, mostrando así su comportamiento a lo largo del tiempo.

4.1.10. Multiprocessing

Multiprocessing [24] es una librería de Python utilizada para paralelizar código mediante la creación de subprocesos, tal y como hace de forma similar la librería 'threading' [25] con la creación de hilos; tanto en sistemas Unix, como en sistemas Windows, permitiéndonos así aprovechar los procesadores de nuestra máquina.

En nuestro caso, la utilizamos para resolver un problema que surgió durante el desarrollo del primer programa del sistema (véase Sección 6.2.1).

4.1.11. PyInstaller

PyInstaller [20] es una aplicación que permite la creación de ejecutables a partir de programas Python, haciendo que estos puedan ser ejecutados sin tener instalados ni Python ni cualquiera de las librerías necesarias en nuestra máquina. Hay que destacar que, para generar un ejecutable que funcione en un sistema operativo concreto, este debe ser creado en una maquina con dicho sistema operativo, es decir que para generar los ejecutables para una maquina Linux, deben ser creados en una maquina Linux.

En nuestro caso, utilizamos la versión 5.9.0 para generar los ejecutables de ambos programas que conforman el sistema.

4.2. Herramientas de gestión de proyecto

4.2.1. GitLab

GitLab [9] es un repositorio de código en la nube para su almacenamiento, que permite un fácil control de versiones y que se ha empleado para organizar el desarrollo.

Dado que la escuela ya cuenta con un servidor GitLab, y que prácticamente todas las asignaturas que implican cualquier tipo de programación en la carrera cuentan con el uso de esta tecnología, se optó por emplearla también en este proyecto.

4.2.2. Overleaf

Overleaf [17] es un editor Latex online que permite modificar documentos de textos de manera colaborativa, además de emplear distintas plantillas según nuestras necesidades, permitiendo así la automatización de ciertas partes, como la escritura de los distintos índices.

Hemos empleado esta herramienta para editar esta memoria durante todo el desarrollo.

4.2.3. Rocket.chat

Rocket.chat [28] es una aplicación de comunicación semejante a Telegram, Discord ..., empleada para la comunicación entre la tutora académica y el alumno durante el desarrollo del TFG.

Capítulo 5

Diseño

En esta sección mostraremos los diagramas y bocetos creados durante la fase de diseño, obtenidos a partir del resultado de la fase de análisis durante el sprint 0 (véase Sección 2.2) y durante el desarrollo del TFG.

5.1. Diseño de interfaces de usuario

En esta sección, mostraremos los bocetos de las distintas vistas del segundo programa (desde la Figura 5.1 hasta la Figura 5.4), obtenidos durante el inicio del desarrollo del TFG. Estos bocetos han sido creados con la herramienta Balsamiq (véase Sección 4.1.6).

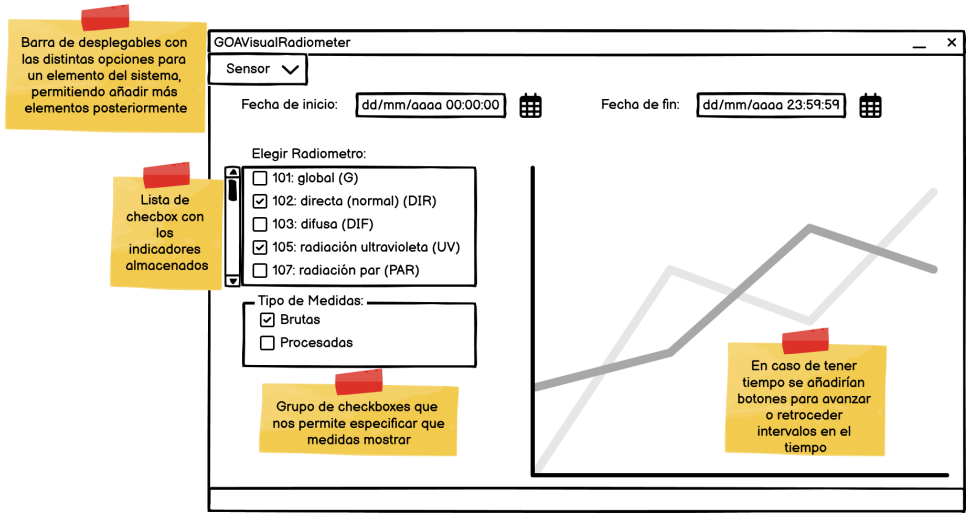

Figura 5.1: Pantalla para visualizar las medidas de los radiómetros

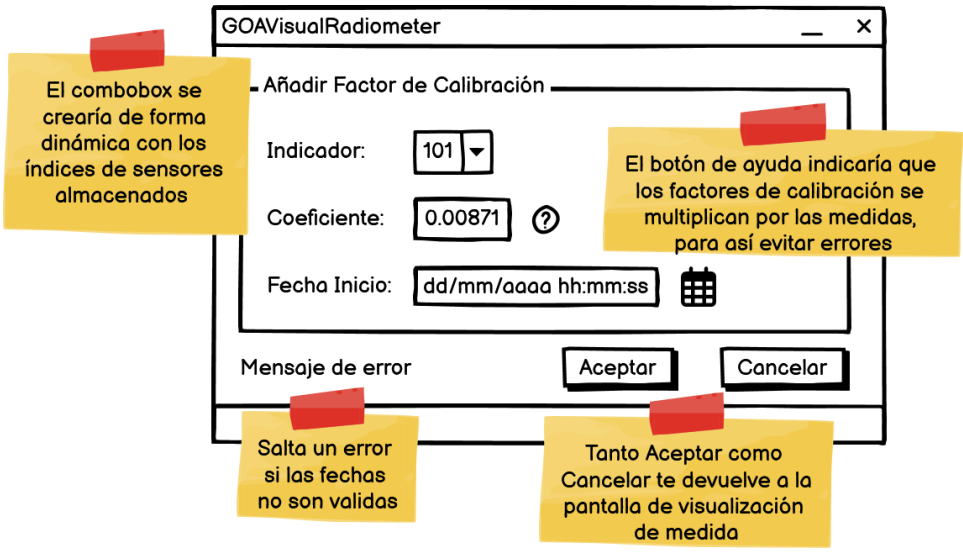

Figura 5.2: Pantalla para añadir un factor de calibración a un sensor

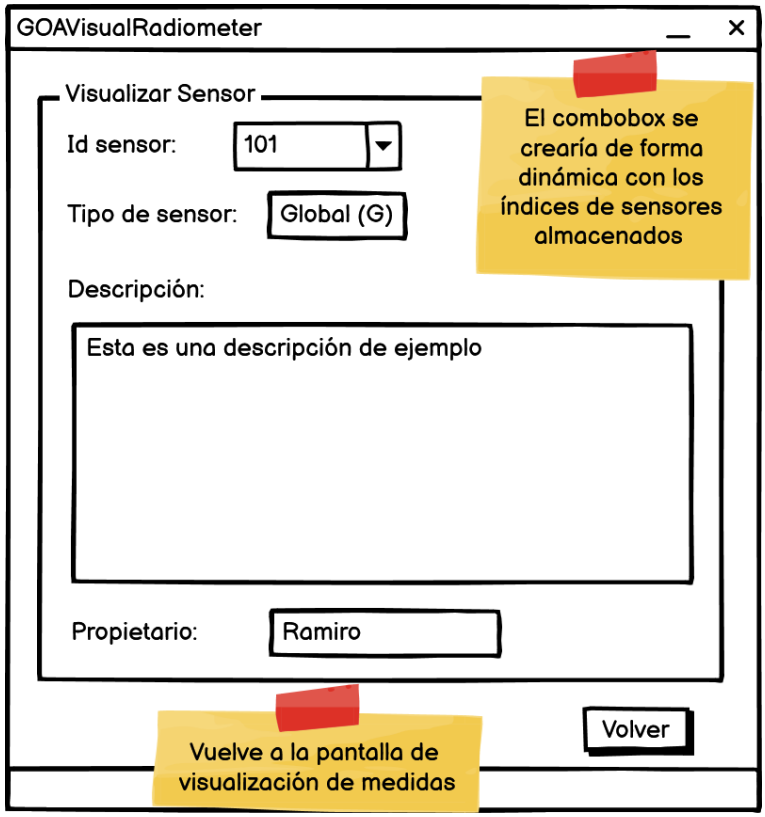

Figura 5.3: Pantalla para la visualización de información de un sensor

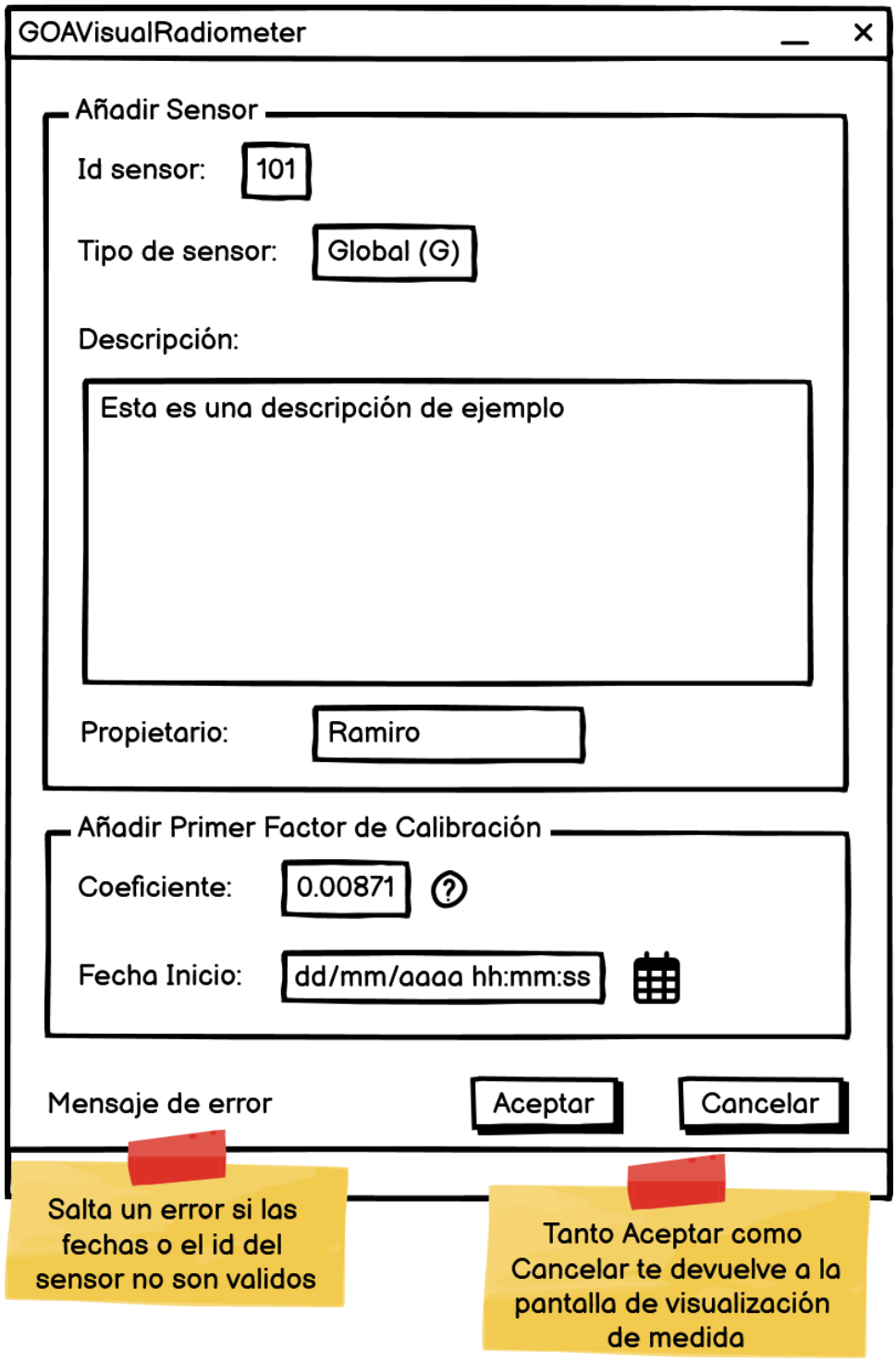

Figura 5.4: Pantalla para añadir un nuevo sensor al sistema

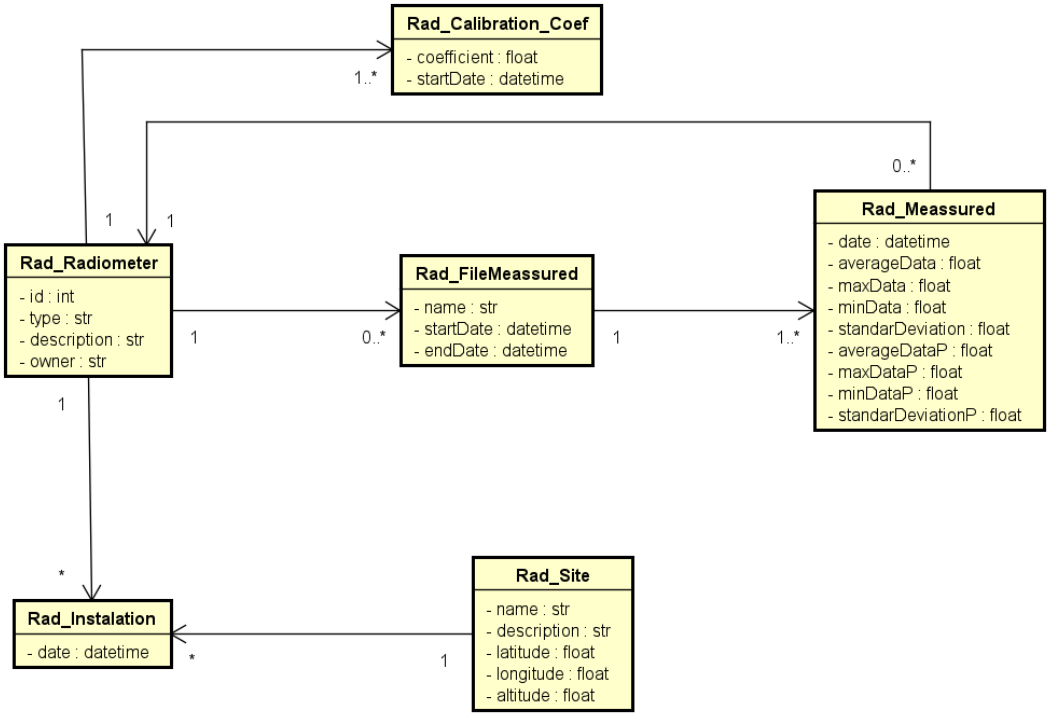

Figura 5.5: Diagrama de clases de diseño del modelo

5.2. Transformación del modelo de dominio en clases de diseño

Creamos el diagrama de clases de diseño a partir del modelo de análisis mostrado anteriormente (véase Sección 3.1) con Astah (véase Sección 4.1.3). Podemos ver el diagrama de clases de diseño en la Figura 5.5.

Este diagrama será el punto de partida del elemento modelo de la arquitectura para aplicar el patrón arquitectónico MVC como se explica posteriormente.

5.3. Diseño de la base de datos

A partir del diagrama de clases de la anterior sección, creamos con Astah (véase Sección 4.1.3) un diagrama con las distintas tablas que forman la base de datos. Podemos ver el diagrama de tablas de la base de datos en la Figura 5.6. Hay que destacar que, en las tablas donde hay más de un campo marcado como clave primaria, estas componen en su conjunto una única clave primaria y que en algunos casos no se ha empleado la típica clave primaria numérica autogenerada por solicitud del cliente.

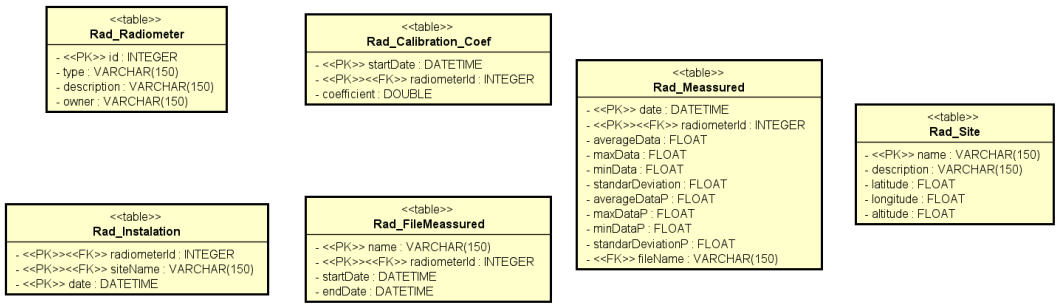

Figura 5.6: Tablas de la BD

5.4. Patrones de diseño empleados

5.4.1. Patrón ORM

El patrón Object-Relational Mapper (ORM) [1] permite conciliar los lenguajes de Programación Orientados a Objetos (POO) con las bases de datos relacionales sin tener que escribir consultas SQL, aunque aun así se va necesitar conocimientos sobre el mismo. Esto se consigue convirtiendo los objetos de los lenguajes orientados a objetos a las distintas tablas que conforman la base de datos, además de convertir las distintas operaciones de la base de datos a sentencias SQL, dando así una capa de abstracción de cara al programador.

Hay dos tipos de patrones ORM [18], los cuales se diferencian principalmente por donde se encuentra localizada la lógica de la base de datos:

- **Active Record:** en este tipo de patrón, la lógica SQL se encuentra dentro de la propia clase que representa la tabla de la base de datos, permitiendo realizar las distintas operaciones SQL como métodos de la propia clase. En muchos ORMs esto queda resuelto para las operaciones CRUD más generales heredando de una clase proporcionada por un framework o biblioteca que, aplicando metaprogramación, ofrece la solución predefinida.
- **Data-Mapper:** en este tipo de patrón, la lógica SQL se encuentra en una clase separada de la de la tabla, a la que comúnmente se denomina 'clase repositorio'.

En nuestro caso, utilizaremos el tipo Data-Mapper mediante la librería SQLAlchemy de Python [31].

5.5. Patrones arquitectónicos empleados

5.5.1. Patrón Modelo-Vista-Controlador (MVC)

El patrón MVC [15] [35], es un patrón arquitectónico comúnmente empleado para la implementación de GUIs, junto con los datos que maneja y su lógica de control, cuyo atractivo se basa en la separación de estos componentes con el objetivo de mejorar su mantenibilidad. A partir de este patrón, se crearon distintas variantes, como por ejemplo el MVVM (Modelo-Vista-Modelo de Vista), MVP (Modelo-Vista-Presentador) etc.

En nuestro caso, emplearemos la variante MVC Jerárquica, que consiste en que las distintas Vistas que conforman el sistema tienen una relación jerárquica entre ellas como, por ejemplo, una ventana principal a partir de la cual se puede acceder a distintas ventanas de diálogo.

Este patrón esta formado en general por tres componentes:

- **Modelo:** es la representación de los datos que utiliza la aplicación, controlando su acceso y modificación. Las peticiones de acceso y modificación, además del envío de información por parte del Modelo; se realizan a través del Controlador.

  Dentro de los Modelos, podemos encontrarnos dos variantes: pasivos o activos. Los Modelos activos son los que, en caso de ser modificados, notifican al resto de componentes; y los pasivos son los que, en caso de ser modificados, no notifican al resto de componentes. En nuestro caso, emplearemos un Modelo pasivo.

- **Vista:** define el cómo se muestran los datos del Modelo, permitiendo la interacción del usuario con los mismos.

- **Controlador:** es el componente que se encarga de modificar tanto la Vista como el Modelo según las distintas interacciones del usuario con la aplicación. Para cumplir su función, este actúa normalmente como un 'manejador de eventos', es decir, que esta compuesto por una serie de métodos, de los cuales cada uno se ejecuta en respuesta a una interacción específica del usuario con la aplicación.

Podemos ver el como interactúan los componentes entre ellos en la Figura 5.7.

Cabe destacar que en nuestro caso, algunos eventos llegarán directamente al Controlador sin pasar por la Vista, dado que en PySide (véase Sección 4.1.8) se cuenta con un sistema por el cual en la Vista se le puede asignar a un evento de un determinado elemento de la misma un método que funciona como un listener, denominado en este caso 'signals & slots'; haciendo que dicho evento de usuario no tenga que pasar por la Vista para llegar al controlador.

Una vez entendido el patrón MVC, podemos ver en la Figura 5.8 como se aplica en el segundo programa (véase Sección 6.1.3), repartiendo las distintas clases que lo conforman entre los distintos componentes.

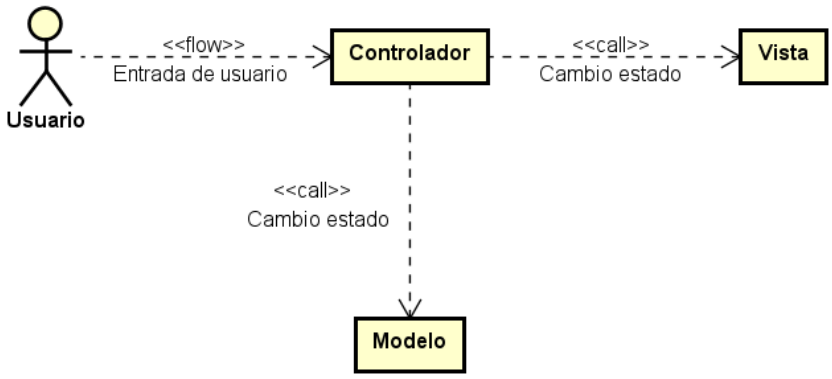

Figura 5.7: Relaciones entre los componentes de un MVC pasivo

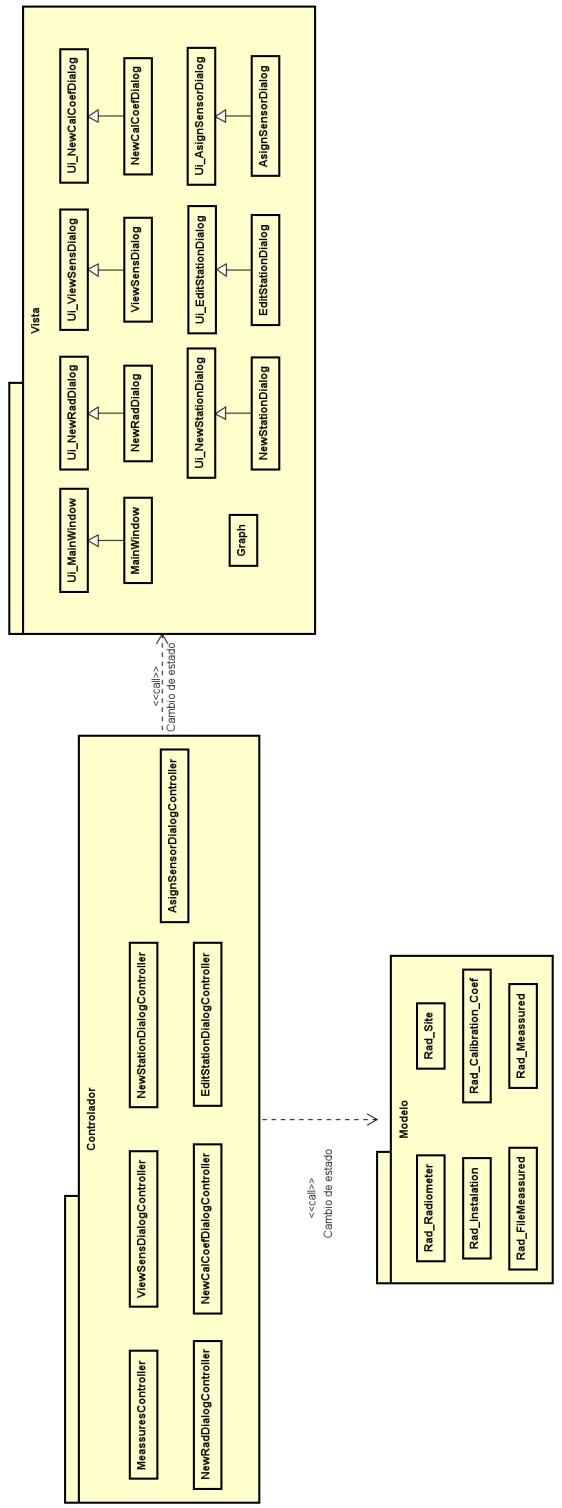

Figura 5.8: Estructura MVC de GOAVisualRadiometer

Como último detalle a destacar, el primer programa (véase Sección 6.1.2) no sigue este patrón dado que es fundamentalmente un script.

5.6. Diagrama de flujo de GOASaveMeassure

Para explicar de forma breve y gráfica el funcionamiento del programa GOASaveMeassure, podemos ver un diagrama de flujo en la Figura 5.9 que lo explica de forma sencilla.

5.7. Diseño del despliegue

En esta sección mostraremos cómo se desplegarán ambos programas una vez se hayan desplegado en el sistema final. Podemos ver el diagrama de despliegue en la Figura 5.10.

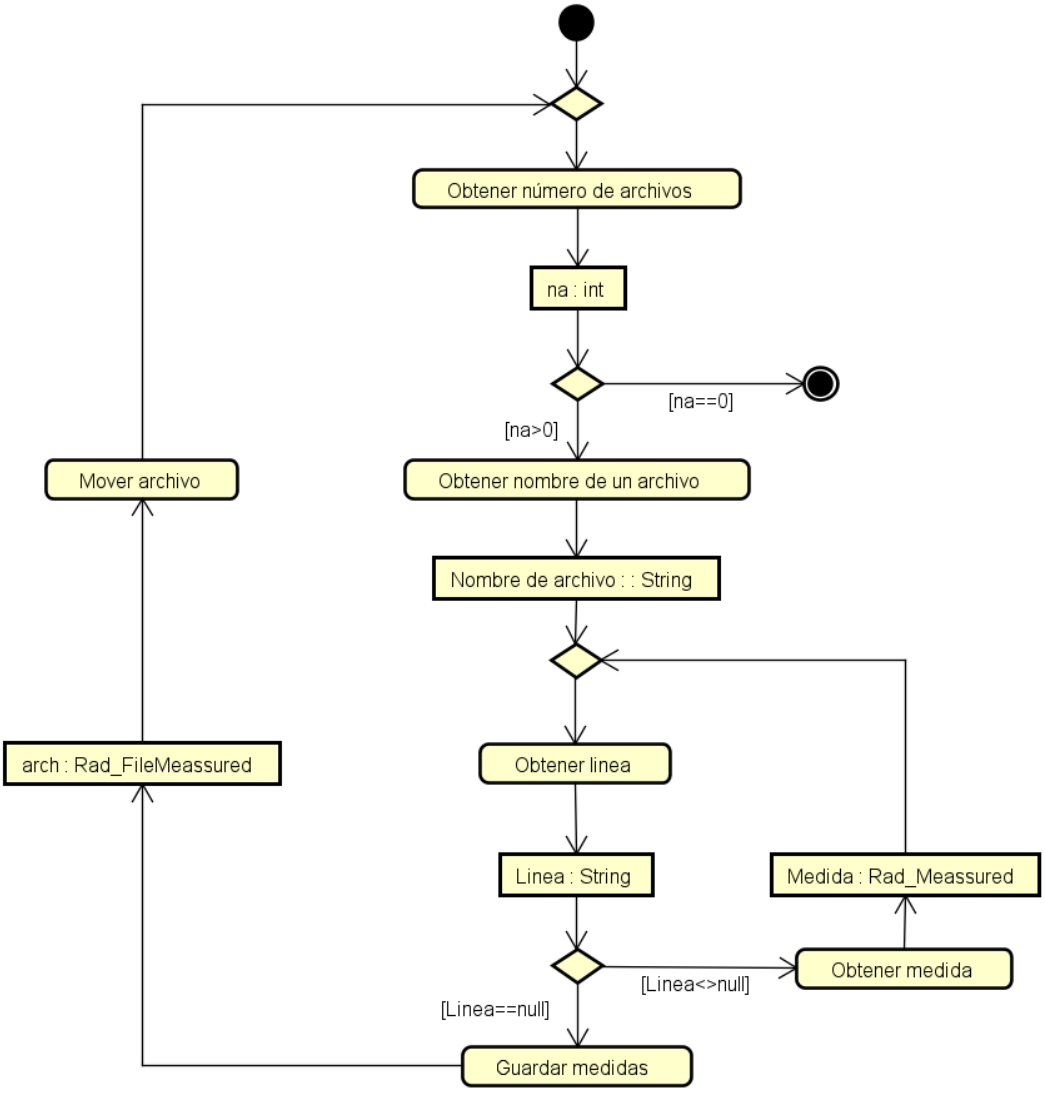

Figura 5.9: Diagrama de flujo de GOASaveMeassure

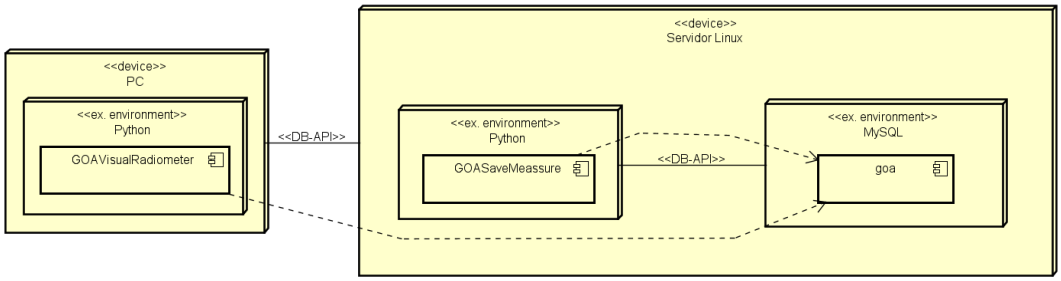

Figura 5.10: Diagrama de despliegue de los programas

Capítulo 6

Implementación y pruebas

6.1. Organización del código

En esta sección explicaremos la estructura del código en el repositorio de forma más o menos detallada. Para mejor organización, lo iremos explicando por carpetas.

6.1.1. Directorio raíz

Fijándonos en la Figura 6.1, podemos ver tres principales elementos:

- **Carpeta 'Pruebas':** esta carpeta contiene todos los archivos utilizados en las pruebas de las Sección 6.3, los cuales como podemos ver abarcan desde el archivo AP-01.dat, hasta el archivo AP-03.dat.
- **Archivo 'requirements.txt':** este archivo sirve para poder resolver las dependencias de los programas durante su mantenimiento (véase Sección A.2), descargando las distintas librerías Python que utilizan.
- **Carpeta 'src':** esta carpeta contiene el código fuente de los dos programas que conforman el sistema: 'GOASaveMeassure' y 'GOAVisualRadiometer'. Hablaremos en mayor detalle de su estructura en las Secciones 6.1.2 y 6.1.3 respectivamente.

6.1.2. GOASaveMeassure

GOASaveMeassure es un programa destinado al guardado de las medidas tomadas por los distintos radiómetros del GOA-UVa en una base de datos. Para mayor detalle, véase la Sección A.3.3.

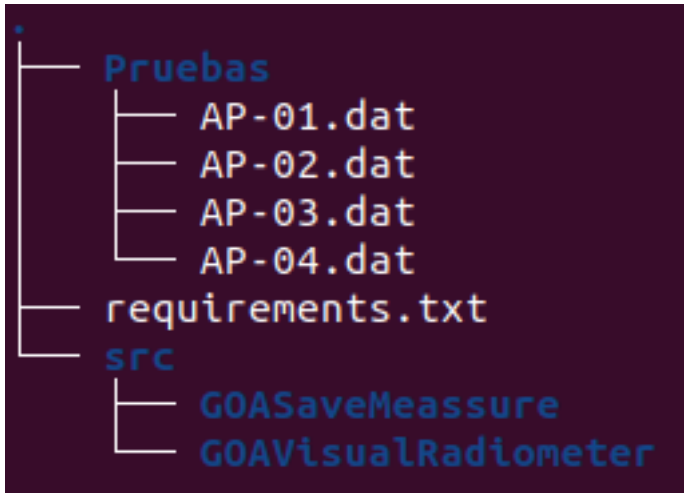

Figura 6.1: Estructura del directorio raíz

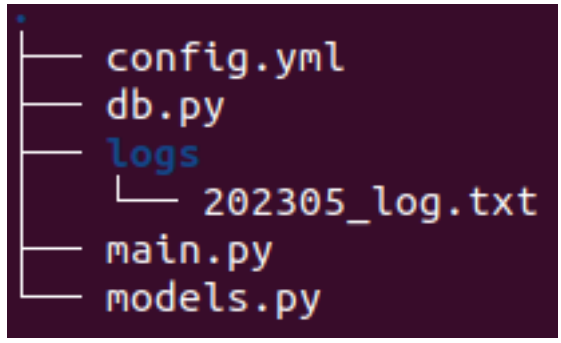

Figura 6.2: Estructura del programa GOASaveMeassure

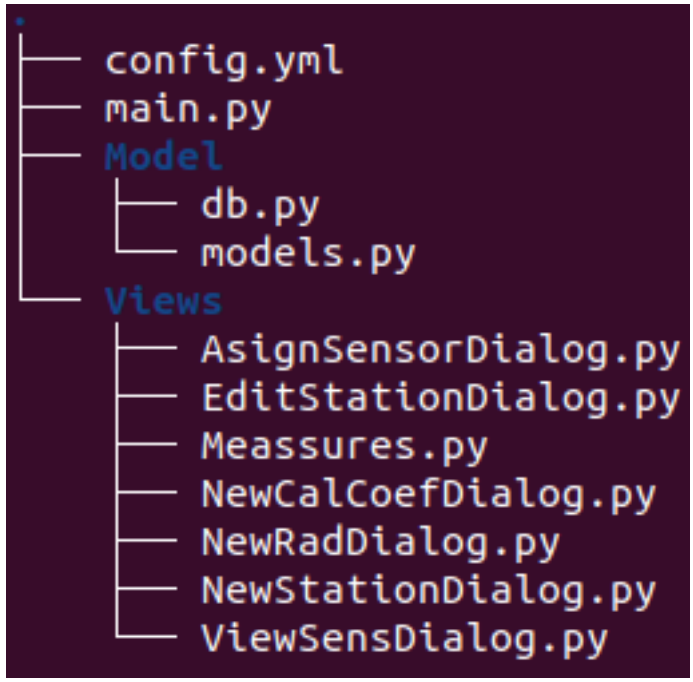

Figura 6.3: Estructura del programa GOAVisualRadiometer

Como podemos ver en la Figura 6.2, el programa se compone principalmente de cuatro archivos y además de un directorio que aparecerá tras el primer arranque. Pasamos a describirlos en mayor detalle:

- **Archivo 'config.yml':** este archivo permite configurar el programa indicándole la base de datos MySQL (véase Sección 4.1.4) y carpetas trabajar. Podemos ver un ejemplo de su estructura en la Figura A.1.
- **Archivo 'db.py':** este archivo se encarga de crear la conexión a la base de datos.
- **Carpeta 'logs':** esta carpeta es la que contendrá los archivos de log mensuales que se crean y editan al ejecutar el programa. Esta carpeta se crea tras la primera ejecución.
- **Archivo 'main.py':** este archivo contiene la lógica del programa y es el que se encarga de arrancarlo.
- **Archivo 'models.py':** este archivo contiene las clases del Modelo que se emplean para el patrón ORM (véase Sección 5.4.1).

6.1.3. GOAVisualRadiometer

GOAVisualRadiometer es un programa destinado a la visualización en gráficas de la evolución de las medidas almacenadas en la base de datos a lo largo del tiempo, además de

visualizar y editar otros elementos de la misma. Para mayor detalle, véase la Sección A.3.4.

Este programa está compuesto por dos archivos principales y dos directorios, cuya función es la de contener los archivos que conforman los componentes del patrón MVC Jerárquico (véase Sección 5.5.1). Teniendo en cuenta la Figura 6.3, vamos a describir cada uno de los componentes:

- **Archivo 'config.yml':** este archivo permite configurar el programa indicándole la base de datos MySQL con la que trabajar. Podemos ver un ejemplo de su estructura en la Figura A.2.
- **Archivo 'main.py':** este archivo se encarga de arrancar el programa.
- **Carpeta 'Model':** esta carpeta contiene los archivos que conforman el Modelo del patrón MVC (véase Sección 5.5.1). Contiene dos archivos:
  - **Archivo 'db.py':** este archivo se encarga de crear la conexión con la base de datos, además de contener las distintas operaciones que el Controlador puede hacer sobre la base de datos.
  - **Archivo 'models.py':** este archivo contiene las clases del Modelo que se emplean para el patrón ORM (véase Sección 5.4.1).
- **Carpeta 'Views':** esta carpeta contiene los archivos que conforman los Controladores y las Vistas del patrón MVC (véase Sección 5.5.1). Cada uno de los archivos contienen un par Vista-Controlador para cada pantalla del programa. Pasamos a indicar con que pantalla se corresponde cada archivo:
  - **Archivo 'AsignSensorDialog.py':** este archivo se corresponde con el diálogo 'Asignar sensor' (véase Figura A.11), que se encarga de la asignación de sensores a una estación.
  - **Archivo 'EditStationDialog.py':** este archivo se corresponde con el diálogo 'Editar estación' (véase Figura A.10), que se encarga de la edición de la información de una estación almacenada.
  - **Archivo 'Meassures.py':** este archivo se corresponde con la ventana 'Medidas' (véase Figura A.5), que se encarga de la visualización gráfica de las medidas y de proporcionar acceso al resto de funciones del programa.
  - **Archivo 'NewCalCoefDialog.py':** este archivo se corresponde con el diálogo 'Añadir factor de calibración' (véase Figura A.8), que se encarga de asignar un coeficiente de calibración a un sensor a partir de una fecha.
  - **Archivo 'NewRadDialog.py':** este archivo se corresponde con el diálogo 'Añadir Sensor' (véase Figura A.6), que se encarga de añadir nuevos sensores a la base de datos.
  - **Archivo 'NewStationDialog.py':** este archivo se corresponde con el diálogo 'Añadir estación' (véase Figura A.9), que se encarga de añadir una nueva estación a la base de datos.
  - **Archivo 'ViewSensDialog.py':** este archivo se corresponde con el diálogo 'Visualizar sensor' (véase Figura A.7), que se encarga de mostrar la información de los sensores almacenados en la base de datos.

6.2. Problemas

En esta sección mostraremos los distintos problemas que se presentaron durante la implementación del sistema.

6.2.1. Mejora de la eficiencia del programa GOASaveMeassure por paralelismo

Durante el sprint 1 (véase Sección 7.1.2), se detectaron unos tiempos de ejecución del primer programa (véase Sección 6.1.2) elevados, dado que el archivo de ejemplo que el Product Owner proporcionó al Development Team contaba con una media de dos millones de líneas que procesar, haciendo que el programa tardase más de tres horas en terminar de ejecutarse.

Ante este problema, durante el sprint 2 (véase Sección 7.1.3) el Development Team decidió paralelizar el código por medio de la librería multiprocessing (véase Sección 4.1.10), para reducir los tiempos de ejecución aprovechando los procesadores de la máquina que ejecute el programa. Sin embargo, dada la escasa formación del Development Team en esta materia, en la primera versión de la paralelización no se aprovecharon de forma correcta los procesadores, haciendo que a pesar de haberse reducido los tiempos de ejecución, al cabo de un rato la máquina se colgara y por lo tanto no se acabara de ejecutar el programa.

Tras estos problemas iniciales con la programación en paralelo, el Development Team decidió contactar con un profesor mejor formado en la materia para poder solucionarlo con su ayuda, además de que durante el Sprint Planning del sprint 2 (véase Sección 7.1.3), la Scrum Master y el Development Team se dieron cuenta de que el archivo proporcionado por el Product Owner podía no ser un buen ejemplo representativo de las entradas que iba a recibir el programa de su tiempo de vida, dado que este abarcaba entradas obtenidas durante un periodo de un año, haciendo que alcanzase el volumen de datos mencionado anteriormente.

Un vez confirmado con el Product Owner que la cantidad de lineas de los ficheros que recibiría el programa iba a ser considerablemente menor, y tras la reunión con el profesor, el Development Team consiguió paralelizar el programa de forma correcta, permitiendo así una reducción en los tiempos de ejecución. Para mayor precisión, se paralelizó la actividad 'Guardar medidas' de la Figura 5.9, dividiendo las medidas de cada sensor entre los distintos procesos para que las fuesen guardando.

En conclusión, este problema surgió a causa de una mala comunicación entre el Development Team y el Product Owner dado que, de haber sabido que el archivo proporcionado no representaba las entradas reales, no se habría ni planteado la idea de paralelizar el programa, aunque finalmente el haberlo hecho ha supuesto una mejora de los conocimientos del Development Team que pueden llegar a ser útiles en el futuro.

6.2.2. Adaptación del código de Qt Designer durante el desarrollo del programa GOAVisualRadiometer

Como dijimos en la Sección 4.1.8, para crear las interfaces de usuario del segundo programa (véase Sección 6.1.3) del sistema, utilizamos la librería PySide6 mediante el programa Qt Designer, que nos permite diseñar dichas interfaces sin pasar por el engorro de programarlas a mano.

Entrando en mayor profundidad en cuanto a cómo funciona Qt Designer [23], esta es una aplicación que, como hemos dicho, nos permite crear de forma intuitiva interfaces de usuario que luego se guardan en archivos con extensión '.ui', dejando al programador con dos opciones para poder usarlas: cargar directamente el archivo '.ui' desde nuestro código, o convertir el archivo '.ui' a código Python mediante la herramienta de comando 'pyside6-uic'. En nuestro caso, utilizaremos la segunda opción, ya que permite al Development Team modificar las interfaces y los distintos elementos que la conforman.

Con las interfaces ya creadas y elegido el método con el que utilizarlas, el Development Team se puso a trabajar con el resultado obtenido de dicha conversión a código Python, encontrándose con que este no funcionaba correctamente. Tras investigar este problema, el Development Team se dio cuenta que el comando 'pyside6-uic' no obtenía directamente la interfaz para ser utilizada, sino que lo que se generaba era una clase que 'construye' dicha interfaz, y que para emplearla hay que crear una nueva clase que herede tanto de la clase generada, como de la clase QMainWindow/QDialog dependiendo del tipo de ventana que estemos utilizando, para así acceder a los métodos de ambas. Sin embargo, esto no solucionó del todo el problema, ya que tras esto sí que se mostraban las interfaces en pantalla al arrancar el programa, pero aun así la adaptación de estas al modelo MVC (véase Sección 5.5.1) inicialmente no era del todo correcta, impidiéndonos así acceder a ciertas funcionalidades de la Vista por parte del Controlador, dado que este se asignaba a la Vista desde el constructor de la clase padre y no desde el constructor de la hija.

Como dijimos en los objetivos del proyecto (véase Sección 1.3), una de las metas al hacer este trabajo era que el Development Team adquiriese experiencia en el lenguaje Python, ya que este no contaba con muchos conocimientos previos. Pues bien, el error final al que tuvo que enfrentarse fue dado por esta falta de experiencia, en concreto, por como maneja Python las herencias múltiples en sus constructores.

De forma resumida, cuando en una clase que hereda de múltiples clases a la vez llamamos al constructor de sus clases padres, este sigue un orden en específico cuando accede a los atributos de las mismas, denominado Method Resolution Order o MRO [8], aunque esto se aplica también a otras propiedades como los métodos de dichas clases. Pues bien, en nuestro código llamábamos inicialmente a un constructor que abarcaba ambas clases padres, de tal forma que seguía el siguiente orden MRO: QMainWindow/QDialog y luego la clase generada a partir de archivo '.ui'; haciendo que no se pudiese asignar un Controlador a la Vista por dicho constructor, ya que este buscaba primero los atributos de la clase QMainWindow/QDialog.

Finalmente, tras encontrar una alternativa en el foro Stack Overflow [32], este problema fue resuelto invocando individualmente a cada uno de los constructores de cada una de las clases padres, pasándoles así sus atributos correspondientes, permitiéndonos así tener toda

la lógica de la Vista en la clase hija y haciéndola así accesible al Controlador.

6.3. Pruebas y resultados

En esta sección mostraremos en primer lugar las distintas pruebas diseñadas para aplicar al sistema en pos de comprobar su calidad y su correcto funcionamiento. Posteriormente se documentarán los resultados obtenidos de realizar los procesos de prueba utilizando las mismas.

6.3.1. Batería de pruebas

A partir de las distintas historias de usuario que hemos ido obteniendo a lo largo del desarrollo (véase Sección 2.6), hemos creado las siguientes pruebas para comprobar que estas se han completado correctamente y que por ende, el sistema presenta todas las funcionalidades deseadas. Podemos ver posteriormente los resultados de estas pruebas en la Sección 6.3.2. Los archivos mencionados en las celdas de entradas los podemos ver en la carpeta 'Pruebas' del repositorio del proyecto (véase Apéndice B)

HU08 - Configurar los archivos y la base de datos sobre la que trabajar (desde la Tabla 6.1 hasta la Tabla 6.2)

HU14 - Generar la estructura de la base de datos automáticamente (desde la Tabla 6.3 hasta la Tabla 6.4)

HU19 - Poblar la base de datos con las estructuras básicas al crearla (Tabla 6.5)

HU21 - Lectura de archivos (Tabla 6.6)

HU15 - Poblar la base de datos con la información proporcionada (desde la Tabla 6.7 hasta la Tabla 6.8)

HU23 - Procesar grandes cantidades de datos (Tabla 6.9)

HU22 - El sistema debe generar logs mensuales (desde la Tabla 6.10 hasta la Tabla 6.14)

HU24 - Paralelización del sistema (Tabla 6.15)

HU05 - Visualizar la información de los sensores (desde la Tabla 6.16 hasta la Tabla 6.17)

HU06 - Añadir nuevos sensores (desde la Tabla 6.18 hasta la Tabla 6.20)

HU07 - Añadir nuevos factores de calibración (desde la Tabla 6.21 hasta la Tabla 6.24)

HU28 - Interfaz principal para acceder a las distintas funcionalidades (Tabla 6.25)

HU32 - Poblar filtros de la pantalla de medidas (Tabla 6.26)

HU33 - Filtros de la pantalla de medidas (Tabla 6.27)

HU31 - Mostrar medidas en gráfico de líneas (desde la Tabla 6.28 hasta la Tabla 6.30)

HU34 - Añadir estaciones (desde la Tabla 6.31 hasta la Tabla 6.33)

HU35 - Editar estaciones (desde la Tabla 6.34 hasta la Tabla 6.35)

HU36 - Asignar sensor a estación (desde la Tabla 6.36 hasta la Tabla 6.37)

HU37 - Filtro de las estaciones en la pantalla de medidas (desde la Tabla 6.38 hasta la Tabla 6.39)

P-01	Configurar carpetas y base de datos
Descripción	Especificar en el archivo YAML las carpetas y la base de datos con la que queremos que trabaje nuestro programa
Programa y/o vista	GOASaveMeassure
Entrada	• username: jaime • password: 69-Dudes • host: localhost • port: 3306 • database: goa • origpath: /home/jaime/Escritorio/4º/2o cuatrimestre /Pract Empr/pruebas • destpath: /home/jaime/Escritorio/4º/2o cuatrimestre /Pract Empr/pruebas2
Salida esperada	El programa se conecta a la base de datos y encuentra las carpetas

Tabla 6.1: P-01: Configurar carpetas y base de datos

P-02	Configurar carpetas y base de datos erróneamente
Descripción	Especificar erróneamente en el archivo YAML la carpeta o la base de datos con la que queremos que trabaje nuestro programa
Programa y/o vista	GOASaveMeassure
Entrada	• username: jaime • password: 69-Dudes • host: hostautentico • port: 3306 • database: goa • origpath: /home/jaime/Escritorio/4º/2o cuatrimestre /Pract Empr/pruebas • destpath: /home/jaime/Escritorio/4º/2o cuatrimestre /Pract Empr/pruebas2
Salida esperada	El programa no continua su ejecución y refleja el error en el log

Tabla 6.2: P-02: Configurar las carpetas y la base de datos erróneamente

P-03	Crear base de datos automáticamente
Descripción	Iniciar el programa para que cree la base de datos en caso de que esta no exista
Programa y/o vista	GOASaveMeassure
Entrada	-
Salida esperada	Se ha creado la base de datos correctamente sin errores

Tabla 6.3: P-03: Crear base de datos automáticamente

P-04	No crear una base de datos existente
Descripción	Iniciar el programa para que compruebe que la base de datos necesaria ya existe y no hace falta crearla
Programa y/o vista	GOASaveMeassure
Entrada	-
Salida esperada	El programa ha continuado su ejecución sin crear la base de datos y usando la ya existente

Tabla 6.4: P-04: No crear una base de datos existente

P-05	Poblar la base de datos creada
Descripción	Cuando el programa crea la base de datos, este la puebla con unas entradas básicas para su correcto funcionamiento
Programa y/o vista	GOASaveMeassure
Entrada	-
Salida esperada	La base de datos ha sido creada y poblada correctamente

Tabla 6.5: P-05: Poblar la base de datos con las estructuras básicas al crearla

P-06	Lectura de archivos
Descripción	El programa debe leer correctamente el contenido de los archivos
Programa y/o vista	GOASaveMeassure
Entrada	Archivo AP-01
Salida esperada	Se leyó correctamente el archivo y su información se guardó correctamente en variables internas del programa

Tabla 6.6: P-06: Lectura de archivos

P-07	Guardado de información
Descripción	El programa guarda la información de los archivos en la base de datos
Programa y/o vista	GOASaveMeassure
Entrada	Archivo AP-01
Salida esperada	Se guarda correctamente la información del archivo en la base de datos

Tabla 6.7: P-07: Guardado de información

P-08	Procesado de datos
Descripción	Al almacenar la información, se procesan los datos multiplicando las medidas por el coeficiente vigente en su fecha de toma
Programa y/o vista	GOASaveMeassure
Entrada	Archivo AP-01
Salida esperada	Las medidas se han almacenado en la base de datos tras ser procesadas

Tabla 6.8: P-08: Procesado de datos

P-09	Procesado de mucha información
Descripción	El programa procesa archivos con gran cantidad de información
Programa y/o vista	GOASaveMeassure
Entrada	Archivo AP-02
Salida esperada	Se guarda correctamente la información del archivo en la base de datos y en un tiempo de ejecución inferior a 5 minutos

Tabla 6.9: P-09: Procesado de mucha información

P-10	Creación de carpeta de logs
Descripción	El programa crea una carpeta de logs si no existe
Programa y/o vista	GOASaveMeassure
Entrada	Archivo AP-01
Salida esperada	Se ha creado una carpeta de logs

Tabla 6.10: P-10: Creación de carpeta de logs

P-11	Creación de log mensual
Descripción	El programa crea un nuevo archivo de log mensual en caso de que no exista
Programa y/o vista	GOASaveMeassure
Entrada	Archivo AP-01
Salida esperada	Se ha creado el nuevo archivo de log de este mes

Tabla 6.11: P-11: Creación de log mensual

P-12	Generación de logs 1
Descripción	El programa procesa archivos con sensores ya existentes y sin entradas duplicadas
Programa y/o vista	GOASaveMeassure
Entrada	Archivo AP-03
Salida esperada	Se guarda en el log una entrada del archivo indicando que se han añadido todas las entradas de cada uno de los sensores

Tabla 6.12: P-12: Generación de logs 1

P-13	Generación de logs 2
Descripción	El programa procesa archivos con sensores ya existentes y no existentes y sin entradas duplicadas
Programa y/o vista	GOASaveMeassure
Entrada	Archivo AP-03
Salida esperada	Se guarda en el log una entrada del archivo indicando que se han añadido todas las entradas de cada uno de los sensores existentes y que se han denegado el guardado de las entradas de los sensores no reconocidos, además de cuantas entradas y que sensores eran

Tabla 6.13: P-13: Generación de logs 2

P-14	Generación de logs 3
Descripción	El programa procesa archivos con sensores ya existentes y no existentes y con entradas duplicadas
Programa y/o vista	GOASaveMeassure
Entrada	Archivo AP-01
Salida esperada	Se guarda en el log una entrada del archivo indicando que se han añadido todas las entradas de cada uno de los sensores existentes, además de indicar cuantas entradas duplicadas se han denegado de los mismos, y que se ha denegado el guardado de las entradas de los sensores no reconocidos, además de cuantas entradas y de que sensores eran

Tabla 6.14: P-14: Generación de logs 3

P-15	Comparativa de tiempos
Descripción	El programa debe ejecutarse en menor tiempo una vez paralelizado
Programa y/o vista	GOASaveMeassure
Entrada	Archivo AP-02
Salida esperada	Tiempos de ejecución significativamente inferiores respecto a la versión en serie del programa

Tabla 6.15: P-15: Comparativa de tiempos

P-16	ComboBox actualizado
Descripción	Al entrar en el diálogo, el ComboBox con los identificadores de los sensores debe de estar actualizado
Programa y/o vista	GOAVisualRadiometer ->ViewSensDialog
Entrada	-
Salida esperada	El ComboBox muestra como opciones todos los identificadores de los sensores almacenados

Tabla 6.16: P-16: ComboBox actualizado

P-17	Visualizar información del sensor
Descripción	Al elegir un sensor en el ComboBox, se muestra su información en los distintos campos
Programa y/o vista	GOAVisualRadiometer ->ViewSensDialog
Entrada	Id del sensor: 101
Salida esperada	El diálogo muestra la información del sensor

Tabla 6.17: P-17: Visualizar información del sensor

P-18	Formulario incompleto
Descripción	Aceptamos la creación de un sensor sin haber rellenado el formulario completo, dándonos un error
Programa y/o vista	GOAVisualRadiometer ->NewRadDialog
Entrada	•Id del sensor:110 •Tipo de sensor: Prueba •Descripción: - •Propietario: DevTeam •Coeficiente: 0,33 •Fecha de inicio: 2000/01/01 00:00:00
Salida esperada	El programa muestra un mensaje de error indicando que no se han rellenado todos los campos del formulario

Tabla 6.18: P-18: Formulario incompleto

P-19	Sensor existente
Descripción	Aceptamos la creación de un sensor cuyo identificador ya existe, dándonos un error
Programa y/o vista	GOAVisualRadiometer ->NewRadDialog
Entrada	•Id del sensor: 101 •Tipo de sensor: Prueba •Descripción: Descripción de prueba •Propietario: DevTeam •Coeficiente: 0,33 •Fecha de inicio: 2000/01/01 00:00:00
Salida esperada	El programa muestra un mensaje de error indicando que ya existe un sensor con ese identificador

Tabla 6.19: P-19: Sensor existente

P-20	Añadir sensor
Descripción	Creamos un nuevo sensor
Programa y/o vista	GOAVisualRadiometer ->NewRadDialog
Entrada	•Id del sensor:110 •Tipo de sensor: Prueba •Descripción: Descripción de prueba •Propietario: DevTeam •Coeficiente: 0,33 •Fecha de inicio: 2000/01/01 00:00:00
Salida esperada	Se ha añadido el nuevo sensor con su correspondiente primer factor de calibración a la base de datos

Tabla 6.20: P-20: Añadir sensor

P-21	ComboBox actualizado
Descripción	Al entrar en el diálogo, el ComboBox con los identificadores de los sensores debe de estar actualizado
Programa y/o vista	GOAVisualRadiometer ->NewCalCoefDialog
Entrada	-
Salida esperada	El ComboBox muestra como opciones todos los identificadores de los sensores almacenados

Tabla 6.21: P-21: ComboBox actualizado

P-22	Formulario incompleto
Descripción	Aceptamos la creación de un nuevo factor de calibración para un sensor sin haber rellenado el formulario completo, dándonos un error
Programa y/o vista	GOAVisualRadiometer ->NewCalCoefDialog
Entrada	•Sensor: 101 •Coeficiente: - •Fecha de inicio: 2024/01/01 00:00:00
Salida esperada	El programa muestra un mensaje de error indicando que no se han rellenado todos los campos del formulario

Tabla 6.22: P-22: Formulario incompleto

P-23	Factor de calibración ya existente
Descripción	Aceptamos la creación de un nuevo factor de calibración para un sensor que ya tiene un factor asignado a partir de la fecha proporcionada
Programa y/o vista	GOAVisualRadiometer ->NewCalCoefDialog
Entrada	•Sensor: 101 •Coeficiente: 0,33 •Fecha de inicio: 2000/01/01 00:00:00
Salida esperada	El programa muestra un mensaje de error indicando que la fecha ya esta asignada para ese sensor

Tabla 6.23: P-23: Factor de calibración ya existente

P-24	Añadir nuevo factor de calibración
Descripción	Aceptamos la creación de un nuevo factor de calibración para un sensor
Programa y/o vista	GOAVisualRadiometer ->NewCalCoefDialog
Entrada	•Sensor: 110 •Coeficiente: 0,33 •Fecha de inicio: 2024/01/01 00:00:00
Salida esperada	Se ha añadido el nuevo factor de calibración a la base de datos

Tabla 6.24: P-24: Factor de calibración ya existente

P-25	Abrir los diálogos
Descripción	Abrimos los distintos diálogos del programa desde la pantalla principal
Programa y/o vista	GOAVisualRadiometer ->MainWindow
Entrada	-
Salida esperada	Se abren los distintos diálogos sin errores

Tabla 6.25: P-25: Abrir los diálogos

P-26	Poblar filtros de medidas
Descripción	Al arrancar el segundo programa, los filtros de la pantalla de visualización de medidas se pueblan según el contenido de la base de datos
Programa y/o vista	GOAVisualRadiometer ->MainWindow
Entrada	-
Salida esperada	El filtro de 'Radiómetros:' y 'Estaciones' se ha poblado con una lista de CheckBoxes con los distintos sensores y estaciones almacenados en la base de datos

Tabla 6.26: P-26: Poblar filtros de medidas

P-27	Crear filtros
Descripción	Al arrancar el segundo programa, la pantalla de visualización de medidas debe ofrecer un filtro de periodo temporal en el que buscar medidas, uno para especificar las estaciones de las que ver los sensores en un periodo, uno para especificar las medidas de que sensor queremos visualizar y otro para especificar que tipo de medidas queremos ver
Programa y/o vista	GOAVisualRadiometer ->MainWindow
Entrada	-
Salida esperada	Se muestran los filtros al arrancar

Tabla 6.27: P-27: Crear filtros

P-28	Mostrar gráfica
Descripción	Al especificar los filtros correctamente, la pantalla muestra una gráfica con las medidas que lo cumplen
Programa y/o vista	GOAVisualRadiometer ->MainWindow
Entrada	•Fecha de inicio: 2017/11/23 00:00:00 •Fecha de fin: 2023/11/23 23:59:00 •Estación: Valladolid •Radiómetros: 101 global (G) •Tipo de datos: Brutos
Salida esperada	Se muestra la gráfica de las medidas

Tabla 6.28: P-28: Mostrar gráfica

P-29	Filtro de fechas erróneo
Descripción	Al especificar el filtro de 'Fecha de inicio' con una fecha mayor que el de 'Fecha de fin', el sistema no muestra la nueva gráfica
Programa y/o vista	GOAVisualRadiometer ->MainWindow
Entrada	•Fecha de inicio: 2017/11/24 00:00:00 •Fecha de fin: 2017/11/23 23:59:00 •Estación: Valladolid •Radiómetros: 101 global (G) •Tipo de datos: Brutos
Salida esperada	No se muestra una gráfica

Tabla 6.29: P-29: Filtro de fechas erróneo

P-30	Leyenda
Descripción	Al especificar los filtros, la gráfica muestra una leyenda con las distintas líneas
Programa y/o vista	GOAVisualRadiometer ->MainWindow
Entrada	•Fecha de inicio: 2017/11/23 00:00:00 •Fecha de fin: 2023/11/23 23:59:00 •Estación: Valladolid •Radiómetros: 101 global (G) •Tipo de datos: Brutos
Salida esperada	El sistema muestra la leyenda correctamente en la gráfica

Tabla 6.30: P-30: Leyenda

P-31	Añadir estación
Descripción	Al especificar todos los campos del diálogo y pulsar el botón 'Aceptar', se añade una nueva estación
Programa y/o vista	GOAVisualRadiometer ->NewStationDialog
Entrada	•Nombre: Ecuador •Descripción: Estación central •Latitud: 0 •Longitud: 0 •Altitud: 0
Salida esperada	Se ha añadido correctamente la nueva estación a la base de datos

Tabla 6.31: P-31: Añadir estación

P-32	Añadir estación vacía
Descripción	Al no especificar todos los campos del diálogo y pulsar el botón 'Aceptar', se muestra un mensaje de error
Programa y/o vista	GOAVisualRadiometer ->NewStationDialog
Entrada	•Nombre: Ecuador •Descripción: - •Latitud: 0 •Longitud: 0 •Altitud: 0
Salida esperada	El sistema muestra un mensaje de error, indicando los campos vacíos

Tabla 6.32: P-32: Añadir estación vacía

P-33	Añadir estación existente
Descripción	Al intentar añadir una estación con un nombre ya almacenado, el sistema muestra un mensaje de error
Programa y/o vista	GOAVisualRadiometer ->NewStationDialog
Entrada	•Nombre: Ecuador •Descripción: Estación central •Latitud: 0 •Longitud: 0 •Altitud: 0
Salida esperada	El sistema muestra un mensaje de error, indicando que ya existe una estación con ese nombre

Tabla 6.33: P-33: Añadir estación existente

P-34	Editar estación existente
Descripción	Editamos una estación y se guardan los cambios
Programa y/o vista	GOAVisualRadiometer ->EditStationDialog
Entrada	•Nombre: Ecuador •Descripción: Estación central •Latitud: 0 •Longitud: 1 •Altitud: 0
Salida esperada	El sistema guarda en la base de datos los cambios

Tabla 6.34: P-34: Editar estación existente

P-35	Editar estación con campos vacíos
Descripción	Intentamos editar una estación dejando campos vacíos, haciendo que se nos muestre un mensaje de error
Programa y/o vista	GOAVisualRadiometer ->EditStationDialog
Entrada	•Nombre: Ecuador •Descripción: - •Latitud: 0 •Longitud: 1 •Altitud: 0
Salida esperada	El sistema muestra un mensaje de error, indicando los campos vacíos

Tabla 6.35: P-35: Editar estación con campos vacíos

P-36	Asignar sensor a estación
Descripción	Asignamos un sensor a una estación a partir de una determinada fecha
Programa y/o vista	GOAVisualRadiometer ->AsignSensorDialog
Entrada	•Sensor:101 •Estación: Ecuador •Fecha de inicio: 2020/01/01 00:00:00
Salida esperada	El sistema añade la nueva instalación a la base de datos

Tabla 6.36: P-36: Asignar sensor a estación

P-37	Crear instalación existente
Descripción	Asignamos un sensor a una estación en una fecha ya asignada, dándonos un error
Programa y/o vista	GOAVisualRadiometer ->AsignSensorDialog
Entrada	•Sensor:101 •Estación: Ecuador •Fecha de inicio: 2000/01/01 00:00:00
Salida esperada	El sistema muestra un mensaje de error, indicando que ya existe una instalación con esa información

Tabla 6.37: P-37: Crear instalación existente

P-38	Poblar filtro de estaciones
Descripción	Al arrancar el segundo programa, el filtro de estaciones se puebla con todas las estaciones almacenadas
Programa y/o vista	GOAVisualRadiometer ->MainWindow
Entrada	-
Salida esperada	El filtro de 'Estaciones:' se ha poblado con una lista de CheckBoxes con las distintas estaciones almacenadas en la base de datos

Tabla 6.38: P-38: Poblar filtro de estaciones

P-39	Poblar filtro de sensores según los filtros de estaciones y de fechas
Descripción	Al seleccionar estaciones en su filtro y las fechas de inicio y de fin en los filtros de fechas, se muestran en el filtro de sensores los radiómetros que fueron instalados en esas estaciones en ese periodo
Programa y/o vista	GOAVisualRadiometer ->MainWindow
Entrada	•Estaciones: Valladolid •Fecha de inicio: 2017/11/23 00:00:00 •Fecha de fin: 2023/11/23 23:59:00
Salida esperada	El filtro de 'Radiómetros:' se ha poblado con una lista de CheckBoxes con los distintos sensores instalados en la estación de Valladolid durante el periodo especificado

Tabla 6.39: P-39: Poblar filtro de sensores según los filtros de estaciones y de fechas

6.3.2. Resultados obtenidos

En esta sección podemos ver los resultados obtenidos durante la ejecución de las pruebas de la Sección 6.3.1. Podemos ver dichos resultados en la Tabla 6.40. De esto podemos decir que los tests más efectivos han sido P-09 y P-15 pues han permitido descubrir errores. Se debe mencionar que, tal como se indica en dicha tabla, estos errores se solucionaron y la versión final entregada ya da OK como resultado en todas las pruebas

Prueba	Resultado	Solución
P-01	OK	-
P-02	OK	-
P-03	OK	-
P-04	OK	-
P-05	OK	-
P-06	OK	-
P-07	OK	-
P-08	OK	-
P-09	**Fallo:** el programa se colgaba y no conseguía terminar su ejecución	Se reestructuró la lógica de guardado de información en la base de datos del programa
P-10	OK	-
P-11	OK	-
P-12	OK	-
P-13	OK	-
P-14	OK	-
P-15	**Fallo:** el programa se colgaba y no conseguía terminar su ejecución	Se gestionaron mejor los procesadores, de forma que estos no recibiesen excesivo trabajo
P-16	OK	-
P-17	OK	-
P-18	OK	-
P-19	OK	-
P-20	OK	-
P-21	OK	-
P-22	OK	-
P-23	OK	-
P-24	OK	-
P-25	OK	-
P-26	OK	-
P-27	OK	-
P-28	OK	-
P-29	OK	-
P-30	OK	-
P-31	OK	-
P-32	OK	-
P-33	OK	-
P-34	OK	-
P-35	OK	-
P-36	OK	-
P-37	OK	-
P-38	OK	-
P-39	OK	-

Tabla 6.40: Resultados obtenidos de las pruebas

Capítulo 7

Seguimiento del proyecto

7.1. Seguimiento de los sprints realizados

En esta sección mostraremos las acciones tomadas y los eventos acontecidos durante cada uno de los sprints calendarizados (véase Sección 2.2).

7.1.1. Sprint 0 (27/02/2023 - 13/03/2023)

Este sprint lo empleamos para realizar la planificación inicial y la formación del alumno, además de empezar las fases de análisis y diseño iniciales del desarrollo, siendo ambas documentadas en la memoria. En total se dedicaron 50 horas a este sprint.

A continuación, desglosamos el sprint en sus distintas fases:

- **Sprint Planning (27/02/2023):** en esta reunión entre el Development Team y la Scrum Master, se decidió qué modelo de proceso emplear para el desarrollo, los medios de comunicación entre ellos (véase Sección 4.2.3), la estructura de esta memoria y además que se iba a empezar a crear los primeros diagramas de la fase de análisis y diseño.
- **Semana 1:** en esta primera mitad se empezaron a crear los diagramas de clase de análisis, de diseño y de las tablas para la base de datos, además de también los bocetos iniciales de la GUI. En cuanto a la memoria, se comenzó a editar los Capítulos 2 y 4.
- **Weekly Scrum (06/03/2023):** en esta reunión el Development Team y la Scrum Master revisaron el progreso realizado en la primera semana, además de decidir las tareas a realizar en la siguiente semana. Para esta siguiente mitad del sprint, se decide definir el presupuesto inicial, desglosar las épicas de la anterior semana en historias de usuario y formar al Development Team en la librería de SQLAlchemy (véase Sección

4.1.7), además de modificar los diagramas y bocetos de la primera mitad con el feedback recibido del Product Owner.

- **Semana 2:** como se dijo en la Weekly Scrum, se realizó la sección de estimación de costes (véase Sección 2.4) y se desglosaron las épicas de la mitad anterior (Tabla 2.18) en nuevas historias de usuario. Podemos ver dichas historias en la Tabla 7.1, donde cabe destacar que las historias de usuario con 0 puntos de historia asignados vendrían a ser las equivalentes a los requisitos no funcionales de las metodologías más tradicionales. Además, se empezó a escribir el contexto del TFG (véase Sección 1.1) y la explicación del patrón ORM (véase Sección 5.4.1).

ID	Historia de usuario	Origen	Puntos de historia
HU01	Como Product Owner quiero que los archivos procesados sean guardados en otra carpeta para saber cuales archivos han sido procesados	E01	0
HU02	Como Product Owner quiero que solo se procesen los archivos ‘‘`.dat`’’ para evitar errores humanos	E01	0
HU03	Como Product Owner quiero visualizar las medidas de los radiómetros para realizar un control de las mismas	E02	4
HU04	Como Product Owner quiero poder aplicar filtros a como se muestran las medidas para poder verlas según me resulte conveniente	E02	3
HU05	Como Product Owner quiero visualizar la información de los sensores para poder controlarlos	E02	4
HU06	Como Product Owner quiero poder añadir nuevos sensores para poder mantener el sistema actualizado	E02	4
HU07	Como Product Owner quiero poder añadir nuevos factores de calibración para poder mantener el sistema actualizado	E02	4
HU08	Como Product Owner quiero poder configurar los archivos y el servidor sobre el que trabajar	E01 y E02	1

Continúa en la siguiente página ...

... viene de la página anterior

ID	Historia de usuario	Origen	Puntos de historia
HU09	Como Product Owner quiero que el sistema procese los datos en los archivos para almacenarlos en la base de datos	E01	3
HU10	Como Product Owner quiero que el sistema funcione en un sistema Linux para poder alojarlo en un servidor Linux	E01 y E02	0
HU11	Como Product Owner quiero poder llevar un seguimiento de las estaciones de los radiómetros para poder saber cuando y donde han estado instalados	E01	3
HU12	Como Product Owner quiero que el sistema compruebe cada hora la carpeta de origen para mantener actualizada la base de datos	E01	0
HU13	Como Development Team quiero que el sistema ignore las medidas cuyos sensores no existan para evitar errores	E01	0

Tabla 7.1: Historias de usuario iniciales

Por otra parte, el Development Team y el Product Owner tuvieron una reunión para poder mejorar los distintos diagramas y bocetos; y también para formar al Development Team sobre SQLAlchemy (véase Sección 4.1.7) con el equipo del GOA-UVa.

- **Sprint Review/Restrospective (13/03/2023):** durante esta reunión, el Devlopment Team y la Scrum Master se reunieron para revisar la lista final de historias de usuarios obtenida en la semana 2 de este sprint y los presupuestos, llegando a consensuar ciertos cambios necesarios que aplicar a ambos antes de pasar al siguiente sprint.

7.1.2. Sprint 1 (13/03/2023 - 27/03/2023)

Este sprint lo emplearemos para empezar a programar las bases del sistema, además de seguir con las fases de análisis y diseño.

- **Sprint Planning (13/03/2023):** en esta reunión entre el Development Team y la Scrum Master, se decidió con cuales historias de usuario empezar a desarrollar el proyecto. Podemos verlos en la Tabla 7.2.
- **Semana 1:** durante esta primera semana, el Development Team comenzó con la programación del sistema y la ampliación de la lista de historias de usuario a medida que se iba avanzando en el desarrollo.

ID	Historia de usuario	Origen	Puntos de historia	Horas invertidas	Estado
HU08	Como Product Owner quiero poder configurar los archivos y la base de datos sobre el que trabajar	E01 y E02	1	2 horas	Completada
HU09	Como Product Owner quiero que el sistema procese los datos en los archivos para almacenarlos en la base de datos	E01	3	14 horas	Completada

Tabla 7.2: Historias de usuario del sprint 1

En cuanto al desarrollo, se crearon los métodos para el procesado de las líneas de los ficheros y la lógica de creación y estructuración de la base de datos, completando con ello las historias de usuario HU14, HU15 y HU19.

En cuanto a las historias de usuario, podemos ver las que se crearon esta semana en la Tabla 7.3, siendo en este caso de la HU14 a la HU19.

ID	Historia de usuario	Origen	Puntos de historia	Horas invertidas	Estado
HU14	Como Development Team quiero que el sistema genere la estructura de la base de datos de forma automática para reducir el trabajo necesario para iniciar el sistema.	HU09	2	7 horas	Completada
HU15	Como Development Team quiero que el sistema sea capaz de poblar la base de datos automáticamente para reducir la intervención del usuario en el proceso	HU09	2	8 horas	Completada

Continúa en la siguiente página ...

... viene de la página anterior

ID	Historia de usuario	Origen	Puntos de historia	Horas invertidas	Estado
HU16	Como Development Team quiero guardar las distintas configuraciones del sistema en archivos YAML para facilitar su lectura por parte del mismo	HU08	0		
HU17	Como Development Team quiero mantener todas las acciones documentadas para llevar un seguimiento de las mismas	E01 y E02	0		
HU18	Como Product Owner quiero que el sistema se entregue en forma de ejecutable para facilitar su uso	E01 y E02	0		
HU19	Como Development Team quiero que el sistema pueble la base de datos al crearla para poder procesar los datos de los archivos	HU09	1	1 hora	Completada
HU20	Como Product Owner quiero poder especificar con que carpetas trabaja el sistema para que este se adapte a mis necesidades	HU08	1	1 hora	Completada
HU21	Como Product Owner quiero que el sistema lea los archivos para que guarde sus datos	HU09	1	1,5 horas	Completada

Continúa en la siguiente página ...

. . . viene de la página anterior

ID	Historia de usuario	Origen	Puntos de historia	Horas invertidas	Estado
HU22	Como Product Owner quiero que el sistema genere logs mensuales para comprobar las acciones que ha ido realizando		2		Sin iniciar
HU23	Como Development Team quiero que el sistema pueda procesar grandes cantidades de datos para evitar que este no funcione correctamente	HU09	2	2 horas	Completada
HU24	Como Development Team quiero que el sistema sea paralelizado para reducir los tiempos de ejecución		3		Sin iniciar
HU25	Como Product Owner quiero poder especificar con que base de datos trabaja el sistema para que este se adapte a mis necesidades	HU08	1	1 hora	Completada

Tabla 7.3: Historias de usuario añadidas al backlog durante el sprint 1

- **Weekly Scrum (20/03/2023):** en esta reunión entre el Development Team y la Scrum Master, se mostraron los avances realizados durante la primera semana por parte del Development Team, además de precisar definitivamente los diagramas iniciales de análisis, diseño y de la base de datos (véase secciones 3.1, 5.2 y 5.3) y los pasos a seguir en la siguiente semana.

- **Semana 2:** durante esta semana, el Development Team se dedicó a la programación de la lectura de ficheros para guardar sus datos y de la función para mover los archivos de una carpeta a otra una vez han sido procesados, con todo esto cubriendo la historias de usuario HU20, HU21, HU23 y HU25. Además, se fue a su vez escribiendo partes de la memoria, como por ejemplo, las nuevas historias de usuario desglosadas durante esta semana que podemos ver en la Tabla 7.3, siendo estas las historias a partir de la entrada HU20 de la misma.

ID	Historia de usuario	Origen	Puntos de historia	Horas Invertidas	Estado
HU22	Como Product Owner quiero que el sistema genere logs mensuales para comprobar las acciones que ha ido realizando		2	4 horas	Completada
HU24	Como Development Team quiero que el sistema sea paralelizado para reducir los tiempos de ejecución		3	8 horas	Completada

Tabla 7.4: Historias de usuario del sprint 2

- **Sprint Review/Restrospective (27/03/2023):** durante esta reunión entre el Development Team y la Scrum Master, se evaluó el avance de las historias de usuario de este sprint, dando como resultado lo reflejado en la Tabla 7.2; además de comentar las dificultades a las que se había enfrentado el Development Team durante el desarrollo, siendo la más importante los largos tiempos de ejecución del sistema actual que llegan a superar las tres horas, ya que como vimos en la historia de usuario HU12 (véase Tabla 7.1), el sistema se ejecutará cada hora, con lo cual se debe acercar lo máximo posible los tiempos de ejecución a esa marca. Esta necesidad la podemos ver reflejada en la historia de usuario HU24 de la Tabla 7.3.

7.1.3. Sprint 2 (27/02/2023 - 10/04/2023)

Este sprint se dedicará a la paralelización del sistema del anterior sprint, reduciendo así su tiempo de ejecución, además de intentar que el sistema genere un log durante su ejecución. Como aspecto a destacar, este sprint se desarrollará durante las vacaciones de Semana Santa, con lo cual el Weekly Scrum se hará a la vez que el Sprint Review/Retrospective y la planificación se realizará teniendo en cuenta esta circunstancia.

- **Sprint Planning (27/02/2023):** durante esta reunión, el Development Team y la Scrum Master decidieron cuales iban a ser las historias de usuario a completar durante el sprint. Podemos verlas en la Tabla 7.4.
- **Semana 1:** durante esta primera semana, el Development Team se dedicó a formarse respecto a cómo paralelizar el código actual, ya que este no contaba con una formación sólida al respecto. Tras esto, el Development Team se dedicó a la optimización del programa y, una vez hecho esto, a crear la lógica de generación de archivos de log mensuales, completando así las dos historias de usuario asignadas a este sprint.

  Dado que dichas historias de usuario se completaron antes de lo planificado, se decidió dedicar la segunda mitad del sprint a la formación del Development Team de cara a

ID	Historia de usuario	Origen	Puntos de historia	Horas Invertidas	Estado
HU26	Como Development Team quiero que el sistema pueda realizar distintas consultas sobre la base de datos para poder acceder y guardar la información		0		
HU27	Como Development Team quiero que el código se mantenga lo más limpio y documentado posible para que este sea claro y fácil de entender		0		En proceso

Tabla 7.5: Historias de usuario añadidas al backlog durante el sprint 2

futuros sprints y a escribir secciones de la memoria.

- **Semana 2:** durante esta mitad del sprint, el Development Team se dedicó a su formación en la librería PySide y la herramienta Qt Designer (véase Sección 4.1.8), para el desarrollo de interfaces de usuario; además de la librería MatPlotLib para la representación de datos en gráficas. Para esta formación se dedicaron ocho horas de trabajo.

  A parte de esto, como se dijo en la semana anterior, se dedicó tiempo a la documentación del TFG, además de generar dos nuevas historias de usuario que podemos ver en la Tabla 7.5. Debemos recordar que las historias de usuario con cero puntos de historia se consideran restricciones y por lo tanto no se le invierte horas como tal, dado que estas se abordan a lo largo del resto de historias de usuario.

- **Sprint Review/Retrospective/Weekly Scrum (10/04/2023):** durante esta reunión entre el Development Team y la Scrum Master, se revisaron las historias de usuario del sprint, con especial énfasis en cuanto a la precisión con la que se había predicho la duración de las mismas mediante los puntos de historia. Como podemos ver en la Tabla 7.4, predecimos un tiempo de trabajo excesivo respecto al que se dedicó finalmente, debido a un problema que podemos ver en mayor detalle en la Sección 6.2.1, dándonos la suficiente experiencia para poder realizar futuras predicciones con mayor precisión.

  Tras esta reunión, se dio prácticamente por concluido el primer programa (véase Sección 6.1.2) del sistema, a la espera de especificar ciertos detalles finales del mismo con el Product Owner y a su limpieza y documentación.

7.1.4. Sprint 3 (10/04/2023 - 24/04/2023)

Durante este sprint se desarrollarán las distintas interfaces de usuario del segundo programa del sistema (véase Sección 6.1.3), además de añadir parte de su funcionalidad.

ID	Historia de usuario	Origen	Puntos de historia	Horas Invertidas	Estado
HU05	Como Product Owner quiero visualizar la información de los sensores para poder controlarlos	E02	2	6 horas	Completada
HU06	Como Product Owner quiero poder añadir nuevos sensores para poder mantener el sistema actualizado	E02	2	5 horas	Completada
HU07	Como Product Owner quiero poder añadir nuevos factores de calibración para poder mantener el sistema actualizado	E02	2	5.5 horas	Completada
HU28	Como Development Team quiero contar una interfaz de usuario principal para poder acceder al resto de funcionalidades del sistema		1	8 horas	Completada

Tabla 7.6: Historias de usuario del sprint 3

- **Sprint Planning (10/04/2023):** en esta reunión entre el Development Team y la Scrum Master, se decidieron qué partes del segundo programa (véase Sección 6.1.3) empezar a desarrollar, dando como resultado el Sprint Backlog de la Tabla 7.6, compuesto por las HU05, 06 y 07; cuyos puntos de historia fueron readaptados tras su especificación inicial (véase Tabla 7.1), además de la HU28, que surgió durante esta reunión. Se decidieron estas historias de usuario, dado que el Development Team no contaba con experiencia previa en la programación de GUIs en Python, con lo cual se optaron por las más sencillas con el objetivo de que éste se acomode a la tecnología, y de tener suficiente tiempo como para emplearlas en un patrón Modelo-Vista-Controlador (MVC), el cual se optó por adoptar para la separación de las distintas lógicas del programa (véase Sección 5.5.1).

  Además, se decidió realizar una reunión con el Product Owner a lo largo del sprint, con el objetivo de precisar los detalles finales del primer programa (véase Sección 6.1.2) del sistema, a parte de precisar también detalles del segundo.

- **Semana 1:** Durante esta primera semana, se crearon las distintas interfaces de usuario teniendo como base los bocetos de las Sección 5.1, además de empezar a programar la lógica de las interfaces de visualizar la información de los radiómetros y de asignar nuevos coeficientes de calibración a un sensor, correspondiéndose respectivamente con

las historias de usuario HU05 y HU07, las cuales fueron completadas posteriormente en esta primera mitad del sprint.

Durante el desarrollo, el Development Team se encontró con dificultades para adaptar el código Python generado por Qt Designer (véase Sección 4.1.8) para que este pudiese mostrar las interfaces (véase Sección 6.2.2), lo cual llevó a tener que reestructurar su código para que funcionase correctamente, invirtiendo así 1.5 horas de desarrollo para dar con la estructura correcta. Además, durante el desarrollo se encontró un error en la lógica del primer programa (véase Sección 6.1.2) a la hora de obtener los coeficientes de calibración vigentes, dado que actualmente solo coge los que tengan una mayor fecha asociada, lo cual puede suponer un problema al establecer fechas del futuro.

Finalmente, se estableció la historia de usuario HU29 en el Product Backlog, con el objetivo de constatar el uso del patrón MVC (véase Sección 5.5.1) durante el desarrollo del segundo programa (véase Sección 6.1.3). Podemos ver dicha historia en la Tabla 7.7. Debemos recordar que las historias de usuario con cero puntos de historia se consideran restricciones y por lo tanto no se le invierte horas como tal, dado que estas se abordan a lo largo del resto de historias de usuario.

- **Weekly Scrum (17/04/2023):** durante esta reunión, el Development Team y la Scrum Master se reunieron para comprobar los productos obtenidos de las historias de usuario HU05, HU07 y HU028, para ver si presentaban algún problema que impedía darlos por completados.

  Además, se comentaron los distintos errores presentados en la primera semana para determinar cómo solucionarlos, y se planificaron las actividades a realizar en la próxima semana, entre las que se encontraban: iniciar y completar la historia de usuario HU06, arreglar los errores previamente mencionados, añadir mensajes de error a las interfaces ya completadas e ir incluyendo nuevas secciones en la memoria del TFG, como por ejemplo las subsecciones 'Problemas' 6.2 y 'Pruebas' 6.3 de la Sección 'Implementación y Pruebas' 6.

- **Semana 2:** durante esta segunda parte del sprint, se completó la historia de usuario HU06 además de solucionar los problemas mencionados en los puntos anteriores, entre los que destacan: modificar la consulta SQL del primer programa (véase Sección 6.1.2) para que solo utilice los coeficientes de calibración vigentes y no los que tengan una fecha mayor, y añadir los mensajes de error para el usuario en caso de que no cumplimenten los formularios de los distintos diálogos.

  Por otra parte, se añadieron a la memoria las subsecciones 'Problemas' 6.2 y 'Pruebas y resultados' 6.3 de la Sección 'Implementación y Pruebas' 6, y las historias de usuario HU30, HU31 y HU32 al Product Backlog. Podemos ver estas historias de usuario en la Tabla 7.7.

- **Sprint Review/Retrospective (24/04/2023):** durante esta reunión, el Development Team y la Scrum Master revisaron el desarrollo del sprint de las últimas dos semanas.

  Destacaron, por ejemplo, la mala predicción del tiempo a invertir en la HU28 (véase Tabla 7.6), que se dio por la falta de experiencia del Development Team y por un problema que surgió durante el desarrollo (véase Sección 6.2.2); la necesidad de una revisión de las pruebas, ya que las entradas indicadas inicialmente eran demasiado

ID	Historia de usuario	Origen	Puntos de historia	Horas Invertidas	Estado
HU29	Como Development Team quiero que el segundo programa siga un patrón MVC para poder separar su distintas lógicas		0		En proceso
HU30	Como Development Team quiero que el segundo programa muestre mensajes de error para informar al usuario		0		En proceso
HU31	Como Development Team quiero que el sistema muestre las medidas en un gráfico de lineas para ver su evolución en el tiempo	HU03	3		Sin iniciar
HU32	Como Development Team quiero que el sistema pueble dinámicamente los filtros para mantenerlos actualizados con todas las opciones disponibles	HU04	2		Sin iniciar

Tabla 7.7: Historias de usuario añadidas al backlog durante el sprint 3

genéricas; y la revisión del software actual, con especial énfasis en como se solucionaron los problemas mencionados en puntos previos, dejando como tarea para futuros sprints hacer que el mensaje de error a la hora de no rellenar un formulario, este sea más específico indicando los campos que se hayan dejado erróneamente vacíos.

Además, finalmente no se realizó la reunión indicada en el Sprint Planning entre el Development Team y el Product Owner, dada la falta de disponibilidad del mismo. Sin embargo, dada la situación, se resolvieron parte de las dudas del Development Team vía correo electrónico, permitiéndole así avanzar en el desarrollo.

7.1.5. Sprint 4 (24/04/2023 - 08/05/2023)

Este sprint se dedicará a la programación de las funciones de visualización de medidas del segundo programa (véase Sección 6.1.3), además de tareas de documentación de la memoria y de resolver problemas surgidos en anteriores sprints.

- **Sprint Planning (24/04/2023):** durante esta reunión entre el Development Team y la Scrum Master, se decidió establecer las historias de usuario HU03, HU04, HU31, HU32 y la HU33, que esta última fue creada durante la reunión; como Sprint Backlog inicial. Podemos verlo en la Tabla 7.8.

  Además, como añadido a dichas historias de usuario, también se decidió dedicar tiempo a la documentación del TFG, en concreto a añadir pruebas a la Sección 6.3 y el modelo de casos de uso a la Sección 3; y también a la resolución de errores de sprints anteriores, como por ejemplo, hacer que los mensajes de error del segundo programa (véase Sección 6.1.3) sean más específicos.

- **Semana 1:** Durante esta primera semana, el Development Team completo las historias de usuario HU32, HU33 y HU04, además de dar casi por terminada la HU31, a la espera de confirmarlo en la reunión del Weekly Scrum; haciendo así que la función de representación gráfica de las medidas quede prácticamente completada y, por tanto, las funciones base del sistema, dejándonos la oportunidad de aprovechar para corregir errores y añadir funciones extra.

  Por otra parte, en cuanto a la documentación, se obtuvo la primera versión del diagrama de casos de uso del sistema y se diseñaron las distintas pruebas de las historias de usuario (véase Sección 6.3).

- **Weekly Scrum (02/05/2023):** durante esta reunión entre el Development Team y la Scrum Master, se dio por finalizada la historia de usuario HU31. Además, se decidió dedicar la siguiente semana del sprint a la corrección de un problema en las Vistas del segundo programa (véase Sección 6.1.3), que hacía que no se cumpliese el patrón MVC (véase Sección 5.5.1); a hacer más precisos los mensajes de error, a añadir pruebas (véase Sección 6.3) y el modelo de casos de uso (véase Sección 3.2) a la memoria, y a reunirnos con el Product Owner para acordar qué funcionalidades añadir al segundo programa (véase Sección 6.1.3) durante el resto de sprints.

- **Semana 2:** como dijimos en el anterior punto, durante esta semana el Development Team se encargó de la reestructuración de las Vistas y a precisar los mensajes de error

ID	Historia de usuario	Origen	Puntos de historia	Horas Invertidas	Estado
HU03	Como Product Owner quiero visualizar las medidas de los radiómetros para realizar un control de las mismas	E02	4	5,5 horas	Completada
HU04	Como Product Owner quiero poder aplicar filtros a como se muestran las medidas para poder verlas según me resulte conveniente	E02	3	5 horras	Completada
HU31	Como Development Team quiero que el sistema muestre las medidas en un grafico de lineas para ver su evolución en el tiempo	HU03	3	5,5 horas	Completada
HU32	Como Development Team quiero que el sistema pueble dinámicamente los filtros para mantenerlos actualizados con todas las opciones disponibles	HU04	2	3 horas	Completada
HU33	Como Development Team quiero que se ofrezca un filtro con los identificadores de los sensores y otro con el tipo de medida para poder personalizar la visualización de las medidas	HU04	1	2 horas	Completada

Tabla 7.8: Historias de usuario del sprint 4

ID	Historia de usuario	Origen	Puntos de historia	Horas Invertidas	Estado
HU34	Como Product Owner quiero poder añadir estaciones a la base de datos para poder mantener actualizada la base de datos				Sin Iniciar
HU35	Como Product Owner quiero poder editar las estaciones para poder mantenerlas actualizadas				Sin Iniciar
HU36	Como Product Owner quiero poder asignar sensores a las estaciones para poder mantener actualizada la base de datos				Sin Iniciar
HU37	Como Product Owner quiero poder filtrar las medidas según los sensores que hay en una estación en un cierto periodo para poder personalizar la visualización de las medidas				Sin Iniciar

Tabla 7.9: Historias de usuario añadidas al backlog durante el sprint 4

del segundo programa (véase Sección 6.1.3) de forma simultanea, dedicando en total 6 horas de trabajo.

Además, se realizó una reunión entre el Development Team y el Product Owner para presentar los avances realizados, de la cual surgieron una serie de errores en el segundo programa (véase Sección 6.1.3); que fueron resueltos tras 2 horas de trabajo; y nuevas funcionalidades que podemos ver reflejadas como historias de usuario en la Tabla 7.9.

El resto de la semana se dedicó a la documentación del TFG, añadiendo pruebas a la Sección 6.3 y el modelo de caso de uso a la Sección 3.2, además de empezar a crear los bocetos para las nuevas funciones.

- **Sprint Review/Retrospective (08/05/2023):** durante esta reunión, el Development Team y la Scrum Master dieron las historias de usuario de este sprint por completadas (véase Tabla 7.8), además de revisar las distintas partes añadidas a la memoria, como por ejemplo el modelo de casos de uso (véase Sección 3.2), las pruebas (véase Sección 6.3) y el manual de despliegue e instalación (véase Apéndice A); dejando ciertos pequeños cambios a realizar para el siguiente sprint.

Otro de los principales temas de la reunión, fue la reunión con el Product Owner durante la segunda semana del sprint, de la que surgieron una serie de nuevas funcionalidades exigidas por parte del mismo (véase Tabla 7.9) y un error en el primer programa (véase Sección 6.1.2) que se originó por una falta de comunicación entre el Development Team y el Product Owner. Finalmente, se decidió abordar ambas tareas en los siguientes sprints.

7.1.6. Sprint 5 (08/05/2023 - 22/05/2023)

Este sprint se dedicará a añadir funcionalidades respecto a las estaciones en las que se instalan los sensores, como por ejemplo añadirlos, editarlos etc.

- **Sprint Planning (08/05/2023):** durante esta reunión entre el Development Team y la Scrum Master, se estableció el Sprint Backlog a partir de las historias de usuario creadas en el sprint anterior (véase Tabla 7.10), además de dejar las tareas de ampliar y resolver los problemas de la memoria detectados en el sprint anterior como trabajo a realizar en caso de disponer de tiempo, como por ejemplo arreglar y ampliar la sección de pruebas (véase Sección 6.3) y el apéndice de manuales (véase Apéndice A).

  Al final, se estableció a modo de guía que para la primera semana había que completar las historias de usuario HU34, HU35 y HU36, y para la segunda semana la historia de usuario HU37.

- **Semana 1:** siguiendo lo planeado en el Sprint Planning, esta semana se completaron las historias de usuario HU34, HU35 y HU36 empleando los bocetos creados durante este mismo sprint (véase Figuras 7.1 y 7.2).

  Además, se dedicó el final del sprint a escribir la sección con los pasos a seguir para el primer arranque y el manual de usuario del primer programa (véase Sección A.3.3); a parte de hacer otros pequeños arreglos a los programas y de crear el archivo de requerimientos del proyecto, denominado *requirements.txt*, que contendrá una especificación de las distintas librerías Python de las que dependen los programas y sus correspondientes versiones.

- **Weekly Scrum (15/05/2023):** durante esta reunión entre el Development Team y la Scrum Master revisaron los productos de las historias de usuario completadas durante la última semana, a partir de lo cual se detectó un error en los inputs de una de las Vistas; y también se revisaron las distintas secciones de la documentación que fueron editadas durante esa misma semana, de lo cual también se detectaron otros errores.

  Finalmente, se decidió seguir con lo planeado y completar la historia de usuario HU37 (véase Tabla 7.10) durante la próxima semana, además de resolver los errores detectados durante la reunión y seguir completando partes de la memoria.

- **Semana 2:** durante esta segunda semana, se completó la historia de usuario HU37 (véase Tabla 7.10), además de completar el manual de usuario del segundo programa (véase Sección A.3.4) y de hacer una revisión de toda la memoria para saber que elementos añadir en los futuros sprints.

ID	Historia de usuario	Origen	Puntos de historia	Horas Invertidas	Estado
HU34	Como Product Owner quiero poder añadir estaciones a la base de datos para poder mantener actualizada la base de datos		2	5,25 horas	Completada
HU35	Como Product Owner quiero poder editar las estaciones para poder mantenerlas actualizadas		2	5,25 horas	Completada
HU36	Como Product Owner quiero poder asignar sensores a las estaciones para poder mantener actualizada la base de datos		2	5 horas	Completada
HU37	Como Product Owner quiero poder filtrar las medidas según los sensores que hay en una estación en un cierto periodo para poder personalizar la visualización de las medidas		2	5 horas	Completada

Tabla 7.10: Historias de usuario del sprint 5

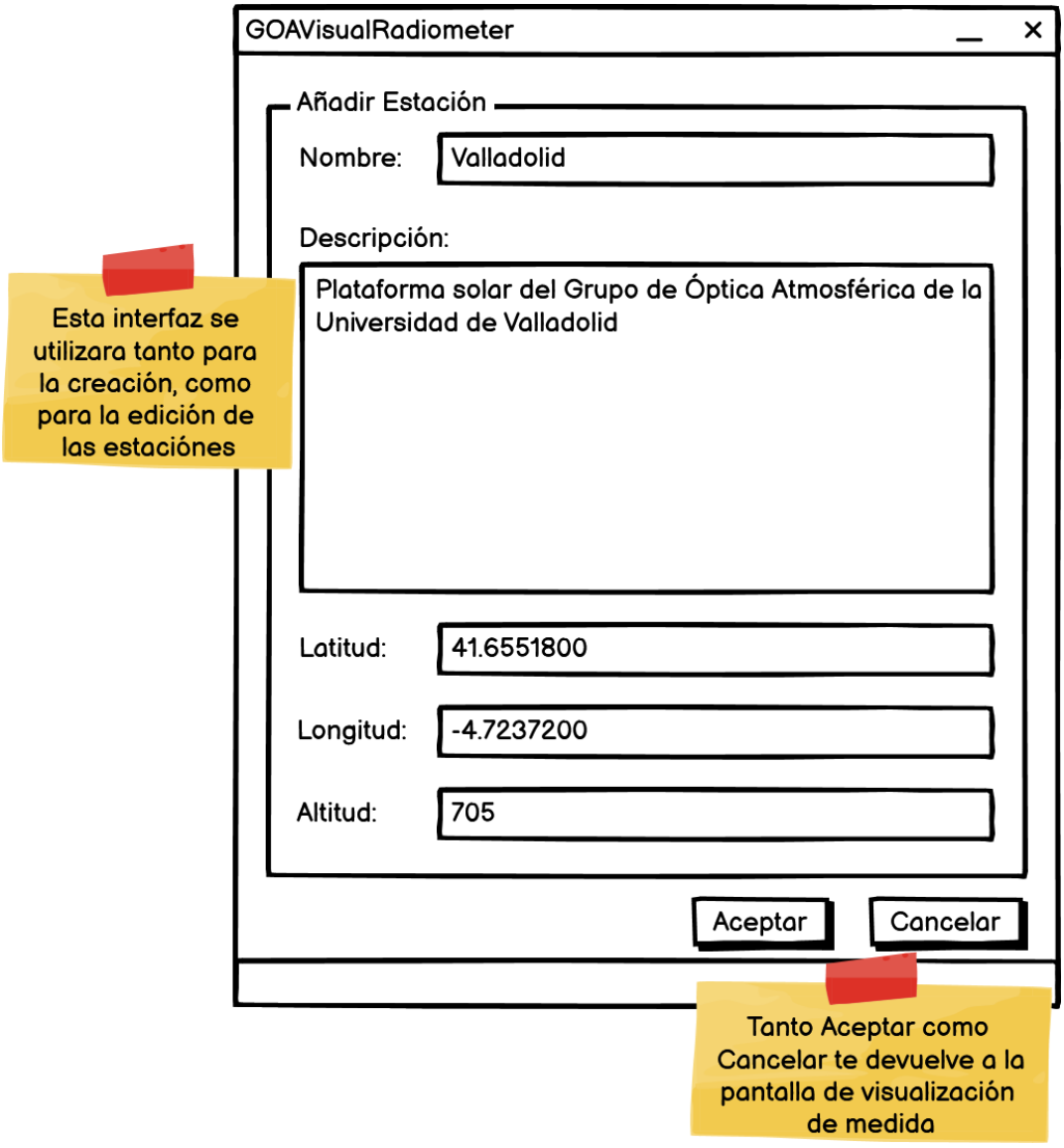

Figura 7.1: Pantalla para añadir/editar una estación

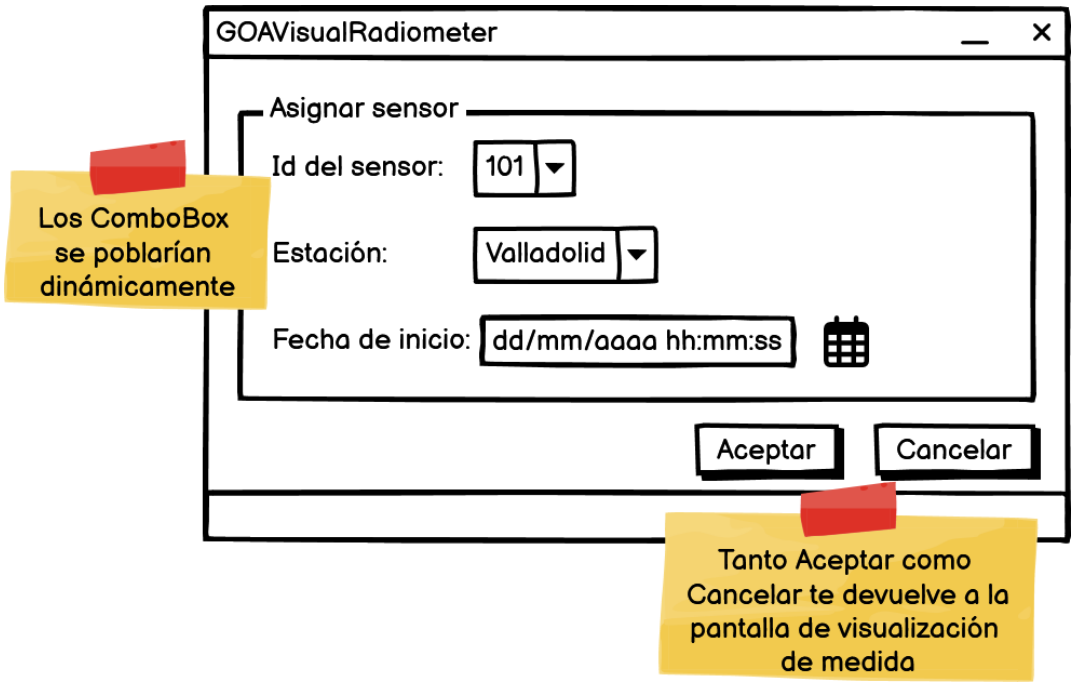

Figura 7.2: Pantalla para asignar un sensor a una estación

- **Sprint Review/Retrospective (22/05/2023):** durante esta reunión entre el Development Team y la Scrum Master, se revisaron los productos obtenidos a partir de las historias de usuario del Sprint Backlog (véase Tabla 7.10) y las distintas secciones de la memoria que fueron añadidas durante el sprint, sin encontrar ningún error significativo en ambos elementos.

7.1.7. Sprint 6 (22/05/2023 - 05/06/2023)

Este sprint lo dedicaremos a resolver el error del primer programa (véase Sección 6.1.2) que fue detectado al final del sprint 4 (véase Sección 7.1.5), además de seguir con la documentación del TFG.

- **Sprint Planning (22/05/2023):** durante esta reunión entre el Development Team y la Scrum Master, se decidió resolver un problema con la obtención de coeficientes de calibración del primer programa (véase Sección 6.1.2), además de seguir escribiendo la memoria.

  Por otra parte, se va a realizar una reunión entre el Development Team y el Product Owner para poder mostrar los productos obtenidos y comprobar que cumplen todas las necesidades del Product Owner.

- **Semana 1:** durante esta primera semana, se solucionó el problema de obtención de coeficientes mencionado al principio del sprint, dando así por concluidos los programas para empezar su fases de prueba y mantenimiento. Además, se siguió completando la memoria.

  Por otra parte, se realizó una reunión entre el Development Team y el Product Owner, donde se les mostró ambos programas y mostraron su conformidad con el producto final.

- **Weekly Scrum (29/05/2023):** durante esta reunión entre el Development Team y la Scrum Master, se revisaron las distintas secciones de la memoria que fueron editadas durante la primera semana en busca de errores.

  Finalmente, se decidió seguir escribiendo la memoria durante la segunda semana, a la espera de recibir feedback del Product Owner de los dos programas del sistema.

- **Semana 2:** durante esta semana se revisó la memoria del TFG, además de producirse una reunión entre el Development Team y el Product Owner para revisar el trabajo realizado durante todo el periodo de desarrollo, a parte de comentar las distintas dificultades que se encontró el personal del GOA-UVa durante el uso de los programas, concluyendo en que, salvo pequeños detalles, no se habían encontrado grandes problemas.

- **Sprint Review/Retrospective (05/06/2023):** durante esta reunión entre el Development Team y la Scrum Master, se revisaron las distintas partes de la memoria editadas durante la segunda semana, además de comentar la reunión con el Product Owner realizada esa misma semana.

  Finalmente, se dejó casi por terminada la memoria, a excepción del resumen final del seguimiento; y por tanto se dio por concluido el desarrollo del proyecto en este sprint.

7.2. Resumen del desarrollo del proyecto

En esta sección se habla de forma resumida sobre cómo se ha ido desarrollando el proyecto, echando la vista atrás para comparar los tiempos de desarrollo y costes planificados (véase Sección 2.2 y Sección 2.4.2), con los que verdaderamente han sido invertidos durante el desarrollo.

7.2.1. Tiempo empleado

Como pudimos ver en la Sección 2.2, la planificación inicial del desarrollo del proyecto se empezó a definir teniendo como base la media de 300 horas como referencia de las 12 ECTS correspondientes a la asignatura [33]. Teniendo esto en cuenta, se decidió organizar el desarrollo inicialmente en 6 sprints de de dos semanas, aunque se decidió añadir otros 2: un sprint inicial o 'Sprint 0' para la planificación y formación del alumno; y otro de refuerzo al final de desarrollo en caso de necesitar más tiempo de desarrollo.

Sprint	Horas invertidas
Sprint 0	52,5
Sprint 1	28,5
Sprint 2	32,5
Sprint 3	37,5
Sprint 4	42,5
Sprint 5	38
Sprint 6	76,5
Total	**308**

Tabla 7.11: Horas invertidas en cada sprint

Con toda esta información, se estimó que el tiempo de desarrollo iba a ser de **350 horas** en caso de cumplir el plazo y que la fecha de finalización iba a ser el 05/06/2023, pero, en caso de no cumplirlo, el desarrollo duraría una **400 horas** y finalizaría el 19/06/2023. Dado que en este caso hemos cumplido con el plazo, y por lo tanto no hemos usado el sprint de refuerzo; nuestra duración de referencia serán las 350 horas planeadas.

Como podemos ver en la Tabla 7.11, finalmente se invirtieron 308 horas en el proyecto, teniendo en cuenta todas las actividades necesarias: formación, documentación, reuniones y actividades propias del desarrollo. Esto resulta en 42 horas menos que las planificadas, lo que indica que nuestra planificación introdujo un exceso de horas previendo inicialmente más complicaciones durante el desarrollo.

Centrándonos más en la horas invertidas en cada sprint (véase Figura 7.3), podemos ver que en el 'Sprint 0' se invirtieron muchas horas si se compara con los siguientes sprints, y esto se debe a que, por aquel entonces, el alumno no contaba con excesivas responsabilidades por parte tanto de las prácticas en empresa, como de las asignaturas de la carrera. En los siguientes cinco sprints, esta cantidad de horas invertidas se reduce drásticamente, dada la aparición de dichas responsabilidades con el resto de asignaturas, aunque a medida que se va avanzando en el proyecto podemos ver que las horas invertidas van aumentando sprint a sprint, puesto que el alumno iba adaptándose mejor al ritmo de trabajo. Finalmente, en el 'Sprint 6' fue donde se invirtieron más horas, ya que al haber finalizado tanto el periodo de clases, como el de prácticas, se dispuso de más tiempo para dedicarlo al TFG lo cual permitió no tener que utilizar el sprint de refuerzo y dedicar muchas horas a finalizar toda la labor de documentación del proyecto y de redacción y corrección de esta memoria.

7.2.2. Costes finales

Como vimos en la Sección 2.4.2, se estimó un coste de **471,55 €** para el desarrollo y dado que hemos cumplido con el plazo, el coste sería el mismo.

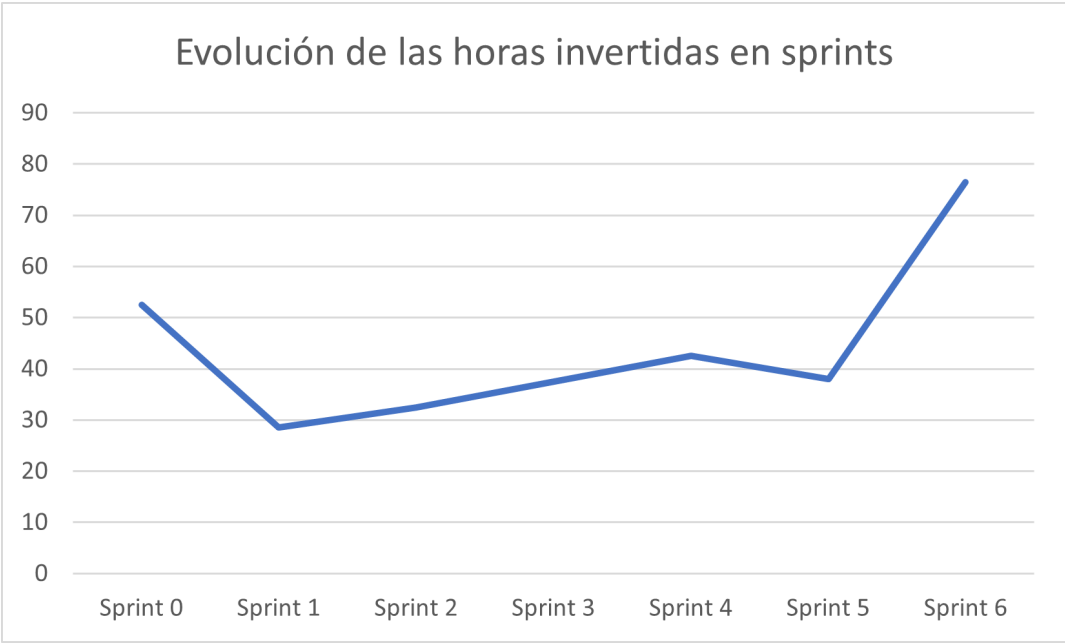

Figura 7.3: Evolución de las horas invertidas durante los sprints

Capítulo 8

Conclusiones

8.1. Conclusiones

Esta ha sido una experiencia enriquecedora, no solo por haber conseguido unas bases muy firmes en el lenguaje Python, sino también por ser mi primera experiencia más cercana a la labor de un ingeniero, intentando resolver las necesidades planteadas por un cliente a través de la informática.

Además, ha supuesto también mi primera experiencia real con la metodología Scrum, donde pude experimentar lo que sucede cuando se producen problemas de comunicación entre las distintas partes del equipo Scrum, siendo en este caso entre el Product Owner y el resto de miembros del equipo, dificultando así el desarrollo del mismo.

Centrándonos más en los objetivos impuestos al principio del desarrollo (véase Sección 1.3 y Sección 1.4), se han cumplido todos los objetivos iniciales, dado que los programas que fueron planeados inicialmente se han desarrollado por completo, e incluso se ha llegado a hacer más de lo esperado ,añadiéndoles funcionalidades extra y completando así los objetivos del desarrollo. En cuanto a los objetivos de formación, como mencioné anteriormente, creo haber conseguido un buen conocimiento general sobre el lenguaje Python, además de recuperar y afianzar mis conocimientos en MySQL, siendo ambas unas herramientas importantes para mi futuro laboral.

8.2. Líneas de trabajo futuras

Se plantean las siguientes ideas como mejoras futuras que hacer a los programas:

- **Añadir nuevas clases/tablas al Modelo:** en un futuro puede llegar a necesitarse añadir nuevas tablas a las bases de datos para poder almacenar nueva información.

- **Mejorar el filtro 'Estaciones' de GOAVisualRadiometer:** actualmente lo que hace este filtro es buscar qué sensores se asociaron a las estaciones seleccionadas en el periodo seleccionado, haciendo una búsqueda en la tabla Rad_Instalation de la base de datos (véase Sección 5.3), cuando lo que debería hacer es mostrar los sensores que tomaron medidas cuando estaban asociados a dichas estaciones en dicho periodo. Dado que esta es una funcionalidad que se implementó prácticamente al final del desarrollo, se optó por mantener su versión actual, dado que la versión mejorada exigía más tiempo de desarrollo para sacarla adelante.
- **Añadir funcionalidades al menubar de GOAVisualRadiometer:** se decidió poner como acceso a las distintas funcionalidades del programa desde la ventana principal a través de un menubar, dado que este tiene un diseño escalable, permitiendo añadir nuevas funcionalidades basadas en las distintas clases del modelo.

Bibliografía

[1] altexsoft. Understanding object-relational mapping: Pros, cons, and types. https://www.altexsoft.com/blog/object-relational-mapping/. Accessed: 2023-3-11.

[2] astah. Leverage the power of software modeling. https://astah.net/. Accessed: 2023-3-4.

[3] Atlassian. Epics ágiles: definición, ejemplos y plantillas. https://www.atlassian.com/es/agile/project-management/epics. Accessed: 2023-3-01.

[4] Atlassian. Historias de usuario con ejemplos y plantillas. https://www.atlassian.com/es/agile/project-management/user-stories. Accessed: 2023-3-01.

[5] Balsamiq. Quick and easy wireframing tool sketch your user interface ideas and get everyone on the same page. https://balsamiq.com/wireframes/. Accessed: 2023-2-28.

[6] Bob Hughes y Mike Cotterell. Software project management, fifth edition, mcgraw-hill 2009. Accessed: 2023-3-02.

[7] Boletin Oficial del Estado. Convenio colectivo estatal de empresas de consultoría de sistemas de información. https://www.boe.es/boe/dias/2018/03/06/pdfs/BOE-A-2018-3156.pdf. Accessed: 2023-3-10.

[8] GeeksforGeeks. Method resolution order in python inheritance. https://www.geeksforgeeks.org/method-resolution-order-in-python-inheritance/. Accessed: 2023-4-19.

[9] GitLab. Software. faster. https://about.gitlab.com/. Accessed: 2023-3-4.

[10] GOA-UVa. Grupo de Óptica atmosférica universidad de valladolid. http://goa.uva.es/. Accessed: 2023-3-10.

[11] Indeed. ¿cuánto se gana como uno desarrollador/a de software en españa? https://es.indeed.com/career/desarrollador-de-software/salaries. Accessed: 2023-3-8.

[12] Le Treut, H., R. Somerville, U. Cubasch, Y. Ding, C. Mauritzen, A. Mokssit, T. Peterson and M. Prather. *Climate Change 2007: The Physical Science Basis. Contribution of*

Working Group I to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change, chapter 2007: Historical Overview of Climate Change, pages 93–128. Cambridge University Press, Cambridge, United Kingdom and New York, 2007.

[13] Matplotlib.org. Interactive navigation. `https://matplotlib.org/3.2.2/users/navigation_toolbar.html`. Accessed: 2023-5-24.

[14] Matplotlib.org. Matplotlib: Visualization with python. `https://matplotlib.org/`. Accessed: 2023-4-8.

[15] mdn web docs. Mvc. `https://developer.mozilla.org/es/docs/Glossary/MVC`. Accessed: 2023-4-17.

[16] MySQL. Mysql workbench. enhanced data migration. `https://www.mysql.com/products/workbench/`. Accessed: 2023-3-14.

[17] Overleaf. Latex, evolucionado el editor de latex fácil de usar, online y colaborativo. `https://es.overleaf.com/`. Accessed: 2023-3-4.

[18] Progressive Coder. Typeorm active record vs data mapper – which one is better? `https://progressivecoder.com/typeorm-active-record-vs-data-mapper/#1-typeorm-active-record-pattern`. Accessed: 2023-3-11.

[19] Project Management Institute. The meaning of risk in an uncertain world. `https://www.pmi.org/learning/library/project-risks-uncertain-world-8392#:~:text=The%20PMBOK%C2%AE%20Guide%20describes,objectives%20of%20the%20planned%20endeavour.` Accessed: 2023-3-02.

[20] PyInstaller. Pyinstaller manual. `https://pyinstaller.org/en/stable/`. Accessed: 2023-5-19.

[21] PyPl. Pyside. `https://pypi.org/project/PySide/`. Accessed: 2023-4-3.

[22] Python. Download the latest version for windows. `https://www.python.org/downloads/`. Accessed: 2023-3-4.

[23] PythonGUIS. First steps with qt designer. `https://www.pythonguis.com/tutorials/pyside6-first-steps-qt-designer/`. Accessed: 2023-4-19.

[24] Python.org. multiprocessing — paralelismo basado en procesos. `https://docs.python.org/es/3/library/multiprocessing.html`. Accessed: 2023-4-19.

[25] Python.org. threading — paralelismo basado en hilos. `https://docs.python.org/es/3/library/threading.html#module-threading`. Accessed: 2023-4-19.

[26] Qt Documentation. Qt designer manual. `https://doc.qt.io/qt-6/qtdesigner-manual.html`. Accessed: 2023-4-3.

[27] Qt Documentation. Using .ui files from designer or qtcreator with quiloader and pyside6-uic. `https://doc.qt.io/qtforpython/tutorials/basictutorial/uifiles.html`. Accessed: 2023-4-3.

[28] Rocket.Chat. Deje que cada conversación fluya - sin concesiones. https://es.rocket.chat/. Accessed: 2023-3-4.

[29] Scrum Guides. Scrum guides: The 2020 scrum guide. https://scrumguides.org/scrum-guide.html. Accessed: 2023-2-27.

[30] Scrum.org. What is scrum? https://www.scrum.org/resources/what-is-scrum. Accessed: 2023-2-27.

[31] SQLAlchemy. The python sql toolkit and object relational mapper. https://www.sqlalchemy.org/. Accessed: 2023-3-6.

[32] Stack Overflow. Calling parent class __init__with multiple inheritance, what's the right way? https://stackoverflow.com/questions/9575409/calling-parent-class-init-with-multiple-inheritance-whats-the-right-way. Accessed: 2023-4-19.

[33] Universidad de Valladolid. Proyecto docente del trabajo de fin de grado 2021-2022 (Mención Ingeniería de Software). https://albergueweb1.uva.es/guia_docente/uploads/2022/545/46976/1/Documento.pdf. Accessed: 2023-6-7.

[34] Visual Studio Code. Code editing.redefined.free. built on open source. runs everywhere. https://code.visualstudio.com/. Accessed: 2023-3-4.

[35] Wikipedia. Modelo–vista–controlador. https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador. Accessed: 2023-4-17.

Apéndice A

Manuales

A.1. Manual de despliegue e instalación

Para poder utilizar ambos programas, se debe instalar Git en nuestra máquina Linux para clonar el repositorio del proyecto, aunque alternativamente se puede descargar desde el propio repositorio (véase apéndice B).

```
$ sudo apt-get install git
```

```
$ git clone https://gitlab.inf.uva.es/jaisaiz/tfggoa-jaime-saiz-losada.git
```

También debemos instalar el sistema gestor de bases de datos MySQL (véase Sección 4.1.4) en nuestra máquina, donde crearemos un usuario con todos los permisos, que será el que indicaremos en los archivos de configuración 'config.yml' de ambos programas.

```
$ sudo apt install mysql-server
```

Una vez tengamos el repositorio descargado o clonado, se podrá acceder al código fuente de ambos programas y, en caso de querer modificarlo o ejecutarlo (véase Sección A.2) se deberá descargar la última versión de Python (véase apéndice B).

A.2. Manual de mantenimiento

A.2.1. Preparación del entorno de trabajo

Para poder realizar el mantenimiento o modificación del código, primero debemos preparar nuestro entorno de trabajo.

Como dijimos en el 'Manual de despliegue e instalación' (véase Sección A.1), debemos tener instalada en nuestra máquina la última versión de Python (véase apéndice B) y el gestor de base de datos MySQL (véase Sección 4.1.4) en nuestra máquina Linux; además de contar con un IDE para poder visualizar y modificar el código cómodamente. En este caso, recomendamos el uso de VSCode (véase Sección 4.1.1), aunque alternativamente se puede emplear otro que cumpla la misma función.

Una vez tengamos todo preparado, debemos resolver las dependencias de librería que puede llegar a presentar el código. Para ello, en el directorio raíz del repositorio podemos encontrar el archivo de requisitos 'requirements.txt', a partir del cual podemos descargar todas las librerías que el proyecto necesita metiendo en nuestro terminal el siguiente comando:

```
$ pip install -r requirements.txt
```

Otra aplicación que conviene tener instalada es PyInstaller (véase Sección 4.1.11), la cual nos va a permitir generar nuevos ejecutables a partir del código modificado. Para instalarla, debemos ejecutar el siguiente comando en un terminal:

```
$ pip install -U pyinstaller
```

Finalmente, aunque no es del todo necesario; podemos instalar en nuestra máquina MySQL Workbench (véase Sección 4.1.5) para poder visualizar y gestionar la base de datos de una forma más intuitiva que por terminal.

A.2.2. Estructura del código

En esta sección nos referiremos únicamente a los archivos contenidos en el directorio *src*, dado que es donde se encuentra el código fuente de ambos programas y además, si se desea obtener más información del código a parte de lo ya explicado en esta sección, dentro del mismo podemos encontrar comentarios que explican cada parte de forma detallada.

GOASaveMeassure

La estructura del primer programa es muy simple, dado que a grandes rasgos es solo un script, pero aun así conviene explicar brevemente qué función cumple cada archivo.

- **config.yml:** este es el archivo de configuración que emplea el programa para crear y conectarse a la base de datos, además de conocer el directorio donde buscar los archivos a procesar y el directorio donde almacenarlos una vez procesados.

  Como se puede deducir por la extensión del archivo, este es un archivo YAML, lo que significa que en caso de querer modificar dicho archivo para pasar más información al programa, se debe cumplir el formato *clave* : '*valor*' por cada campo añadido.

- **db.py:** este archivo se encarga de leer los campos de 'config.yml' relativos a la base de datos para establecer una conexión con la misma, además de crear una sesión para poder realizar consultas y los metadatos para el ORM (véase Sección 5.4.1).

- **main.py:** este archivo se encarga de dirigir la funcionalidad de todo el programa, desde la captura de datos de los archivos, hasta su almacenamiento en la base de datos tras su procesado.

- **models.py:** este archivo contiene las clases que conforman las tablas de la base de datos y que emplearemos para crearla y gestionarla siguiendo el patrón ORM (véase Sección 5.4.1).

GOAVisualRadiometer

La estructura de este programa es mucho más compleja que la de GOASaveMeassure, dado que sigue patrones más complejos como el patrón MVC (véase Sección 5.5.1). Su estructura sería la siguiente:

- **Model:** esta carpeta contiene todos los archivos que conforman el Modelo del patrón MVC (véase Sección 5.5.1). Contiene los siguientes archivos:

  - **db.py:** este archivo se encarga de establecer la conexión con la base de datos leyendo los campos del archivo 'config.yml', además de almacenar todas las operaciones que puede realizar el programa sobre la base de datos.
  - **models.py:** este archivo contiene las clases que conforman las tablas de la base de datos y que emplearemos para gestionarla siguiendo el patrón ORM Data-Mapper (véase Sección 5.4.1).

- **Views:** esta carpeta contiene todos los archivos que conforman las Vistas y sus correspondientes Controladores del patrón MVC (véase Sección 5.5.1). Para mayor detalle, cada archivo contiene la clase generada por Qt Designer (véase Sección 4.1.8) de una Vista en concreto, luego la clase que permite utilizar la anterior clase como una Vista, además de contener sus métodos correspondientes; y finalmente nos encontraríamos con la clase Controlador.

- **config.yml:** este es el archivo de configuración que emplea el programa para conectarse a la base de datos.

  Como se puede deducir por la extensión del archivo, este es un archivo YAML, lo que significa que en caso de querer modificar dicho archivo para pasar más información al programa, se debe cumplir el formato *clave* : '*valor*' por cada campo añadido.

- **main.py:** este es el archivo que pone en ejecución todo el programa.

A.2.3. Actualización del archivo de requerimientos y creación de los ejecutables

Una vez se hayan realizado todas las modificaciones deseadas a los programas, convendría que actualizases el archivo de requerimientos y que generes nuevos ejecutables a partir del código fuente.

- **Para actualizar el archivo de requerimientos**, se debe abrir un terminal en el directorio raíz del repositorio en tu equipo, y ejecutar el siguiente comando:

  ```
  $ pip freeze > requirements.txt
  ```

  De esta forma, en caso de haber añadido nuevas dependencias al proyecto, estas se verán reflejadas en el archivo 'requirements.txt' para futuras modificaciones.

- **Para generar el nuevo ejecutable de un programa**, se debe abrir un terminal en el directorio raíz del código fuente del programa y usar la herramienta PyInstaller (véase Sección 4.1.11) ejecutar el siguiente comando:

  ```
  $ pyinstaller.exe –onefile -n <nombre del programa> main.py
  ```

  Una vez hecho esto, se habrán generado varias carpetas, entre ellas la carpeta *dist*, que contendrá el ejecutable del programa.

A.3. Manual de usuario

A.3.1. Creación de los ejecutables

Antes de empezar a utilizar los programas, debemos crear sus ejecutables a partir del código fuente.

Para ello, primero tenemos que tener instalada la última versión de Python (véase Apéndice B) y el programa PyInstaller (véase Sección 4.1.11) mediante el siguiente comando:

```
$ pip install -U pyinstaller
```

Una vez tengamos todo preparado, debemos resolver las dependencias de librería que puede llegar a presentar el código. Para ello, en el directorio raíz del repositorio podemos encontrar el archivo de requisitos 'requirements.txt', a partir del cual podemos descargar todas las librerías que el proyecto necesita metiendo en nuestro terminal el siguiente comando:

```
$ pip install -r requirements.txt
```

Finalmente, crearemos los ejecutables con el programa PyInstaller (véase Sección 4.1.11) que instalamos anteriormente. Para ello, nos situaremos en el directorio raíz de cada uno de los códigos fuente de los programas, y ejecutaremos el siguiente comando en ambos directorios desde un terminal:

```
# Campos de conexión a la BD
username: '' #Nombre de usuario
password: '' #Contraseña
host: '' #Nombre de host
port: '' #Puerto
database: '' #Nombre BD

# Campos de los archivos con los que trabajar
origpath: '' #Carpeta donde buscar archivos de medidas
destpath: '' #Carpeta donde se almacenan los archivos procesados
```

Figura A.1: Configuración de GOASaveMeassure

```
# Campos de conexión a la BD
username: '' #Nombre de usuario
password: '' #Contraseña
host: '' #Nombre de host
port: '' #Puerto
database: '' #Nombre BD
```

Figura A.2: Configuración de GOAVisualRadiometer

```
$ pyinstaller.exe –onefile -n <nombre del programa> main.py
```

Una vez hayamos hecho esto, se habrán generado varias carpetas, entre ellas la carpeta *dist*, que contendrá el ejecutable del programa y que junto con su correspondiente archivo de configuración que podemos ver en las Figuras A.1 y A.2, ya podremos empezar a usar los programas.

A.3.2. Primer arranque

Antes del primer arranque, debemos cumplimentar los archivos 'config.yml' que se encuentran junto a los ejecutables de ambos programas y de los que podemos ver unos ejemplos en las Figuras A.1 y A.2. **Importante:** para cada línea debe respetarse el formato *clave* : '*valor*' en ambos archivos y cada archivo de configuración se debe de encontrar en la misma carpeta que su correspondiente ejecutable, además de que ambos ejecutables deben de estar en carpetas distintas. Otra cosa que debemos destacar, es que si no se configuran bien estos archivos de configuración, ninguno de los programas funcionará. En el caso de GOASaveMeassure, se reflejará el error de conexión en el archivo de log (véase Sección A.3.3), y en el caso de GOAVisualRadiometer, el programa directamente no arrancará, con lo cual habrá que revisar la configuración para solucionarlo.

Una vez tengamos ambos archivos configurados, ejecutaremos el programa GOASave-

Meassure sin tener un archivo '.dat' en la carpeta especificada en el campo *origpath*, con el objetivo de crear la base de datos que va a usar ambos programas y poblarla con unas entradas básicas para su correcto funcionamiento inicial.

Para terminar, la versión inicial de la base de datos contará con la información de los sensores 101, 102, 103, 105 y 107, además de unos coeficientes de calibración para cada uno de ellos y estarán asignados a la estación de Valladolid, la cual estará almacenada con su correspondiente información (toda esta información, salvo los coeficientes, se pueden visualizar mediante GOAVisualRadiometer. Para ello véase Sección A.3.4). Es por esto que, antes de empezar a almacenar información de los archivos, se recomienda utilizar GOAVisualRadiometer para establecer los coeficientes de calibración adecuados y así no procesar de forma errónea las medidas (véase Sección A.3.4).

Si ya hemos completado todos estos pasos, ya podemos utilizar ambos programas.

A.3.3. GOASaveMeassure

Funcionamiento

Este programa fue concebido para que fuese ejecutado automáticamente en un sistema Linux cada cierto periodo de tiempo, programando dichas ejecuciones mediante el gestor de tareas, aunque también se puede ejecutar manualmente.

Su funcionamiento es muy sencillo: cada vez que se ejecuta el programa, este busca archivos con extensión '.dat' (para más información, véase Sección A.3.3) en la carpeta especificada en el campo *origpath* del archivo 'config.yml', para procesarlos multiplicando las medidas por el coeficiente de calibración vigente en la fecha de toma, almacenar su información en la base de datos, y una vez hayan sido completamente procesados, mover dichos archivos a la carpeta especificada en el campo *destpath* del archivo de configuración.

Finalmente, cada vez que se procesa un archivo, el programa genera una entrada en el archivo de log del mes de ejecución en la carpeta 'logs' (para más información, véase Sección A.3.3).

Archivos

GOASaveMeassure procesa archivos '.dat' que cumplen un formato del que podemos ver ejemplos en la Figura A.3 y que pasaremos a describir a continuación.

Cada línea en los archivos representa una medida de un sensor tomada en un momento determinado. Para representar toda esa información, cada línea se divide en una serie de campos separados por comas que pasaremos a detallar apoyándonos en la Figura A.3 como ejemplo:

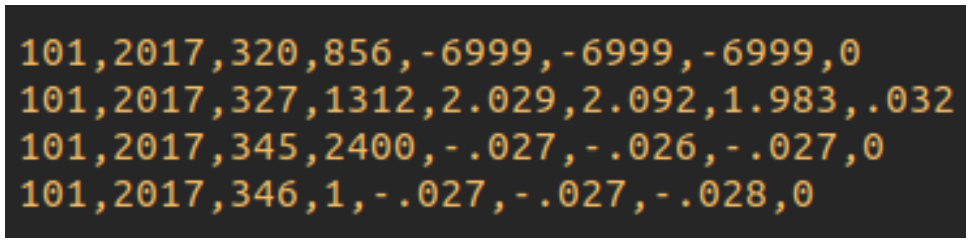

Figura A.3: Ejemplo de medidas de un archivo

- **Identificador del sensor:** el primer campo que nos encontramos es el identificador numérico del sensor con el que se ha tomado la medida.
- **Año:** el segundo campo indica el año en el que se ha tomado la medida.
- **Día del año:** el tercer campo es el día del año en el que se ha tomado la medida.
- **Hora de toma:** el cuarto campo es la hora a la que se ha tomado la medida, así pues en la Figura A.3, la primera medida fue tomada a la 8:56, la segunda a las 13:12, la tercera a las 00:00 del siguiente día y la cuarta a las 00:01.
- **Medida media:** el quinto campo indica la media de las medidas observadas durante el minuto.
- **Medida máxima:** el sexto campo indica la máxima de las medidas observadas durante el minuto.
- **Medida mínima:** el séptimo campo indica la mínima de las medidas observadas durante el minuto.
- **Desviación estándar:** el último campo indica la desviación estándar de las medidas observadas durante el minuto.

Finalmente, hay que destacar que en el caso de que dos o más líneas tengan el mismo identificador, año, día y hora; solo se aceptará como válida la primera línea que se encuentre el programa, y descartará el resto.

Logs

Cuando se ejecuta por primera vez GOASaveMeassure, este crea en el mismo directorio una carpeta llamada 'logs', donde se guardarán los archivos de log que el programa va a ir generando.

Cada vez que el programa se ejecuta, este busca un archivo de log correspondiente al mes en el que se ejecuta y en caso de que no lo encuentre, lo genera. Para poder identificarlos, cada archivo es nombrado siguiendo la siguiente estructura: *<año en formato de cuatro cifras><mes en formato de dos cifras>*_log.txt, de tal forma que si tenemos por

ejemplo un archivo nombrado como '202305_log.txt', quiere decir que ese archivo de log es el correspondiente al mes de mayo de 2023.

Centrándonos ya en el contenido de los logs, cada vez que el programa procesa un archivo añade una entrada titulada con su nombre, como podemos ver en la Figura A.4, donde podemos ver una entrada para los archivos *202303091136_CR23X_FSArea1.dat* y *202304191136_CR23X_FSArea1.dat*. Después del título, se indica la información de lo que ha hecho el programa con las medidas de cada sensor dependiendo de si dicho sensor está o no almacenado en la base de datos, o si por el contrario no se ha encontrado un coeficiente de calibración para procesar las medidas:

- **En el caso de estar almacenado**, para cada sensor se indicará la cantidad de medidas que se han guardado e ignorado, en caso de que esta esté duplicada (véase Sección A.3.3). Podemos ver ejemplos de este tipo de líneas en la Figura A.4, en la entrada del archivo *202303091136_CR23X_FSArea1.dat*; donde para todos los sensores se han añadido 0 medidas, dado que ya estaban en la base de datos; y se han ignorado dichas medidas duplicadas, siendo una por cada sensor.
- **En el caso de no estar almacenado**, para cada sensor se indicarán las medidas que han sido ignoradas, dado que al no estar los sensores almacenados estas no se aceptan; con el objetivo de indicar qué sensores añadir a futuro. Podemos ver varios ejemplos de estas en la Figura A.4, en la entrada del archivo *202304191136_CR23X_FSArea1.dat*; donde para todos los sensores se han ignorado sus medidas, ya que estos no están en la base de datos.
- **En el caso de no encontrar un coeficiente de calibración para una medida**, se añadirá una línea como podemos ver en las primeras líneas de la entrada del archivo *202304191137_CR23X_FSArea1.dat* en la Figura A.4, donde indicamos que se han ignorado dos medidas del sensor 101 y una medida del sensor 102 porque no se les han encontrado un coeficiente de calibración con el que procesarlas.

Finalmente, hay otros dos tipo de entradas que pueden aparecer en el log y que no están relacionadas con los archivos, sino con el archivo 'config.yml' (véase Figura A.4):

- **Error al conectarse a la base de datos:** este error ocurre cuando el programa no ha podido conectarse a la base de datos por una mala configuración de sus campos.
- **Error encontrar las carpetas:** este error ocurre cuando las carpetas especificadas en los campos *origpath* y *destpath* no existen o no están bien especificadas.

A.3.4. GOAVisualRadiometer

A grandes rasgos, el funcionamiento de este programa consiste en la visualización gráfica de las medidas almacenadas por medio de GOASaveMeassure (véase Sección A.3.3) en la base de datos, además de añadir nuevos elementos a la misma, como por ejemplo nuevos sensores, estaciones ..., y muchas más funcionalidades que pasaremos a describir ventana a ventana.

```
Error al encontrar las carpetas

Error al conectarse a la base de datos

202303091136_CR23X_FSArea1.dat
------------------------------
Sensor 103: se han añadido 0 medidas e ignorado 1 medidas duplicadas
Sensor 102: se han añadido 0 medidas e ignorado 1 medidas duplicadas
Sensor 101: se han añadido 0 medidas e ignorado 1 medidas duplicadas

202304191136_CR23X_FSArea1.dat
------------------------------
Sensor 104: se han ignorado 208041 medidas dado que el sensor no esta en la base de datos
Sensor 106: se han ignorado 208041 medidas dado que el sensor no esta en la base de datos
Sensor 108: se han ignorado 208044 medidas dado que el sensor no esta en la base de datos
Sensor 109: se han ignorado 208044 medidas dado que el sensor no esta en la base de datos

202304191137_CR23X_FSArea1.dat
------------------------------
Sensor 101: se han ignorado 2 medidas dado que no se le ha encontrado un coeficiente de calibración
Sensor 102: se han ignorado 1 medidas dado que no se le ha encontrado un coeficiente de calibración
Sensor 102: se han añadido 2 medidas e ignorado 0 medidas duplicadas
Sensor 101: se han añadido 2 medidas e ignorado 2 medidas duplicadas
```

Figura A.4: Ejemplo de archivo de log

Medidas

Esta es la ventana que se nos muestra una vez arrancamos el programa (véase Figura A.5) y que sirve como nexo para acceder al resto de funcionalidades del programa.

Como podemos ver a simple vista, la principal funcionalidad de esta ventana es la de mostrar de forma gráfica las medidas de los sensores almacenadas en la base de datos, para ver su evolución a lo largo del tiempo. Para ello, contamos con varios filtros:

- **Fecha de inicio/fin:** estos filtros sirven para indicar el periodo de tiempo del que queremos ver representadas las medidas, además de trabajar en conjunto con el filtro 'Estaciones' para mostrar los sensores asignados en dicho periodo a las estaciones seleccionadas en el filtro 'Radiómetros'.
- **Estación:** este filtro nos mostrará las distintas estaciones almacenadas en la base de datos y nos servirá para, en conjunto con los filtros 'Fecha de inicio/fin', poblar el filtro 'Radiómetros' tal y como dijimos en el anterior punto.
- **Radiómetros:** este filtro nos permite seleccionar de que sensores queremos ver las medidas, lo cual hará que la pantalla se actualice mostrando las gráficas y una leyenda explicativa de las mismas, tal y como podemos ver en la esquina superior derecha de la gráfica en la Figura A.5.
- **Tipo de datos:** este filtro nos permite elegir que tipo de medidas deseamos ver, si las brutas, las procesadas o ambas. Las medidas brutas se representarán en lineas continuas, mientras que la procesadas por lineas discontinuas y esto se verá reflejado en la leyenda.

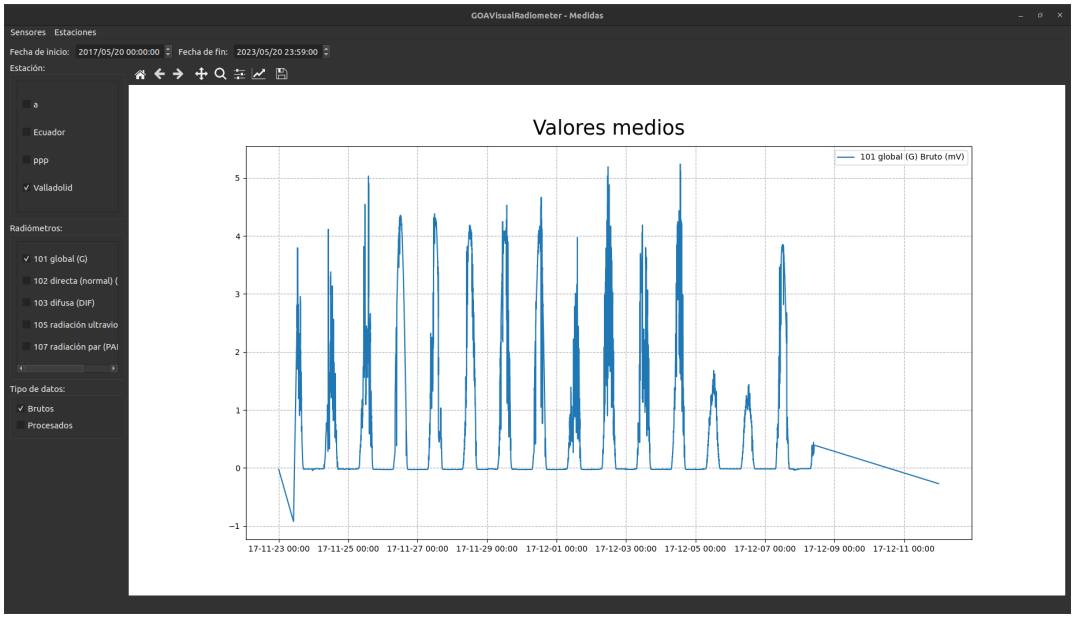

Figura A.5: Pantalla 'Medidas'

Otro elemento que cabe destacar, es que encima de la gráfica contamos con una barra de herramientas que pone a nuestra disposición varias funcionalidades sobre la gráfica. Para ver en mayor detalle para que sirve cada botón, se puede ver una descripción de las mismas en [13].

Finalmente, para acceder al resto de funcionalidades del programa, tenemos un menú en la parte superior izquierda de la pantalla con acceso a las distintas funcionalidades, que en este caso son 'Sensores' y 'Estaciones'; como podemos ver en la Figura A.5.

Sensores

Una vez pulsamos la opción 'Sensores' en la pantalla 'Medidas' (véase Sección A.3.4), se nos presenta un nuevo submenú con distintas funciones relativas a los sensores y en el que cada opción nos lleva a una pantalla distinta que pasaremos a detallar:

- **Añadir sensor:** esta ventana que podemos ver en la Figura A.6 nos permite añadir un nuevo sensor a la base de datos junto con su primer coeficiente de calibración.
- **Visualizar sensor:** esta ventana que podemos ver en la Figura A.7 nos permite visualizar la información de los sensores almacenados en la base de datos.
- **Añadir factor de calibración:** esta ventana que podemos ver en la Figura A.8 nos permite asignar un nuevo coeficiente de calibración a un sensor ya almacenado a partir de una determinada fecha.

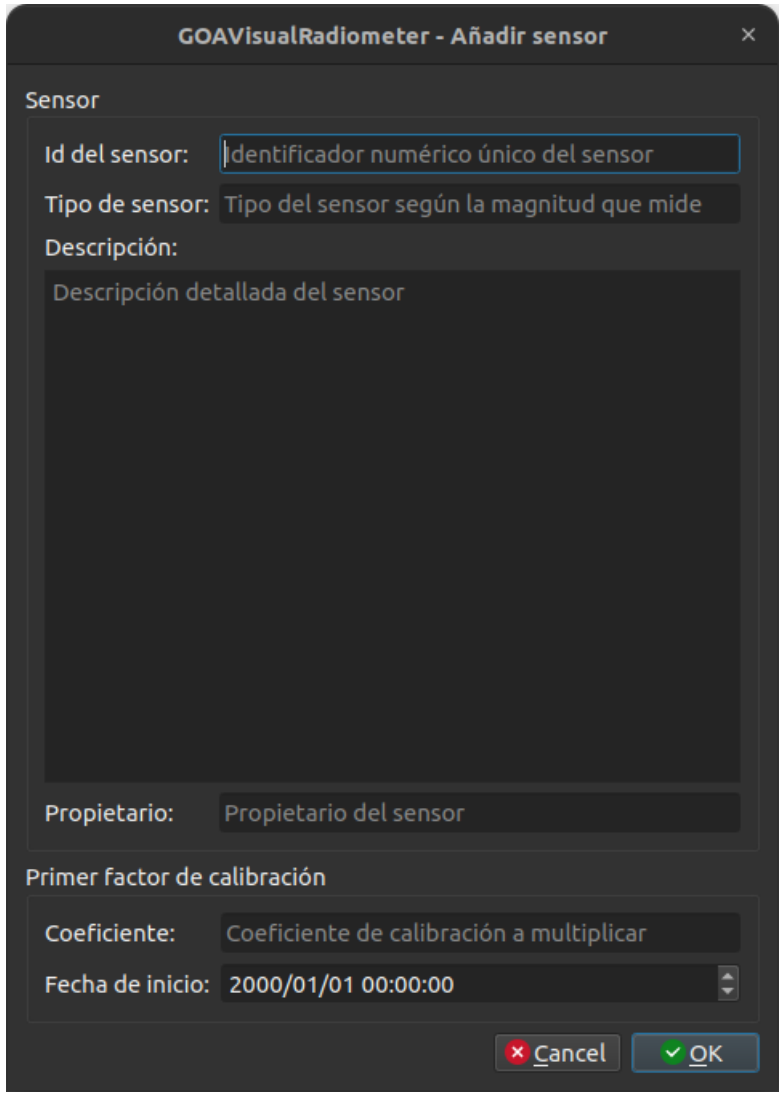

Figura A.6: Pantalla 'Añadir sensor'

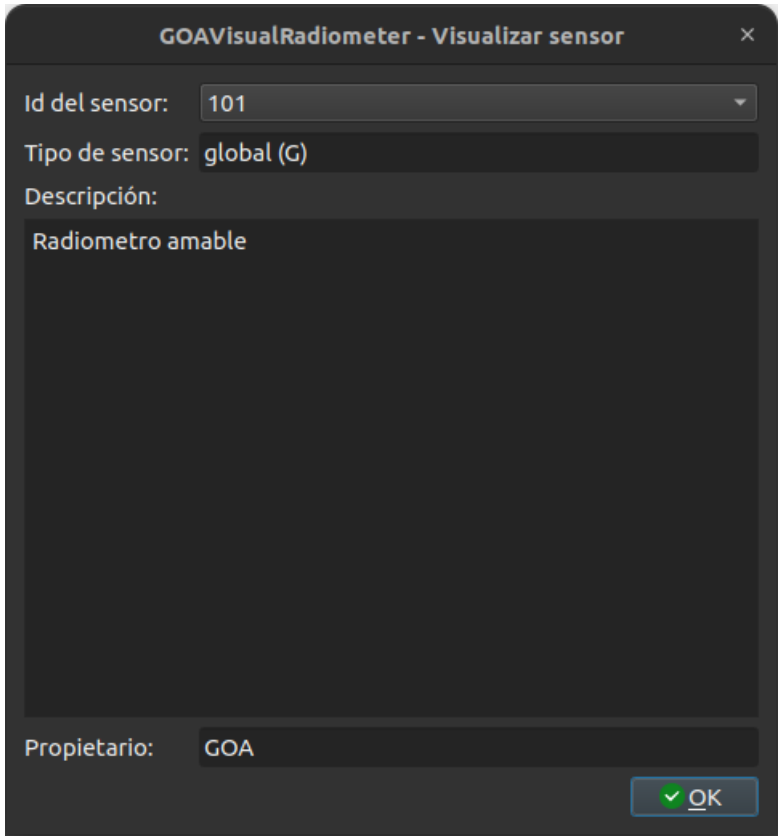

Figura A.7: Pantalla 'Visualizar sensor'

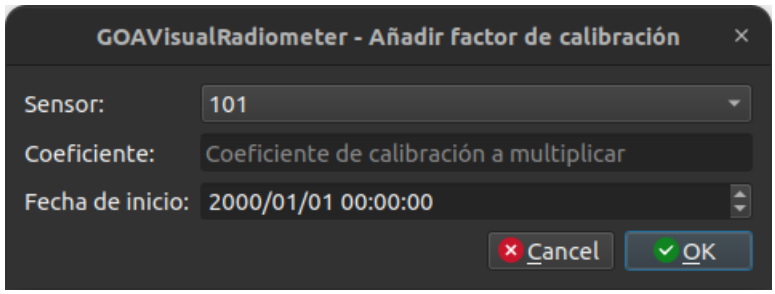

Figura A.8: Pantalla 'Añadir factor de calibración'

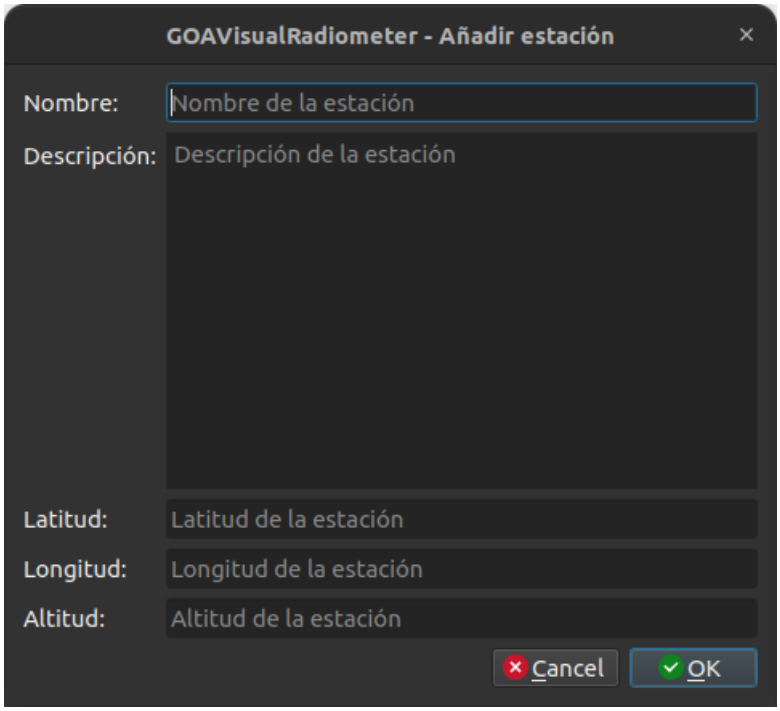

Figura A.9: Pantalla 'Añadir estación'

Estaciones

Una vez pulsamos la opción 'Estaciones' en la pantalla 'Medidas' (véase Sección A.3.4), se nos presenta un nuevo submenú con distintas funciones relativas a las estaciones y en el que cada opción nos lleva a una pantalla distinta que pasaremos a detallar:

- **Añadir estación:** esta ventana que podemos ver en la Figura A.9 nos permite añadir una nueva estación a la base de datos. **Importante:** si quieres que las nuevas estaciones aparezcan en el filtro 'Estaciones:' de la ventana 'Medidas' (véase Sección A.3.4), debes reiniciar el programa.
- **Editar estación:** esta ventana que podemos ver en la Figura A.10 nos permite editar los datos de una estación ya almacenada en la base de datos.
- **Asignar sensor:** esta ventana que podemos ver en la Figura A.11 nos sirve para poder asignar un sensor a una estación a partir de una determinada fecha.

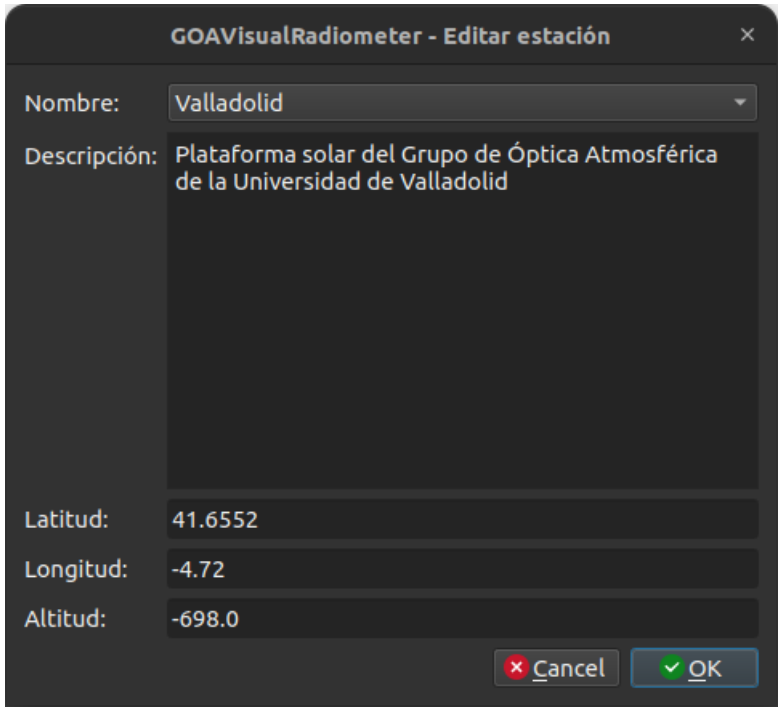

Figura A.10: Pantalla 'Editar estación'

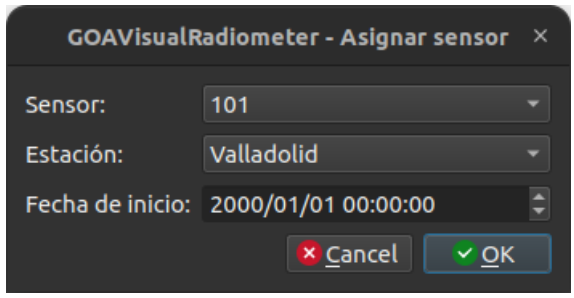

Figura A.11: Pantalla 'Asignar sensor'

Apéndice B

Resumen de enlaces adicionales

Los enlaces útiles de interés en este Trabajo Fin de Grado son:

- Repositorio del código: `https://gitlab.inf.uva.es/jaisaiz/tfggoa-jaime-saiz-losada`.
- Descarga de Python. Accedido el 13/2/2023 (utilizada la versión 3.10.6): `https://www.python.org/downloads/`