



UNIVERSIDAD DE GRANADA

TRABAJO FIN DE GRADO
INGENIERÍA INFORMÁTICA

Herramienta de Análisis y Recuperación de Datos en Sistemas de Monitoreo Móvil de la Calidad del Aire

Autor

Marta Xiaoyang Moraga Hernández

Directores

María del Campo Bermúdez Edo
Daniel Bolaños Martínez

Escuela Técnica Superior de Ingenierías Informática y de
Telecomunicación

—
Granada, noviembre de 2025



**UNIVERSIDAD
DE GRANADA**

Herramienta de Análisis y Recuperación de Datos en Sistemas de Monitoreo Móvil de la Calidad del Aire

Autor

Marta Xiaoyang Moraga Hernández

Directores

María del Campo Bermúdez Edo
Daniel Bolaños Martínez

Herramienta de Análisis y Recuperación de Datos en Sistemas de Monitoreo Móvil de la Calidad del Aire

Marta Xiaoyang Moraga Hernández

Resumen

Este proyecto describe el desarrollo, siguiendo la metodología ágil Scrum, de una herramienta de código abierto, programada en Python. La herramienta permite la visualización, a través de una interfaz gráfica de usuario creada utilizando la biblioteca PyQt5, de series temporales obtenidas de un sensor móvil de partículas en suspensión. Además recopila datos de ubicación al conectarse de forma inalámbrica a un teléfono móvil. Las series temporales se almacenan internamente como DataFrames de pandas y la visualización de los datos se puede realizar sobre un mapa, utilizando la biblioteca Folium, o sobre un gráfico de líneas, mediante matplotlib. Asimismo, el software puede realizar la imputación de valores faltantes utilizando uno de seis algoritmos, implementados mediante las funciones que ofrece pandas. La herramienta posee funciones de manipulación suficientes para crear una serie temporal con valores perdidos a partir de una original. La implementación sigue el patrón arquitectónico modelo-vista-controlador, la vista estando compuesta por *widgets* pertenecientes a PyQt5, y la comunicación entre las diferentes partes sigue el patrón *publish-subscribe*, siguiendo la implementación de *pyqtSignals* de PyQt5. El resultado es una herramienta software fácil de usar, portable y extensible, especialmente en lo que respecta a métodos de imputación puesto que se utiliza un patrón estrategia, utilizable sobre los sistemas operativos Windows y Linux.

Palabras claves: visualización de datos, imputación, valores faltantes, series temporales, partículas en suspensión, código abierto, desarrollo ágil

Data Analysis and Recovery Tool for Mobile Air Quality Monitoring Systems

Marta Xiaoyang Moraga Hernández

Abstract

This Bachelor's Degree Final Project main goal is describing the development, following the agile Scrum methodology, of an open-source tool, programmed in Python, that allows the visualization, through a graphical user interface created using the PyQt5 library, of time series obtained from a mobile particulate matter sensor that also collects location data by wirelessly connecting to a mobile phone; moreover, this work aims to strengthen and expand the student's previous skills in software development and implementation; and, at the same time, to acquire knowledge about time-series data manipulation. The time series are stored internally as pandas DataFrames and the data is visualized on a map, using the Folium library, or a line graph, using matplotlib. The software can also perform missing value imputation using one of six algorithms, based on and implemented using the functions offered by pandas, and has sufficient manipulation functions to create a time series with missing values from an original. The implementation follows the model-view-controller architectural pattern, the view being composed of PyQt5 widgets, and the communication between the different parts follows the publish-subscribe pattern, following the PyQt5 implementation of pyqtSignals. The result is an easy-to-use, portable, and extensible software tool, especially with regard to imputation methods, since it uses a strategy pattern, usable on both Windows and Linux platforms; furthermore, a deeper understanding of agile project management, software development, Python and time series analysis is gained by the writer.

Keywords: data visualization, imputation, missing values, time series, particulate matter, open source, agile development

Yo, **Marta Xiaoyang Moraga Hernández**, alumno de la titulación INGENIERÍA INFORMÁTICA de la Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación de la Universidad de Granada, autorizo la ubicación de la siguiente copia de mi Trabajo Fin de Grado en la biblioteca del centro para que pueda ser consultada por las personas que lo deseen.

Granada a 11 de Noviembre de 2025.

Dña. María del Campo Bermúdez Edo, Profesora del Departamento Lenguajes y Sistemas de la Universidad de Granada.

D. Daniel Bolaños Martínez, Profesor del Departamento Lenguajes y Sistemas de la Universidad de Granada.

Informan:

Que el presente trabajo, titulado **Herramienta de Análisis y Recuperación de Datos en Sistemas de Monitoreo Móvil de la Calidad del Aire**, ha sido realizado bajo su supervisión por **Marta Xiaoyang Moraga Hernández**, y autorizamos la defensa de dicho trabajo ante el tribunal que corresponda.

Y para que conste, expiden el presente informe en Granada a 11 de noviembre de 2025.

Agradecimientos

Este proyecto no hubiera sido posible sin mis compañeros, junto a los que he aprendido y crecido, y mis profesores durante la carrera que han trabajado para formarnos como futuros profesionales. En especial, quiero dar las gracias a mis directores de proyecto, María y Daniel, cuya ayuda ha sido indispensable para la realización del mismo. Además, quiero agradecer a mis amigos, que han aguantado todas mis quejas y estrés respecto al trabajo y siempre han estado ahí para animarme. Y finalmente, a mi familia, que siempre me han apoyado y han hecho todo en su mano para que yo pudiera estudiar.

Índice general

Capítulo 1. Introducción.....	1
1.1 Contexto.....	1
1.2 Motivación.....	1
1.3 Objetivos.....	2
1.4 Organización de la memoria.....	2
Capítulo 2. Estado del Arte y de la Técnica.....	3
2.1 Algorítmica.....	3
2.2 Tecnologías para la implementación.....	5
2.2.1 R.....	5
2.2.2 Python.....	7
2.2.3 Otras soluciones.....	10
Capítulo 3. Descripción de la propuesta.....	12
3.1 Metodología de gestión del proyecto.....	12
3.2 Requisitos.....	13
3.2.1 Requisitos funcionales.....	16
3.2.2 Requisitos no funcionales.....	17
3.3 Presupuesto.....	17
3.4 Planificación temporal.....	18
3.4.1 Descripción de los sprints.....	19
3.4.2 Diagrama de Gantt.....	42
3.5 Diseño.....	47
3.5.1 Diagrama de clases.....	47
3.5.2 Diagramas de secuencia.....	54
3.5.3 Pruebas de validación.....	58
Capítulo 4. Resultado final.....	68
4.1 Funcionalidades.....	68
4.1.1 Importación y exportación de archivos.....	69
4.1.2 Gestión de pestañas.....	70
4.1.3 Manipulación secuencias.....	71
4.1.4 Visualización.....	75
4.1.5 Imputación.....	77
4.2 Testar.....	80
Capítulo 5. Conclusiones.....	82
Bibliografía.....	84

Índice de figuras

Figura 2.1: Ejemplo de diagrama de dispersión en un mapa usando Plotly extraído de (Plotly, 2025).....	6
Figura 2.2: Diez primeros en el Índice TIOBE agosto 2025 extraído de (Jansen, 2022).....	8
Figura 2.3: Comparación de interpolaciones utilizando pandas obtenido de “Working with missing data” (Working With Missing Data — Pandas 2.3.3 Documentation, 2025).....	9
Figura 2.3: Ejemplo de gráfica de burbujas geográficas independientes extraído de (Crear Mapas Con Datos de Latitud y Longitud, 2025).....	11
Figura 3.1: Distribución de tareas por días en el sprint 1.....	20
Figura 3.2: Distribución de tareas por días en el sprint 2.....	21
Figura 3.3: Prototipo de mapa con puntos.....	23
Figura 3.4: Distribución de tareas por días en el sprint 3.....	24
Figura 3.5: Distribución de tareas por días en el sprint 4.....	25
Figura 3.6: Incremento del sprint 4.....	25
Figura 3.7: Distribución de tareas por días en el sprint 5.....	26
Figura 3.8: Distribución de tareas por días en el sprint 6.....	27
Figura 3.9: Incremento resultante del sprint 6.....	28
Figura 3.10: Distribución de tareas por días en el sprint 7.....	29
Figura 3.11: Boceto de interfaz de usuario, base.....	30
Figura 3.12: Boceto de interfaz de usuario, diálogo de exportación.....	30
Figura 3.13: Boceto de interfaz de usuario, diálogo de importación.....	31
Figura 3.14: Boceto de interfaz de usuario, diálogo de eliminación de valores.....	31
Figura 3.15: Distribución de tareas por días en el sprint 8.....	32
Figura 3.16: Distribución de tareas por días en el sprint 9.....	33
Figura 3.17: Distribución de tareas por días en el sprint 10.....	34
Figura 3.18: Captura del incremento resultante de tras el sprint 10.....	35
Figura 3.19: Distribución de tareas por días en el sprint 11.....	36
Figura 3.20: Distribución de tareas por días en el sprint 12.....	37
Figura 3.21: Distribución de tareas por días en el sprint 13.....	38
Figura 3.22: Distribución de tareas por días en el sprint 14.....	39
Figura 3.23: Distribución de tareas por días en el sprint 15.....	39
Figura 3.24: Distribución de tareas por días en el sprint 16.....	40
Figura 3.25: Captura del incremento resultante de tras el sprint 16.....	41
Figura 3.26: Distribución de tareas por días en el sprint 17.....	42
Figura 3.27: Diagrama de Gantt completo, desde el sprint 1 al 7.....	45
Figura 3.28: Diagrama de Gantt completo, desde el sprint 8 al 17.....	46
Figura 3.29: Diagrama de clases completo.....	49
Figura 3.30: Sección del modelo del diagrama de clases.....	50
Figura 3.31: Sección del controlador del diagrama de clases.....	51
Figura 3.32: Primera sección de la vista.....	52
Figura 3.33: Segunda sección de la vista.....	53
Figura 3.34: Diagrama de secuencia para la importación de series temporales de archivos.....	55
Figura 3.35: Diagrama de secuencia para aplicar una corrección a una serie.....	56

Figura 3.36: Diagrama de secuencia para la selección de método de imputación.....	57
Figura 4.1: Áreas de la interfaz.....	68
Figura 4.2: Opciones de importación/exportación.....	69
Figura 4.3: Diálogo de importación.....	69
Figura 4.4: Diálogo de exportación.....	70
Figura 4.5: Opciones para añadir pestañas.....	70
Figura 4.6: Eliminación de pestañas.....	70
Figura 4.7: Renombramiento de pestañas.....	71
Figura 4.8: Desplazamiento de pestañas.....	71
Figura 4.9: Opciones sobre secuencias.....	71
Figura 4.10: Eliminar porcentaje de valores.....	73
Figura 4.11: Eliminar intervalo de valores.....	73
Figura 4.12: Aplicar corrección a una serie.....	74
Figura 4.13: Muestra de cambio de nombre de serie temporal.....	74
Figura 4.14: Ejemplo de protección contra repetición de nombres.....	75
Figura 4.15: Ejemplo de visualización en gráfico de líneas.....	76
Figura 4.16: Ejemplo de visualización en mapa.....	76
Figura 4.17: Ejemplo de memoria entre pestañas 1.....	77
Figura 4.18: Ejemplo de memoria entre pestañas 2.....	77
Figura 4.19: Muestra de control de visibilidad.....	78
Figura 4.20: Ejemplo de pestaña de análisis.....	78
Figura 4.21: Menú de imputación.....	79
Figura 4.22: Selector de imputación.....	80
Figura 4.23: Muestra de conversión a serie temporal del resultado de imputación.....	80

Índice de tablas

Tabla 3.1: Historia de usuario 1, visualizar secuencias temporales.....	14
Tabla 3.2: Historia de usuario 2, imputar valores estimados a secuencias temporales con valores perdidos.....	14
Tabla 3.3: Historia de usuario 3, manipulación de secuencias temporales.....	15
Tabla 3.4: Historia de usuario 4, importar archivos csv con medidas de sensores.....	15
Tabla 3.5: Historia de usuario 5, exportar resultados a archivos csv.....	16
Tabla 3.6: Resumen del presupuesto.....	18
Tabla 3.7: Descripción del sprint 1.....	19
Tabla 3.8: Descripción del sprint 2.....	20
Tabla 3.9: Comparación de las tecnologías de desarrollo.....	21
Tabla 3.10: Descripción del sprint 3.....	23
Tabla 3.11: Descripción del sprint 2.....	24
Tabla 3.12: Descripción del sprint 5.....	25
Tabla 3.13: Descripción del sprint 6.....	26
Tabla 3.14: Historia de usuario 6, personalizar la visualización de series temporales.....	27
Tabla 3.15: Descripción del sprint 7.....	28
Tabla 3.16: Descripción del sprint 8.....	31
Tabla 3.17: Descripción del sprint 9.....	32
Tabla 3.18: Descripción del sprint 10.....	33
Tabla 3.19: Descripción del sprint 11.....	35
Tabla 3.20: Historia de usuario 7, usar modelos de aprendizaje automático.....	36
Tabla 3.21: Descripción del sprint 12.....	36
Tabla 3.22: Descripción del sprint 13.....	37
Tabla 3.23: Descripción del sprint 13.....	38
Tabla 3.24: Descripción del sprint 15.....	39
Tabla 3.25: Descripción del sprint 16.....	40
Tabla 3.26: Descripción del sprint 17.....	41
Tabla 3.27: División de tiempo por paquete de trabajo.....	44
Tabla 3.28: Caso de prueba 1.....	58
Tabla 3.29: Caso de prueba 2.....	58
Tabla 3.30: Caso de prueba 3.....	58
Tabla 3.31: Caso de prueba 4.....	59
Tabla 3.32: Caso de prueba 5.....	59
Tabla 3.33: Caso de prueba 6.....	59
Tabla 3.34: Caso de prueba 7.....	60
Tabla 3.35: Caso de prueba 8.....	60
Tabla 3.36: Caso de prueba 9.....	61
Tabla 3.37: Caso de prueba 10.....	61
Tabla 3.38: Caso de prueba 11.....	61
Tabla 3.39: Caso de prueba 12.....	62
Tabla 3.40: Caso de prueba 13.....	62
Tabla 3.41: Caso de prueba 14.....	62

Tabla 3.42: Caso de prueba 15.....	63
Tabla 3.43: Caso de prueba 16.....	63
Tabla 3.44: Caso de prueba 17.....	63
Tabla 3.45: Caso de prueba 18.....	64
Tabla 3.46: Caso de prueba 19.....	64
Tabla 3.47: Caso de prueba 20.....	64
Tabla 3.48: Caso de prueba 21.....	65
Tabla 3.49: Caso de prueba 22.....	65
Tabla 3.50: Caso de prueba 23.....	65
Tabla 3.51: Caso de prueba 24.....	66
Tabla 3.52: Caso de prueba 25.....	66
Tabla 3.53: Caso de prueba 26.....	66
Tabla 3.54: Caso de prueba 27.....	67
Tabla 3.55: Caso de prueba 28.....	67

Capítulo 1. Introducción

1.1 Contexto

Una serie temporal es un conjunto de observaciones o medidas tomadas siguiendo cierto orden cronológico y entre las que suele estar implícita una relación de correlación. Su análisis permite descubrir relaciones subyacentes y utilizar los datos recopilados para predecir futuros u obtener deducciones sobre el total (Wei, 2013b). No obstante, para realizar un análisis eficaz se necesita una alta cantidad de datos fiables con los que trabajar y la pérdida de datos supone un gran obstáculo en el procesamiento. El problema de los datos faltantes es especialmente problemático para los sistemas de monitorización a gran escala de los sistemas del Internet de las cosas (IoT) debido a que requieren recabar gran cantidad de datos de diferentes localizaciones, describen Du et al. (Du et al., 2020b), y, por tanto, es un tema de gran interés para muchos autores.

Un ámbito dónde se enfrenta frecuentemente este dilema es en la investigación medioambiental. Junninen et al. (Junninen et al., 2004b) mencionan observaciones insuficientes, errores en las medidas o problemas en la adquisición cómo las posibles causas y comparan algunos de los diferentes métodos existentes para imputar y sustituir los valores faltantes por otros. En esta comparativa concluyen que un esquema adecuado de imputación puede ayudar a remediar el problema pero que los métodos de imputación no son infalibles y que la naturaleza de los datos y situación dictan cuál tendrá un mejor desempeño. Más recientemente, Hadeed et al. (Hadeed et al., 2020c) expresa una conclusión parecida y, por ejemplo, expresa la utilidad de algoritmos más simples en estudios de períodos cortos.

Imputar y analizar los datos suele requerir el uso de algún software estadístico, como los programas STATA (StataCorp LLC, 2025) o SAS (SAS Institute Inc., 2025) y/o conocimiento de algún lenguaje de programación como R y Python. Por otro lado, la dificultad para obtener datos depende de la variable a medir, no obstante, obtener sensores de bajo costo es alcanzable para la mayoría de personas y su uso suele requerir conocimientos básicos.

1.2 Motivación

En lo que a la herramienta software se refiere, la motivación principal es ofrecer una aplicación libre, fácil de usar y extensible. De tal forma que equipos pequeños de investigación y usuarios individuales puedan analizar series temporales gráficamente en una interfaz lista para usar, así como adaptarla a sus necesidades dependiendo de su conocimiento en las tecnologías involucradas.

El desarrollo de este trabajo busca reforzar y ampliar las competencias previas de la estudiante sobre el desarrollo e implementación de software; y, al mismo tiempo, se busca adquirir conocimientos sobre la manipulación de datos en series temporales. Esto último viene motivado por la situación actual en la industria del software donde el análisis de datos ha crecido en importancia y es necesario enfrentarse al problema de la pérdida de datos. Además, la planificación y toma de decisiones pretenden emular la de un encargo profesional para poner en

práctica competencias de ingeniería de requisitos, ingeniería del software, gestión de proyectos y modelado; aplicando la metodología ágil SCRUM.

1.3 Objetivos

De acuerdo al contexto y motivación detallados se establece un **objetivo principal**: Crear una herramienta software libre que permita manipular, imputar y visualizar series temporales procedentes de sensores móviles que capturan la cantidad de partículas en suspensión en el aire que sea fácil de usar y extender.

Tras esto, se definen tres **objetivos específicos**:

1. Aprender y utilizar la metodología ágil SCRUM para desarrollar un proyecto de desarrollo software.
2. Discernir y aplicar las tecnologías más adecuadas para el desarrollo de una aplicación de manipulación de datos en base a la capacidad de imputación, creación de interfaces y facilidad de aprendizaje.
3. Estudiar y comprender suficientemente las bases de diversos métodos de imputación de datos perdidos en series temporales para que puedan ser utilizados en la herramienta software resultante.

1.4 Organización de la memoria

La memoria se estructura de la siguiente manera:

- El **capítulo 1** detalla el ámbito del trabajo, explicando el contexto, motivación y objetivos de la creación de la herramienta.
- El **capítulo 2** describe las posibles soluciones a los problemas de imputación e implementación de la aplicación. Se divide en una primera parte sobre métodos de imputación, algunos simples y otros complejos; y una segunda parte exponiendo todas las alternativas software que fueron consideradas para la implementación.
- El **capítulo 3** detalla el proceso del desarrollo de la propuesta. Incluye: los requisitos funcionales y no funcionales, el presupuesto, la planificación temporal y el diseño.
- El **capítulo 4** muestra la herramienta software final, explicando la instalación y funcionalidades.
- El **último capítulo** expone las conclusiones del proyecto, detallando los objetivos cumplidos y las limitaciones finales.
- Finalmente, se incluyen las referencias bibliográficas utilizadas.

La aplicación resultante, el código fuente y el manual se pueden encontrar en el repositorio: <https://github.com/MartaxM/TFG-PyQt-Imputation-App.git>

Capítulo 2. Estado del Arte y de la Técnica

En este capítulo se llevará a cabo una revisión de la literatura existente sobre la imputación de datos perdidos, el análisis y representación de las series temporales y las herramientas para la creación de una interfaz. Se divide en dos secciones, una dedicada a examinar los métodos de imputación existentes y otra centrada en las herramientas de desarrollo.

2.1 Algorítmica

Esta primera sección examina qué algoritmos de imputación son usados comúnmente tanto para la imputación de series temporales de forma general como para aquellas conformadas por valores de partículas en suspensión en el aire. Las variables de interés para este TFG son las que ofrece el sensor de contaminación de bajo coste que se utiliza, y que mide partículas (particulate matter) PM_{10} y $PM_{2.5}$, (partículas con un diámetro menor que 10 que 2.5 micras respectivamente) y de forma secundaria los valores de latitud y longitud, que se obtienen del móvil que acompaña al sensor.

Una de las formas más simples de enfrentar la pérdida de datos, después de simplemente ignorar los valores perdidos, es sustituir los valores por una constante. Esta constante puede ser obtenida de diversas formas. Por ejemplo, se podría utilizar el valor de la media, la mediana o la moda. El problema de esta sustitución es asumir que la serie temporal tiene una dispersión baja, es decir, que las medidas tienden a un valor central. Otro candidato para realizar la sustitución es la anterior medida válida más cercana en el tiempo, esto se llama *forward-fill*. La acción inversa sería *backward-fill* y se haría utilizando la siguiente medida válida. Ambos métodos están implementados y listos para usar en la biblioteca de *pandas* (*pandas development team, 2025a*) y suelen suavizar la serie y, como cualquier dato constante, perder utilidad cuando hay ráfagas de datos perdidos (gran cantidad de medidas perdidas seguidas).

Una alternativa algo más compleja es aplicar una interpolación, que se basa en estimar los valores perdidos en base a las medidas conocidas cercanas (Davis, 1975). La distancia entre medidas se puede calcular de diversas maneras. Por ejemplo, si sólo importa la cercanía en el tiempo, se puede aplicar el enfoque de ventana móvil. Esto quiere decir utilizar un subconjunto de medidas dentro de una ventana de tiempo cercano a la medida perdida para realizar el cálculo de la constante. Lo más común es calcular la media móvil, pero se pueden calcular otras métricas. La ventana móvil tiende a suavizar la serie temporal, y su utilidad depende del tamaño de la ventana y la cantidad de valores faltantes. En los casos en los que hay un número elevado de pérdidas se podrían utilizar valores anteriormente imputados, por otro método o el mismo, para realizar el cálculo de los siguientes, sin embargo, apoyarse en estimaciones anteriores para generar nuevas distorsiona el resultado final.

Una interpolación más compleja sería la interpolación de los k-vecinos más cercanos o *k-nearest neighbours (k-NN) interpolation*. Cuando hay un valor faltante se calcula la media ponderada de las k medidas más similares, afectando al cálculo de la similitud todas las variables disponibles (James et al., 2023). La similitud o distancia se calcula utilizando la

distancia Euclídea. Dependiendo de la cantidad de vecinos que se usan se denomina 1-NN, 2-NN..., k-NN. Por ejemplo, una forma de 1-NN es avanzar la última observación, o *last observation carried forward* (LOCF). Es decir, el valor faltante se sustituye por la última medida válida más similar. LOCF puede usarse en conjunto con *next value carried backward*, que usa la siguiente medida válida, eligiendo la medida más similar. Aunque LOCF es simple y de uso común, la utilidad de este método va a depender de la similitud con la última medida; por tanto, lo ideal es tener gran cantidad de medidas. Lachin (2015) cuestiona la validez de utilizar LOCF para análisis de datos. Esta conclusión nace del resultado de su estudio en el cual los resultados del método sólo se podían considerar imparciales cuando la distribución de los valores observados eran exactamente iguales que la de los valores faltantes. Aún así, utilizar k-NN, aunque un algoritmo de aprendizaje supervisado simple, es considerado una solución efectiva para imputar series temporales multivariantes. Ahn et al. (2021) experimentan con varios métodos convencionales de imputación y k-NN muestra los mejores resultados en tres de cuatro conjuntos de datos: Air Quality, GECCO2015-A y CNNpred. Del mismo modo, el estudio concluye que los resultados de los métodos de imputación multivariantes obtienen resultados más favorables que los univariados.

Otra forma efectiva de realizar una imputación es modelar la relación entre la variable que se quiere imputar y otra u otras que la influyen, a esto se le llama regresión (James et al., 2023). La regresión lineal es un método de aprendizaje supervisado. Núñez et al. (2011) explican que la regresión necesita encontrar el método adecuado para calcular las constantes que encaje con la estructura de los datos, estudiando cómo se emplea en el contexto de las ciencias de la salud. En las secuencias temporales, las variables se deben modelar a lo largo del tiempo conocido como *joint modeling regression*. Una de las formas más directas es la regresión lineal simple, la cual establece que la relación entre la variable dependiente es aproximadamente modelada por una ecuación lineal, es decir, la suma de una constante más la variable independiente multiplicada por otra constante (James et al., 2023). Este enfoque es extendido por la regresión lineal múltiple, en dónde se busca la relación de la variable dependiente con más de una variable independiente representada cada una por un nuevo término en la ecuación lineal. La dificultad de estos métodos se encuentra en calcular las constantes de forma precisa. Richman et al. (2008) comparan diversos métodos de imputación y la regresión lineal simple presenta, en general, una mejora muy reducida respecto al cálculo de la media y la regresión lineal múltiple, aunque mejora a los anteriores, se ve superada por métodos como las redes neuronales artificiales.

Por una vía similar, el aprendizaje profundo ha mostrado gran capacidad para modelar datos perdidos. El aprendizaje profundo, describen LeCun et al. (LeCun et al., 2015), es una forma de aprendizaje de características, que aprende la representación más útil de los datos de forma automática, con varios niveles de representación obtenidos de módulos simples pero no lineales que transforman cada representación a una más abstracta. Uno de los ejemplos más conocidos son las redes neuronales artificiales, las cuales imitan el funcionamiento y conexiones de un cerebro humano utilizando un conjunto de funciones parametrizadas no lineales conectadas entre sí, llamadas neuronas o nodos, que pueden ser de entrada, de salida e intermedios, los cuales realizan la mayoría del procesamiento (Dreyfus, 2005). Estos nodos están conectados de forma que los datos entran a la red y por cada nodo por el que pasan se les aplica un cálculo en base a la estrategia seleccionada y luego el resultado pasa al siguiente hasta llegar a un nodo de salida, este proceso es el entrenamiento del modelo (Dreyfus, 2005). Existen gran cantidad de redes neuronales artificiales según la estrategia que siguen, como son sus

conexiones, como se realiza el entrenamiento, etc. Por ejemplo, Transformers (Islam et al., 2023) permite identificar la importancia de cada elemento de una secuencia con respecto a los demás utilizando un mecanismo de autoatención para identificar relaciones entre distintos instantes en el tiempo y poder realizar la imputación. Otra estrategia de aprendizaje automático que también utiliza autoatención es SAITS (*Self-Attention-based Imputation for Time Series*), propuesto por W. Du et al. (2022), el cual está diseñado específicamente para imputar valores faltantes en series temporales mediante la captura de dependencias a largo plazo entre diferentes puntos de la serie temporal.

2.2 Tecnologías para la implementación

Esta sección describe qué herramientas software existen para el desarrollo de una aplicación que cumpla los objetivos del proyecto teniendo en cuenta el dominio del problema. El objetivo es entender qué lenguaje de programación y qué herramientas son las más adecuadas para la implementación atendiendo a diversos criterios. El primer criterio de análisis, y uno de los de mayor importancia, es la viabilidad de la implementación de los diversos algoritmos de imputación y el tratamiento de secuencias temporales. Seguidamente, se observará su capacidad para el trazado y visualización de datos, con especial énfasis en los gráficos de líneas y marcadores en mapas de coordenadas. Además se debe tener en cuenta las herramientas disponibles para la creación de una interfaz de usuario, la facilidad de uso, comunidad existente y experiencia propia. Cabe destacar que el foco estará en R, debido a su especialización estadística, y Python, por el gran soporte y versatilidad que presenta; sin embargo, otros dos posibilidades, Julia y la opción de usar un lenguaje distinto para la interfaz de usuario y los cálculos propios, también se tendrán en cuenta.

2.2.1 R

R es un lenguaje y entorno creado por Ross Ihaka y Robert Gentleman del Departamento de Estadística de la Universidad de Auckland (Peng, 2016). Es un proyecto derivado del lenguaje S, desarrollado por John Chambers et al. para Bell Telephone Laboratories, originalmente parte de AT&T Corp. Ambos lenguajes, R y S, son lenguajes sofisticados para la computación estadística, manipulación de datos y visualización gráfica.

Una de las características principales de R es ser un proyecto de código abierto con gran capacidad de extensibilidad; lo que, unido a las frecuentes publicaciones, ha permitido una mayor expansión en comparación a S, que requiere una licencia comercial para su uso. Además, R produce salidas mínimas, mientras que la filosofía de S divide el trabajo estadístico en múltiples pasos con resultados intermedios que se almacenan en objetos (The CRAN team, 2025).

Respecto a las capacidades de implementación, R presenta una gran comunidad y elevado número de bibliotecas estadísticas disponibles. La mayoría de métodos de estadística clásica y otros más avanzados, así como procedimientos específicos para temas como la imputación están recogidos en diferentes librerías (o paquetes) de R. Por ejemplo, *imputeTS* es un paquete especializado en imputación de series temporales univariantes que ofrece la implementación de algoritmos de imputación de última generación y trazado de datos (Moritz & Bartz-Beielstein, 2017). Otra instancia es el paquete *pan*, el cuál es comúnmente recomendado

para la imputación multinivel de modelos con parámetros en varios niveles (Grund et al., 2016).

De igual manera, R permite el empleo y diseño de técnicas de alto nivel actualmente como es el aprendizaje automático y el aprendizaje profundo, como explica Ramasubramanian y Singh en “*Machine Learning Using R*” (Ramasubramanian & Singh, 2018). Los autores detallan cómo es posible dominar el proceso de trabajo, los algoritmos y casos reales de uso, sin necesidad de aprender otros lenguajes como Python que suele ser el recomendado. En su libro, explican el uso de *Apache Hadoop* y *Spark* para el procesamiento y análisis de grandes volúmenes de datos, así como de *Keras* y *TensorFlow* para el desarrollo y entrenamiento de modelos de aprendizaje profundo.

Por otra parte, para el trazado de gráficos se hallan diversas opciones como el paquete de R, *ggplot2* (Comprehensive R Archive Network (CRAN), 2025), que es un sistema para crear gráficos basado en “*The Grammar of Graphics*” capítulo del “*Handbook of Computational Statistics*” (Wilkinson, 2011). Otra posibilidad para gráficos listos para publicar es usar *Potly*, una librería de gráficos interactivos de python, integrable con *ggplot2*, que incluye mapas (Figura 2.1), diagramas de dispersión, histogramas, etc. (Plotly, 2025). Ambos son de uso gratuito, con Potly ofreciendo otras opciones comerciales.



Figura 2.1: Ejemplo de diagrama de dispersión en un mapa usando Plotly extraído de (Plotly, 2025).

En lo que respecta a la creación de interfaces gráficas R incluye *tcltk* de base, un paquete básico combinación entre un lenguaje de scripting y un conjunto de herramientas para interfaces gráficas de usuario («A Primer On The R-Tcl/Tk Package», 2001). Una opción más completa es *Shiny*, aunque este es un paquete que permite crear solo aplicaciones web desde R que se pueden abrir desde *RStudio*, entorno de desarrollo especializado para R, o un navegador web (Wickham, 2021). Otra alternativa pero para aplicaciones de escritorio es *RGtk2*, que provee funciones que llaman a la librería *GTK+* subyacente escrita en C (CRAN: Package *RGTK2*, 2025), sin embargo, está obsoleta para las versiones más actuales. Como alternativa se

podría usar *gWidgets2*, sin embargo es importante remarcar que esta biblioteca necesita usarse en combinación con otro paquete que de acceso a librerías gráficas como *tcltk* o *RGtk2* (Verzani, 2024).

En conclusión, R es una opción que destaca en el ámbito de análisis de datos y uso estadístico y, aunque se encuentra más limitado en el desarrollo de interfaces, presenta alternativas suficientes.

2.2.2 Python

La siguiente opción es Python, un lenguaje de código abierto de alto nivel de propósito general, compatible con múltiples paradigmas de programación como desarrollo orientado a objetos o programación procedimental (Mehare et al., 2023). Guido van Rossum comenzó a desarrollar Python alrededor de los años ochenta como sucesor del lenguaje de programación ABC.

Una ventaja que presenta Python es su gran popularidad; como se puede observar en posición en el índice TIOBE (Jansen, 2022) (Figura 2.2), cuyas posiciones son calculadas según las búsquedas relacionadas en 25 motores de búsqueda y páginas relevantes, como son Google o Wikipedia (*TIOBE Index - TIOBE*, 2022). Esto no quiere decir que el lenguaje sea mejor, pero sí que presentará una mayor comunidad y, por tanto, un repertorio amplio de bibliotecas y soluciones de otros usuarios a problemas comunes. El origen de esta popularidad es su flexibilidad, su facilidad para el aprendizaje por su sintaxis directa y tipado dinámico, su expresividad, la multitud de plataformas en las se puede utilizar y la amplia librería estándar que incluye («Python – The Fastest Growing Programming Language», 2017). Sin embargo, Srinath en el mismo documento también describe las limitaciones que presenta, por ejemplo, la confusión durante el mantenimiento que produce el tipado dinámico al depender del contexto o la lentitud del lenguaje debido al referenciado constante que esto causa.

Aug 2025	Aug 2024	Change	Programming Language	Ratings	Change
1	1		 Python	26.14%	+8.10%
2	2		 C++	9.18%	-0.86%
3	3		 C	9.03%	-0.15%
4	4		 Java	8.59%	-0.58%
5	5		 C#	5.52%	-0.87%
6	6		 JavaScript	3.15%	-0.76%
7	8		 Visual Basic	2.33%	+0.15%
8	9		 Go	2.11%	+0.08%
9	25		 Perl	2.08%	+1.17%
10	12		 Delphi/Object Pascal	1.82%	+0.19%

Figura 2.2: Diez primeros en el Índice TIOBE agosto 2025 extraído de (Jansen, 2022)

Respecto a la imputación, Python presenta menos opciones listas para usar que R, puesto que no es un lenguaje especializado en la estadística, sino de uso general. Sin embargo, no carece de capacidad. Python tiene una base sólida de librerías de código abierto de cálculo numérico como *NumPy*; una librería de matrices multidimensionales con álgebra lineal básica que ofrece múltiples funciones matemáticas, transformaciones y métodos (NumPy team, 2025), y *SciPy*; con algoritmos de interpolación y ecuaciones diferenciales entre otras operaciones que provee (SciPy Team, 2025). En una línea similar, *Pandas* es una librería de código abierto para el análisis y manipulación de datos. Incorpora métodos de sustitución de valores faltantes como la interpolación y permite la implementación de métodos más complejos (*Working With Missing Data — Pandas 2.3.3 Documentation*, 2025). De hecho, incluye capacidades de trazado de datos como muestra la figura 2.3.

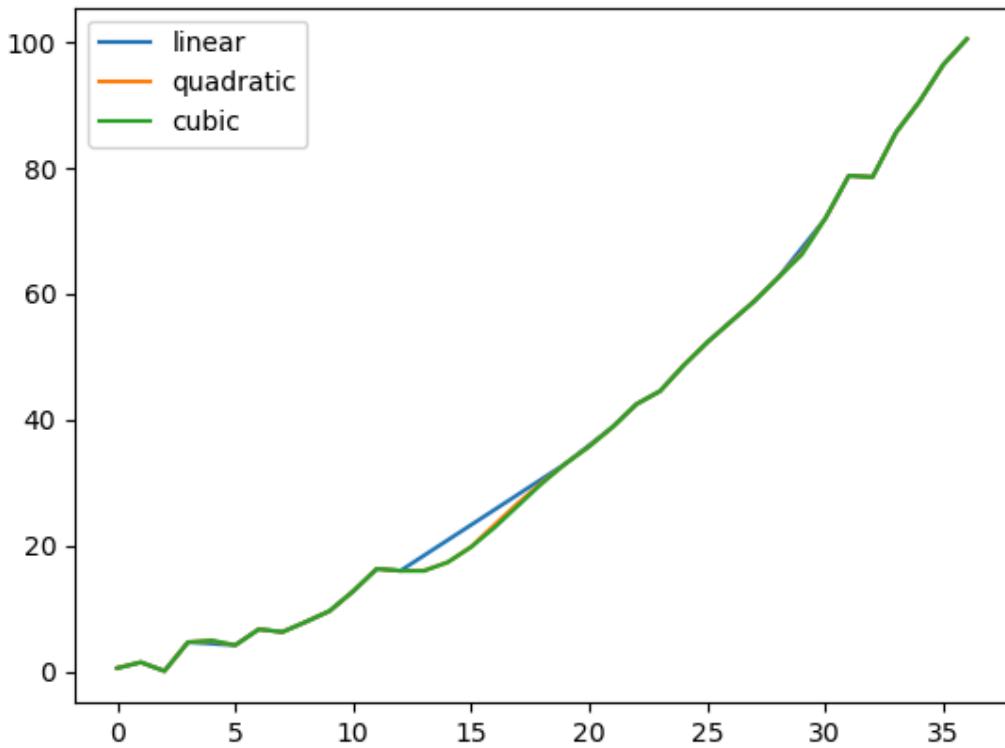


Figura 2.3: Comparación de interpolaciones utilizando pandas obtenido de “Working with missing data” (*Working With Missing Data — Pandas 2.3.3 Documentation*, 2025)

Dónde Python destaca más que otras alternativas es en aprendizaje automático, el cual comúnmente hace uso de operaciones algebraicas, las cuales pueden ser paralelizadas gracias a librerías como *NumPy* o *SciPy* mencionadas anteriormente , sobre matrices multidimensionales (Raschka et al., 2020), para las cuales tanto *NumPy*, *SciPy* o *pandas* proveen estructuras para representarlas, cómo la clase *DataFrame* de *pandas*, una tabla mutable de dos dimensiones que permite alinear operaciones sobre sus columnas o filas. Un ejemplo más específico de librería de análisis de datos faltantes en series temporales es *PyPOTS*; es de código abierto y se centra en la minería de datos y el análisis de series multivariadas con datos faltantes; permitiendo: imputación, previsión, detección de anomalías, clasificación y agrupamiento (W. Du, 2023). Para la imputación, *PyPOTS* posibilita el uso de varios modelos como *Transformer*, *TEFN* y *SAITS*.

Con respecto al trazado de gráficos, se podría usar la propia funcionalidad de *pandas*. También existe la popular *Matplotlib*, una amplia librería para crear gráficos estáticos, animados e interactivos en Python (Hunter, 2007). *Matplotlib* nace emulando los comandos de *MATLAB*, un lenguaje con gran capacidad de trazado pero muy limitado en otros ámbitos en comparación a Python, y busca crear gráficos de forma directa y simple. Lo que es más, a través de la clase *Basemap*, *Matplotlib* permite crear gráficos sobre mapas. No obstante, también sería posible usar *Folium*, una biblioteca que mezcla la manipulación de datos de Python con el trazado de mapas de la librería Leaflet.js de Javascript (Story, 2025). Permite la exportación de los mapas a formato HTML y añadirles marcadores, capas y polígonos.

Finalmente, la creación de la interfaz de usuario sería posible con el paquete *tkinter*, ya incluido en las versiones actuales de Python, que es un conjunto de distintos módulos cada uno

con su propia funcionalidad y documentación (*Tkinter — Python Interface To Tcl/Tk*, 2025). Tkinter es la interfaz a las herramientas de Tcl, un lenguaje dinámico de interpretación; Tk, un paquete de Tcl implementado en C; y Ttk, una nueva familia de widgets de Tk. Sin embargo, también existen opciones para GUIs más complejas como PyQt que permite el uso del sistema Qt, un framework multiplataforma orientado a objetos en C++ de código abierto que es ampliamente utilizado para el desarrollo de interfaces de usuario (Riverbank Computing, 2025b).

Concluyendo, Python es completo y flexible. Su mayor debilidad para este proyecto puede ser la falta de especialización, sin embargo, debido a su popularidad presenta opciones para todo tipo de funcionalidades e implementaciones de otros lenguajes.

2.2.3 Otras soluciones

Además de los lenguajes más conocidos como lo son R y Python, también existen otros específicos para el análisis de datos como lo es MATLAB y Julia.

MATLAB es tanto un entorno de computación numérica como un lenguaje de programación de comandos orientado a la manipulación de matrices y álgebra lineal. Nace en los años setenta como una calculadora interactiva de matrices y aparece como lenguaje de programación y software comercial en 1984. Matlab presenta multitud de “cajas de herramientas” especializadas para diferentes disciplinas que se apoyan en algoritmos matemáticos, como aprendizaje automático, y gran capacidad para trazado de gráficos de calidad, incluido dibujo sobre mapas (*MATLAB - el Lenguaje del Cálculo Técnico*, 2024). Incluso presenta opciones para la creación de interfaces de usuario utilizando MATLAB GUI. No obstante, al tratarse de software y lenguaje propietario es necesario pagar una licencia para su uso y para el despliegue de aplicaciones.

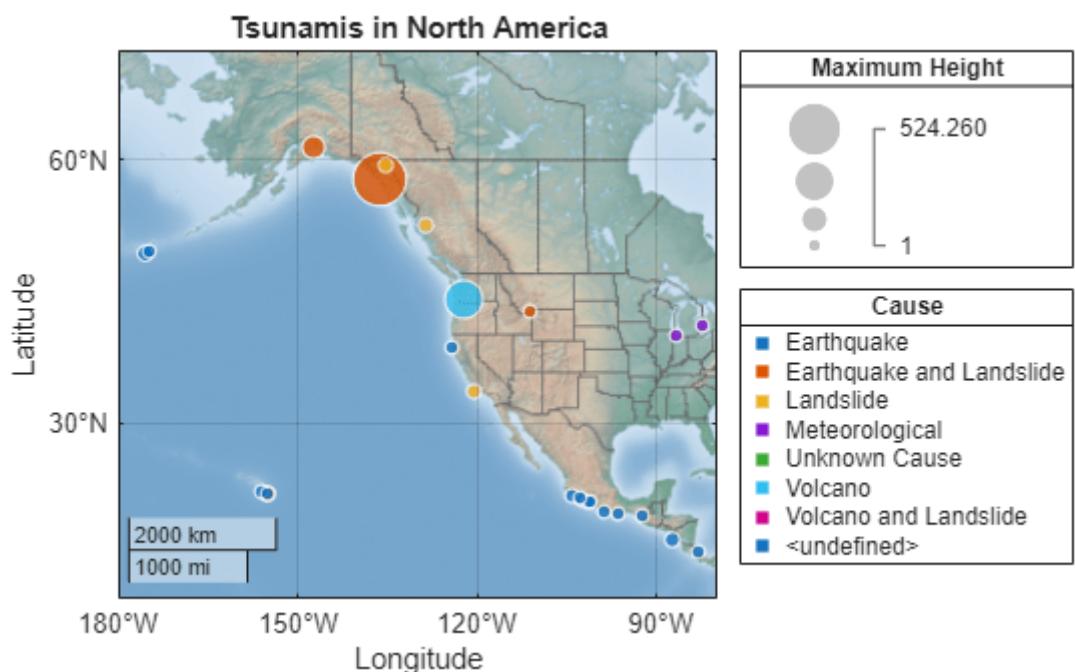


Figura 2.3: Ejemplo de gráfica de burbujas geográficas independientes extraído de (*Crear Mapas Con Datos de Latitud y Longitud*, 2025)

Por otro lado, Julia es una alternativa de código abierto diseñada para alto rendimiento. Al igual que Python es de tipado dinámico, pero no soporta clases con métodos encapsulados (Bezanson et al., 2017). Está diseñado para el paralelismo y para especializarse en la computación numérica y científica, y provee herramientas para el procesamiento de datos tales como DataFrames.jl, para trabajar con conjuntos de datos; CSV.jl, para leer archivos csv; Plots.jl, para funcionar como interfaz común para visualizar datos con bibliotecas como GR.jl y PyPlot.jl. Para la creación de aplicaciones cuenta con Genie (HPC Stipple SL, 2025), un marco de trabajo para crear aplicaciones web.

Finalmente, también existe la posibilidad de utilizar un lenguaje distinto para la interfaz de usuario y la manipulación de los datos. Por ejemplo, Jython (The Python Software Foundation, 2025) presenta una implementación de Python en Java que permite la interacción entre ambos y cuenta con licencias no comerciales y comerciales. Por otro lado, JPype (*JPype Documentation — JPype 1.6.1.dev0 Documentation*, 2018) es un módulo de código abierto que da acceso a Java desde Python, permitiendo usar sus librerías y estructuras. Otra alternativa podría ser Cython, un derivado de código abierto de Python, que añade soporte para llamar funciones y declarar tipos de C y C++. Aunque encontrar el marco de trabajo correcto podría presentar las ventajas de ambos lenguajes, también añade bastante complejidad al proyecto.

Capítulo 3. Descripción de la propuesta

La propuesta de este trabajo es desarrollar una aplicación específica para la gestión, análisis y visualización de series temporales.

3.1 Metodología de gestión del proyecto

El desarrollo se llevará a cabo siguiendo una metodología ágil; un enfoque que enfatiza proveer software funcional de forma continua, responder a la retroalimentación del cliente eficazmente durante el desarrollo y adaptarse al cambio de requisitos durante este (Cunningham, 2001). La metodología ágil específica será una simplificación de Scrum, que no es un proceso estandarizado, si no un marco de desarrollo flexible, incremental e iterativo, para organizar el trabajo basado en los principios ágiles (Rubin, 2012). Scrum es un término que se toma del rugby que se refiere al reinicio del juego tras una falta accidental o cuando el balón ha salido fuera de juego, que surge en 1986 acuñado por Hirotaka Takeuchi y Ikujiro Nonaka en “*The New New Product Development Game*” (Takeuchi & Nonaka, 1986). Comparan la forma en la que el balón es pasado entre los miembros de un equipo con el nuevo enfoque holístico de las empresas de Japón y Estados Unidos caracterizado por fases que se superponen, control sutil y autoorganización de los equipos de un proyecto, entre otros (Takeuchi & Nonaka, 1986). La razón de utilizar esta metodología es posibilitar comenzar el proyecto sin necesidad de planificación excesiva y utilizar la flexibilidad que permite para adaptarse a los requisitos a medida que estos se vayan descubriendo. Además, Scrum es una solución viable para equipos que se concentran enteramente en el desarrollo de software y garantiza cierto nivel de calidad al requerir constante comunicación entre los involucrados y conocimiento común de los problemas y cambios (Srivastava et al., 2017).

Scrum presenta tres roles: *Product Owner*, el propietario del producto quién protege los intereses del cliente y garantiza el incremento de valor tras cada *sprint*; *ScrumMaster*, que enfoca como debe trabajar el equipo; y *Development Team*, los profesionales dedicados al desarrollo (Abuchar, 2023). Sin embargo, en esta simplificación, al ser un equipo de una sola persona ésta toma los roles de *Development Team* y *ScrumMaster*, mientras que gran parte de las responsabilidades del *Product Owner* pasan a ser del cliente, quien participa intrínsecamente en el proceso. Esto garantiza que el cliente guíe el desarrollo según sus necesidades y comparta su mayor nivel de experiencia en el dominio del problema.

Por otra parte, la unidad principal de tiempo en el Scrum es el *sprint*, mínimo periodo en el que se genera valor (Abuchar, 2023). Cada uno se considera una iteración, derivando de la técnica de *timeboxing* o bloque de tiempo, que asigna un periodo de tiempo inamovible para realizar determinadas tareas. La duración suele ser entre dos semanas y un mes, este límite de tiempo fuerza la priorización, motiva la finalización de tareas y controla el alcance de desarrollo (Rubin, 2012). El conjunto de tareas que se realizan durante un *sprint* con el objetivo de proveer la funcionalidad prometida es conocido como *sprint backlog*, o pila del *sprint*, y define los planes de diseño, implementación, integración y testeo para este periodo, que son el objetivo a cumplir para la iteración (Rubin, 2012). Por otro lado el *product backlog*, o pila del producto, es la lista de necesidades y requerimientos completos del proyecto, estas se reafirman en cada iteración y es el dueño del producto quien decide la priorización (Abuchar, 2023). Cualquier

artículo de la pila del producto puede ser revalorizado, eliminado o añadido al ritmo que las condiciones de negocio cambien o el entendimiento del producto evolucione tras recibir retroalimentación, este proceso de obtención de elementos y refinamiento de la pila de producto se conoce como *grooming* (Rubin, 2012).

Tanto la pila de *sprint* como la pila del producto son artefactos de Scrum. Los artefactos son los productos tangibles que resultan del proceso de desarrollo y su existencia busca crear consistencia entre los equipos que utilizan el mismo modelo de desarrollo (Bass, 2016). Otro artefacto principal es el incremento, el resultado o fragmento del producto conseguido al finalizar las tareas de cada *sprint*. Scrum se considera incremental porque tras cada iteración resulta un incremento.

Una descripción más detallada de los *sprints* en los que se divide este proyecto, el flujo de trabajo y las tareas realizadas se llevará a cabo en la sección de Planificación Temporal de este documento.

3.2 Requisitos

Los requisitos en desarrollo software son las capacidades, condiciones o estándares que se deben cumplir para satisfacer un acuerdo de desarrollo software (Macaulay, 1996). Estos son acordados por todos los participantes. Su obtención es necesaria para identificar las necesidades y cómo cumplirlas. Asimismo, fijan las expectativas para declarar el desarrollo exitoso.

En contraste con los requisitos en el desarrollo secuencial, los cuales son detallados al comienzo y pretenden ser inmutables, Scrum ve los requisitos como algo manipulable para lograr los objetivos empresariales. Si, por ejemplo, durante el desarrollo aparecen circunstancias que dictan que algún requisito es menos favorable en comparación a otro o se descubre una necesidad esencial faltante, los requisitos se adaptarán (Rubin, 2012). Con esta metodología se evita delinejar prematuramente requerimientos que luego se desvelen innecesarios y se puede comenzar a desarrollar sin necesidad de tener el documento de requisitos completo. Al no conocer todos los requisitos al completo se utilizan “comodines” que deben ser generales y negociables al comienzo, y a lo largo del proyecto se detallarán suficientemente para pertenecer a la pila de un *sprint*. Estos “comodines” son los elementos de la pila de producto, los cuales se pueden representar de diversas maneras, por ejemplo, en este documento se presentan como historias de usuario y se obtienen mediante conversaciones, pero también sería posible exponerlos como casos de uso. Las historias de usuario tienen distinta magnitud según el nivel de detalle que presenten, las más amplias y abstractas son útiles para planear en alto nivel y las más específicas corresponden a un nivel más bajo de abstracción y más adecuado para la pila de *sprint* (Rubin, 2012). Además, cada historia de usuario presenta un nivel de prioridad, que es la importancia que deben tomar en el desarrollo, y ciertos criterios de aceptación, que deben cumplirse para considerar el desarrollo exitoso. Para facilitar la comunicación, se acuerda durante las conversaciones el siguiente glosario de términos simples:

Serie o secuencia temporal: Colección de medidas tomadas a lo largo de un tiempo finito, ordenadas en una sucesión determinada por el momento de captura.

Valores de una serie temporal: Cada medida o registro tomados por los sensores que conforman la serie temporal.

Valores perdidos o faltantes: Medidas faltantes del conjunto de datos.

Imputación: Acción de sustituir valores faltantes de una secuencia por otros que han sido estimados.

Método de imputación: Técnica o algoritmo utilizado para realizar la imputación, es decir, la secuencia de pasos seguidos para obtener un valor por el que sustituir el valor faltante.

A continuación, en las tablas de la 3.1 a la 3.5, se representan las historias de usuario que se han acordado con el cliente.

Tabla 3.1: Historia de usuario 1, visualizar secuencias temporales

HU-1	Visualizar secuencias temporales
Historia de usuario	Como miembro del equipo de investigación quiero visualizar los valores de secuencias temporales en un mapa o en una gráfica de líneas para analizar su comportamiento en el tiempo, prestando atención a las coordenadas donde se realizó la medición o sólo fijándose en la secuenciación temporal.
Criterios de aceptación	<ol style="list-style-type: none"> 1. La aplicación permite seleccionar entre ambos modos de visualización y cambiar entre ellos fácilmente. 2. La visualización en el mapa debe mostrar qué valor se encuentra en cada coordenada. 3. La visualización en la gráfica es en dos dimensiones, una correspondiente al orden o tiempo secuencial y la otra al valor correspondiente. 4. Ambos modos muestran una o varias secuencias temporales elegidas por el usuario.
Descripción	Esta historia de usuario se centra en la visualización de valores de dos formas distintas. Esta función pretende dar al usuario libertad para elegir el método de visualización que más se adapte a las necesidades de su análisis. Es dependiente del formato en el que se almacenan las secuencias temporales.
Prioridad	Alta

Tabla 3.2: Historia de usuario 2, imputar valores estimados a secuencias temporales con valores perdidos

HU-2	Imputar valores estimados a secuencias temporales con valores perdidos
Historia de usuario	Como miembro del equipo de investigación quiero imputar secuencias temporales con valores perdidos utilizando diversos métodos de imputación para completar la secuencia temporal.
Criterios de aceptación	<ol style="list-style-type: none"> 1. La aplicación permite seleccionar el método de imputación que se le aplica a una secuencia. 2. La secuencia original se conserva y se puede visualizar junto al resultado. 3. El resultado es visualizable inmediatamente finaliza la imputación y, si existe, se puede comparar con la secuencia temporal original completa.

Descripción	Esta historia de usuario se enfoca en la imputación de secuencias, permitiendo la elección de la metodología de imputación por el usuario, y mostrar los resultados obtenidos. Depende del formato en el que se almacenan las secuencias temporales y el sistema de visualización.
Prioridad	Alta

Tabla 3.3: Historia de usuario 3, manipulación de secuencias temporales

HU-3	Manipulación de secuencias temporales
Historia de usuario	Como miembro del equipo de investigación quiero fusionar, duplicar y eliminar valores de secuencias temporales para crear secuencias temporales útiles para el estudio de la efectividad de diferentes metodologías de imputación.
Criterios de aceptación	<ol style="list-style-type: none"> La eliminación permite seleccionar a qué rango temporal pertenecen los valores (ráfaga de datos perdidos) o ser un porcentaje de valores aleatorios (datos aislados). El original y el resultado de cualquier operación pueden ser diferenciados fácilmente en la interfaz. Los valores borrados serán sustituidos por el valor correspondiente a “no disponible”.
Descripción	Esta historia de usuario se enfoca en la manipulación de secuencias temporales. La funcionalidad tiene como objetivo que se pueda modelar cualquier secuencia para incrementar su utilidad en el momento de analizar un método de imputación. Depende del formato en el que se almacenan las secuencias temporales y el sistema de visualización.
Prioridad	Baja

Tabla 3.4: Historia de usuario 4, importar archivos csv con medidas de sensores

HU-4	Importar archivos csv con medidas de sensores
Historia de usuario	Como miembro del equipo de investigación quiero importar archivos csv, con las medidas tomadas por un sensor, para gestionar e imputar las secuencias temporales que contienen.
Criterios de aceptación	<ol style="list-style-type: none"> Un diálogo permite seleccionar qué archivo o archivos van a importar. La importación debe aceptar solo archivos csv. El formulario de importación debe mostrar opciones para especificar si se desea obtener una sola secuencia temporal o varias de la importación. La aplicación debe mostrar retroalimentación visual para confirmar la correcta importación.
Descripción	Esta historia de usuario se centra en el proceso de importación de un archivo csv para poder obtener secuencias temporales sobre las

que operar. El funcionamiento es dependiente de la verificación de archivos y el formato en el que se almacenan las secuencias temporales.

Prioridad	Alta
------------------	------

Tabla 3.5: Historia de usuario 5, exportar resultados a archivos csv

HU-5	Exportar resultados a archivos csv
Historia de usuario	Como miembro del equipo de investigación quiero exportar los resultados de la modificación o imputación de cualquier secuencia temporal para almacenarlo y compartirlo.
Criterios de aceptación	<ol style="list-style-type: none"> 1. La aplicación debe permitir seleccionar cualquier secuencia temporal para someterse al proceso de exportación. 2. Varias secuencias se pueden exportar a la vez, pudiendo elegir si fusionarlas o exportarlas en distintos archivos. 3. Todos los archivos resultantes de esta operación serán del tipo csv.
Descripción	Esta historia de usuario se centra en el proceso de exportación de secuencias temporales a archivos csv para su distribución y almacenaje. El funcionamiento es dependiente del formato en el que se almacenan las secuencias temporales.
Prioridad	Media

En las siguientes subsecciones se presenta la lista de requisitos final y en la sección 3.3 Planificación Temporal, se detalla cuando surgen los requisitos. Puesto que no se utiliza una base de datos y un usuario no puede tener acceso a información de otros, no es necesario describir requisitos de información.

3.2.1 Requisitos funcionales

RF-1. Gestión de secuencias temporales

- RF-1.1.** La aplicación permitirá al usuario añadir secuencias temporales a la aplicación.
- RF-1.2.** Será posible eliminar valores de una secuencia temporal.
- RF-1.3.** La modificación de una secuencia temporal no implicará la pérdida de los valores originales.
- RF-1.4.** Se podrá extraer una secuencia temporal de un archivo externo.
- RF-1.5.** Los valores de cualquier secuencia temporal podrán ser guardados en archivos externos.

RF-2. Imputación de secuencias temporales

- RF-2.1.** Se podrá realizar la imputación sobre una secuencia temporal con valores faltantes.
- RF-2.2.** El método de imputación utilizado se podrá seleccionar a través de la interfaz.

RF-2.3. Será posible comparar la secuencia obtenida tras la imputación con la secuencia original.

RF-3. Visualización de datos

RF-3.1 Se debe poder visualizar los datos de cualquier secuencia temporal agregada a la instancia de la aplicación, así como los resultados de cualquier modificación a una secuencia.

RF-3.2 Varias secuencias temporales podrán ser visualizadas de forma simultánea.

RF-3.3 La visualización de las secuencias temporales debe permitir apreciar las coordenadas geográficas de las medidas.

RF-3.4 Desde la interfaz de usuario se podrá controlar cuándo una secuencia temporal se encuentra visible u oculta.

3.2.2 Requisitos no funcionales

RNF-1. Facilidad de uso: La interfaz de usuario debe ser suficientemente intuitiva y sencilla para que una persona sin experiencia sea capaz de darle uso tras un aprendizaje de menos de 30 minutos de duración.

RNF-2. Accesibilidad: Toda la información necesaria para su uso y extensión deberá ser fácilmente accesible mediante un manual de usuario o documento similar.

RNF-3. Plataforma: La aplicación debe funcionar en el sistema operativo Windows y Linux.

RNF-4. Estándar admitido: Los archivos externos aceptados serán del tipo Comma Separated Values o CSV.

RNF-5. Extensibilidad: La lógica interna, la interfaz y los métodos de imputación deben ser fácilmente extensible por otros desarrolladores con conocimiento de las tecnologías utilizadas y acceso a la documentación del proyecto.

RNF-6. Licencia: La aplicación no necesitará la compra de licencias externas para su uso y distribución.

3.3 Presupuesto

El presupuesto se discute en la primera reunión en donde se hace una estimación y se acuerda durante la segunda reunión tras una contemplación más profunda del problema. Aunque el cliente tiene una visión clara de las capacidades deseadas, debido a la naturaleza de los proyectos nacidos de la metodología ágil, los requisitos pueden crecer durante el desarrollo y eso se debe contemplar a la hora de negociar. Primero se discute las capacidades deseadas y la formación a realizar por el equipo de desarrollo. Se acuerda que el cliente se posicionará como el experto en lo que respecta a la manipulación e imputación de secuencias temporales y el equipo, además de realizar el desarrollo, funcionará como asesor para elegir las técnicas y herramientas más adecuadas para la implementación. El equipo será responsable de realizar la formación pertinente y de informarse lo necesario sobre el tema de la aplicación.

Tras esto, se discute el tiempo de desarrollo estimado y puesto que un trabajo de fin de grado supone 12 créditos y cada uno debe ser unas 25 horas de trabajo, se hace una aproximación de 300 horas. El sueldo base promedio de un programador/a junior en España es de 20.911€ al año o 10,47€ por hora según la página de Indeed España (Indeed, 2025b) lo que coincide con lo que muestra Glassdoor con un sueldo base de 18.000€ hasta 24.000€ al año (Glassdoor, 2025). Teniendo en cuenta que el proyecto requiere cierto grado de investigación

para realizar la correcta asesoría respecto a tecnologías y formación extra del equipo, se decide un sueldo de 15,5€ la hora, bastante más cercano al sueldo promedio de un programador con cierta experiencia o el de un investigador asociado, que es 19,60€/hora (Indeed, 2025a).

Asimismo, se discuten los costes de desarrollo teniendo en cuenta el hardware y las licencias de software necesarias para el uso y despliegue. Por el lado de hardware, la máquina de desarrollo es un portátil HP OMEN 15 con un procesador AMD Ryzen 7 4800H, 16 GB de RAM y un SSD de 512GB que en el momento de compra costó alrededor de 900€ y se consideran 4 años de vida útil o un valor de 225€ al año. Además, el sensor utilizado para las pruebas procede de un kit de montaje con un coste de 63,30€ (*Sensor Community KIT (SDS011/BME280), English Language, Harness Cable Edition*, 2025) en el momento de la consulta, aunque este ha sido un préstamo por los directores de proyecto y por tanto se considera hardware que ya pertenecería al cliente y no es necesario incluirlo. Por el lado del software, la instalación de Windows 11 conlleva un gasto extra de 145€ como detalla Microsoft en su web comercial (Microsoft, 2025). Mientras que el uso de PyQt5 no necesita de una licencia, siempre y cuando, la distribución de la aplicación se mantenga bajo la licencia *GNU General Public License* (Free Software Foundation, Inc., 2007), la cual obliga a mantener disponible el código fuente. En caso de no querer distribuir el código fuente o tener motivaciones comerciales es necesario adquirir una licencia comercial para PyQt5 de 600€ (Riverbank Computing, 2025a) y obtener la versión comercial de Qt o ceñirse a su licencia LGPL. En la tabla 3.6 se muestra la estimación del presupuesto, incluyendo como coste opcional la licencia PyQt.

Tabla 3.6: Resumen del presupuesto

PRESUPUESTO			
Descripción	Cantidad	Precio	Total
Desarrollador	760 horas	15,5€/hora	11.780€
Licencia Windows 11	1 unidad	145€	145€
Ordenador	0,5 años	225€/año	112,5€
			Total 12.037,5€
Licencia PyQt5 (opcional)	1 unidad	600€	600€
			Total con gastos opcionales 12.637,5€

3.4 Planificación temporal

Como se describe al principio del capítulo, el desarrollo se divide en *sprints*. Para determinar qué elementos de la pila de producto se llevan a cabo en cada uno se realiza una planificación del *sprint* previamente, en la que se define una meta del *sprint*, estableciendo qué se quiere lograr y sirviendo para priorizar elementos de la pila de producto, y se forma la pila del *sprint*, una segunda pila que define las tareas a completar (Rubin, 2012). Asimismo, al final de cada sprint se realiza una revisión y retrospectiva del *sprint*, una conversación enfocada a observar las funcionalidades completadas y ofrecer retroalimentación para guiar el desarrollo

futuro y sirve como una discusión sobre lo que ha funcionado o no del proceso, con el objetivo de mejorarlo.

En esta sección, se muestra el desarrollo de cada sprint. Cada uno incluye la fecha de comienzo, la fecha de finalización, la pila de *sprint*, las tareas realizadas y el estado en el que acaban tras el *sprint* (“Completada”, la tarea se finaliza en el *sprint* correspondiente, o “Abierta”, la tarea no ha sido finalizada en ese *sprint*), la duración y la capacidad, que es la cantidad de días hábiles asignados. Estos datos se muestran en las tablas descriptivas de cada sprint que incluyen desde la 3.7 hasta la 3.26, excluyendo 3.9, al ser una comparativa de las posibles tecnologías de desarrollo, y 3.15 y 3.20, que describen historias de usuario definidas durante el desarrollo; además se adjuntan figuras para describir la división de tareas por *sprint* y los incrementos resultantes (Figuras 3.1 - 3.26). Cada tarea incluye una estimación en días hábiles que tomará, se da por hecho que este tiempo contiene la generación de una documentación mínima necesaria aunque luego existan tareas de documentación aparte. Igualmente, el comienzo de cada *sprint* incluye una reunión con el cliente, la cual sirve como revisión y retrospectiva del sprint anterior, y planificación del siguiente y se indica el momento en el surge cada requisito o cambia alguna decisión de diseño, junto a apuntes relevantes. Se cierra la sección con un diagrama Gantt que describe la planificación final y un resumen de las tareas realizadas agrupándolas en paquetes de trabajo en base a su motivación principal como diseño, documentación o implementación.

El razonamiento de utilizar días en vez de horas es abstraer de forma simple la unidad de tiempo, de forma que sea más sencillo relativizar la duración de una tarea en comparación a las demás en un *sprint* y representar de forma más directa la duración de las tareas sobre el diagrama, como se mostrará en la sección 3.4.2 de este documento. Por ejemplo, en un *sprint* de dos semanas de duración, eliminando los fines de semana, hay 10 días hábiles para realizar el desarrollo de tres tareas A, B y C asignando 5, 3 y 2 días a cada una; entonces, utilizando los días se puede crear la correspondencia siguiente: de lunes a viernes de la primera semana se realiza A, de lunes a miércoles de la segunda semana se atiende B y la tarea C obtiene los días restantes; de esta forma, la disposición en el diagrama se obtiene directamente y no es necesario describir que a lo mejor un día se realizaron 4 horas de trabajo real y otro sólo 1.

3.4.1 Descripción de los sprints

Tabla 3.7: Descripción del sprint 1

SPRINT 1			
Fecha de comienzo		Miércoles, 20 de noviembre de 2024	
Fecha de finalización		Miércoles, 4 de diciembre de 2024	
Duración	14 días	Capacidad	10 días
Objetivo	Analizar las necesidades del proyecto y cómo satisfacerlas		
Pila de <i>sprint</i>			
ID	Tareas	Estado	Estimación

T001	Definición de requisitos provisionales	Completado	3 días
T002	Investigación posibles herramientas de desarrollo	Completado	7 días

SPRINT 1		2024									
ID	Tarea	20-11	21-11	22-11	25-11	26-11	27-11	28-11	29-11	2-12	3-12
T001	Definición de requisitos provisionales										
T002	Investigación posibles herramientas de desarrollo										

Figura 3.1: Distribución de tareas por días en el *sprint* 1

Durante la primera reunión se tiene una conversación inicial con el cliente en la que se definen las expectativas iniciales y la metodología de trabajo. Se propone la metodología ágil explicada utilizando sprints de dos semanas, con posibilidad de variabilidad según disponibilidad. Asimismo, se definen y discuten ciertos requisitos básicos:

- Visualización de datos (RF-3)
- Capacidades de imputación (RF-2)
- Plataforma (RNF-3)
- Estándar en el que se almacenan los datos (RNF-4)
- Facilidad de uso (RNF-1)

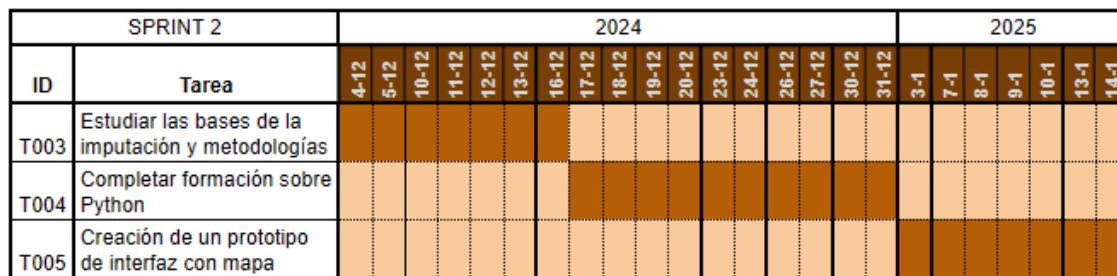
Estos se refinan durante el sprint para continuar la conversación en la siguiente reunión y se especifican: la capacidad de imputación (RF-2.1) y el control sobre la misma (RF-2.2), y la apreciación de las coordenadas en la visualización (RF-3.3).

Las tareas de este sprint son de análisis e investigación, por lo que no están relacionadas a las historias de usuario como tal. No obstante, el objetivo es encontrar la forma de satisfacer las necesidades que nacen de estas.

Tabla 3.8: Descripción del sprint 2

SPRINT 2			
Fecha de comienzo		Miércoles, 4 de diciembre de 2024	
Fecha de finalización		Miércoles, 15 de enero de 2025	
Duración	42 días	Capacidad	24 días
Objetivo	Completar formación y crear el primer prototipo		
Pila de <i>sprint</i>			
ID	Tareas	Estado	Estimación
T003	Estudiar las bases de la imputación y metodologías	Completada	7 días

T004	Completar formación sobre Python	Completada	10 días
T005	Creación de un prototipo de interfaz con mapa	Completada	7 días

Figura 3.2: Distribución de tareas por días en el *sprint 2*

Se comienza por una retrospectiva del *sprint 2*, en la que se presenta un resumen y comparativa de las herramientas investigadas en base a la capacidad de imputación, visualización de datos y desarrollo de un programa e interfaz (Tabla 3.10). Puesto que la experiencia del equipo de desarrollo con cada lenguaje es bastante reducida, se añade una categoría extra que expresa la facilidad de aprendizaje, incluyendo la consideración de que el software se deba extender en el futuro.

Todos las alternativas estudiadas cumplen con un mínimo de capacidad para las distintas áreas de comparación. Por ello, se establecen tres niveles para expresar la satisfacción en cada ámbito:

SE - Sobre pasa expectativas. El lenguaje es más que apto para el desarrollo y no presenta impedimentos.

CN - Cubre necesidades. El lenguaje es apto para esta área y permitiría el desarrollo de la aplicación sin inconvenientes notables.

AC - Ámbito carente. La utilización del lenguaje presentaría significantes complicaciones al desarrollo.

Tabla 3.9: Comparación de las tecnologías de desarrollo

	Imagen	Visualización	Aplicación e interfaz	Aprendizaje
R	SE. Lenguaje especializado	SE. Librerías de calidad	CN. Opciones limitadas	CN. Sintaxis simple pero distinta a lo que acostumbra el equipo
Python	CN. Propósito general	SE. Librerías calidad	SE. Mayor variedad de opciones	SE. Comunidad grande, sintaxis simple
MATLAB	SE. Lenguaje especializado	SE. Librerías calidad	CN. Opciones limitadas MATLAB GUI	AC. Licencia de pago

Julia	CN. Lenguaje especializado, métodos prefabricados limitados	SE. Librerías calidad	AC. de Alternativas carentes	CN. Comunidad algo menor
Soluciones mixtas	SE. Capacidad para poseer las características positivas de ambos lenguajes involucrados		AC. Se debe aprender varios lenguajes y un marco de trabajo concreto	

Durante la reunión se decide que se valora positivamente la capacidad de extensibilidad (RNF-5) y que el programa resultante no debe necesitar licencias de pago (RNF-6).

En el proceso de análisis MATLAB y Julia son los primeros en ser descartados. Aunque fueron considerados por sus grandes capacidades y especialización en cálculos matemáticos, presentan demasiadas dificultades. Por un lado, MATLAB necesita una licencia de pago, lo que no sólo dificulta el aprendizaje, si no que también afectaría al coste de desarrollo. Por otro lado, Julia tiene capacidades carentes para el desarrollo de interfaces en comparación a otras opciones más versátiles.

Con respecto a la opción de utilizar un marco de trabajo que permita la mezcla de dos lenguajes, en principio se sopesó con intención de utilizar un lenguaje ya conocido por el equipo y con el que tuvieran más experiencia desarrollando interfaces como lo son C, C++ o Java. Sin embargo, la dificultad añadida de aprender el propio marco de trabajo y el segundo lenguaje para la computación convierte esta alternativa en similar a simplemente aprender un nuevo lenguaje que permita la computación y la creación de la interfaz. De igual forma, complicaría la extensión y el mantenimiento del software resultante, debido a que presentaría mayor complejidad estructural y dificultad de lectura del código.

Finalmente, la decisión entre R y Python es la más compleja. Ambos son muy utilizados en la computación de datos y aprendizaje automático, presentan herramientas robustas de trazado de gráficos y alternativas para el desarrollo de una interfaz. No obstante, la conclusión final es que la versatilidad de Python, su facilidad de aprendizaje y mejores capacidades para crear interfaces son más importantes para el proyecto que las ventajas del lenguaje especializado que es R ofrece. Las librerías utilizadas serán PyQt5 para la interfaz, pandas para la manipulación de las secuencias, Folium para la representación sobre el mapa y matplotlib para la representación sobre la gráfica de líneas. Una vez tomada esta decisión el equipo puede comenzar la formación y prototipado. El resultado es el primer incremento (Figura 3.3), una simple interfaz creada utilizando PyQt5 que muestra un mapa creado mediante Folium y marcadores que representan la información.

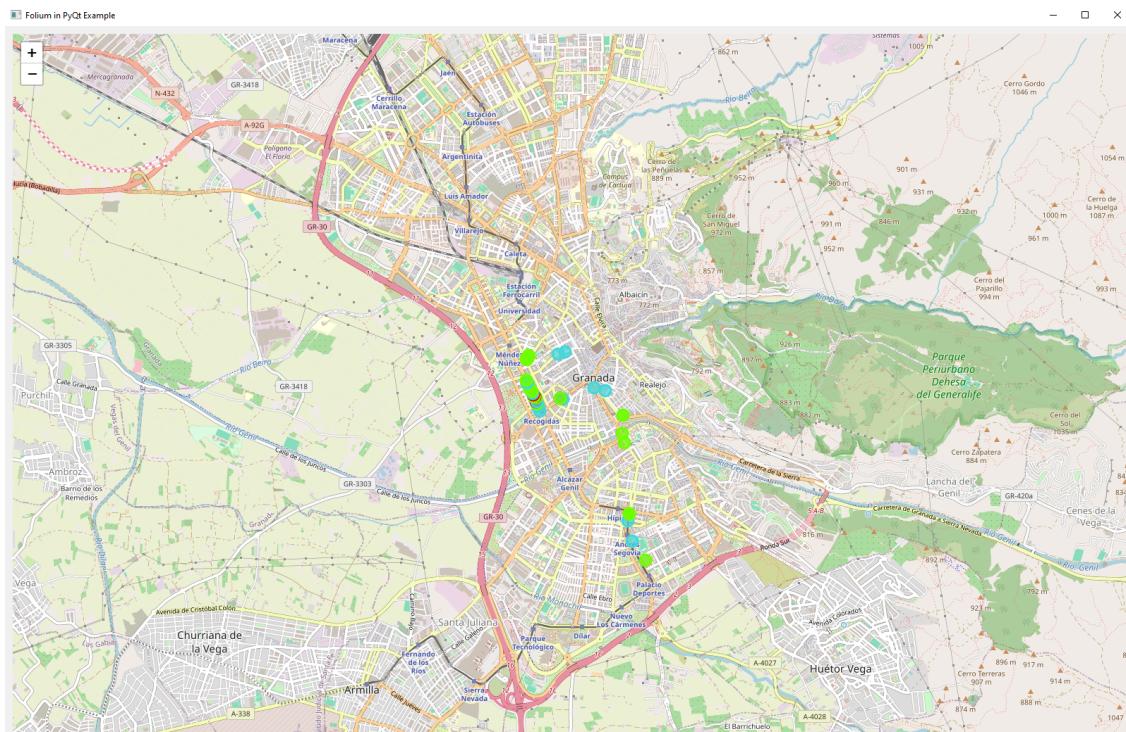
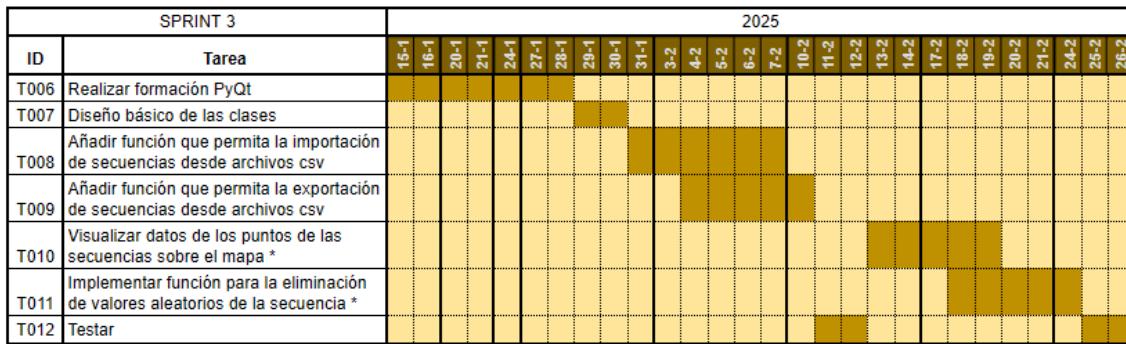


Figura 3.3: Prototipo de mapa con puntos

Tabla 3.10: Descripción del sprint 3

SPRINT 3			
Fecha de comienzo		Miércoles, 15 de enero de 2025	
Fecha de finalización		Jueves, 27 de febrero de 2025	
Duración	43 días	Capacidad	28 días
Objetivo	Implementar funciones básicas		
Pila de <i>sprint</i>			
ID	Tareas	Estado	Estimación
T006	Realizar formación PyQt	Completada	7 días
T007	Diseño básico de las clases	Completada	2 días
T008	Añadir función que permita la importación de secuencias desde archivos csv	Completada	4 días
T009	Añadir función que permita la exportación de secuencias desde archivos csv	Completada	3 días
T010	Visualizar datos de los puntos de las secuencias sobre el mapa	Abierta	4 días
T011	Implementar función para la eliminación de valores aleatorios de la secuencia	Abierta	4 días

T012	Testar	Completada	4 días
------	--------	------------	--------

Figura 3.4: Distribución de tareas por días en el *sprint 3*

Durante la reunión de planificación se decide que es necesario incluir un nuevo requisito referente a la manipulación de las secuencias temporales (RF-1), con el objetivo de que el propio programa permita preparar las secuencias para la imputación. Por ejemplo, las secuencias se deben de poder añadir al área del trabajo (RF-1.1) y se deben poder extraer de un archivo externo (RF-1.4) y exportar (RF-1.5). Además, el poder eliminar valores de la secuencia (RF-1.2) permite crear secuencias con datos faltantes artificiales que, junto a la secuencia original, posibilitan comparar y estudiar la eficacia de los diferentes algoritmos de imputación. Adicionalmente, los valores originales deben ser conservados de alguna forma y ser accesibles (RF-1.3).

Tabla 3.11: Descripción del sprint 2

SPRINT 4			
Fecha de comienzo		Jueves, 27 de febrero de 2025	
Fecha de finalización		Jueves, 13 de marzo de 2025	
Duración	14 días	Capacidad	9 días
Objetivo	Refinar el prototipo		
Pila de <i>sprint</i>			
ID	Tareas	Estado	Estimación
T010	Visualizar datos de los puntos de las secuencias sobre el mapa	Completada	2 (6) días
T011	Implementar función para la eliminación de valores aleatorios de la secuencia	Completada	1 (5) días
T013	Crear división para la visualización en forma de gráfica	Completada	3 días
T012	Testar	Completada	3 días

SPRINT 4		2025								
ID	Tarea	27-2	3-3	4-3	5-3	6-3	7-3	10-3	11-3	12-3
T010	Visualizar datos de los puntos de las secuencias sobre el mapa *									
T011	Implementar función para la eliminación de valores aleatorios de la secuencia *									
T013	Crear división para la visualización en forma de gráfica									
T012	Testar									

Figura 3.5: Distribución de tareas por días en el *sprint 4*

Al final del *sprint* anterior quedan tareas abiertas que se continúan durante este *sprint*, por ello el primer número es la cantidad de días que toma durante el el *sprint* actual, mientras que se marcan los días totales dedicados en paréntesis, es decir, el total sumando los días empleados en el anterior y en este. Además, se vuelve a realizar una tarea para testar el incremento, a diferencia del desarrollo software más tradicional donde las pruebas se llevan a cabo más cerca del final de este, en esta metodología es importante testar con frecuencia para poder implementar las correcciones lo antes posible y asegurar que cada incremento sea funcional. Es interesante comentar que no existe una tarea para implementar una función relacionada con un calendario y aún así aparece en la figura del incremento (Figura 3.6), esto es porque es simplemente una prueba visual para poder discutir la posibilidad de controlar la visualización acotando un intervalo.

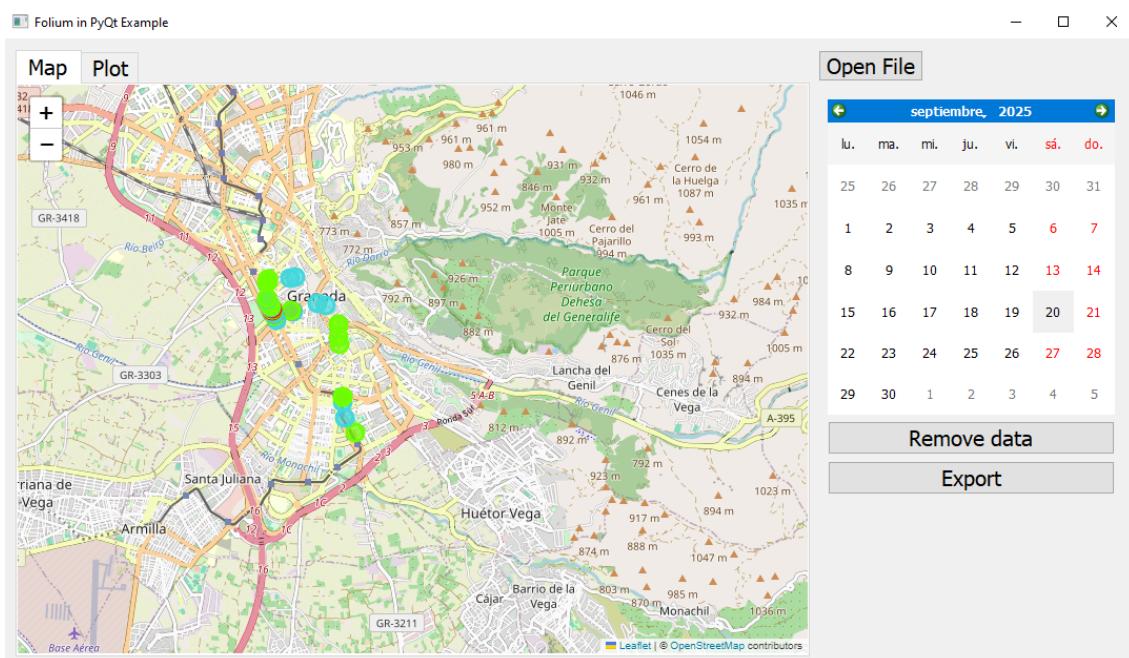


Figura 3.6: Incremento del sprint 4

Tabla 3.12: Descripción del sprint 5

SPRINT 5	
Fecha de comienzo	Jueves, 13 de marzo de 2025

Fecha de finalización		Jueves, 27 de marzo de 2025					
Duración	14 días		Capacidad	10 días			
Objetivo	Realizar la primera implementación de imputación y visualización en gráfica						
<i>Pila de sprint</i>							
ID	Tareas			Estado	Estimación		
T014	Implementar visualización en gráfico de líneas			Completada	4 días		
T015	Implementar imputación de secuencia con datos faltantes utilizando la media			Completada	3 días		
T016	Añadir comparativa de utilizando el cálculo de la raíz del error cuadrático medio (RECM)			Abierta	2 días		
T012	Testar			Completada	1 días		

SPRINT 5		2025									
ID	Tarea	13-3	14-3	17-3	18-3	19-3	20-3	21-3	24-3	25-3	26-3
T014	Implementar visualización en gráfico de líneas										
T015	Implementar imputación de secuencia con datos faltantes utilizando la media										
T016	Añadir comparativa de utilizando el cálculo de la raíz del error cuadrático medio (RECM) *										
T012	Testar										

Figura 3.7: Distribución de tareas por días en el *sprint 5*

Durante la reunión del *sprint* se decide una nueva funcionalidad. En el *sprint 3* se decidió que se debe poder manipular las secuencias para crear datos faltantes de forma artificial, y en este, se define en los requisitos que se debe poder comparar una secuencia imputada con la original si esta existe (RF-2.3). Esto es en dos sentidos, ambas secuencias se podrán visualizar a la vez (RF-3.2) y se podrá calcular el error cuadrático medio (RECM) entre ambas. Además en este *sprint* se implementa el primer método de imputación siendo este la sustitución de los valores faltantes por la media de una ventana móvil que engloba el valor anterior y el siguiente en el tiempo.

Tabla 3.13: Descripción del *sprint 6*

SPRINT 6			
Fecha de comienzo		Jueves, 27 de marzo de 2025	
Fecha de finalización		Jueves, 10 de abril de 2025	
Duración	14 días	Capacidad	10 días
Objetivo	Aumentar el repertorio de métodos de imputación		

Pila de sprint				
ID	Tareas	Estado	Estimación	
T016	Añadir comparativa de utilizando el cálculo de la raíz del error cuadrático medio (RECM)	Completada	1 (4) días	
T017	Implementar el método de imputación de relleno hacia atrás	Completada	2 días	
T018	Implementar el método de imputación de relleno hacia delante	Completada	2 días	
T019	Implementar el método de imputación usando la mediana	Completada	2 días	
T020	Incluir selector de método de imputación para PM2.5 y PM10	Completada	1 días	
T012	Testar	Completada	2 días	

SPRINT 6		2025									
ID	Tarea	27-3	28-3	31-3	1-4	2-4	3-4	4-4	7-4	8-4	9-4
T016	Añadir comparativa de utilizando el cálculo de la raíz del error cuadrático medio (RECM)										
T017	Implementar el método de imputación de relleno hacia atrás										
T018	Implementar el método de imputación de relleno hacia delante										
T019	Implementar el método de imputación usando la mediana *										
T020	Incluir selector de método de imputación para PM2.5 y PM10										
T012	Testar										

Figura 3.8: Distribución de tareas por días en el *sprint 6*

Puesto que se decidió que se podrían visualizar varias secuencias a la vez, se acuerda como requisito el control sobre la visualización de cada una (RF-3.4) y de los resultados de cualquier manipulación (RF-3.1) desde la interfaz de usuario. Para detallar el control que se debe tener sobre la interfaz se crea una nueva historia de usuario (Tabla 3.15).

Tabla 3.14: Historia de usuario 6, personalizar la visualización de series temporales

HU-6	Personalizar la visualización de series temporales
Historia de usuario	Como miembro del equipo de investigación quiero personalizar la visualización de forma que pueda elegir qué secuencias y series son visibles y de qué forma.
Criterios de aceptación	<p>1. Varias secuencias temporales pueden ser visualizadas a la vez.</p> <p>2. Se puede controlar la visibilidad de cada secuencia individualmente desde la interfaz de la aplicación.</p>

-
3. El resultado de una imputación o cualquier tipo de manipulación de datos de una secuencia se puede visualizar.
 4. Es posible cambiar entre la visualización de mapa y de gráfica de líneas de forma rápida e intuitiva.
-

Descripción	Esta historia de usuario se enfoca en la capacidad de personalizar cómo se visualizan las secuencias temporales. La meta de estas funciones es permitir al usuario control sobre la visualización para facilitar la comparación y estudio de las secuencias temporales. Depende del sistema de visualización y del flujo de trabajo esperado.
Prioridad	Media

Asimismo, se agregan tres métodos de imputación; relleno para delante y relleno para atrás, basados en *forward fill* y *backward fill* de pandas que sustituyen con el valor con la última observación válida y la siguiente válida, respectivamente; y sustituir por la mediana de forma similar como se hizo con la media. En la Figura 3.3 se puede ver el resultado del sprint 6, con dos selectores, actualmente en “Average”, que permiten seleccionar el método de imputación para las dos variables que se esperan en la serie temporal: PM2.5 y PM10; y dos etiquetas que mostrarán el resultado de calcular el RMSE de la secuencia introducida y el resultado de imputación. En el estado actual de la aplicación, sólo se puede importar y trabajar sobre una serie temporal al mismo tiempo y no hay forma en la interfaz de controlar la visualización.

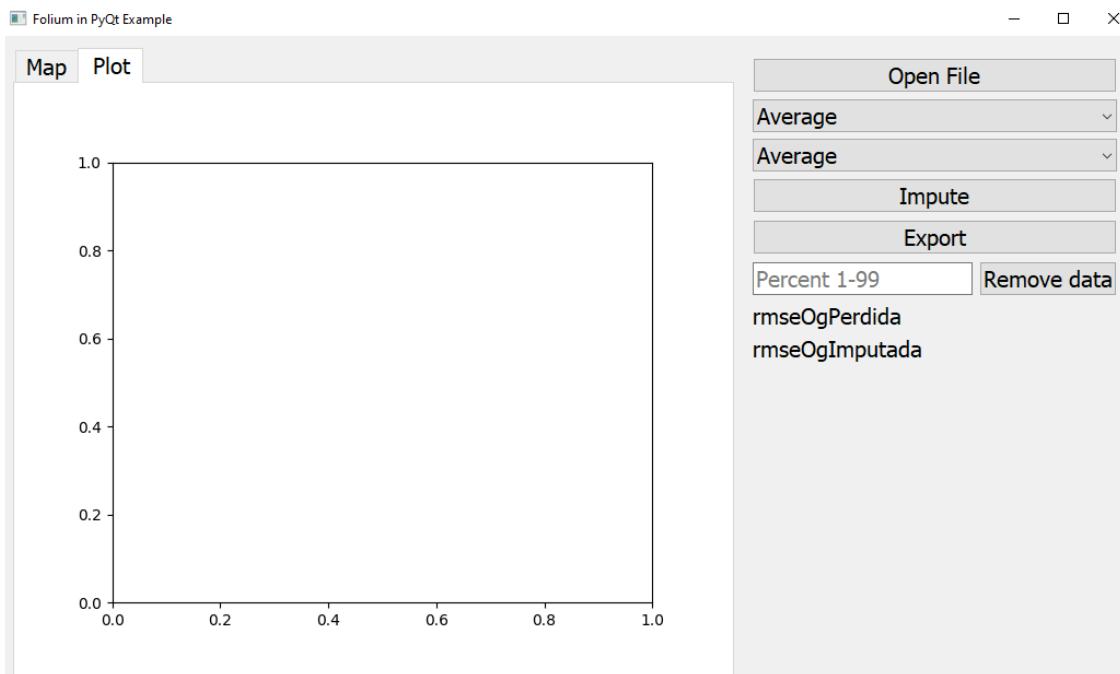


Figura 3.9: Incremento resultante del sprint 6

Tabla 3.15: Descripción del sprint 7

SPRINT 7

Fecha de comienzo	Jueves, 10 de abril de 2025		
Fecha de finalización	Jueves, 24 de abril de 2025		
Duración	14 días	Capacidad	10 días
Objetivo	Organizar y diseñar la interfaz de usuario		
<i>Pila de sprint</i>			
ID	Tareas	Estado	Estimación
T021	Corregir la visualización en gráfico de líneas	Completada	1 días
T022	Estructurar documentación de desarrollo	Completada	3 días
T023	Diseño de la interfaz la aplicación	Completada	3 días
T024	Crear diagrama de clases provisional	Completada	2 días
T025	Crear diagramas de secuencia de la aplicación	Completada	1 día

SPRINT 7		2025							
ID	Tarea	10-4	11-4	14-4	15-4	16-4	21-4	22-4	23-4
T021	Corregir la visualización en gráfico de líneas								
T022	Estructurar documentación de desarrollo								
T023	Diseño de la interfaz la aplicación								
T024	Crear diagrama de clases provisional								
T025	Crear diagramas de secuencia de la aplicación								

Figura 3.10: Distribución de tareas por días en el *sprint 7*

El objetivo de este *sprint* es estructurar la documentación y diseñar la interfaz. La documentación realizada hasta el momento es bastante breve y es la mínima necesaria que se sobreentiende se va generando naturalmente durante el desarrollo, por lo que una tarea fundamental se vuelve limpiar y regularizar el texto, y detallar toda la documentación. Además, se emprende una tarea de diseño que obtiene como resultado los *mock-ups* representados en las figuras 3.4, 3.5, 3.6 y 3.7. Una de las características principales es la división entre una zona de visualización y otra de control. Las sugerencias se realizan con la meta de poder visualizar, controlar y manipular varias secuencias desde la interfaz como se enfatiza en los requisitos RF-3.2 y R.F-3.4

La interfaz propuesta muestra un nuevo menú de control con dos zonas notables (Figura 3.11): el panel de los *Datasets*; el cual permite controlar las secuencias agregadas al área de trabajo, y el panel de imputación; que sirve para controlar la imputación de cada variable. El panel de *Datasets* está inspirado por las herramientas de edición en las cuales puedes decidir la visibilidad de cada capa o elemento, cada secuencia podrá ser mostrada u ocultada, mostrará retroalimentación respecto a su visibilidad y podrá ser manipulada desde un menú contextual que se despliega con un botón. Por otro lado, la zona de visualización está dividida en pestañas que permiten varias vistas, ya sea sobre un mapa o un gráfico de las secuencias, estas pestañas

recordarán que se estaba visualizando en cada una, pero las secuencias son comunes para todas. También se propone un diálogo para exportar (Figura 3.12) e importar (Figura 3.13) más complejo en vez de simplemente seleccionar un archivo, añadiendo la opción de importar y exportar varios archivos a la vez, fusionados o sueltos. Además, un diálogo para eliminar valores de la secuencia (Figura 3.14) se agrega para poder seleccionar método, en vez de tener la opción directamente sobre la interfaz.

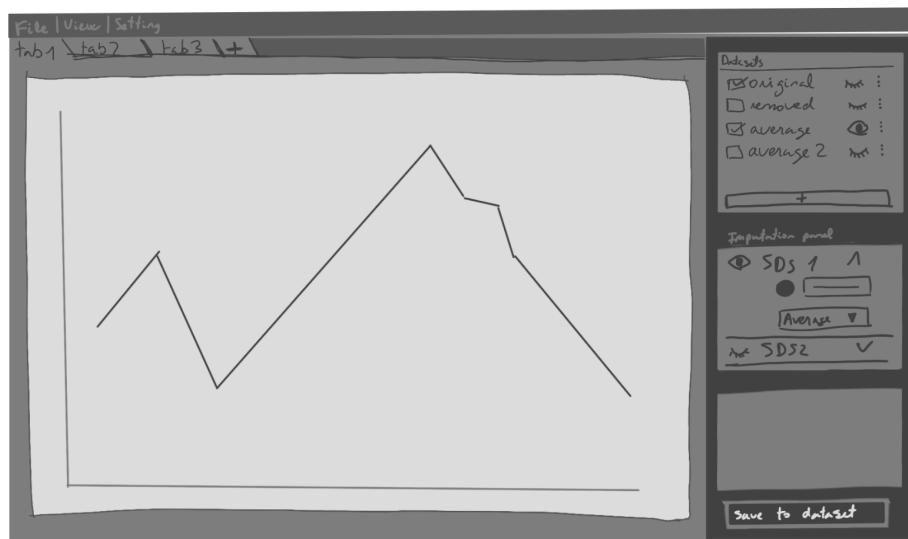


Figura 3.11: Boceto de interfaz de usuario, base

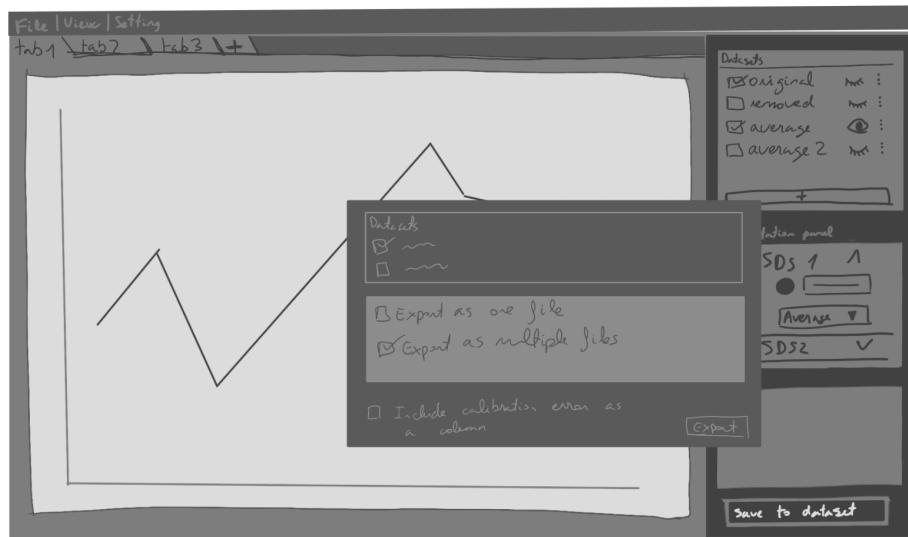


Figura 3.12: Boceto de interfaz de usuario, diálogo de exportación

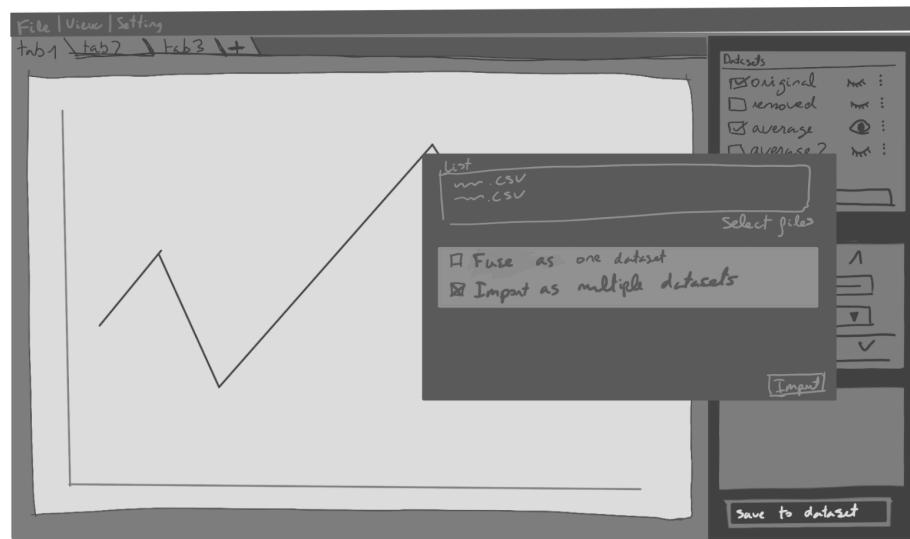


Figura 3.13: Boceto de interfaz de usuario, diálogo de importación

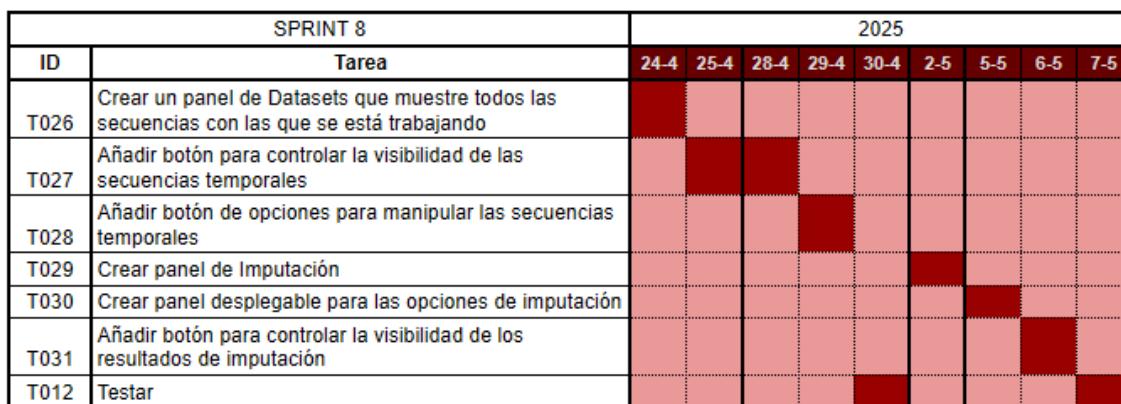


Figura 3.14: Boceto de interfaz de usuario, diálogo de eliminación de valores.

Tabla 3.16: Descripción del sprint 8

SPRINT 8			
Fecha de comienzo		Jueves, 24 de abril de 2025	
Fecha de finalización		Jueves, 8 de mayo de 2025	
Duración	14 días	Capacidad	9 días
Objetivo	Dividir la interfaz de usuario según utilidades		
Pila de <i>sprint</i>			
ID	Tareas	Estado	Estimación

T026	Crear un panel de Datasets que muestre todos las secuencias con las que se está trabajando	Completada	1 día
T027	Añadir botón para controlar la visibilidad de las secuencias temporales	Completada	2 días
T028	Añadir botón de opciones para manipular las secuencias temporales	Completada	1 día
T029	Crear panel de Imputación	Completada	1 día
T030	Crear panel desplegable para las opciones de imputación	Completada	1 día
T031	Añadir botón para controlar la visibilidad de los resultados de imputación	Completada	1 día
T012	Testar	Completada	2 días

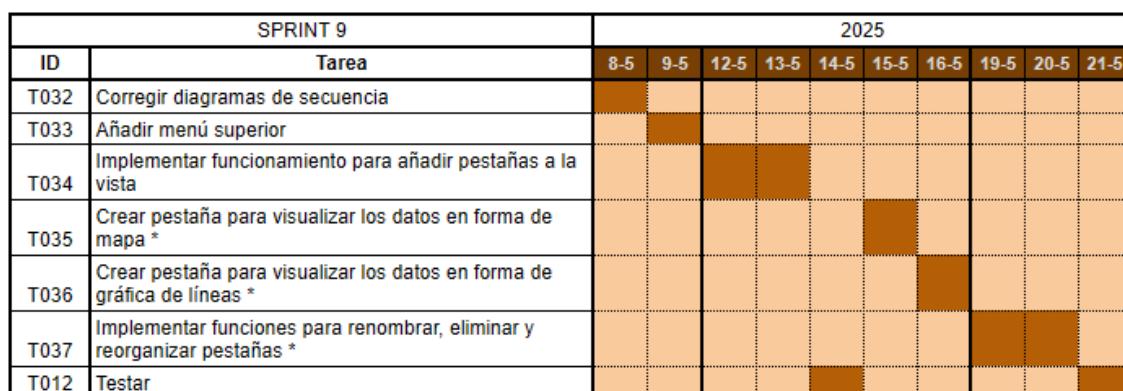
Figura 3.15: Distribución de tareas por días en el *sprint* 8

En la reunión de inicio del *sprint* se recibe retroalimentación referente a los indicativos visuales; qué secuencia está seleccionada pasa a diferenciarse por el color del fondo y el estado de visibilidad pasa a ser la caja. Tener ambos, el ojo y la caja, era incómodo. Durante este *sprint* se realiza la división de la interfaz en los paneles propuestos y se trabaja para agregar las ideas propuestas en el diseño. El panel de *datasets* es por dónde se controlan las secuencias temporales. Cada secuencia aparecerá como un elemento en la lista, y además de controlar la visibilidad, será la forma principal de interactuar con ellas. Cualquier modificación modifique la serie temporal, con excepción de cambiarle el nombre o eliminarla, creará una nueva secuencia que se agrega al panel; de forma que no se pierdan los datos originales (RF-1.3).

Tabla 3.17: Descripción del sprint 9

SPRINT 9			
Fecha de comienzo		Jueves, 8 de mayo de 2025	
Fecha de finalización		Jueves, 22 de mayo de 2025	
Duración	14 días	Capacidad	10 días

Objetivo	Personalizar la visualización		
Pila de <i>sprint</i>			
ID	Tareas	Estado	Estimación
T032	Corregir diagramas de secuencia	Completada	1 día
T033	Añadir menú superior	Completada	1 día
T034	Implementar funcionamiento para añadir pestañas a la vista	Completada	2 días
T035	Crear pestaña para visualizar los datos en forma de mapa	Abierta	1 día
T036	Crear pestaña para visualizar los datos en forma de gráfica de líneas	Abierta	1 día
T037	Implementar funciones para renombrar, eliminar y reorganizar pestañas	Abierta	2 días
T012	Testar	Completada	2 días

Figura 3.16: Distribución de tareas por días en el *sprint* 9

Este *sprint* se enfoca en la personalización del espacio de visualización (RF-3.2) mediante un sistema de pestañas renombrables y reorganizables que podrán contener tanto un mapa como una gráfica de líneas, por tanto se crea una interfaz base que contiene los métodos que ambos deben implementar de forma que la clase controlador pueda aplicar los cambios necesarios sin necesidad de saber qué caso es. Además se añade un menú superior con las opciones que permiten añadir estas pestañas a la vista.

Tabla 3.18: Descripción del sprint 10

SPRINT 10	
Fecha de comienzo	Jueves, 22 de mayo de 2025
Fecha de finalización	Jueves, 5 de junio de 2025

Duración	14 días	Capacidad	10 días
Objetivo	Corregir errores de visualización		
<i>Pila de sprint</i>			
ID	Tareas	Estado	Estimación
T035	Crear pestaña para visualizar los datos en forma de mapa	Completada	1 (2) días
T036	Crear pestaña para visualizar los datos en forma de gráfica de líneas	Completada	1 (2) días
T037	Implementar funciones para renombrar, eliminar y reorganizar pestañas	Completada	2 (4) días
T038	Añadir opción para guardar los resultados de la imputación como secuencia temporal	Completada	2 día
T039	Implementar opción para eliminar secuencia del área de trabajo	Completada	2 día
T012	Testar	Completada	2 días

SPRINT 10		2025									
ID	Tarea	22-5	23-5	26-5	27-5	28-5	29-5	30-5	2-6	3-6	4-6
T035	Crear pestaña para visualizar los datos en forma de mapa *	■									
T036	Crear pestaña para visualizar los datos en forma de gráfica de líneas *		■								
T037	Implementar funciones para renombrar, eliminar y reorganizar pestañas *			■	■						
T038	Añadir opción para guardar los resultados de la imputación como secuencia temporal						■	■			
T039	Implementar opción para eliminar secuencia del área de trabajo								■	■	
T012	Testar					■					■

Figura 3.17: Distribución de tareas por días en el *sprint* 10

La figura 3.18 muestra el resultado de este *sprint* permitiendo apreciar la nueva distribución de funciones por la interfaz, se distinguen las pestañas de visualización, el menú superior y los paneles de *dataset*, con una serie temporal con un botón de visibilidad y otro para abrir ajustes, e imputación, teniendo un botón de visibilidad de resultados y un menú desplegable para cada variable: SDS_P1, correspondiente a la medida PM2.5 y SDS_P2, a PM10; que permite cambiar el método de imputación para cada una. El calendario que se mostraba en la anterior versión desaparece, puesto que se idea que acotar un intervalo de tiempo puede ser implementado como método para eliminar valores, opción posible mediante el botón de opciones de cada serie temporal y que crea una nueva serie en vez de modificar la original.

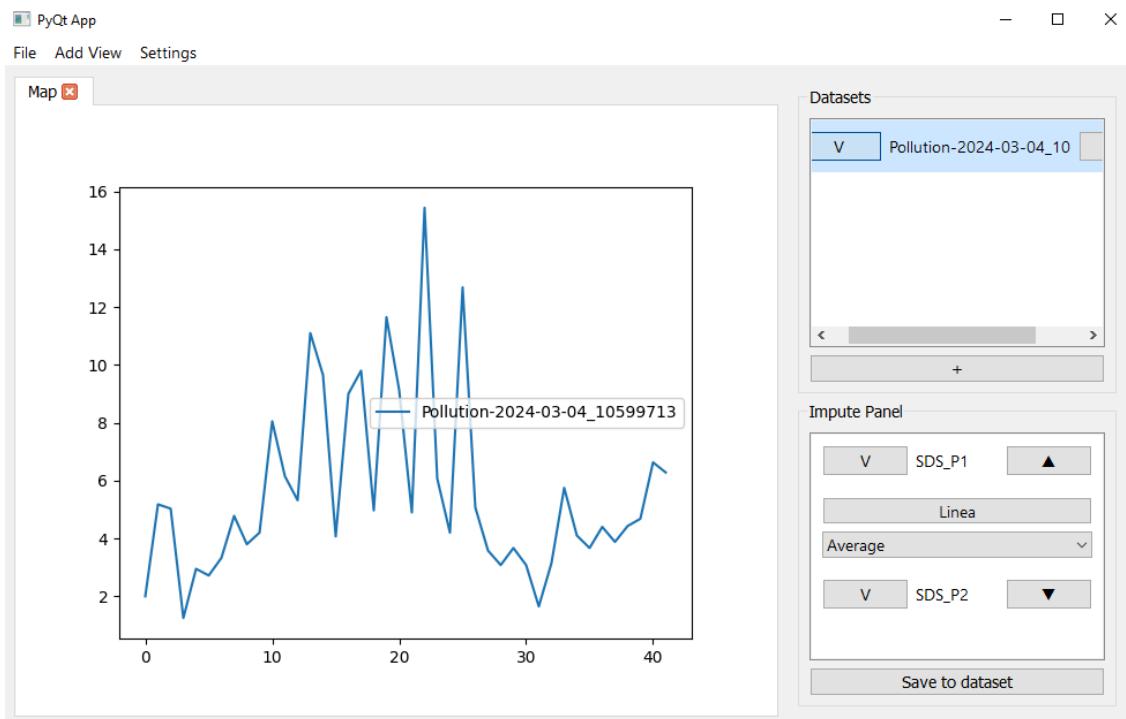


Figura 3.18: Captura del incremento resultante de tras el sprint 10

Tabla 3.19: Descripción del sprint 11

SPRINT 11			
Fecha de comienzo		Jueves, 5 de junio de 2025	
Fecha de finalización		Jueves, 12 de junio de 2025	
Duración	7 días	Capacidad	5 días
Objetivo	Agregar herramienta de comparación		
Pila de sprint			
ID	Tareas	Estado	Estimación
T040	Crear pestaña para mostrar la comparativa entre dos secuencias utilizando el cálculo de la raíz del error cuadrático medio (RECM)	Completada	2 días
T041	Investigación para implementar métodos de aprendizaje automática	Abierta	3 días

SPRINT 11			2025				
ID	Tarea	5-6	6-6	9-6	10-6	11-6	
T040	Crear pestaña para mostrar la comparativa entre dos secuencias utilizando el cálculo de la raíz del error cuadrático medio (RECM)						
T041	Investigación para implementar métodos de aprendizaje automática *						

Figura 3.19: Distribución de tareas por días en el *sprint* 11

Durante la reunión de planificación se dialoga el interés por añadir métodos de imputación basados en aprendizaje automático que queda reflejado en una nueva historia de usuario.

Tabla 3.20: Historia de usuario 7, usar modelos de aprendizaje automático

HU-7	Usar modelos de aprendizaje automático	
Historia de usuario	Como miembro del equipo de investigación quiero poder utilizar y entrenar modelos de aprendizaje automático para realizar la imputación de valores faltantes.	
Criterios de aceptación	1. Entre los métodos de imputación seleccionables incluidos en la aplicación deben encontrarse métodos de aprendizaje automático SAITS y Transformers. 2. Es posible entrenar modelos y guardar el resultado del entrenamiento en memoria permanente.	
Descripción		
Prioridad	Media	

A los dos tipos de pestañas, las de mapa y las de gráfica, se añade una nueva, la de análisis. En esta se incluye el cálculo RECM implementado anteriormente y se deja espacio para otras posibles variables que se puedan usar para la comparación en caso de extensión de la aplicación.

Tabla 3.21: Descripción del sprint 12

SPRINT 12			
Fecha de comienzo		Jueves, 12 de junio de 2025	
Fecha de finalización		Jueves, 26 de junio de 2025	
Duración	14 días	Capacidad	9 días
Objetivo	Mejorar importación y exportación		
Pila de <i>sprint</i>			
ID	Tareas	Estado	Estimación
T042	Implementar diálogo de importación de múltiples archivos	Completada	2 días
T043	Implementar diálogo de exportación de secuencias temporales un archivo o múltiples	Completada	3 días

T041	Investigación para implementar métodos de aprendizaje automática	Completada	2 (5) días
T012	Testar	Completada	2 días

SPRINT 12		2025								
ID	Tarea	12-6	13-6	16-6	17-6	18-6	20-6	23-6	24-6	25-6
T041	Investigación para implementar métodos de aprendizaje automática *									
T042	Implementar diálogo de importación de múltiples archivos									
T043	Implementar diálogo de exportación de secuencias temporales un archivo o múltiples									
T012	Testar									

Figura 3.20: Distribución de tareas por días en el *sprint 12*

Cómo se aprecia en la tabla 3.21 las principales tareas son las de mejorar la importación y exportación con diálogos que permitan seleccionar el comportamiento en caso de seleccionar varios archivos o series temporales en la aplicación, dando dos opciones: fusión en un sólo elemento o creación de múltiples. Asimismo, el tiempo sobrante se emplea en investigar métodos de aprendizaje automático cómo se describe en la historia de usuario 7, sin embargo, este resulta insuficiente.

Tabla 3.22: Descripción del sprint 13

SPRINT 13			
Fecha de comienzo		Jueves, 26 de junio de 2025	
Fecha de finalización		Miércoles, 9 de julio de 2025	
Duración	13 días	Capacidad	9 días
Objetivo	Expandir capacidades de gestión de las secuencias temporales del área de trabajo		
Pila de <i>sprint</i>			
ID	Tareas	Estado	Estimación
T044	Implementar opción para eliminar secuencias temporales del área de trabajo	Completada	1 día
T045	Implementar opción para duplicar secuencias temporales del área de trabajo	Completada	1 día
T046	Implementar opción para renombrar secuencias temporales del área de trabajo	Completada	1 día
T047	Implementar función para la eliminación de valores dentro o fuera de un intervalo concreto	Completada	1 día

T048	Corregir error de cálculo de la pantalla de comparación	Abierta	1 día
T049	Implementar memoria de visibilidad a las pestañas	Completada	2 día
T012	Testar	Completada	2 días

Figura 3.21: Distribución de tareas por días en el *sprint* 13

Las tareas de este *sprint* se centran en la gestión de secuencias temporales, permitiendo duplicarlas, eliminarlas o renombrarlas, lo cual es especialmente delicado puesto que su nombre o etiqueta es lo que las diferencia en la visualización. Un caso interesante es la tarea T047 implementa un nuevo método para eliminar valores de una secuencia temporal: eliminar aquellos que se encuentran fuera o contenidos por un intervalo de tiempo. Este método sustituye el calendario que se mostraban en el incremento del *sprint* 4, puesto, como se explicaba anteriormente, esta funcionalidad de mostrar sólo algunos valores en función a un intervalo de tiempo se convierte en una función para eliminar valores; y, al igual que otras funciones que modifican la secuencia, genera una nueva serie temporal que se agrega al panel de secuencias de forma que se puede observar la original y la resultante.

Tabla 3.23: Descripción del sprint 13

SPRINT 14			
Fecha de comienzo	Miércoles, 9 de julio de 2025		
Fecha de finalización	Jueves, 24 de julio de 2025		
Duración	15 días	Capacidad	11 días
Objetivo	Implementar método de aprendizaje automático		
Pila de <i>sprint</i>			
ID	Tareas	Estado	Estimación
T048	Corregir error de cálculo de la pantalla de comparación	Completada	2 (3) días

T050	Implementar PyPots SAITS como método de imputación	Completada	6 días
T012	Testar	Completada	3 días

SPRINT 14		2025										
ID	Tarea	9-7	10-7	11-7	14-7	15-7	16-7	17-7	18-7	21-7	22-7	23-7
T048	Corregir error de cálculo de la pantalla de comparación *											
T050	Implementar PyPots SAITS como método de imputación											
T012	Testar											

Figura 3.22: Distribución de tareas por días en el *sprint 14*

Al comienzo de este *sprint 14* se declara la necesidad de crear material para enseñar el uso del programa (RNF-2) y se hace hincapié en que debe ser fácilmente extensible, especialmente para agregar nuevos algoritmos de imputación que ya se tenía en cuenta eligiendo el patrón estrategia para la implementación de los métodos de imputación. Además, se implementa el primer método de aprendizaje automático, aplicando el modelo SAITS de la biblioteca PyPots.

Tabla 3.24: Descripción del sprint 15

SPRINT 15			
Fecha de comienzo	Jueves, 24 de julio de 2025		
Fecha de finalización	Jueves, 31 de julio de 2025		
Duración	7 días	Capacidad	5 días
Objetivo	Implementar segundo método de aprendizaje automático		
Pila de <i>sprint</i>			
ID	Tareas	Estado	Estimación
T051	Implementar PyPots Transformers como método de imputación	Abierta	3 días
T012	Testar	Completada	2 días

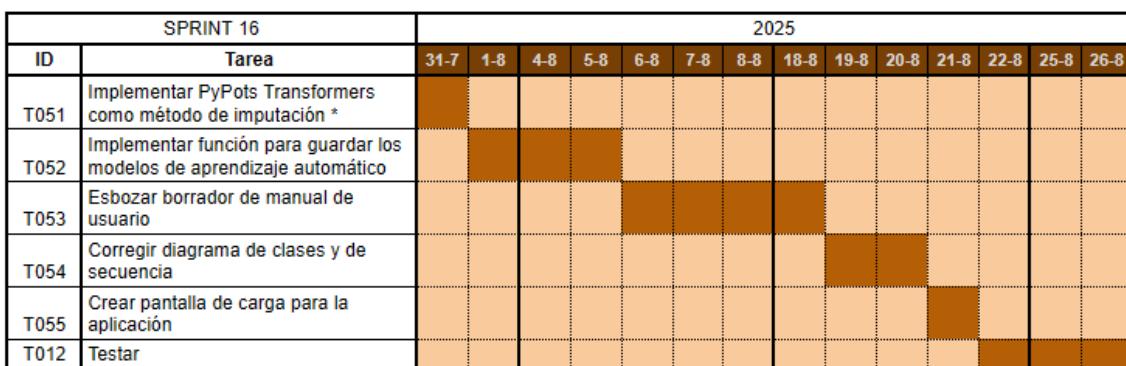
SPRINT 15		2025				
ID	Tarea	24-7	25-7	28-7	29-7	30-7
T051	Implementar PyPots Transformers como método de imputación *					
T012	Testar					

Figura 3.23: Distribución de tareas por días en el *sprint 15*

El *sprint* 15 es especialmente corto con razón de tener una última reunión antes de agosto, mes en el que se toman vacaciones y sería imposible coincidir, por ello sólo se comienza la implementación del segundo método de aprendizaje automático, Transformers de la biblioteca PyPots, y se hacen pruebas del incremento actual.

Tabla 3.25: Descripción del sprint 16

SPRINT 16			
Fecha de comienzo	Jueves, 31 de julio de 2025		
Fecha de finalización	Lunes, 1 de septiembre de 2025		
Duración	32 días	Capacidad	14 días
Objetivo	Preparar el cierre del desarrollo		
Pila de <i>sprint</i>			
ID	Tareas	Estado	Estimación
T051	Implementar PyPots Transformers como método de imputación	Completada	1 (4) días
T052	Implementar función para guardar los modelos de aprendizaje automático	Completada	3 días
T053	Esbozar borrador de manual de usuario	Completada	4 días
T054	Corregir diagrama de clases y de secuencia	Completada	2 días
T055	Crear pantalla de carga para la aplicación	Completada	1 día
T012	Testar	Completada	3 días

Figura 3.24: Distribución de tareas por días en el *sprint* 16

Aunque este *sprint* 16 ocupa la mayoría de agosto, la capacidad real es mucho menor por los días que se toman libres. Este es el penúltimo sprint y el último con tareas de desarrollo, que se centran en completar la agregación de los métodos de aprendizaje automático, cambios estéticos y probar las últimas funciones generadas. Debido al tiempo que añaden las bibliotecas de aprendizaje automático al lanzamiento del programa, se decide crear una pantalla de carga

animada que aparecerá de forma inmediata y desaparecerá cuando la aplicación esté lista. El incremento resultante ya se puede considerar la herramienta software objetivo (Figura 3.25) con todas las funcionalidades deseadas que contiene: una zona de visualización personalizable mediante pestañas, un panel de series temporales que permite controlar su visibilidad y manipularlas, y un panel de imputación que permite mostrar u ocultar los resultados y seleccionar el método a utilizar para cada variable de interés. Además, se corrige el diagrama de clases de forma que sea más adecuado a la implementación real final y se esboza un manual de usuario.

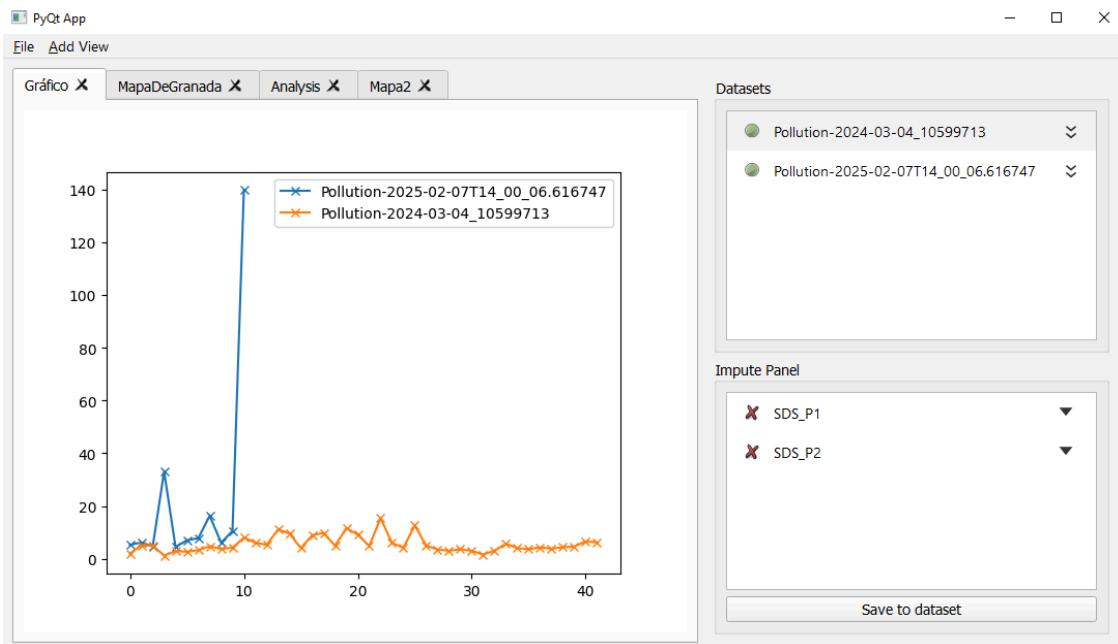


Figura 3.25: Captura del incremento resultante de tras el sprint 16

Tabla 3.26: Descripción del sprint 17

SPRINT 17			
Fecha de comienzo		Lunes, 1 de septiembre de 2025	
Fecha de finalización		Martes, 30 de septiembre de 2025	
Duración	29 días	Capacidad	21 días
Objetivo	Cerrar el desarrollo		
Pila de <i>sprint</i>			
ID	Tareas	Estado	Estimación
T056	Redactar y limpiar la documentación final	Completada	11 días
T057	Redactar manual de usuario final	Completada	6 días
T058	Preparar presentación	Completada	3 días
T059	Grabar demo	Completada	1 día

SPRINT 17		2025																						
ID	Tarea	1-9	2-9	3-9	4-9	5-9	6-9	7-9	8-9	9-9	10-9	11-9	12-9	15-9	16-9	17-9	18-9	19-9	22-9	23-9	24-9	25-9	26-9	29-9
T056	Redactar y limpiar la documentación final																							
T057	Redactar manual de usuario final																							
T058	Preparar presentación																							
T059	Grabar demo																							

Figura 3.26: Distribución de tareas por días en el *sprint* 17

Este último *sprint* cierra el desarrollo del trabajo y busca recopilar toda la información generada, por ejemplo, la organización temporal de las tareas se organiza en un diagrama de Gantt completo como se observa en la siguiente sección o se redacta los requisitos completos que se han ido discutiendo, además se redactan las conclusiones y el manual final de usuario. Tanto en la reunión inicial como en la final se hace una retrospectiva sobre el desarrollo, teniendo en cuenta los requisitos y objetivos.

3.4.2 Diagrama de Gantt

En total el proyecto ha sido constituido por 59 tareas agrupadas según qué paquete de trabajo cumplen y estimando la cantidad de tiempo dedicado a cada paquete (Tabla 3.27) en 205 días entre los que se reparten 300 horas de trabajo. En esta sección se muestra cada paquete titulado en negrita y acompañado por todas las tareas que lo conforman, además de un Diagrama de Gantt completo (Figuras 3.27 y 3.28) de todas las tareas y dividido en secciones para una mejor interpretación. A lo largo de la descripción de los *sprints* de la sección anterior y en el diagrama completo de esta se puede observar cierta desviación respecto a la planificación original de cada *sprint* que queda reflejado en las tareas que quedan abiertas y se deben de retomar en el siguiente. Como apunte especial, en el caso del paquete de testeo, sólo se encuentra una tarea referente a las pruebas que se va repitiendo cuando se necesita en diferentes *sprints*, esto es debido a que *scrum*, como otras metodologías iterativas, hace hincapié en generar incrementos de software usable y por tanto lo ideal es probar cada uno de ellos.

Investigación del ámbito del problema

- T002 Investigación posibles herramientas de desarrollo
- T003 Estudiar las bases de la imputación y metodologías
- T041 Investigación para implementar métodos de aprendizaje automática

Formación para el desarrollo del software

- T004 Completar formación sobre Python
- T006 Realizar formación PyQt

Diseño y prototipado

- T005 Creación de un prototipo de interfaz con mapa
- T007 Diseño básico de las clases
- T023 Diseño de la interfaz la aplicación
- T024 Crear diagrama de clases provisional
- T025 Crear diagramas de secuencia de la aplicación
- T032 Corregir diagramas de secuencia
- T054 Corregir diagrama de clases y de secuencia

Desarrollo e implementación

- T008 Añadir función que permita la importación de secuencias desde archivos csv
T009 Añadir función que permita la exportación de secuencias desde archivos csv
T010 Visualizar datos de los puntos de las secuencias sobre el mapa
T011 Implementar función para la eliminación de valores aleatorios de la secuencia
T013 Crear división para la visualización en forma de gráfica
T014 Implementar visualización en gráfico de líneas
T015 Implementar imputación de secuencia con datos faltantes utilizando la media
T016 Añadir comparativa de utilizando el cálculo de la raíz del error cuadrático medio (RECM)
T017 Implementar el método de imputación de relleno hacia atrás
T018 Implementar el método de imputación de relleno hacia delante
T019 Implementar el método de imputación usando la mediana
T020 Incluir selector de método de imputación para PM2.5 y PM10
T021 Corregir la visualización en gráfico de líneas
T026 Crear un panel de Datasets que muestre todos las secuencias con las que se está trabajando
T027 Añadir botón para controlar la visibilidad de las secuencias temporales
T028 Añadir botón de opciones para manipular las secuencias temporales
T029 Crear panel de Imputación
T030 Crear panel desplegable para las opciones de imputación
T031 Añadir botón para controlar la visibilidad de los resultados de imputación
T033 Añadir menú superior
T034 Implementar funcionamiento para añadir pestañas a la vista
T035 Crear pestaña para visualizar los datos en forma de mapa
T036 Crear pestaña para visualizar los datos en forma de gráfica de líneas
T037 Implementar funciones para renombrar, eliminar y reorganizar pestañas
T038 Añadir opción para guardar los resultados de la imputación como secuencia temporal
T039 Implementar opción para eliminar secuencia del área de trabajo
T040 Crear pestaña para mostrar la comparativa entre dos secuencias utilizando el cálculo de la raíz del error cuadrático medio (RECM)
T042 Implementar diálogo de importación de múltiples archivos
T043 Implementar diálogo de exportación de secuencias temporales un archivo o múltiples
T044 Implementar opción para eliminar secuencias temporales del área de trabajo
T045 Implementar opción para duplicar secuencias temporales del área de trabajo
T046 Implementar opción para renombrar secuencias temporales del área de trabajo
T047 Implementar función para la eliminación de valores dentro o fuera de un intervalo concreto
T048 Corregir error de cálculo de la pantalla de comparación
T049 Implementar memoria de visibilidad a las pestañas
T050 Implementar PyPots SAITS como método de imputación
T051 Implementar PyPots Transformers como método de imputación
T052 Implementar función para guardar los modelos de aprendizaje automático
T055 Crear pantalla de carga para la aplicación

Testar

- T012 Testar
Esta tarea se repite cada vez que es necesario testar funcionalidades.

Redactar memoria y documentación

- T001 Definición de requisitos provisionales
T022 Estructurar documentación de desarrollo
T053 Esbozar borrador de manual de usuario
T056 Redactar y limpiar la documentación final

- T057 Redactar manual de usuario final
- T058 Preparar presentación
- T059 Grabar demo

Tabla 3.27: División de tiempo por paquete de trabajo

Paquetes de trabajo	Tiempo invertido, en días	Tiempo invertido, en porcentaje	Tiempo invertido, en horas
Investigación del ámbito del problema	19	9,27%	27,8
Formación de desarrollo del software	17	8,29%	24,88
Diseño y prototipado	18	8,78%	26,34
Desarrollo e implementación	92	44,88%	134,63
Testar	28	13,66%	40,98
Redactar memoria y documentación	31	15,12%	45,37

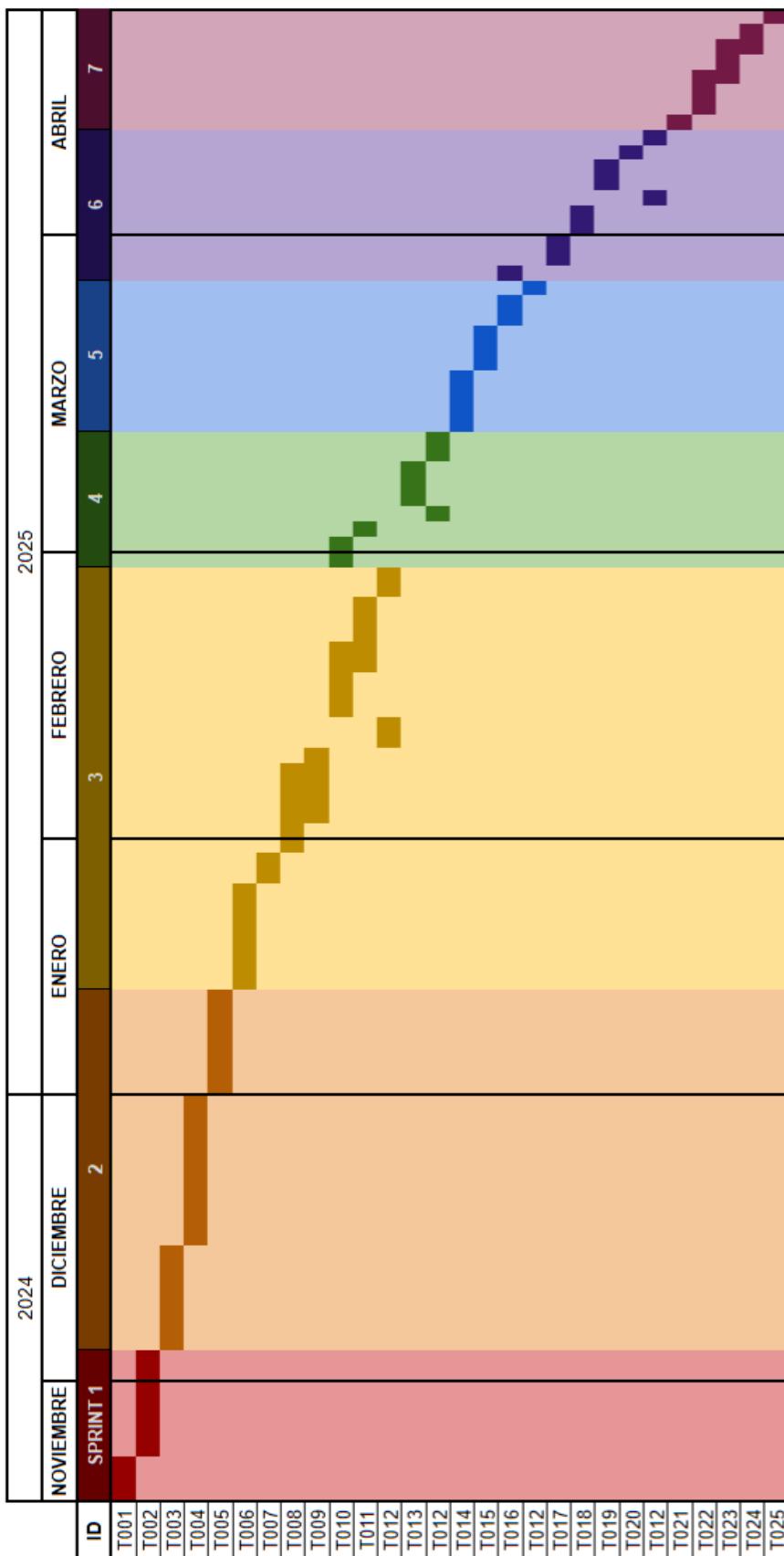


Figura 3.27: Diagrama de Gantt completo, desde el sprint 1 al 7

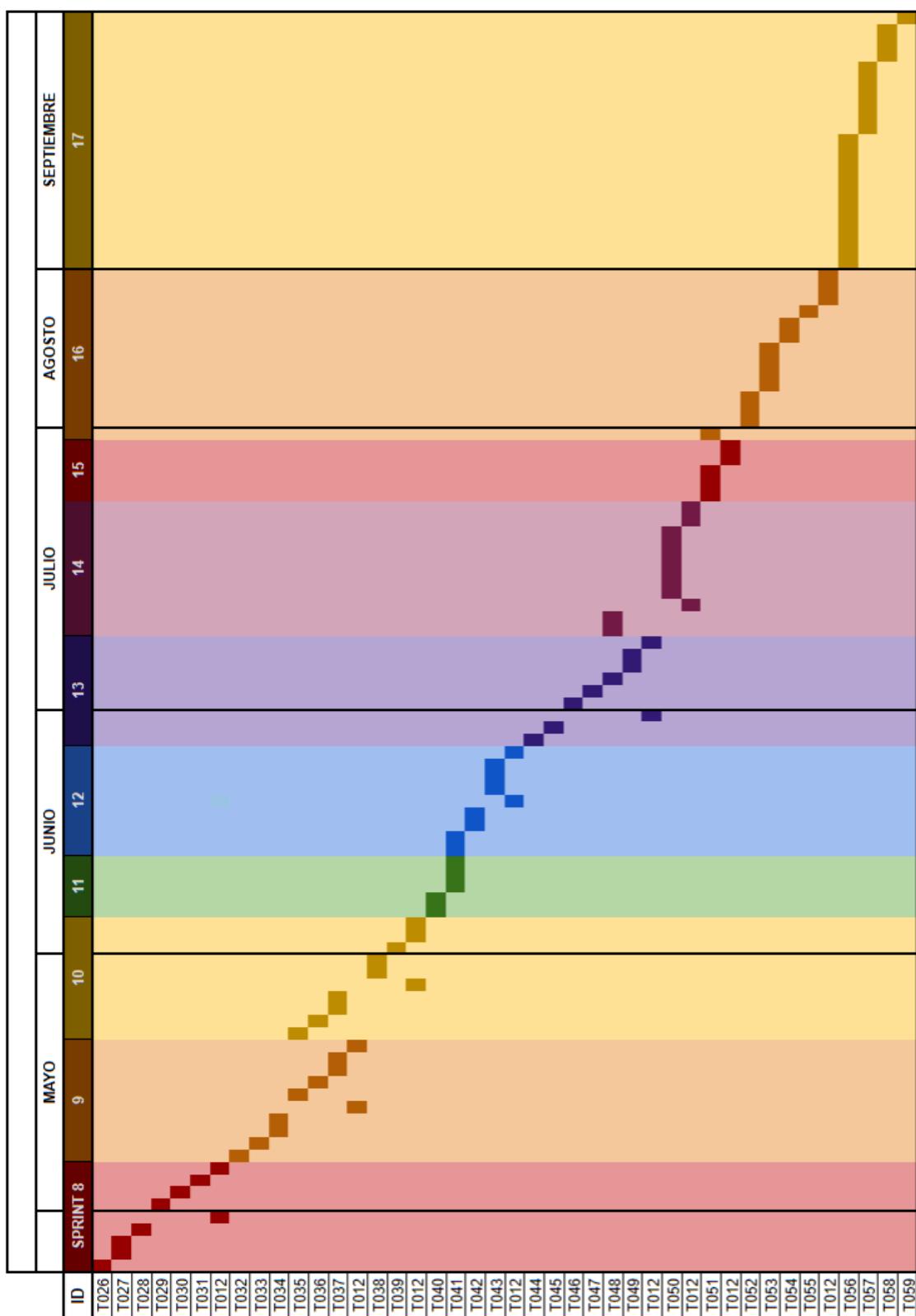


Figura 3.28: Diagrama de Gantt completo, desde el sprint 8 al 17

3.5 Diseño

3.5.1 Diagrama de clases

Para la implementación de la aplicación se sigue la arquitectura de Modelo-Vista-Controlador, dónde la idea es dividir la representación interna de la información de cómo ésta se muestra al usuario. En el MVC, la Vista es lo que observa el usuario y con lo que puede interactuar, el Modelo contiene la información y la lógica para manipularla y el Controlador es quien facilita la comunicación entre la Vista y el Controlador y es la autoridad de la aplicación (Syromiatnikov & Weyns, 2014). Esta división es especialmente popular en diseño de aplicaciones web, por su compatibilidad con la lógica Cliente/Servidor, pero también es de gran utilidad todo tipo de software interactivo al desacoplar la interfaz de usuario y la información. Utilizando PyQt5 las interfaces se construyen mediante widgets, elementos gráficos reutilizables para visualizar e interaccionar. De forma nativa se puede utilizar un patrón Modelo-Vista, puesto que las clases que forman la vista pueden ser emparejadas con otras que funcionan como modelo, sin embargo, para información más compleja se debe implementar modelos y controladores propios.

A continuación, en esta sección, se muestra y describe el diagrama de clases de la aplicación, primero de forma completa (Figura 3.29) sin incluir los métodos y atributos para poder distinguir las relaciones y luego seccionado en partes para realizar una visualización más clara y detallada (Figura 3.30-3.33). Este diagrama ha ido variando a lo largo de los sprints y el mostrado en las figuras es el resultado final. Se puede apreciar que la implementación de los distintos métodos de imputación se realiza utilizando el patrón estrategia (Figura 3.30), de forma que se puede seleccionar el método a usar durante la ejecución del programa. Asimismo, se utiliza el patrón *publish-subscribe* mediante señales para mantener cierto desacople en la comunicación entre las partes de forma que un receptor esté escuchando los cambios producidos en un emisor sin que este necesite conocer quién está escuchando. Estas señales son implementadas utilizando la clase `pyqtSignal` de PyQt y para poder suscribirse a ellas es necesario que sean públicas para que las demás clases conozcan de su existencia y realizar un *connect* sobre ellas. El problema es que esto hace posible que se pueda realizar una publicación o *emit* desde otra clase para esto existen varias soluciones pero antes es necesario explicar la visibilidad en Python.

Python no impone la limitación de acceso de una variable o método privado de la forma que lo hacen otros lenguajes, si no que utiliza una convención de nomenclatura basada en añadir un guion bajo delante del nombre de la variable para indicar que no pertenece a la API pública de la clase y, que al ser de uso interno, puede cambiar su implementación en cualquier momento (Python Software Foundation, 2001). Esto sigue la filosofía general de Python de confiar en los demás programadores y que ellos harán uso de las clases creadas por los demás bajo su propio criterio y responsabilidad, que es la causante de que no se suelen utilizar *setters* y *getters* dedicados cuando una clase tiene atributos simples. Sin embargo, hay una alternativa para crear variables privadas de verdad, utilizar dos guiones bajos delante del nombre de la variable lo que hace uso de el llamado *name mangling* y que lo que realmente está haciendo es añadir el nombre de la clase y otros dos guiones bajos a la variable, de forma que no se puede acceder al nombre original, esto reduce la posibilidad de que se use la variable, pero no lo imposibilita (Python Software Foundation, 2001). Aún así, esto permite cosas como utilizar el decorador “`@property`” para generar *getters* y *setters* a decisión del desarrollador, como se hace en el

siguiente fragmento de código de TabInfo para sólo permitir obtener el valor de `__type` pero no poder modificarlo.

```
Python
def __init__(self, type, title = "", visible = []
             , selection = [], results={}, maxSelection = 1):
    ...
    ...
    self.__type = type
    self.__maxSelection = maxSelection

@property
def type(self):
    return self.__type
```

Sin embargo, esto no detendría el uso de `emit`, puesto que se sigue devolviendo la variable. Entonces, las soluciones posibles son: dejar las señales como públicas, confiando en que se usarán con buen criterio y responsabilidad cómo se asume en la programación en Python; marcar la variable con un guión bajo, privada pero no oculta; o crear una clase que solo permita `connect` para envolver a la señal real, que estaría oculta con dos guiones bajos para uso interno, y utilizar `@property` para devolver la correcta. La forma más simple y que se alinea con el pensamiento de Python es dejar públicas las señales.

Otra información importante es que las clases que empiezan por la letra Q como QWidget, QObject, etc. no son de implementación propia, sino que pertenecen a las bibliotecas de PyQt. Los elementos en una interfaz gráfica de PyQt suelen estar formados por agregación, la mayoría de clases heredan de QWidget y pueden contener otros elementos QWidget, los cuales podrían existir independientes de los padres, como es en el caso de QTabWidget quién posee un conjunto de QWidgets como pestañas y que pueden existir sin necesidad de ser contenidos en QTabWidget. Por otro lado, también se encuentran relaciones de composición, como aquella entre QListWidget y QListWidgetItem, puesto que QListWidget está formado por un conjunto de QListWidgetItems, cuya existencia depende de la de QListWidget y no es significativa sin pertenecer a este. Las clases propias del diseño para la implementación de la interfaz siguen esta lógica y heredan de alguna de PyQt, como por ejemplo DatasetListItem hereda de QWidget y por tanto puede ser contenido por un QListWidgetItem que es un elemento de de QListWidget. Asimismo, existe una relación de dependencia unidireccional entre el controlador y el modelo y la vista, puesto que el controlador conoce y está escuchando las señales que emiten los otros dos y llama a sus funciones cuando es necesario; mientras que modelo y vista desconocen del otro y del controlador.

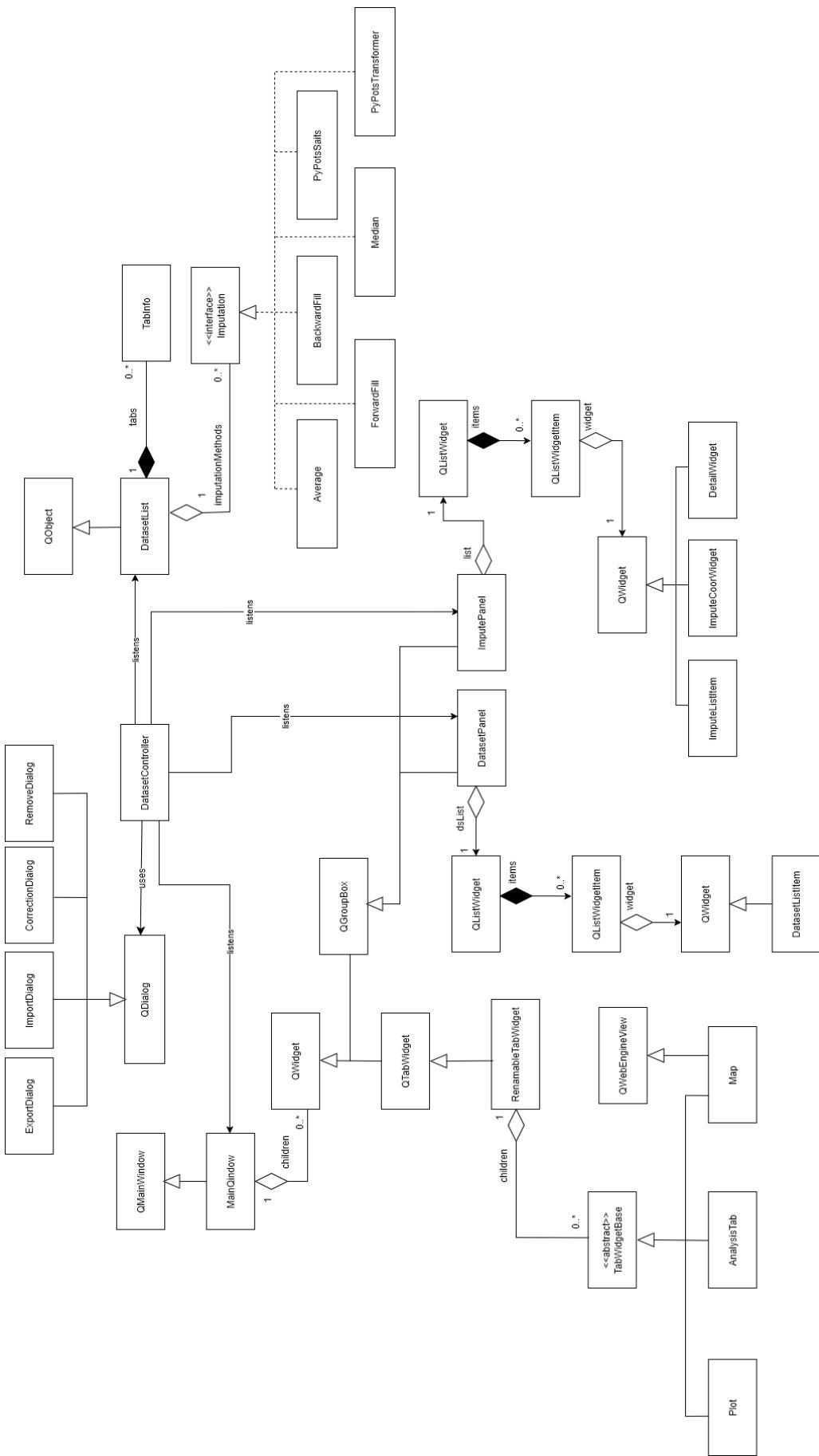


Figura 3.29: Diagrama de clases completo

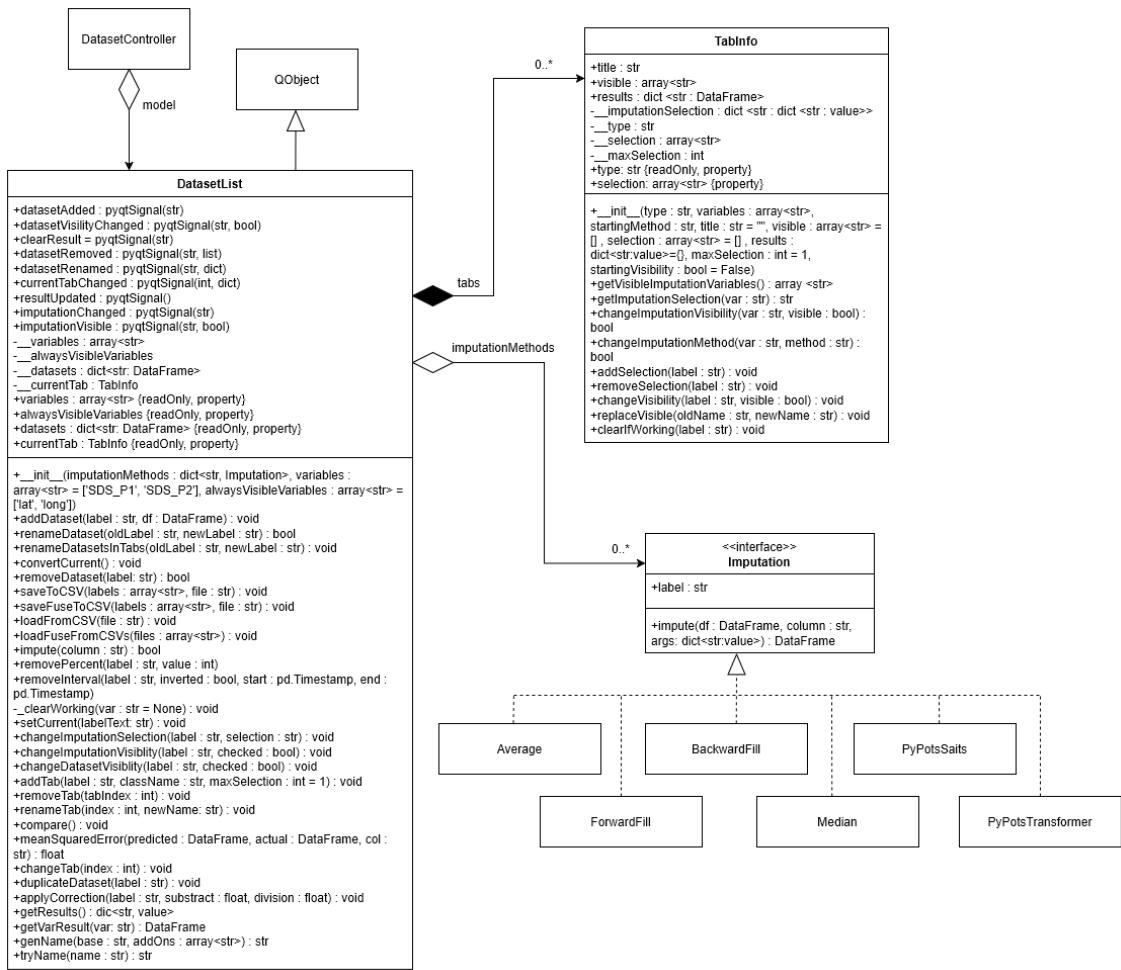


Figura 3.30: Sección del modelo del diagrama de clases

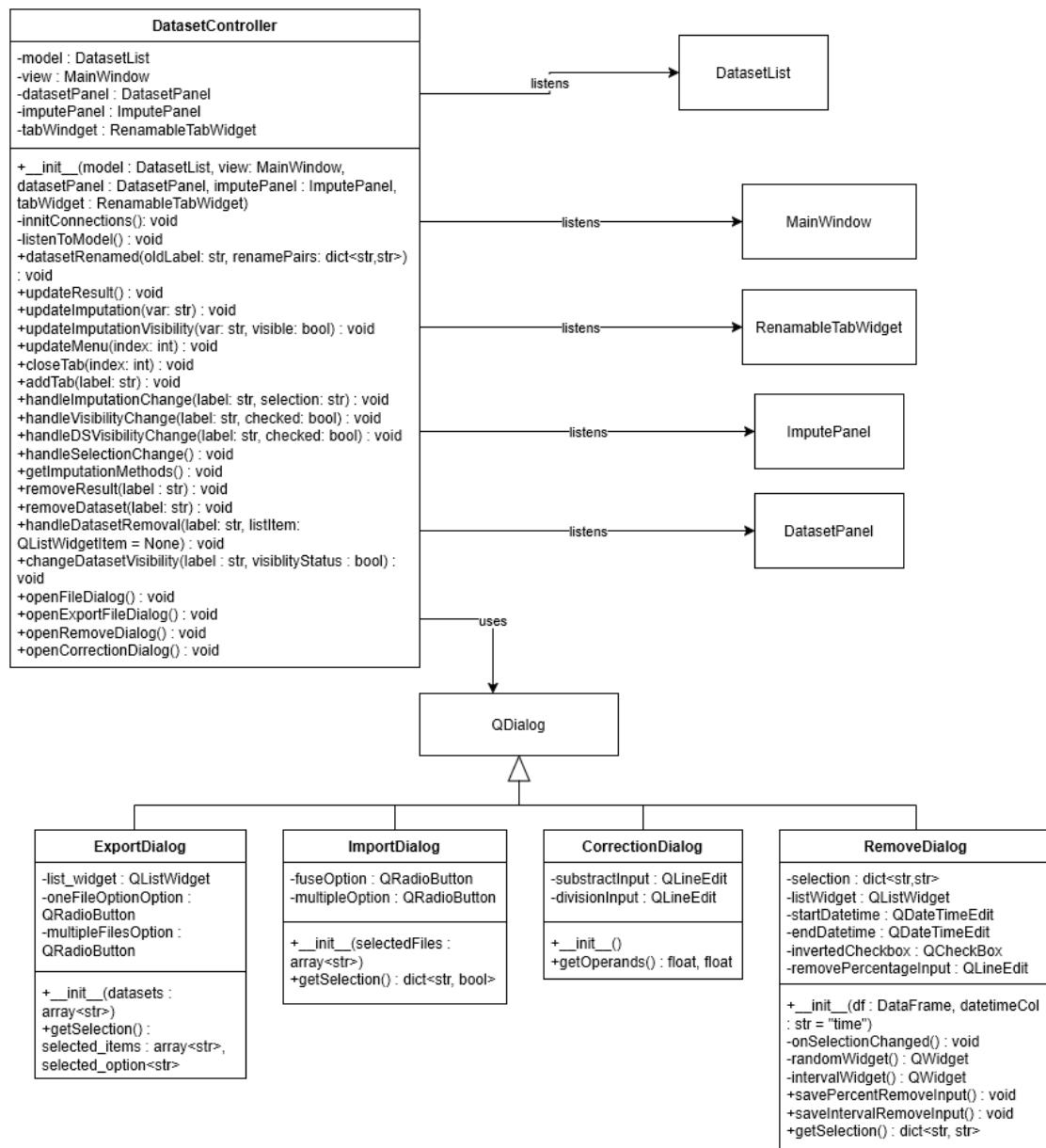


Figura 3.31: Sección del controlador del diagrama de clases

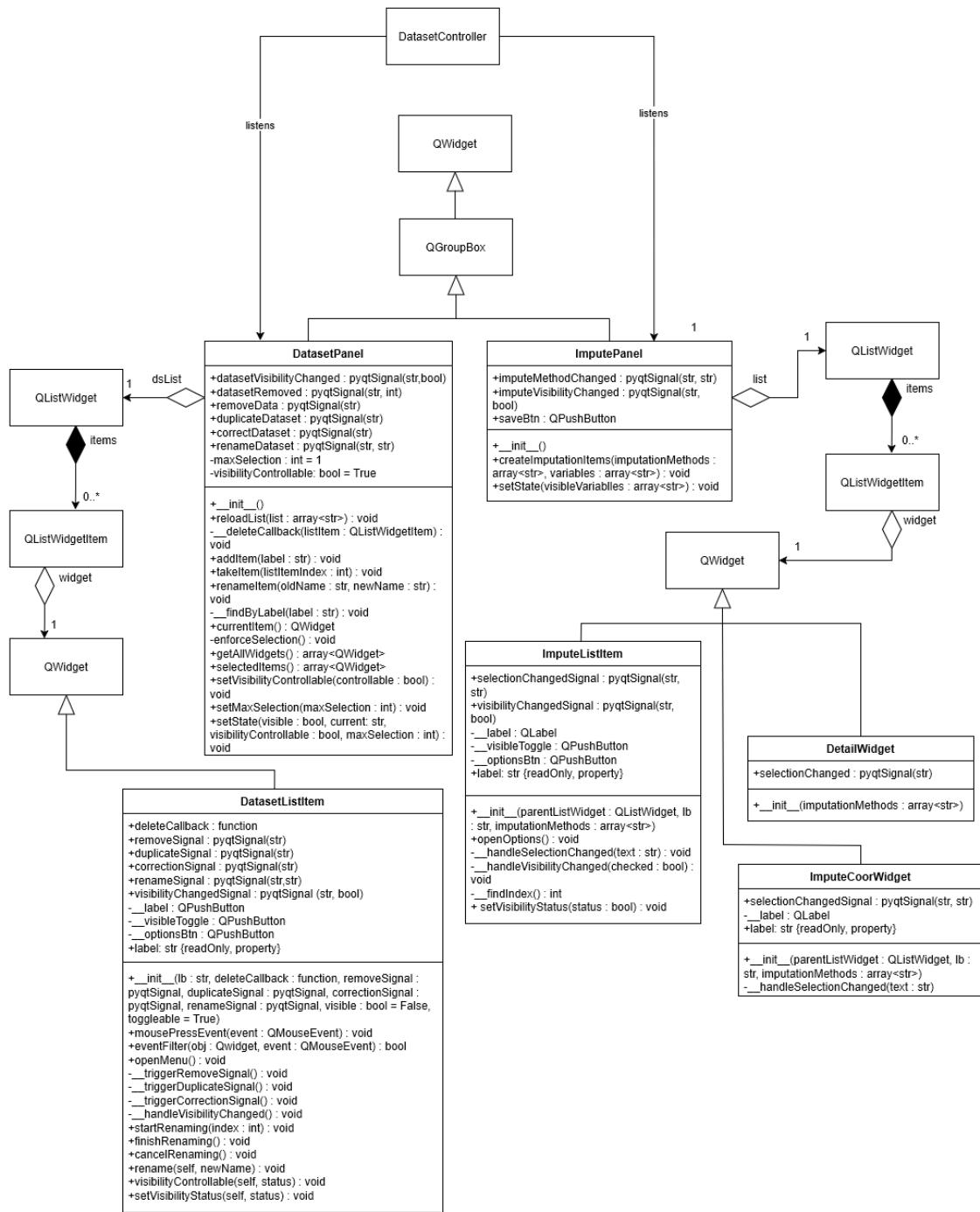


Figura 3.32: Primera sección de la vista

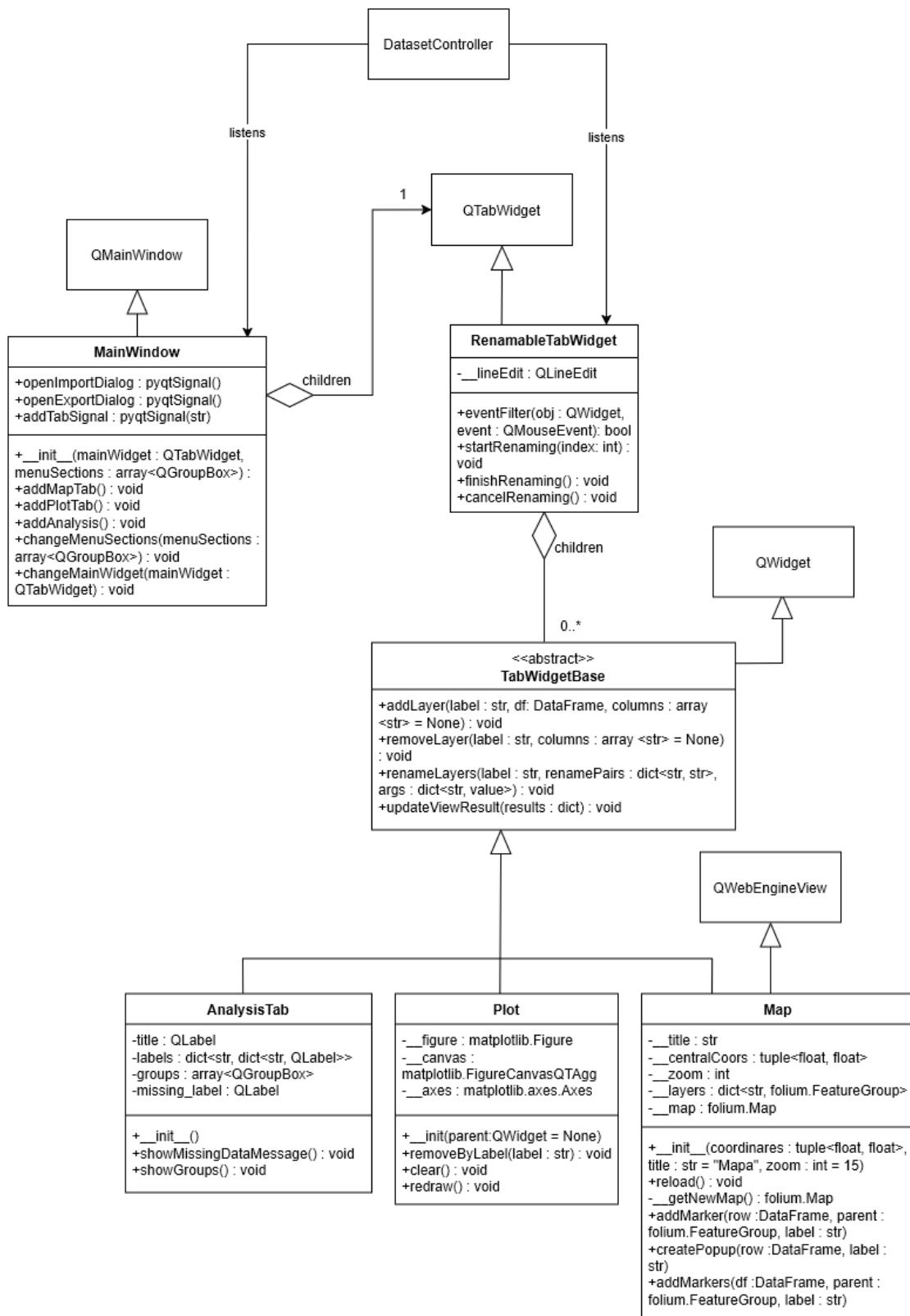


Figura 3.33: Segunda sección de la vista

3.5.2 Diagramas de secuencia

En esta sección se muestran algunos diagramas de secuencia para mostrar el funcionamiento de la aplicación.

Previamente es importante entender la forma en que los elementos de una interfaz PyQt5 interaccionan. La comunicación entre objetos se realiza mediante señales, ranuras y eventos. Las señales son notificaciones emitidas por los elementos de la interfaz cuando algo pasa y las ranuras son las que reciben estas señales. Por otra parte, los eventos son las representaciones de distintos tipos de interacciones, como tocar una tecla. Se puede utilizar la clase `pyqtSignal` para declarar señales propias, que pueden incluir variables como caracteres o valores de verdad que reciben las ranuras conectadas mediante `connect`. Cuando las condiciones deseadas se cumplen se puede emitir la señal correspondiente utilizando el método `emit`, que se emitirá a todos los objetos que hayan hecho el `connect`.

Por ejemplo, para cambiar el método de imputación (Figura 3.36) el panel de imputación tendrá una señal que avisa del cambio mediante `emit` a la que se ha suscrito con `connect` el controlador, quien avisa al modelo; entonces, cuando el modelo realice las operaciones necesarias para cambiar la selección en la representación interna, emitirá una señal a la que está suscrita el controlador y el controlador realizará las acciones pertinentes. En resumidos términos, los elementos de la vista y el modelo emiten señales utilizando `emit` a los que se suscribe el controlador utilizando `connect` y el controlador es quien toma las decisiones para reaccionar a cada señal. Volviendo al ejemplo de cambiar el método de imputación: cuando se realiza la selección desde un elemento de la lista del panel de imputación, el panel emite una señal indicando que variable, PM5 o PM10, y qué método han sido seleccionados y el controlador, que está escuchando la señal, le pide al modelo que cambie la imputación. Cuando almacena el cambio, si la imputación está en modo visible, realiza la imputación con el nuevo método y emite una señal de que los resultados se han actualizado. Cuando el controlador escucha dicha señal del modelo, le pide a la pestaña que se actualice con los nuevos datos.

Asimismo, las señales permiten independencia entre las clases. La clase que emite no tiene que conocer quien recibe, y viceversa; algo representante del patrón *publish-subscribe* que utilizan las señales. En este caso el controlador es quien se encarga de conectar las señales que emite la interfaz o el modelo a lo que debe hacer o a lo que otro elemento debe hacer. De hecho, una señal puede ser ranura de otra y, por tanto, emitirse directamente cuando esta lo haga. En las figuras 3.34 y 3.35, cuando hay que añadir una nueva secuencia resultado de aplicar una corrección a los datos de otra original, el modelo emite una señal avisando de que se ha añadido y el controlador se encarga de que la vista se actualice.

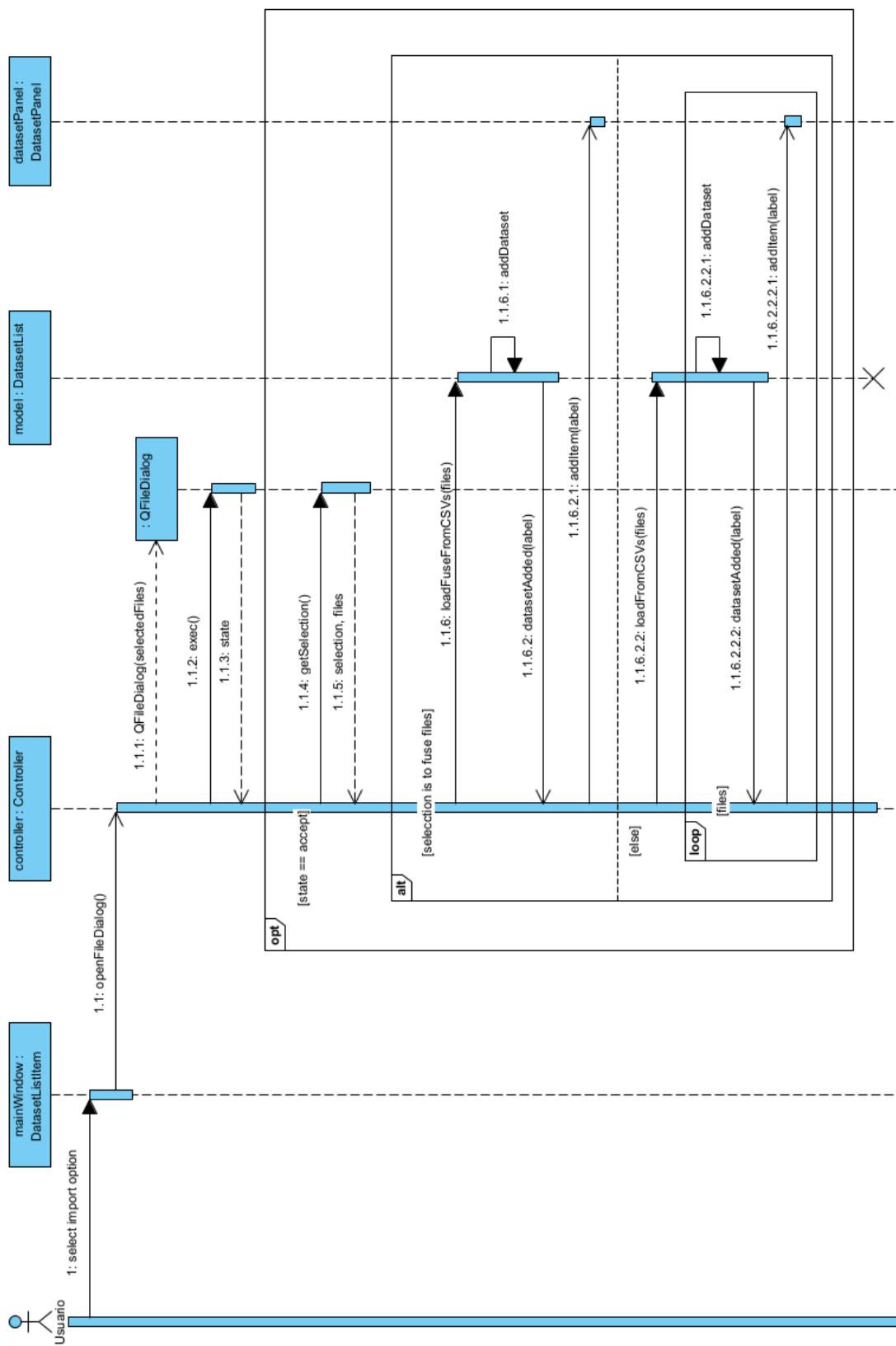


Figura 3.34: Diagrama de secuencia para la importación de series temporales de archivos

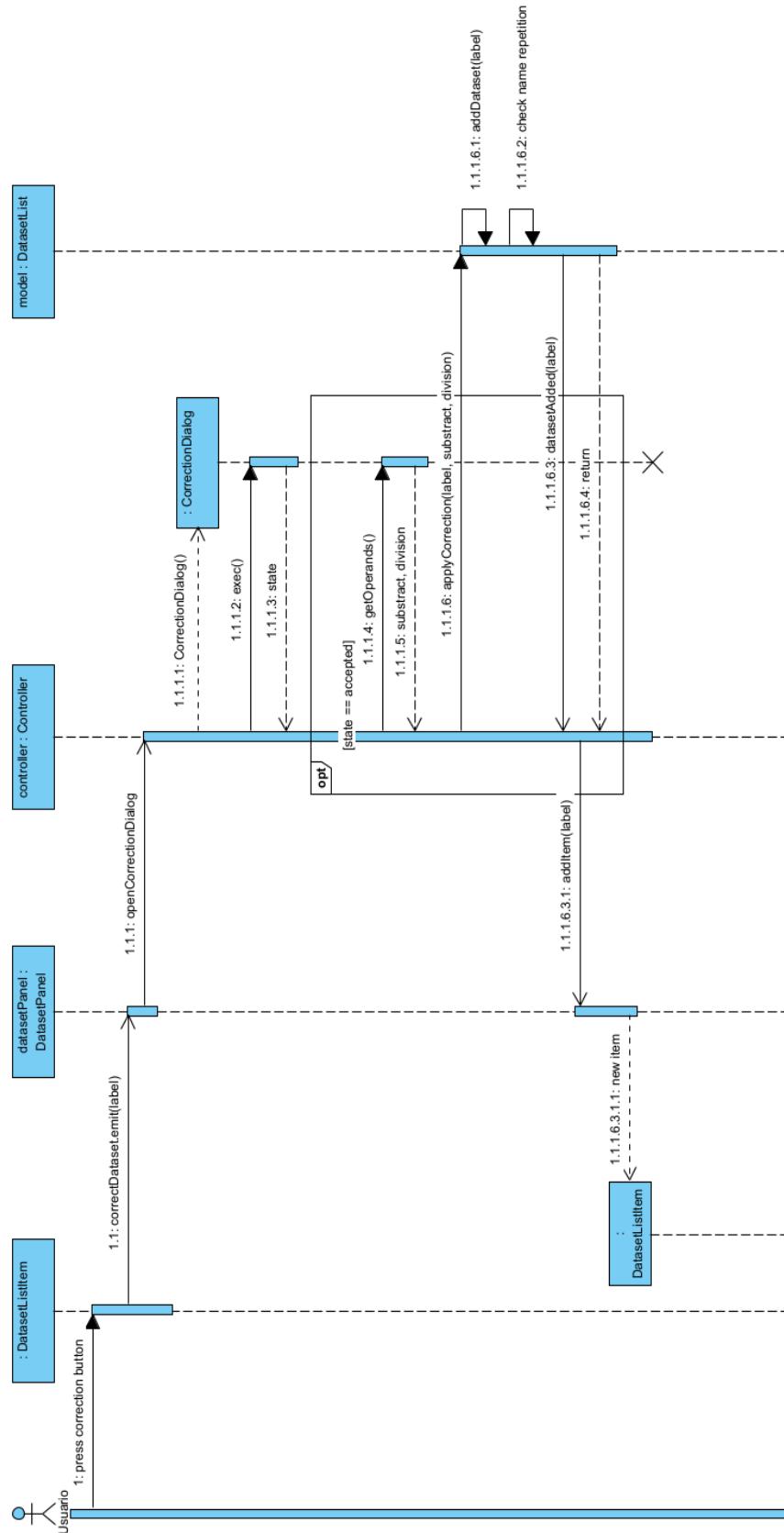


Figura 3.35: Diagrama de secuencia para aplicar una corrección a una serie

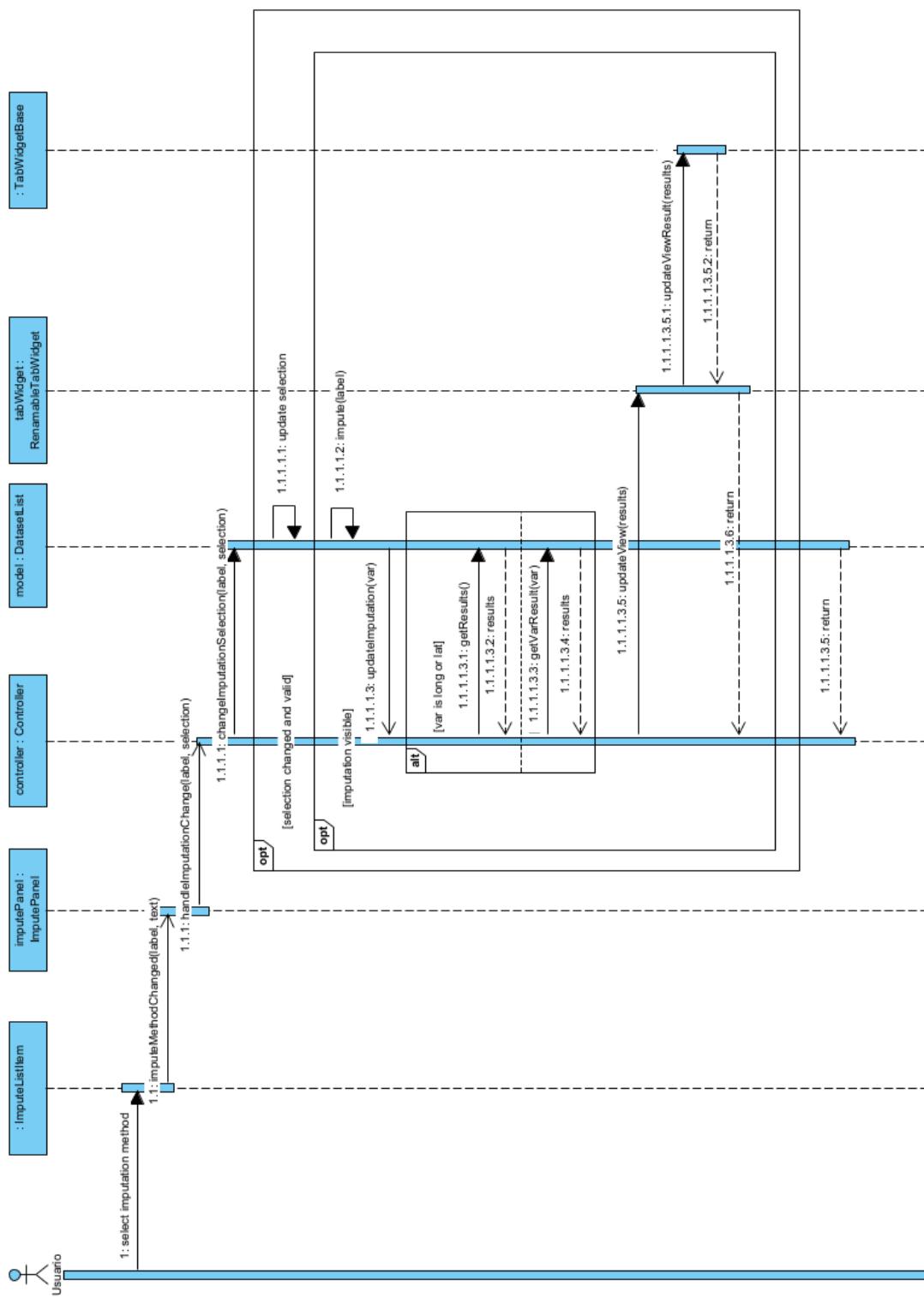


Figura 3.36: Diagrama de secuencia para la selección de método de imputación

3.5.3 Pruebas de validación

Durante el desarrollo se deben realizar pruebas de validación para asegurar que cada incremento es funcional y aceptable, en esta sección se recogen todas las pruebas que han sido realizadas. En todos los *sprints* se realizan pruebas manuales para comprobar la implementación, sin embargo, a medida que crece la aplicación, también es necesario crear pruebas automatizadas para testar de forma más rápida y coherente. Las pruebas automatizadas realizadas se clasifican en dos grupos: pruebas unitarias del modelo, puesto que es quién alberga la lógica del programa y merece pruebas exhaustivas, de caja blanca (Tablas 28-51) y pruebas de integración del sistema de caja negra, para garantizar que modelo-vista-controlador funcionen juntos de forma efectiva (Tablas 52-55). Las pruebas han sido desarrolladas utilizando *pytest* (holger krekel and pytest-dev team, 2015) y el plugin *pytest-qt* para aquellas que necesitan simular la interacción de un usuario con la interfaz o detectar la emisión de una señal.

Tabla 3.28: Caso de prueba 1

ID	01
Nombre	Generar nombres únicos
Descripción	Cuando se crea un nombre para identificar una serie temporal este debe ser único. Se prueba la función “tryName(label)” del modelo.
Acciones	Introducir nombre repetido
Datos iniciales	Modelo creado y en la lista de secuencias hay una llamada “data1”
Resultado esperado	La lista de secuencias contiene dos secuencias llamadas “data1” y “data1(1)”

Tabla 3.29: Caso de prueba 2

ID	02
Nombre	Generación de nombre correcta
Descripción	Generar nombre a partir de una base y añadidos. Se prueba la función “genName(label, addOns)” del modelo.
Acciones	Utilizar la función para generar nombre con una base y algún añadido
Datos iniciales	Modelo creado
Resultado esperado	Se obtiene una cadena de caracteres que incluye la base y los añadidos separados por guiones

Tabla 3.30: Caso de prueba 3

ID	03
Nombre	Comprobar agregación de serie temporal correcta

Descripción	Se añaden series temporales a la aplicación y el modelo emite la señal correspondiente. Se prueba la función “addDataset(label, df)” del modelo.
Acciones	Crear una serie temporal y agregarla al modelo.
Datos iniciales	Modelo creado
Resultado esperado	El modelo emite la señal que contiene la etiqueta de la serie temporal agregada y su lista de series temporales contiene la serie agregada.

Tabla 3.31: Caso de prueba 4

ID	04
Nombre	Renombramiento de serie temporal correcto
Descripción	Debe ser posible renombrar una serie temporal exitosamente y se debe avisar correctamente. Se prueba la función “renameDataset(oldLabel, newLabel)” del modelo.
Acciones	Renombrar la serie de datos y esperar respuesta.
Datos iniciales	Modelo creado y serie temporal agregada.
Resultado esperado	La función devuelve verdadero, se ha emitido una señal que contiene el nombre viejo y el nuevo, y la serie temporal renombrada tiene el nuevo nombre.

Tabla 3.32: Caso de prueba 5

ID	05
Nombre	Casos fallidos de renombramiento de serie temporal
Descripción	Renombrar una serie temporal es innecesario si el nuevo nombre ya es su nombre o si la nueva etiqueta está vacía, puesto que esto no es un buen identificador. Se prueba la función “renameDataset(oldLabel, newLabel)” del modelo.
Acciones	Intentar renombrar con el nombre que ya posee y seguidamente con una cadena vacía.
Datos iniciales	Modelo creado y serie temporal agregada.
Resultado esperado	La función devuelve falso, pero no se produce error.

Tabla 3.33: Caso de prueba 6

ID	06
----	----

Nombre	Eliminación de serie temporal correcta
Descripción	Al eliminar una serie temporal de forma exitosa se debe emitir una señal con la etiqueta de la serie eliminada. Se prueba la función “removeDataset(label)” del modelo.
Acciones	Eliminar serie terminal utilizando su etiqueta
Datos iniciales	Modelo creado, pestaña agregada y serie temporal agregada
Resultado esperado	La función devuelve verdadero, se ha emitido una señal avisando de la eliminación y la lista de series temporales no contiene la serie eliminada

Tabla 3.34: Caso de prueba 7

ID	07
Nombre	Eliminación incorrecta de una serie temporal
Descripción	Que una eliminación de serie temporal no tenga éxito no debe conllevar error. Se prueba la función “removeDataset(label)” del modelo.
Acciones	Eliminar serie terminal utilizando su etiqueta cuando dicha etiqueta no existe entre la lista de series
Datos iniciales	Modelo creado, pestaña agregada y serie temporal agregada
Resultado esperado	La función devuelve falso, no se ha emitido una señal avisando de la eliminación y la lista de series temporales no ha cambiado

Tabla 3.35: Caso de prueba 8

ID	08
Nombre	Duplicación correcta
Descripción	La duplicación correcta conlleva la creación de una nueva serie temporal idéntica, pero de distinto nombre, a la original. Se prueba la función “duplicateDataset(label)” del modelo.
Acciones	Duplicar serie indicando la etiqueta de la original
Datos iniciales	Modelo creado y serie temporal agregada
Resultado esperado	La señal de agregar ha sido emitida con el nombre de la réplica que contiene la palabra “ <i>duplicate</i> ” en el nombre

Tabla 3.36: Caso de prueba 9

ID	09
Nombre	Aplicar exitosamente una corrección a los valores medidos por el sensor
Descripción	Se prueba la función “applyCorrection(label, subtract, division)” del modelo.
Acciones	Aplicar corrección con valores de prueba
Datos iniciales	Modelo creado y serie temporal agregada
Resultado esperado	La señal de agregar ha sido emitida con el nombre de la serie resultante de la corrección, que contiene la palabra “ <i>corrected</i> ” en el nombre, cuyos valores son el resultado de restarles el valor de <i>subtract</i> y dividir entre <i>division</i>

Tabla 3.37: Caso de prueba 10

ID	10
Nombre	Exportar correctamente fusión de resultados a archivo csv
Descripción	Se prueba la función “saveFuseToCSV(labels, file)” del modelo.
Acciones	Realizar exportación indicando nombre de las series temporales y el archivo resultante deseado
Datos iniciales	Modelo creado y más de una serie temporal agregada
Resultado esperado	Se obtiene un sólo archivo csv que contiene la combinación de valores entre las series temporales seleccionadas y se ha nombrado con la cadena de caracteres introducida

Tabla 3.38: Caso de prueba 11

ID	11
Nombre	Importar correctamente archivo csv como serie temporal
Descripción	Se prueba la función “loadFromCSV(file)” del modelo.
Acciones	Crear una serie temporal y almacenar en archivo csv temporal, el cual se seleccionará para la importación
Datos iniciales	Modelo creado
Resultado esperado	La señal de agregar ha sido emitida con el nombre de la serie importada, la cual se encuentra agregada a la lista de series y es

idéntica la serie que se creó para la prueba

Tabla 3.39: Caso de prueba 12

ID	12
Nombre	Exportación de varias series fusionadas en un archivo
Descripción	Se prueba la función “loadFuseFromCSVs(files)” del modelo.
Acciones	Crear varias series temporales y almacenarlas en archivos csv temporales para la importación
Datos iniciales	Modelo creado
Resultado esperado	La señal de agregar ha sido emitida con el nombre de la serie importada la cual ha recibido un nombre contenido la cadena “ <i>Combined</i> ” y es la combinación de las series temporales creadas para la prueba

Tabla 3.40: Caso de prueba 13

ID	13
Nombre	Limpiar todos los resultados de imputación actuales
Descripción	Se prueba la función “clearWorking()” del modelo.
Acciones	Almacenar valores como resultados en el modelo de la pestaña de prueba y ejecutar la función
Datos iniciales	Modelo y objeto de información de pestaña creados
Resultado esperado	Se ha emitido una señal de eliminación de serie temporal y los resultados de imputación de la pestaña deben encontrarse vacíos

Tabla 3.41: Caso de prueba 14

ID	14
Nombre	Limpiar resultados de imputación de una variable dada
Descripción	Se prueba la función “clearWorking(var)” del modelo.
Acciones	Almacenar valores como resultados en el modelo de la pestaña de prueba y ejecutar la función
Datos iniciales	Modelo y objeto de información de pestaña creados
Resultado esperado	Se ha emitido una señal de eliminación de serie temporal y los resultados de imputación de la pestaña no deben contener los

resultados asociados a la variable dada, pero sí todos los demás

Tabla 3.42: Caso de prueba 15

ID	15
Nombre	Cambiar de método de imputación correctamente
Descripción	Se prueba la función “changeImputationSelection(variable, selectedMethod)” del modelo.
Acciones	Cambiar el método de imputación a uno válido
Datos iniciales	Modelo, serie temporal y objeto de información de pestaña creado, y método de imputación seleccionado
Resultado esperado	La señal del cambio de imputación ha sido emitida y el método de imputación relacionado a la variable ha sido actualizado

Tabla 3.43: Caso de prueba 16

ID	16
Nombre	Ignorar cambio de método de imputación
Descripción	Cuando la variable a imputar o el método elegido correspondiente son inválidos, no debe dar fallo, pero no debe haber cambios. Se prueba la función “changeImputationSelection(variable, selectedMethod)” del modelo.
Acciones	Intentar cambiar el método de imputación a uno inexistente e intentarlo para una variable inválida
Datos iniciales	Modelo, serie temporal y objeto de información de pestaña creado, y método de imputación seleccionado
Resultado esperado	No debe haber cambio en la representación interna

Tabla 3.44: Caso de prueba 17

ID	17
Nombre	Cambio de resultado de imputación a visible
Descripción	Cuando el resultado de imputación asociado a una variable se cambia a visible, este se debe de actualizar. Se prueba la función “changeImputationVisibility(variable, checked)” del modelo.
Acciones	Intentar cambiar la visibilidad del resultado de imputación asociado a una variable a visible

Datos iniciales	Modelo, serie temporal y objeto de información de pestaña creado, y método de imputación seleccionado
Resultado esperado	La señal del cambio de viabilidad de imputación ha sido emitida, informando de la variable asociado y del estado de visibilidad como cierto; además de que este cambio ha sido aplicado a la representación interna

Tabla 3.45: Caso de prueba 18

ID	18
Nombre	Cambiar a visible una serie temporal
Descripción	Se prueba la función “changeDatasetVisibility(label, checked)” del modelo.
Acciones	Cambiar serie temporal al estado de visible
Datos iniciales	Modelo, serie temporal y objeto de información de pestaña creados
Resultado esperado	La señal de serie temporal agregada a las visibles ha sido emitida con el nombre de la serie hecha visible y se ha agregado a la lista de series visibles en la pestaña

Tabla 3.46: Caso de prueba 19

ID	19
Nombre	Cambiar a no visible una serie temporal
Descripción	Se prueba la función “changeDatasetVisibility(label, checked)” del modelo.
Acciones	Cambiar serie temporal al estado de no visible
Datos iniciales	Modelo, serie temporal y objeto de información de pestaña creados
Resultado esperado	La señal de eliminar serie temporal ha sido emitida con el nombre de la serie eliminada de visible y se ha eliminado de la lista de series visibles en la pestaña

Tabla 3.47: Caso de prueba 20

ID	20
Nombre	Cambiar a pestaña válida
Descripción	Se prueba la función “changeTab(index)” del modelo en caso de índice válido.

Acciones	Crear una pestaña válida, agregarla al modelo manualmente y utilizar el método changeTab para cambiar a esa pestaña
Datos iniciales	Modelo creado
Resultado esperado	Se ha emitido la señal de cambio de pestaña y la pestaña actual es la pestaña creada

Tabla 3.48: Caso de prueba 21

ID	21
Nombre	Cambiar a pestaña inválida
Descripción	Se prueba la función “changeTab(index)” del modelo en caso de índice inválido.
Acciones	Intentar cambiar la pestaña a una cuyo índice excede la cantidad de pestañas actuales
Datos iniciales	Modelo creado
Resultado esperado	Sin cambios internos

Tabla 3.49: Caso de prueba 22

ID	22
Nombre	Calcular RMSE correctamente
Descripción	Se prueba que la función “rootMeanSquaredError(predicted, actual, col)” del modelo devuelve un resultado correcto.
Acciones	Importar numpy y preparar dos series
Datos iniciales	Modelo creado
Resultado esperado	Valor cercano al cálculo del RMSE entre las dos series

Tabla 3.50: Caso de prueba 23

ID	23
Nombre	Obtener una copia de los resultados de imputación
Descripción	Se prueba la función “getResults()” del modelo.
Acciones	Simular resultados de imputación
Datos iniciales	Modelo creado y objeto de información de pestaña creados

Resultado esperado	El objeto devuelto no es el mismo que el que representa los resultados pero contiene los mismos valores
--------------------	---

Tabla 3.51: Caso de prueba 24

ID	24
Nombre	Obtener copia de los resultados de imputación de una variable concreta
Descripción	Se prueba la función “getVarResult(var)” del modelo.
Acciones	Simular resultados de imputación
Datos iniciales	Modelo creado y objeto de información de pestaña creados
Resultado esperado	Resultado asociado a una variable concreta

Tabla 3.52: Caso de prueba 25

ID	25
Nombre	Crear pestaña inicial
Descripción	Al iniciar se crea una pestaña inicial en la vista por orden del controlador
Acciones	Iniciar aplicación
Datos iniciales	Vista (ventana principal, pestañas y paneles), controlador y modelo preparados
Resultado esperado	No hay fallos y la vista contiene una pestaña inicial

Tabla 3.53: Caso de prueba 26

ID	26
Nombre	Añadir pestaña y añadir serie temporal
Descripción	Cuando la vista o el modelo emitan señales de agregación, será el controlador quién las escuche y reaccione de acuerdo a cada señal recibida, logrando que se muestren los cambios al usuario y se graben en la representación interna
Acciones	Añadir nueva pestaña y preparar serie de prueba, seguidamente, emitir señales correspondientes desde ventana principal y panel de series temporales
Datos iniciales	Vista (ventana principal, pestañas y paneles), controlador y modelo preparados

Resultado esperado	Se ha añadido una nueva pestaña a la vista y una nueva serie temporal al panel de series
--------------------	--

Tabla 3.54: Caso de prueba 27

ID	27
Nombre	Eliminar serie temporal
Descripción	Debe ser posible eliminar series temporales desde el panel correspondiente de forma que el controlador actúe como intermediario y este cambio acabe reflejado en el modelo.
Acciones	Añadir serie de prueba en el modelo y esperar a la agregación en el panel, después, desencadenar eliminación desde el panel de series temporales
Datos iniciales	Vista (ventana principal, pestañas y paneles), controlador y modelo preparados
Resultado esperado	Señal de eliminación emitida en el modelo y serie temporal no presente en este

Tabla 3.55: Caso de prueba 28

ID	28
Nombre	Cambiar método de imputación
Descripción	Cambiar el método de imputación en el panel de imputación se ve reflejado en la representación interna
Acciones	Cambiar método de imputación desde el panel de imputación
Datos iniciales	Vista (ventana principal, pestañas y paneles), controlador y modelo preparados
Resultado esperado	La selección de método de imputación para la variable seleccionada ha cambiado a la nueva selección

Capítulo 4. Resultado final

La aplicación resultante, el manual de usuario y el código fuente se pueden encontrar en: <https://github.com/MartaxM/TFG-PyQt-Imputation-App.git>. Además, la información de este capítulo también se encuentra en el manual de usuario.

La licencia de este software es la Licencia Pública General de GNU o *GNU General Public License* (o sus siglas GNU GPL). En concreto la versión 3. Esta licencia define el software como libre para usar, compartir y modificar, y a la misma vez exige que los derivados y copias ofrezcan las mismas libertades (*The GNU General Public License v3.0 - GNU Project - Free Software Foundation, 2007*). Esto pone en práctica el llamado *copyleft*, es decir, el uso de los derechos sobre una obra para ofrecerla de forma libre y evitar su privatización. Al utilizar GNU GPL no hace falta adquirir una licencia comercial para utilizar PyQt, puesto que esto respeta las condiciones que detalla la compañía propietaria Riverbank Computing (Riverbank Computing, 2025c).

4.1 Funcionalidades

En esta parte detallaremos las funcionalidades de la aplicación. Esta información también se detalla en el manual de usuario

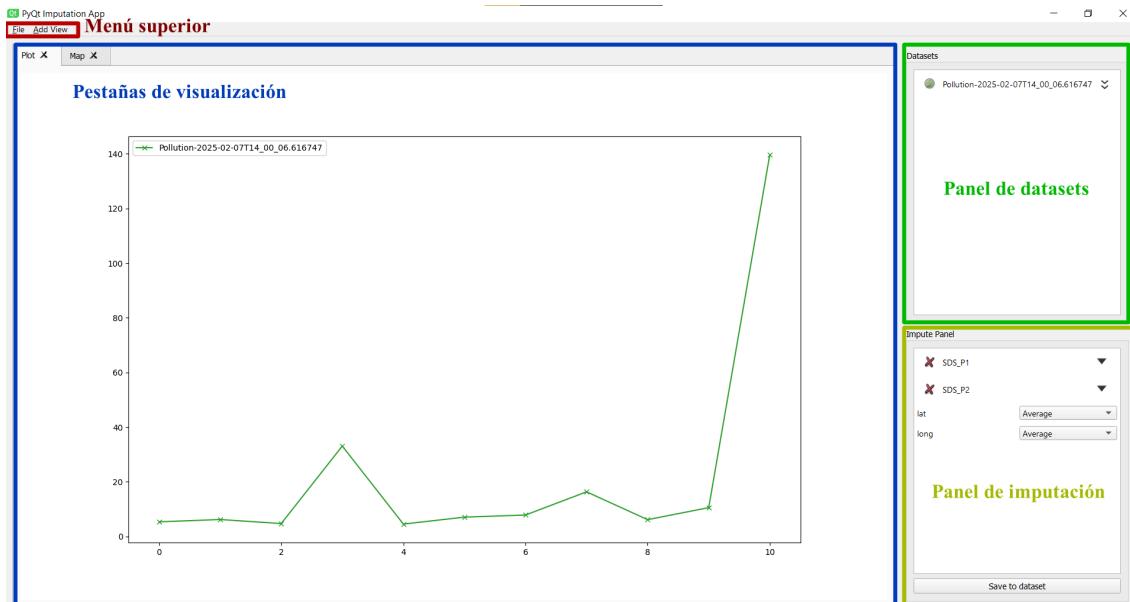


Figura 4.1: Áreas de la interfaz

Como se ve en la figura 4.1, la interfaz se divide en tres secciones principales y el menú superior. Los dos paneles laterales forman el menú lateral.

4.1.1 Importación y exportación de archivos

La aplicación trabaja extrayendo y guardando las secuencias temporales en archivos CSV. Ambas acciones se acceden desde el menú de herramientas superior (Figura 4.2).

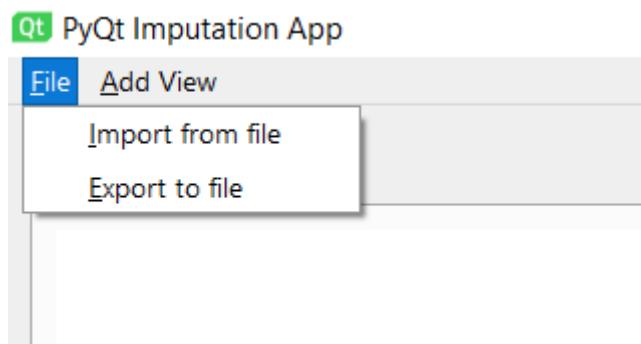


Figura 4.2: Opciones de importación/exportación

Importación

La opción “*Import from file*” abrirá un explorador de archivos para seleccionar el o los archivos CSV a importar. En caso de elegir un sólo archivo, se añadirá directamente una nueva secuencia no visible al panel de imputación con el mismo nombre del archivo. En caso de seleccionar varios archivos, aparecerá un segundo diálogo (Figura 4.3) que permite elegir entre fusionar las secuencias o importarlas por separado.

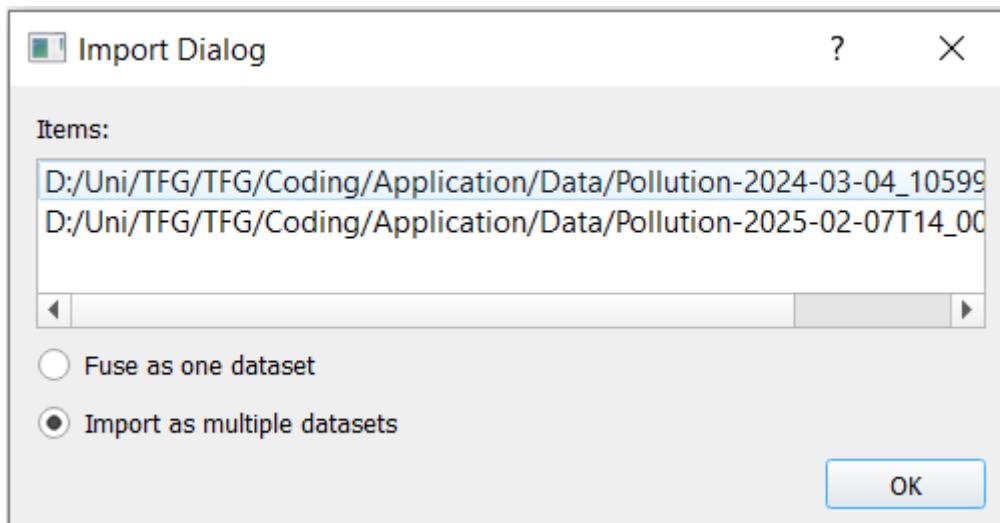


Figura 4.3: Diálogo de importación

Exportación

La opción “*Export to file*” abrirá un diálogo (Figura 4.4) para seleccionar qué secuencias exportar. Pulsar un nombre de la lista lo selecciona y volver a pulsarlo lo deselecciona. Además, se puede si se quieren fusionar las secuencias en un solo archivo o exportarlas en múltiples archivos. Tras esto, aparecerá un explorador de archivos para seleccionar el lugar de guardado y el nombre del archivo.

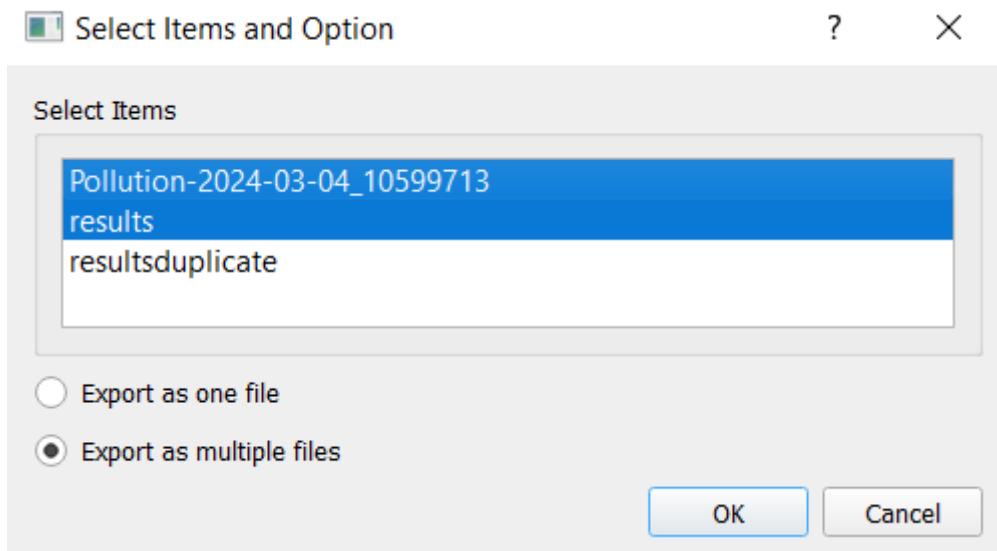


Figura 4.4: Diálogo de exportación

4.1.2 Gestión de pestañas

Añadir pestañas

Para añadir una pestaña de visualización se utiliza la opción “*Add View*” del menú superior, el cual permite elegir cuál pestaña añadir a la vista (Figura 4.5). Cualquier elección generará la pestaña correspondiente con el nombre predeterminado.

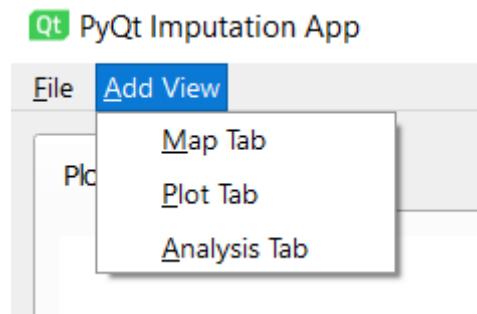


Figura 4.5: Opciones para añadir pestañas

Eliminar pestañas

Para eliminar una pestaña es suficiente con pulsar el botón “X” al lado de su nombre (Figura 4.6). En caso de eliminar todas las pestañas el panel de imputación desaparecerá y será imposible la visualización, sin embargo, seguirá disponible la exportación, importación y manipulación de series mediante el panel de *datasets*.

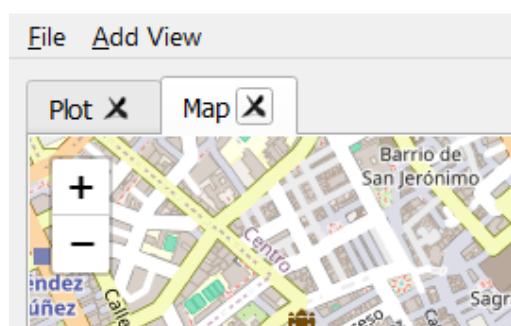


Figura 4.6: Eliminación de pestañas

Renombrar pestañas

Pulsar el nombre de una pestaña dos veces seguidas y en corto tiempo, el conocido “*doble-click*”, permite renombrarla (Figura 4.7). Para finalizar basta con pulsar “*Enter*”.

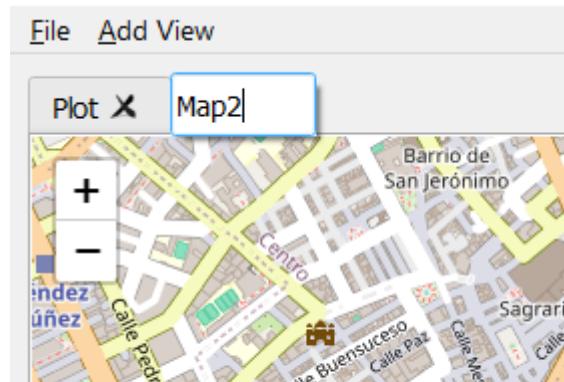


Figura 4.7: Renombramiento de pestañas

Cambiar orden de las pestañas

Mantener pulsado y arrastrar coloca la pestaña en la posición deseada (Figura 4.8).

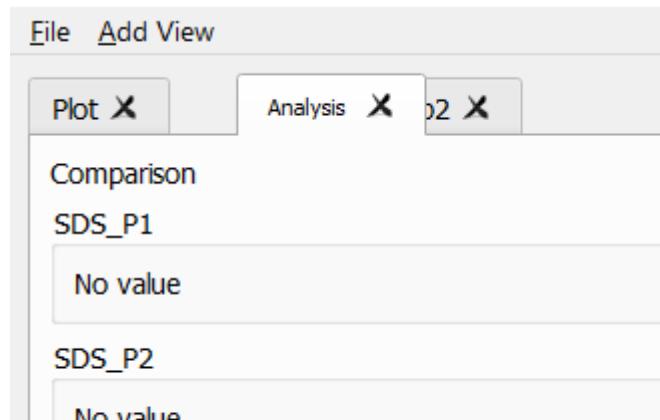


Figura 4.8: Desplazamiento de pestañas

4.1.3 Manipulación secuencias

Cada secuencia tiene un botón para abrir las opciones de manipulación (Figura 4.9).



Figura 4.9: Opciones sobre secuencias

Quitar valores

La opción “*Remove data*” abre un diálogo para seleccionar de qué forma eliminar valores de la secuencia. Esto es útil para generar una secuencia temporal con valores faltantes perdidos de forma artificial, de forma que se pueda comparar con la secuencia original y observar la eficacia del método de imputación elegido.

Por una parte, la opción “*Random Percent*” (Figura 4.10), elimina una cantidad de valores igual a un porcentaje introducido por el usuario del total de valores en la secuencia.

Mientras que “*Interval*” (Figura 4.11) ofrece la posibilidad de eliminar todos los valores dentro de un intervalo o todos los que queden fuera, a elección del usuario. Cabe destacar que los valores que aparecen inicialmente son el primer valor de tiempo y el último de la secuencia.

El nombre o etiqueta de la secuencia resultante es el mismo que el de la original con el añadido de “*-rm*” y el método usado. Por tanto, las dos posibilidades son “*-rmPercent*” y “*-rmInterval*”.

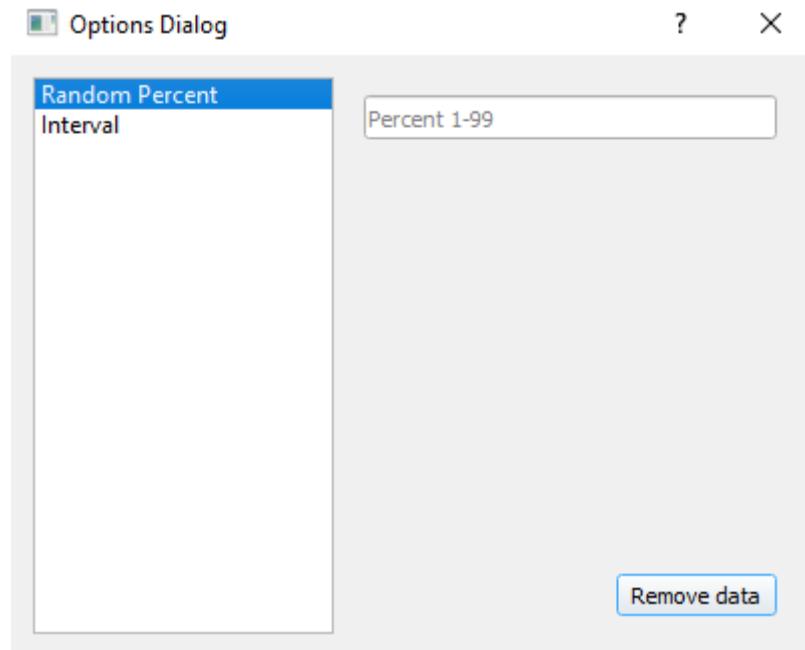


Figura 4.10: Eliminar porcentaje de valores

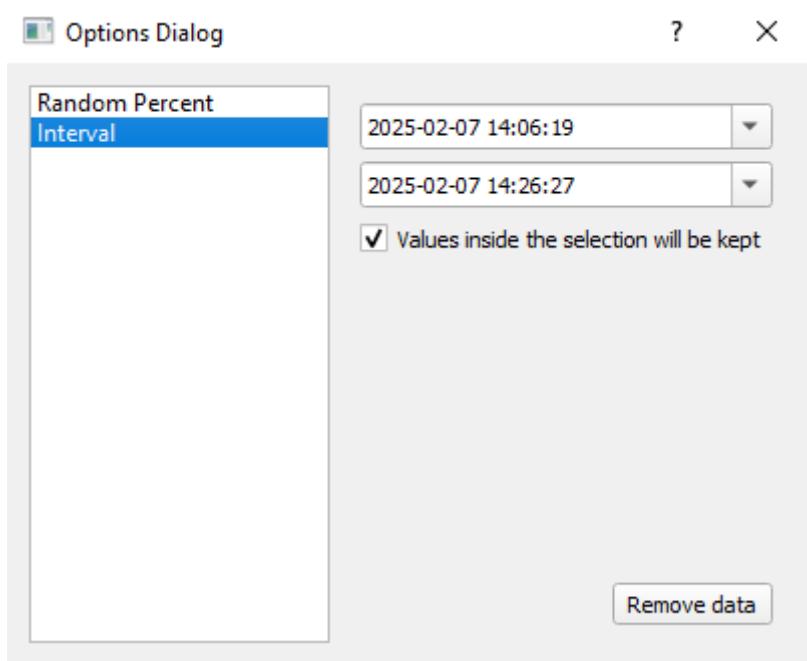


Figura 4.11: Eliminar intervalo de valores

Duplicar

“Duplicate” simplemente genera una copia exacta de la secuencia original. El nombre o etiqueta de la secuencia resultante es el mismo que el de la original con el añadido de “-duplicate”.

Aplicar una corrección de calibración del sensor

Cualquier sensor toma las medidas con cierto error de calibración, para corregir los valores de las medidas se ha calibrado el sensor de bajo coste utilizado en este trabajo con un sensor comercial de alta precisión, perteneciente a un grupo de investigación de la UGR de la facultad de ciencias. Resultado de esta calibración para calcular el valor real de la contaminación a partir del valor medido por el sensor de bajo coste se debe aplicar la siguiente fórmula de corrección:

$$v_{real} = \frac{v_{medido} - x}{y}$$

Los valores para corregir el error en el caso del sensor utilizado para las pruebas son $x = 1,5$ y $y = 0,261$.

La acción “Apply correction” abre el diálogo (Figura 4.12) que se muestra en la Figura X para llenar los valores de la ecuación. El resultado de esta operación es una secuencia igual que la original pero con los valores de partículas en el aire cambiados y con “-corrected” añadido a la etiqueta.

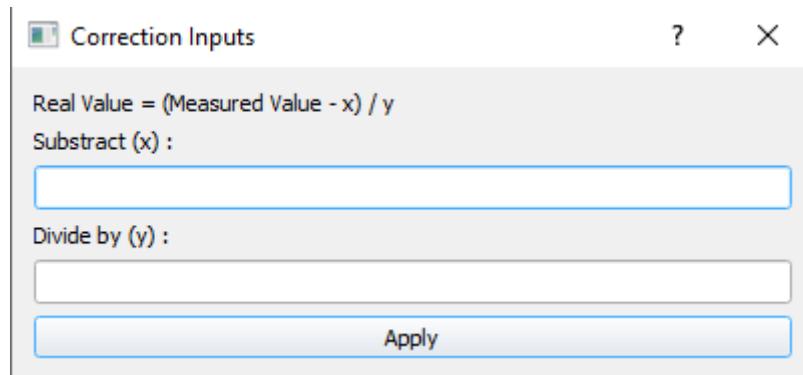


Figura 4.12: Aplicar corrección a una serie

Renombrar

Tanto la opción “Rename” o pulsar la etiqueta de una secuencia dos veces seguidas y en corto tiempo (*doble-click*), activa la modificación del nombre (Figura 4.13). Los nombres de las secuencias resultantes de la mayoría de acciones ganan algún sufijo con el objetivo de diferenciarse de la original. Es por esto que la capacidad de renombrar secuencias es importante.

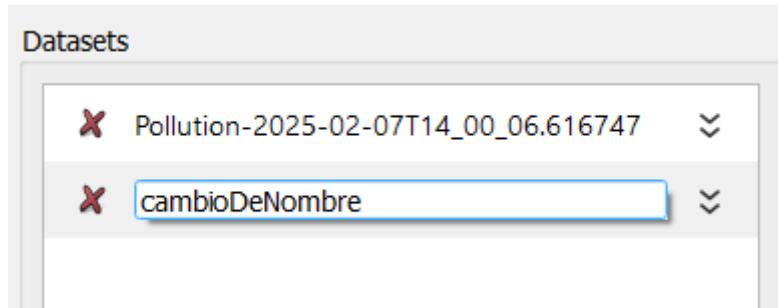


Figura 4.13: Muestra de cambio de nombre de serie temporal

Eliminar una secuencia

La eliminación de una secuencia de la aplicación es mediante la opción “Delete”. Esta acción es inmediata y eliminará la secuencia de la visualización en todas las pestañas, así como sus resultados de imputación si los había.

Repetición de nombres

El nombre de una secuencia es la etiqueta que lo identifica en modelo interno y se diferencia de las demás, por tanto no se puede repetir. Por tanto, para cada acción que añada una secuencia nueva al panel se comprueba si la etiqueta ya existe. En caso de que sea así, el nombre de la nueva secuencia contendrá un paréntesis con el número de secuencias que comparten el mismo nombre en el momento de su creación (Figura 4.14). Esto también afecta cuando se realiza un cambio de nombre.

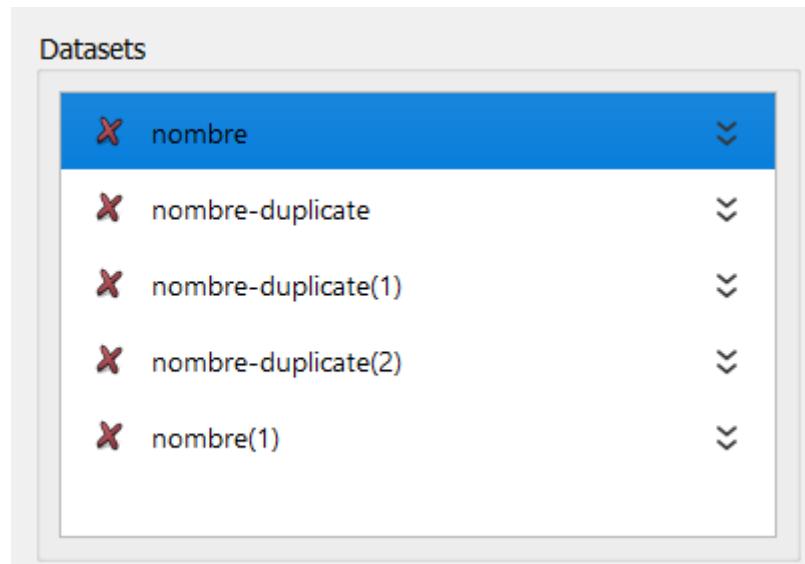


Figura 4.14: Ejemplo de protección contra repetición de nombres

4.1.4 Visualización

Visibilidad de las secuencias

A la izquierda de la etiqueta de nombre de cada secuencia se encuentra un ícono que representa el estado de visibilidad de la misma. Si el ícono es una “X” roja, es que no se encuentra visible y si es un círculo verde, lo contrario. Se puede alternar entre estados pulsando el ícono.

Como se observa en la figura 4.15, en la gráfica de líneas se puede diferenciar a qué secuencia pertenece un valor en base al color. Sin embargo, en el mapa (Figura 4.16) se debe pulsar el punto que simboliza el valor para abrir un desplegable, el cual muestra la secuencia de origen.

Además, en el mapa, los valores aparecen de distinto color según el valor de PM_{2.5} (SDS_P1) y PM₁₀ (SDS_P2).

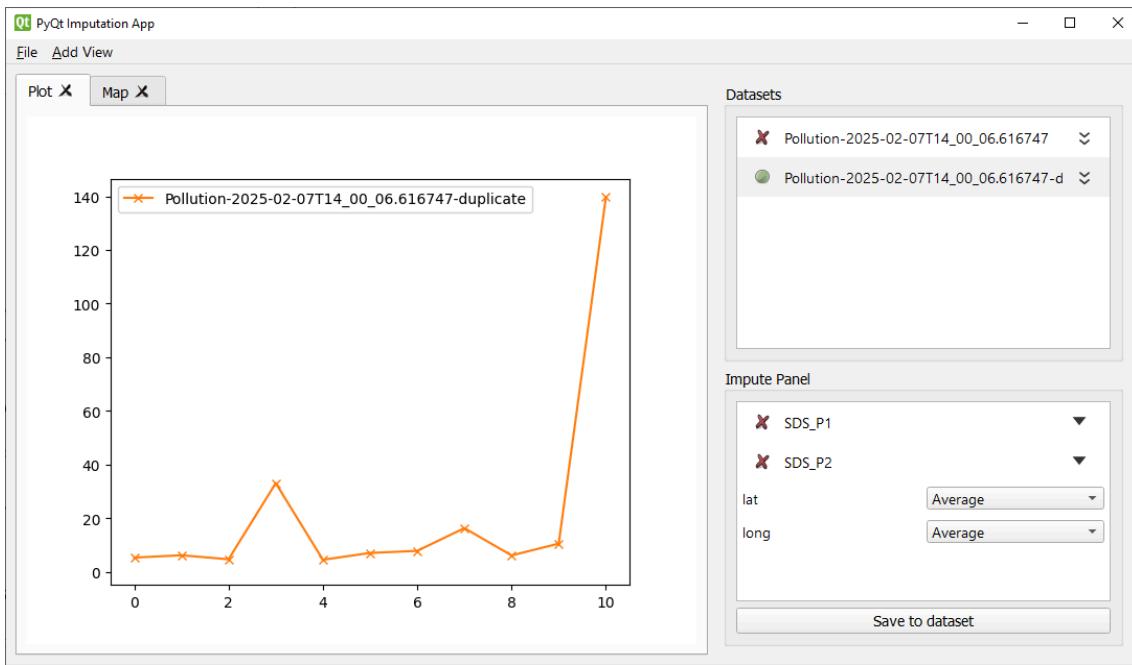


Figura 4.15: Ejemplo de visualización en gráfico de líneas

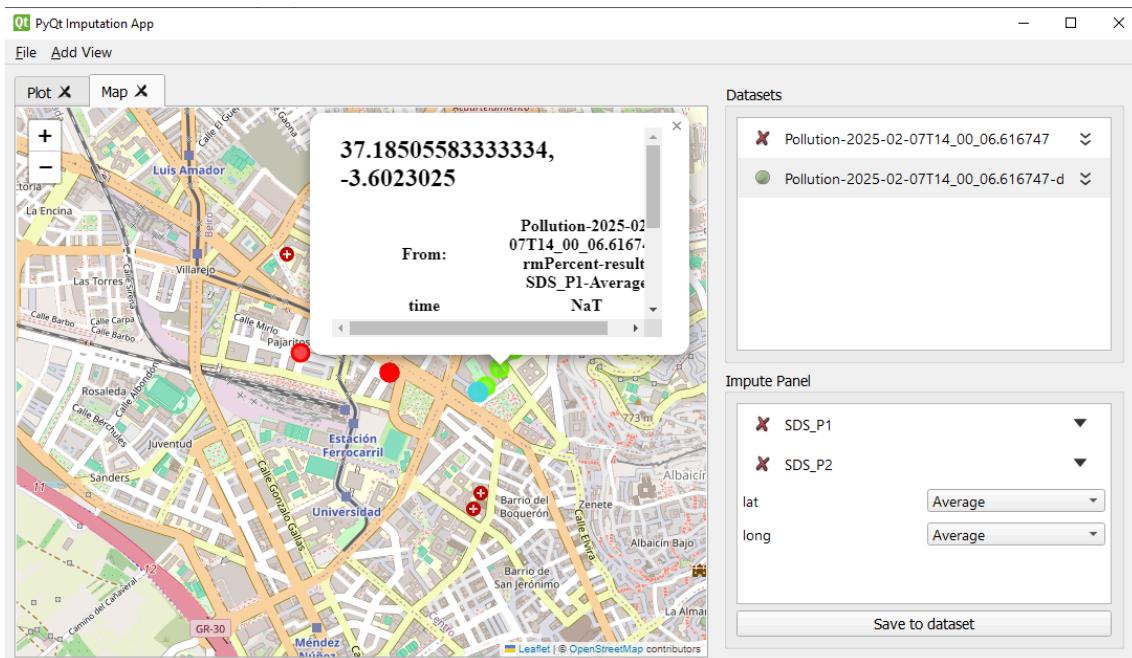


Figura 4.16: Ejemplo de visualización en mapa

Memoria al cambiar las pestañas

Cada pestaña permite tener datasets visibles u ocultos independientemente de su estado de visibilidad en el resto de pestañas. Lo que quiere decir que se almacenan los estados de visibilidad de cada secuencia en cada pestaña (Figura 4.17) y al cambiar de pestaña (Figura 4.18) se visualizará acorde a lo almacenado.

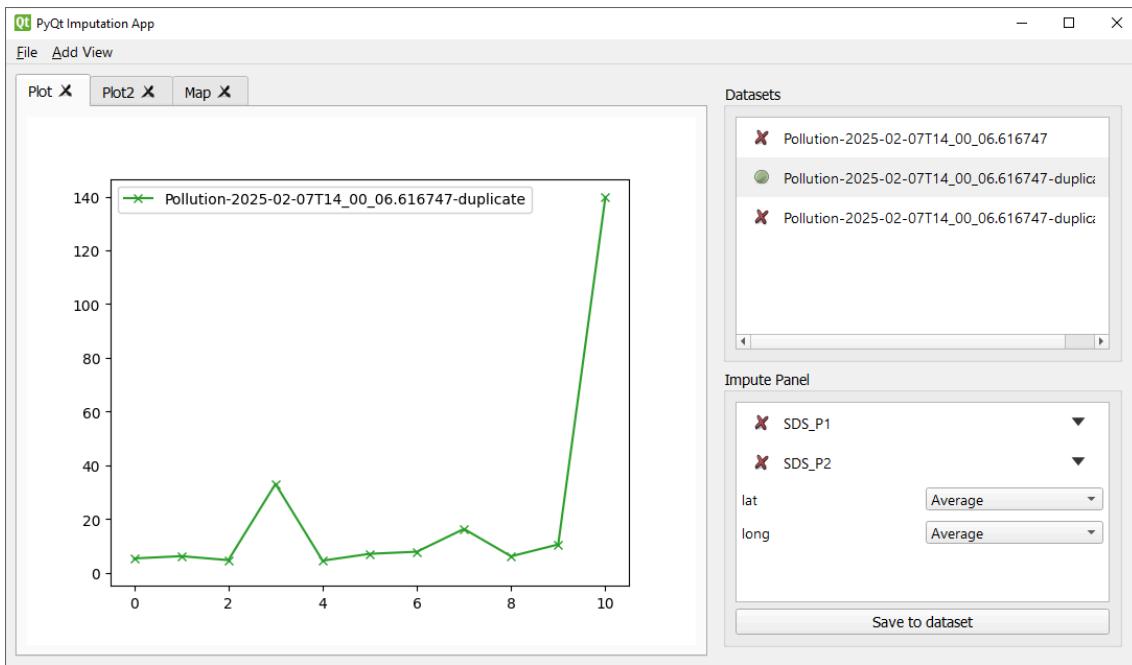


Figura 4.17: Ejemplo de memoria entre pestañas 1

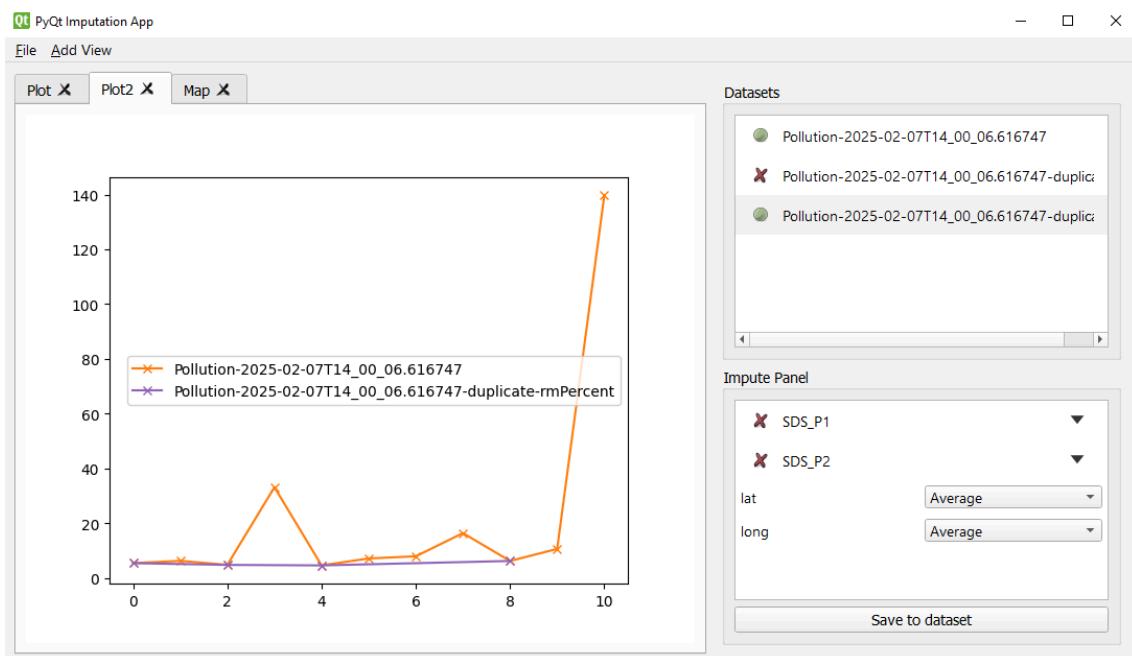


Figura 4.18: Ejemplo de memoria entre pestañas 2

4.1.5 Imputación

Visibilidad de los resultados de imputación

Para realizar la imputación de una secuencia esta se debe seleccionar en el panel de *datasets* y la imputación de la variable debe estar visible (Figura 4.19). A la izquierda de la etiqueta de nombre de cada variable de imputación se encuentra un ícono que representa el estado de visibilidad, similar a los que se encuentran en las secuencias. Si el ícono es una “X” roja, es que no se encuentra visible y si es un círculo verde, lo contrario. Se puede alternar entre estados pulsando el ícono.

No obstante, la visibilidad de las variables de latitud y longitud no se puede cambiar. Esto es porque estas siempre deben imputarse, especialmente para representar los datos en la pestaña de mapa.

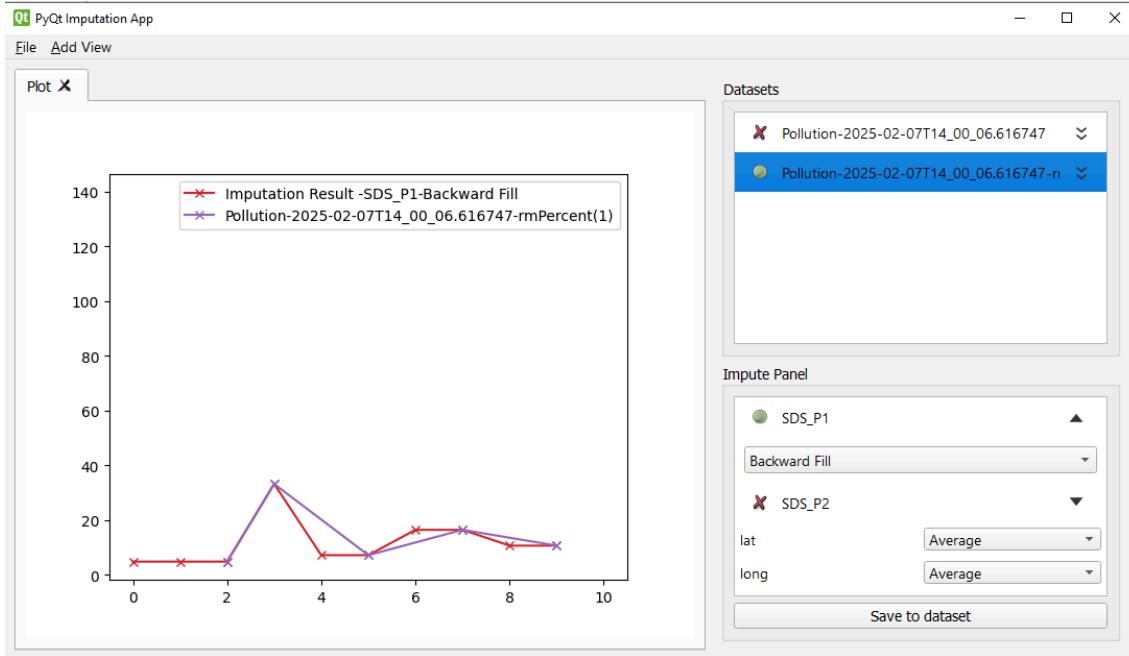


Figura 4.19: Muestra de control de visibilidad

Análisis de resultados

Además de comparar visualmente el resultado de la imputación con la secuencia original, también es posible calcular el RMSE entre ambas. Para ello, se puede crear una pestaña de análisis con “*Add View*” > “*Analysis Tab*”. En esta pestaña desaparece el panel de imputación y se pueden seleccionar dos secuencias para realizar el cálculo (Figura 4.20).

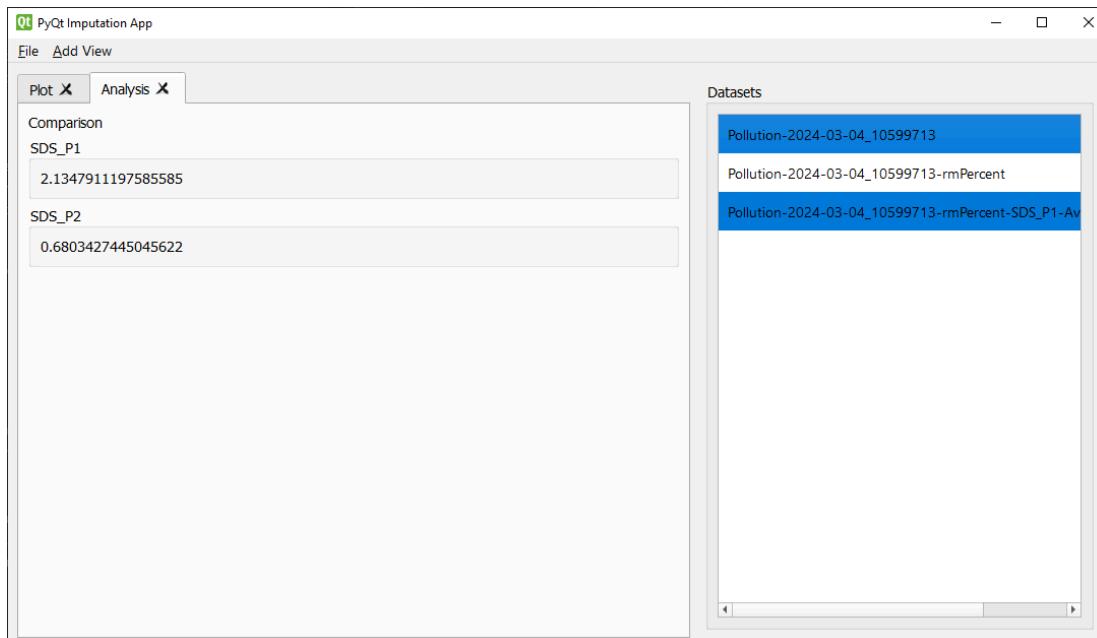


Figura 4.20: Ejemplo de pestaña de análisis

Cambiar método de imputación

Se puede cambiar el método de imputación con el selector al lado de las variables (Figura 4.21). El cambio sólo será efectivo para la variable en la que se aplica el cambio. Los selectores de la latitud y la longitud no se pueden ocultar, puesto que siempre deben imputarse, pero los de SDS_P1 (la variable que mide PM2.5) y SDS_P2 (la variable que mide PM10) se despliegan utilizando la flecha a la derecha de estas. Esto es para ocultar posibles errores cuando sólo importa una de las variables.

Cómo se muestra en la figura 4.22, se pueden seleccionar entre seis métodos:

- **Average:** El valor se imputa utilizando la media del anterior valor no nulo y el siguiente valor no nulo.
- **Backward Fill:** El valor se impute con el siguiente valor no nulo.
- **Forward Fill:** El valor se impute con el anterior valor no nulo.
- **Median:** El valor se imputa utilizando la mediana del anterior valor no nulo y el siguiente valor no nulo.
- **PyPots SAITS:** El valor se imputa entrenando un modelo de aprendizaje automático SAITS.
- **PyPots Transformer:** El valor se imputa entrenando un modelo de aprendizaje automático Transformer.

En el caso de PyPots SAITS y PyPots Transformer, al ser aprendizaje automático, se guarda el modelo entrenado en un archivo .sav con el nombre de la secuencia y el método utilizados.

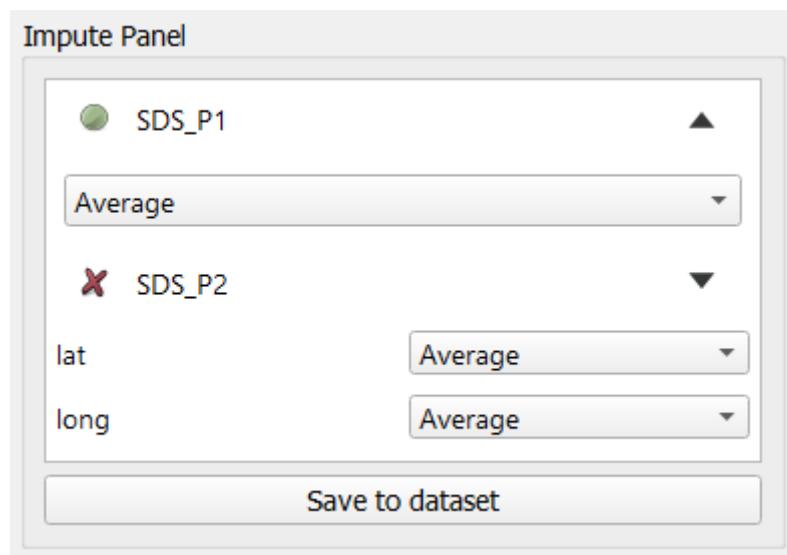


Figura 4.21: Menú de imputación

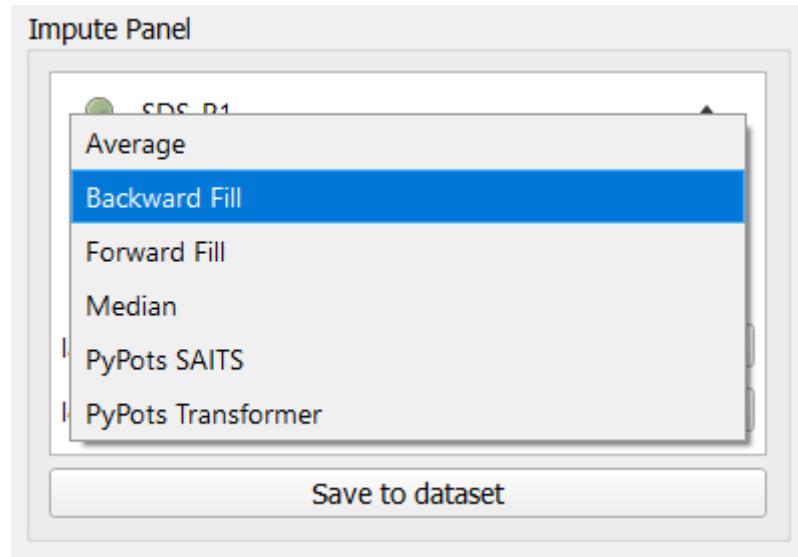


Figura 4.22: Selector de imputación

Guardar resultados

Tras imputar se puede guardar los resultados como secuencia que se agrega al panel de datasets (Figura 4.23). Su nombre es el mismo que el de la original con un sufijo añadido conformado por la variable imputada y el método usado. En caso de estar imputando ambas variables, SDS_P1 y SDS_P2, aparecerán en el resultado. Esta nueva secuencia se puede exportar a un archivo como cualquier otra secuencia.

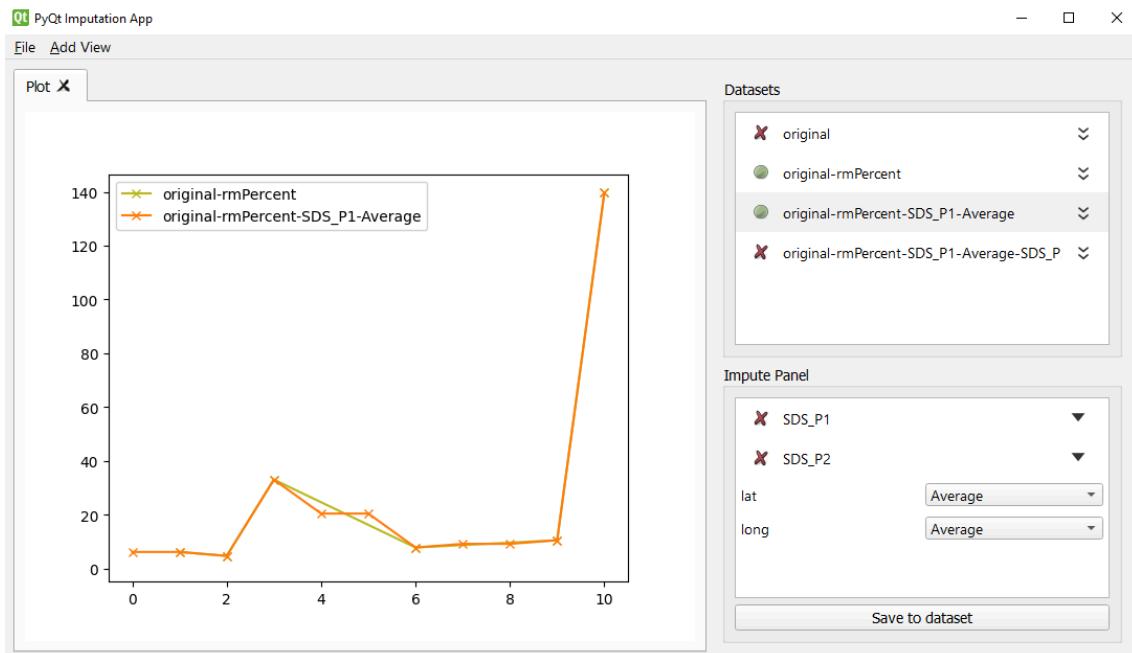


Figura 4.23: Muestra de conversión a serie temporal del resultado de imputación

4.2 Testar

Para ejecutar las pruebas automatizadas descritas en la sección 3.5.3 se hace uso del comando `pytest [ruta de la prueba]`, estado estas en la carpeta `/tests` del código agregado al github. En su interior, las pruebas unitarias de caja blanca del modelo se encuentran en los

archivos de la carpeta */unit*, mientras que las pruebas de integración entre el modelo-vista-controlador de caja negra se encuentran en */integration*.

Capítulo 5. Conclusiones

Finalmente, este trabajo ha conseguido producir una herramienta de código abierto para la visualización e imputación de series temporales procedentes de sensores de calidad del aire. La implementación se ha hecho en Python, utilizando PyQt5 para la interfaz, *pandas* para la manipulación de los datos, *matplotlib* para los gráficos de líneas y *folium* para la visualización sobre el mapa, respetando las licencias de cada paquete utilizado como describe la ética informática y ofreciendo el código de forma pública bajo la licencia GNU GPL. Además de imputación, incluye diversas funciones como la eliminación de valores para crear series con valores faltantes de forma artificial, para que los investigadores puedan comprobar la efectividad de los algoritmos de imputación de valores perdidos con series temporales anotadas con el valor real absoluto (ground true); o un sistema de pestañas para facilitar la visualización. Por lo tanto, esta herramienta final, cumple con todos los objetivos planteados al principio del trabajo. Asimismo, como consideraciones extras: no es necesario preocuparse por la protección de datos puesto que no se maneja información sensible, ni se almacena información de forma externa, y el problema abordado de la imputación de series temporales es especialmente importante en la ciencia medioambiental y el entendimiento de los cambios en la calidad del aire, tema relacionado con los objetivos de desarrollo sostenible de las naciones unidas.

Se ha seguido un ciclo de vida del software estándar, dónde se ha pasado por las fases de planificación (3.4, especialmente 3.4.1), análisis (capítulo 2), diseño (sección 3.5), implementación (descrita en 3.4.1, 3.5 y código disponible en el repositorio enlazado al final de las conclusiones), tests (sección 3.4.3) y despliegue (capítulo 4). Este ciclo se ve afectado por la utilización de la metodología ágil SCRUM como se puede observar en el Capítulo 3 que describe la propuesta, dado que los incrementos se deben probar de forma temprana para asegurar que cada uno sea funcional. El desarrollo se divide en *sprints* y en 3.3.1 Descripción de los sprints se crean 17 tablas resumen con los objetivos, tareas y duración de cada uno, asimismo, en la mayoría de estos se obtiene un incremento y se realizan las pruebas correspondientes. Además, siguiendo el espíritu de SCRUM los requisitos se han mantenido flexibles, indicándose el momento en que se solidifica cada uno en la misma sección y los requisitos finales en la 3.1 Requisitos.

Por otro lado, el análisis de las tecnologías más adecuadas para el desarrollo se muestra en la sección 2.2 Tecnologías para la implementación, dónde se analizan los módulos disponibles y las capacidades que ofrecen cinco alternativas. Se consideran cuatro lenguajes principales y la alternativa de mezclar lenguajes en la implementación, mostrando adhesión al ámbito del problema, puesto que sólo se analizan lenguajes relevantes. Como se observa en la sección, sólo se realiza un análisis más profundo de dos de los lenguajes, ya que durante el proceso de investigación se consideran más prometedores. En la tabla 3.9, se muestra una síntesis del análisis.

Finalmente, haciendo una valoración personal, este TFG ha supuesto utilizar y expandir mis conocimientos en desarrollo de software y gestión de proyectos, puesto que ha simulado un proyecto profesional y profundizado en una técnica en concreto del desarrollo ágil. También ha sido un gran ejercicio de aprendizaje al tener que aprender Python, PyQt, pandas y folium con

los que no tenía experiencia previa. Por otra parte, también he adquirido conocimientos sobre imputación y análisis de datos, algo fuera de mi zona de confort. Mirando hacia atrás, la parte que he abordado de forma más negativa es la priorización, aunque, gracias a la guía y frecuentes reuniones con los tutores, creo que es algo que he acabado mejorando. También, por eso mismo, mi habilidad para recibir y aplicar retroalimentación ha aumentado. En conclusión, este proyecto ha involucrado y mejorado tanto mi forma personal de abordar dificultades, como habilidades técnicas adquiridas durante la carrera, por ejemplo, el uso y entendimiento de patrones y arquitectura software de “Desarrollo Software” o las capacidades de planificación temporal y organización de tareas de las asignaturas “Dirección y Gestión de Proyectos” y “Metodologías de desarrollo Ágil” cursadas durante el ERASMUS, además de las propias competencias de programación desarrolladas durante toda carrera que son transferibles a otros lenguajes de programación.

La aplicación resultante, el código fuente y el manual se pueden encontrar en el repositorio: <https://github.com/MartaxM/TFG-PyQt-Imputation-App.git>

Bibliografía

- A Primer on the R-Tcl/Tk Package. (2001, septiembre). *R News*. Recuperado 16 de octubre de 2025, de <https://journal.r-project.org/articles/RN-2001-026/RN-2001-026.pdf>
- Abuchar, A. (2023). *Metodologías ágiles para el desarrollo de software*.
<https://doi.org/10.69740/ud.9789587875119.9789587875133.9789587875126>
- Bass, J. M. (2016). Artefacts and agile method tailoring in large-scale offshore software development programmes. *Information And Software Technology*, 75, 1-16.
<https://doi.org/10.1016/j.infsof.2016.03.001>
- Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, 59(1), 65-98. <https://doi.org/10.1137/141000671>
- Bhatia, N., & Vandana. (2010). Survey of Nearest Neighbor Techniques. *arXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.1007.0085>
- Comprehensive R Archive Network (CRAN). (2025, 11 septiembre). *Create Elegant Data Visualisations Using the Grammar of Graphics [R package ggplot2 version 4.0.0]*.
<https://cran.r-project.org/web/packages/ggplot2/index.html>
- CRAN: Package RGtk2. (2025). Recuperado 16 de octubre de 2025, de
<https://cran.r-project.org/web/packages/RGtk2/index.html>
- Crear mapas con datos de latitud y longitud. (2025). Mathworks.es. Recuperado 16 de octubre de 2025, de
https://es.mathworks.com/help/matlab/creating_plots/plot-in-geographic-coordinates.html
- Cunningham, W. (2001). *Manifiesto por el Desarrollo Ágil de Software*.
<https://agilemanifesto.org/iso/es/manifesto.html>
- Davis, P. J. (1975). *Interpolation and Approximation*. Courier Corporation.

- Dreyfus, G. (2005). *Neural networks: Methodology and Applications*. Springer Science & Business Media.
- Du, J., Hu, M., & Zhang, W. (2020). Missing Data Problem in the Monitoring System: A Review. *IEEE Sensors Journal*, 20(23), 13984-13998.
<https://doi.org/10.1109/jsen.2020.3009265>
- Du, W. (2023). PyPOTS: A Python Toolbox for Data Mining on Partially-Observed Time Series. *arXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.2305.18811>
- Du, W., Côté, D., & Liu, Y. (2022). SAITS: Self-Attention-based Imputation for Time Series. *arXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.2202.08516>
- Glassdoor. (2025). *Sueldos para el puesto de Programador Junior en España*. Recuperado 16 de octubre de 2025, de https://www.glassdoor.es/Sueldos/programador-junior-sueldo-SRCH_KO0,18.htm
- Grund, S., Lüdtke, O., & Robitzsch, A. (2016). Multiple Imputation of Multilevel Missing Data. *SAGE Open*, 6(4). <https://doi.org/10.1177/2158244016668220>
- Hadeed, S. J., O'Rourke, M. K., Burgess, J. L., Harris, R. B., & Canales, R. A. (2020). Imputation methods for addressing missing data in short-term monitoring of air pollutants. *The Science Of The Total Environment*, 730, 139140.
<https://doi.org/10.1016/j.scitotenv.2020.139140>
- holger krekel and pytest-dev team. (2015). *pytest documentation*.
<https://docs.pytest.org/en/stable/>
- HPC Stipple SL. (2025). *Genie Framework - Build Data Apps in Julia*. Genie Framework - Build Data Apps In Julia. <https://genieframework.com/>
- Hunter, J. D. (2007). Matplotlib: a 2D Graphics environment. *Computing In Science & Engineering*, 9(3), 90-95. <https://doi.org/10.1109/mcse.2007.55>
- Indeed. (2025a). *Salario de Investigador/a asociado/a en España*. Recuperado 16 de octubre de 2025, de https://es.indeed.com/career/investigador-asociado/salaries?from=top_sb
- Indeed. (2025b). *Salario de Programador/a junior en España*. Recuperado 16 de octubre de 2025, de <https://es.indeed.com/career/programador-junior/salaries>

- Islam, S., Elmekki, H., Elsebai, A., Bentahar, J., Drawel, N., Rjoub, G., & Pedrycz, W. (2023). A Comprehensive Survey on Applications of Transformers for Deep Learning Tasks. *arXiv (Cornell University)*. <https://doi.org/10.48550/arxiv.2306.07303>
- James, G., Witten, D., Hastie, T., Tibshirani, R., & Taylor, J. (2023). Linear regression. En *Springer texts in statistics* (pp. 69-134). https://doi.org/10.1007/978-3-031-38747-0_3
- Jansen, P. (2022, 3 junio). *TIOBE Index - TIOBE*. TIOBE. <https://www.tiobe.com/tiobe-index/>
- JPype documentation — JPype 1.6.1.dev0 documentation.* (2018). <https://jpype.readthedocs.io/en/latest/>
- Junninen, H., Niska, H., Tuppurainen, K., Ruuskanen, J., & Kolehmainen, M. (2004). Methods for imputation of missing values in air quality data sets. *Atmospheric Environment*, 38(18), 2895-2907. <https://doi.org/10.1016/j.atmosenv.2004.02.026>
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436-444. <https://doi.org/10.1038/nature14539>
- Macaulay, L. A. (1996). Requirements Engineering. En *Applied computing*. <https://doi.org/10.1007/978-1-4471-1005-7>
- MATLAB - El lenguaje del cálculo técnico.* (2024). Mathworks.com. Recuperado 16 de octubre de 2025, de https://es.mathworks.com/products/matlab.html?s_tid=hp_products_matlab
- Mehare, H. B., Anilkumar, J. P., & Usmani, N. A. (2023). The Python programming language. En *Springer eBooks* (pp. 27-60). https://doi.org/10.1007/978-3-031-22206-1_2
- Microsoft. (2025). *Compra y descarga Windows 11 Home | Microsoft*. Microsoft Store. <https://www.microsoft.com/es-es/d/windows-11-home/dg7gmgf0krt0>
- Moritz, S., & Bartz-Beielstein, T. (2017, junio). imputeTS: Time Series Missing Value Imputation in R. *DigitalCommons@University Of Nebraska - Lincoln*. <https://digitalcommons.unl.edu/r-journal/453/>
- NumPy team. (2025). *NumPy*. <https://numpy.org/>
- Núñez, E., Steyerberg, E. W., & Núñez, J. (2011). Regression modeling strategies. *Revista Española de Cardiología (English Edition)*, 64(6), 501-507. <https://doi.org/10.1016/j.rec.2011.01.017>

- pandas development team. (2025a). *pandas - Python Data Analysis Library*.
<https://pandas.pydata.org/>
- pandas development team. (2025b). *Working with missing data — pandas 2.3.3 documentation*.
https://pandas.pydata.org/docs/user_guide/missing_data.html
- Peng, R. D. (2016). R Programming for Data Science [Pdf]. En *Leanpub*. Leanpub.
<http://leanpub.com/rprogramming>
- Plotly. (2025). *Plotly*. <https://plotly.com/r/>
- Python – The Fastest Growing Programming Language. (2017, diciembre). *International Research Journal Of Engineering And Technology (IRJET)*. Recuperado 16 de octubre de 2025, de <https://www.irjet.net/archives/V4/i12/IRJET-V4I1266.pdf>
- Python Software Foundation. (2001). *9. classes*. Python Documentation. Recuperado 16 de octubre de 2025, de <https://docs.python.org/3/tutorial/classes.html#private-variables>
- Ramasubramanian, K., & Singh, A. (2018). Machine Learning using R. En *Apress eBooks*.
<https://doi.org/10.1007/978-1-4842-4215-5>
- Raschka, S., Patterson, J., & Nolet, C. (2020). Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence. *Information*, 11(4), 193. <https://doi.org/10.3390/info11040193>
- Richman, M. B., Trafalis, T. B., & Adrianto, I. (2008). Missing data imputation through machine learning algorithms. En *Springer eBooks* (pp. 153-169).
https://doi.org/10.1007/978-1-4020-9119-3_7
- Riverbank Computing. (2025a). *Riverbank Computing | Buy PyQT*.
<https://riverbankcomputing.com/commercial/buy>
- Riverbank Computing. (2025b). *Riverbank Computing | Introduction*.
<https://www.riverbankcomputing.com/software/pyqt/>
- Riverbank Computing. (2025c). *Riverbank Computing | License FAQ*.
<https://riverbankcomputing.com/commercial/license-faq>
- Rubin, K. S. (2012). *Essential scrum: A Practical Guide to the Most Popular Agile Process*. Addison-Wesley Professional.

- SAS Institute Inc. (2025). *SAS: Soluciones de datos e inteligencia artificial*. SAS. https://www.sas.com/es_es/home.html
- SciPy Team. (2025). *SciPy*. <https://scipy.org/>
- Sensor:community KIT (SDS011/BME280), English language, harness cable edition*. (2025). <https://nettigo.eu/products/sensor-community-kit-sds011-bme280-english-language-harness-cable-edition>
- Srivastava, A., Bhardwaj, S., & Saraswat, S. (2017). SCRUM model for agile methodology. *2017 International Conference On Computing, Communication And Automation (ICCCA)*, 864-869. <https://doi.org/10.1109/ccaa.2017.8229928>
- StataCorp LLC. (2025). *Why use Stata?* <https://www.stata.com/why-use-stata/>
- Story, R. (2025). *Folium — Folium 0.20.0 documentation*. <https://python-visualization.github.io/folium/latest/index.html>
- Syromiatnikov, A., & Weyns, D. (2014). A Journey through the Land of Model-View-Design Patterns. *2014 IEEE/IFIP Conference On Software Architecture*, 21-30. <https://doi.org/10.1109/wicsa.2014.13>
- Takeuchi, H., & Nonaka, I. (1986). The new new product development game. *Journal Of Product Innovation Management*, 3(3), 205-206. [https://doi.org/10.1016/0737-6782\(86\)90053-6](https://doi.org/10.1016/0737-6782(86)90053-6)
- The CRAN team. (2025). *The comprehensive R Archive Network*. <https://cran.r-project.org/>
- Free Software Foundation, Inc. (2007, 29 junio). *The GNU General Public License v3.0 - GNU Project - Free Software Foundation*. gnu.org. Recuperado 16 de octubre de 2025, de <https://www.gnu.org/licenses/gpl-3.0.html#license-text>
- The Python Software Foundation. (2025). *Home*. Jython. <https://www.jython.org/>
- TIOBE Index - TIOBE. (2022, 3 junio). TIOBE. https://www.tiobe.com/tiobe-index/programminglanguages_definition/
- tkinter — Python interface to Tcl/Tk*. (2025). Python Documentation. Recuperado 16 de octubre de 2025, de <https://docs.python.org/3/library/tkinter.html>

Verzani, J. (2024, 7 diciembre). *Rewrite of gWidgets API for Simplified GUI Construction [R package gWidgets2 version 1.0-10]*.

<https://cran.r-project.org/web/packages/gWidgets2/index.html>

Wei, W. W. (2013). Time Series analysis. En *Oxford University Press eBooks*.

<https://doi.org/10.1093/oxfordhb/9780199934898.013.0022>

Wickham, H. (2021, abril). *Welcome | Mastering Shiny*. Mastering Shiny.

<https://mastering-shiny.org/>

Versión web del libro

Wilkinson, L. (2011). The Grammar of Graphics. En *Springer eBooks* (pp. 375-414).

https://doi.org/10.1007/978-3-642-21551-3_13