Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática (Mención en Tecnologías de la Información)

CÆLIS Viewer: Herramienta para la visualización de datos de la red AERONET

Autor:
Jesús Brezmes Gil-Albarellos

Universidad de Valladolid

Escuela de Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática (Mención en Tecnologías de la Información)

CÆLIS Viewer: Herramienta para la visualización de datos de la red AERONET

Autor:
Jesús Brezmes Gil-Albarellos

Tutores:
Manuel Ángel González Delgado
Ramiro González Catón

Resumen

Dentro del ámbito del cambio climático, el estudio de los aerosoles atmosféricos ha crecido exponencialmente en las últimas décadas debido a su importancia. Es dentro de este marco en el que el Grupo de Óptica Atmosférica de la Universidad de Valladolid (GOA-UVa) centra sus líneas de investigación. La red mas importante de esta línea de estudio es la red AERONET, red fundada por NASA y que junto a la propia NASA y a la Universidad de Lille (Francia) el GOA-UVa gestiona y calibra una parte considerable de sus equipos. Es para esta parte de gestión de la red por parte del GOA-UVa para la que se desarrolla esta aplicación.

Se trata de una herramienta intuitiva de visualización de los datos generados por dicha red, que de una forma simple ofrece acceso a una cantidad importante de información muy útil y valiosa para la gestión y mantenimiento de la red. La herramienta consulta los datos directamente de la base de datos CAELIS, propiedad del propio GOA-UVa, y los muestra de acuerdo a ciertos estándares para que sean accesibles por cualquier usuario, independientemente de su nivel de conocimiento de la base de datos.

Abstract

Within the scope of climate change, the study of atmospheric aerosols has grown exponentially in recent decades due to its importance. It is within this framework that the Atmospheric Optics Group of the University of Valladolid (GOA-UVa) focuses its lines of research. The most important network in this line of study is the AERONET network, a network founded by NASA and which, together with NASA itself and the University of Lille (France), the GOA-UVa manages and calibrates a considerable part of its equipment. It is for this part of network management by the GOA-UVa for which this application is developed.

It is an intuitive tool to visualize the data generated by said network, which in a simple way offers access to an important quantity of very useful and valuable information for the management and maintenance of the network. The tool consults the data directly from the CAELIS database, owned by the GOA-UVa itself, and displays it according to certain standards so that it is accessible by any user, regardless of their level of knowledge of the database.

Índice de Contenidos

Índice de figuras

Capítulo 1

Introducción

Resumen: *En este capítulo se hace una introducción al marco general en el que se va a desarrollar el proyecto, así como la exposición de los objetivos principales de este.*

1.1 Marco de trabajo

Este proyecto está enmarcado dentro del ámbito de la red AERONET la cual se encarga del estudio de aerosoles atmosféricos. Se entiende como aerosol atmosférico a cualquier partícula, ya sea en estado sólido o líquido, en suspensión en la atmósfera el tiempo suficiente para ser medido.

El proyecto AERONET (AErosol RObotic NETwork) es una federación de redes remotas de detección de aerosoles fundada por la NASA y PHOTONS (PHOtométrie pour le Traitement Opérationnel de Normalization Satellitaire; Universidad De Lille). Esta red nace con el objetivo de validar las medidas satelitales, pero con el paso del tiempo, y la comprensión de la influencia que estos aerosoles tienen en el cambio climático sus datos se han visto ampliamente usados por otras redes y colaboradores de diferentes agencias nacionales (ESA, EUMESAT…), institutos de investigación, Universidad, ...

Una de las características más llamativas de esta red es que ha proporcionado durante más de 25 años una base de datos de dominio público y a largo plazo, la cual es fácilmente accesible. En esta base de datos encontramos información sobre el espesor óptico de aerosoles (AOD por sus siglas en ingles: Aerosol Optical Depth), que es una estimación de la cantidad de aerosol presente en la atmósfera, así como de otras propiedades microfísicas y ópticas de estos aerosoles, lo que permite la investigación y caracterización de estos.

Figura 1.1: Página oficial de la red AERONET.

AERONET permite la sinergia con otras bases de datos de ámbito mundial debido a su estandarización sobre los instrumentos, calibración, procesamiento y distribución, lo cual permite que medidas realizadas en distintos puntos del planeta con instrumentos distintos sean directamente comparables entre ellas.

Pero ¿qué son estos instrumentos de detección remota para la medida de los aerosoles? El instrumento central de la red AERONET es el CIMEL CE318. Se trata de un fotómetro que mide la iradiancia solar, o lunar, y la radiancia del cielo en 8 o 10 bandas espectrales (dependiendo del modelo). La ubicación de esas bandas es fija y atiende a criterios para excluir bandas de absorción de gases presentes en la atmósfera mas inestables, a excepción del vapor de agua. Estas bandas están distribuidas en el espectro visible, ultravioleta e infrarrojo y tienen un ancho de banda FWHM (FullFull Width Half Maximum) de 10 nanómetros en las bandas centrales, en el ultravioleta su ancho es de 2 nm y en el infrarrojo de 25. Alguna de sus cualidades más destacadas son la capacidad de almacenamiento de datos y un sistema automático de tracking para un seguimiento preciso del posicionamiento del sol o de la luna.

Figura 1.2: Fotómetro solar, GOA-UVA.

Durante el primer congreso de AERONET se estipuló que era imposible seguir manteniendo una estructura centralizada de la red AERONET/PHOTONS (NASA y Universidad de Lille respectivamente). Esto hizo que la red se dividiera en subredes federadas para descentralizar las labores de calibración y mantenimiento de todos los instrumentos que componen la red, aunque siempre cumpliendo con los estándares de AERONET.

De este modo el Grupo de Óptica Atmosférica de la Universidad de Valladolid (GOA-UVa) encabeza una propuesta, en la que inicialmente colaboran otras universidades españolas, para crear un centro de calibración de la red AERONET en las instalaciones del GOA-Uva. Inicialmente el objetivo de esta subred era la calibración y mantenimiento de los equipos prestados por estas Universidades Españolas, pero con el paso del tiempo, y debido al gran incremento de estaciones que ha sufrido AERONET, el GOA-UVa pasó a gestionar equipos de estaciones de prácticamente todo el mundo.

Figura 1.3: Pagina oficial GOA-UVA.

El GOA-UVA es un grupo de investigación fundamentalmente centrado en el estudio de la atmósfera con capacidad operativa sobre la red AERONET. Fue creado en 1996 por los Profs. Ángel de Frutos y Victoria Cachorro. Posee una organización interdepartamental, en la que hay miembros de los siguientes departamentos de la Universidad: «Física Teoría, Atómica y Óptica» (departamento anfitrión del grupo), «Física Aplicada» y «Didáctica de las Ciencias Experimentales».

1.2 Objetivos

El objetivo principal de este TFG es la replicación de una herramienta existente encargada de dar servicios de visualización y graficación de los datos recogidos por los fotómetros. Dichos datos se encuentran almacenados en CÆLIS, la cual es una herramienta completamente desarrollada por el GOA-UVa. Dentro de este objetivo podemos englobar la realización del proyecto en dos tareas principales: “Visualización del uso de los fotómetros asignados al GOA-UVA” y “Graficación de las medidas recogidas por los distintos fotómetros”.

- Visualización del uso de los fotómetros asignados al GOA-UVA:

El principal objetivo de esta tarea es visualizar el uso de cada fotómetro en una línea temporal, es decir cuando ha estado activo y recogiendo medidas. Además, debe ser interactivo y navegable respecto a una línea temporal. Se debe poder aplicar determinados filtros para la búsqueda de los diferentes fotómetros.

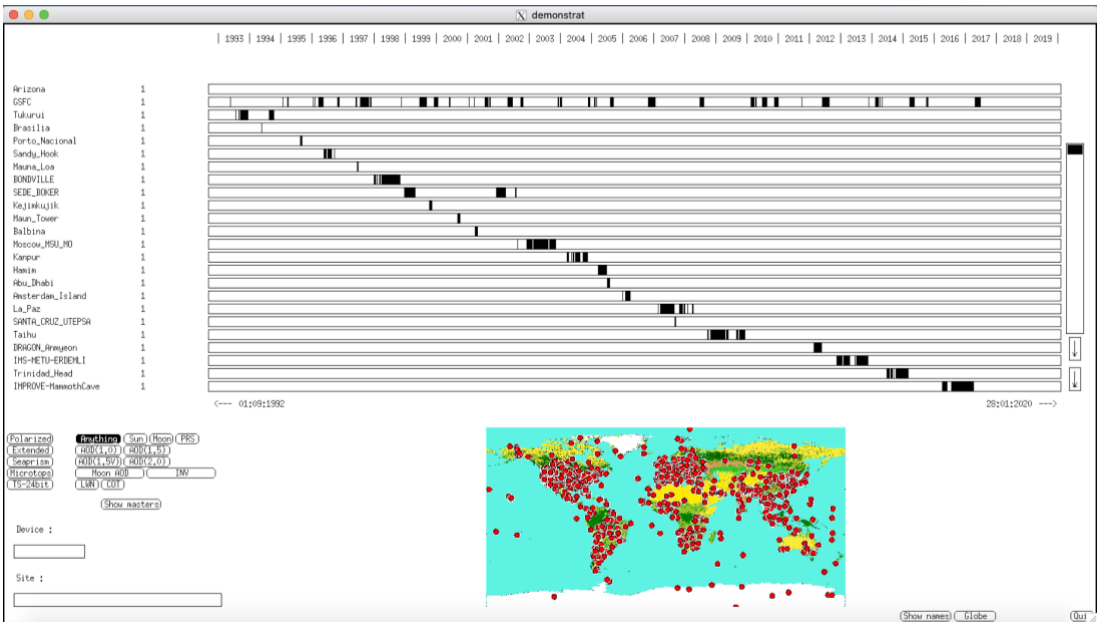


Figura 1.4: Ventana de Uso de Fotómetros de la herramienta a replicar.

- Graficación de las medidas recogidas por los distintos fotómetros:

El objetivo de esta segunda tarea es poder visualizar las diferentes medidas recogidas por cada fotómetro, pudiendo ser navegable e interactiva respecto a una línea temporal. A continuación, se muestran cada una de las medidas que se van a tener en cuenta:

  - Temperatura: se refiere a la temperatura, en el interior de la cabeza sensora del fotómetro, en el momento de la medida. Esto se utiliza porque el fotómetro CIMEL CE318 no tiene los detectores estabilizados en temperatura, por lo tanto, es necesario corregir las medidas en función de la temperatura para obtener unas medidas lo más exactas posibles.

  - Vapor de agua: Se trata de un indicador de la cantidad de vapor de agua precipitable en la columna de la atmósfera.

  - WExp (este parámetro se llama realmente alfa de Angström): es una dependencia espectral que relaciona el tamaño de las partículas con la longitud de onda incidente, este parámetro sirve para caracterizar el aerosol atmosférico y determinar su tipo.

- AOD: Espesor óptico de aerosoles (por sus siglas en inglés aerosol optical depth) es un indicador de la cantidad de aerosoles en la columna de la atmósfera.

- PWR: Es la pendiente de la curva que se forma al medir la radiancia según te vas alejando del disco solar. Este parámetro es determinante para conocer la presencia de nubes altas en la atmósfera (cirros) que puedan interferir en la medida de los aerosoles.

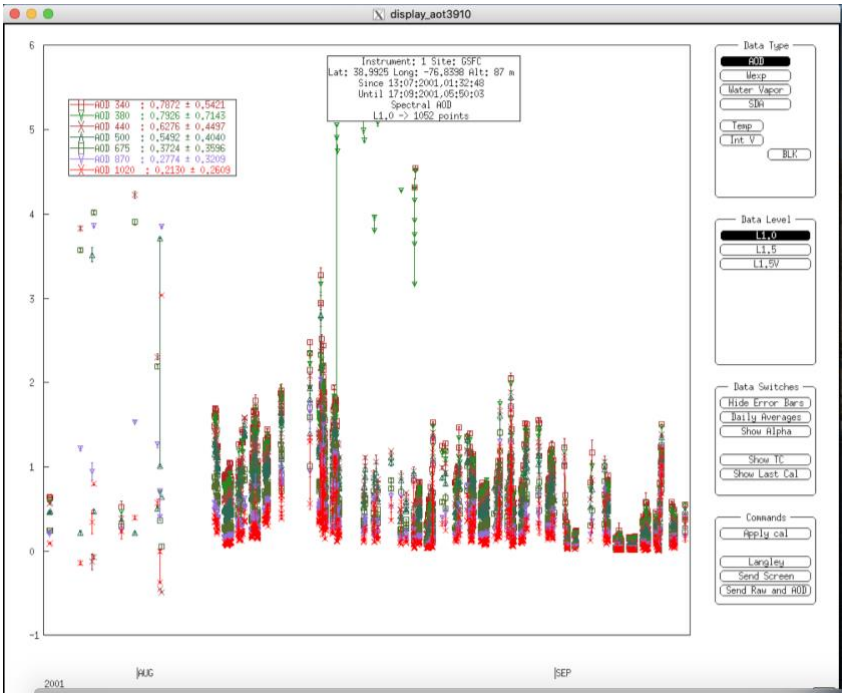


Figura 1.5: Ventana de Graficación de las medidas de la herramienta a replicar.

Aún siendo estas dos tareas prácticamente individuales, uno de los objetivos finales es su unión. A partir del fotómetro y la fecha, seleccionada durante la tarea de visualización, se generará la tarea de graficación de las medidas.

Capítulo 2

Plan de desarrollo del proyecto

Resumen: *En este capítulo se expone la planificación que se ha seguido para la elaboración del proyecto, así como los problemas que han surgido y han podido alterar dicha planificación.*

2.1 Plan de Desarrollo Software

Se llama plan de desarrollo software al documento que gestiona un proyecto, definiendo para ellos las técnicas, actividades y tareas necesarias para concluir satisfactoriamente dicho proyecto. En esta sección se pretende dar a exponer los elementos básicos de un proyecto y de esta manera detallar los aspectos básicos tal y como han sido entendidos y formulados.

2.1.1 Visión General

Como se ha visto en los objetivos propuestos para el desarrollo del proyecto, existen dos tareas diferenciadas que en un final formarán un conjunto. Es por eso por lo que su realización se llevará a cabo de forma paralela y conjunta durante la ejecución del plan.

En este punto se describe la metodología utilizada para la realización del proyecto. Tras investigar diversos métodos de trabajo, se decide tomar como referencia el UPEDU (Unified Process for Education). Este tipo de proceso de desarrollo software es iterativo y una implementación del modelo en espiral. Se divide en el tiempo en cuatro fases bien definidas:

- Fase de inicio: El objetivo principal es determinar el alcance del sistema adecuadamente como base para validar los costos iniciales y los presupuestos. Para ello se deben realizar las siguientes tareas:
    - Comprensión del problema.
    - Análisis de los requisitos, riesgos y definición del modelo de casos de uso del sistema.
    - Realizar estimaciones iniciales de planificación y costes.
    - Obtener una arquitectura candidata.

Los artefactos para entregar en esta fase serán:

- Lista de Riesgos
- Especificación de Requisitos Software
- Especificación de Casos de Uso
- Plan de Iteración Inicial
- Plan de Iteración de Elaboración
- Plan de Desarrollo Software
- Documento de Seguimiento del Proyecto

- Fase de elaboración: El objetivo principal es mitigar los elementos clave de riesgo identificados por análisis hasta el final de esta fase. La fase de elaboración es donde el proyecto comienza a tomar forma. Para ello se deben realizar las siguientes tareas:
    - Definición de los diagramas de clases, el modelo de datos e implementación de la interfaz de usuario y de los casos de uso más importantes, consiguiendo así versiones de prueba ya utilizables y reduciendo los costes de desarrollo.
    - Revisión de los requisitos.

Los artefactos para entregar en esta fase serán:

- Documento General de Análisis
- Documento de Arquitectura Software
- Realización de Casos de Uso
- Plan de Pruebas
- Casos de Prueba
- Plan de Iteración de Construcción

- Fase de construcción: El objetivo principal es construir el sistema de software. En esta fase, el foco principal está en el desarrollo de componentes y otras características del sistema. Para ello se deben realizar las siguientes tareas:

  - Añadir el total de funcionalidades al prototipo considerando los nuevos requisitos surgidos de la revisión.
  - Elaborar el plan de pruebas.

Los artefactos para entregar en esta fase serán:

- Modelo de Implementación
- Plan de Iteración de Transición
- Aplicación Finalo

- Fase de transición: El objetivo principal es 'transitar' el sistema desde el desarrollo hasta la producción, haciéndolo disponible y entendido por el usuario final. Para ello se deben realizar las siguientes tareas:

  - Entrega de la documentación al cliente.
  - Conseguir que el cliente final acepte el producto y sea capaz de utilizarlo y entenderlo.
  - Prueba del sistema en el entorno de explotación.

Los artefactos para entregar en esta fase serán:

- Manual de Usuario.
- Manual de Instalación

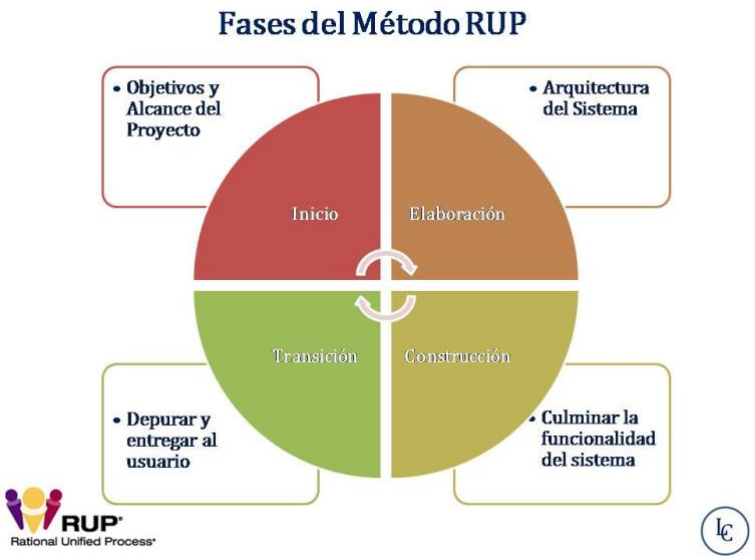


Figura 2.1: Fases del Método RUP.

2.1.2 Plan de iteración de la etapa de Inicio

El objetivo principal de esta fase es saber el problema al que nos enfrentamos, identificar los recursos necesarios para poder llevar a cabo el proyecto y definir una planificación completa del trabajo que se va a llevar a cabo. Además, se va a desarrollar en dos iteraciones, en una primera se hará un acercamiento al problema propuesto, a la tecnología y herramientas exigidas para la realización del proyecto y se comenzará con el análisis. En la segunda iteración se procederá a continuar con el análisis en función del trabajo desarrollado durante la primera iteración.

Cuando se termine esta fase tienen que estar perfectamente definidos los requisitos del proyecto, los casos de usos principales y su consiguiente análisis de riesgos. Lo fundamental es saber el trabajo que hay que realizar y cómo llevarlo a cabo.

En la Figura 2.2 se puede apreciar la planificación de esta primera etapa, junto con la Figura 2.3 la cual es su correspondiente diagrama de Gantt.

Nº	Título	Duración	Comienzo	Fin	Predecesores
0	**▼ Plan de desarrollo**		**4 mar 2020**	**4 jun 2020**	
1	**▼ Inicio**		**4 mar 2020**	**2 abr 2020**	
2	**▼ Primera Iteración**	**12 días**	**4 mar 2020**	**16 mar 2020**	
3	Plan Iteración Inicio	3 días	4 mar 2020	6 mar 2020	
4	Conocimientos técnicos del Software	2 días	6 mar 2020	7 mar 2020	3
5	Plan de desarrollo Software	2 días	9 mar 2020	10 mar 2020	4
6	Definición de Riesgos Software	2 días	11 mar 2020	12 mar 2020	5
7	Recogida de Requisitos	2 días	12 mar 2020	13 mar 2020	6
8	Primera Versión Casos de uso	2 días	14 mar 2020	16 mar 2020	7
9	**▼ Segunda Iteración**	**9 días**	**16 mar 2020**	**24 mar 2020**	**2**
10	Lista de Riesgos Software	1 día	16 mar 2020	17 mar 2020	8
11	Especificación de Requisitos Software	3 días	18 mar 2020	20 mar 2020	10
12	Especificación de Casos de Uso	2 días	20 mar 2020	21 mar 2020	11
13	Plan de Pruebas	2 días	21 mar 2020	22 mar 2020	12
14	Plan de iteración de la Evolución	1 día	23 mar 2020	23 mar 2020	13
15	Informe de Seguimiento	1 día	24 mar 2020	24 mar 2020	14

Figura 2.2: Duración de las tareas de la fase Inicio.

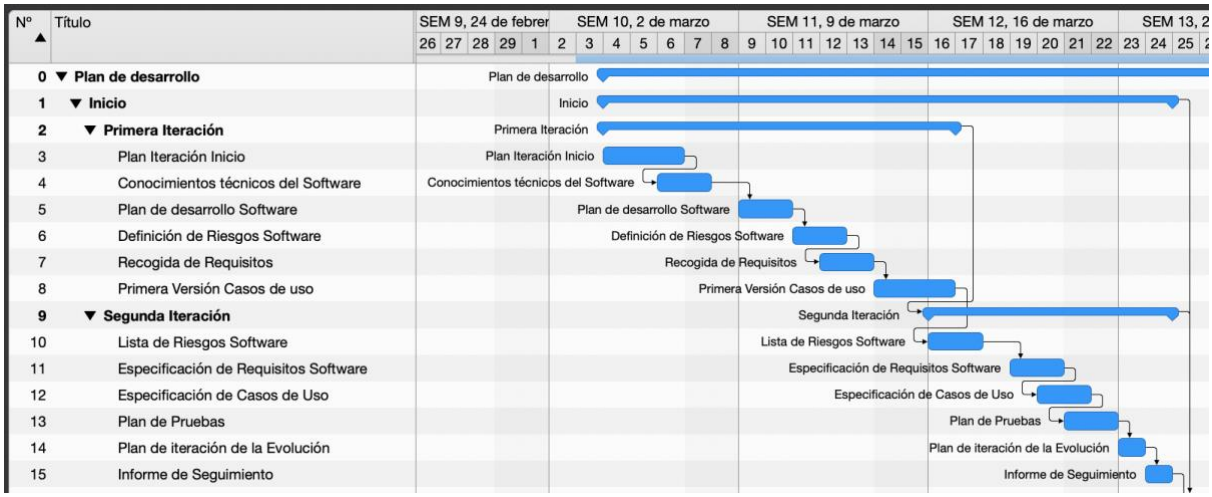


Figura 2.3: Diagrama de Gantt de la fase de Inicio.

2.1.3 Plan de iteración de la etapa de Elaboración

Cuando se termine esta fase, se tiene que haber obtenido la mayor parte del análisis y un primer prototipo de las interfaces de usuario, así como la implementación de los casos de uso más importantes. Esta fase también estará dividida en dos iteraciones, en una primera se revisará el contenido de análisis realizado en la segunda iteración de la fase de Inicio y se procederá a realizar el diseño inicial del Sistema. En la segunda iteración se trabajará para recoger todo el análisis realizado en anteriores iteraciones y realizar el prototipo inicial.

En la Figura 2.4 se puede apreciar la planificación de esta etapa de Elaboración, junto con la Figura 2.5 la cual es su correspondiente diagrama de Gantt.

16	**▼ Elaboración**	**31 días**	**25 mar 2020**	**25 abr 2020**	**1**
17	**▼ Primera Iteración**	**17 días**	**25 mar 2020**	**11 abr 2020**	**9**
18	Revisión de Requisitos	3 días	25 mar 2020	27 mar 2020	15
19	Revisión de Modelo de Casos de Uso	2 días	27 mar 2020	28 mar 2020	18
20	Diagrama de Análisis	4 días	30 mar 2020	2 abr 2020	19
21	Diseño inicial del Sistema	4 días	3 abr 2020	6 abr 2020	20
22	Diagramas de Diseño	3 días	6 abr 2020	8 abr 2020	21
23	Definición de la Arquitectura básica	2 días	9 abr 2020	10 abr 2020	22
24	Plan de la Segunda Iteración	1 día	11 abr 2020	11 abr 2020	23
25	**▼ Segunda Iteración**	**14 días**	**13 abr 2020**	**25 abr 2020**	**17**
26	Documento de Análisis del Proyecto	4 días	13 abr 2020	16 abr 2020	24
27	Documento de Diseño del Sistema	4 días	17 abr 2020	20 abr 2020	26
28	Documento de Arquitectura Software	4 días	21 abr 2020	24 abr 2020	27
29	Plan de la siguiente Iteración	1 día	24 abr 2020	24 abr 2020	28
30	Informe de Seguimiento	1 día	25 abr 2020	25 abr 2020	29
31	Realización del prototipo inicial	31 días	25 mar 2020	4 abr 2020	15

Figura 2.4: Duración de las tareas de la fase de Elaboración.

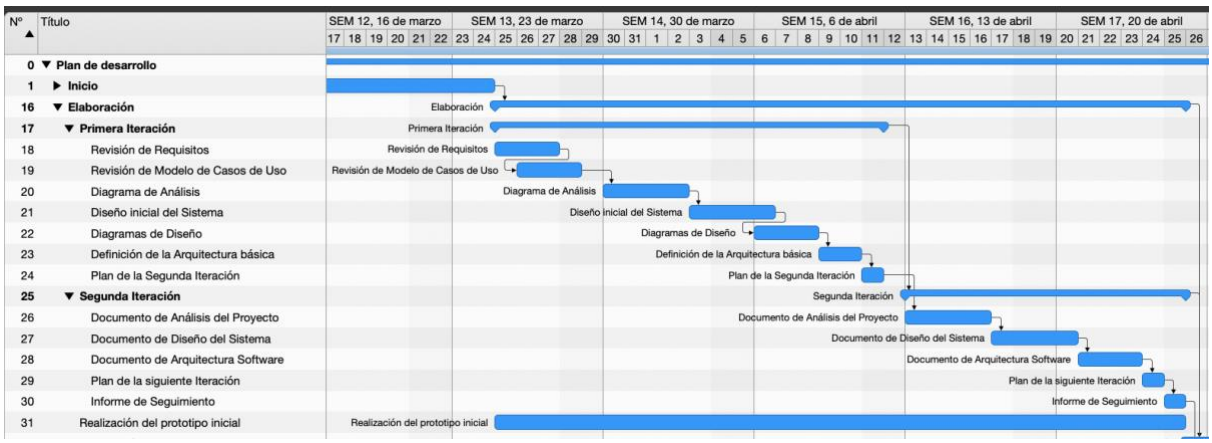


Figura 2.5: Diagrama de Gantt de la fase de Elaboración.

2.1.4 Plan de iteración de la etapa de Construcción

El objetivo de esta fase es construir todas las funcionalidades de la aplicación. El producto final se desarrolla en dos iteraciones como en las fases anteriores, haciendo las pruebas correspondientes después de cada una de ellas. Al final de esta fase se han añadido todas las funcionalidades al prototipo anterior, añadiendo los nuevos requisitos que hayan podido surgir en las revisiones anteriores.

En la Figura 2.6 se puede apreciar la planificación de esta etapa de Construcción, junto con la Figura 2.7 la cual es su correspondiente diagrama de Gantt.

32	**▼ Construcción**	**30 días**	**26 abr 2020**	**26 may 2020**
33	**▼ Primera Iteración**	**16 días**	**26 abr 2020**	**12 may 2020**
34	Revisión del documento de Análisis	1 día	26 abr 2020	26 abr 2020
35	Revisión del documento de Diseño	1 día	27 abr 2020	27 abr 2020
36	Revisión del documento de arquitectura Sw.	1 día	28 abr 2020	28 abr 2020
37	Modelo inicial de Implementación	9 días	29 abr 2020	7 may 2020
38	Realización de Pruebas	4 días	8 may 2020	12 may 2020
39	**▼ Segunda Iteración**	**14 días**	**13 may 2020**	**26 may 2020**
40	Modelo Final de Implementación	8 días	13 may 2020	20 may 2020
41	Realización de Pruebas	4 días	21 may 2020	24 may 2020
42	Plan de la siguiente Iteración	1 día	25 may 2020	25 may 2020
43	Informe de Seguimiento	1 día	26 may 2020	26 may 2020
44	Implementación	31 días	26 abr 2020	26 may 2020

Figura 2.6: Duración de las tareas de la fase de Construcción.

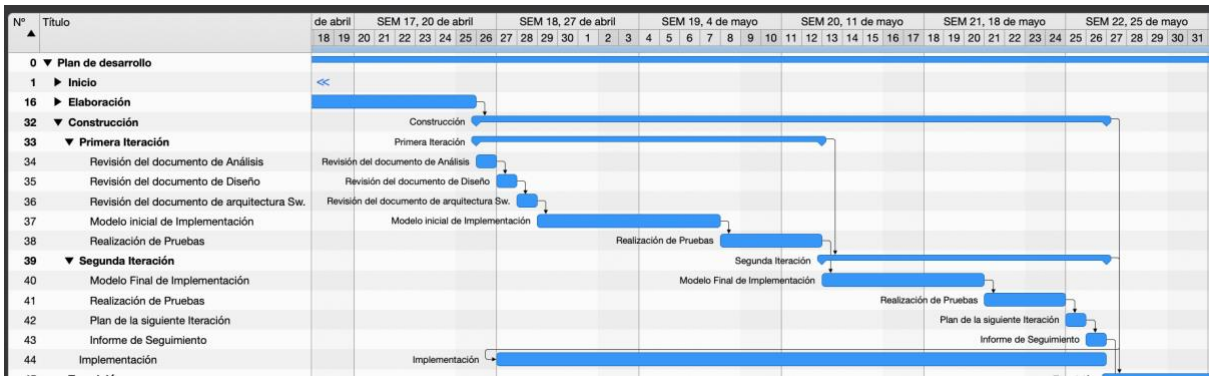


Figura 2.7: Diagrama de Gantt de la fase de Construcción.

2.1.5 Plan de iteración de la etapa de Transición

Durante la fase de transición se le entregará al cliente el manual de usuario y el manual de instalación del sistema. Debido a la situación de confinamiento por los acontecimientos relacionados con el SARS-COV-2, se realizará de forma electrónica.

En la Figura 2.8 se puede apreciar la planificación de esta última etapa, junto con la Figura 2.9 la cual es su correspondiente diagrama de Gant.

45	**▼ Transición**	**8 días**	**27 may 2020**	**4 jun 2020**
46	Resultado de Pruebas	3 días	27 may 2020	29 may 2020
47	Manual de Usuario	2 días	29 may 2020	30 may 2020
48	Manual de Instalación	2 días	1 jun 2020	2 jun 2020
49	Informe de seguimiento	2 días	3 jun 2020	4 jun 2020

Figura 2.8: Duración de las tareas de la fase de Construcción.

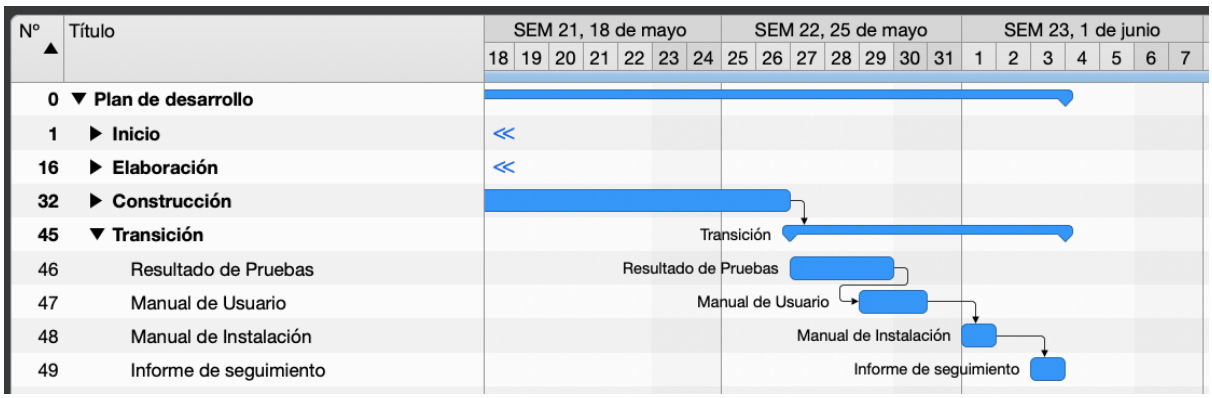


Figura 2.9: Diagrama de Gantt de la fase de Construcción.

2.1.6 Visión general de la Planificación

En este apartado se mostrará una visión general de la planificación realizada, centrándose en especial en la duración de las iteraciones y de las fases a las que pertenecen. En la Figura 2.10 podemos observar la planificación general de las etapas y la duración de sus iteraciones.

Nº	Título	Duración	Comienzo	Fin	Predecesores
0	**▼ Plan de desarrollo**		**4 mar 2020**	**4 jun 2020**	
1	**▼ Inicio**		**4 mar 2020**	**2 abr 2020**	
2	**▶ Primera Iteración**	**12 días**	**4 mar 2020**	**16 mar 2020**	
9	**▶ Segunda Iteración**	**9 días**	**16 mar 2020**	**24 mar 2020**	**2**
16	**▼ Elaboración**	**31 días**	**25 mar 2020**	**25 abr 2020**	**1**
17	**▶ Primera Iteración**	**17 días**	**25 mar 2020**	**11 abr 2020**	**9**
25	**▶ Segunda Iteración**	**14 días**	**13 abr 2020**	**25 abr 2020**	**17**
31	Realización del prototipo inicial	31 días	25 mar 2020	4 abr 2020	15
32	**▼ Construcción**	**30 días**	**26 abr 2020**	**26 may 2020**	**16**
33	**▶ Primera Iteración**	**16 días**	**26 abr 2020**	**12 may 2020**	**25**
39	**▶ Segunda Iteración**	**14 días**	**13 may 2020**	**26 may 2020**	**33**
44	Implementación	31 días	26 abr 2020	26 may 2020	39
45	**▶ Transición**	**8 días**	**27 may 2020**	**4 jun 2020**	**32**

Figura 2.10: Duración de las diferentes Iteraciones.

En la Figura 2.11 se observa el diagrama de Gantt correspondiente de la planificación de la Figura 2.10. Y por último la Figura 2.12 representa la duración global de cada una de las tareas.

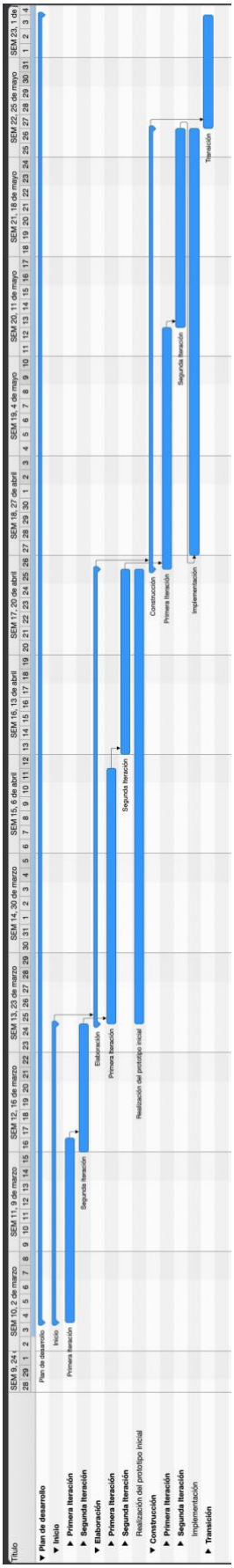


Figura 2.11: Diagrama de Gantt de las diferentes Iteraciones.

Nº	Título	Duración	Comienzo	Fin	Predecesores
0	**▼ Plan de desarrollo**		**4 mar 2020**	**4 jun 2020**	
1	**▼ Inicio**		**4 mar 2020**	**2 abr 2020**	
2	**▼ Primera Iteración**	**12 días**	**4 mar 2020**	**16 mar 2020**	
3	Plan Iteración Inicio	3 días	4 mar 2020	6 mar 2020	
4	Conocimientos técnicos del Software	2 días	6 mar 2020	7 mar 2020	3
5	Plan de desarrollo Software	2 días	9 mar 2020	10 mar 2020	4
6	Definición de Riesgos Software	2 días	11 mar 2020	12 mar 2020	5
7	Recogida de Requisitos	2 días	12 mar 2020	13 mar 2020	6
8	Primera Versión Casos de uso	2 días	14 mar 2020	16 mar 2020	7
9	**▼ Segunda Iteración**	**9 días**	**16 mar 2020**	**24 mar 2020**	**2**
10	Lista de Riesgos Software	1 día	16 mar 2020	17 mar 2020	8
11	Especificación de Requisitos Software	3 días	18 mar 2020	20 mar 2020	10
12	Especificación de Casos de Uso	2 días	20 mar 2020	21 mar 2020	11
13	Plan de Pruebas	2 días	21 mar 2020	22 mar 2020	12
14	Plan de iteración de la Evolución	1 día	23 mar 2020	23 mar 2020	13
15	Informe de Seguimiento	1 día	24 mar 2020	24 mar 2020	14
16	**▼ Elaboración**	**31 días**	**25 mar 2020**	**25 abr 2020**	**1**
17	**▼ Primera Iteración**	**17 días**	**25 mar 2020**	**11 abr 2020**	**9**
18	Revisión de Requisitos	3 días	25 mar 2020	27 mar 2020	15
19	Revisión de Modelo de Casos de Uso	2 días	27 mar 2020	28 mar 2020	18
20	Diagrama de Análisis	4 días	30 mar 2020	2 abr 2020	19
21	Diseño inicial del Sistema	4 días	3 abr 2020	6 abr 2020	20
22	Diagramas de Diseño	3 días	6 abr 2020	8 abr 2020	21
23	Definición de la Arquitectura básica	2 días	9 abr 2020	10 abr 2020	22
24	Plan de la Segunda Iteración	1 día	11 abr 2020	11 abr 2020	23
25	**▼ Segunda Iteración**	**14 días**	**13 abr 2020**	**25 abr 2020**	**17**
26	Documento de Análisis del Proyecto	4 días	13 abr 2020	16 abr 2020	24
27	Documento de Diseño del Sistema	4 días	17 abr 2020	20 abr 2020	26
28	Documento de Arquitectura Software	4 días	21 abr 2020	24 abr 2020	27
29	Plan de la siguiente Iteración	1 día	24 abr 2020	24 abr 2020	28
30	Informe de Seguimiento	1 día	25 abr 2020	25 abr 2020	29
31	Realización del prototipo inicial	31 días	25 mar 2020	4 abr 2020	15
32	**▼ Construcción**	**30 días**	**26 abr 2020**	**26 may 2020**	**16**
33	**▼ Primera Iteración**	**16 días**	**26 abr 2020**	**12 may 2020**	**25**
34	Revisión del documento de Análisis	1 día	26 abr 2020	26 abr 2020	30
35	Revisión del documento de Diseño	1 día	27 abr 2020	27 abr 2020	34
36	Revisión del documento de arquitectura Sw.	1 día	28 abr 2020	28 abr 2020	35
37	Modelo inicial de Implementación	9 días	29 abr 2020	7 may 2020	36
38	Realización de Pruebas	4 días	8 may 2020	12 may 2020	37
39	**▼ Segunda Iteración**	**14 días**	**13 may 2020**	**26 may 2020**	**33**
40	Modelo Final de Implementación	8 días	13 may 2020	20 may 2020	38
41	Realización de Pruebas	4 días	21 may 2020	24 may 2020	40
42	Plan de la siguiente Iteración	1 día	25 may 2020	25 may 2020	41
43	Informe de Seguimiento	1 día	26 may 2020	26 may 2020	42
44	Implementación	31 días	26 abr 2020	26 may 2020	39
45	**▼ Transición**	**8 días**	**27 may 2020**	**4 jun 2020**	**32**
46	Resultado de Pruebas	3 días	27 may 2020	29 may 2020	43
47	Manual de Usuario	2 días	29 may 2020	30 may 2020	46
48	Manual de Instalación	2 días	1 jun 2020	2 jun 2020	47
49	Informe de seguimiento	2 días	3 jun 2020	4 jun 2020	48

Figura 2.12: Duración de cada una de las actividades.

2.2 Seguimiento del Proyecto

En este punto se expone la evolución real del proyecto, comparándola con la propuesta en la planificación. Se indican también los motivos por los cuales se ha modificado la planificación, en caso de haberse producido.

2.2.1 Evolución del Proyecto

El proyecto tenía una planificación estimada del 4 de marzo de 2020 al 4 de junio de 2020. Esto se debe a que TFG es equivalente a 12 ECTS, y cada ECTS corresponde con 25 horas de trabajo. Es decir, se dispone de 300 horas a desarrollar durante 3 meses. El proyecto ha sufrido retrasos en diferentes partes de su planificación, por lo que no se ha terminado en la fecha establecida.

Esta demora se debe, en parte, a que no ha sido posible disponer de todo el tiempo estimado en la planificación. Esto se debe en gran parte a la ya conocida pandemia, el SARS-COV-2, y al estado de alarma que ha impedido realizar algunas tareas en la universidad. Por otra parte, también se tuvo en cuenta en la planificación un tiempo de aprendizaje de la instrumentación y de las diferentes herramientas algo escaso. Por otra parte, el tiempo atrasado se recupera mejor debido a la situación de confinamiento, es decir, no afecta al resto de etapas.

Como consecuencia de estos retrasos la duración de cada fase se ha visto modificada a la siguiente, Figura 2.

1	▼ **Inicio**	**27 días**	**4 mar 2020**	**30 mar 2020**
2	▶ **Primera Iteración**	**15 días**	**4 mar 2020**	**19 mar 2020**
9	▶ **Segunda Iteración**	**12 días**	**19 mar 2020**	**30 mar 2020**
16	▼ **Elaboración**	**31 días**	**30 mar 2020**	**30 abr 2020**
17	▶ **Primera Iteración**	**17 días**	**30 mar 2020**	**16 abr 2020**
25	▶ **Segunda Iteración**	**14 días**	**16 abr 2020**	**30 abr 2020**
31	Realización del prototipo inicial	31 días	30 mar 2020	30 abr 2020
32	▼ **Construcción**	**30 días**	**30 abr 2020**	**30 may 2020**
33	▶ **Primera Iteración**	**16 días**	**30 abr 2020**	**16 may 2020**
39	▶ **Segunda Iteración**	**14 días**	**17 may 2020**	**30 may 2020**
44	Implementación	31 días	30 abr 2020	30 may 2020
45	▶ **Transición**	**8 días**	**30 may 2020**	**8 jun 2020**

Figura 2.13: Duración real de cada una de las actividades.

A continuación, se mostrará la diferencia de tiempos entre la planificación y la duración rea, Figura 2.

Fase	Duración estimada (días)	Duración real (días)
Inicio	21	27
Elaboración	31	31
Construcción	30	30
Transición	8	8

Figura 2.14: Diferencia de tiempos entre la planificación y la duración real.

2.3 Organización del Proyecto

2.3.1 Interfaces Externas

La interfaz externa de la aplicación planteada, junto con el resto del proyecto, será desarrollada en Python 3. Este lenguaje de programación interpretado está centrado en la legibilidad de su código, además que es una herramienta muy potente y con mucha proyección en la actualidad. Volviendo a la interfaz, esta va a ser construida con PyQt, cuya versión más optimizada es la 5.

PyQt es un binding de la biblioteca gráfica Qt para el lenguaje de programación Python. Qt es un framework multiplataforma que se utiliza para el desarrollo de aplicaciones que utilicen una interfaz gráfica de usuario. Es un software libre y de código abierto donde participa tanto la comunidad de usuarios, como desarrolladores de Nokia, Digia y otras empresas.

PyQt además nos proporciona una herramienta llamada Qt Designer, la cual estará descrita en el apartado de herramientas utilizadas (Apartado 2.5), que nos permite probar las diferentes interfaces de forma interactiva y gráfica. En el Anexo I podemos ver como transformar dichas interfaces a código python de forma sencilla.

2.3.2 Estructura Interna

A la hora de definir la estructura interna del proyecto se debe tener en cuenta que es una única persona la que realiza todas las tareas establecidas. Partiendo de esto seré tanto el analista, diseñador, programador y jefe del proyecto en cada una de las fases de la planificación, análisis, diseño e implementación. El rol de cliente lo asumirán mis tutores de TFG, Manuel Ángel González y Ramiro González. En el siguiente apartado se hará una breve descripción de los roles tomados por cada uno de los implicados en este proyecto.

2.3.3 Roles y Responsabilidades

En este punto se describen los roles que se van a utilizar a lo largo del proyecto, así como las personas o persona que realizará cada cual.

- Jefe de Proyecto - Jesús Brezmes Gil-Albarellos: Se encargará de dirigir cada tarea en las diferentes etapas propuestas. Algunas de sus tareas son las siguientes: Realizar la planificación del proyecto, revisión de los requisitos y documentos elaborados, revisión del sistema en desarrollo acorde a los requisitos, comunicación con el cliente…

- Analista - Jesús Brezmes Gil-Albarellos: Se encargará de recoger los requisitos y objetivos de la aplicación, así como de su continua revisión.

- Diseñador - Jesús Brezmes Gil-Albarellos: Se encargará de la creación de un concepto de sistema para asegurar las características de accesibilidad, navegabilidad, interactividad y usabilidad por parte del usuario.

- Programador - Jesús Brezmes Gil-Albarellos: Se encargará del desarrollo del código.

- Cliente - Manuel Ángel González Delgado, Ramiro González Catón: Se encarga de establecer las funcionalidades que debe cumplir el sistema, así como de su futura comprobación y revisión a lo largo y final del proyecto.

2.4 Plan de Control y Riesgos

2.4.1 Plan de Control

- Se dispondrá de las copias de seguridad de la documentación y recursos del proyecto, las cuales se realizarán con cierta regularidad a lo largo del tiempo. Todo ello será almacenado en Google Drive.
- Se dispondrá de un control de versiones con copias de seguridad sobre el código de la aplicación. El control de versiones se realizará a través de la herramienta GitKraken.
- Los requisitos de la aplicación estarán sometidos a revisión al final de cada fase para garantizar el correcto desarrollo de la aplicación.
- El cliente tomará parte en el desarrollo del proyecto probando las diferentes versiones de la aplicación para asegurar el correcto desempeño de los requisitos, pudiendo proponer modificaciones.

2.4.2 Gestión de Riesgos

Para el correcto desarrollo y consecución del proyecto se procede a elaborar el plan de gestión de riesgos. Este nos va a permitir establecer procedimientos para reaccionar ante los posibles riesgos que puedan ir surgiendo. En este punto están reunidos la identificación, análisis, planificación y evaluación de los riesgos.Para su descripción me voy a apoyar en la matriz de riesgos, la cual nos permite clasificarlos según su impacto y la probabilidad que tienen de producirse.

		PROBABILIDAD				
		Raro	Poco probable	Posible	Muy probable	Casi seguro
CONSECUENCIAS	Despreciable	Bajo	Bajo	Bajo	Medio	Medio
	Menores	Bajo	Bajo	Medio	Medio	Medio
	Moderadas	Medio	Medio	Medio	Alto	Alto
	Mayores	Medio	Medio	Alto	Alto	Muy alto
	Catastróficas	Medio	Alto	Alto	Muy alto	Muy alto

Figura 2.15: Matriz de Riesgos.

Los diferentes riesgos se describirán desde la siguiente perspectiva:

- Identificador: Se enumerará cada riesgo con un identificador único.
- Nombre: Nombre del riesgo.
- Descripción: Se proporcionará una pequeña descripción.
- Probabilidad: Probabilidad de que se produzca el riesgo en una escala entre (Bajo, Medio, Alto, Muy Alto).

- Consecuencias: Impacto de las consecuencias medido en una escala entre (Bajo, Medio, Alto, Muy Alto).
- Impacto: Descripción del impacto del riesgo si este llega a materializarse.
- Plan de mitigación: Estrategia para evitar que el riesgo se materialice en un hecho.
- Plan de contingencia: Plan a seguir una vez se haya materializado el riesgo en un hecho.

R-01	**Planificación temporal errónea.**
Descripción	Incumplimiento de los plazos establecidos en la planificación para el desarrollo del proyecto.
Probabilidad	Alta.
Consecuencias	Alto.
Impacto	Incumplir plazos de planificación y entrega del proyecto. El impacto aumenta junto con el paso del tiempo, debido a que impide reaccionar con tiempo.
Plan de mitigación	Disponer en la planificación de pequeños márgenes para poder permitirse ligeros retrasos.
Plan de contingencia	Aumentar el tiempo de trabajo diario hasta alcanzar la planificación establecida, o renegociar con el cliente los plazos de entrega.

Figura 2.16: R-01 Planificación temporal errónea.

R-02	**Modificación de los requisitos.**
Descripción	Hay que considerar que el documento de requisitos establecidos por el cliente puede verse en la necesidad de ser modificado.
Probabilidad	Baja
Consecuencias	Alto.
Impacto	Añadir o modificar requisitos equivale a rehacer partes del diseño/documentación. El impacto es mayor según avanza el proyecto, en las primeras fases tiene una influencia menor.
Plan de mitigación	Revisiones y pruebas por parte del cliente con cierta regularidad. De esta forma evitamos grandes modificaciones.
Plan de contingencia	Si se decide aceptarlos, se revisarán los requisitos, documentación y código afectados.

Figura 2.17: R-02 Modificación de los requisitos.

R-03	Requisitos poco/mal definidos.
Descripción	No se ha concretado bien el alcance o detalles de los requisitos.
Probabilidad	Baja.
Consecuencias	Alto.
Impacto	Rehacer partes del diseño/documentación. El impacto es mayor según avanza el proyecto, en las primeras fases tiene una influencia menor.
Plan de mitigación	Revisiones y pruebas por parte del cliente con cierta regularidad. De esta forma evitamos grandes modificaciones.
Plan de contingencia	Revisar la documentación y código afectados.

Figura 2.18: R-03 Requisitos poco/mal definidos.

R-04	Alcance/Objetivos del proyecto mal definidos.
Descripción	El alcance establecido con el cliente es inviable o no está bien detallado.
Probabilidad	Bajo.
Consecuencias	Alto.
Impacto	Imposibilidad de continuar con el proyecto hasta concretar el alcance. El impacto es mayor según avanza el proyecto.
Plan de mitigación	Realizar de forma regular reuniones con el cliente para asegurar la evolución del alcance.
Plan de contingencia	Reevaluación del alcance y de los objetivos para su correcta definición.

Figura 2.19: R-04 Alcance/Objetivos del proyecto mal definidos.

R-05	Desconocimiento o inexperiencia.
Descripción	Dedicar tiempo al aprendizaje de las herramientas requeridas para el desarrollo de la aplicación en las diferentes fases de este.
Probabilidad	Medio.
Consecuencias	Alto.
Impacto	Necesidad de dedicar mucho tiempo al aprendizaje de las herramientas. Cuanta mayor sea la complejidad del proyecto, esta aumentará aún más con el tiempo.
Plan de mitigación	Tener en cuenta en la planificación tiempo para el desarrollo de los conocimientos necesarios y aspectos técnicos.
Plan de contingencia	Una vez materializado este riesgo y con la imposibilidad de que algún miembro del equipo nos preste ayuda (debido a que está formado por un sólo miembro), pedir ayuda a los tutores o incluso ayuda externa al proyecto.

Figura 2.20: R-05 Desconocimiento o inexperiencia.

R-06	Pérdida de datos.
Descripción	Pérdida de información debido a múltiples razones: cortes eléctricos, pérdida de dispositivos… Lo que conlleva pérdida de la documentación o del código.
Probabilidad	Baja.
Consecuencias	Alto.
Impacto	Rehacer partes de código o de la documentación producen retrasos en la planificación y entrega del proyecto.
Plan de mitigación	Realizar copias de seguridad y control de versiones con mucha rigurosidad.
Plan de contingencia	Tratar de recuperar los datos. En caso de imposibilidad de esta primera parte, proceder con la versión más reciente de los recursos perdidos.

Figura 2.21: R-06 Pérdida de datos.

R-07	Diseño Incorrecto
Descripción	El diseño del sistema resulta inadecuado. Al realizar actividades de implementación puede encontrarse que el diseño carece del suficiente nivel de detalle o está mal enfocado, bien por la naturaleza del problema, o bien por restricciones de uso impuestas por tecnologías de terceros.
Probabilidad	Baja.
Consecuencias	Alto.
Impacto	Se pueden producir retrasos en la finalización y entrega del proyecto. Además, también modificaciones en los artefactos entregados
Plan de mitigación	Realizar de forma regular reuniones con el cliente para asegurar el correcto desarrollo de la aplicación.
Plan de contingencia	Aumentar el tiempo de trabajo diario hasta alcanzar la planificación establecida, o renegociar con el cliente los plazos de entrega.

Figura 2.22: R-07 Diseño incorrecto.

R-08	Fallo hardware.
Descripción	Fallo en los recursos físicos asociados al proyecto.
Probabilidad	Baja.
Consecuencias	Alto.
Impacto	Pueden dañarse parte de las copias de seguridad, produce inactividad hasta la resolución del problema...
Plan de mitigación	Realizar un buen mantenimiento del equipo y disponer de las copias de seguridad en otros dispositivos o en la nube.
Plan de contingencia	Reparar el sistema dañado, o en el peor de los casos sustituir el equipo y continuar con las copias de seguridad.

Figura 2.23: R-08 Fallo hardware.

R-09	Fallo software.
Descripción	Imposibilita la realización de tareas relacionadas a un determinado software. Puede producirse por un virus, error, mantenimiento...
Probabilidad	Baja.
Consecuencias	Medio.
Impacto	Retraso en la realización de tareas asociadas a ese software.
Plan de mitigación	Disponer de software similar al dañado para realizar su tarea.
Plan de contingencia	Tratar de solucionar el problema o continuar el proyecto con una aplicación similar.

Figura 2.24: R-09 Fallo software.

R-10	Consecuencias derivadas de agentes externos.
Descripción	Cortes de luz o internet, incomunicación debido a desastres naturales...
Probabilidad	Baja.
Consecuencias	Medio.
Impacto	Depende de la duración y trascendencia de lo ocurrido.
Plan de mitigación	Si los problemas son de índole grave no existe plan de mitigación.
Plan de contingencia	En el caso de un corte de luz, se pueden utilizar los datos móviles para el acceso a la red. Si los hechos son peores, esperar a que se solucione el problema y revisar la planificación.

Figura 2.25: R-10 Consecuencias derivadas de agentes externos.

R-11	Enfermedad
Descripción	Se puede producir contagio por alguna enfermedad o virus y que este imposibilite o reduzca el rendimiento a la hora de continuar con el proyecto.
Probabilidad	Baja.
Consecuencias	Medio.
Impacto	Retrasos en la planificación y entrega del proyecto. Cuanto peor sea la enfermedad las consecuencias aumentan.
Plan de mitigación	Dependiendo del impacto de la enfermedad existe o no plan de mitigación. En el caso de ser de larga duración no se podría realizar ninguna prevención.
Plan de contingencia	Volver a planificar el proyecto acorde con el tiempo perdido a causa de la enfermedad.

Figura 2.26: R11- Enfermedad.

2.5 Herramientas para el desarrollo

En esta sección se nombran y describen brevemente los recursos y herramientas utilizadas a lo largo del proyecto. Todas ellas son gratuitas, aunque alguna dispone de algunas operaciones restringidas de pago o incluso periodos gratuitos de prueba.

2.5.1 Eclipse

Eclipse es una plataforma de software compuesto por un conjunto de herramientas de programación de código abierto multiplataforma para desarrollar lo que el proyecto llama "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. En concreto se ha utilizado PyDev.

- PyDev: PyDev es un IDE de Python para Eclipse, que puede usarse en el desarrollo de Python, Jython y IronPython. En nuestro caso Python3.

Figura 2.27: Logotipo de PyDev.

2.5.2 Qt Designer

Qt Designer es la herramienta Qt para diseñar y construir interfaces gráficas de usuario (GUI) con Qt Widgets. En nuestro caso enfocado a Python, lo que se denomina PyQt.

Figura 2.28: Logotipo de Qt Designer.

2.5.3 Merlin Project

Software profesional de gestión de proyectos. Utilizado para la creación de la planificación del proyecto. Dispone de una prueba gratuita de 30 días.

Figura 2.29: Logotipo de Merlin Project.

2.5.4 GitKraken

Para el control de versiones de la aplicación se ha utilizado este cliente Git compatible con Windows, Linux y Mac. Proporciona una interfaz gráfica por la que la utilización de Git se vuelve más cómoda y rápida.

Figura 2.30: Logotipo de GitKraken.

2.5.5 MySQLWorkbench

MySQL Workbench es una herramienta visual unificada para desarrolladores de bases de datos. Está disponible en Windows, Linux y Mac OS X. MySQL Workbench proporciona modelado de datos, desarrollo de SQL y herramientas de administración integrales para la configuración del servidor, la administración de usuarios, la copia de seguridad…

Figura 2.31: Logotipo de MySQLWorkbench.

2.5.6 Google Drive

Para el almacenamiento en la nube de los diferentes documentos, guías de la planificación, memoria de prácticas y backups se ha utilizado Google Drive.

Figura 2.32: Logotipo de Google Drive.

2.5.7 Visual Paradigm

Para la realización de diagramas y como editor de UML se ha utilizado Visual Paradigm.

Figura 2.33: Logotipo de UML editor.

2.5.8 Equipo de Trabajo

Si hablamos de los recursos hardware utilizados, se compone de un ordenador portátil con las siguientes características:

Figura 2.34: Características del portátil.

Capítulo 3

Análisis de la Aplicación

Resumen: *A lo largo de este capítulo se hablará sobre los requisitos que definen el proyecto, la especificación de los casos de uso y a continuación se presentará el Modelo de Dominio, este nos proporcionará una visualización de los conceptos del dominio. Finalmete se mostrará el modelo entidad relación con el que se describirá como la aplicación realizará el tratamiento de los datos.*

3.1 Requisitos

Para definir lo que es un requisito nos basamos en la definición que nos da RUP (Rational Unified Process), “Una condición o capacidad que debe ser conformada por el sistema”. Es decir, definen las funcionalidades y cualidades del sistema a desarrollar. A continuación, se expondrán los requisitos que se han encontrado para esta aplicación, clasificados en Funcionales, no Funcionales y de Información.

3.1.1 Requisitos Funcionales

Identificador	**Nombre**	**Descripción**
RF-01	Graficar el uso de fotómetros.	El sistema debe proporcionar una gráfica con el uso de todos los fotómetros a lo largo del tiempo.
RF-02	Navegabilidad en la gráfica uso de fotómetros.	El sistema debe permitir la interacción con la gráfica de uso de fotómetros. Realizar zoom, restablecer valores...
RF-03	Filtrado de fotómetros.	El sistema deberá permitir filtrar los fotómetros que se muestran en la gráfica de uso de fotómetros según su eprom_type o subeprom_type.
RF-04	Selección de fotómetro.	El sistema deberá permitir generar y mostrar las medidas de un fotómetro, seleccionando el correspondiente y recogiendo el rango de fechas actual de la gráfica de uso de fotómetros.
RF-05	Medidas AOD.	El sistema deberá permitir generar y mostrar la gráfica con las medidas AOD para un fotómetro y un rango de fechas dado.
RF-06	Medidas WExp.	El sistema deberá permitir generar y mostrar la gráfica con las medidas WExp para un fotómetro y un rango de fechas dado.

RF-07	Medidas Vapor de Agua.	El sistema deberá permitir generar y mostrar la gráfica con las medidas de vapor de agua para un fotómetro y un rango de fechas dado.
RF-08	Medidas de la Temperatura.	El sistema deberá permitir generar y mostrar la gráfica con las medidas de temperatura para un fotómetro y un rango de fechas dado.
RF-09	Medidas de PWR (reactores de agua ligera a presión)	El sistema deberá permitir generar y mostrar la gráfica de medidas PWR para un fotómetro y un rango de fechas dado.
RF-10	Navegabilidad en cualquier gráfica de representación de medidas.	El sistema debe permitir la interacción cualquiera de las gráficas de representación de medidas. Realizar zoom, restablecer valores...
RF-11	Filtrado installation_type	El sistema deberá tener en cuenta sólo aquellos fotómetros cuyo installatio_type sea: • Master • Routine • Calibration
RF-12	Control de errores.	El sistema debe disponer de un control de errores para la consistencia de la aplicación
RF-13	Disponibilidad.	El sistema deberá estar en funcionamiento a cualquier hora del día y cualquier día del año.
RF-14	Tiempo de respuesta.	El sistema deberá responder a las peticiones del usuario en un periodo de tiempo razonable, teniendo en cuenta la gran cantidad de registros en la base de datos que se manejan en alguno de los casos de uso.
RF-15	Aspecto/Rendimiento.	El sistema debe priorizar el rendimiento sobre la estética y aspecto de la aplicación.
RF-16	Usuarios.	El sistema sólo tendrá en cuenta un usuario y un nivel de acceso.

Figura 3.1: Requisitos Funcionales.

3.1.2 Requisitos no Funcionales

Identificador	Nombre	Descripción
RnF-01	Usabilidad.	El sistema debe ser usable, es decir, al ser una réplica de una herramienta ya existente, que los usuarios no noten mucho cambio con la versión anterior.
RnF-02	Control de errores.	El sistema mostrará los diferentes errores de la forma más clara posible para facilitar su resolución.
RnF-03	Interfaz de Usuario.	El sistema ofrecerá una interfaz de usuario basada en PyQt.
RnF-04	Gestión de la Aplicación.	El sistema gestionará la aplicación utilizando tecnología Python3.
RnF-05	Diseño de la arquitectura.	El sistema estará diseñado bajo una arquitectura basada en MVC (Modelo, Vista, Controlador).
RnF-06	Acceso a los datos.	El sistema debe permitir la conexión a la base de datos vía MySQL.
RnF-07	Diferenciación de los fotómetros en la gráfica de uso. .	El sistema deberá distinguir por colores, según el eprom_type y subeprom_type, los diferentes tipos de fotómetros existentes. Para ello se a de seguir las siguientes directrices: • Digital extended: Marrón • Triple: Naranja • Standard: Gris/Negro • Dualpolar (No triple): Azul
RnF-08	Representación de la medida AOD.	El sistema deberá, a la hora de graficar este tipo de medidas, indicar la media para cada canal, desviación estándar, máximo y mínimo a lo largo del tiempo.
RnF-09	Colores según la banda espectral.	El sistema deberá representar cada conjunto de datos de una banda espectral con un determinado color. El cliente ha entregado la siguiente codificación en hexadecimal para cada banda (nanómetros): • 1640 : #008000 • 1240 : #008000 • 1020 : #ff0000 • 1020 : #0000ff • 935 : blue

		• 870 : #9370db • 675 : #556b2f • 667 : #606a31 • 551 : #ff6347 • 532 : #8b4513 • 500 : #2e6b57 • 490 : #2e6b57" • 443 : #a54141 • 440 : #a54141 • 412 : #228b22 • 380 : #228b22 • 340 : #b22222

Figura 3.2: Requisitos no Funcionales.

3.1.3 Requisitos de Información

Identificador	**Nombre**	**Descripción**
RdI-01	Fotómetros.	El sistema deberá coleccionar la siguiente información sobre los fotómetros: • Número de fotómetro. • Nombre de la estación. • Fecha de inicio de la muestra. • Fecha de fin de la muestra. • Tipo de eprom_type • Tipo de eprom_subtype
RdI-02	Medidas del fotómetro.	El sistema deberá coleccionar la información realtiva a las siguientes medidas: • AOD: ○ Existen entre 8 y 10 bandas espectrales en las que el fotómetro recoge datos, estas bandas están centradas en las longitudes de onda de: ■ 340 nm. ■ 38 nm. ■ 440 nm. ■ 500 nm. ■ 675 nm. ■ 870 nm. ■ 935 nm. ■ 1020 nm. ■ 1640 nm. ○ Para cada fecha y canal dispone de tres medidas. Hay que seleccionar la media,

		el máximo y el mínimo. • Temperatura. • Vapor de agua. • WExp: ○ Dispone de dos canales distintos de información, alpha 440-870 y alpha 380-500. • PWR: ○ Existe para el canal 1, que cuenta con una banda espectral de 1020 nanómetros.
RdI-03	Filtrado medidas del fotómetro.	El sistema deberá coleccionar la información del RdI-02 con las variantes de filtrado por niveles: • Level 1.0: Filtro quitando null_values y notL10 values. • Level 1.5: Filtro con medidas que sean cloud_free o restoration.

Figura 3.3: Requisitos de Información.

3.2 Casos de Uso

A continuación, se mostrará los diagramas de casos de uso, divididos por los dos objetivos marcados, es decir, las dos ventanas a desarrollar. Ver Figura 3.4 y Figura 3.5.

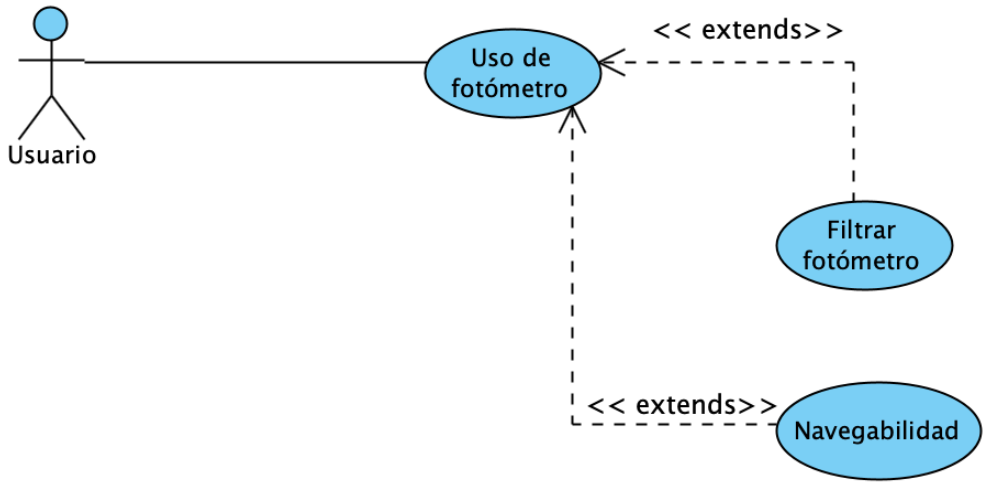


Figura 3.4: Casos de uso para la "*Ventana de Uso de fotómetros*".

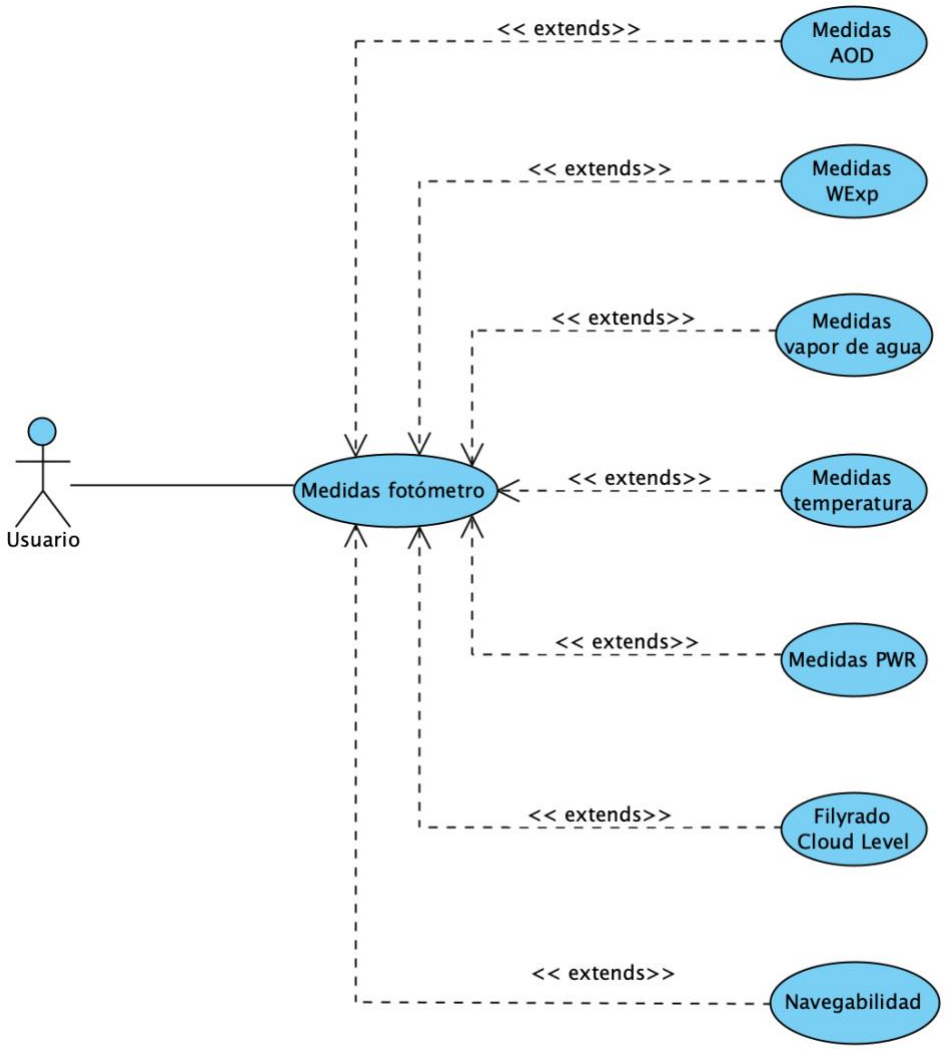


Figura 3.5: Casos de Uso para la "*Ventana Graficación de medidas*".

3.2.1 Especificación de los Casos de Uso

CU-01	Uso de fotómetro.
Descripción	Representar la gráfica de uso de los fotómetros.
Actor	Usuario.
Precondiciones	Se dispone de la lista de fotómetros a representar.
Flujo normal	1. El Usuario selecciona representar los fotómetros 2. El Sistema analiza los datos de los fotómetros y realiza el tratamiento para su representación. Una vez realizado el sistema genera la gráfica y la muestra.
Postcondiciones	
Flujo alternativo	

Figura 3.6: CU-01 Uso de fotómetro.

CU-02	Filtrar fotómetro.
Descripción	Filtrar los fotómetros a analizar según su nombre, estación o tipo.
Actor	Usuario.
Precondiciones	Se ha realizado el CU-01.
Flujo normal	1. El Usuario selecciona los filtros que quiere aplicar. 2. El Sistema recupera los datos relativos a los fotómetros acorde a los filtros introducidos por el Usuario. 3. El Sistema genera la gráfica y la muestra.
Postcondiciones	
Flujo alternativo	2.1. No hay datos disponibles para dicho filtro por lo que el caso de uso queda sin efecto.

Figura 3.7: CU-02 Filtrar fotómetro.

CU-03	Navegabilidad uso fotómetro.
Descripción	Navegabilidad en la gráfica uso de fotómetro, realizar zoom, desplazarse por los valores...
Actor	Usuario.
Precondiciones	Se ha realizado el CU-01.
Flujo normal	1. El usuario selecciona una de las opciones para navegar por la gráfica. 2. El Sistema comienza a recoger información relativa al puntero del ratón. 3. El Usuario realiza la opción para la navegación 4. El Sistema realiza los cambios relativos a la acción seleccionada.
Postcondiciones	
Flujo alternativo	3.1. El usuario decide no realizar la acción y el Caso de uso queda sin efecto.

Figura 3.8: CU-03 Navegabilidad uso fotómetro.

CU-04	Ventana medidas de fotómetro.
Descripción	Se muestra una nueva ventana destinada a la graficación de las medidas de un fotómetro en concreto.
Actor	Usuario.
Precondiciones	
Flujo normal	1. El Usuario selecciona un fotómetro para su análisis. 2. El Sistema comprueba si existe la información necesaria de dicho fotómetro. 3. El Sistema recupera los datos por defecto, AOD, es decir, realiza el CU-05.
Postcondiciones	
Flujo alternativo	2.1. No hay datos para dicho fotómetro en las fechas dadas por lo que el caso de uso queda sin efecto.

Figura 3.9: CU-04 Ventana medidas de fotómetro.

CU-05	Medidas AOD.
Descripción	Representación de las medidas AOD relativas a un fotómetro.
Actor	Usuario.
Precondiciones	Se ha realizado el CU-04.
Flujo normal	1. El Usuario selecciona el botón para la representación de las medidas AOD. 2. El Sistema recupera los datos del fotómetro para la obtención de sus medidas AOD. Una vez obtenidos genera la gráfica y la muestra.
Postcondiciones	
Flujo alternativo	

Figura 3.10: CU-05 Medidas AOD.

CU-06	Medidas Temperatura
Descripción	Representación de las medidas de Temperatura relativas a un fotómetro.
Actor	Usuario.
Precondiciones	Se ha realizado el CU-04.
Flujo normal	1. El Usuario selecciona el botón para la representación de las medidas de Temperatura. 2. El Sistema recupera los datos del fotómetro para la obtención de sus medidas de Temperatura. Una vez obtenidos genera la gráfica y la muestra.
Postcondiciones	
Flujo alternativo	

Figura 3.11: CU-06 Medidas Temperatura.

CU-07	Medidas Vapor de Agua
Descripción	Representación de las medidas de Vapor de Agua relativas a un fotómetro.
Actor	Usuario.
Precondiciones	Se ha realizado el CU-04.
Flujo normal	1. El Usuario selecciona el botón para la representación de las medidas de Vapor de Agua. 2. El Sistema recupera los datos del fotómetro para la obtención de sus medidas de Vapor de Agua. Una vez obtenidos genera la gráfica y la muestra.
Postcondiciones	
Flujo alternativo	

Figura 3.12: CU-07 Medidas Vapor de Agua.

CU-08	Medidas WExp
Descripción	Representación de las medidas WExp relativas a un fotómetro.
Actor	Usuario.
Precondiciones	Se ha realizado el CU-04.
Flujo normal	1. El Usuario selecciona el botón para la representación de las medidas WExp. 2. El Sistema recupera los datos del fotómetro para la obtención de sus medidas WExp. Una vez obtenidos genera la gráfica y la muestra.
Postcondiciones	
Flujo alternativo	

Figura 3.13: CU-08 Medidas WExp.

CU-09	Medidas PWR
Descripción	Representación de las medidas PWR relativas a un fotómetro.
Actor	Usuario.
Precondiciones	Se ha realizado el CU-04.
Flujo normal	1. El Usuario selecciona el botón para la representación de las medidas PWR. 2. El Sistema recupera los datos del fotómetro para la obtención de sus medidas PWR. Una vez obtenidos genera la gráfica y la muestra.
Postcondiciones	
Flujo alternativo	

Figura 3.14: CU-09 Medidas PWR.

CU-10	Navegabilidad gráfica de medidas.
Descripción	Navegabilidad por la gráfica medidas de un fotómetro, realizar zoom, desplazarse por los valores...
Actor	Usuario.
Precondiciones	Se ha realizado el CU-04.
Flujo normal	1. El Usuario selecciona una de las opciones para navegar por la gráfica. 2. El Sistema comienza a recoger información relativa al puntero del ratón. 3. El Usuario realiza la opción para la navegación 4. El Sistema realiza los cambios relativos a la acción seleccionada.
Postcondiciones	
Flujo alternativo	3.1. El Usuario decide no realizar la acción y el Caso de uso queda sin efecto.

Figura 3.15: CU-10 Navegabilidad gráficas de medidas.

CU-11	Filtrado Cloud Level.
Descripción	Filtrado de las medidas de los fotómetros en función de las nubes.
Actor	Usuario.
Precondiciones	Se ha realizado el CU-04.
Flujo normal	1. El Usuario selecciona el filtrado de nubes que quiere realizar. 2. El Sistema aplica el filtro seleccionado por el Usuario a los datos. 3. El Sistema realiza alguno de los casos de uso: CU-05 - CU-09, en función de cual estuviera activo en ese momento.
Postcondiciones	
Flujo alternativo	

Figura 3.16: CU-11 Filtrado Cloud Level.

3.3 Modelo de Dominio

El modelo de dominio nos ayuda a comprender los objetos modelados del mundo real y que forman nuestro sistema, además en él también se observan las relaciones existentes entre ellos. En esta aplicación el elemento principal es el fotómetro, y los secundarios los diferentes tipos de medidas que podemos obtener de ellos. Dentro de los dos objetivos principales podemos decir que la “Ventana de Uso de los Fotómetros” estará centrada en “fotómetro” e “installation”, y la de “Graficación de medidas” en “fotómetro” y en “medias_simples” y derivados. Estas clases se pueden observar en la Figura 3.17.

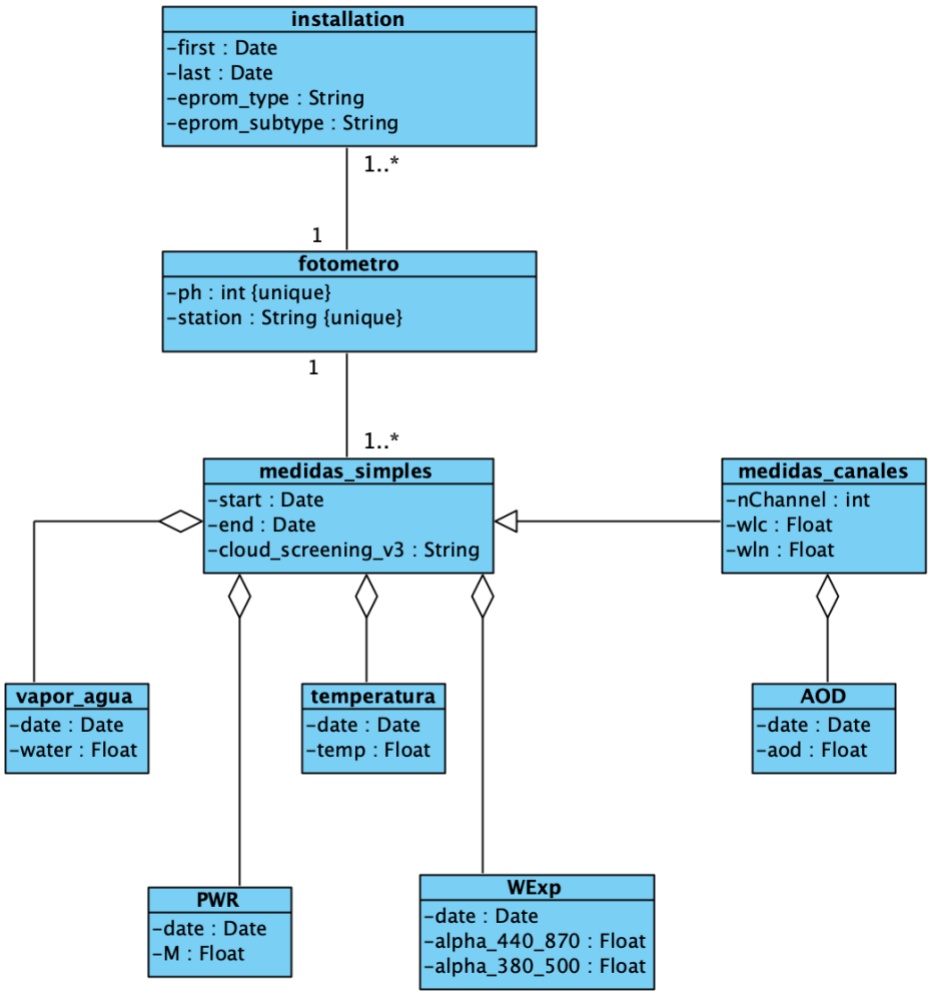


Figura 3.17: Diagrama de Clases de Análisis.

3.3.1 Descripción de las clases de Análisis

fotómetro		
Esta clase se encarga de modelar los fotómetros.		
Atributo	**Tipo**	**Descripción**
ph	INT	Identificador del fotómetro de la red AERONET.
station	VARCHAR(45)	Localización del fotómetro.

Figura 3.18: Clase de Análisis: Fotómetro.

installation		
Esta clase se encarga de modelar las diferentes instalaciones por las que pasa un fotómetro.		
Atributo	**Tipo**	**Descripción**
ph	INT	Identificador del fotómetro de la red AERONET.
station	VARCHAR(45)	Localización del fotómetro.
first	DATETIME	Fecha en la que se realizó la instalación del fotómetro.
last	DATETIME	Fecha en la que termina el periodo de utilización del fotómetro.

Figura 3.19: Clase de Análisis: Installation.

medida_simples		
Esta clase se encarga de modelar las medidas, es decir, que no disponen de diferentes canales para cada banda espectral.		
Atributo	**Tipo**	**Descripción**
start	INT	Fecha a partir de la cual se tienen en cuenta las medidas para la muestra.
end	VARCHAR(45)	Fecha hasta la cual se tienen en cuenta las medidas para la muestra.
cloud_screening_V3	VARCHAR(45)	Filtro de cloud level.

Figura 3.20: Clase de Análisis: Medidas Simples.

vapor_agua		
Esta clase se encarga de modelar las medidas de vapor de agua recogidas para una fecha..		
Atributo	**Tipo**	**Descripción**
date	DATETIME	Fecha en la que se recoge la medida de vapor de agua..
water	FLOAT	Valor recogido de vapor de agua en una determinada fecha.

Figura 3.21: Clase de Análisis: Vapor de Agua.

temperatura		
Esta clase se encarga de modelar las medidas de temperatura recogidas para una fecha.		
Atributo	**Tipo**	**Descripción**
date	DATETIME	Fecha en la que se recoge la medida de temperatura.
temp	FLOAT	Valor recogido de la temperatura en una determinada fecha.

Figura 3.22: Clase de Análisis: Temperatura.

WExp		
Esta clase se encarga de modelar las medidas de WExp recogidas para una fecha.		
Atributo	**Tipo**	**Descripción**
date	DATETIME	Fecha en la que se recoge la medida de WExp..
alpha_440-870	FLOAT	Valor recogido de WExp para alpha_440-870 en una determinada fecha.
alpha_380-500	FLOAT	Valor recogido de WExp para alpha_380-500 en una determinada fecha.

Figura 3.23: Clase de Análisis: WExp.

PWR		
Esta clase se encarga de modelar los fotómetros.		
Atributo	**Tipo**	**Descripción**
date	DATETIME	Fecha en la que se recoge la medida de PWR.
M	FLOAT	Valor recogido de PWR en una determinada fecha.

Figura 3.24: Clase de Análisis: PWR.

medida_canal		
Esta clase se encarga de modelar las medidas, es decir, que no disponen de diferentes canales para cada banda espectral.		
Atributo	**Tipo**	**Descripción**
nChannel	INT	Identificador del canal para la banda espectral.
wln	DECIMAL(4, 3)	Banda espectral de la toma de medidas, pero nominal.
wlc	FLOAT	Banda espectral de la toma de medidas.

Figura 3.25: Clase de Análisis: Medida Canal.

AOD		
Esta clase se encarga de modelar los fotómetros.		
Atributo	**Tipo**	**Descripción**
date	DATETIME	Fecha en la que se recoge la medida de AOD.
aod	FLOAT	Valor recogido de AOD en una determinada fecha.

Figura 3.26: Clase de Análisis: AOD.

3.4 Modelo Entidad Relación

Este modelo es la mejor forma de representar la estructura de la base de datos a la que se tiene acceso para el desarrollo de la aplicación, CÆLIS. Volviendo a enfocarnos en los dos objetivos principales podemos decir que la "Ventana de Uso de los Fotómetros" estará centrada en la vista "cml_view_installation_interval", y la de "Graficación de medidas" se centrará en el resto. Dichas tablas podemos observarlas en la Figura 3.27.

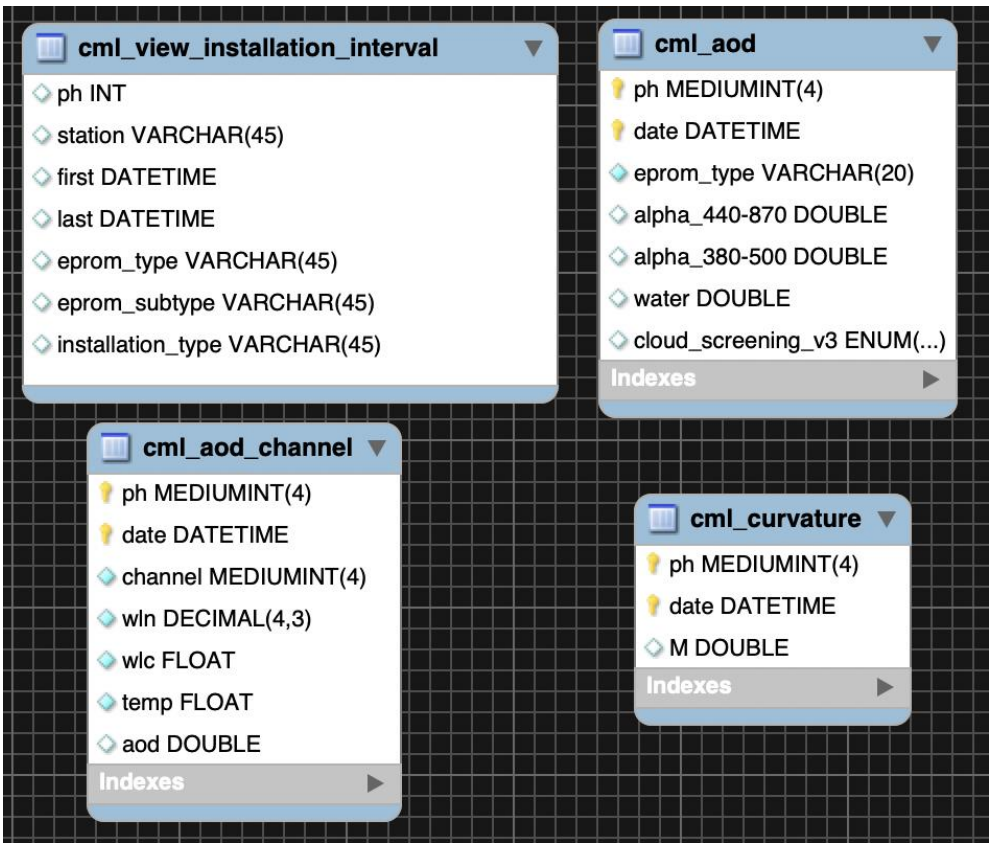


Figura 3.27: Diagrama de Entidad Relación.

Todas estas tablas pertenecen a una base de datos existente, y más grande, a la que se ha proporcionado el acceso. El diseño de esta base de datos viene impuesto por el cliente. Cml_view_installation_interval es una vista ya existente la cual, para esta aplicación, es tratada como una tabla más. El diagrama entidad-realción completo de CÆLIS se muestra en el Anexo III.

3.4.1 Descripción de las entidades

cml_view_installation_interval			
Almacena la información relativa al uso de los fotómetros.			
Atributo	**Tipo**	**Descripción**	**Relación**
ph	INT	Identificador del fotómetro de la red AERONET.	
station	VARCHAR(45)	Localización del fotómetro.	
first	DATETIME	Fecha en la que se realizó la instalación del fotómetro.	
last	DATETIME	Fecha en la que termina el periodo de utilización del fotómetro.	
eprom_type	VARCHAR(45)	Tipo de fotómetro para el cual la eprom está diseñada. Definido por: Enum('standard','extended','polarized','dualpolar','seaprism').	
eprom_subtype	VARCHAR(45)	Tipo interno del fotómetro en cuanto si es analógico, digital o triple. Definido por: Enum('digital','analog','triple').	
installation_type	VARCHAR(45)	Tipo de instalación en la que se encuentra el fotómetro. Los que interesan son: ('master', 'routine', 'calibration').	

Figura 3.28: Descripción de la vista cml_view_installation_interval.

cml_aod			
Almacena la información relativa al uso de los fotómetros.			
Atributo	**Tipo**	**Descripción**	**Relación**
ph	INT	Identificador del fotómetro de la red AERONET.	
date	DATETIME	Fecha en la que termina el periodo de utilización del fotómetro.	
eprom_type	VARCHAR(45)	Tipo de fotómetro para el cual la eprom está diseñada. Definido por: Enum('standard','extended','polarized','dualpolar','seaprism').	
alpha_440-870	DOUBLE	Campo en el que se guardan las mediciones de WExp para alpha_440-870.	
alpha_380-500	DOUBLE	Campo en el que se guardan las mediciones de WExp para alpha_380-500.	
water	DOUBLE	Campo en el que se guardan las mediciones del vapor de agua.	
cloud_screening_V3	VARCHAR(45)	Filtro de cloud level. Definido por: enum('null_values','not_evaluated','cloud_free','restoration','optical_airmass','potential_meassures','large_triplet','amstrong_exponent','smoothness_criteria','curvature_check','standalones','3-sigma','notL10','curvature_check_alm','curvature_check_pp').	

Figura 3.29: Descripción de la tabla cml_aod.

cml_aod_channel			
Almacena la información relativa al uso de los fotómetros.			
Atributo	**Tipo**	**Descripción**	**Relación**
ph	INT	Identificador del fotómetro de la red AERONET.	cml_aod.ph
date	DATETIME	Fecha en la que termina el periodo de utilización del fotómetro.	cml_aod.date
channel	INT	Identificador para el canal.	
wln	DECIMAL(4, 3)	Banda espectral de la toma de medidas, pero nominal.	
wlc	FLOAT	Banda espectral de la toma de medidas.	
temp	FLOAT	Campo en el que se guardan las mediciones de temperatura.	
aod	DOUBLE	Campo en el que se guardan las mediciones de AOD.	

Figura 3.30: Descripción de la tabla cml_aod_channel.

cml_curvature			
Almacena la información relativa al uso de los fotómetros.			
Atributo	**Tipo**	**Descripción**	**Relación**
ph	INT	Identificador del fotómetro de la red AERONET.	
date	DATETIME	Fecha en la que termina el periodo de utilización del fotómetro.	
M	DOUBLE	Campo que guarda las mediciones de PWR.	

Figura 3.31: Descripción de la tabla cml_curvature.

Capítulo 4

Diseño de la Aplicación

Resumen: *En este capítulo se define la arquitectura software de la aplicación, cómo es representada, su descomposición en subsistemas o paquetes y el estudio más en profundidad de estos.*

4.1 Elección tecnológica

A la hora de seleccionar la tecnología para el desarrollo de la aplicación, estamos condicionados por los requerimientos y necesidades del sistema, así como de las herramientas ya existentes que se nos han presentado, CÆLIS.

El sistema gestor de la base de datos CÆLIS utiliza MySQL, por lo que a la hora de desarrollar el acceso a los datos es conveniente utilizar este mismo gestor. Pero esta media obligación no es un inconveniente, es decir, MySQL es uno de los sistemas de gestión de bases de datos más utilizados, llegando a ser considerado como la base de datos de código abierto más popular del mundo.

Otra necesidad de nuestro sistema consiste en el manejo de grandes cantidades de información, llegando a tener que gestionar decenas de miles de resultados para su posterior tratamiento y representación. Esto implica que debemos hacer una clara separación entre las diferentes partes que componen nuestro sistema. La solución elegida es llevar a cabo una arquitectura MVC, Modelo Vista Controlador, de este modo seremos capaces de realizar una separación de la lógica de negocio de la aplicación, de su representación y del módulo que gestiona las comunicaciones y eventos.

Para reducir la dependencia de código, flexibilizar el desarrollo del sistema y disminuir la complejidad de la comunicación, se ha determinado que se implementará el patrón fachada, el cual permite, al estar dividido la aplicación en subsistemas, delegar determinadas peticiones a las partes encargadas de su realización.

Finalmente, y para desarrollar la aplicación se ha decidido usar el lenguaje de programación Python, este lenguaje de alto nivel es muy transigente a la hora de la legibilidad y limpieza del código, algo fundamental para la realización de nuestra aplicación. Además, dispone de una librería muy interesante para la representación y graficación de datos, Matplotlib, lo cual es una de las metas más importantes de este TFG.

4.2 Arquitectura del sistema

Este concepto hace referencia a cómo se va a estructurar el sistema, como se van a organizar los componentes de este, o cómo interactúan y se relacionan entre sí. Esto representa un diseño de alto nivel del sistema y tiene dos objetivos principales:

- Servir como guía en el desarrollo de la aplicación.
- Satisfacer los atributos, requisitos y cualidades de la aplicación, (calidad, seguridad, desempeño…)

4.2.1 Modelo Vista Controlador

El patrón Modelo Vista Controlador realiza una separación de la lógica de negocio de la aplicación, de su representación y del módulo que gestiona las comunicaciones y eventos. Para ello este propone la elaboración de tres componentes bien diferenciados:

- Modelo: es la representación de los datos con los cuales va a operar el sistema. Este se encarga tanto del acceso a la información como de las actualizaciones que sean necesarias aplicar en esta. En el caso de existir, también se encarga de implementar los privilegios de acceso especificados según el tipo de usuario. Todas las peticiones que llegan al modelo lo hacen de mano del controlador.

- Vista: representa los datos del modelo de una forma en la que se pueda interactuar con ellos, es decir, la interfaz de usuario.

- Controlador: podrá ser definido como el intermediario entre modelo y vista, ya que es el encargado de responder a eventos (generados normalmente por el usuario en la vista) y generar peticiones al modelo. También puede generar peticiones de cambio a la vista, si se diera la necesidad.

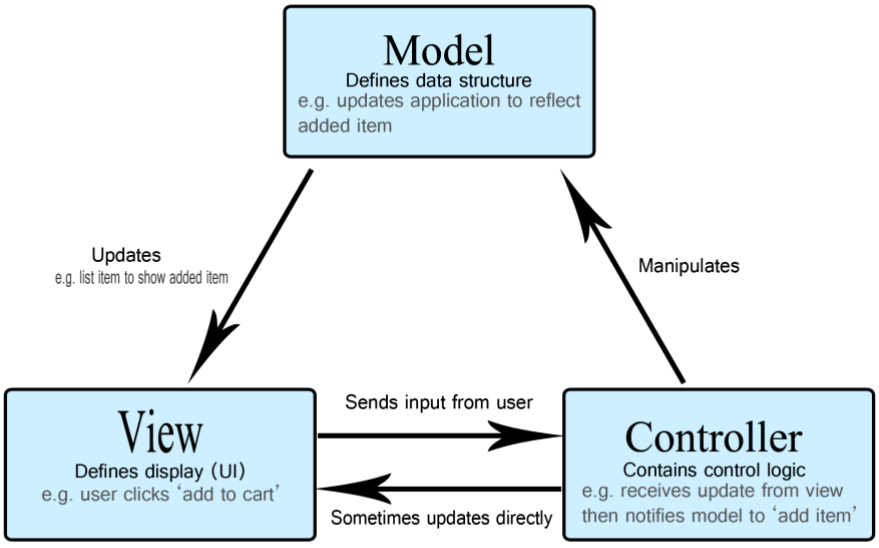


Figura 4.1: Modelo Vista Controlador.

En nuestro caso vamos a utilizar una variante del MVC, el pasivo. Las principales diferencias o variantes de este son:

- El controlador es el único encargado de manipular el modelo.
- El controlador se encarga de comunicar a la vista los cambios en el modelo.
- El modelo es prácticamente independiente de la vista y del controlador.

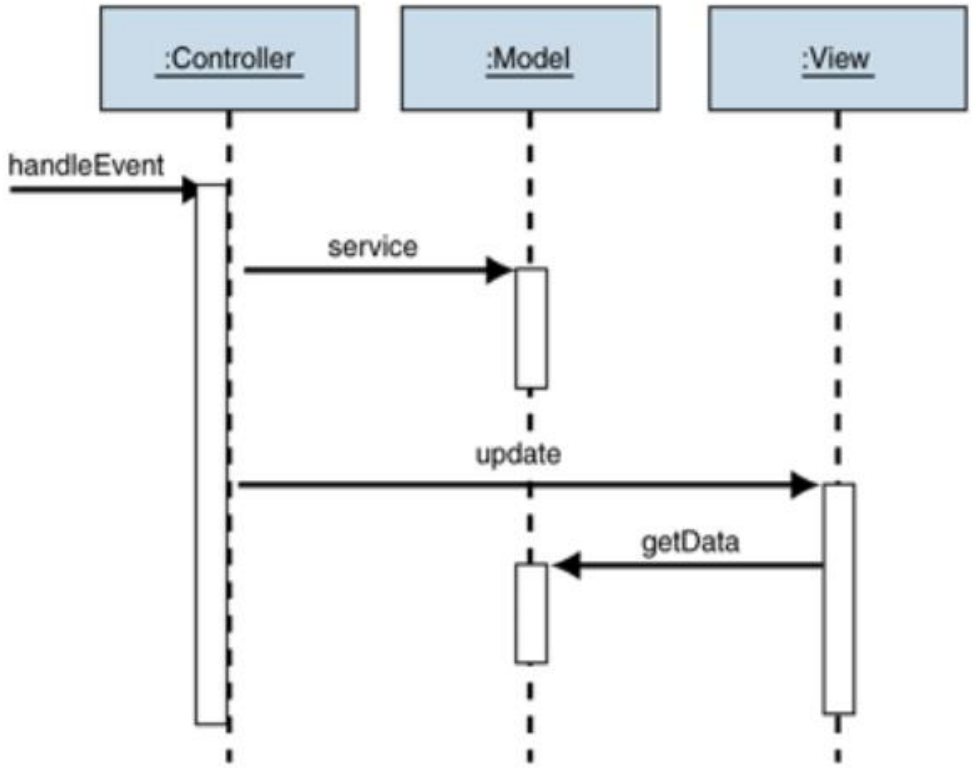


Figura 4.2: MVC Pasivo.

4.2.2 Patrón Fachada

El patrón fachada permite reducir las dependencias entre subsistemas, y así proporcionar una interfaz simple para un subsistema complejo. En este patrón arquitectónico intervienen dos participantes:

- Fachada (Facade): se encarga de delegar las peticiones a los objetos apropiados de un subsistema, esto se debe a que conoce qué clases del sistema se encargan de realizar dichas peticiones.
- Subclase: se encargan de realizar una funcionalidad del sistema, es decir, el trabajo que les solicita la fachada.

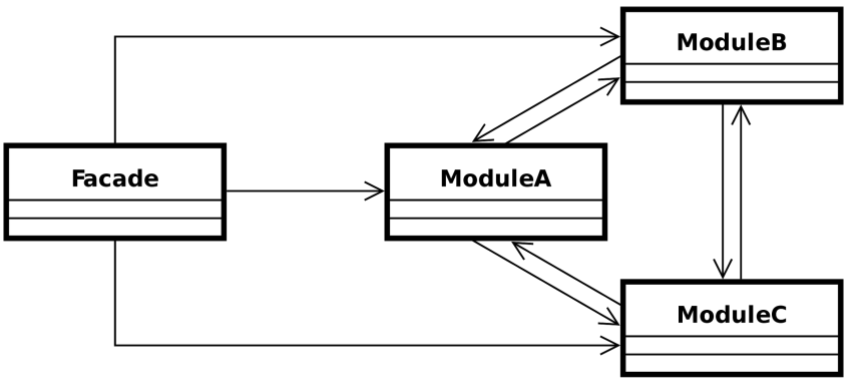


Figura 4.3: Patrón Fachada.

Los objetos del subsistema son los que realizan el trabajo, pero la fachada puede tener que realizar alguna tarea para pasar de su interfaz a las del subsistema. Todo esto se realiza para que los clientes usen la fachada sin tener que acceder a la complejidad de los objetos del subsistema. El controlador se hará cargo de hacer de fachada entre la vista y el modelo.

4.3 Diagrama de clases de Diseño

A continuación, se mostrará el diagrama de clases de diseño, el cual se encarga de describir todos y cada uno de los componentes que forman el sistema. También se puede observar cómo se relacionan dichos componentes, y de qué operaciones y atributos están compuestos. En la Figura 4.4 podemos observar el diagrama de diseño desde una perspectiva general. El diagrama desarrollado se mostrará en los siguientes puntos dividido por cada uno de los componentes MVC en los que se basa el sistema.

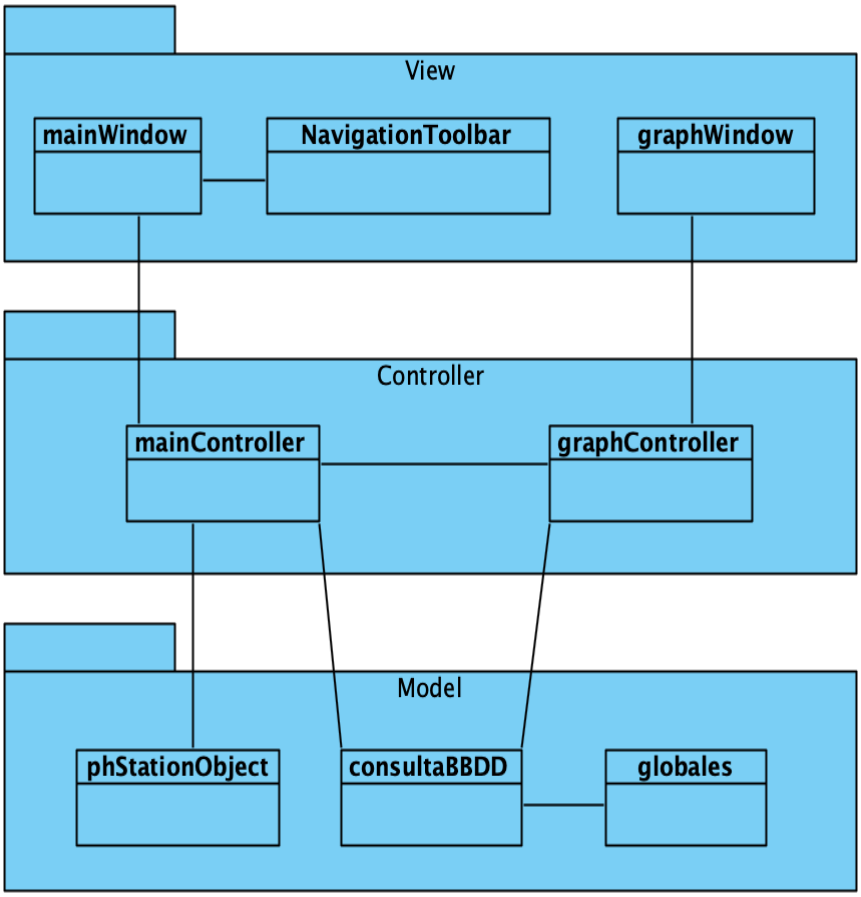


Figura 4.4: Diagrama de clases de Diseño.

4.3.1 Modelo

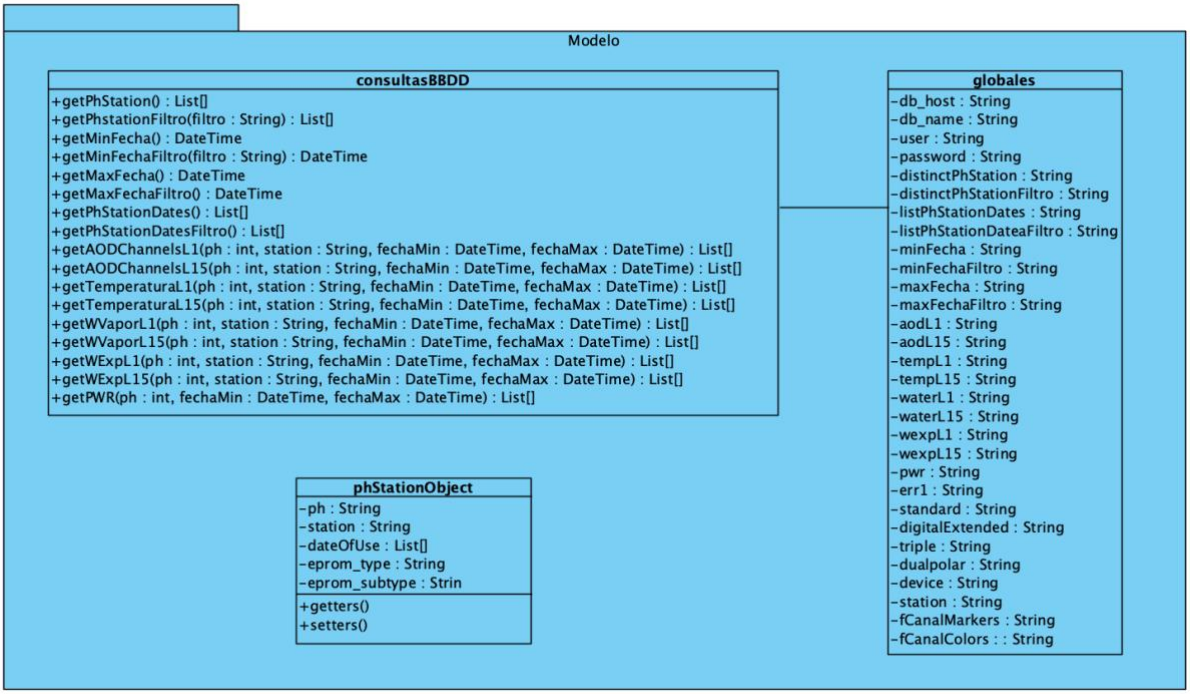


Figura 4.5: Diagrama de clases de Diseño - Modelo.

4.3.2 Vista

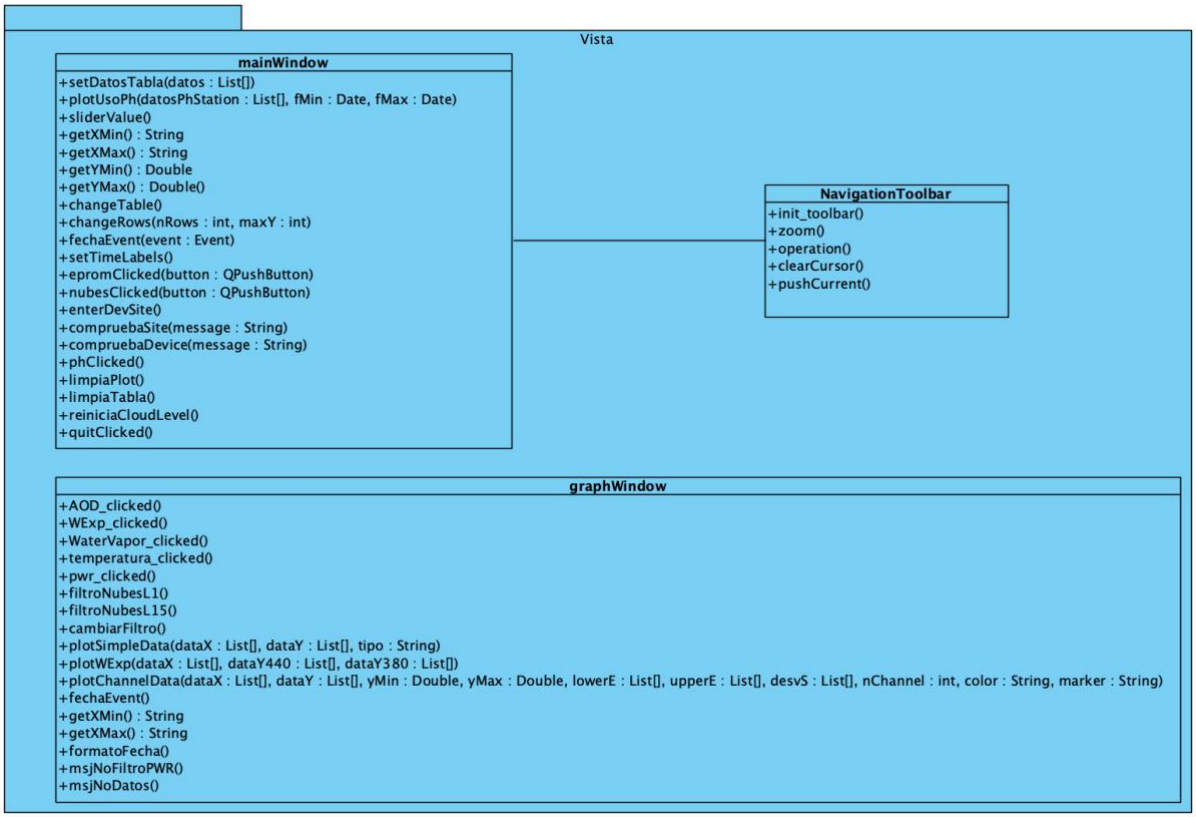


Figura 4.6: Diagrama de clases de Diseño - Vista.

4.3.1 Controlador

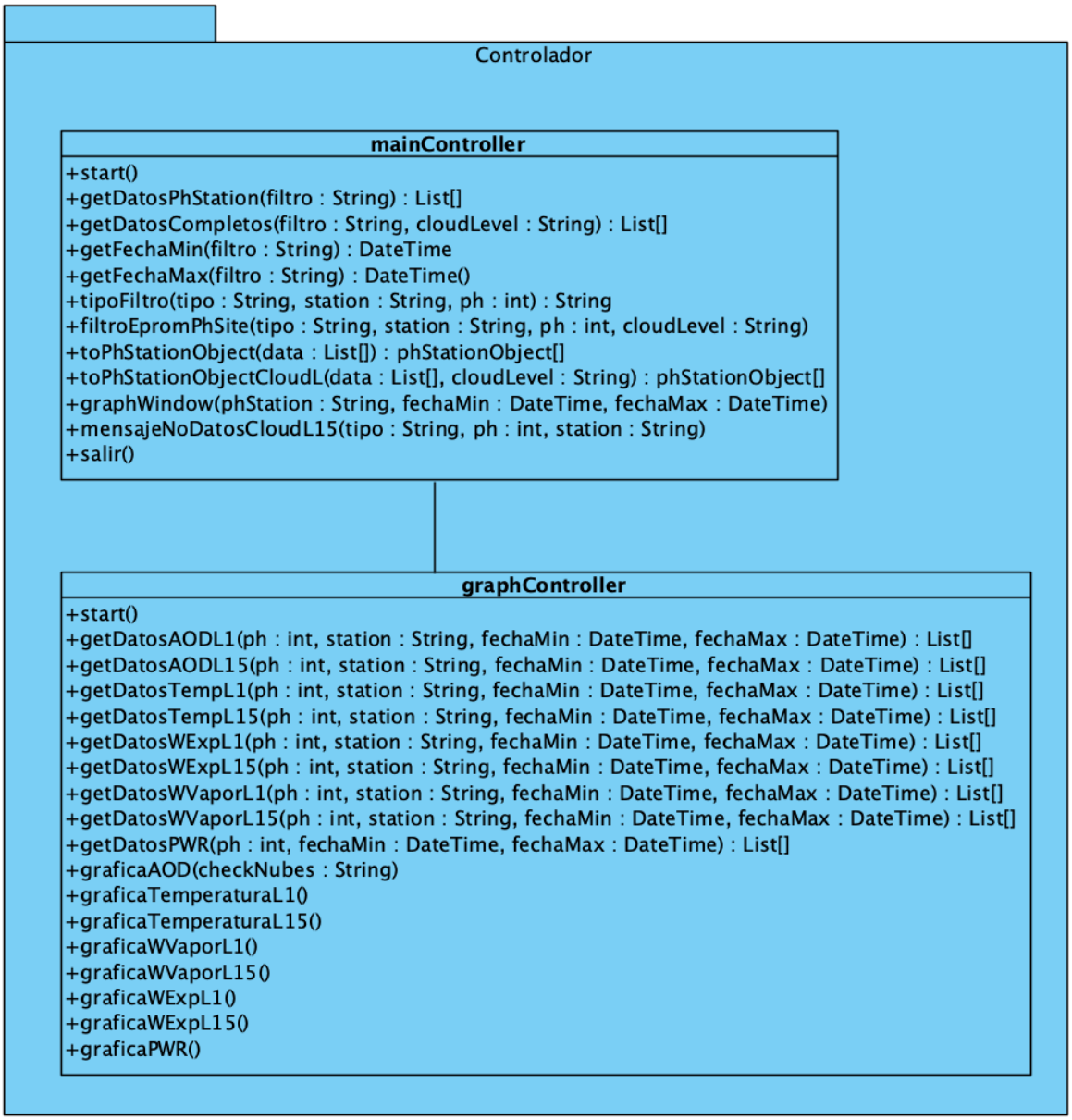


Figura 4.7: Diagrama de clases de Diseño - Controlador.

4.4 Descripción de las clases de Diseño

En esta sección se va a describir cada una de las clases que forman los componentes del MVC vistos en la sección anterior. Se procederá con una breve descripción de la clase y se enumerarán sus atributos.

4.4.1 Modelo

Nombre	**consultasBBDD**
Descripción	Representa la clase intermediaria entre la base de datos y la aplicación, se encarga de realizar las consultas necesarias para poblar el modelo y disponer de la información que hay que representar.
Atributos	database, cursor

Figura 4.8: Descripción clase consultasBBDD.

Nombre	**globales**
Descripción	Esta clase contiene las variables globales de la aplicación: • Datos de conexión con la base de datos. • Código sql para la realización de las consultas. • Errores posibles. • Tipos de filtros posibles para aplicar a las consultas. • Estilos y colores para la representación de las gráficas.
Atributos	• database_host, database_name, user, password. • distinctPhStation, distinctPhStationFiltro, listPhStationDates, listPhStationDatesFiltro, minFecha, minFechaFiltro, maxFecha, maxFechaFiltro, aodL1, aodL15, tempL1, tempL15, waterL1, waterL15, wexpL1, wexpL15, pwr. • err1, err2. • standard, digitalExtended, triple, dualpolar, device, station. • fCanalColors, fCanalMarkers.

Figura 4.9: Descripción clase globales.

Nombre	phStationObject
Descripción	Clase que modela un fotómetro y el uso que se hace de este a lo largo del tiempo.
Atributos	ph, station, dateOfUse, eprom_type, eprom_subtype.

Figura 4.10: Descripción clase phStationObject.

4.4.2 Vista

Nombre	**mainWindow**
Descripción	Ventana principal de la aplicación, se muestra nada más iniciar la aplicación y desde la que comienzan todas las acciones.
Atributos	mainController, tableWidget, quitButton, scrollbar, figura, axis, canvas, toolbar, labelFMin, labelFMax, auxEprom, auxNubes, labelDevice, device, labelSite, site, standardBtn, digitalBtn, tripleBtn, dualpolarBtn, nubes1Btn, nubes15Btn.
Operaciones	
plotUsoPh()	
Descripción	Se encarga de generar la gráfica de uso de los fotómetros.
Parámetros	• ph: identificador del fotómetro • datosPhSt: lista con los identificadores y nombres de los fotómetros, sin repetirse. • datosCompletos: lista con la información completa de los fotómetros, fechas, eprom_type, eprom_subtype. • fechaMin: fecha mínima del conjunto de datos. • fechaMax: fecha máxima del conjunto de datos. • cloudLevel: campo para ver si está aplicado el cloud level para los datos.
getXMin(), getXMax()	
Descripción	Recupera de la gráfica el valor mínimo y máximo, cada función respectivamente, del eje X.
getYMin(), getYMax()	
Descripción	Recupera de la gráfica el valor mínimo y máximo, cada función respectivamente, del eje Y.
sliderValue()	
Descripción	Acción del slider para navegar por la gráfáica.
changeTable(), changeRow()	
Descripción	Modifican y pueblan la tabla de los fotómetros, respectivamente, tras

	realizar alguna operación de navegabilidad sobre la gráfica.
fechaEvent()	
Descripción	Función que detecta el evento de cambio en la gráfica y se encarga de pasar el control de cambio de fecha y de las tablas a sus respectivas funciones.
epromClicked()	
Descripción	Acción de seleccionar alguno de los botones de filtrar por eprom_type/subtype.
nubesClicked()	
Descripción	Acción de seleccionar alguno de los botones de filtrar por cloud level.
enterDevSite()	
Descripción	Acción de pulsar enter en los campos device y site, los cuales son campos de entrada de datos.
compruebaSite(), compruebaDevice()	
Descripción	Se encargan de verificar si hay datos introducidos en los campos device y site, y de si tienen un formato correcto.
phClicked()	
Descripción	Acción de pulsar dos veces sobre un elemento de la tabla fotómetros.
limpiaPlot()	
Descripción	Acción de reiniciar la gráfica y volver a dar formato a sus ejes.
limpiaTabla()	
Descripción	Acción de reiniciar y limpiar la tabla de fotómetros.
reiniciaCloudLevel()	
Descripción	Se encarga de cambiar el estado de filtro cloud level al de por defecto.
quitClicked()	
Descripción	Acción de pulsar el botón salir de la aplicación.

Figura 4.11: Descripción clase mainWindow.

Nombre	graphWindow
Descripción	Ventana encargada de la visualización de las medidas de un fotómetro. Esta es iniciada cuando un usuario en la mainWindow comienza la acción de graficar los datos de un fotómetro.
Atributos	praphController, fMin, fMax, check, checkNubes, figura, axis, canvas, toolbar, aodBtn, WExpBtn, waterBtn, tempBtn, pwrBtn, cloudL1Btn, cloudL15Btn, hebBtn, daBtn, saBtn, stcBt, slcBtn, langleyBtn, sendSBtn.
Operaciones	
aod_clicked()	
Descripción	Acción de pulsar sobre el botón aod.
wexp_clicked()	
Descripción	Acción de pulsar sobre el botón wexp.
waterVapor_clicked()	
Descripción	Acción de pulsar sobre el botón vapor de agua.
temperatura_clicked()	
Descripción	Acción de pulsar sobre el botón de temperatura.
pwr_clicked()	
Descripción	Acción de pulsar sobre el botón pwr.
filtroNubesL1(), filtroNubesL15()	
Descripción	Acción de cambiar el filtro cloud level de 1.0 a 1.5 y a la inversa, respectivamente cada función.
cambiarFiltro()	
Descripción	Comprueba si el filtro que se quiere aplicar es el mismo que el anterior, este está aplicado a los eprom_type/subtype por el cloud level.
plotSimpleData()	
Descripción	Se encarga de generar la gráfica de datos simples a partir de los datos que le son enviados. Con datos siemples me refiero a: temperatura, vapor de agua y PWR.
Parámetros	

	• dataX: valores en el eje X para la representación. • dataY: valores en el eje Y para la representación. • tipo: campo con el tipo de datos que se va a representar; temperatura, vapor de agua o PWR.
plotWExp()	
Descripción	Se encarga de generar la gráfica de de los datos relativos a las medidas WExp.
Parámetros	• dataX: valores en el eje X para la representación. • dataY440: valores en el eje Y, de alpha_440, para la representación. • dataY380: valores en el eje Y, de alpha_380, para la representación.
plotChannelData()	
Descripción	Se encarga de generar la gráfica de las medidas que disponen de diferentes canas, bandas espectrales. En nuestro caso las medidas de AOD.
Parámetros	• dataX: valores en el eje X para la representación. • dataY: valores en el eje Y para la representación. • yMin: valor mínimo de la representación en el eje Y. • yMax: valor máximo de la representación en el eje Y. • lowerErr: valores mínimos de la media para representar el error. • upperErr: valores máximos de la media para representar el error. • desvS: valores para la representación de la desviación estándar. • banda: identificador de la banda para realizar la leyenda de la gráfica. • color: color para la representación de cada conjunto de medidas, separadas por las bandas espectrales. • marker: aspecto de los puntos a la hora de representar cada conjunto de medidas, separadas por las bandas espectrales.
getXMin(), getXMax()	
Descripción	Recupera de la gráfica el valor mínimo y máximo, cada función respectivamente, del eje X.
fechaEvent()	
Descripción	Función que detecta el evento de cambio en la gráfica, para ajustar el formato de la fecha a los requisitos del cliente.
msjNoFiltroPwr()	

Descripción	Mensaje que indica que no se puede aplicar el cloud level, 1.0 o 1.5, al conjunto de medidas PWR.
msjNoDatos()	
Descripción	Mensaje que indica que no hay datos para la solicitud realizada, normalmente con el campo PWR.
limpiaPlot()	
Descripción	Acción de reiniciar la gráfica y volver a dar formato a sus ejes.

Figura 4.12: Descripción clase graphWindow.

Nombre	**NavigationToolbar**
Descripción	Clase NavigationToolbar2 de matplotlib.backend_bases modificada para la ventana mainWindow. Mirar Apéndice II para la explicación.
Atributos	canvas, parent, nav_stack, coordinates, actions, toolitems.

Figura 4.13: Descripción clase NavigationToolbar.

4.4.3 Controlador

Nombre	**mainController**
Descripción	Clase que actúa como fachada entre la vista y el modelo.
Atributos	mainWindow, db, cursor
Operaciones	
start()	
Descripción	Se encarga de obtener los datos por defecto del uso de los fotómetros y de inicializar la mainWindow.
tipoFiltro()	
Descripción	Determina el filtro que hay que aplicar para obtener los datos solicitados por el usuario.
filtroEpromPhSite()	
Descripción	Responde a la solicitud de la mainWindow de aplicar una serie de filtros a los datos. Solicita los datos y los devuelve para su representación.
Parámetros	• tipo: eprom_type/subtype. • ph: identificador del fotómetro. • station: localización del fotómetro. • cloudLevel: tipo de filtro cloud level, 1.0 o 1.5.
toPhStationObject()	
Descripción	Transforma la lista obtenida del uso de fotómetros, junto con sus datos, y devuelve una lista del tipo phStationObject. Esto se hace para su correcta graficación.
Parámetros	• data: lista obtenida de realizar la consulta de obtener la información de uso de los fotómetros.
graphWindow()	
Descripción	Se encarga de inicializar un nuevo graphController, para que gestione la funcionalidad de graphWindow.
Parámetros	• phStation: identificador del fotómetro y localización del fotómetro.

	• fechaMin: fecha de inicio de toma de muestras. • fechaMax: fecha fin de toma de muestras.
msjNoDatosCloud15()	
Descripción	Mensaje que indica que no hay datos para el filtro aplicado con cloud level 1.5.
salir()	
Descripción	Se encarga de finalizar la ejecución de la aplicación.

Figura 4.14: Descripción clase mainController.

Nombre	**graphController**
Descripción	Se encarga de la gestión de las ventanas “graphWindow”. Es decir, el mainController delega en él este conjunto de funcionalidades para la comunicación entre la vista y el modelo en dichas ventanas.
Atributos	graphWindow, db, cursor, ph, station, fechaMin, fechaMax
Operaciones	
start()	
Descripción	Se encarga de obtener los datos por defecto, aod, para los parámetros introducidos, y de inicializar la graphWindow.
Parámetros	• ph: identificador del fotómetro. • station: localización del fotómetro. • fechaMin: fecha de inicio de toma de muestras. • fechaMax: fecha fin de toma de muestras.
graficaAod()	
Descripción	Solicita los datos de las medidas AOD y los manda a la vista para su representación.
Parámetros	• checNubes: filtro cloud level, 1.0 o 1.5.
graficaTempL1(), graficaTempL15()	
Descripción	Solicita los datos de las medidas de temperatura, según el cloud level una función u otra, y los manda a la vista para su representación.

graficaWVL1(), graficaWVL15()	
Descripción	Solicita los datos de las medidas de vapor de agua, según el cloud level una función u otra, y los manda a la vista para su representación.
graficaWExpL1(), graficaWExpL15()	
Descripción	Solicita los datos de las medidas WExp, según el cloud level una función u otra, y los manda a la vista para su representación.
graficaPWR()	
Descripción	Solicita los datos de las medidas PWR y los manda a la vista para su representación.

Figura 4.15: Descripción clase graphController.

4.5 Realización de los diagramas de secuencia

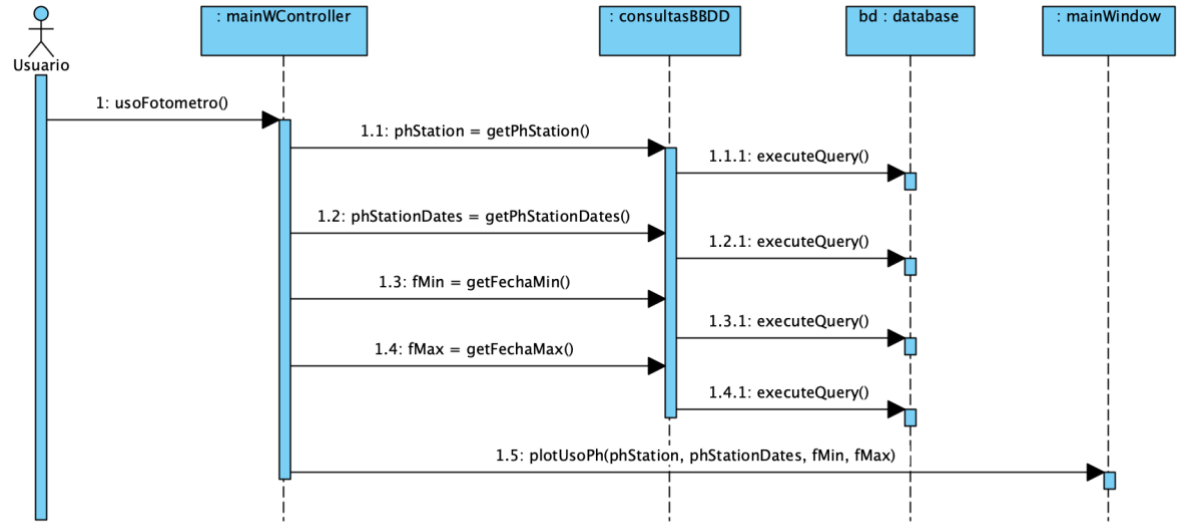


Figura 4.16: Diagrama de secuencia CU-01 Uso de fotómetro.

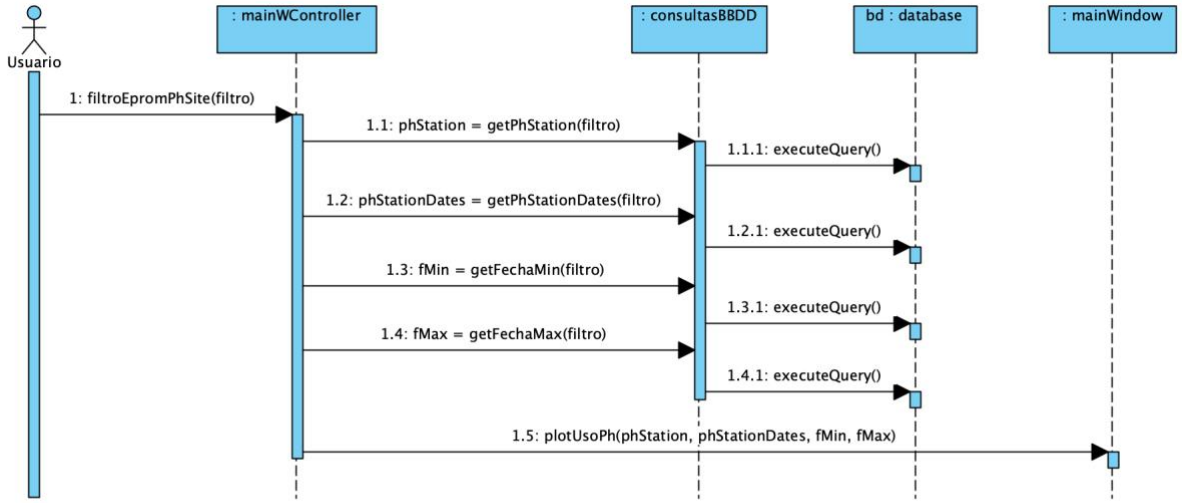


Figura 4.17: Diagrama de secuencia CU-02 Filtrar Fotómetro.

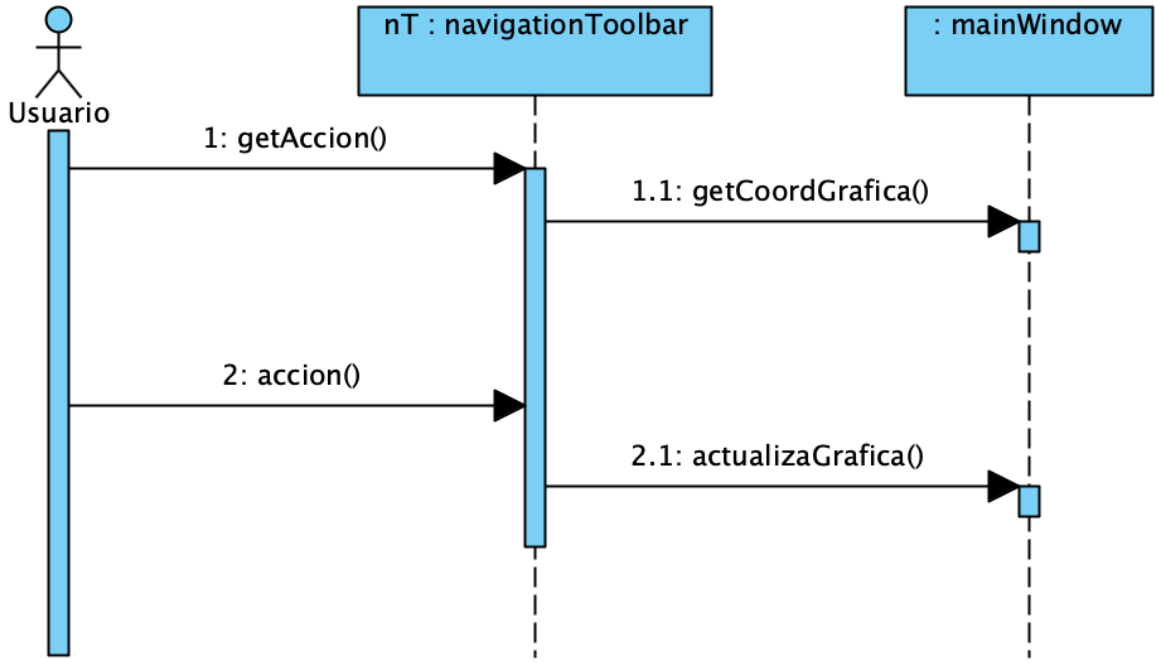


Figura 4.18: Diagrama de secuencia CU-03 Navegabilidad uso Fotómetro.

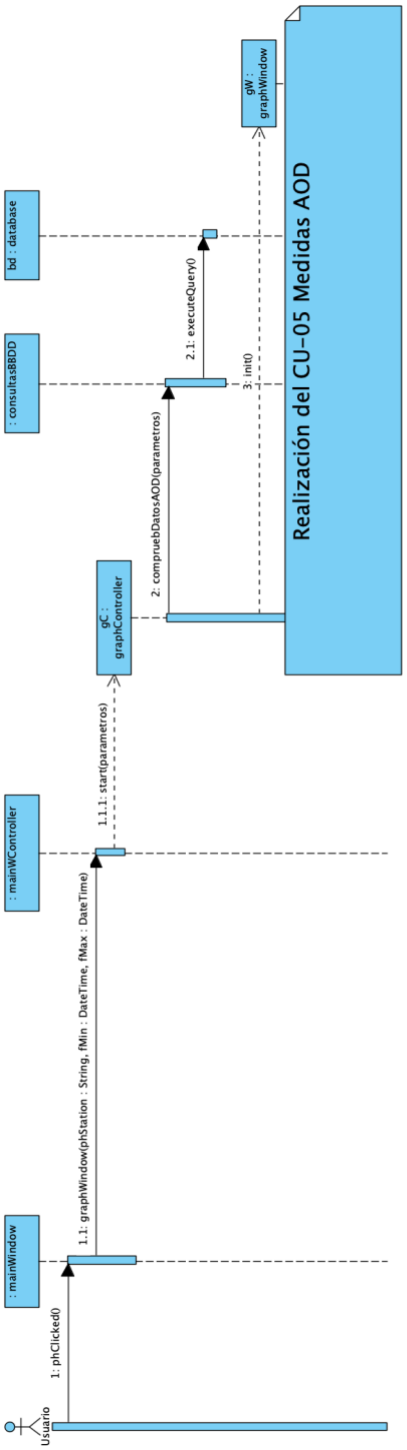


Figura 4.19: Diagrama de secuencia CU-04 Ventana medidas de Fotómetro.

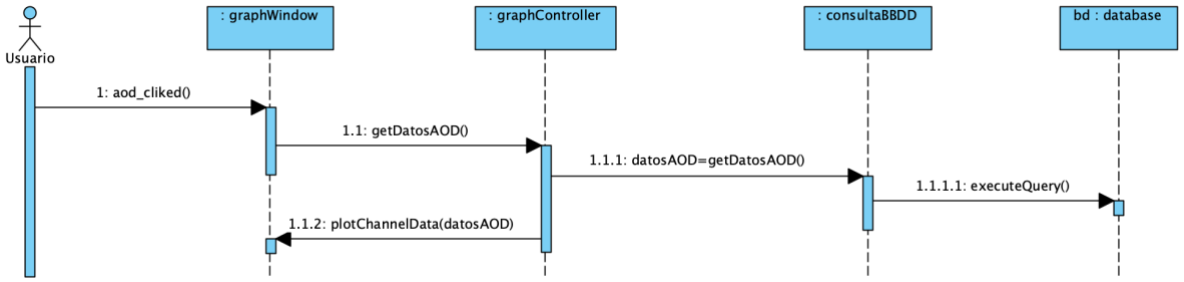


Figura 4.20: Diagrama de secuencia CU-05 Medidas AOD.

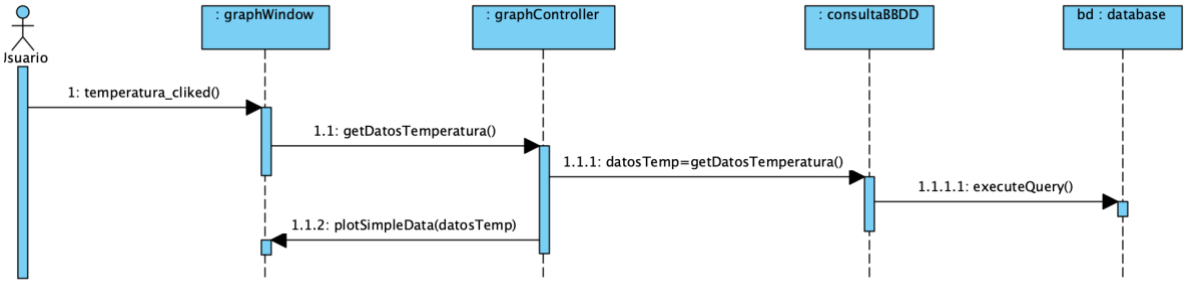


Figura 4.21: Diagrama de secuencia CU-06 Medidas Temperatura.

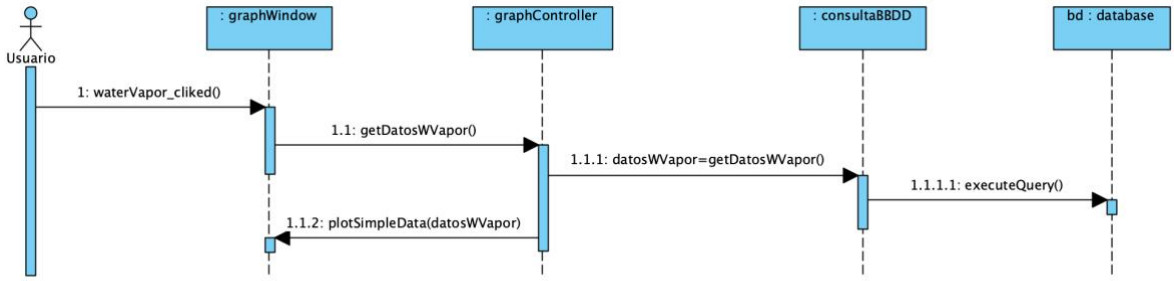


Figura 4.22: Diagrama de secuencia CU-07 Medidas Vapor de Agua.

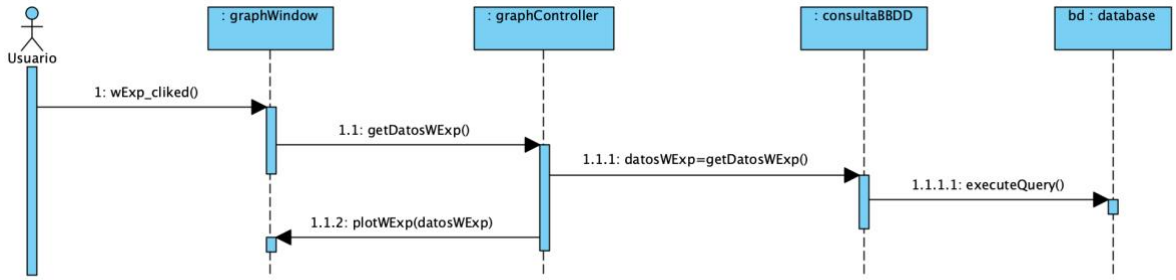


Figura 4.23: Diagrama de secuencia CU-08 Medidas WExp.

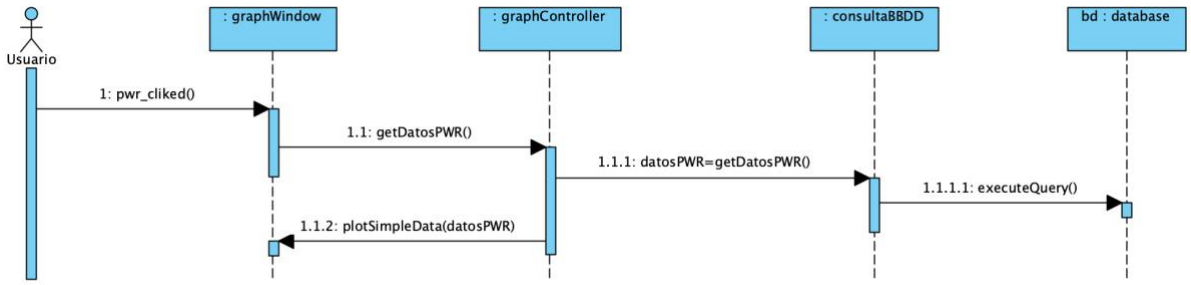


Figura 4.24: Diagrama de secuencia CU-09 Medidas PWR.

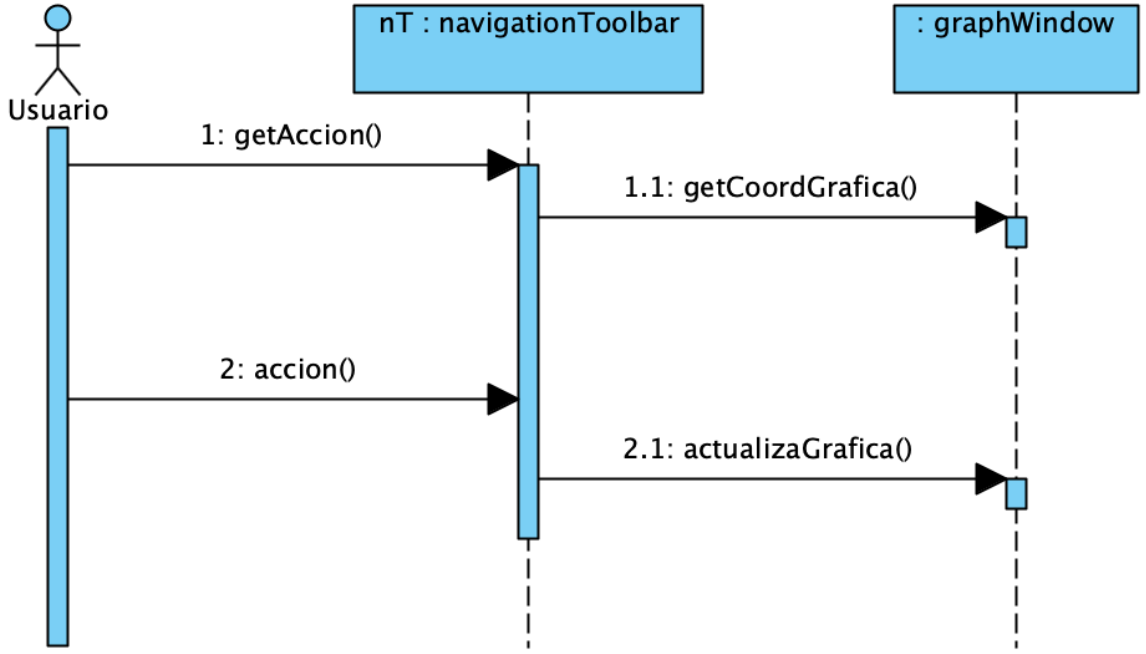


Figura 4.25: Diagrama de secuencia CU-10 Navegabilidad gráficas de medida.

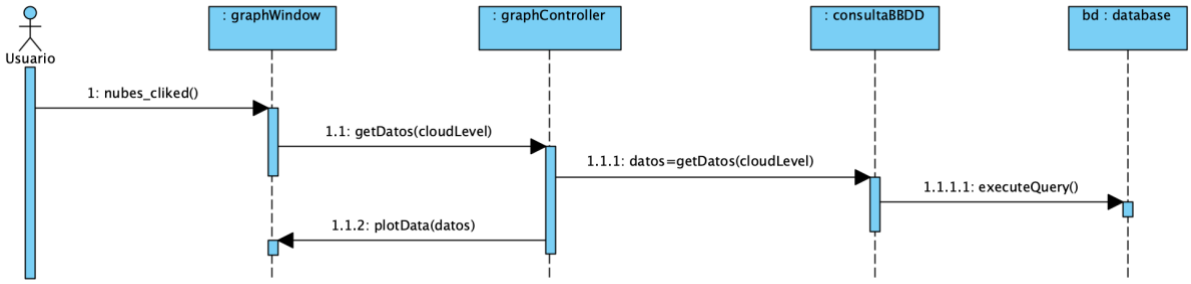


Figura 4.26: Diagrama de secuencia CU-11 Filtrado de Nubes.

Capítulo 5

Implementación

Resumen: *Este capítulo hace referencia a la implementación. Puesto que no conviene plasmar el código entero de la aplicación, se describirán tanto algunas de las librerías más destacadas como algunas de las funciones que estas nos brindan para el desarrollo de nuestra aplicación.*

5.1 MatplotLib

Matplotlib es una biblioteca que permite representar gráficas a partir de de diferentes tipos de datos, variables, listas, arrays… Esto lo convierte en uno de los pilares más importantes de esta aplicación, la cual está enfocada principalmente a la representación de la información recopilada de CÆLIS.

5.5.1 Pyplot

Es una colección de funciones que permiten que matplotlib puede funcionar de una forma similar a la famosa herramienta MATLAB. Gracias a este conjunto de funcionalidades se ha podido representar diferentes tipos de datos en esta aplicación. A continuación, se expondrán las funciones más utilizadas y los parámetros que utiliza (Se considerará matplotlib como plt):

- *plt.plot(X, Y, formato, label=label)*

Esta función ha sido utilizada para representar medidas simples, como por ejemplo la temperatura o el vapor de agua. Parámetros:

  - X: Valor, o lista de valores, del eje x.
  - Y: Valor, o lista de valores, del eje y.
  - Formato: estilo y color de la representación de los datos. Ejemplo: “ro-”, esto significa color rojo (r), con los puntos grandes (o) y la línea que une los puntos contínua (-).
  - Label: ponerle un identificador al conjunto de datos representados, por si se quiere representar la leyenda en la gráfica.

A continuación, se puede observar un pequeño ejemplo del funcionamiento de esta función.

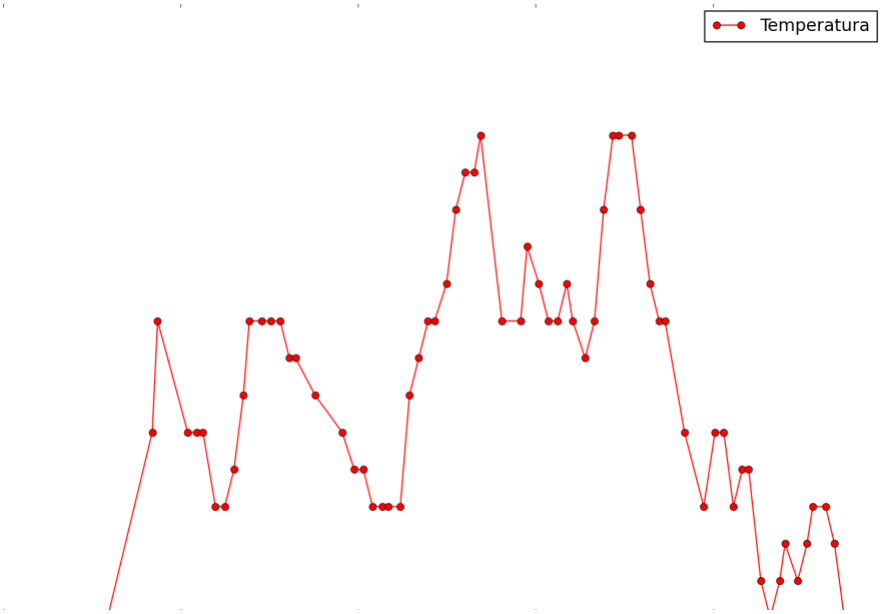


Figura 5.1: Ejemplo plt.plot.

- *plt.errorbar(X, Y, yErr=[lowerE, upperE] , color = color, marker = marker, label = label)*

Esta función ha sido utilizada para representar datos más complejos, como las medidas de AOD, ya que permite representar cada medida con su error, o en esta aplicación, con el valor mínimo y máximo del conjunto de las medidas recogidas en una misma fecha.

- X: Valor, o lista de valores, del eje x.
- Y: Valor, o lista de valores, del eje y.
- yErr = [lowerE, upperE]: Valor, o lista de valores, que forman el error de la medida. En nuestro caso al ser un error asimétrico, lowerE corresponde con el error menor y upperE corresponde con el error mayor.
- Color: color para la representación de los datos.
- Marker: forma para representar cada dato. Ejemplos: “o”, “+”, “D” (diamante)...
- Label: ponerle un identificador al conjunto de datos representados, por si se quiere representar la leyenda en la gráfica.

A continuación, se puede observar un pequeño ejemplo del funcionamiento de esta función.

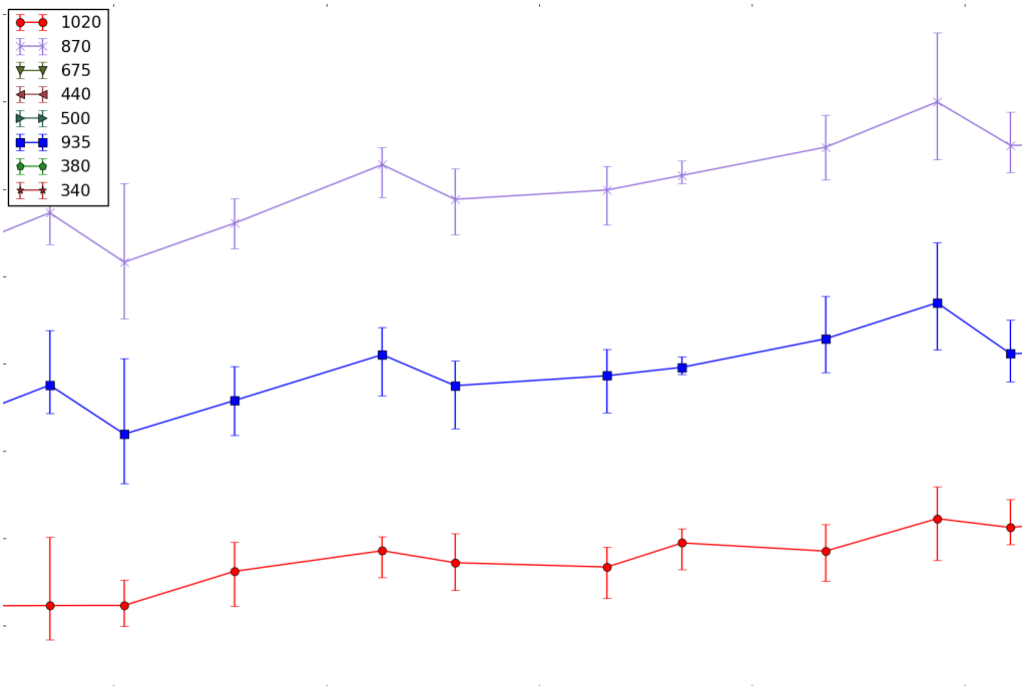


Figura 5.2: Ejemplo plt.errorbar.

- *plt.legend()*

Esta función se utiliza para representar la leyenda de la gráfica, como se puede observar en las Figuras 5.1 y 5.2. Simplemente con incluir en la representación de los datos el parámetro label se generará automáticamente al invocarlo en el código.

5.5.2 FigureCanvasQTAgg

FigureCanvasQtAgg es un widget que proporciona matplotlib para poder embeber de forma sencilla las gráficas dentro de nuestra interfaz QT. Pero no todo son ventajas, también tiene un problema importante, y esque este widget lo que está mostrando es una imágen de la representación. Esto quiere decir que desconoce la posición de los objetos de la trama, sólo conoce las coordenadas de x y los movimientos del ratón sobre este. Pero para completar la funcionalidad matplotlib nos introduce el siguiente widget de esta sección, el cual se complementa a la perfección con este, NavigationToolbar2QT.

5.5.3 NavigationToolbar2QT

NavigationToolbar2QT es un widget que funciona como una barra de herramientas que nos presta diversas operaciones sobre la gráfica representada. Ver el Apéndice II para ver la configuración y ajuste de este widget. Las principales operaciones son:

- Home: reiniciar todas las operaciones realizadas.
- Back/Forward: deshacer/rehacer las operaciones realizadas.
- Pan: desplazarse por los ejes de la gráfica.
- Zoom: realizar zoom sobre una zona seleccionada.
- Save: captura y permite guardar la actual disposición y contenido de la gráfica.

5.5.5 Event Handling

Tras ver las diferentes acciones que se pueden realizar sobre nuestra gráfica, vistas en el punto anterior, necesitamos una forma de actualizar la interfaz una vez se produzcan estas. Para ello matplotlib nos proporciona una buena API de manejo de eventos, lo que nos permite detectar cambios sobre la gráfica o incluso en los propios ejes de esta. En esta aplicación se ha utilizado este manejador en dos ocasiones, las cuales se pueden ver a continuación:

- *FigureCanvas.mpl_connect("draw_event", evento())*

Esta línea nos permite conectar la gráfica, embebida en la FigureCanvas, con la acción evento() ante cualquier acción de redibujar (“draw_event”).

- *callbacks.connect('xlim_changed', evento())*

Esta línea crea un callback que conecta la gráfica con la acción evento() ante cualquier redimensión del eje X de la gráfica (“xlim_change”).

5.5.6 PollyCollections

Esta API de matplotlib permite el dibujo eficiente de grandes colecciones de objetos que comparten una gran parte de sus propiedades. A continuación, podemos ver un ejemplo de su implementación:

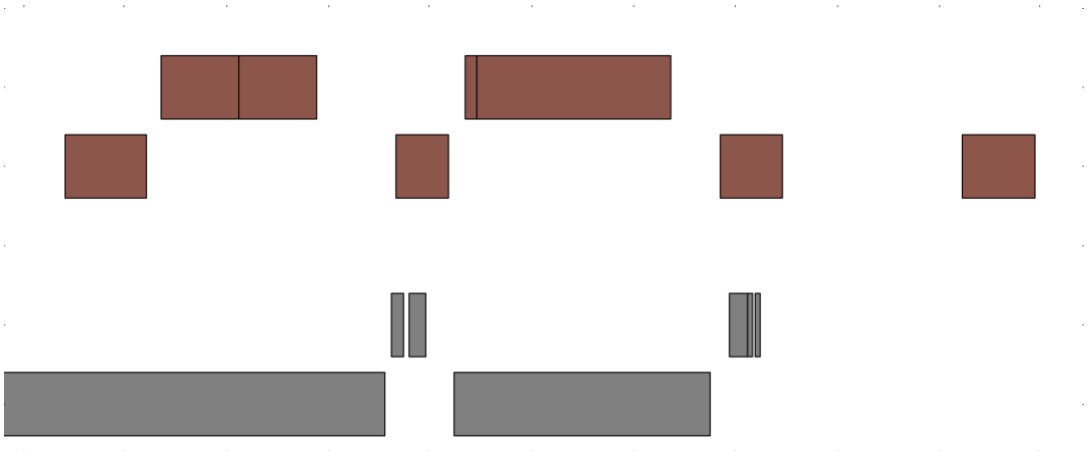


Figura 5.3: Ejemplo implementación PollyCollection.

Como se puede observar en la Figura 5.4 también nos permite seleccionar los colores de cada figura. Pero ¿cómo representa estos polígonos?, la respuesta es sencilla, calcula la posición de los 4 vértices en función de los valores del eje Y (el cual en nuestro caso es contínuo) y del eje X (cuyo valor es el que cambia).

5.2 Numpy

Numpy es un paquete fundamental para el desarrollo aplicaciones científicas con python. Además de sus obvios usos científicos, NumPy también se puede utilizar como un eficiente contenedor multidimensional de datos genéricos. En esta aplicación se ha utilizada principalmente para calcular la desviación estándar de una serie de conjuntos de datos. A continuación, podemos ver cómo es la implementación:

```
medidas = numpy.array([aodMinY, aodMaxY, aodMedY])
desvS = numpy.std(medidas, axis=0)
```

Figura 5.4: Ejemplo implementación Numpy - Desviación estándar.

En la Figura 5.5 podemos observar cómo numpy calcula la desviación estándar de cada valor en la misma posición de los vectores aodMinY, aodMaxY y aodMedY.

Capítulo 6

Pruebas

Resumen: *En este capítulo se trata de probar el correcto funcionamiento de la aplicación y mostrar las pruebas que se han realizado para ello.*

6.1 Definición teórica

La realización de pruebas sobre una aplicación es una de las partes más importantes del control de calidad. Cada prueba está encargada de comprobar y determinar el correcto funcionamiento del sistema en estudio. Para la realización de las pruebas vamos a dividirlas en dos tipos:

- Pruebas de caja negra: estas son típicamente las pruebas software funcionales, en estas se comprueba el comportamiento del sistema según está descrito en los requisitos y casos de uso. Esto quiere decir que se verifica la funcionalidad sin tener en cuenta la estructura interna del código o los detalles de la implementación, es decir, nos centramos en la entrada y salida del sistema.

- Pruebas de caja blanca: estas son típicamente las pruebas de software estructurales. Para este tipo de pruebas se utiliza el concepto de cobertura, normalmente en forma de porcentaje, que nos indica si cada uno de los caminos, método o módulos de la aplicación son recorridos alguna vez, y analiza todos los bucles en sus límites operacionales. Para ello existen herramientas de cálculo de la cobertura en el código.

6.2 Pruebas de caja Negra

Este conjunto de pruebas se ha realizado sobre todos y cada uno de los casos de uso. Para su documentación se ha tenido en cuenta el caso de uso, su descripción, el resultado esperado, el obtenido y si disponen de algún flujo alternativo de salida.

CU-01	**Uso de fotómetro.**
Descripción	Representar la gráfica de uso de los fotómetros.
Salida esperada	Se inicia la aplicación con la ventana uso de fotómetros con los valores de filtro por defecto.
Salida	Correcta.
Salida alternativa	• Sino hay conexión a internet, o no se establece conexión con la base de datos, la aplicación no inicia y muestra un mensaje de error en la terminal. RESULTADO: Correcto.

Figura 6.1: Pruebas CU-01 Uso de fotómetro.

CU-02	**Filtrar fotómetro.**
Descripción	Filtrar los fotómetros a analizar según su nombre, estación o tipo.
Salida esperada	Una nueva gráfica con el uso de los fotómetros acorde a los filtros establecidos por el usuario.
Salida	Correcta.
Salida alternativa	• Si no hay resultados para el filtro, la tabla de fotómetros y la gráfica quedan en blanco y se muestra un mensaje indicando que no hay coincidencias. RESULTADO: Correcto. • Si no hay datos al intentar aplicar el filtro aod(1.5), se mostrará un mensaje en el que se comunica que no hay datos para el dicho filtro. RESULTADO: Correcto. • Si los datos introducidos en el campo device no son un número entero, se muestra un mensaje indicando que el campo device debe ser un número. RESULTADO: Correcto. • Si los datos introducidos en el campo site son un número, se muestra un mensaje indicando que este campo no puede ser numérico. RESULTADO: Correcto.

Figura 6.2: Pruebas CU-02 Filtrar fotómetro.

CU-03	Navegabilidad uso fotómetro.
Descripción	Navegabilidad en la gráfica uso de fotómetro, realizar zoom, desplazarse por los valores...
Salida esperada	La gráfica y la tabla de fotómetros actualizadas respecto la operación que se quiere realizar.
Salida	Correcta.
Salida alternativa	• Si se selecciona un zoom demasiado pequeño respecto del eje Y, la tabla de fotómetros y la gráfica quedan en blanco. RESULTADO: Correcto

Figura 6.3: Pruebas CU-03 Navegabilidad uso fotómetro.

CU-04	Ventana medidas de fotómetro.
Descripción	Se muestra una nueva ventana destinada a la graficación de las medidas de un fotómetro en concreto.
Salida esperada	Ventana con las medidas aod, con filtro 1.0 por defecto, para el fotómetro y fechas seleccionadas.
Salida	Correcta.
Salida alternativa	• Si no hay datos para ese fotómetro y fechas, se muestra un mensaje indicando que no hay datos disponibles. RESULTADO: Correcto

Figura 6.4: Pruebas CU-04 Ventana medidas de fotómetro.

CU-05	Medidas AOD.
Descripción	Representación de las medidas AOD relativas a un fotómetro.
Salida esperada	Se actualiza la gráfica con las medidas de aod, en función del filtro cloud level seleccionado.
Salida	Correcta.
Salida alternativa	

Figura 6.5: Pruebas CU-05 Medidas AOD.

CU-06	Medidas Temperatura
Descripción	Representación de las medidas de Temperatura relativas a un fotómetro.
Salida esperada	Se actualiza la gráfica con las medidas de temperatura, en función del filtro cloud level seleccionado.
Salida	Correcta.
Salida alternativa	

Figura 6.6: Pruebas CU-06 Medidas Temperatura.

CU-07	Medidas Vapor de Agua
Descripción	Representación de las medidas de Vapor de Agua relativas a un fotómetro.
Salida esperada	Se actualiza la gráfica con las medidas de vapor de agua, en función del filtro cloud level seleccionado.
Salida	Correcta.
Salida alternativa	

Figura 6.7: Pruebas CU-07 Medidas Vapor de Agua.

CU-08	Medidas WExp
Descripción	Representación de las medidas WExp relativas a un fotómetro.
Salida esperada	Se actualiza la gráfica con las medidas WExp, en función del filtro cloud level seleccionado.
Salida	Correcta.
Salida alternativa	

Figura 6.8: Pruebas CU-08 Medidas WExp.

CU-09	Medidas PWR
Descripción	Representación de las medidas PWR relativas a un fotómetro.
Salida esperada	Se actualiza la gráfica con las medidas PWR.
Salida	Correcta.
Salida alternativa	• Si no hay resultados de PWR, la gráfica queda en blanco y se muestra un mensaje indicando que no hay datos PWR disponibles para ese fotómetro en esas fechas. RESULTADO: Correcto. • Si se intenta aplicar cualquiera de los dos filtros, L1.0 o L1.5, a las medidas PWR se muestra un mensaje indicando que no es posible aplicar el filtro a este tipo de medidas. RESULTADO: Correcto.

Figura 6.9: Pruebas CU-09 Medidas Pwr.

CU-10	Navegabilidad gráfica de medidas.
Descripción	Navegabilidad por la gráfica medidas de un fotómetro, realizar zoom, desplazarse por los valores...
Salida esperada	La gráfica y la tabla de fotómetros actualizadas respecto la operación que se quiere realizar.
Salida	Correcta.
Salida alternativa	

Figura 6.10: Pruebas CU-10 Navegabilidad gráficas de medidas.

CU-11	Filtrado Cloud Level.
Descripción	Filtrado de las medidas de los fotómetros en función de las nubes.
Salida esperada	La gráfica actualizada con las medidas actuales cambiadas de filtro, por ejemplo, de temperatura L1.0 a L1.5 y a la inversa.
Salida	Correcta.
Salida alternativa	• Si se intenta aplicar cualquiera de los dos filtros, L1.0 o L1.5, a las medidas PWR se muestra un mensaje indicando que no es posible aplicar el filtro a este tipo de medidas. RESULTADO: Correcto.

Figura 6.11: Pruebas CU-11 Filtrado Cloud Level.

6.3 Pruebas de caja Blanca

Para realizar estas pruebas se ha utilizado una herramienta que proporciona python llamada coverage. Esta permite ejecutar el código de nuestra aplicación y mientras tanto contabiliza el porcentaje de código que se ha recorrido. En el caso de esta aplicación se ha podido utilizar de modo condicionada, y con esto quiero decir, como se apreciará en los resultados, que no se va a poder obtener el 100% de todos los módulos en una sóla ejecución. Esto se debe a que las partes del código que no se recorren corresponden con fallos de internet, errores de conexión con la base de datos, y cuestiones similares, lo que conlleva cerrar la ejecución de la aplicación. A continuación, se muestra el resultado de la prueba realizada de una ejecución completa de la aplicación:

```
MacBook-Pro-de-Jesus:Aeronet jesus$ coverage run src/main.py
SQL Server: ('5.7.29-0ubuntu0.18.04.1',)
MacBook-Pro-de-Jesus:Aeronet jesus$ coverage report
Name                                    Stmts   Miss  Cover
-------------------------------------------------------------
src/controlador/graphController.py        219    107    52%
src/controlador/mainWController.py        169     95    44%
src/main.py                                21      3    86%
src/modelo/consultasBBDD.py               202     98    51%
src/modelo/globales.py                     37      0   100%
src/modelo/phStationObject.py              22      7    68%
src/vista/NavigationToolbar.py             89     29    67%
src/vista/graphWindow.py                  254    103    60%
src/vista/mainWindow.py                   452    152    66%
-------------------------------------------------------------
TOTAL                                    1465    594    60%
```

Figura 6.12: Ejecución de la aplicación con Coverage.

En la Figura 6.12, la columna name representa los archivos que componen la aplicación, la columna stmts las líneas de código, la columna miss las líneas de código no cubiertas y la columna cover el porcentaje de líneas cubiertas.

Capítulo 7

Manuales

Resumen: *En este capítulo se mostrará como instalar y utilizar la aplicación.*

7.1 Manual de Instalación

7.1.1 Requisitos para la instalación

Esta aplicación está preparada para ser ejecutada en un entorno Linux. Dicho entorno debe tener, como requisito previo, instalado python. La versión con la que se ha desarrollado esta aplicación de python es la 3.7.3.

7.1.2 Instalación.

Para la instalación se procederá a clonar el directorio desde GitHub, para ello se ejecutará la siguiente línea:

- *$ git clone https://github.com/jesusBGA/TFG.git ruta_descarga/nombre_carpeta*

Una vez descargado, vamos a la carpeta Aeronet donde encontraremos el script que se encarga de ejecutar la aplicación, "*aeronetTFG*". Una vez en la carpeta y teniendo localizado este script, hay que abrirlo y especificar la ruta en la que se ha clonado el repositorio. En concreto la línea que hay que modificar es la siguiente, añdiendo en "*ruta_aplicación*" el path a la carpeta que contiene Aeronet.

- *python3 ruta_aplicacion/Aeronet/src/main.py*

El paso siguiente es concederle permisos de ejecución, lo que conseguiremos con el siguiente comando:

- *$ chmod +x aeronetTFG*

Desde esta misma carpeta procederemos a copiar este archivo en /usr/local/bin para poder ejecutar el script desde cualquier directorio del sistema solamente escribiendo su nombre. El comando es el siguiente:

- $ *cp aeronetTFG /usr/local/bin/*

Por último y por culpa de la codificación que utiliza eclipse como entorno de trabajo debemos modificar alguno de imports que encontramos en los archivos que componen el programa. Se trata simplemente de borrar el "src." de todos los imports que se refieren a un archivo de la aplicación. Hay que realizar cambios en:

- *Aeronet/src/main.py:* 2 líneas
- *Aeronet/src/controlador/mainWController.py:* 5 líneas
- *Aeronet/src/controlador/graphController.py:* 3 líneas
- *Aeronet/src/modelo/consultasBBDD.py:* 1 línea
- *Aeronet/src/vista/mainWindow.py:* 1 línea

A continuación, se muestra un ejemplo de cómo hay que cambiarlo. En la Figura 7.1 se ve cómo estará al clonar el proyecto, y en la Figura 7.2 como debe quedar. El ejemplo es con "*Aeronet/src/controlador/mainWController.py",* en el que hay más líneas a cambiar.

```
import src.modelo.globales as g
import src.vista.mainWindow as v
import src.modelo.consultasBBDD as c
from src.controlador.graphController import graphController
import src.modelo.phStationObject as objecto
```

Figura 7.1: Instalación - mainWController.py sin modificar.

```
import modelo.globales as g
import vista.mainWindow as v
import modelo.consultasBBDD as c
from controlador.graphController import graphController
import modelo.phStationObject as objecto
```

Figura 7.2: Instalación - mainWController.py modificado.

A partir de este punto ya se puede invocar el script "*aeronetTFG*" desde cualquier directorio del sistema.

7.2 Manual de Usuario

Una vez realizada la instalación ya se puede proceder con la ejecución de la aplicación. Una vez ejecutada se mostrará la pantalla principal.

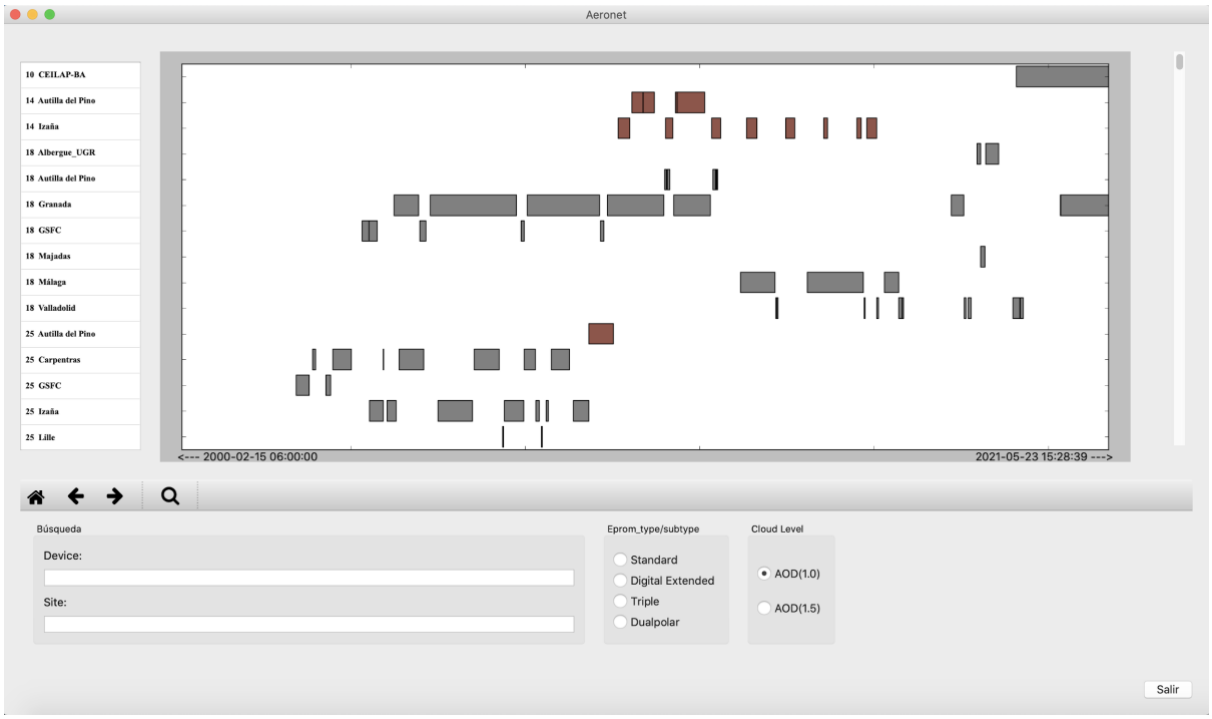


Figura 7.3: Ventana principal de la Aplicación.

Para proceder con la explicación de las funcionalidades que se pueden realizar se va a dividir la ventana en dos, en una primera parte la gráfica junto con las opciones de navegabilidad (Figura 7.4), y en una segunda las operaciones de búsqueda y filtro (Figura 7.5).

Figura 7.4: Ventana principal - Gráfica y operaciones.

Para la Figura 7.4 se va a tomar como referencia la gráfica central, cuyos datos representan la utilización de los fotómetros (eje Y), en una línea temporal. Justo debajo de la gráfica se puede observar la fecha mínima y el máxima de la representación actual. El formato de estas fechas es año-mes-día hora:minutos:segundos.

La tabla de la izquierda de la gráfica representa los fotómetros que se están visualizando, además cada uno de los campos de dicha tabla lleva incorporada, ante un doble click, la representación de los datos de ese fotómetro pulsado, para las fechas mínima y máxima actuales de la gráfica. Estas celdas son las responsables de que se inicie la ventana de graficación de medidas (Figura 7.8). En el caso de no haber datos para ese fotómetro se muestra un mensaje como el siguiente:

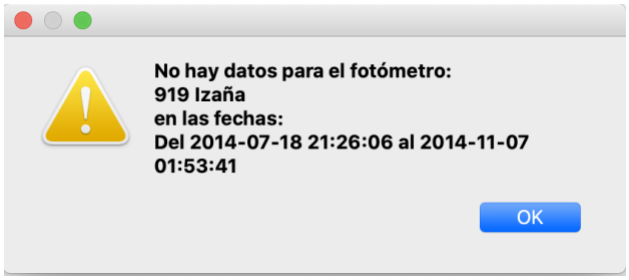


Figura 7.5: Fotómetro sin datos para las fechas introducidas.

A la derecha de la gráfica encontramos un slider el cual nos permite desplazarnos por el conjunto de fotómetros, sólo funciona si hay fotómetros que no están en la representación actual, es decir, por no caber.

Por último, podemos observar debajo de la gráfica el Navigation Toolbar. Este, como se ha explicado en la implementación, permite realizar diferentes operaciones sobre la propia gráfica. De izquierda a derecha realizan las siguientes operaciones:

- Home: reinicia los valores por defecto desde la actualización de los datos que componen la gráfica.
- Back: deshace la última operación realizada sobre la gráfica. No se aplica a los movimientos realizados por el slider.
- Forward: rehace la última operación realizada, es necesario haber deshecho dicha operación para poder aplicar esta operación.
- Zoom: permite seleccionar una zona de la gráfica para ampliarla. Si se selecciona un zoom demasiado pequeño, respecto del eje Y, se muestra el mensaje de la Figura 7.7.

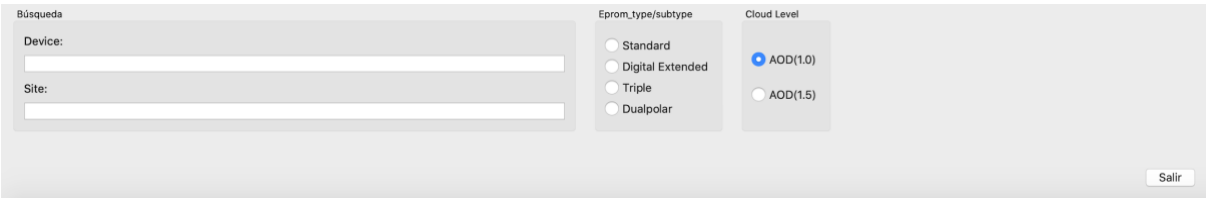


Figura 7.6: Ventana principal - Búsqueda y filtros.

En la Figura 7.6 se observa las operaciones de búsqueda y filtros para aplicar a los datos. Se procederá a explicar cada una de ellas de izquierda a derecha:

- Operaciones de búsqueda: dispone de dos campos de entrada de datos en los que se puede especificar el número de fotómetro y la localización de este. Para aplicar la búsqueda basta con presionar la tecla enter en cualquiera de los dos campos.
    - Device: número de fotómetro, puede ser el número completo o el inicio del número. Por ejemplo, si se presiona un 4 devolverá todos los fotómetros cuyo identificador sea o comience por un 4.
    - Site: localización del fotómetro, como con device la búsqueda se hace en función de esa cadena o de que la localización empiece por esta misma.

- Filtro Eprom_type/subtype: este filtro aparece desactivado ya que se puede querer realizar la búsqueda sin filtro. En el momento que se pulsa cualquiera de los checkbox se realiza la búsqueda por ese tipo, si se elimina la pulsación se elimina el filtro.

- Filtro Cloud level: este filtro tienes 2 opciones, el aod(1.0) es el de por defecto y debido a condiciones de rendimiento de la aplicación no se aplica a la hora de realizar las búsquedas. El aod(1.5) permite buscar los fotómetros cuyo cloud screening es “cloud_free” o “restoration”, se recomienda aplicarlo sólo cuando haya un número no excesivamente de fotómetros para la búsqueda.

- Botón salir: cierra todas las ventanas de la aplicación, es decir, se finaliza la ejecución.

Ante cualquier operación de búsqueda o filtro sin datos se mostrará el siguiente mensaje:

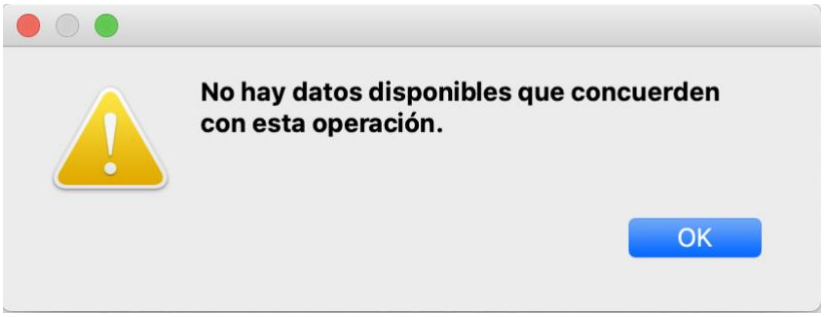


Figura 7.7: Sistema sin datos para la operación realizada.

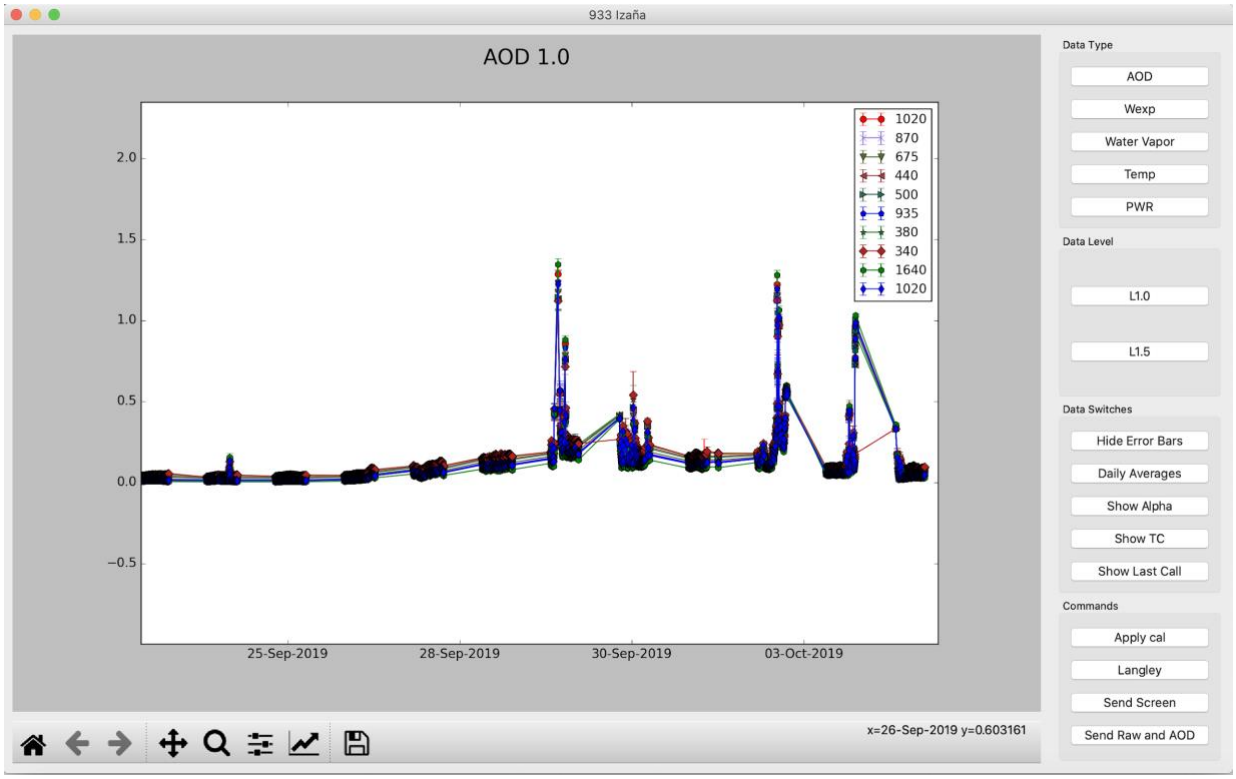


Figura 7.8: Ventana de graficación de las medidas.

En la Figura 7.8 se ve la ventana de graficación de las medidas del fotómetro, la cual se inicia tras pulsar un fotómetro de la tabla de la pantalla principal. En el título de la pantalla se puede observar el identificador y localización del fotómetro seleccionado. La gráfica representa por defecto al iniciarse las medidas AOD con un cloud level 1.0.

Para proceder con la explicación de las funcionalidades que se pueden realizar se va a dividir la ventana en dos, en una primera parte la gráfica junto con las opciones de navegabilidad (Figura 7.9), y en una segunda las operaciones de cambio de medida y filtro cloud level (Figura 7.10).

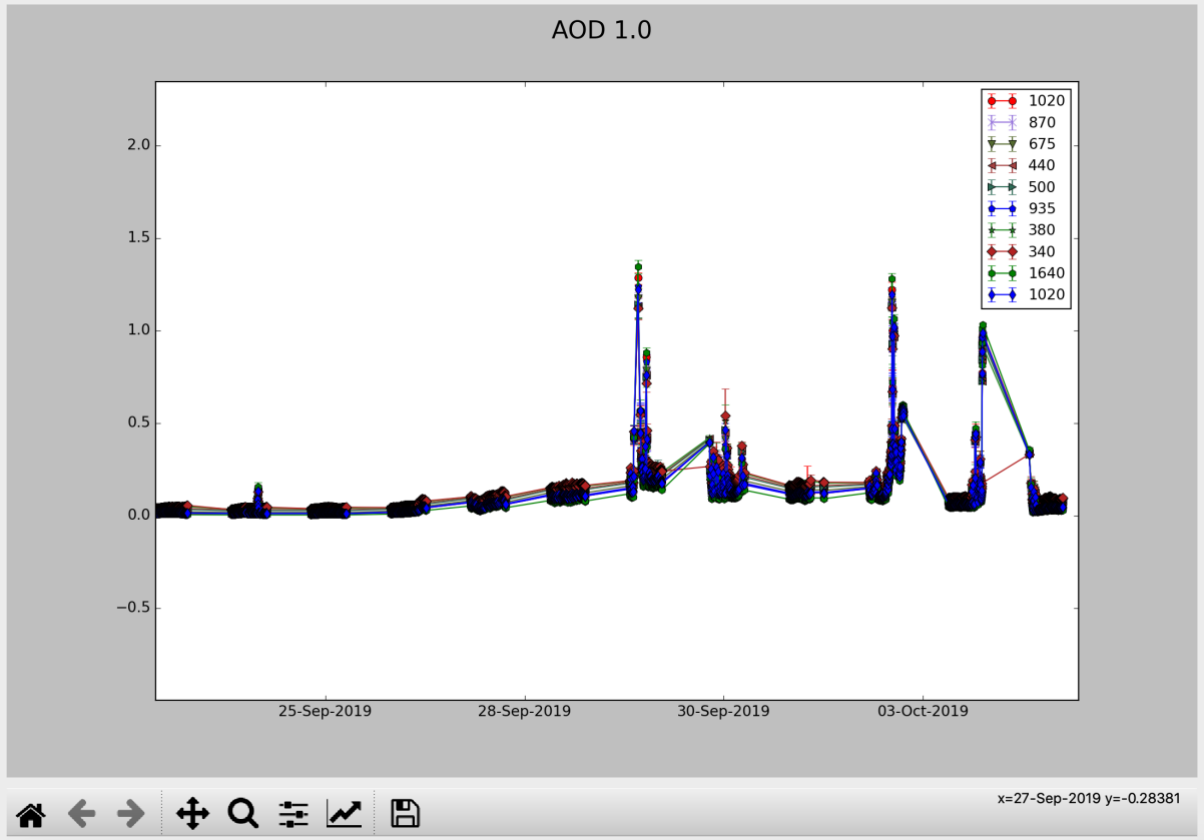
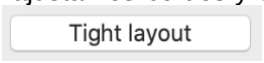


Figura 7.9: Ventana graficación - Gráfica y operaciones.

Para describir la Figura 7.9 se tomará como referencia la gráfica con la representación de las medidas. El eje X de cada una de las medias corresponde con la fecha y el eje Y con la medida que se está representando. En este ejemplo se puede ver la leyenda con los tipos de bandas espectrales a la derecha, pero esto no tiene que cumplirse ya que está programado para que se coloque en el lugar óptimo.

Debajo de la gráfica se sitúa el Navigation Toolbar, cuyos iconos de izquierda a derecha realizan las siguientes operaciones:

-  Home: reinicia los valores por defecto desde la actualización de los datos que componen la gráfica.
-  Back: deshace la última operación realizada sobre la gráfica. No se aplica a los movimientos realizados por el slider.
-  Forward: rehace la última operación realizada, es necesario haber deshecho dicha operación para poder aplicar esta operación.
-  Move: permite desplazarse por los ejes de la gráfica arrastrando el ratón.
-  Zoom: permite seleccionar una zona de la gráfica para ampliarla.

-  Configure subplots: permite ajustar los bordes y espacios, no está sincronizada con la aplicación, sólo está por el botón 

que aparece al pulsar esta opción, la cual hace que se aumente el tamaño de la gráfica.

-  Edit axis: permite cambiar el mínimo y máximo de los ejes X e Y.

-  Save: permite guardar la gráfica actual como una imágen, es decir, realiza una captura de pantalla de la gráfica.

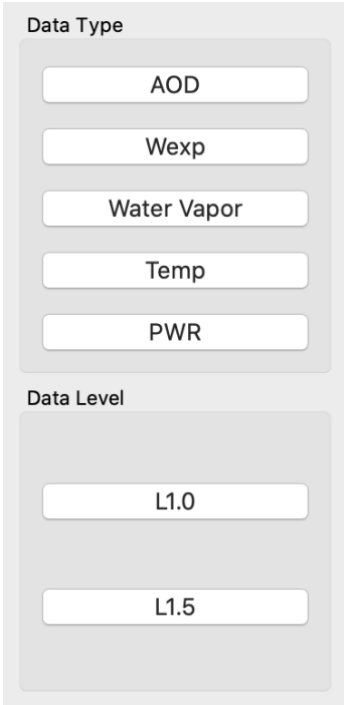


Figura 7.10: Ventana graficación - Operaciones.

En la Figura 7.10 podemos observar dos conjuntos de botones, los primeros corresponden con el tipo de medida que representan, y el segundo el filtro que se aplica a las medidas (L1.0 o L1.5). Estos últimos se pueden aplicar a todas las medidas menos a las de PWR. En el caso de intentar aplicarlo se muestra el siguiente mensaje:

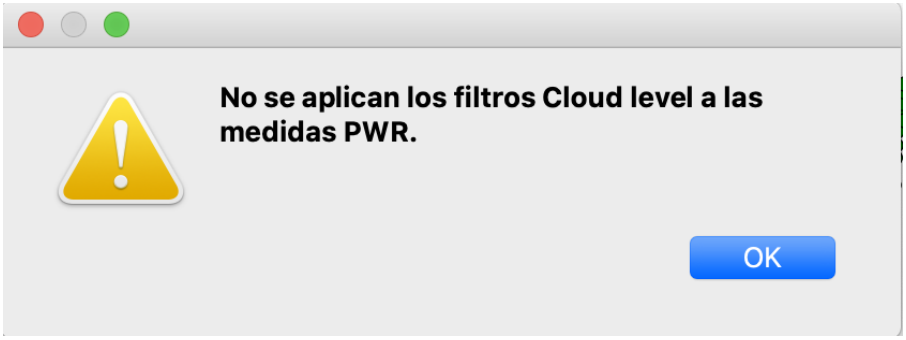


Figura 7.11: Mensaje indicando que no se puede aplicar filtro a PWR.

Capítulo 8

Conclusiones y líneas futuras

Resumen: *En este capítulo se plasman las conclusiones obtenidas de la realización de la aplicación, así como las posibles líneas futuras para mejorarla.*

8.1 Conclusiones

En lo referente a los objetivos que se marcaron al inicio de este TFG se puede concluir que se ha cumplido con ellos y el cliente ha quedado satisfecho con la aplicación resultante. Esto se debe a que se ha logrado cumplir con los requisitos que se establecieron al iniciar el proyecto. El sistema se encuentra en explotación y el cliente ya ha podido realizar pruebas reales y trabajar con él.

Este proyecto en colaboración con el GOA-UVA me ha permitido conocer el trabajo, trasfondo y las labores que realizan, a pesar de tener carencias en la temática física de la que se encarga el grupo, me ha generado un creciente interés por la meteorología y las expediciones y viajes que realizan por el mundo, como por ejemplo a la Antártida.

Respecto a la planificación y gestión de los riesgos he podido observar, debido a la situación generada por el SARS-COV-2, lo importante que es tener conocimientos y su posterior aplicación a la hora de desarrollar cualquier proyecto software. Considero que la mejor forma de realizar una planificación lo más eficiente posible es algo que se obtiene de la experiencia, y gracias a este TFG he podido ver como en determinados momentos hay que reacondicionar los planes según los problemas que vayan apareciendo.

Este TFG me ha permitido incrementar mis conocimientos en el lenguaje de programación Python, el cual es uno de los lenguajes más populares en la actualidad y ha dado paso a prácticas muy innovadoras a la hora de programar. También ha mejorado mi capacidad para estructurar código gracias al estudio y aplicación de patrones como el MVC o el fachada. También me gustaría destacar la cantidad de herramientas gratuitas a nuestro alcance, las cuales facilitan y mejoran el trabajo que se debe realizar.

Quiero destacar por último el aprendizaje y experiencia que me han aportado manejar una base de datos con un tamaño mucho mayor al que he estado acostumbrado durante la carrera. Esto me ha permitido mejorar mis conocimientos de gestión de bases de datos, así como el manejo de MySQL y la optimización de las consultas para consumir el menor tiempo y recursos posibles.

8.2 Líneas futuras

Respecto al trabajo futuro se establecen los siguientes puntos a llevar a cabo para completar la funcionalidad y aspecto de la aplicación:

- Implementar las funcionalidades relacionadas con el control de errores sobre las gráficas de datos en la ventana de “Graficación de medidas”.

- Añadir más funcionalidades a la hora de comparar o generar los datos obtenidos de cada fotómetro.

- Añadir más filtros para aplicar sobre los datos representados para poder ser más precisos en la búsqueda que se quiere realizar.

- Mejorar el rendimiento del sitema ante la representación masiva de datos, es decir, la representación de las medidas de un fotómetro en un periodo largo de tiempo.

- Añadir distintas medidas que se quieran representar a mayores de las que se permite actualmente (temperatura, vapor de agua, aod, pwr, wExp).

- En un futuro también puede esta herramienta puede ser utilizada para la calibración de los equipos.

Bibliografía

[1] AERONET, Aerosol Robotic Netrwork. [En línea]. Available: https://aeronet.gsfc.nasa.gov . [Último acceso: 18 de Marzo de 2020].

[2] Savernet, “Fotómetro solar Aeronet”. [En línea]. Available: http://www.savernet-satreps.org/es/ciencia/instrumentos/item/3-fotometro-solar-aeronet . [Último acceso: 18 de Marzo de 2020].

[3] Goa UVa, “Grupo de óptica atmosférica, Universidad de Valladolid”. [En línea]. Available: http://goa.uva.es/ . [Último acceso: 18 de Marzo de 2020].

[4] Aemet, “Nuevo portal de la Red Ibérica de Medida fotométrica de Aerosoles”. [En línea]. Available: http://izana.aemet.es/index.php?option=com_content&view=article&id=149:new-web-portal-of-the-iberian-network-for-aerosol-measurements&catid=10:news&Itemid=49&lang=es . [Último acceso: 18 de Marzo de 2020].

[5] Estilos de aprendizaje, “Plan de Desarrollo de Software”. [En línea]. Available: https://ttestilosdeaprendizaje.wordpress.com/fase-de-inicio/plan-de-desarrollo-de-software/ . [Último acceso: 23 de Marzo de 2020].

[6] P. N. Robillard, P. Kruchten y P. d'Astous, «YOOPEEDOO (UPEDU): A Process for Teaching Software Process,» 19-21 Febrero 2001.

[7] Wikipedia, “UPEDU”. [En línea]. Available: https://en.wikipedia.org/wiki/UPEDU . [Último acceso: 1 de Marzo 2020].

[8] Wikipedia, “RUP”. [En línea]. Available: https://en.wikipedia.org/wiki/Rational_Unified_Process . [Último acceso: 1 de Marzo 2020].

[9] Merlin Project, “Gestión de proyectos profesional con un poco de magia”. [En línea]. Available: https://www.projectwizards.net/es/merlin-project . [Último acceso: 2 de Marzo 2020].

[10] EsencialBlog, “Alternativas a MS Project para Mac”. [En línea]. Available: https://www.esencialblog.es/es/alternativas-a-ms-project-para-mac/. [Último acceso: 1 de Marzo 2020].

[11] Dytoc, “Modelo RUP, IBM”. [En línea]. Available: https://dtyoc.com/2016/06/07/modelo-rup-ibm/ . [Último acceso: 1 de Marzo 2020].

[12] Wikipedia, “PyQt”. [En línea]. Available: https://es.wikipedia.org/wiki/PyQt . [Último acceso: 20 de Marzo de 2020].

[13] Wikipedia, “Qt (Biblioteca). [En línea]. Available: https://es.wikipedia.org/wiki/Qt_(biblioteca)#Bindings . [Último acceso: 20 de Marzo de 2020].

[14] Qt Designer Manual. [En línea]. Available: https://doc.qt.io/qt-5/qtdesigner-manual.html . [Último acceso: 20 de Marzo de 2020].

[15] Eclipse Foundation. [En línea]. Available: https://www.eclipse.org/. [Último acceso: 1 de Marzo 2020].

[16] PyDev. [En línea]. Available: https://www.pydev.org/. [Último acceso: 1 de Marzo 2020].

[17] MySQL, “MySQL Workbench”. [En línea]. Available: https://www.mysql.com/products/workbench/ . [Último acceso: 1 de Marzo 2020].

[18] GitKraken. [En línea]. Available: https://www.gitkraken.com/ . [Último acceso: 28 de Marzo 2020].

[19] Visual Paradigm. [En línea]. Available: https://www.visual-paradigm.com/ . [Último acceso: 28 de Marzo 2020].

[20] Desarrollo web, "¿Què es MVC?. [En línea]. Available: https://desarrolloweb.com/articulos/que-es-mvc.html . [Último acceso: 15 de Marzo de 2020].

[21] Modelo Vista Controlador, "MVC Pasivo" . [En línea]. Available: https://prestashop5estrellas.wordpress.com/2010/03/29/el-patron-mvc-modelo-vista-controlador/ . [Último acceso: 15 de Marzo de 2020].

[22] Medium.com, "Los 10 patrones comunes de arquitectura de software". Wilber Ccori huaman, 7-Sept-2018. [En línea]. Available: https://medium.com/@maniakhitoccori/los-10-patrones-comunes-de-arquitectura-de-software-d8b9047edf0b . [Último acceso: 15 de Marzo de 2020].

[23] Programación.net, "Patrones de Diseño (XI): Patrones Estructurales - Facade". [En línea]. Available: https://programacion.net/articulo/patrones_de_diseno_xi_patrones_estructurales_facade_1014. [Último acceso: 15 de Marzo de 2020].

[24] Reactive Programming.io, "Facade". [En línea]. Available: https://reactiveprogramming.io/blog/es/patrones-de-diseno/facade. [Último acceso: 15 de Marzo de 2020].

[25] StackOverflow, "MVC design with Qt Designer and PyQt / PySide". [En línea]. Available: https://stackoverflow.com/questions/26698628/mvc-design-with-qt-designer-and-pyqt-pyside . [Último acceso:].

[26] MatplotLib. [En línea]. Available: https://matplotlib.org . [Último acceso: 9 de Marzo de 2020].

[27] MatplotLib Documentation, "Event handling and picking". [En línea]. Available: https://matplotlib.org/3.1.1/users/event_handling.html . [Último acceso: 17 de Marzo de 2020].

[28] Youtube, "How to make error bars in Python". [En línea]. Available: https://www.youtube.com/watch?v=xhizVO9SPjU . [Último acceso: 19 de Marzo de 2020].

[29] StackOverflow, "pyqt4 scrollArea Event and matplotlib wheelEvent". [En línea]. Available: https://stackoverflow.com/questions/43334255/pyqt4-scrollarea-event-and-matplotlib-wheelevent . [Último acceso: 24 de Marzo de 2020].

[30] Gráficos en IPython, jueves, 21 de agosto de 2014. [En línea]. Available: https://python-para-impacientes.blogspot.com/2014/08/graficos-en-ipython.html . [Último acceso: 12 de Marzo de 2020].

[31] ProgramCreek, "Python matplotlib.dates.DateFormatter() Examples". [En línea]. Available: https://www.programcreek.com/python/example/61483/matplotlib.dates.DateFormatter . [Último acceso: 14 de Marzo de 2020].

[32] StackOverflow, "How to modify the navigation toolbar easily in a matplotlib figure window". [En línea]. Available: https://stackoverflow.com/questions/12695678/how-to-modify-the-navigation-toolbar-easily-in-a-matplotlib-figure-window . [Último acceso: 19 de Marzo de 2020].

[33] Matplotlib Documentation, "Matplotlib.Figure". [En línea]. Available: https://matplotlib.org/3.2.1/api/_as_gen/matplotlib.figure.Figure.html . [Último acceso: 19 de Marzo de 2020].

[34] Numpy. [En línea]. Available: https://numpy.org . [Último acceso: 24 de Marzo de 2020].

[35] Software QA – ¿Cuáles son los tipos de pruebas software?; 11 de Febrero de 2015. [En línea]. Available: https://www.panel.es/blog/software-qa-cuales-son-los-tipos-de-pruebas-software/. [Último acceso: 28 de Abril de 2020].

[36] Pruebas de caja negra; 20 de Febrero de 2017. [En línea]. Available: http://www.pmoinformatica.com/2017/02/pruebas-de-caja-negra-ejemplos.html. [Último acceso: 28 de Abril de 2020].

[37] EcuRed, “Pruebas de caja blanca”. [En línea]. Available: https://www.ecured.cu/Pruebas_de_caja_blanca. [Último acceso: 28 de Abril de 2020].

[38] Carlos Toledano. Climatología de los aerosoles mediante la caracterización de propiedades ópticas y masas de aire en la estación 'El Arenosillo' de la red AERONET PhD Thesis. Universidad de Valladolid, 2005.

[39] Giles, D. M., Sinyuk, A., Sorokin, M. S., Schafer, J. S., Smirnov, A., Slutsker, I., Eck, T. F., Holben, B. N., Lewis, J., Campbell, J.,Welton, E. J., Korkin, S., and Lyapustin, A.: Advancements in the Aerosol Robotic Network (AERONET) Version 3 Database - Automated Near Real-Time Quality Control Algorithm with Improved Cloud Screening for Sun Photometer Aerosol Optical Depth(AOD) Measurements, Atmospheric Measurement Techniques Discussions, 2018, 1–78, https://doi.org/10.5194/amt-2018-272, https://www.atmos-meas-tech-discuss.net/amt-2018-272/.

Anexos

Anexo I - Transformar archivo .ui a .py

Transformar el archivo con la interfaz generada en Qt Designer (con formato .ui) a código (con formato .py). De esta forma podemos crear las interfaces de forma sencilla y después importarlas directamente al código una vez transformadas. También cabe la posibilidad de importar el .ui directamente, pero en el caso de este proyecto es necesario añadir implementación que no se puede incluir con la herramienta Qt Designer. La herramienta de transformación nos la proporciona directamente python, y es la siguiente:

MacBook-Pro-de-Jesus:~jesus$ python3 /Library/Frameworks/Python.framework/Versions/3.7/bin/pyuic5 -o qt_designer.py qt_designer.ui

- /Library/Frameworks/Python.framework/Versions/3.7/bin/pyuic5: Localización del comando para realizar la conversión.
- -o qt_designer.py: Opción para elegir el output de la conversión.
- qt_designer.ui: Archivo obtenido de realizar la interfaz con Qt Designer.

Anexo II - Ajustes NavigationToolbar

Para la ventana de “Uso de fotómetros” se ha requerido modificar la clase NavigationToolbar2, la cual se puede encontrar en matplotlib.backend_bases, debido a que aportaba funcionalidades que no cubrían con los requisitos de la aplicación o eran innecesarios. La representación de la clase sin ninguna modificación es la siguiente:

Figura Anexo II.1: NavigationToolbar2 sin modificar.

Pero para la navegabilidad por esta ventana ya mencionada era necesario implementar un slider, en vez de la opción cuarta de desplazar, la cual hacía la navegabilidad lenta y tediosa. El slider es la barra de la derecha para desplazarse por la gráfica. De este modo y tras sobreescribir esta clase el resultado ha sido el siguiente:

Figura Anexo II.2: NavigationToolbar2 modificado junto con el Slider.

Tras esto surge el problema, el slider y el nuevo NavigationToolbar2 no están sincronizados, cuando uno realiza una acción el otro no la detecta, y a la inversa. Para ello se han implementado los siguientes métodos los cuales han permitido limpiar y actualizar las coordenadas de la gráfica y poder sincronizar ambos objetos:

```
#Limpia el cursor sobre la grafica para sincronizar con el slider
def clearCursor(self):
    self._nav_stack.clear()

#Actualiza las coordenadas actuales de la gráfica
def push_current(self):
    self._nav_stack.push(
        WeakKeyDictionary(
            {ax: (ax._get_view(),
                  # Store both the original and modified positions.
                  (ax.get_position(True).frozen(),
                   ax.get_position().frozen()))
             for ax in self.canvas.figure.axes}))
    self.set_history_buttons()
```

Figura Anexo II.3: Funciones para la sincronización entre NavigationToolbar2 y el Slider.

Anexo III – Base de datos CÆLIS

En este anexo se muestra el diagrama entidad-relación completo de la base de datos CÆLIS, base de datos de la que esta herramienta obtiene los datos para su graficación.

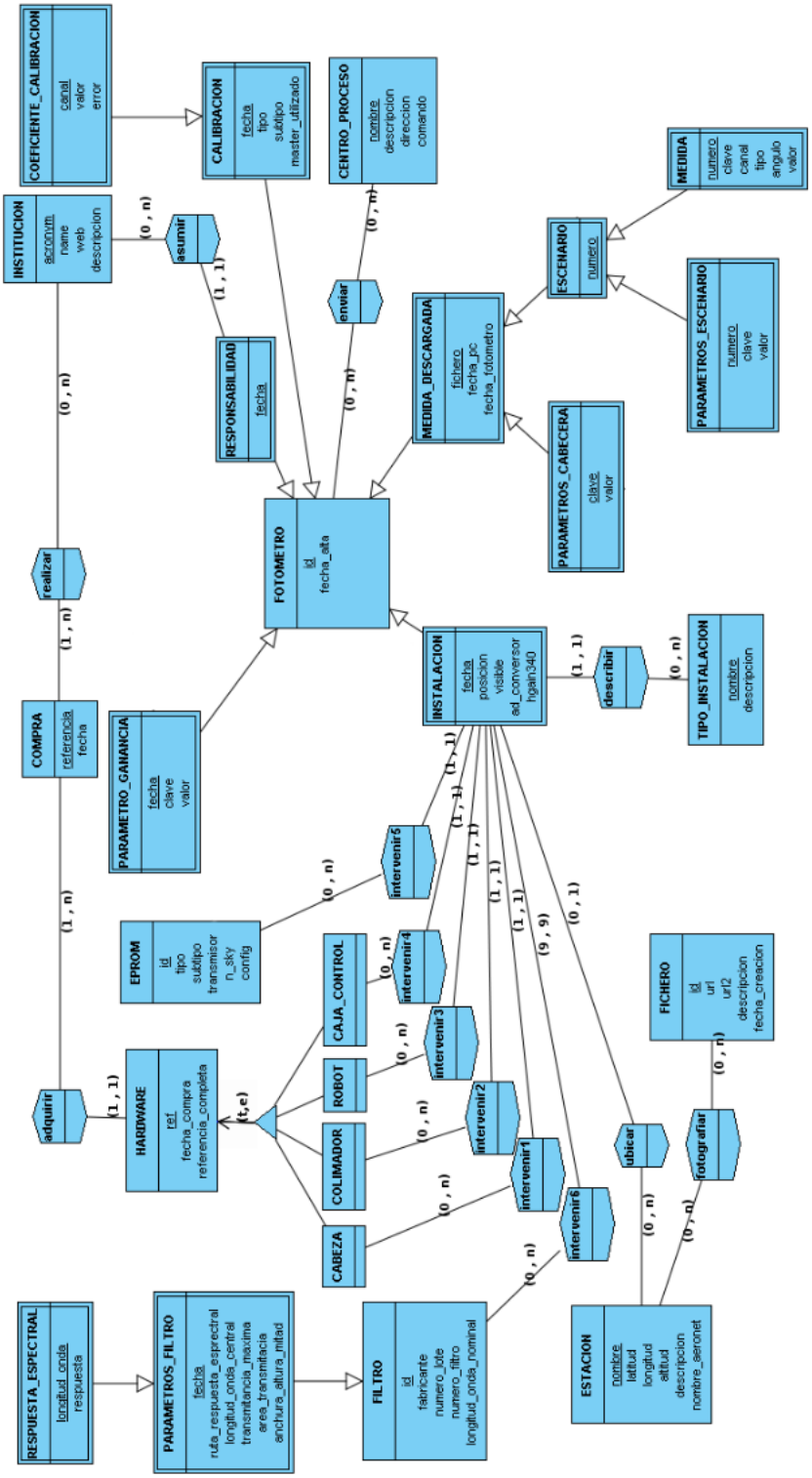


Figura Anexo III.1: Diagrama entidad-relación de CÆLIS.