**UNIVERSIDAD POLITÉCNICA DE MADRID**
Escuela Técnica Superior de Ingeniería de Sistemas Informáticos


# Diseño e Implementación de una Herramienta Software para la Recopilación y el Procesamiento de Datos Heterogéneos orientada a la Extracción de Conocimiento en el Ámbito de las Enfermedades Raras

## PROYECTO FIN DE GRADO

**Mario Bravo Cuadro**

Grado en Ingeniería del Software

Madrid, 2025



UNIVERSIDAD POLITÉCNICA DE MADRID
Escuela Técnica Superior de Ingeniería de Sistemas Informáticos

**Grado en Ingeniería del Software**

# Diseño e Implementación de una Herramienta Software para la Recopilación y el Procesamiento de Datos Heterogéneos orientada a la Extracción de Conocimiento en el Ámbito de las Enfermedades Raras

# PROYECTO FIN DE GRADO

**Mario Bravo Cuadro**

Grado en Ingeniería del Software

Bajo la dirección de:
Dra. Silvia Alba Uribe Mayoral
Y
Dra. María del Carmen Sánchez González

Madrid, 2025

Título: Diseño e Implementación de una Herramienta Software para la Recopilación y el Procesamiento de Datos Heterogéneos orientada a la Extracción de Conocimiento en el Ámbito de las Enfermedades Raras.

Autores: Mario Bravo Cuadro y Emil Stelian Pintilie

Grado en: Ingeniería del Software

Dirección: Dra. Silvia Alba Uribe Mayoral Y Dra. María del Carmen Sánchez González

**UNIVERSIDAD POLITÉCNICA DE MADRID**
Escuela Técnica Superior de Ingeniería de Sistemas Informáticos


# Diseño e Implementación de una Herramienta Software para la Recopilación y el Procesamiento de Datos Heterogéneos orientada a la Extracción de Conocimiento en el Ámbito de las Enfermedades Raras

# PROYECTO FIN DE GRADO

**Emil Stelian Pintilie**

Grado en Ingeniería del Software

Bajo la dirección de:
Dr. Gustavo Adolfo Hernández Peñaloza
Y
Dra. Eva Bermejo Sánchez

Madrid, 2025


UNIVERSIDAD POLITÉCNICA DE MADRID
Escuela Técnica Superior de Ingeniería de Sistemas Informáticos

**Grado en Ingeniería del Software**

# Diseño e Implementación de una Herramienta Software para la Recopilación y el Procesamiento de Datos Heterogéneos orientada a la Extracción de Conocimiento en el Ámbito de las Enfermedades Raras

# PROYECTO FIN DE GRADO

**Emil Stelian Pintilie**

Grado en Ingeniería del Software

Bajo la dirección de:
Dr. Gustavo Adolfo Hernández Peñaloza
Y
Dra. Eva Bermejo Sánchez

Madrid, 2025

Título: Diseño e Implementación de una Herramienta Software para la Recopilación y el Procesamiento de Datos Heterogéneos orientada a la Extracción de Conocimiento en el Ámbito de las Enfermedades Raras.

Autores: Emil Stelian Pintilie y Mario Bravo Cuadro

Grado en: Ingeniería del Software

Dirección: El Dr. Gustavo Adolfo Hernández Peñaloza y la Dra. Eva Bermejo Sánchez

*Dedicatoria de Mario:*

*"A mis compañeros de la universidad;*

*Piyush, Johannes, Pablo, Jesús, Dani, Álex, Adri:*

*Gracias por hacer de la ETSISI un lugar tolerable."*

## Agradecimientos Mario:

En primer lugar, agradezco a mi padre y a mi madre, por enseñarme a trabajar y a luchar por lo que uno quiere.

En segundo lugar, agradezco a Carmen Banegas Gil, mi profesora del instituto IES Ciudad de Los Ángeles, que me introdujo en el mundo de la tecnología y me mostró lo que es disfrutar creando algo nuevo desde cero.

Por último, a la universidad pública, que, sin ella, el hecho de ser ingeniero sólo sería un sueño para muchos de nosotros.

## Agradecimientos conjuntos de Emil y Mario

En primer lugar, queremos agradecer a las investigadoras, María del Carmen Sánchez González y Eva Bermejo Sánchez, su predisposición y constante apoyo a lo largo del proyecto. Su cercanía y atención nos han facilitado mucho el trabajo y nos ha permitido también disfrutar del proyecto al máximo.

En segundo lugar, queremos expresar nuestro agradecimiento a nuestros tutores, Silvia Alba Uribe Mayoral y Gustavo Adolfo Hernández Peñaloza, por la oportunidad de llevar a cabo este proyecto y por guiarnos en el proceso. Su apoyo siempre nos ha dejado la puerta abierta para compartir nuestras distintas ideas y para encontrar la mejor manera de llevarlas a cabo.

*Dedicatorias para el lector de Emil:*

*"Donde tus talentos y las necesidades del mundo se cruzan, ahí está tu vocación."*

*-Aristóteles*

*"La tecnología no es ni buena ni mala en sí misma, depende de cómo se utilice. Debemos utilizar la tecnología para empoderar a las personas y no para controlarlas."*

*-Richard Matthew Stallman*

**Agradecimientos de Emil**

Agradezco profundamente a mi familia y a todas las personas que me han acompañado y permitido crecer durante esta etapa corta, pero significativa, de mi vida.

También quiero expresar mi gratitud a todas las personas que he conocido y con las que he tenido el privilegio de trabajar en la ETSISI. Su compañía, apoyo y colaboración han dejado una huella importante en mi recorrido.

Por último, agradezco a la comunidad Open Source por haberme brindado la oportunidad de aprender, compartir y aportar en los diversos temas que abarca el apasionante campo de la informática.

**Abstract**

The objective of this project is to develop a biomedical knowledge extraction system that facilitates the process of obtaining relevant information for rare disease research. The system is designed to address the common challenges faced by biomedical researchers when accessing and processing information distributed across multiple heterogeneous data sources.

The project focuses on developing a platform that provides key information in a structured and efficient manner, significantly reducing the time for extracting critical information from weeks to minutes. This allows researchers to fully invest their efforts in analysis and decision-making based on the obtained information, thus optimizing biomedical research processes.

Although the system has been developed in specific collaboration with representatives from the Institute for Rare Disease Research (IIER) associated with the Instituto de Salud Carlos III, its modular and extensible architecture allows its application in other biomedical research contexts. The project has been developed in a real working environment, where the obtained requirements, as well as the product validation and the different prototypes built have been elaborated under the context of a concrete need from a real client. This characteristic translates into more detailed attention to the different development phases and the obligation to put into practice transversal competencies that go beyond the usual objectives of a Final Degree Project, such as effective communication, time management, and work in multidisciplinary teams.

The specific need addressed by the system lies in the widespread difficulty of obtaining useful information for decision-making due to the heterogeneous nature of biomedical data and its irregular distribution among different specialized data sources.

The system is developed in Python and has an API that allows generating predefined and modifiable reports. The platform internally and programmatically accesses different biomedical data sources, transforming the received data to give them a clear and visual format: it combines information from different sources to generate a structured final report that facilitates research work.

**Keywords:** bioinformatics, rare diseases, knowledge extraction, heterogeneous data processing, biomedical databases.

**Resumen**

El objetivo de este proyecto es desarrollar un sistema de extracción de conocimiento biomédico que facilite el proceso de obtención de información relevante para la investigación en enfermedades raras. El sistema está diseñado para abordar los desafíos comunes que enfrentan los investigadores biomédicos al acceder y procesar información distribuida en múltiples fuentes de datos heterogéneas.

El proyecto se centra en el desarrollo de una plataforma que proporciona información clave de manera estructurada y eficiente, reduciendo significativamente los tiempos de extracción de información determinante de semanas a minutos. Esto permite a los investigadores invertir plenamente su esfuerzo en el análisis y toma de decisiones basadas en la información obtenida, optimizando así los procesos de investigación biomédica.

Aunque el sistema ha sido desarrollado en colaboración específica con representantes del Instituto de Investigación de Enfermedades Raras (IIER) asociado al Instituto de Salud Carlos III, su arquitectura modular y extensible permite su aplicación en otros contextos de investigación biomédica. El proyecto se ha desarrollado en un entorno de trabajo real, donde los requisitos obtenidos, así como la validación del producto y los distintos prototipos construidos se han elaborado bajo el contexto de una necesidad concreta de un cliente real. Esta característica se traduce en una atención más detallada a las distintas fases de desarrollo y a la obligación de poner en práctica competencias transversales que van más allá de los objetivos habituales de un Trabajo de Fin de Grado, como la comunicación efectiva, la gestión de tiempos y el trabajo en equipos multidisciplinares.

La necesidad específica que aborda el sistema reside en la dificultad generalizada para obtener información útil para la toma de decisiones debido al carácter heterogéneo de los datos biomédicos y su irregular distribución entre distintas fuentes de datos especializadas.

El sistema está desarrollado en Python y dispone de una API que permite generar informes predefinidos y modificables. La plataforma accede internamente y de forma programática a las distintas fuentes de datos biomédicos, transformando los datos recibidos para darles un formato claro y visual: combina la información de distintas fuentes para generar un informe final estructurado que facilite el trabajo de investigación.

**Palabras clave:** bioinformática, enfermedades raras, extracción de conocimiento, procesamiento de datos heterogéneos, bases de datos biomédicas.

# Tabla de Contenido

# Lista de Figuras

**Abreviaturas y Acrónimos**

| | |
|---|---|
| UPM | Universidad Politécnica de Madrid |
| IIER | Instituto de Investigación de Enfermedades Raras |
| ISCIII | Instituto de Salud Carlos III |
| HTTP | Hypertext Transfer Protocol |
| API | Application Programming Interface |
| IDG | Illuminating the Druggable Genome |
| TCRD | Target Central Resource Database |
| NIH | National Institutes of Health |
| FEDER | Federación Española de Enfermedades Raras |
| RDF | Resource Description Framework |
| JSON | JavaScript Object Notation |
| GATV | Grupo de Aplicación de Telecomunicaciones Visuales |
| UI | User Interface |
| MVC | Modelo-Vista-Controlador |

# 1. Introducción

Una enfermedad rara es aquella que afecta a un número reducido de personas en comparación con la población general. En la Unión Europea, se considera rara a una enfermedad que afecta a menos de 5 personas por cada 10.000 habitantes. Estas enfermedades suelen ser crónicas, progresivas y, en muchos casos, de origen genético. Se estima que existen entre 7.000 y 8.000 enfermedades raras, que en conjunto afectan aproximadamente al 6-8% de la población. (1)

Las enfermedades raras, a menudo, no se atienden con la severidad que merecen por su aparición escasa y su difícil diagnóstico y tratamiento: al ser los casos registrados tan escasos, las enfermedades raras son en gran medida, desconocidas.

La información que se tiene sobre ellas es ínfima, lo que hace que su investigación sea lenta y dificultosa. Según la Federación Española de Enfermedades Raras (FEDER) (2), el tiempo promedio para obtener un diagnóstico en España es superior a 4 años, llegando en un 20% de los casos a más de una década. (3)

Además de la escasez de datos, el acceso a ellos tiene una carga manual muy pronunciada, ralentizando así aún más los procesos de investigación y diagnóstico de estas enfermedades; los datos rara vez se encuentran estandarizados, y no existe un repositorio centralizado que contenga toda la información necesaria para que los investigadores de enfermedades raras puedan cumplir con sus labores.

De la combinación de estas características se deriva que uno de los primeros puntos de mejora en los procesos de investigación de enfermedades raras reside en la manera en la que los datos clave se recogen. Citando a SpringerLink: la capacidad de combinar datos heterogéneos distribuidos en todo el mundo es críticamente importante para impulsar la investigación sobre enfermedades raras, pero presenta una serie de desafíos metodológicos, representacionales y de automatización:

"The ability to combine heterogeneous data distributed across the globe is critically important to boost research on rare diseases, but it presents a number of methodological, representational and automation challenges. In this scenario, biomedical ontologies are of critical importance for enabling computers to aid in information retrieval and analysis across data collections." (4)

Por tanto, se identifica la necesidad de desarrollar una solución que aborde estos desafíos mediante la automatización de la recolección de datos y la integración de fuentes biomédicas heterogéneas, con el objetivo de reducir considerablemente el tiempo invertido en la obtención de conocimiento a través de accesos programáticos estructurados.

## 1.1. Contexto general

El Instituto de Investigación de Enfermedades Raras (IIER) (5) es una institución española dedicada al estudio e investigación de enfermedades raras.

El IIER forma parte del Sistema Nacional de Salud y su objetivo principal es mejorar el diagnóstico, tratamiento y calidad de vida de los pacientes que padecen estas enfermedades.

Con el objetivo de establecer un canal de colaboración entre la UPM y el IIER, se dio cabida a este proyecto en el que se pretende la mejora en las investigaciones y estudios llevados a cabo por el segundo mediante la implementación, por parte del primero, de un producto software personalizados que atiendan a sus necesidades concretas.

Una de estas necesidades está relacionada con la dificultad que los investigadores asumen a la hora de encontrar información relevante relacionada con enfermedades raras a investigar. Es en este punto donde el sistema desarrollado en este TFG se ofrece como una solución, que se describirá en detalle en los siguientes capítulos.

Es importante destacar que una de las necesidades fundamentales del proyecto, dada la naturaleza cambiante del ecosistema de datos biomédicos, es la de crear un sistema que sea lo suficientemente flexible para tolerar cambios en las fuentes y la estructura de los datos ya que los requisitos del sistema cambian al ritmo que aparecen nuevas fuentes de datos de interés; la modularidad y extensibilidad del sistema son dos de los retos más importantes que se presentan en el desarrollo de esta solución.

## 1.2. Contexto del cliente

Para una explicación más concisa sobre el desarrollo de este proyecto, el cliente final para el que se crea este sistema facilita la siguiente información y contexto sobre el proyecto:

El programa SpainUDP (https://spainudp.isciii.es/ ) es un programa desarrollado en el Instituto de Investigación de Enfermedades Raras (IIER) del Instituto de Salud Carlos III (ISCIII) en Madrid. Su principal objetivo es la búsqueda del diagnóstico de aquellas personas en las que se sospecha una enfermedad rara (enfermedad que ocurre en menos de 5 personas de cada 10.000) o ultrarrara (menos de 1 de cada 50.000), que, tras años de búsqueda en el sistema sanitario, todavía no ha sido determinada, por lo que es preciso pasar del plano asistencial al plano de la investigación para tratar de encontrar ese diagnóstico (que a su vez será la puerta al tratamiento). De este modo, SpainUDP se alinea con los objetivos del consorcio internacional IRDiRC (International Rare Diseases Research Consortium), que establece los objetivos que, a nivel mundial, deben guiar toda investigación sobre enfermedades raras.

El programa aborda diferentes vías innovadoras y colaborativas de estudio e investigación, a nivel nacional e internacional. Tras completar varias fases de estudio de cada paciente, si se encuentra una variante en un gen con potencial efecto patogénico, se emite un informe indicando estos datos. La información adicional sobre el gen afectado se recupera a través de múltiples fuentes y recursos de internet.

Con los objetivos de: (i) sistematizar y dar robustez a la búsqueda de esta información, (ii) armonizar la secuencia de extracción de datos, (iii) acortar el tiempo de recuperación de los datos más relevantes (p.ej., clase, expresión, vías afectadas, interacciones, potenciales fármacos o moléculas con actividad asociada) y (iv) homogeneizar y complementar los informes emitidos por el programa para los clínicos y para el propio programa, se establece una colaboración con el Departamento de Sistemas Informáticos de la Universidad Politécnica de Madrid.

A través de esta colaboración científica se construye una herramienta que facilita la creación de estos informes complementarios, actualizables y escalables, a partir de la extracción automática de múltiples fuentes de información.

## 1.3. Objetivos

El objetivo principal del proyecto es desarrollar un sistema que sea capaz de obtener información sobre distintos elementos de la biomedicina obtenida de distintas fuentes y organizar esta información en un formato ordenado y visual, atendiendo a los requerimientos especificados por los profesionales del IIER. Esta solución debe atender a la posibilidad de que las fuentes consultadas dejen de ser accesibles y a la posibilidad de incluir nuevas fuentes de información a lo largo del tiempo.

Para lograr este objetivo principal se identifican los siguientes objetivos específicos:

- **Identificar y analizar las necesidades de investigación**: Colaborar con el grupo de investigación del IIER para determinar los requisitos específicos del sistema y las funcionalidades necesarias para apoyar la investigación en enfermedades raras.
- **Evaluar y caracterizar fuentes de datos biomédicos**: Investigar y catalogar las fuentes de datos biomédicos disponibles, analizando su relevancia, accesibilidad y calidad de información en el contexto de enfermedades raras.
- **Desarrollar un sistema de integración de datos biomédicos**: Crear una plataforma capaz de obtener, procesar y combinar información de múltiples fuentes heterogéneas de datos biomédicos de forma automatizada.
- **Implementar una arquitectura robusta y escalable**: Diseñar un sistema con capacidades de tolerancia a fallos, modularidad y extensibilidad que permita su evolución y mantenimiento a largo plazo.
- **Facilitar el acceso a información biomédica compleja**: Desarrollar interfaces y mecanismos que permitan a investigadores no técnicos acceder y utilizar eficientemente la información integrada.
- **Crear una solución interoperable**: Desarrollar un backend que pueda integrarse fácilmente con interfaces gráficas externas y otros sistemas de análisis biomédico.

- **Garantizar el cumplimiento normativo**: Asegurar que la solución respete las políticas de uso de datos y la legislación aplicable en el manejo de información biomédica.

- **Validar la utilidad práctica del sistema**: Demostrar la efectividad de la solución mediante casos de uso reales y la validación con usuarios finales del dominio de investigación.

## 1.4. Estructura del documento

Este documento se estructura en diez capítulos principales, cada uno de los cuales aborda un aspecto clave del desarrollo del proyecto:

- **Capítulo 1 – Introducción**

  Presenta el contexto general del problema, el cliente, los objetivos del proyecto y la estructura del documento.

- **Capítulo 2 – Estado de la cuestión**

  Analiza las herramientas existentes en el ámbito de la extracción de conocimiento biomédico, identifica sus limitaciones y justifica la necesidad de la solución propuesta.

- **Capítulo 3 – Metodología de desarrollo**

  Describe las metodologías tradicionales y ágiles consideradas, así como el enfoque híbrido adoptado para el desarrollo del sistema.

- **Capítulo 4 – Diseño de la solución**

  Expone el proceso de elicitación de requisitos, las distintas arquitecturas evaluadas y la solución final basada en principios SOLID.

- **Capítulo 5 – Implementación de la solución**

  Detalla la implementación técnica de la API y la interfaz gráfica, incluyendo los endpoints, flujos de trabajo y mecanismos de interacción.

- **Capítulo 6 – Validación y verificación**

  Explica las pruebas realizadas para asegurar el correcto funcionamiento del sistema y la validación por parte del cliente.

- **Capítulo 7 – Consideraciones ambientales, sociales y éticas**

Reflexiona sobre el impacto del proyecto en términos de legalidad, accesibilidad, sostenibilidad y contribución a los ODS.

- **Capítulo 8 – Conclusiones**

  Resume los logros alcanzados, las competencias adquiridas y la valoración final del proyecto.

- **Capítulo 9 – Referencias**

  Recoge todas las fuentes bibliográficas y documentales utilizadas a lo largo del trabajo.

- **Capítulo 10 – Anexos**

  Incluye documentación técnica complementaria como guías de desarrollo y el inventario de extracción de información.

# 2. Estado de la cuestión

## 2.1. Aplicaciones y fuentes relacionadas

A continuación, se van a describir brevemente algunas aplicaciones y fuentes de información relacionadas con la extracción de conocimiento en el campo de la biomedicina. Se evaluará su impacto, así como sus pros y sus contras para desarrollar el proyecto con toda la información posible.

### 2.1.1. Pharos

Pharos es una aplicación web interactiva desarrollada por el programa Illuminating the Druggable Genome (IDG) de los Institutos Nacionales de Salud (NIH) de Estados Unidos. Esta plataforma proporciona acceso integrado a información sobre targets farmacológicos, con especial énfasis en proteínas poco estudiadas que podrían representar nuevas oportunidades terapéuticas. Pharos integra datos de múltiples fuentes incluyendo UniProt, ChEMBL, DrugCentral, y TCRD (Target Central Resource Database), organizando la información en diferentes niveles de conocimiento: Tclin (targets con fármacos clínicos aprobados), Tchem (targets con actividad química conocida), Tbio (targets con actividad biológica conocida), y Tdark (targets con conocimiento limitado). (6)

Por su interfaz fácil de utilizar y su eficiente extracción de información que presenta, usamos esta aplicación no solo como referencia para crear nuestro sistema sino también como fuente de información integrada en el proyecto.

Sin embargo, se han identificado limitaciones significativas en el uso de Pharos como herramienta principal de investigación. La plataforma presenta problemas recurrentes de disponibilidad y estabilidad, experimentando caídas frecuentes del servicio que interrumpen los flujos de trabajo de investigación y comprometen la fiabilidad del acceso a los datos. Además, aunque Pharos proporciona información valiosa sobre targets farmacológicos, no incluye toda la información especializada necesaria para la investigación comprehensiva de enfermedades raras que requieren las investigadoras, como datos específicos de variantes genéticas, fenotipos clínicos detallados, o información sobre la prevalencia y manifestaciones específicas de estas condiciones poco frecuentes.

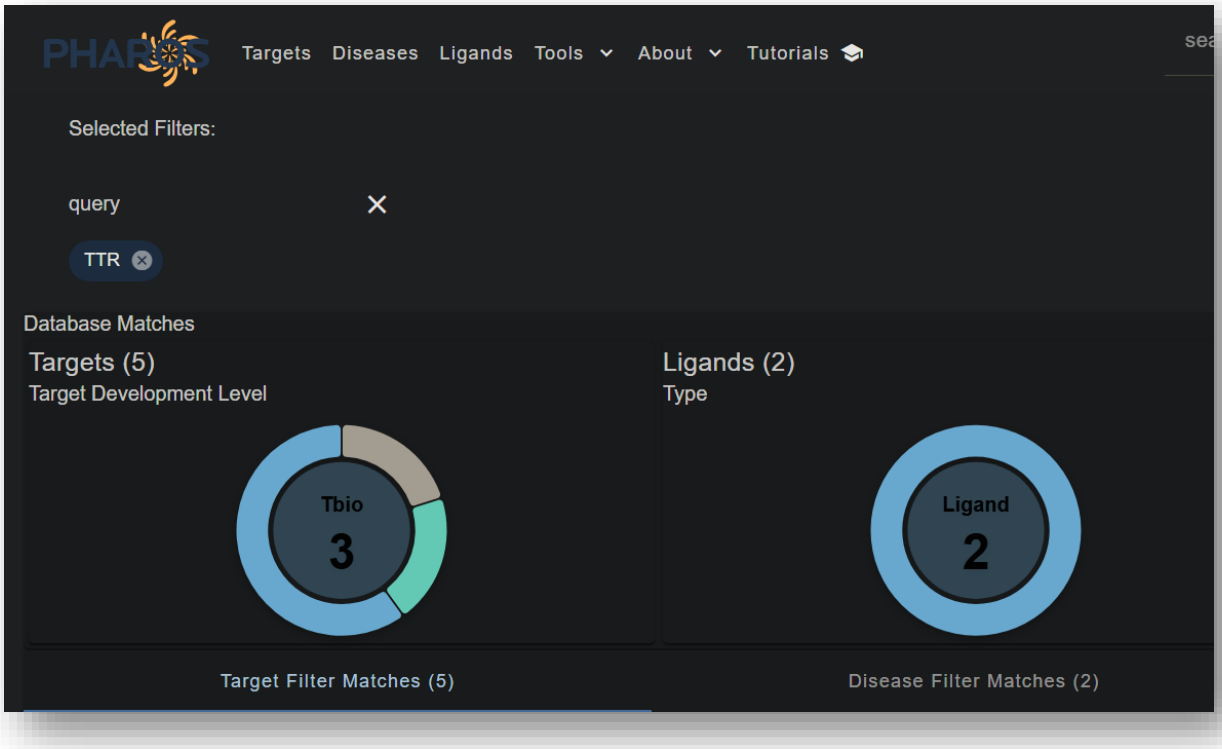



Figura 1. Ejemplo de búsqueda de un gen en Pharos. Fuente: https://pharos.nih.gov/

### 2.1.2. PandaOmics

PandaOmics es una plataforma de inteligencia artificial desarrollada por Insilico Medicine que revoluciona el descubrimiento de fármacos mediante el análisis integrado de datos ómicos multimodales. Esta aplicación web utiliza algoritmos de aprendizaje automático avanzados y redes neuronales profundas para identificar y priorizar targets farmacológicos, predecir asociaciones target-enfermedad, y generar hipótesis de investigación basadas en la integración de datos genómicos, transcriptómicos, proteómicos y fenotípicos. PandaOmics se distingue por su capacidad para procesar grandes volúmenes de datos biológicos heterogéneos y proporcionar insights accionables para el desarrollo de terapias, especialmente en áreas de necesidad médica no satisfecha como las enfermedades raras y el envejecimiento. (7)

Aunque nuestro sistema no hace análisis con inteligencia artificial, PandaOmics ha servido como referencia para el diseño de nuestra arquitectura de integración de datos y la conceptualización de workflows de análisis biomédico. Su enfoque de integración de múltiples fuentes de datos biomédicas y la presentación de resultados estructurados han inspirado nuestro sistema de proveedores de datos y la arquitectura de orquestación de workflows.

A su vez, adoptamos principios similares de integración de datos de UniProt, OpenTargets, Pharos y otras fuentes especializadas, proporcionando una interfaz unificada para el acceso a información biomédica compleja.

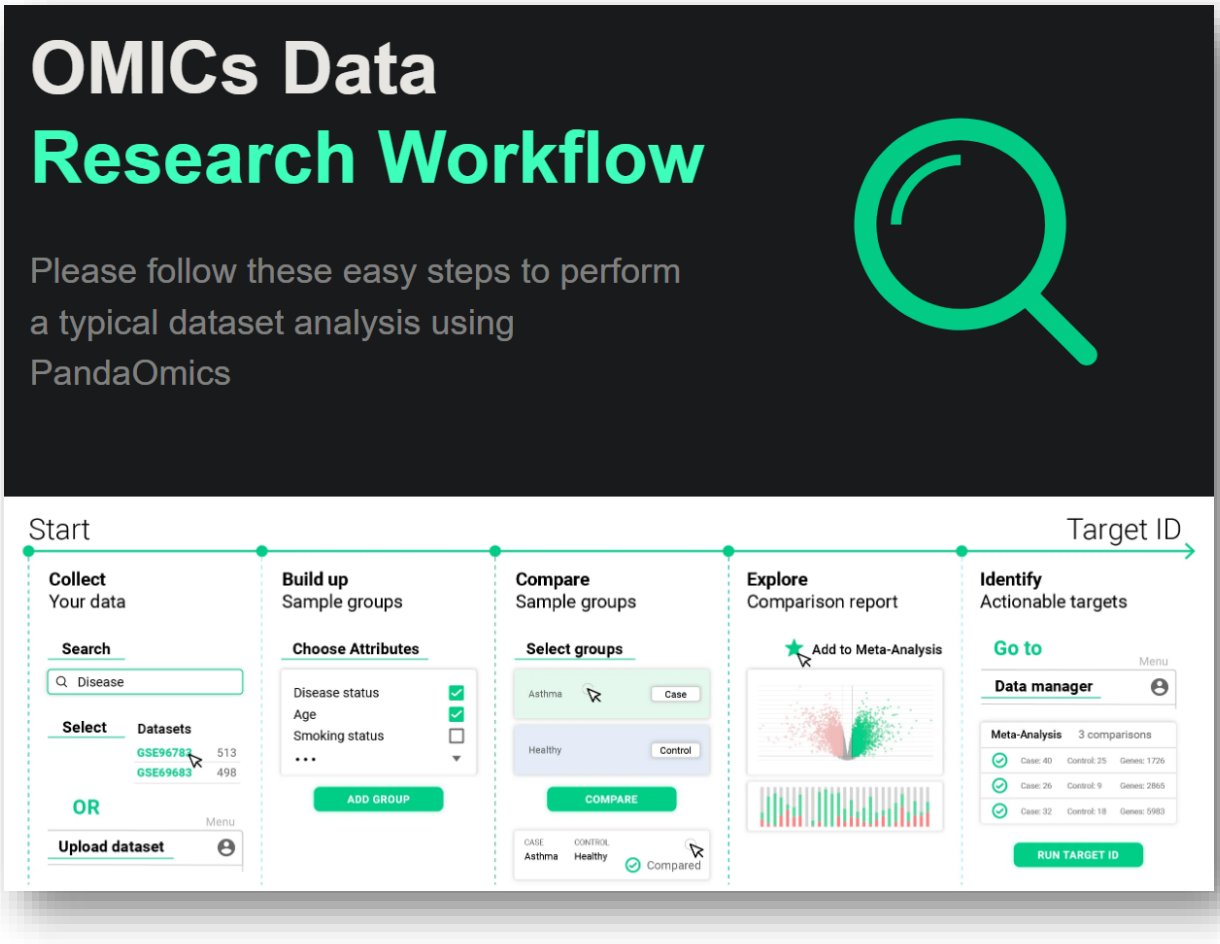



Figura 2. Ejemplo de Workflow en PandaOmics. Fuente: https://pharma.ai/pandaomics/help/workflow

No obstante, PandaOmics presenta una limitación fundamental para la comunidad investigadora: se trata de una plataforma comercial de pago que requiere suscripciones costosas, lo que la hace inaccesible para muchos investigadores académicos y grupos de investigación con recursos limitados. Esta barrera económica impide que la comunidad científica pueda aprovechar plenamente sus capacidades, especialmente en el contexto de la investigación en enfermedades raras, donde los recursos suelen ser escasos. Por tanto, se adoptaron principios similares de integración de datos de UniProt, OpenTargets, Pharos y otras fuentes especializadas, pero desarrollando una interfaz unificada de código abierto y acceso gratuito para la información biomédica compleja.

### 2.1.3. Uniprot

UniProt es una base de datos de proteínas de acceso libre y ampliamente reconocida que proporciona información completa y de alta calidad sobre secuencias y funciones de proteínas. Desarrollada por una colaboración entre el Instituto Europeo de Bioinformática (EMBL-EBI) (8), el Instituto SIB de Bioinformática Suizo (9) y el Protein Information Resource (PIR) (10), UniProt integra datos de múltiples fuentes y proporciona anotaciones detalladas sobre estructura, función, localización subcelular, variantes genéticas, enfermedades asociadas e interacciones proteicas. La base de datos se organiza en secciones claramente estructuradas que incluyen información general, secuencias, anotaciones funcionales, patología y referencias bibliográficas, con enlaces cruzados a más de 150 bases de datos especializadas. UniProt se compone de tres componentes principales: UniProtKB (base de conocimiento), UniRef (clusters de secuencias) y UniParc (archivo de secuencias). (11)

Una de las principales fortalezas de UniProt es su naturaleza gratuita y de acceso abierto, lo que la convierte en un recurso fundamental para la comunidad investigadora mundial. La plataforma proporciona información meticulosamente curada y estructurada por secciones temáticas, cada una respaldada por referencias bibliográficas de alta calidad que permiten la verificación y profundización en los datos presentados. Esta organización sistemática facilita la navegación y comprensión de la información proteica compleja, mientras que su API REST permite el acceso programático eficiente para aplicaciones bioinformáticas.

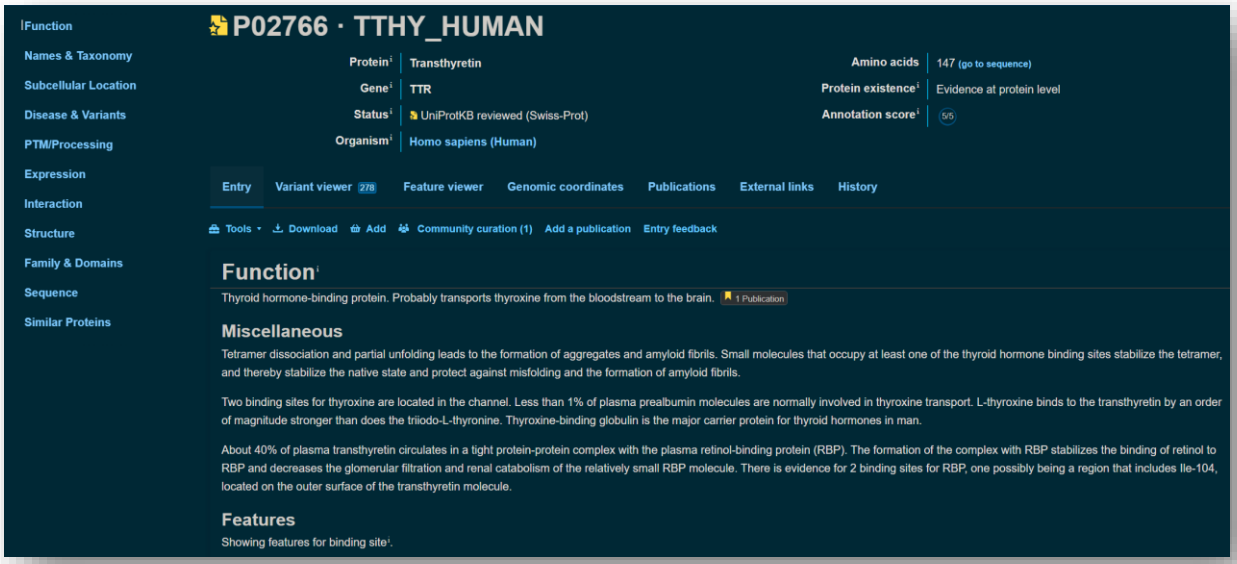



Figura 3. Ejemplo de resultado de búsqueda en UniProt para la proteína Transtirretina. Fuente: https://www.uniprot.org/uniprotkb/P02766/entry

Sin embargo, para el contexto específico de la investigación en enfermedades raras, UniProt presenta limitaciones importantes. Aunque contiene información valiosa sobre proteínas asociadas a estas condiciones, no incluye toda la información especializada necesaria para la investigación comprehensiva de enfermedades raras que requieren las investigadoras, como datos integrados sobre prevalencia de enfermedades, tratamientos específicos disponibles, información farmacológica detallada sobre compuestos terapéuticos, o análisis de pathways específicos relacionados con la patogénesis. Además, el acceso a la información requiere navegación manual a través de la interfaz web para consultas exploratorias complejas, lo que resulta en un proceso laborioso y poco eficiente cuando se necesita consultar múltiples proteínas o realizar análisis comparativos extensos, limitando significativamente la productividad en proyectos de investigación que requieren procesamiento de grandes volúmenes de datos interrelacionados.

## 2.2. Análisis comparativo y carencias identificadas

Del análisis de las herramientas existentes se identifican las siguientes carencias:

1. **Falta de especialización en enfermedades raras**: Las plataformas actuales abordan el campo biomédico de forma general

2. **Acceso limitado a datos integrados**: No existe una solución de código abierto que integre múltiples fuentes especializadas
3. **Dependencia de acceso manual**: La mayoría requiere navegación manual por interfaces web
4. **Falta de workflows personalizables**: Las herramientas no permiten crear flujos de trabajo adaptados a necesidades específicas de investigación

## 2.3. Tecnologías de implementación utilizadas

Tras hacer un análisis de los productos presentes actualmente en el sector y evaluar las distintas características de cada uno, se explica a continuación las tecnologías utilizadas para el desarrollo del proyecto.

### 2.3.1. HTTP

HTTP (Hypertext Transfer Protocol) es un protocolo de aplicación para sistemas de información distribuidos, colaborativos e hipermedia. Es la base de la comunicación de datos en la World Wide Web, donde los documentos de hipertexto incluyen hipervínculos a otros recursos a los que el usuario puede acceder fácilmente. (12)

En este proyecto, HTTP sirve como el protocolo fundamental sobre el que se construyen todas las comunicaciones con servicios externos, APIs bioinformáticas y entre los componentes internos del sistema, como la comunicación entre el front y el backend. La gestión eficiente de solicitudes y respuestas HTTP es crucial para optimizar el rendimiento y la escalabilidad del sistema. La importancia de este protocolo se observa en el hecho de que nuestro sistema actúa tanto como de servidor (emisor de información) como cliente (receptor de información).

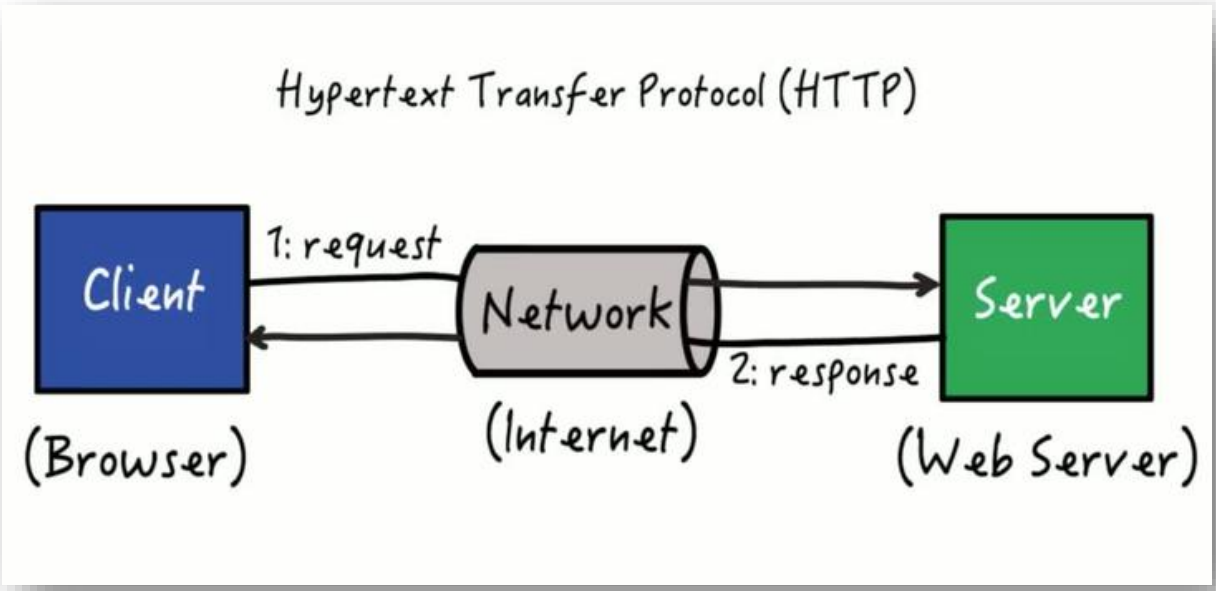



Figura 4. Representación explicativa del protocolo HTTP. Fuente: https://researchhubs.com/post/computing/web-application/the-hypertext-transfer-protocol-http.html

### 2.3.2. GraphQL

GraphQL es un lenguaje de consulta y manipulación de datos para APIs, así como un entorno para ejecutar estas consultas con datos existentes. Proporciona una descripción completa y comprensible de los datos en la API, otorga a los clientes el poder de solicitar exactamente lo que necesitan, facilita la evolución de las APIs a lo largo del tiempo y habilita potentes herramientas de desarrollo. (13)

En nuestro caso, utilizamos GraphQL para optimizar la recuperación de datos complejos e interrelacionados de fuentes biológicas, permitiendo consultas específicas que recuperan exactamente la información necesaria en cada caso de uso, reduciendo la sobrecarga de transferencia de datos y mejorando la eficiencia del sistema cuando se trabaja con grandes conjuntos de datos biológicos. Algunas fuentes de datos como Pharos y OpenTargets exigen el conocimiento de esta tecnología para el acceso programático a sus datos.

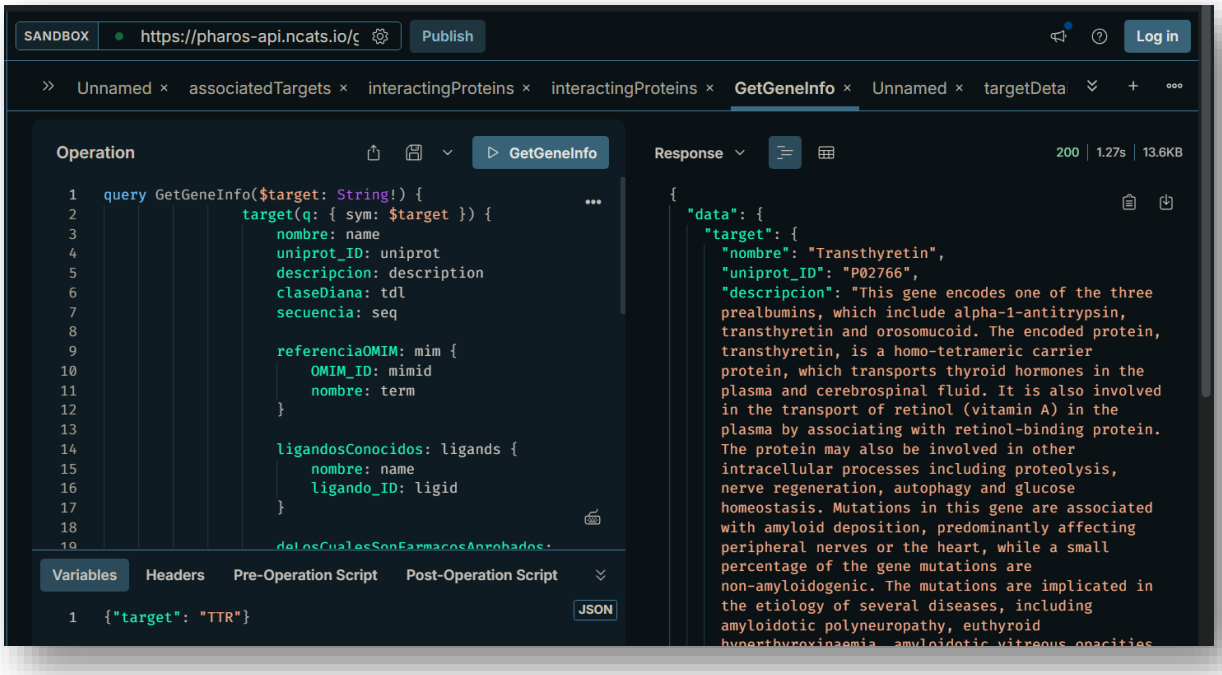



Figura 5. Ejemplo de uso de GraphQL para el acceso programático a Pharos.

### 2.3.3. Git

Git es un sistema de control de versiones distribuido gratuito y de código abierto diseñado para manejar todo desde proyectos pequeños hasta muy grandes con velocidad y eficiencia. Fue creado por Linus Torvalds en 2005 para el desarrollo del kernel de Linux. Git permite a los desarrolladores rastrear cambios en archivos y coordinar el trabajo en esos archivos entre múltiples personas, manteniendo un historial completo de todas las modificaciones realizadas en el proyecto. (14)

En nuestro proyecto, Git es fundamental para el control de versiones del código fuente, permitiendo la colaboración eficiente entre desarrolladores, el seguimiento de cambios en los diferentes componentes del sistema y la gestión de ramas para el desarrollo de nuevas funcionalidades. La estructura distribuida de Git facilita el trabajo en paralelo en diferentes módulos del sistema, manteniendo la integridad del código y permitiendo la reversión de cambios cuando sea necesario. Lo utilizamos junto a los repositorios en remoto de GitHub que se explicarán más adelante.

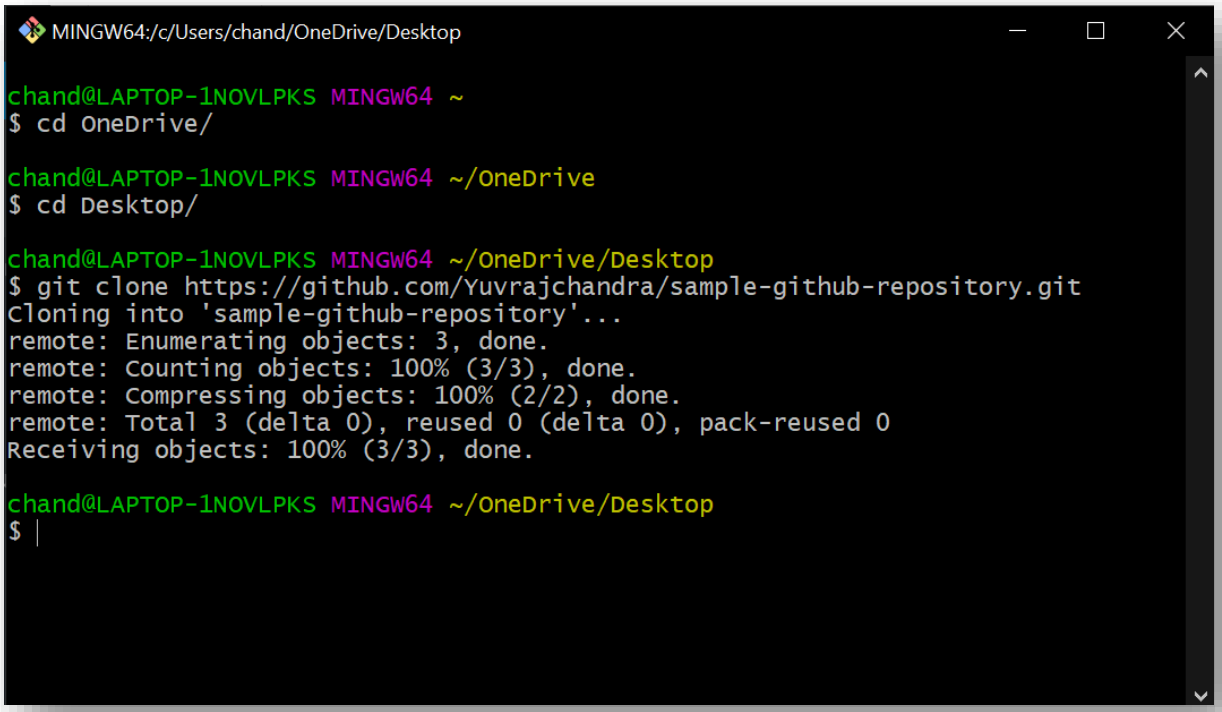



Figura 6. Representación de la línea de comandos de Git: Git Bash.

### 2.3.4. Python

Python es un lenguaje de programación interpretado, de alto nivel y de propósito general, creado por Guido van Rossum y lanzado por primera vez en 1991. (15) Su filosofía de diseño enfatiza la legibilidad del código con su notable uso de espacios en blanco significativos. Python se caracteriza por su sintaxis clara y expresiva que permite escribir código conciso y legible, lo que reduce el costo de mantenimiento del programa. Soporta múltiples paradigmas de programación, incluyendo programación orientada a objetos, imperativa, funcional y procedural. Python cuenta con un sistema dinámico de tipos y gestión automática de memoria, siendo adecuado para desarrollo rápido de aplicaciones y como lenguaje de scripting para conectar componentes existentes.

Dentro del lenguaje de Python, se destaca el uso de algunos elementos que se explica a continuación.

### 2.3.4.1. Pandas

Pandas es una biblioteca de software libre para el lenguaje de programación Python que proporciona estructuras de datos y herramientas de análisis de datos de alto rendimiento y fáciles de usar. En particular, ofrece estructuras de datos y operaciones para manipular tablas numéricas y series temporales. Pandas es fundamental para la ciencia de datos, permitiendo importar datos de varios formatos de archivo, manipular, transformar y analizar datos de manera eficiente. (16)

En nuestro sistema, Pandas se utiliza para procesar los datos obtenidos de diversas fuentes biológicas, estructurar la información en DataFrames, y realizar análisis cruzados que permiten identificar patrones y relaciones entre genes, proteínas y enfermedades raras.

In [2]:
```
# The convention is to import Pandas with shortcut 'pd'
import pandas as pd
import os
```

In [25]:
```
# You can create a database using a dictionary of lists.
# Each column is a dictionary key and the key becomes the column name.
# All the lists need to be the same length, and these become the rows.
new_dataframe = pd.DataFrame(
    {
        "column_1": [1,2,3,4,5],
        "another_column": ['this', 'column', 'has', 'strings', 'inside!'],
        "float_column": [0.1, 0.5, 33, 48, 42.5558],
        "binary_solo": [True, False, True, True, False]
    }
)
# you can look at your new dataframe in Jupyter, by simply typing it's name:
new_dataframe
```

Out[25]:

| | another_column | binary_solo | column_1 | float_column |
|---|---|---|---|---|
| 0 | this | True | 1 | 0.1000 |
| 1 | column | False | 2 | 0.5000 |
| 2 | has | True | 3 | 33.0000 |
| 3 | strings | True | 4 | 48.0000 |
| 4 | inside! | False | 5 | 42.5558 |

Figura 7. Representación de ejemplo de un Pandas Dataframe

### 2.3.4.2. Requests

Requests es una biblioteca HTTP para Python que permite enviar solicitudes HTTP/1.1 de manera extremadamente sencilla. Esta biblioteca abstrae la complejidad de realizar solicitudes HTTP, proporcionando una API intuitiva para acceder a servicios web y APIs REST. (17)

En nuestro proyecto, Requests juega un papel crucial al permitir la comunicación con diversas APIs bioinformáticas como UniProt, Pharos, OpenTargets y Ensembl, facilitando la obtención de datos actualizados sobre genes, proteínas y sus relaciones con enfermedades raras.

```
import requests

#the required first parameter of the 'get' method
is the 'url':
x =
requests.get('https://w3schools.com/python/demopa
ge.html')

#print the response text (the content of the
requested file):
print(x.text)
```

```
<!DOCTYPE html>
<html>
<body>

<h1>This is a Test Page</h1>

</body>
</html>
```

Figura 8. Representación de ejemplo de una petición HTTP Get con la librería Requests

### 2.3.4.3. Flask

Flask es un micro framework web escrito en Python y basado en la biblioteca WSGI Werkzeug y la biblioteca Jinja2. Es clasificado como un microframework porque no requiere herramientas o bibliotecas particulares, manteniendo un núcleo simple pero extensible mediante diversas extensiones. (18)

En nuestro sistema, Flask proporciona la infraestructura para el servidor web que expone las APIs RESTful, permitiendo que la aplicación frontend se comunique con los diferentes procesadores de datos biológicos de manera eficiente y estructurada, facilitando la orquestación de flujos de trabajo complejos.

#### 2.3.4.4. BeautifulSoup

BeautifulSoup es una biblioteca de Python para extraer datos de archivos HTML y XML. Proporciona métodos idiomáticos para navegar, buscar y modificar árboles de análisis, lo que la convierte en una herramienta valiosa para web scraping y análisis de contenido. (19)

En el proyecto, BeautifulSoup se utiliza principalmente para extraer información de recursos web como SelleckChem, donde se obtienen datos sobre compuestos químicos y medicamentos relacionados con las proteínas y genes de interés, enriqueciendo la información disponible para el análisis de posibles tratamientos.

#### 2.3.4.5. Selenium

Selenium es un framework portable para testear aplicaciones web. Proporciona una herramienta de reproducción para crear pruebas funcionales sin necesidad de aprender un lenguaje específico de scripting de pruebas. También proporciona una API que permite escribir scripts de prueba en varios lenguajes de programación como Java, C#, Python, etc. (20)

Usamos Selenium para la automatización de navegadores web, facilitando la extracción de datos de sitios que requieren interacción dinámica, como portales de información farmacológica donde la información no está disponible a través de APIs convencionales. Es el caso de aquellas fuentes de datos de las que obtenemos información a través de la técnica del web scraping.

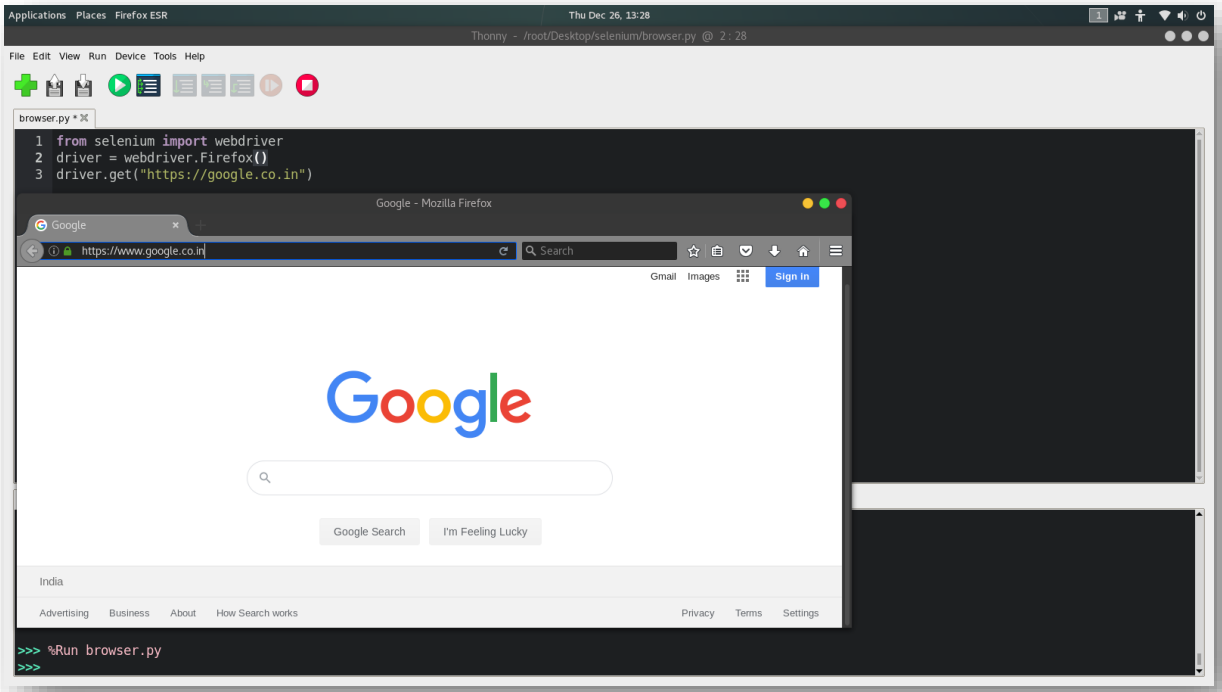



Figura 9. Ejemplo de uso de la Librería Selenium. Se simula de forma programática el acceso a la web de Google.

### 2.3.5. React

React es una biblioteca de JavaScript de código abierto desarrollada por Facebook (ahora Meta) para construir interfaces de usuario, especialmente para aplicaciones web de una sola página (SPA). Fue creada por Jordan Walke y lanzada públicamente en 2013. React se basa en un enfoque declarativo y componente-basado que permite a los desarrolladores crear interfaces de usuario complejas mediante la composición de componentes reutilizables. Su arquitectura utiliza un Virtual DOM que optimiza las actualizaciones del DOM real, mejorando significativamente el rendimiento de las aplicaciones web. (21)

En el proyecto, React se utiliza para el desarrollo del frontend de la interfaz de usuario que consume la API del sistema backend. Su capacidad para manejar el estado de la aplicación y renderizar componentes de manera eficiente permite crear una experiencia de usuario fluida y responsive para la visualización de los informes biomédicos generados. La modularidad de React facilita la creación de

componentes especializados para diferentes tipos de datos (tablas, gráficos, secciones de información) que pueden ser reutilizados a lo largo de la aplicación.

### 2.3.6. Node.js

Node.js es un entorno de ejecución de JavaScript del lado del servidor construido sobre el motor V8 de Chrome, creado por Ryan Dahl en 2009. Node.js permite ejecutar JavaScript fuera del navegador web, utilizando un modelo de E/O no bloqueante y orientado a eventos que lo hace ligero y eficiente. Su ecosistema incluye npm (Node Package Manager), el mayor repositorio de bibliotecas de software del mundo, que facilita la gestión de dependencias y la reutilización de código. (22)

En el proyecto, Node.js proporciona el entorno de ejecución para las herramientas de desarrollo del frontend, incluyendo la gestión de dependencias, el proceso de build y el servidor de desarrollo para la aplicación React. Su capacidad para manejar múltiples conexiones concurrentes de manera eficiente y su amplio ecosistema de paquetes lo convierten en una plataforma ideal para el desarrollo de aplicaciones web modernas y la integración con diversas herramientas de desarrollo.

### 2.3.7. Tkinter

Tkinter (Tk interface) es la biblioteca estándar de Python para crear interfaces gráficas de usuario (GUI), incluida por defecto en la distribución estándar de Python. Basada en el toolkit Tk de Tcl, Tkinter proporciona una interfaz orientada a objetos para crear aplicaciones de escritorio multiplataforma. Fue desarrollada originalmente por Guido van Rossum y ha sido parte integral del ecosistema Python desde sus primeras versiones, ofreciendo widgets básicos como ventanas, botones, menús y campos de texto. (23)

En el proyecto, Tkinter se utiliza para crear prototipos rápidos de interfaces de usuario durante las fases iniciales de desarrollo y validación con el cliente. Su simplicidad y disponibilidad inmediata en Python lo convierten en una herramienta ideal para crear demos funcionales y mockups interactivos que permiten validar conceptos de interfaz de usuario antes de proceder con el desarrollo del frontend web definitivo en React. La facilidad de uso de Tkinter

permite iterar rápidamente sobre diferentes diseños de interfaz y recoger feedback temprano del usuario.

## 2.4. Herramientas colaborativas

### 2.4.1. Microsoft Teams

Microsoft Teams es una plataforma unificada de comunicación y colaboración que combina chat persistente en el lugar de trabajo, reuniones de video, almacenamiento de archivos e integración de aplicaciones. Este servicio se integra con el conjunto de productividad de Office 365 y cuenta con extensiones que pueden integrarse con productos que no son de Microsoft.

Se hace uso de Microsoft Teams para la comunicación entre los miembros del equipo, la coordinación de tareas, el intercambio de documentación técnica y la realización de reuniones de seguimiento, facilitando un entorno colaborativo eficiente.

### 2.4.2 OneNote

OneNote es una aplicación de toma de notas digitales desarrollada por Microsoft que forma parte del ecosistema de Microsoft 365. Esta herramienta colaborativa permite crear, organizar y compartir notas estructuradas en formato de cuaderno digital, facilitando la captura y gestión de información de manera flexible y accesible desde múltiples dispositivos y plataformas con sincronización automática en la nube. (11)

En el contexto del desarrollo del proyecto, OneNote se utiliza como plataforma central para la documentación ágil y colaborativa de ideas emergentes. El equipo de desarrollo emplea esta herramienta para registrar apuntes rápidos durante las reuniones con el cliente del IIER y las sesiones internas de planificación, aprovechando sus capacidades de dibujo y esquematización para representar arquitecturas de sistema, flujos de datos y diseños de interfaces de manera visual. La naturaleza colaborativa en tiempo real permite que ambos miembros del equipo contribuyan simultáneamente, manteniendo un registro dinámico y centralizado de los puntos emergentes inmediatos que van surgiendo a lo largo del proyecto.

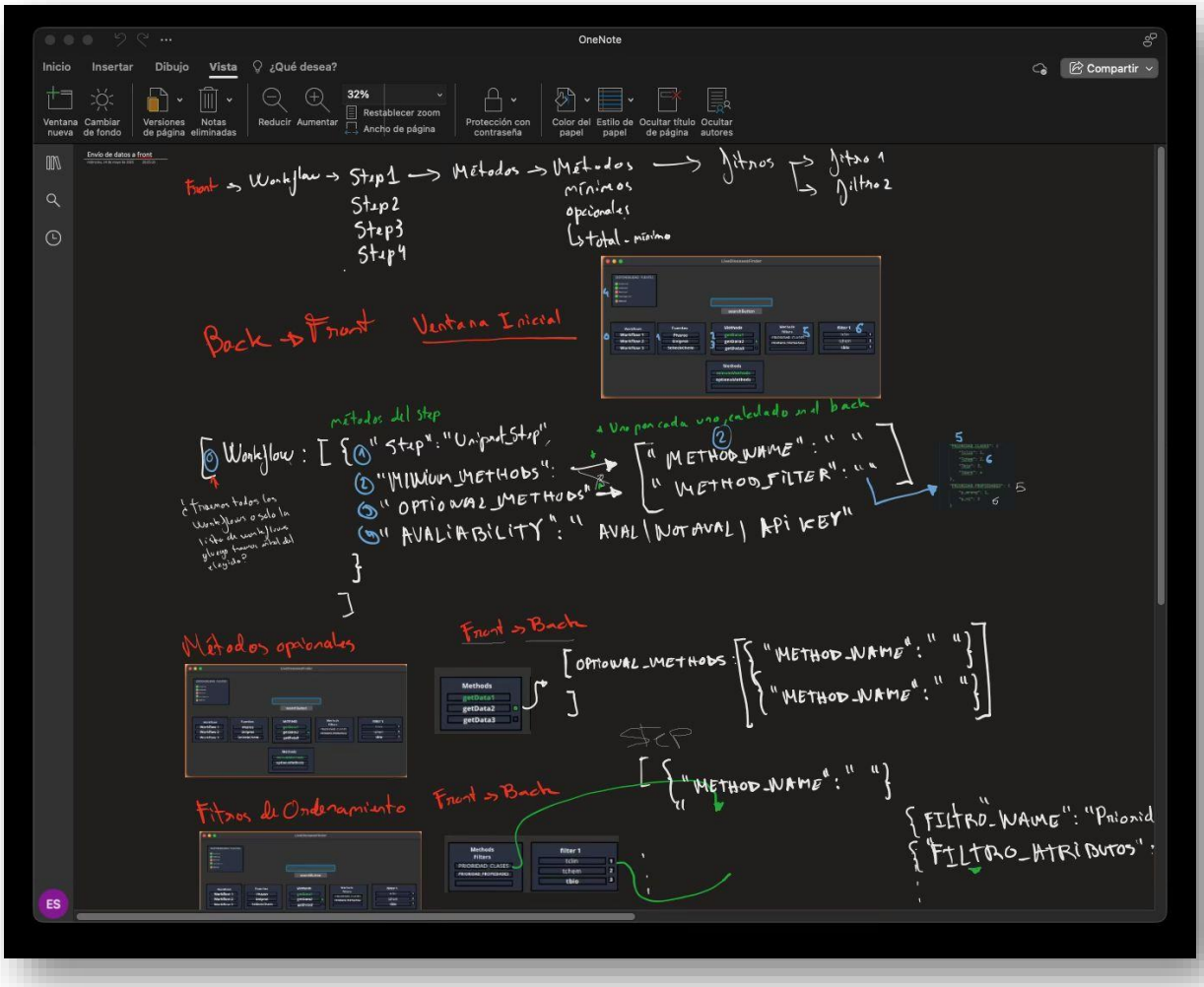



Figura 10. Representación de uso de OneNote para discutir ideas de proyecto.

### 2.4.3 GitHub

GitHub es una plataforma de desarrollo colaborativo basada en la nube que utiliza Git como sistema de control de versiones. Fue fundada en 2008 por Tom Preston-Werner, Chris Wanstrath y PJ Hyett, y posteriormente adquirida por Microsoft en 2018. GitHub proporciona hosting para repositorios de código fuente, facilitando la colaboración entre desarrolladores a través de características como pull requests, issues, wikis, y herramientas de revisión de código. La plataforma se ha convertido en el estándar de facto para el desarrollo de software colaborativo, albergando millones de proyectos de código abierto y privado. (24)

En nuestro proyecto, GitHub desempeña un papel especialmente fundamental como plataforma de alojamiento y colaboración para el desarrollo del proyecto. Utilizamos GitHub para mantener tanto el repositorio del backend (https://github.com/MarioBravoCuadro/RareDiseaseFinder) como el frontend (https://github.com/jennygatv/front-RD/), permitiendo la colaboración efectiva entre los miembros del equipo mediante el uso de ramas para gestionar el desarrollo de nuevas funcionalidades y correcciones de código. La visibilidad pública de nuestros repositorios facilita la transparencia del proyecto y permite contribuciones de la comunidad científica.

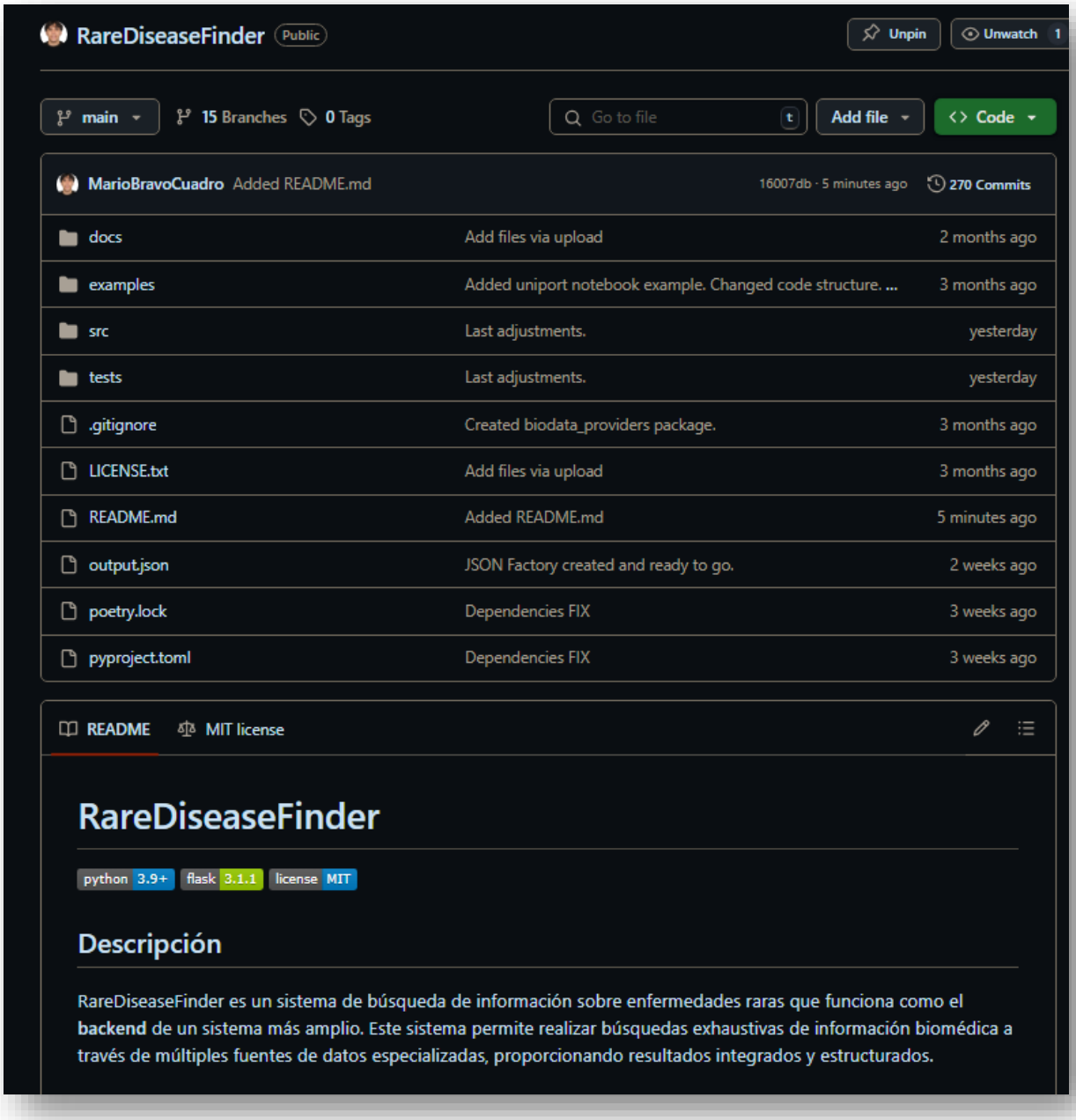



Figura 11. Representación del repositorio de nuestro proyecto. Fuente: https://github.com/MarioBravoCuadro/RareDiseaseFinder

# 3. Metodología de desarrollo

La elección de una metodología de desarrollo correcta es esencial para poder completar el proyecto de manera organizada y ordenada. Existen diferentes enfoques en materia de desarrollo software, cada una de ellas con sus puntos positivos y negativos. Una de las responsabilidades es baremar las diferentes

metodologías existentes y seleccionar la que mejor se adapte, pero no sólo a los desarrolladores; también al contexto del proyecto y a las personas implicadas.

Dentro de las metodologías de desarrollo software, éstas se clasifican en dos grupos: metodologías tradicionales y ágiles. Entre estas dos filosofías se encuentran las metodologías mixtas que mezclan elementos de ambas vertientes, pero sin despreciar el uso de reglas y procesos establecidos.

En este apartado se describirán algunas de estas metodologías y se dará una explicación de la adopción o rechazo de los elementos de cada una de ellas.

## 3.1. Metodologías tradicionales

Las metodologías tradicionales se caracterizan por definir total y rígidamente los requisitos al inicio de los proyectos de ingeniería de software. Los ciclos de desarrollo son poco flexibles y no permiten realizar cambios, al contrario que las metodologías ágiles; lo que ha propiciado el incremento en el uso de las segundas. (25)

Aún sabiendo estas características, evaluaremos algunas de las metodologías tradicionales más comunes para ver qué elementos de estas podríamos incluir en nuestra metodología de trabajo personalizada.

Algunos de los ejemplos más destacados de esta metodología son:

- **Cascada (Waterfall)**: Modelo lineal y secuencial donde cada fase debe completarse antes de iniciar la siguiente. Este modelo establece una progresión ordenada a través de requisitos, diseño, implementación, verificación y mantenimiento. (26)

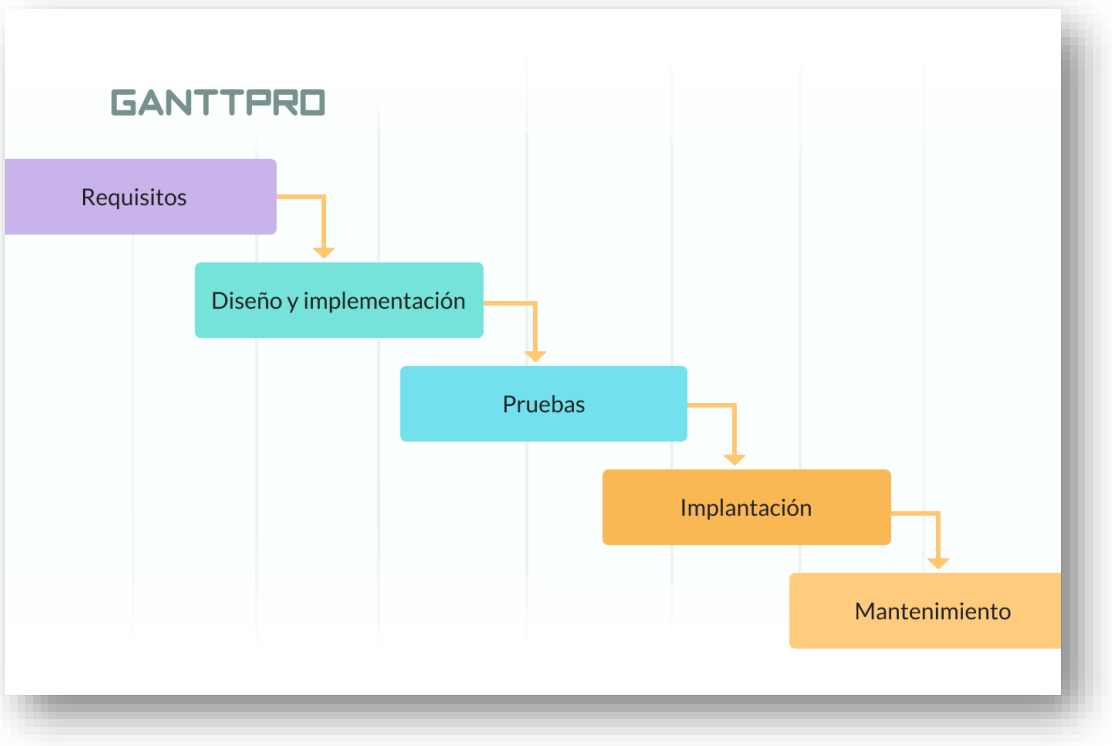



Figura 12. Representación del modelo en cascada. Fuente: https://blog.ganttpro.com/es/metodologia-de-cascada/

- Para nuestro proyecto no es ideal, puesto que el entorno es muy cambiante, sin una definición clara al principio del desarrollo y con actualizaciones constantes del dominio.

- **Rational Unified Process (RUP)**: Metodología que divide el proyecto en cuatro fases (inicio, elaboración, construcción y transición) con múltiples iteraciones, pero manteniendo un enfoque formal en la documentación y el modelado UML. (27)

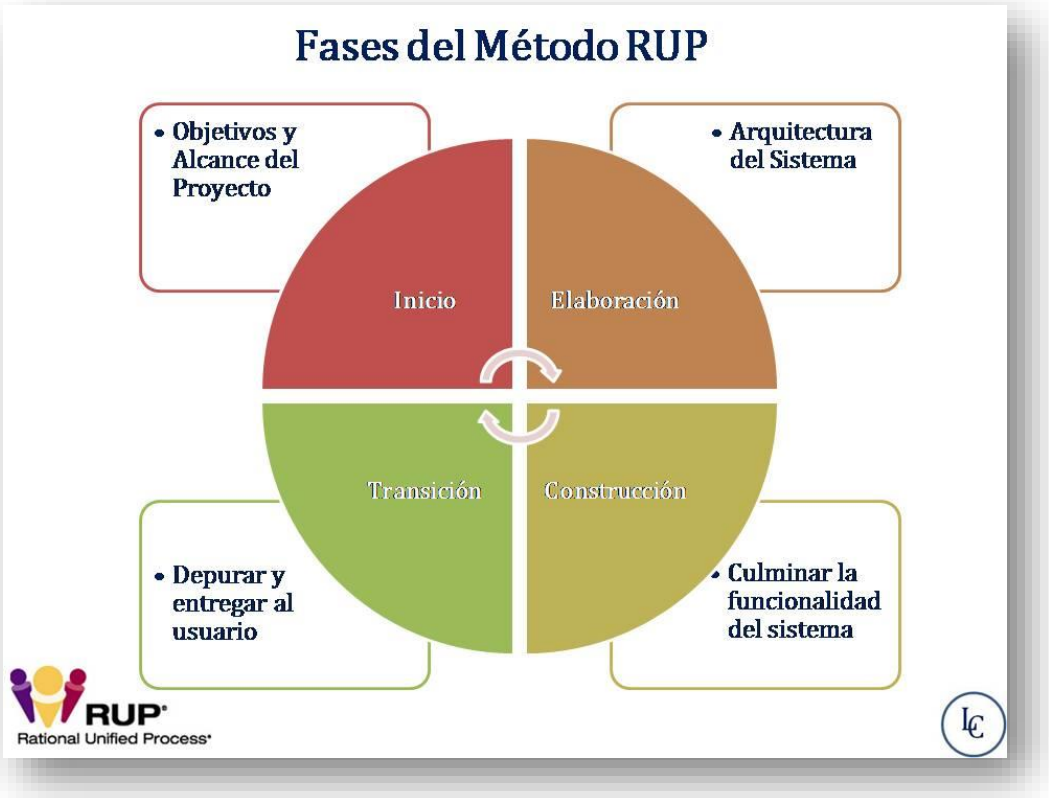



Figura 13: Representación del modelo RUP. Fuente: https://lean-management.site/rup/

- Adoptamos su filosofía incremental y el uso del modelado UML para algunos elementos como los diagramas de clases, aunque desestimamos su rigidez en las fases de desarrollo ante la naturaleza flexible de nuestro proyecto.

- Modelo en V: Extensión del modelo en cascada que enfatiza la verificación y validación en cada etapa, estableciendo una correspondencia entre las fases de desarrollo y pruebas. (28)

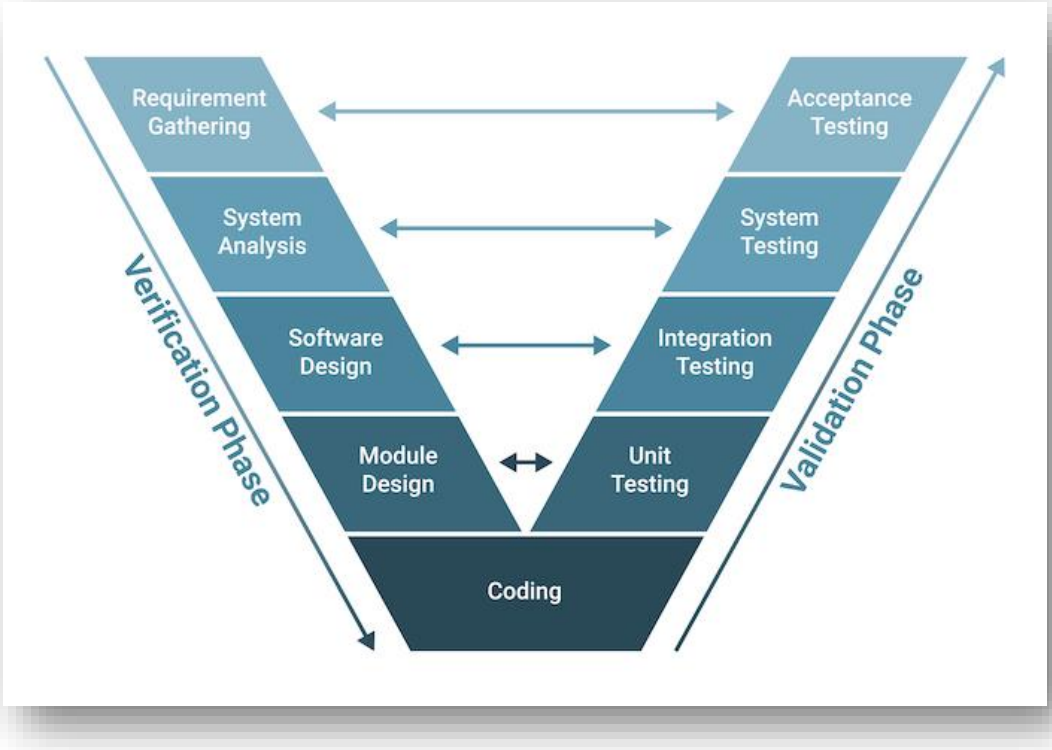



Figura 14: Representación del modelo en V. Fuente: https://dorleco.com/v-model-development-techniques-to-design/

- Desestimamos su estructura secuencial puesto que no se asocia con las necesidades de la solución. Además, el largo tiempo que transcurre entre la fase de elicitación de requisitos y la de los tests de aceptación no tendría sentido en un proyecto como el nuestro cuyos requisitos están en constante evolución.

## 3.2. Metodologías ágiles

Las metodologías ágiles surgieron como respuesta a las limitaciones percibidas en los enfoques tradicionales, especialmente en entornos donde los requisitos cambian con frecuencia. Reconocen la naturaleza impredecible del desarrollo de software y se adaptan a ella en lugar de intentar controlarla mediante una planificación extensa. (29)

A continuación se evaluarán las principales metodologías ágiles más utilizadas para seleccionar de cada una de ellas aquellos elementos que sean útiles para el transcurso del proyecto.

Algunos de los ejemplos más destacados de las metodologías ágiles son:

- **Scrum**: Marco de trabajo iterativo e incremental que organiza el desarrollo en sprints de duración fija (típicamente 2-4 semanas), con roles definidos como Product Owner, Scrum Master y equipo de desarrollo. (30) Esta metodología además incluye actividades especiales como reuniones diarias entre desarrolladores, definición del trabajo de cada sprint o reuniones retrospectivas de cada iteración.

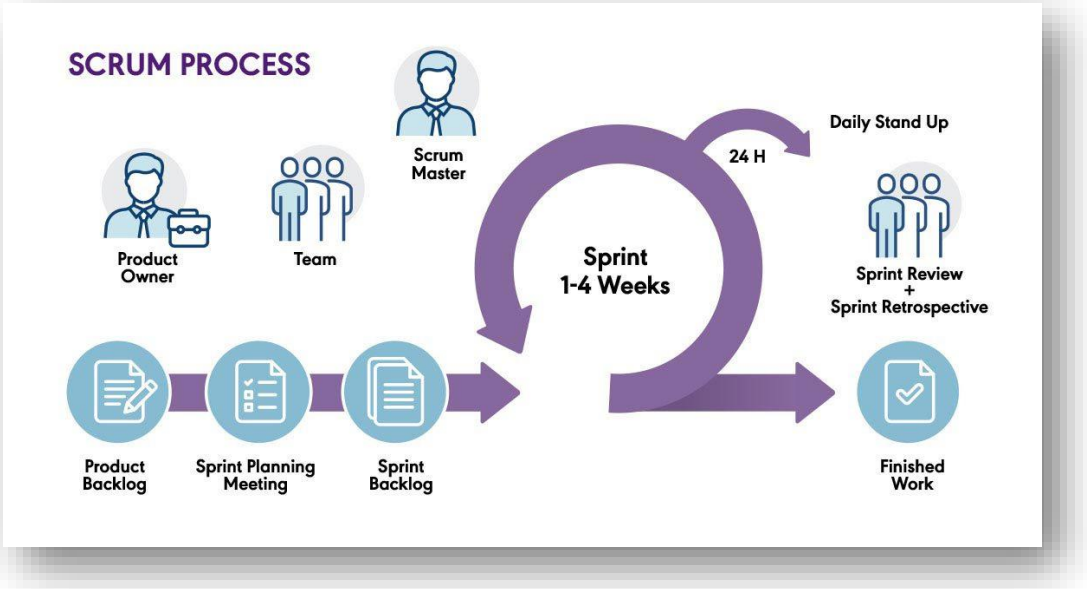



Figura 15: Representación de la metodología Scrum. Fuente: https://melonhelp.com/que-es-la-metodologia-scrum/

  - Esta metodología resulta especialmente interesante, no sólo por su carácter incremental en el que el producto final va ganando valor de forma progresiva a medida que se van incluyendo funcionalidades, sino por la comunicación regular y diaria entre los desarrolladores. Se desestiman los roles y las actividades de grupo típicas de Scrum puesto que el proyecto no cuenta con un equipo lo suficientemente numeroso para ello al ser tan solo dos desarrolladores que además asumen funciones de entrevistador con el cliente final.

- Extreme programming (XP): Metodología que enfatiza en la calidad del código mediante prácticas como la programación en parejas, desarrollo guiado por pruebas, integración continua y refactorización frecuente (31) .

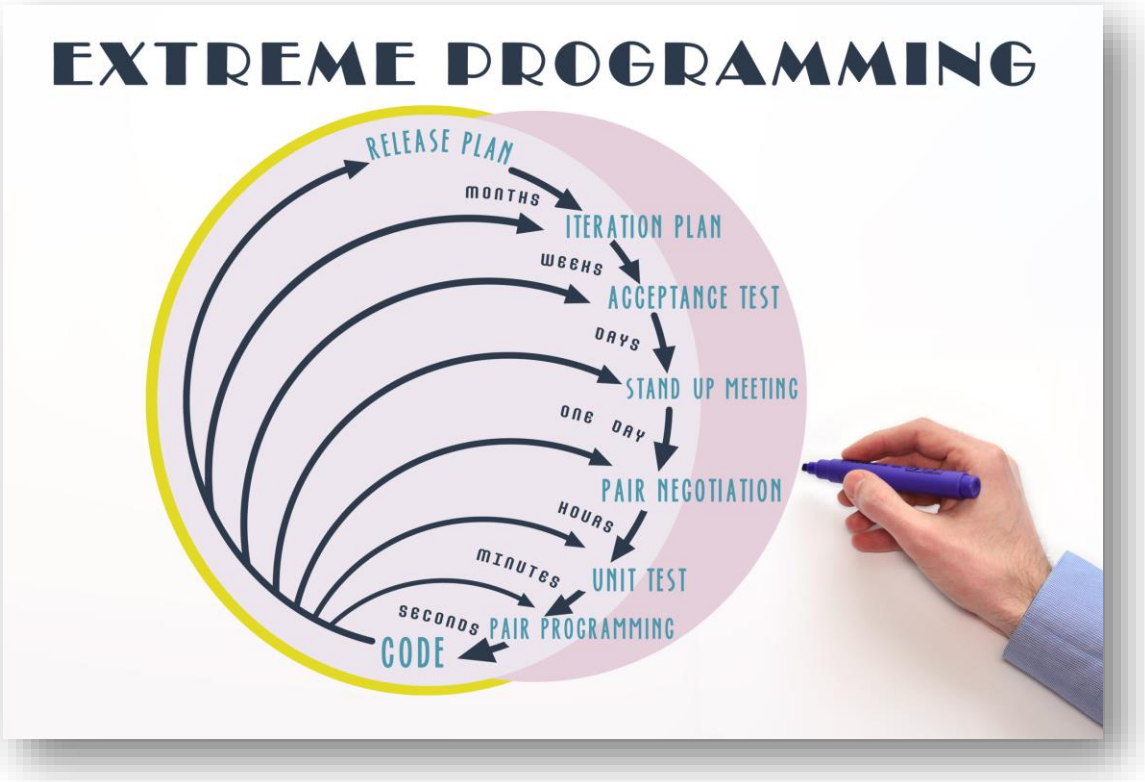



Figura 16: Representación de la metodología XP. Fuente: https://contenteratechspace.com/blog/what-is-extreme-programming-in-agile/

- Este modelo es especialmente interesante por su naturaleza altamente adaptativa: es idóneo para proyectos en los que los requisitos del cliente no están claros y cambian a menudo, De él se adoptan la comunicación constante con el cliente haciendo entrega, en periodos muy cortos de tiempo, pequeñas funcionalidades que el cliente pueda validar rápidamente.

- Kanban: Enfoque visual para la gestión del flujo de trabajo que limita el trabajo en progreso y optimiza la entrega continua. (32) Ofrece una definición clara para darle un estado específico a cada elemento a desarrollar en el proyecto.

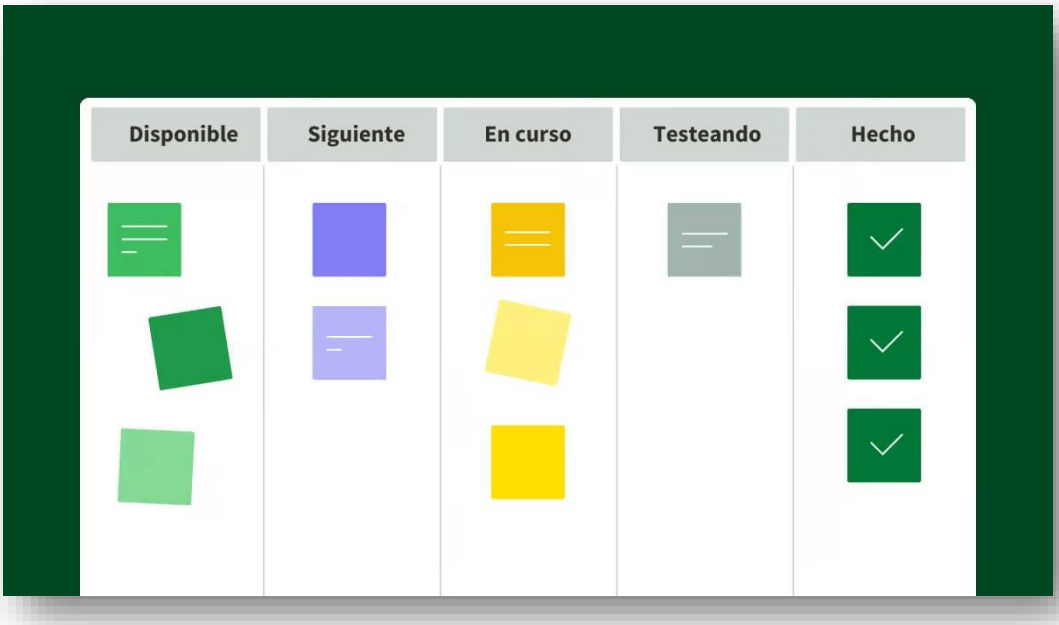



Figura 17: Representación de un tablero Kanban. Fuente: https://www.pipedrive.com/es/blog/metodo-kanban

- Aunque decidimos no adoptar ningún elemento de la metodología Kanban, reconocemos gracias a esta metodología la importancia de incluir un elemento puramente visual que permita hacer un seguimiento constante de los elementos del sistema. En específico, encontramos muy útil este planteamiento para hacer un seguimiento del estatus de desarrollo del acceso a cada fuente de datos biomédicos; cada fuente de datos puede encontrarse en distintas fases: investigación de la fuente, implementación, testeo y completada.

## 3.3. Enfoque de metodología adoptado

Para el desarrollo del sistema se adopta un enfoque híbrido que combina distintos elementos tanto de metodologías tradicionales como ágiles, adaptándose así a las necesidades específicas del proyecto y al contexto de colaboración con los investigadores del IIER.

Como se ha repasado en el punto anterior, se han consultado exhaustivamente algunas de las metodologías más utilizadas de estas dos filosofías del desarrollo software, resultando en el enfoque explicado a continuación.

### 3.3.1. Estrategia de desarrollo dual

La dinámica de trabajo con los investigadores del IIER nos ha obligado a tomar dos enfoques simultáneos para el proyecto. Por un lado, existía la necesidad de poder compartir ideas sobre la solución de forma directa y clara con los investigadores, tratando de llevar nuestros avances y ocurrencias a un entorno no técnico en el que hacernos comprender y llegar a la solución más cercana a sus necesidades posible. Por el contrario, era necesario trabajar en un entorno que asegurara la calidad de la solución, donde poder plantear con la debida planificación los aspectos más abstractos del proyecto para construir un software robusto y sostenido por conceptos técnicos previamente trabajados y discutidos, exigiendo así una claridad manifiesta en los requisitos y el diseño de nuestro sistema y en los patrones y principios software aprendidos a lo largo de la titulación.

Esta dicotomía entre la forma de afrontar el desarrollo del proyecto nos hizo inclinarnos por una estrategia que consiste en dos ramas de desarrollo cuyas características potencian lo más útil y lo más necesitado de las metodologías de desarrollo software.

### 3.3.2. Rama de prototipos rápidos (enfoque ágil):

En esta rama, se hace uso de prototipos desechables que nos permiten validar rápidamente las necesidades reales del cliente en un formato comprensible para el grupo de investigación del IIER. Sus características principales son:

- **Implementada en Google Colab:** Plataforma en la nube que permite un despliegue rápido con configuraciones mínimas. Esta plataforma proporciona un entorno de ejecución inmediato sin necesidad de instalaciones locales complejas. Esta es una elección clave para demostrar funcionalidades a los investigadores del IIER sin las barreras técnicas que podrían presentar otros entornos de desarrollo.

- **Orientada a la validación temprana con usuarios finales:** Este enfoque permite verificar la utilidad real de las funcionalidades desarrolladas no solo desde las primeras etapas del proyecto, sino de forma constante a lo largo del mismo. Gracias a esta característica, reduciendo así el riesgo de invertir recursos en características que no satisfacen las necesidades específicas de los investigadores del IIER; el software desarrollado en este proyecto

pertenece a un dominio altamente especializado donde la experiencia del usuario es crítica.

- **Permite desarrollar prototipos de "usar y tirar":** La elaboración de implementaciones experimentales que pueden ser descartadas sin impacto significativo en el desarrollo global facilita la exploración de múltiples aproximaciones a los distintos problemas de obtención, procesamiento o presentación de información que puedan surgir dinámicamente. Esta flexibilidad es especialmente valiosa en un proyecto de estas características, donde las necesidades del cliente pueden evolucionar durante el proceso de desarrollo.

- **Facilita la interactividad y demostración inmediata:** Los cuadernos de Google Colab permiten combinar código, visualizaciones y explicaciones en un formato interactivo donde los investigadores pueden observar resultados en tiempo real y proponer modificaciones sobre la marcha. Este nivel de interactividad resulta fundamental para validar la correcta obtención de datos biomédicos complejos y garantizar que la herramienta responde adecuadamente a las exigencias de los investigadores del IIER.

- **Prioriza la velocidad de implementación sobre la estructura del código:** Adoptar un enfoque inicialmente en la funcionalidad por encima de la arquitectura, permite presentar soluciones operativas en plazos reducidos, factor crucial para nosotros al trabajar con equipos de investigación con limitaciones temporales y franjas de tiempo reducidas y muy marcadas. Esta priorización facilita una evaluación temprana de la viabilidad de integrar las distintas fuentes de datos biomédicos que el cliente exige, no sólo teniendo en cuenta si incluir cierta funcionalidad es plausible o no, sino evaluando también el factor de tiempo y esfuerzo que supondría incluirla.

- **Genera retroalimentación rápida de las investigadoras del IIER:** La disponibilidad de prototipos funcionales en todas las iteraciones del proyecto establece un ciclo de retroalimentación continua que permite refinar en todo momento las funcionalidades según las necesidades específicas de los

investigadores del IIER. Este diálogo constante entre desarrolladores e investigadores minimiza el riesgo de desviaciones conceptuales en este dominio tan especializado y ayuda a los investigadores, que no tienen un trasfondo técnico, a ver de primera mano los avances del proyecto, entiendo así en más detalle el funcionamiento y las características completas del sistema.

### 3.3.3. Rama de desarrollo formal (enfoque tradicional)

Aunque esta rama incorpora elementos iterativos del desarrollo ágil, se denomina como "tradicional" por su base en el diseño previo y requisitos validados. Este enfoque permite construir una solución que combina la solidez estructural necesaria para un sistema de obtención de información biomédica complejo con la capacidad de adaptación a las necesidades emergentes en la investigación de enfermedades raras. Las características de esta rama son las siguientes:

- **Implementada en el repositorio principal de GitHub:** La centralización del desarrollo en un sistema de control de versiones profesional facilita la colaboración estructurada entre desarrolladores, el seguimiento riguroso de cambios y la integración continua. Esta infraestructura tecnológica proporciona la base necesaria para un desarrollo sostenible y mantenible a largo plazo, crucial para un sistema que debe evolucionar con el avance de nuevos accesos programáticos y nuevas fuentes biomédicas que puedan requerirse en el futuro.

- **Se inicia tras la validación de conceptos en los prototipos:** Este enfoque secuencial, aunque mantiene elementos iterativos, se fundamenta en la premisa de que solo las funcionalidades completamente validadas por los usuarios finales merecen una implementación formal. Esta transición controlada, entre exploración de posibles funcionalidades y consolidación de las funcionalidades reales, optimiza recursos y garantiza que el esfuerzo de desarrollo se concentre exclusivamente en características de valor demostrado para la investigación en enfermedades raras.

- **Aplica principios SOLID y patrones de diseño:** La implementación sistemática de estos principios de ingeniería del software no es meramente

una decisión técnica, sino una estrategia deliberada para asegurar que el sistema sea modificable, extensible y mantenible a largo plazo. Estos principios guían la estructura del código hacia una mayor cohesión y menor acoplamiento, facilitando la integración de nuevas fuentes de datos biomédicos sin comprometer la estabilidad del sistema.

- **Arquitectura previamente diseñada:** A diferencia del enfoque puramente ágil, en esta rama se invierte tiempo sustancial en el diseño arquitectónico previo a la implementación. Esta planificación anticipada establece los fundamentos estructurales del sistema, considerando aspectos como escalabilidad, rendimiento y seguridad desde las primeras etapas, lo que minimiza la necesidad de refactorizaciones profundas en fases avanzadas del desarrollo.

- **Documentación técnica completa:** Se elabora una documentación exhaustiva que abarca desde diagramas arquitectónicos hasta especificaciones detalladas de interfaces, proporcionando una referencia clara para el desarrollo y facilitando la incorporación de nuevos miembros al equipo. Esta práctica, característica de metodologías tradicionales, resulta especialmente valiosa en un dominio científico donde la precisión en la interpretación de datos es crítica.

- **Iteraciones con alcance definido:** Aunque mantiene el carácter iterativo propia de enfoques ágiles, cada ciclo de desarrollo se ejecuta sobre requisitos completamente validados y especificados, con un alcance claramente delimitado. Esta combinación de planificación detallada con entregas incrementales permite mantener la dirección estratégica del proyecto mientras se incorpora la retroalimentación de los investigadores de manera estructurada.

- **Enfoque en calidad y robustez:** Más allá de la funcionalidad, esta rama prioriza aspectos como la gestión de errores, pruebas unitarias y de integración, y optimización de rendimiento. Este énfasis en la calidad resulta fundamental para una herramienta que debe procesar datos

biomédicos complejos y proporcionar resultados confiables para la investigación científica en enfermedades raras, donde la precisión es imperativa.

## 3.4. Características transversales de la metodología

La implementación efectiva de nuestra estrategia híbrida descansa sobre diversos elementos transversales que articulan la colaboración entre el equipo de desarrollo y los usuarios finales. Estos componentes metodológicos han resultado fundamentales para el éxito del proyecto, facilitando tanto la comunicación como el seguimiento estructurado del desarrollo. A continuación, se describen cada uno de estos elementos transversales y se justifican para cada uno de ellos su uso y utilidad dentro de la metodología de desarrollo propuesto.

### 3.4.1. Validación continua mediante reuniones periódicas

Un elemento central de nuestra metodología ha sido el establecimiento de un ciclo regular de reuniones semanales con las investigadoras IIER. Estas reuniones se realizan por Microsoft Teams, y en ellas hay tres roles muy marcados:

- **Equipo de desarrollo:** conformado por los alumnos desarrolladores del proyecto. Se encargan de concretar las reuniones, proponer dudas y pedir aclaraciones en caso de ser necesario. Son los que se encuentran en contacto más directo con los investigadores del IIER.
- **Cliente/usuario final:** conformado por los investigadores del IIER. Aclaran dudas, ceden documentación relevante sobre las fuentes a incluir y describen en detalle la información clave que necesitan para tomar decisiones en su campo.
- **Supervisores:** conformado por los tutores del TFG, cuya función es la de aclarar dudas de carácter legal y administrativo, asesorando al equipo de desarrollo siempre que sea necesario en las implicaciones que las decisiones de proyecto puedan tener tanto en el presente como en una futura línea de trabajo.

Adicionalmente, estas sesiones presentan características distintivas que potencian su eficacia:

- **Demostración de prototipos funcionales:** En cada reunión se presentan las implementaciones desarrolladas en Google Colab, mencionadas anteriormente, durante la semana. Estas demostraciones proporcionan una experiencia interactiva donde las investigadoras pueden observar el comportamiento real del sistema con datos biomédicos relevantes para su trabajo; los investigadores comparan los datos que ellos obtienen manualmente frente a los datos que el sistema obtiene de forma programática y prestan atención a las diferencias y al formato que los datos adoptan. La revisión de estos puntos les ayuda a tomar una mejor decisión sobre qué pasos seguir más adelante y hace que el sistema que se desarrolla se acerque lo máximo posible a lo que el cliente realmente necesita, gestionando de forma óptima las expectativas del cliente.

- **Validación en tiempo real:** A diferencia de revisiones basadas en documentos o descripciones abstractas propias de metodologías tradicionales, estas demostraciones permiten a los usuarios finales interactuar directamente con la herramienta en desarrollo, permitiéndoles evaluar con precisión si las funcionalidades implementadas satisfacen sus necesidades específicas. Estas revisiones son claves para ahorrar tiempo y añadir precisión a los requisitos del sistema, y no limita a documentar posteriormente en detalle los distintos conceptos de este; se trata de una práctica que aporta gran valor al proyecto sin cerrar la puerta a la documentación clásica.

- **Retroalimentación inmediata:** La estructura de estas reuniones facilita que los investigadores proporcionen feedback detallado sobre aspectos tanto funcionales como de usabilidad, señalando áreas de mejora o confirmando que las implementaciones cumplen con sus requisitos profesionales en el contexto de la investigación biomédica.

  Este aspecto además de aportar un valor técnico y de comunicación, dota de gran transparencia en la dinámica de trabajo, dando lugar a un entorno de confianza entre las partes interesadas. Un entorno que, citando el manifiesto ágil, permite descubrir verdades incómodas en etapas tempranas, cuando aún son baratas de corregir, y fomenta la colaboración genuina entre personas con diferentes especialidades y prioridades, transformando potenciales conflictos en oportunidades de innovación. (33)

En nuestro caso, este enfoque ha permitido detectar y corregir desviaciones con un coste mínimo, al identificarlas no solo en las etapas tempranas del desarrollo, sino a medida que iban surgiendo a lo largo del proyecto, situación muy común teniendo en cuenta el ambiente altamente cambiante en el que el sistema se desenvuelve.

### 3.4.2. Instrumentos de seguimiento y documentación

Para garantizar una gestión efectiva del conocimiento y facilitar la coordinación entre los diferentes participantes del proyecto, se implementan diversos instrumentos de seguimiento. Estos instrumentos de seguimiento han sido esenciales para la trazabilidad del proyecto y han permitido gestionar la creación de los distintos elementos de nuestra solución de una manera clara, justificada y ordenada. También destaca que estos documentos han sido esenciales para guardar registro de todas las modificaciones y actualizaciones del proyecto teniendo en mente su carácter de cambio continuo. Los elementos que se describirán a continuación pueden encontrarse en nuestro repositorio de GitHub para el proyecto:

https://github.com/MarioBravoCuadro/RareDiseaseFinder/tree/main/docs/guides

Algunos de estos instrumentos de seguimiento y documentación son:

#### 3.4.2.1. Canal de difusión de seguimiento

Se creó un canal de difusión dentro del proyecto con el equipo de desarrollo donde se escribía diariamente las novedades y avances diarios que se completaban a lo largo del proyecto. De esta forma, el equipo de desarrollo se mantenía actualizado en todos los puntos referentes al proyecto y facilitaba así la toma de decisiones y el seguimiento de cada uno de los elementos del sistema.

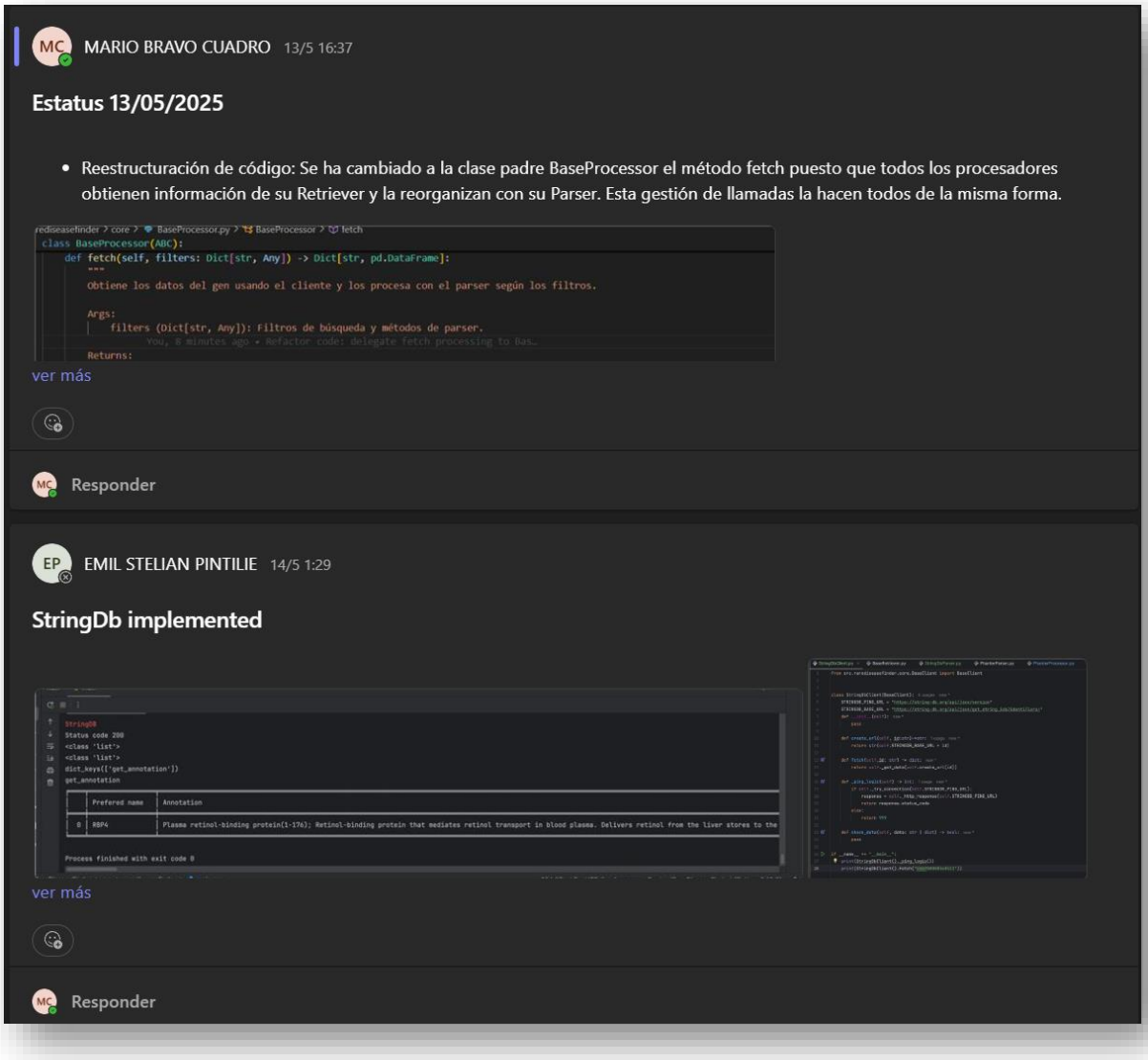



Figura 18. Representación del canal de difusión de seguimiento. Creación propia.

### 3.4.2.2. Registro estructurado de fuentes biomédicas

Se ha desarrollado un documento Excel exhaustivo que cataloga cada una de las fuentes de datos biomédicos que se han tenido en cuenta para el sistema, ya sean fuentes integradas o fuentes tenidas en cuenta por indicaciones del cliente. Este Excel puede encontrarse en el repositorio de GitHub del proyecto: Este registro incluye campos específicos como:

- **Identificación y descripción de la fuente**: Nombre, URL, tipo de información proporcionada y relevancia para la investigación en enfermedades raras.

| Fuente de datos | Descripción |
| --- | --- |
| https://pharos.nih.gov/ | Pharos es una base de datos de información sobre proteínas y compuestos, enfocada en la biología de los fármacos y la investigación de medicamentos. Proporciona datos sobre dianas terapéuticas, compuestos bioactivos y su relación con enfermedades.<br>Su disponibilidad es irregular. No es fiable. |
| https://www.orpha.net/es | Orphanet es una base de datos pública que proporciona información sobre enfermedades raras, su diagnóstico, tratamiento y recursos disponibles. Está destinada a pacientes, profesionales de la salud, investigadores y otros interesados en el área de enfermedades raras. Además de información sobre enfermedades, Orphanet ofrece datos sobre medicamentos huérfanos, ensayos clínicos y expertos en enfermedades raras. |

Figura 19. Representación de la descripción de las fuentes. Creación propia.

- **Características de acceso**: Disponibilidad de APIs oficiales, formatos de datos soportados, limitaciones de consulta y requisitos de autenticación.
- **Detalles técnicos de implementación**: Protocolos de comunicación utilizados, estructura de las respuestas, particularidades de la normalización de datos y desafíos específicos de integración.
- **Documentación adicional**: Documentación de apoyo que pueda servir para entender mejor la fuente y sus accesos: videos explicativos, documentación oficial, archivos descargables de apoyo, etc.

- **Estado de uso**: Indicador del estado de la fuente. Indica si la fuente está implementada en el sistema, si está pendiente de hacerse o si se incluye de forma estática con la inserción de un link a la web de la misma.

| API | Formato de datos | Detalles técnicos | Documentación útil | En uso |
|---|---|---|---|---|
| GraphQL API | RDF o JSON | Haríamos un fetch a través de una consulta GraphQL y lo pasaríamos a JSON.<br>Hay muchas APIs desarrolladas para esta web, sería interesante investigar su funcionamiento para aplicarlo no sólo a la extracción sino también al procesamiento de los datos. | Video explicativo de Pharos https://www.youtube.com/watch?v=EVir_va8xaY<br>https://pharos.nih.gov/api<br><br>Doc detallado de Pharos https://pdfs.semanticscholar.org/ca86/314ac0b18d0ec9aded7377329e24b718d962.pdf | Sí |
| API RESTful basada en HTTPS para acceder a sus datos | JSON o XML | Configuraríamos los parámetros de búsqueda y enviaríamos una solicitud HTTP GET para quedarnos con la información y procesarla a través de un JSON | Debemos registrarnos para el acceso a la API aquí:<br><br>https://omim.org/api | No |
| SPARQL endpoint | RDF o XML | ORDO (Orphanet Rare Disease Ontology) es una ontología especializada en enfermedades raras que forma parte del proyecto Orphanet. ORDO tiene como objetivo proporcionar una estructura formal para clasificar y categorizar las enfermedades raras de manera sistemática. Esta ontología se utiliza para mejorar la interoperabilidad entre bases de datos, investigación y atención médica relacionada con enfermedades raras.<br><br>Algunas bases de datos a las que accedemos como OMIM utilizan dicha ontología a la hora de estructurar sus datos. Esta estructura puede ser un buen punto de partida para entender cómo se organizan los datos. | https://www.orpha.net/consor/cgi-bin/index.php https://www.rd-code.eu/introduction/ | No |

Figura 20. Representación de la información clave de la fuente. Creación propia.

Este documento se corresponde con el apartado del anexo: Excel de seguimiento de fuentes

### 3.4.2.3. Guía de Desarrollador BackEnd

Esta guía proporciona las especificaciones completas para la implementación de nuevos proveedores de datos biológicos en el sistema. Se utiliza este documento para guiar a los desarrolladores que continúen en la mejora del proyecto en la correcta integración de fuentes de datos biomédicos. La guía explica la arquitectura

del sistema, las metodologías de implementación paso a paso, y los patrones de herencia que garantizan la consistencia y mantenibilidad del código base.

Se trata de un documento de suma importancia puesto que no sólo permite el desarrollo de nuevas integraciones de datos de acuerdo con los estándares arquitectónicos del sistema y los requisitos de escalabilidad, sino que también sirve como documento explicativo del patrón de diseño utilizado que ayuda a futuros desarrolladores a comprender de forma clara la estructura modular del proyecto y cómo extender sus capacidades.

Este documento se corresponde con el apartado del anexo: Guía de desarrollador Backend

#### 3.4.2.4. Guía de Desarrollador Frontnd

Esta documentación técnica proporciona las especificaciones completas para la comunicación entre el frontend y backend a través de la API REST. Se usa este documento para explicar a la desarrolladora de la interfaz de usuario cómo el frontend del sistema debe comunicarse con el backend, así como una explicación de la lógica utilizada y de cómo el front debe responder a los distintos pasos por los que transita el sistema.

Este documento es de suma importancia porque no sólo permite el desarrollo de una interfaz de acuerdo con las especificaciones del cliente y los requisitos del sistema, sino que también sirve como documento explicativo del funcionamiento del sistema que ayuda a futuros desarrolladores a entender de una forma clara los elementos más técnicos del proyecto.

Este documento se corresponde con el apartado del anexo: Guía de desarrollador Frontend

#### 3.4.2.5. Inventario de extracción de información

Este documento proporciona un catálogo exhaustivo de todas las capacidades de extracción de datos del sistema, organizando la información en siete secciones temáticas que abarcan desde descripción básica hasta referencias bibliográficas. Este documento permite proporcionar a desarrolladores, investigadores y usuarios finales una visión completa de los tipos de información biomédica que pueden obtenerse del sistema, detallando cada método disponible, su función específica, y las posibilidades de agrupación y estructuración de los datos resultantes.

Como en los documentos anteriores, no sólo permite comprender el alcance completo de las capacidades del sistema de acuerdo con las necesidades de investigación en enfermedades raras y los requisitos de integración de datos biomédicos, sino que también sirve como documento de referencia que ayuda a usuarios y desarrolladores a identificar qué información está disponible, cómo acceder a ella, y cómo puede combinarse para crear workflows personalizados que optimicen sus investigaciones específicas.

Este documento se corresponde con el apartado del anexo: Inventario de extracción de información

# 4. Diseño de la solución

En este apartado se presentarán todos los pasos y consideraciones que se han llevado a cabo para esbozar los elementos que componen la solución final.

## 4.1. Elicitación y especificación de Requisitos

Al tratarse de un proyecto que se desarrolla en un contexto real, la elicitación de requisitos se ha conseguido a través de una serie de reuniones semanales donde representantes del IIER y del equipo de desarrollo llegan a un consenso con el que se determina con exactitud qué debe hacer el sistema.

Estas reuniones periódicas se realizan como talleres de requisitos donde se concretan conjuntamente cada uno de los requisitos del sistema, donde los clientes son los investigadores del IIER, los entrevistadores lo componen el equipo de desarrollo y los mediadores son los tutores del Trabajo de Fin de Grado.

Destaca también el carácter multidisciplinar de estas reuniones, puesto que los conocimientos de los participantes así como sus antecedentes profesionales son muy variados: por un lado, los investigadores del IIER poseen un gran conocimiento en el campo de la biomedicina, pero ninguno en el de la informática; el experto en frontend del grupo de investigación de la UPM, que ayuda con su conocimiento de interfaces pero no conoce el contexto del proyecto; los tutores del Trabajo de Fin de Grado, que tienen amplia experiencia en gestión de proyectos pero no se sumergen por completo en el contexto de este; y finalmente, el equipo de desarrolladores, que poseen conocimientos en el área de la informática pero no poseen la experiencia ni el conocimiento del dominio del sector de la salud.

Esta diversidad de perfiles obliga a adoptar una postura de comunicación proactiva durante el desarrollo del proyecto, lo que ha implicado un esfuerzo constante por mantener un lenguaje neutral que todas las partes interesadas pudieran entender. Asimismo, la necesidad de interpretar adecuadamente las necesidades y requerimientos del cliente, desde el punto de vista de un campo de conocimiento ajeno al equipo de desarrollo como es el de la salud, manifiesta la necesidad de a ir un paso más allá en la comprensión del contexto, la terminología específica y los objetivos del proyecto, garantizando así una colaboración efectiva y una integración coherente de los distintos elementos que lo componen.

Los contenidos para la elicitación de requisitos incluyen:

- Apuntes sobre las reuniones en OneNote.
- Documentos Word explicativos con la estructura esperada del reporte final.
- Un documento Excel de seguimiento donde se registran todas las fuentes planteadas a incluir en el sistema y cómo se accede a ellas (si se accede).
- Documentación específica de los accesos programáticos de cada fuente en específico.
- Presentaciones PowerPoint y otros prototipos que se han ido creando durante el desarrollo del proyecto.

Tras todas las reuniones atendidas, los requisitos del proyecto serían los siguientes:

### 4.1.1. Requisitos de usuario

Son descripciones sobre que tiene que ofrecer el sistema y las restricciones sobre las que este debe funcionar.

#### Requisitos funcionales:

| IDENTIFICADOR | DESCRIPCIÓN | IMPORTANCIA |
|---|---|---|
| REQ-U-FUN-1 | Como usuario quiero tener la capacidad de hacer una búsqueda sobre diferentes fuentes de biomedicina. | Alta |
| REQ-U-FUN-2 | Como usuario quiero tener la capacidad de seleccionar entre los diferentes tipos de flujos de búsqueda sobre fuentes de biomedicina. | Alta |

| | | |
|---|---|---|
| REQ-U-FUN-3 | Como usuario quiero tener la capacidad de seleccionar información adicional a obtener en la búsqueda de una fuente especifica. | Media |
| REQ-U-FUN-4 | Como usuario quiero tener la capacidad de añadir filtros opcionales de ordenamiento sobre fuentes que me permitan hacerlo. | Baja |
| REQ-U-FUN-5 | Como usuario quiero tener la capacidad de seleccionar el gen o proteína a buscar. | Alta |
| REQ-U-FUN-6 | Como usuario quiero tener la capacidad de ver el estado de las fuentes para saber su disponibilidad. | Baja |
| REQ-U-FUN-7 | Como usuario quiero tener la capacidad de ver que fuentes van a ser consultadas en cada flujo de trabajo. | Alta |
| REQ-U-FUN-8 | Como usuario quiero tener la capacidad de generar el informe a partir de la búsqueda que haya seleccionado. | Alta |
| REQ-U-FUN-9 | Como usuario quiero tener la capacidad de descargar el informe. | Alta |

De estos requisitos se han validado todos.

### Requisitos no funcionales:

| IDENTIFICADOR | DESCRIPCIÓN | IMPORTANCIA |
|---|---|---|
| REQ-U-NOFUN-1 | Como usuario quiero que las búsquedas sean fáciles de realizar. | Alta |
| REQ-U-NOFUN-2 | Como usuario quiero que las búsquedas se realicen en un tiempo razonable. | Alta |
| REQ-U-NOFUN-3 | Como usuario quiero que me acorten el tiempo que tardo en buscar información sobre fuentes biomédicas. | Alta |
| REQ-U-NOFUN-4 | Como usuario quiero que se me presente información de fuentes fiables y verificables. | Alta |

## 4.1.2. Requisitos del sistema

Son descripciones que establecen con detalles, las funciones y restricciones operativas que tiene el sistema.

### Requisitos funcionales:

| IDENTIFICADOR | DESCRIPCIÓN | IMPORTANCIA |
|---|---|---|
| REQ-S-FUN-1 | El sistema debe tener la capacidad de permitir hacer búsquedas a fuentes biomédicas | Alta |

| | | |
|---|---|---|
| REQ-S-FUN-2 | El sistema debe tener la capacidad de permitir seleccionar diferentes tipos de "Workflows", siendo estos flujos de búsqueda y procesamiento de información biomédica | Alta |
| REQ-S-FUN-3 | El sistema debe tener la capacidad de permitir al usuario seleccionar campos de información adicional a buscar en alguna de las fuentes. | Media |
| REQ-S-FUN-4 | El sistema debe tener la capacidad de permitir aplicar filtros de ordenamiento adicional en campos de algunas fuentes. | Baja |
| REQ-S-FUN-5 | El sistema debe tener la capacidad de permitir introducir el término del gen o proteína a buscar. | Alta |
| REQ-S-FUN-6 | El sistema debe tener la capacidad de generar el informe solicitado. | Alta |
| REQ-S-FUN-7 | El sistema debe tener la capacidad de descargar el informe generado. | Alta |
| REQ-S-FUN-8 | El sistema debe de informar al usuario de la disponibilidad de las fuentes antes de realizar la búsqueda. | Alta |
| REQ-S-FUN-9 | El sistema debe de informar al usuario de las fuentes que se van a consultar antes de realizar la búsqueda. | Alta |

De estos requisitos se han validado todos.

### 4.1.3. Requisitos no funcionales

| IDENTIFICADOR | DESCRIPCIÓN | IMPORTANCIA |
|---|---|---|
| REQ-S-NOFUN-1 | El sistema debe de disponer de una interfaz fácil de utilizar por usuarios relacionados al campo de la investigación. | Alta |
| REQ-S-NOFUN-2 | El sistema debe de realizar las consultas, procesamiento y generación del informe en un tiempo inferior a los 20 minutos. | Alta |
| REQ-S-NOFUN-3 | El sistema debe soportar realizar la búsqueda de términos de genes y proteínas sin que este falle. | Media |
| REQ-S-NOFUN-4 | El sistema debe tener la capacidad de realizar las búsquedas sin que el sistema falle. | Baja |
| REQ-S-NOFUN-5 | El sistema debe tener la capacidad de conseguir el estado de disponibilidad de una fuente cada 1 minuto. | Alta |
| REQ-S-NOFUN-6 | El sistema debe tener la capacidad de generar el informe solicitado sin que este contenga formatos incorrectos o este illegible o corrupto. | Alta |
| REQ-S-NOFUN-7 | El sistema debe tener la capacidad de descargar el informe generado en un formato HTML. | Alta |
| REQ-S-NOFUN-8 | El sistema debe tener la capacidad de descargar el informe generado en un formato PDF. | Alta |
| REQ-S-NOFUN-9 | El sistema debe de estar disponible solo para el uso de las investigadoras. | Alta |
| REQ-S-NOFUN-10 | El sistema debe de tener una interfaz gráfica ambientada en la biomedicina. | Medio |

| | | |
|---|---|---|
| REQ-S-NOFUN-11 | El sistema debe de ser accesible desde un navegador basado en Chronium o Firefox | Alta |

De estos requisitos se han validado todos menos REQ-S-NOFUN–8, debido a limitaciones de tiempo, sin embargo, el cliente para la primera entrega acepta el requisito REQ-S-NOFUN–7 como equivalente.

Requisitos inesperados:

| IDENTIFICADOR | DESCRIPCIÓN | IMPORTANCIA |
|---|---|---|
| REQ-S-INE-1 | El sistema ofrece una vista del buscador minimalista en el caso de que quiera usar el mismo Workflow, campos adicionales y filtros que la consulta anterior, en esta se muestra solamente el campo de búsqueda sin la información de la disponibilidad de las fuentes en diferentes Workflows. | Baja |
| REQ-S-INE-2 | El sistema permite visualizar el informe desde la propia aplicación. | Alta |
| REQ-S-INE-3 | El sistema genera un índice interactivo en la ventana de la visualización del informe. | Media |

Se han validado todos los requisitos con alto grado de satisfacción por parte del cliente.

## 4.2. Problemas en el diseño y soluciones planteadas

El diseño del sistema comenzó en etapas tempranas, al principio, al no tener todos los requisitos claros y tener requisitos sobre fuentes cambiantes, se ha decidido desarrollar un Producto Mínimo Viable (PMV) en la herramienta Google Collab, al cabo de pocas semanas el código existente estaba altamente acoplado y no era mantenible ni sostenible, tras esto al tener más claro los requisitos del sistema se empezó a diseñar una arquitectura que se iría adaptando en diferentes iteraciones hasta llegar a la arquitectura final, las diferentes fases se van a documentar a continuación.

### 4.2.1. Arquitectura V1

La primera iteración de la arquitectura consistía en un Pipeline definido en un notebook de Google Collab. Se extraían los datos de las diferentes fuentes con diferentes métodos, llamadas a API's de diferentes tipos, o el uso de scrappers con Sellenium, se procesaban los datos obtenidos y se almacenaban en Pandas

dataframes, al final el sistema contaba una celda donde se creaba el informe y se mostraba.

Esta aproximación fue útil hasta que se empezó a tener más de 5 fuentes, las cuales sumaban complejidad, anidación y poca modularidad del código, otro problema que tenía la arquitectura era la dificultad de gestionar los casos en los que una fuente no estaba disponible o traía datos en formatos inesperados.

A pesar de evolucionar la arquitectura, Google Collab se siguió utilizando para desarrollar y validar los nuevos requisitos de fuentes que iban surgiendo.

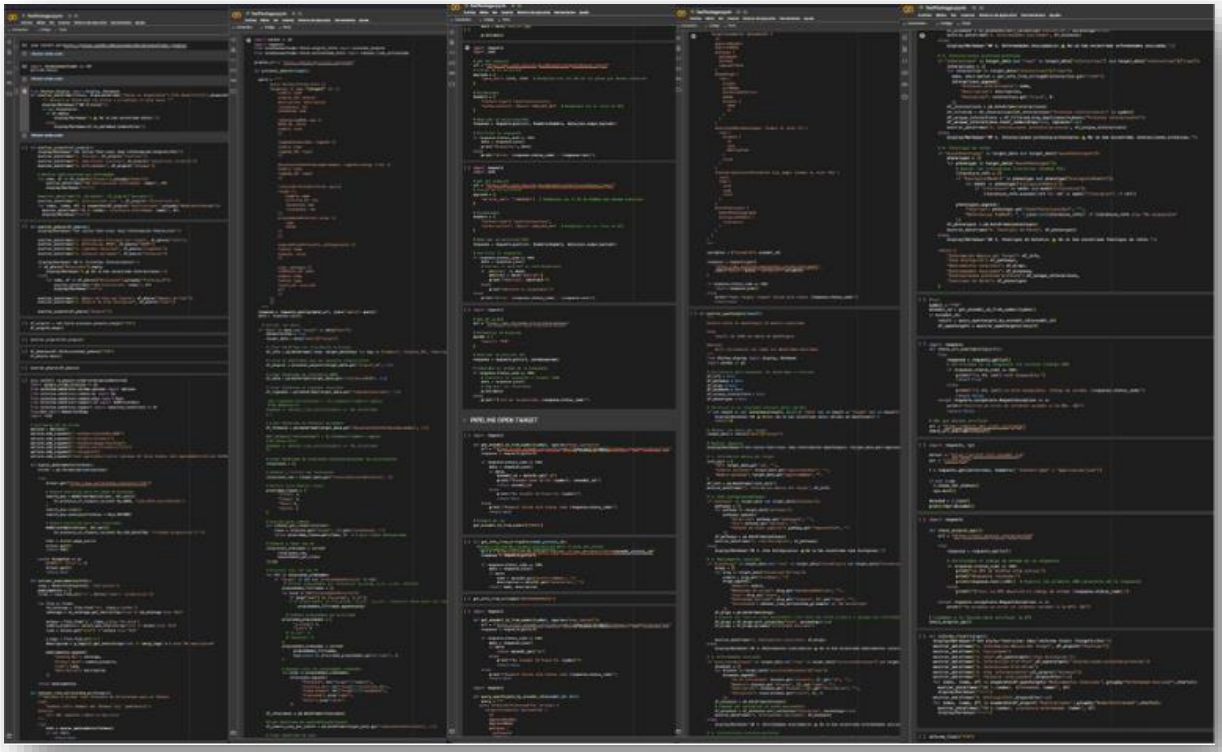


Figura 21. Tamaño del código tras 4 semanas de desarrollo.

### 4.2.2. Arquitectura V2

Para abarcar los diferentes tipos de mecanismos de recolección de datos; Sellenium, GraphQL, API's Restfull se decidió separar las responsabilidades en Cliente y Parser.

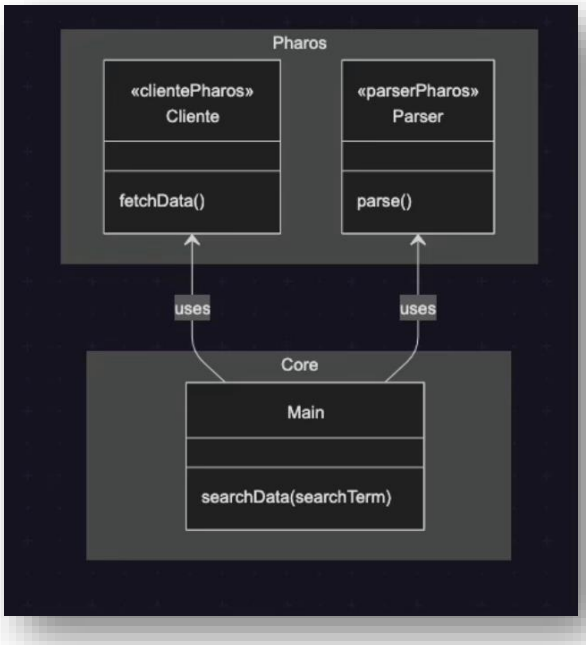



Figura 22. Diagrama de clases conceptual de Cliente y Parser

El Cliente de cada fuente se encargaría de implementar los mecanismos de conexión y petición de datos de la fuente mientras que el Parser implementaría mecanismos para decodificar, ordenar y seleccionar los puntos de la información que fuesen interesantes para el informe.

Tras el diseño de los Clientes y los Parsers se encontró otro problema, el acoplamiento entre las fuentes, esto nace a raíz de que hay fuentes que tienen como entrada la salida de otra fuente, esto con más clases generaría problemas de acoplamiento al escalar a un número mayor de fuentes.

En la figura 24 se puede observar como el Parser de Pharos hace uso de del Scrapper de Selleckchem y Cliente de Uniprot, el cliente de Parser de Pharos haría una llamada a Selleckchem y Uniprot y anidaría la salida a los datos de Pharos.

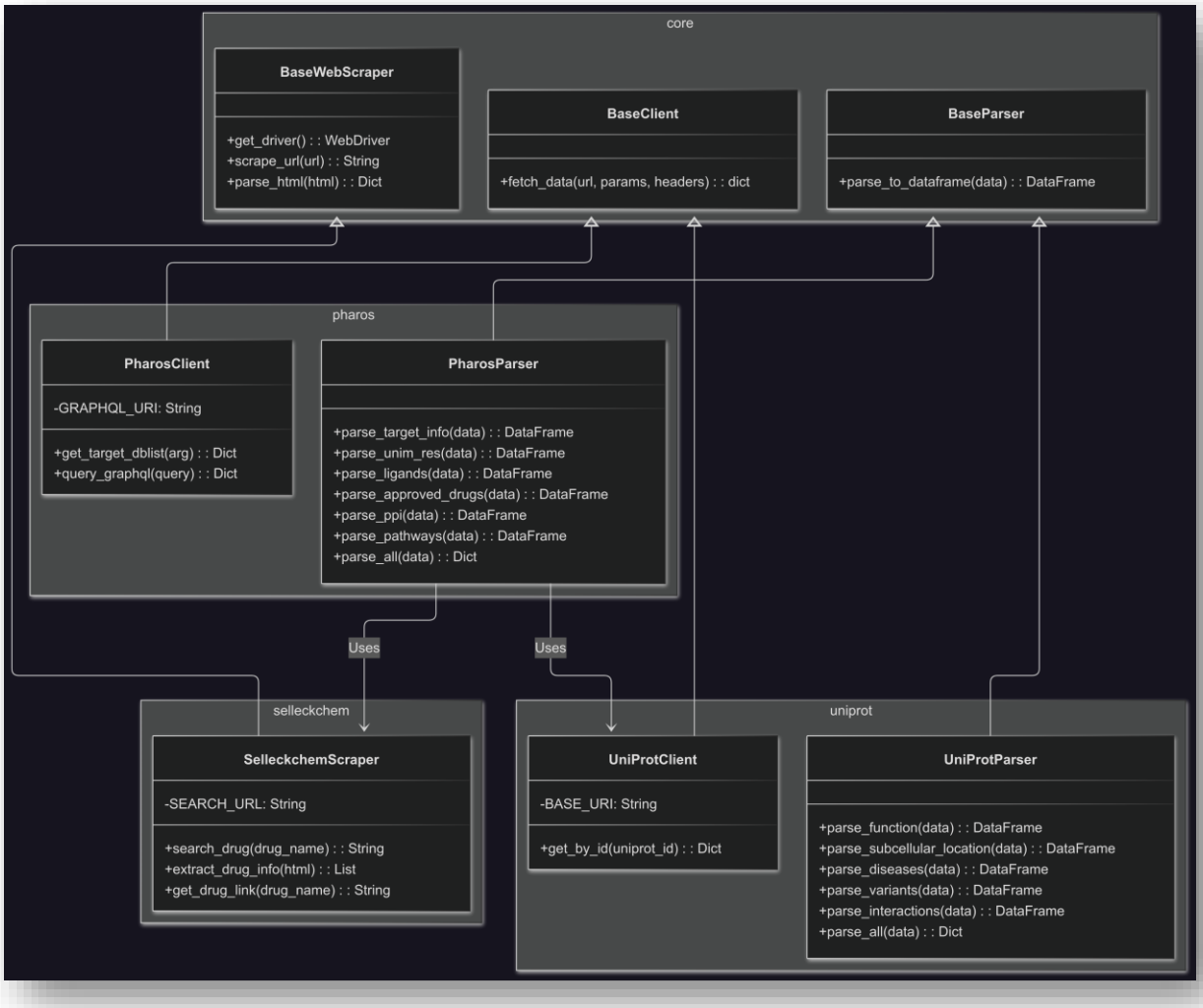



Figura 23. Diagrama de clases conceptual indicando acoplamiento entre diferentes fuentes

Para solucionar el acoplamiento de datos se decidió añadir un componente superior llamado orquestador el cual interactuaría con otro módulo llamado DataSource, este se encargaría de llamar a cada implementación de las fuentes.

Se introdujo un nuevo módulo que sería el procesador, una interfaz que se comunicaría con el Cliente y el Parser, de esta manera se desacoplaba y el procesador solo tenía que saber que término buscar y se encargaba de devolver el Dataframe con la información.

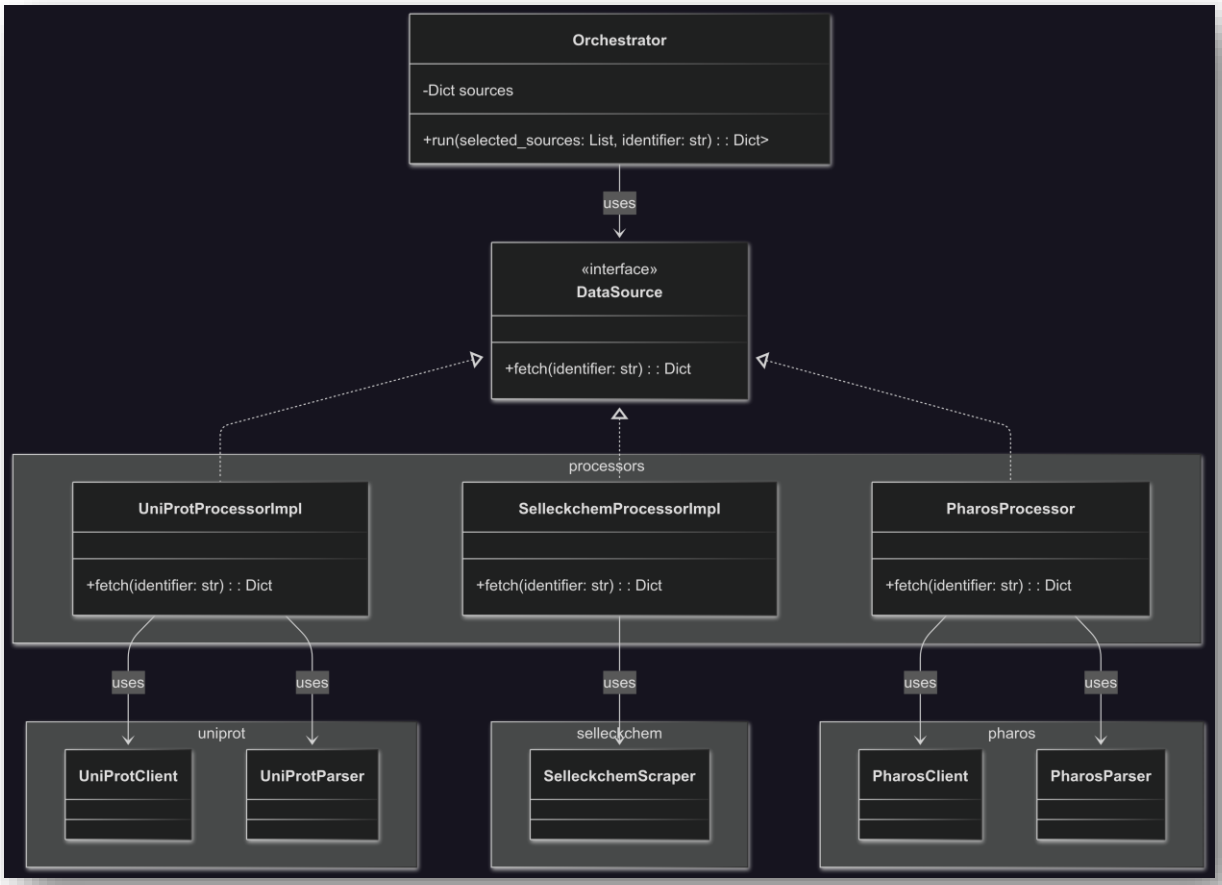



Figura 24. Diagrama de clases conceptual indicando el componente de nivel superior llamado Orquestador, DataSource y Procesadores

Esta solución seguía presentando el problema del acoplamiento, como se indica en la siguiente figura.

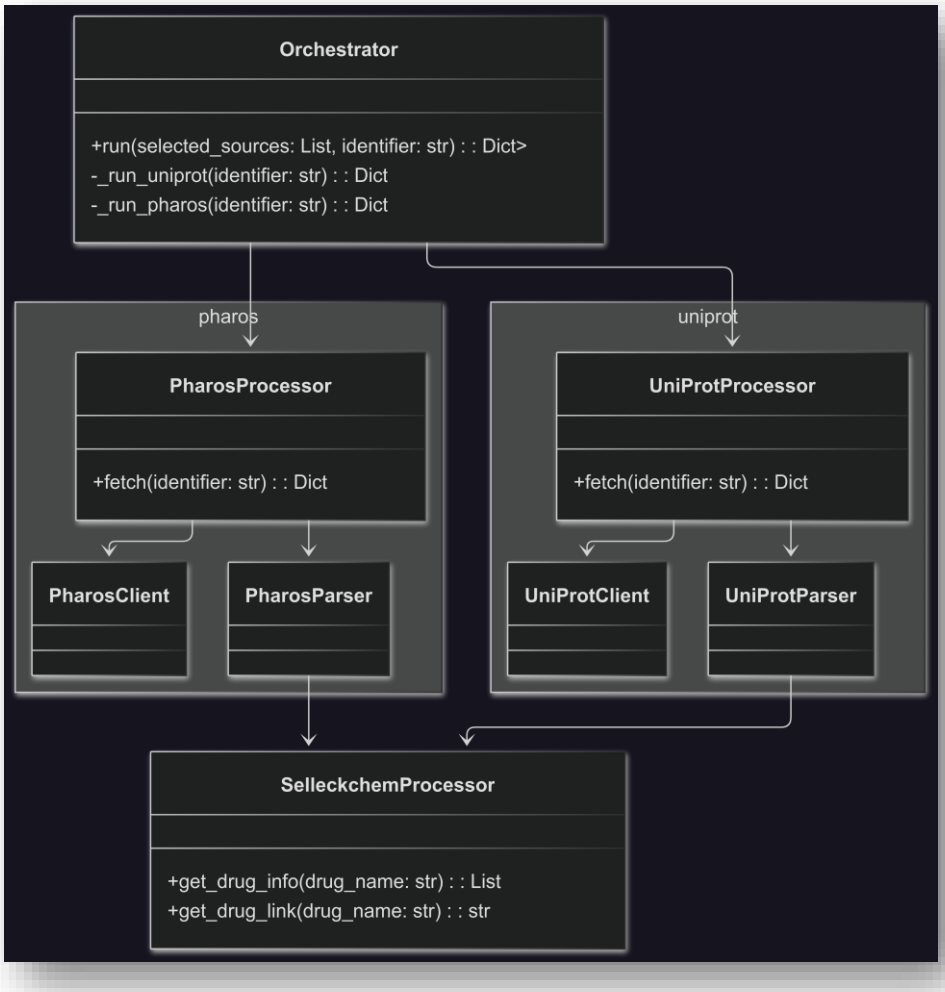



Figura 25. Diagrama de clases conceptual indicando el componente de nivel superior llamado Orquestador, DataSource y Procesadores

Frente a la imposibilidad de desacoplar las dependencias de datos se decidió pensar en diferentes mecanismos manteniendo los conceptos de Procesador Cliente y Parser.

### 4.2.3. Arquitectura V3

Para intentar solucionar el acoplamiento se estudiaron diferentes soluciones que ofrecían diferentes campos.

- **Big Data:**

El problema al que se estaba enfrentando eran similares a los del campo del BigData, la ingesta de datos, procesamiento y posterior visualización, tras investigar pipelines se encontró la similitud con los Pipelines ETL (Extract Transform Load), El acoplamiento en estos casos se producía también en la ingesta de los datos.

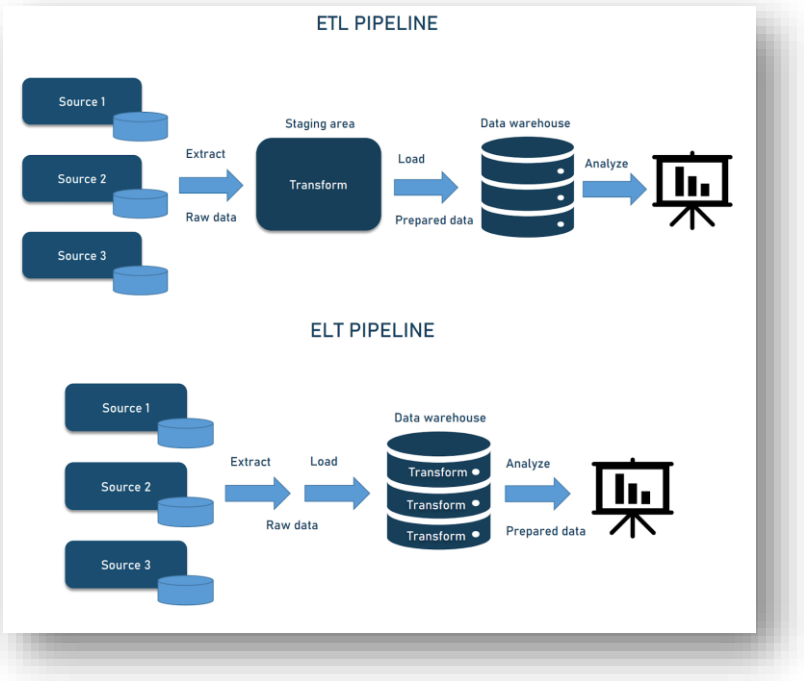



Figura 26. Representación del concepto del pipeline ETL. Fuente: https://estuary.dev/blog/data-pipeline-automation/

Se estudió la viabilidad de utilizar herramientas del ecosistema Apache, como el orquestador Apache Airflow, el cual ofrecía una solución robusta para la ejecución continua, programada y controlada de flujos de trabajo complejos.

Siguiendo la recomendación de Apache, parecía una herramienta viable:

"Airflow® is designed for finite, batch-oriented workflows. While you can trigger DAGs using the CLI or REST API, Airflow is not intended for continuously running, event-driven, or streaming workloads. That said, Airflow often complements streaming systems like Apache Kafka. Kafka handles real-time ingestion, writing data to storage. Airflow can then periodically pick up that data and process it in batch. If you prefer clicking over coding, Airflow might not be the best fit. The web UI simplifies workflow management, and the developer

experience is continuously improving, but defining workflows as code is central to how Airflow works — so some coding is always required." (34)

Pero debido a la complejidad que añadiría tener un orquestador tan complejo del cual usaríamos muy pocas características, se decidió descartar su uso.

- **Servicios usados en WEB:**

Se investigaron diferentes patrones que se usaban en la web para solucionar el problema y se encontraron dos patrones principales, el patrón Orchestrator y el patrón Choreography.

- **Patrón Orchestrator:**

Es un patrón utilizado en arquitectura web para gestionar múltiples servicios. Se basa en un componente central llamado *orquestador*, que coordina de manera inteligente las llamadas a los servicios específicos, controlando el flujo y la lógica de las interacciones.

- **Patrón Choreography:**

También gestiona servicios, pero lo hace sin un componente central. En este patrón, cada servicio conoce su papel y actúa de acuerdo con una lógica predefinida, reaccionando a eventos. En este caso, se usaría una orquesta (orquestador) para acoplar diferentes fuentes y centralizar su interacción.

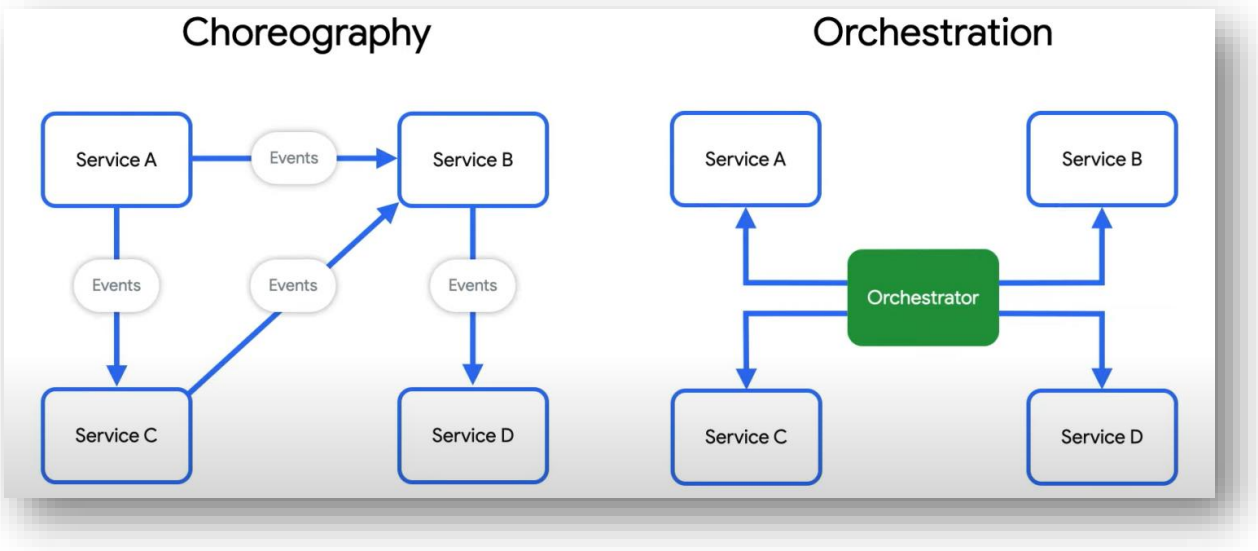



Figura 27. Representación del patrón Choreography y Orchestration. Fuente; https://www.linkedin.com/pulse/microservices-orchestration-vs-choreography-sachin-gupta

Se usará una combinación de ambas arquitecturas prescindiendo de ciertas cualidades como pueden ser la reacción por eventos o la necesidad de transaccionalidad.

En primer lugar, se introduce el concepto de **Workflow Step**, cada fuente tiene uno y este se representa la unidad que se encarga de comprobar si la fuente está disponible y procesar los datos el cual se comunica con el procesador de esta, para posteriormente devolver un dataframe con la información pedida.

Por encima está el **Workflow**, es la orquestra, en esta unidad se encuentran los diferentes Steps de las diferentes fuentes y aquí es donde se combinan los datos y se organizan las llamadas a las fuentes, de esta forma se consigue controlar el acoplamiento para que solo este en este módulo.

El encargado de gestionar cada Workflow es el **Orquestrador**, el cual se encarga de seleccionar entre diferentes Workflows y prepararlos para hacer las llamadas correspondientes, este se encarga de usar un Workflow o no dependiendo de la disponibilidad de sus Steps o fuentes, además se encarga de suministrar el campo de búsqueda al Workflow que a su vez se lo suministra a los Steps y estos a sus respectivos procesadores.

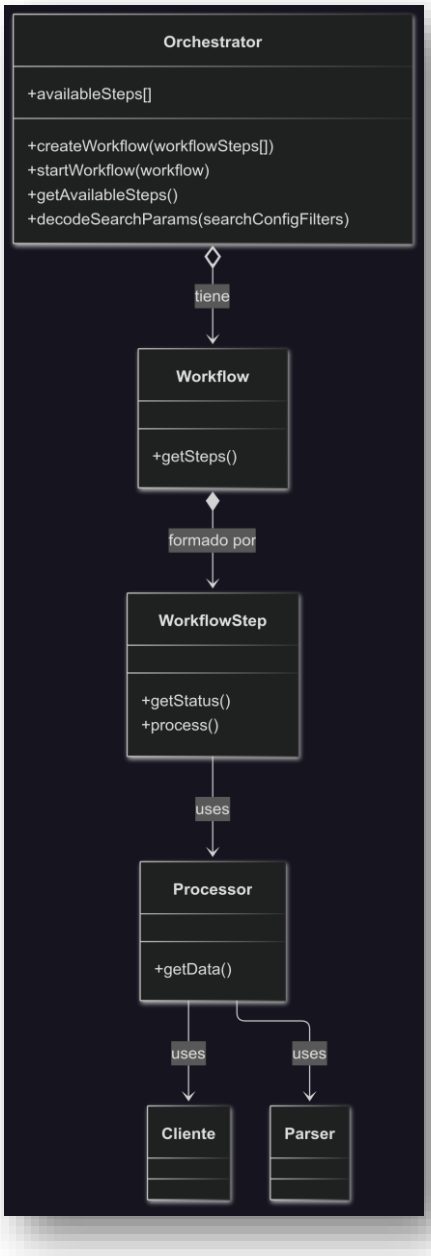



Figura 28. Diagrama de clases conceptual indicando como se relaciona el Orquestador con los Workflow y los Workflow Steps

## 4.3. Refinamiento de la Arquitectura siguiendo los Principios SOLID

Siguiendo los principios SOLID se refinó la arquitectura de forma que se garantizara un diseño robusto, mantenible y extensible para el sistema de Workflows de procesamiento de datos biomédicos. Este refinamiento permitió crear una estructura que respeta las mejores prácticas del diseño orientado a objetos y facilita la evolución del sistema.

### 4.3.1. Aplicación de los Principios SOLID

#### 4.3.1.1. Principio de Responsabilidad Única (SRP)

Siguiendo los principios SOLID se refinó la arquitectura de forma que se asignara una responsabilidad específica a cada componente:

- **Orchestrator**: Exclusivamente responsable de la orquestación de workflows y gestión de pasos disponibles.
- **BaseWorkflow**: Clase base que define la estructura fundamental para todos los workflows.
- **Workflow**: Implementación concreta que concentra la lógica de coordinación y secuenciación de pasos.
- **BaseWorkflowStep**: Clase base que establece el comportamiento fundamental para todos los pasos.
- **WorkflowStep**: Implementación especializada en la ejecución individual de cada paso del proceso.
- **Processor**: Dedicado exclusivamente al procesamiento y transformación de datos.
- **Cliente**: Maneja únicamente las comunicaciones con servicios externos.
- **Parser**: Especializado en el parseo y transformación de formatos de los datos.

#### 4.3.1.2. Principio Abierto/Cerrado (OCP)

Siguiendo los principios SOLID se refinó la arquitectura de forma que se pudiera extender la funcionalidad sin modificar el código existente:

- Se establecieron las clases base BaseWorkflow y BaseWorkflowStep como fundamento para extensiones.
- El sistema permite la incorporación de nuevos tipos de workflows heredando de BaseWorkflow.
- Los nuevos pasos de procesamiento se pueden agregar mediante herencia de BaseWorkflowStep.

La arquitectura está abierta a extensiones, pero cerrada a modificaciones en las clases base.

#### 4.3.1.3. Principio de Sustitución de Liskov (LSP)

Siguiendo los principios SOLID se refinó la arquitectura de forma que se garantizara la intercambiabilidad de implementaciones:

- Workflow puede sustituir completamente a BaseWorkflow sin comprometer la funcionalidad.
- WorkflowStep es completamente intercambiable con BaseWorkflowStep.
- Las clases derivadas respetan fielmente el comportamiento definido por sus clases base.
- Cualquier instancia de las clases base puede ser reemplazada por sus implementaciones concretas.

#### 4.3.1.4. Principio de Segregación de Interfaces (ISP)

Siguiendo los principios SOLID se refinó la arquitectura de forma que se crearan clases base cohesivas y específicas:

- BaseWorkflow contiene exclusivamente métodos relacionados con la gestión de workflows (getSteps(), getAvailableSteps()).
- BaseWorkflowStep se limita a operaciones esenciales de ejecución (getStatus(), process()).
- Las clases base son granulares y evitan dependencias innecesarias en las clases derivadas.
- Cada clase base proporciona solo la funcionalidad que realmente necesitan sus implementaciones.

#### 4.3.1.5. Principio de Inversión de Dependencias (DIP)

Siguiendo los principios SOLID se refinó la arquitectura de forma que se invirtieran las dependencias hacia abstracciones:

- Orchestrator depende de la clase base BaseWorkflow en lugar de implementaciones concretas.
- Workflow se compone de elementos BaseWorkflowStep, favoreciendo la flexibilidad.
- Las dependencias se orientan hacia las clases base abstractas, no hacia implementaciones específicas.
- El acoplamiento se reduce al depender de abstracciones estables.

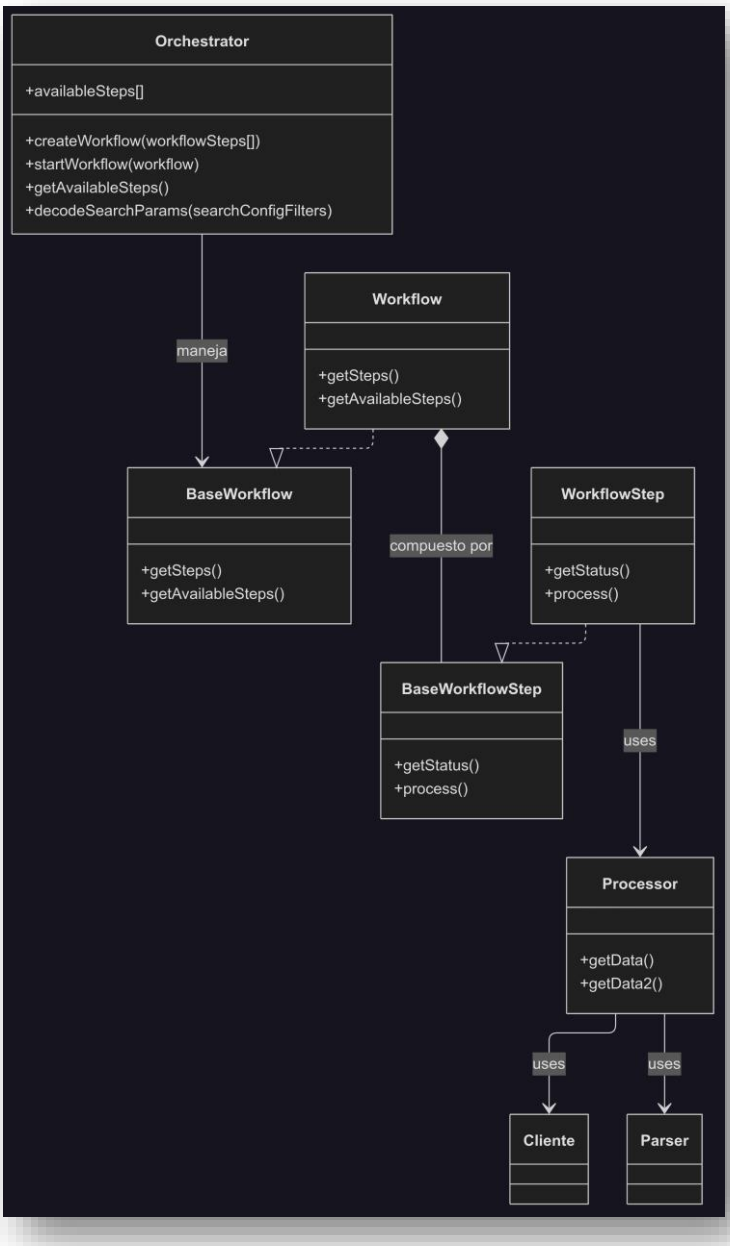



Figura 29. Diagrama de clases conceptual de la arquitectura refinada.

Para explicar el diseño final del sistema se irá explicando desde un nivel alto iterando por los distintos módulos del sistema.

En primer lugar, el sistema se divide en el siguiente árbol:

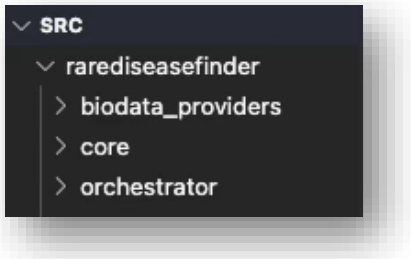



Figura 30. Representación de la jerarquía de módulos del código fuente.

El proyecto a alto nivel sigue una arquitectura por capas donde se separan las responsabilidades en 3 módulos principales.

En la siguiente figura se detalla la relación de cada clase:

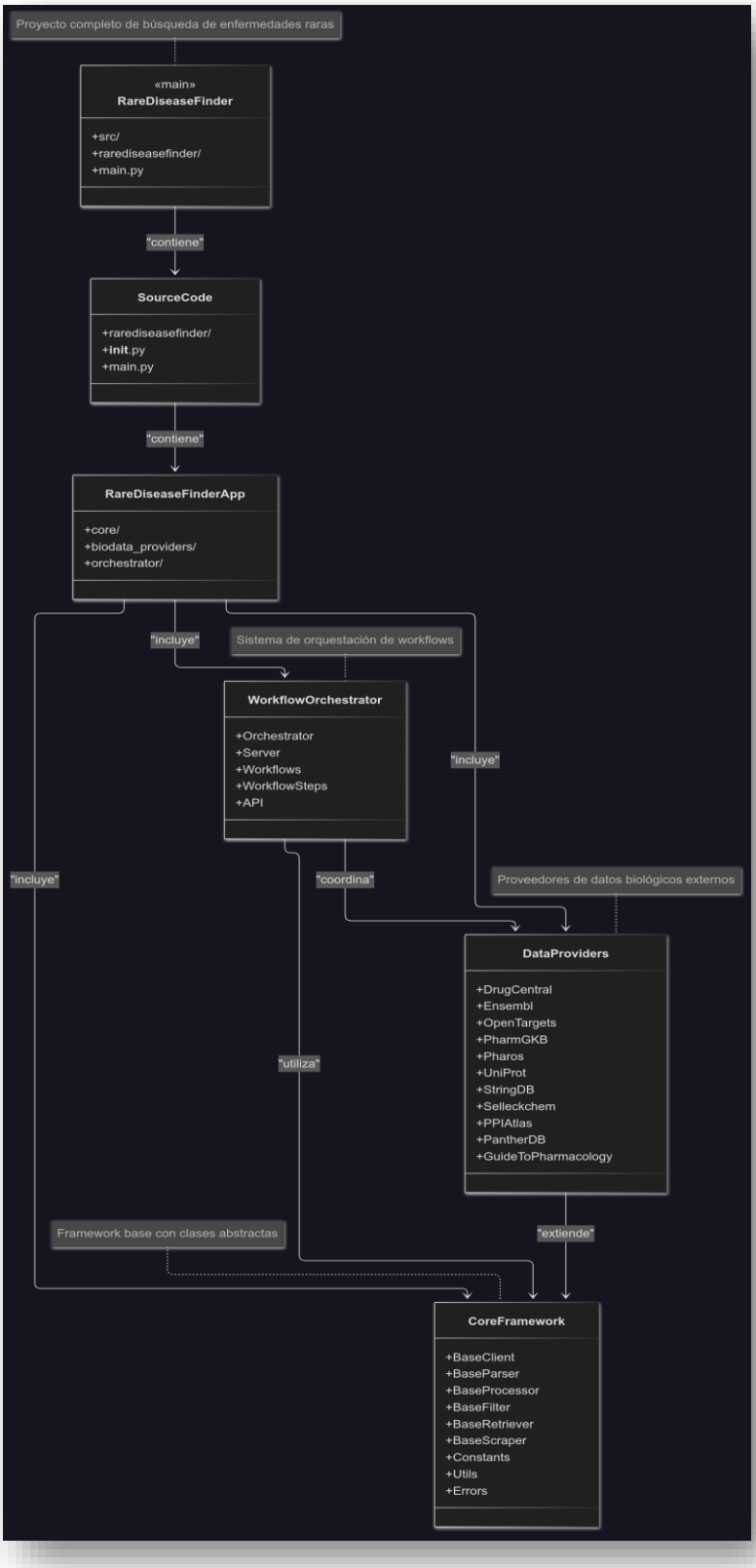



Figura 31. Descripción de la jerarquía del sistema por capas.

- **Biodata Providers:**

  Contiene las implementaciones de las fuentes, en este módulo cada fuente tiene las implementaciones del Procesador, Parser y Cliente.

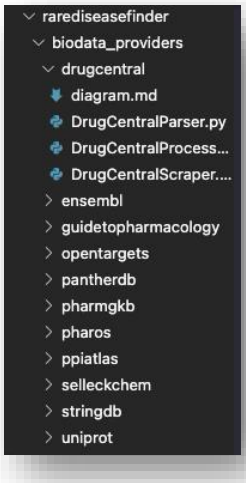



Figura 32. Árbol con la estructura del módulo Biodata_Providers

Se puede observar en la siguiente el flujo de los datos para una fuente en específico como puede ser Pharos:

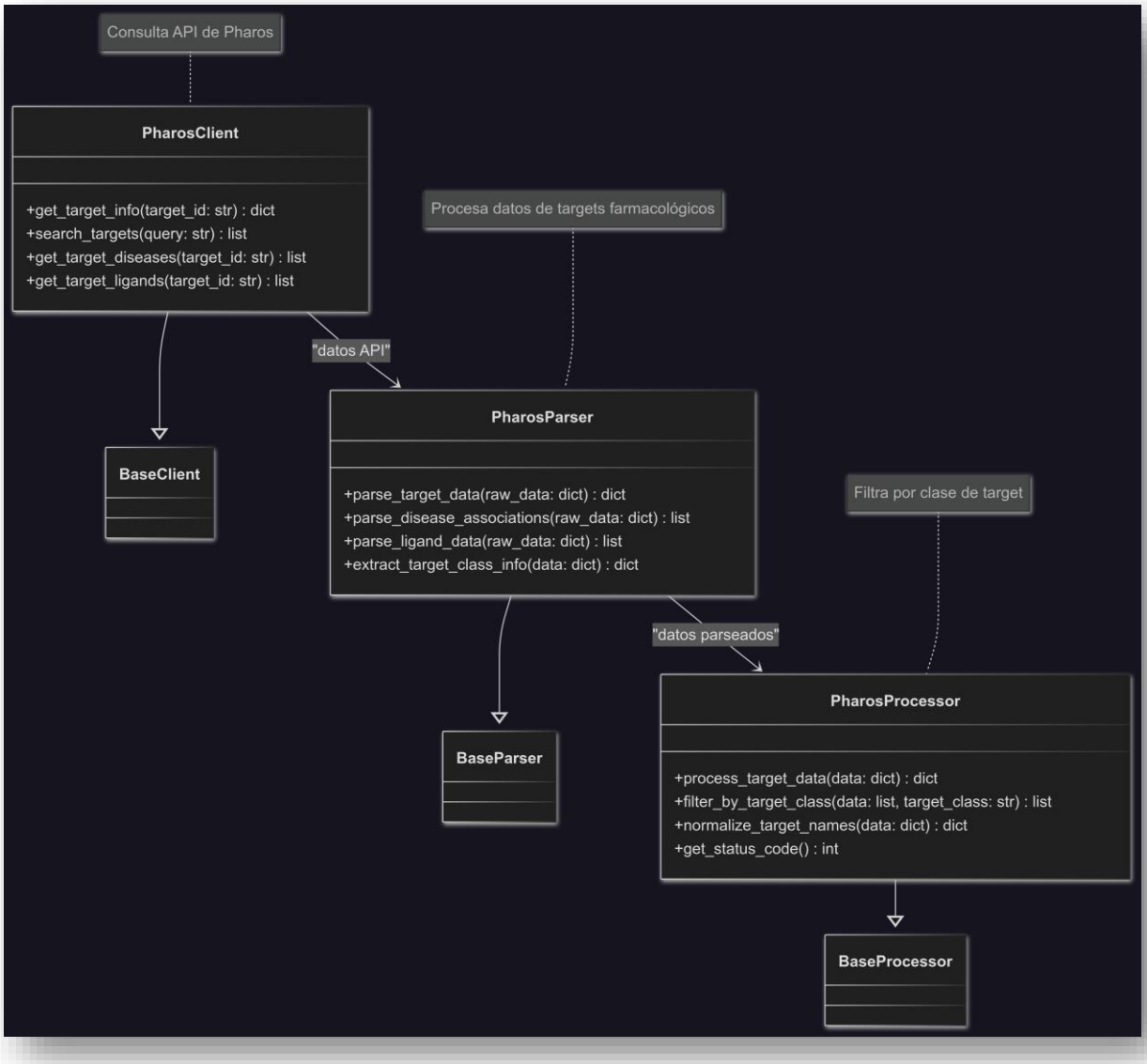



Figura 33. Diagrama ilustrando flujo de datos entre objetos Cliente, Parser y Procesador para la fuente Pharos.

- **Core**:

  Contiene las clases Base de Procesador, Cliente y Parser, donde se implementan los métodos mínimos que tiene cada entidad, también tiene clases donde se implementan utilidades como puede ser parsear Dataframes a objetos JSON.

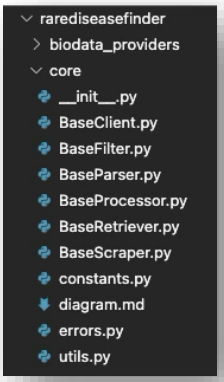



Figura 34. Árbol con la estructura del módulo Core

Se destaca BaseRetriever que sirve como clase padre de BaseClient y BaseScrapper, se decidió usar esta abstracción para separar las responsabilidades de cada clase, de esta forma BaseClient se centra en implementar mecanismos para las peticiones a las APIs y BaseScrapper implementa mecanismos para usar Sellenium, de esta forma para el nivel superior, el Procesador, no hay problema para hacer el fetch en fuentes que tienen distinta naturaleza de obtención de datos.

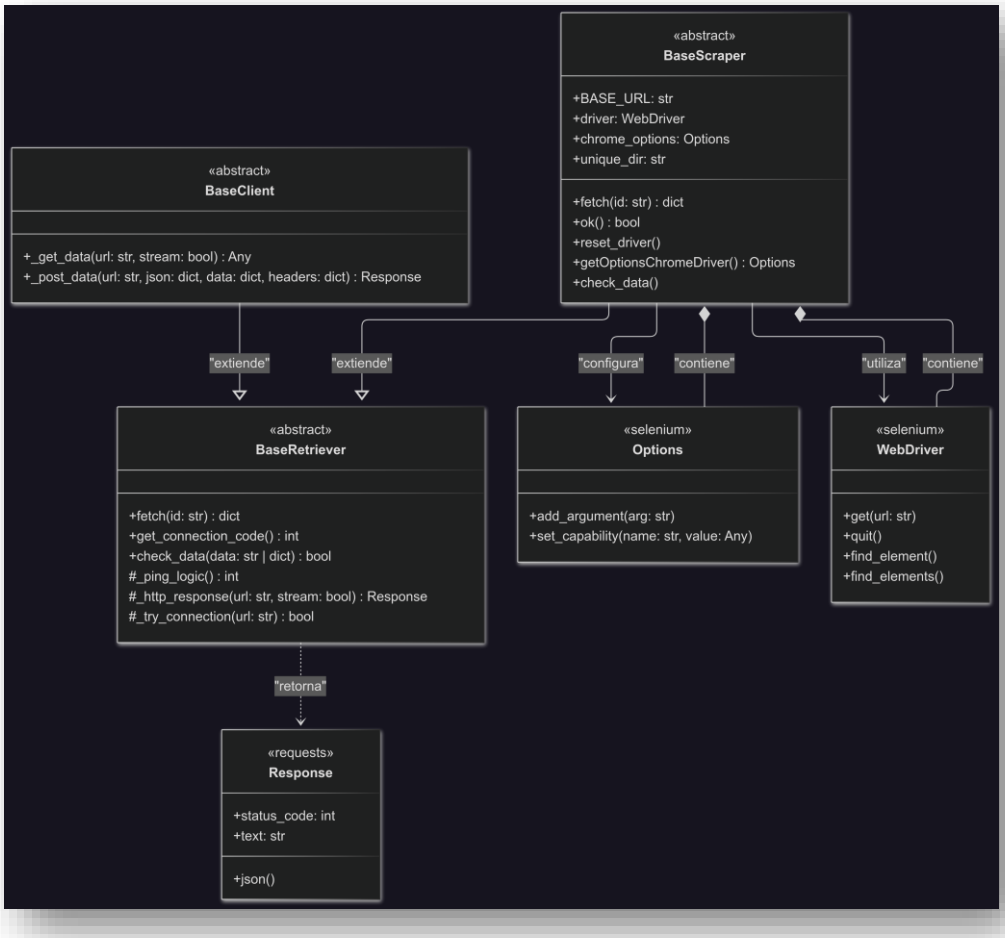



Figura 35. Diagrama de clases indicando la herencia de Base Retriever

- **Orquestrator**:

  Contiene las diferentes clases para la gestión y uso de los workflows, además de tener la implementación del servidor Flask para la API.

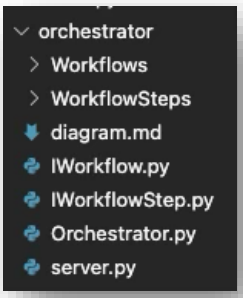



Figura 36. Árbol con la estructura del módulo del Orquestrador

En la sección implementación se explicará en detalle cómo se gestionan los diferentes tipos de Workflows junto a los WorkflowSteps.

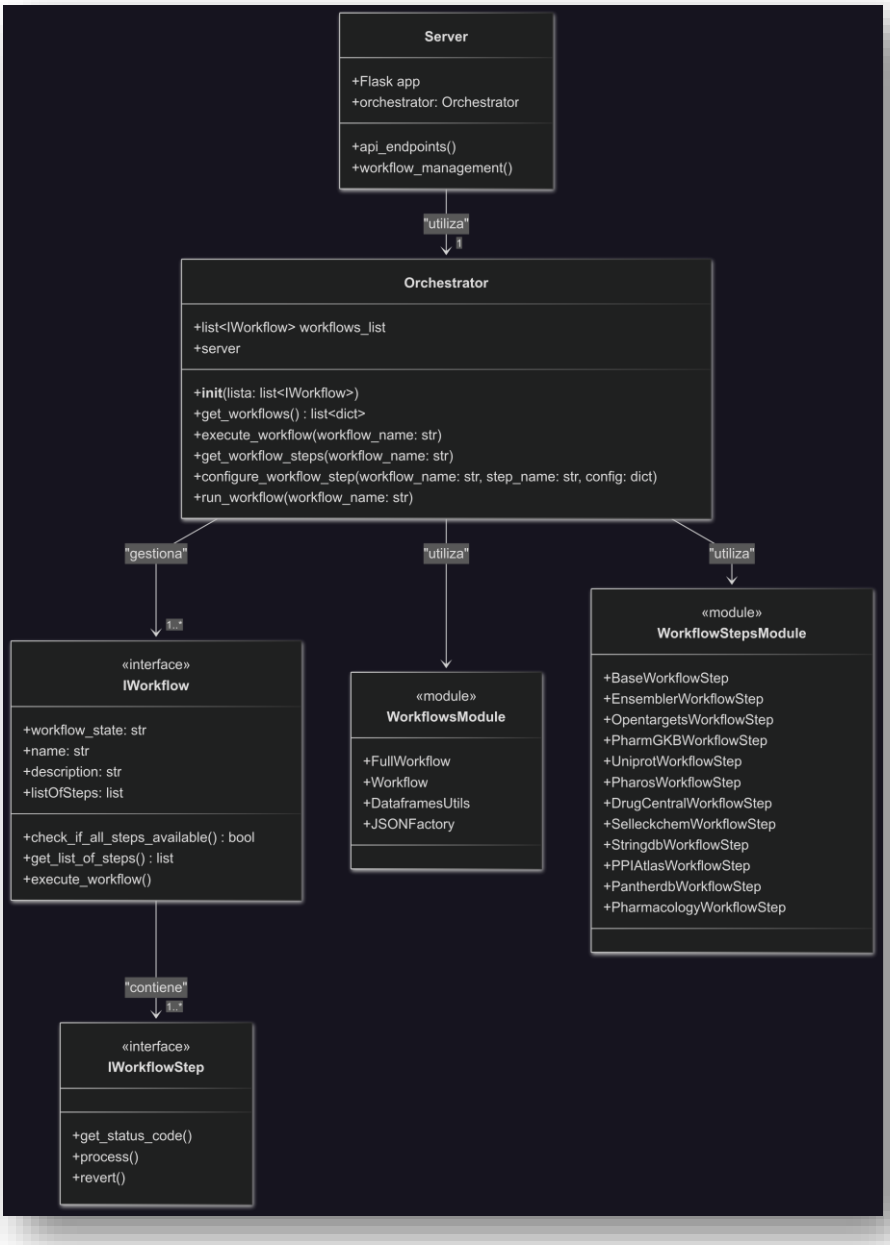



Figura 37. Diagrama de clases indicando la arquitectura de alto nivel.

## 4.4. Funcionamiento interno

En este apartado se profundizará en el diseño de ciertos componentes del sistema, destacando aquellas soluciones que se han implementado de forma elegante y eficiente.

### 4.4.1. Comunicación entre Workflows, Steps y Procesadores

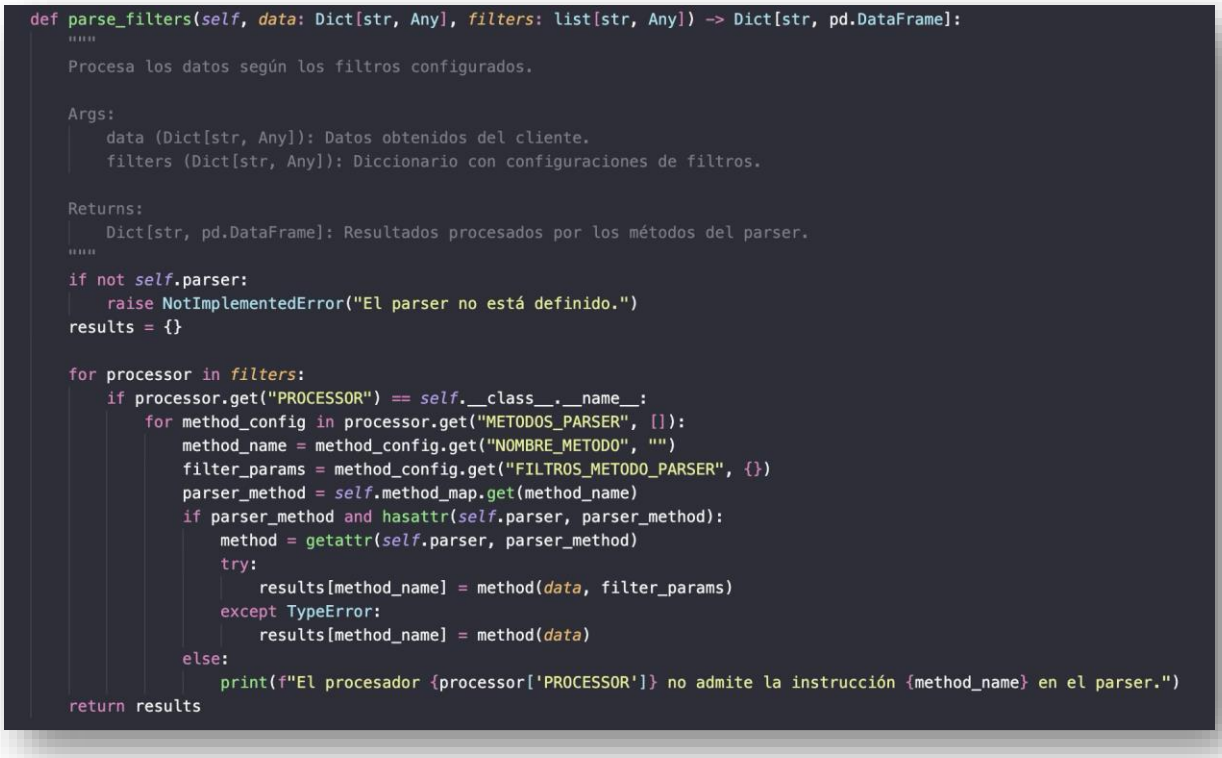
Para explicar este apartado es necesario explicar en primer lugar el mecanismo que usan los objetos heredados de BaseWorkflow, BaseWorkflowStep y BaseProcesador para comunicarse, el procesador necesita recibir un objeto JSON que contiene el campo de búsqueda, el procesador que necesita ser llamado, métodos del procesador que se van a llamar y los respectivos parámetros para estos métodos, para la implementación se diseñaron sistemas para parsear el JSON junto a métodos para limitar el control de procesador solo a los métodos que estuviesen declarados en su mapa de métodos, sobre este mapa se hablará más adelante.


Figura 38. Documento JSON que espera el procesador para poder ejecutarse.

Como se comentaba los procesadores esperan recibir las instrucciones mediante un objeto JSON que sigue la estructura de la figura 39, cada procesador tiene un mapa de métodos los cuales están mapeados con las funciones del parser, al decodificar el JSON, si el nombre del método está en el mapa del procesador se hará una llamada a ese método en específico.


Figura 39. Fragmento de código de la clase PharosProcessor indicando el mapa de métodos

La clase BaseProcessor implementa el mecanismo que comparten todos los hijos para la decodificación del JSON, este método itera sobre el JSON comprobando si el procesador que se va a llamar es el mismo que está recibiendo el objeto, comprueba que los métodos están en el mapa y hace las llamadas, como habíamos mencionado, algunos métodos tienen parámetros que pueden llegar mediante el JSON, en ese caso se hace una comprobación también en caso de que el método no admita parámetros.



```python
def parse_filters(self, data: Dict[str, Any], filters: list[str, Any]) -> Dict[str, pd.DataFrame]:
    """
    Procesa los datos según los filtros configurados.

    Args:
        data (Dict[str, Any]): Datos obtenidos del cliente.
        filters (Dict[str, Any]): Diccionario con configuraciones de filtros.

    Returns:
        Dict[str, pd.DataFrame]: Resultados procesados por los métodos del parser.
    """
    if not self.parser:
        raise NotImplementedError("El parser no está definido.")
    results = {}

    for processor in filters:
        if processor.get("PROCESSOR") == self.__class__.__name__:
            for method_config in processor.get("METODOS_PARSER", []):
                method_name = method_config.get("NOMBRE_METODO", "")
                filter_params = method_config.get("FILTROS_METODO_PARSER", {})
                parser_method = self.method_map.get(method_name)
                if parser_method and hasattr(self.parser, parser_method):
                    method = getattr(self.parser, parser_method)
                    try:
                        results[method_name] = method(data, filter_params)
                    except TypeError:
                        results[method_name] = method(data)
                else:
                    print(f"El procesador {processor['PROCESSOR']} no admite la instrucción {method_name} en el parser.")
    return results
```

Figura 40 Implementación del método de parseo del JSON.

Los procesadores son gestionados por los Steps, en este caso PharosStep es un intermediario que espera este filtro del nivel superior que en este caso es el BaseWorkflow, a continuación, se hablará como se compone un Workflow.

Como mencionábamos antes el concepto de Workflow responde a la necesidad de solventar el problema del acoplamiento, al tener varías fuentes, se crea esta entidad donde se realizan las tareas de preparar los ficheros JSON que se suministrarán a los Steps y estos a los procesadores y donde se realizará combinación de datos y preparación del informe en formato JSON, en la siguiente figura se muestra un ejemplo de dependencia de datos que existe.

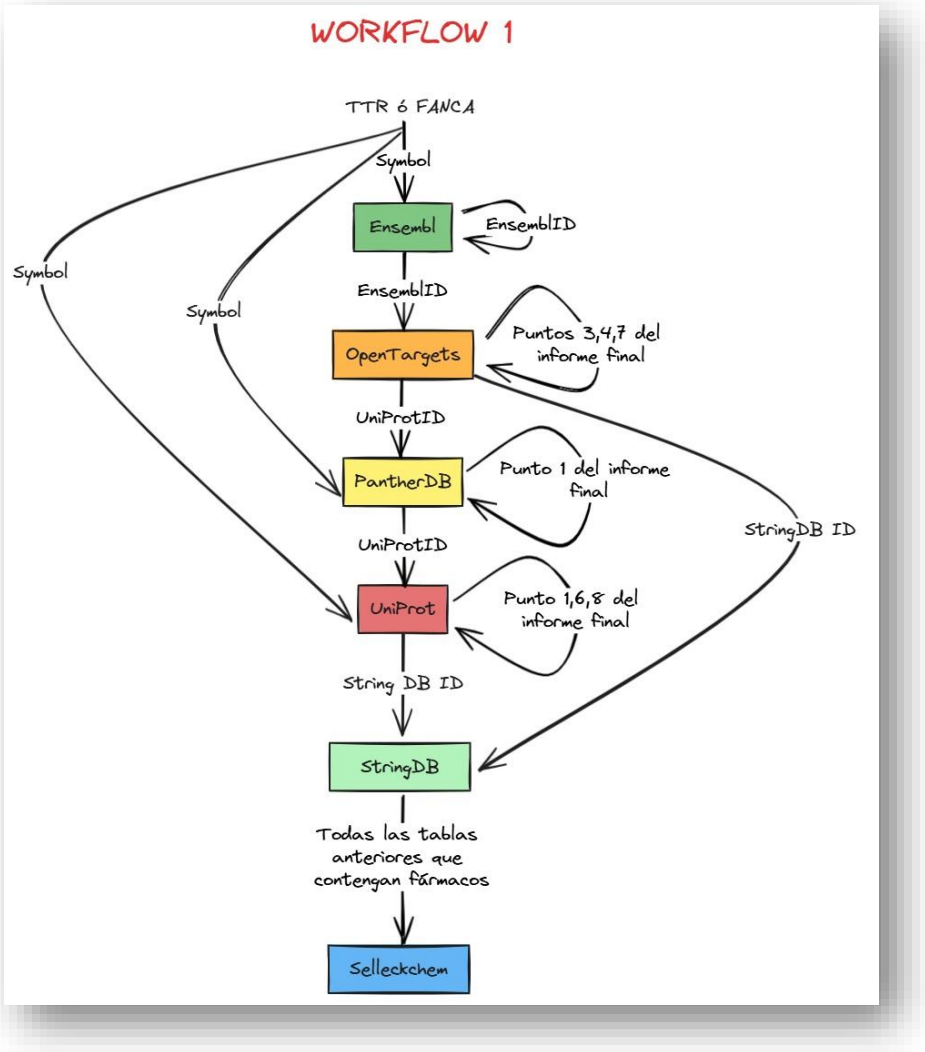



Figura 41. Diagrama con las dependencias de fuentes.

### 4.4.2. Funcionamiento filtros editables por el usuario

El sistema dispone de la funcionalidad de personalizar los métodos que se van a llamar desde el Front, este requisito generaba problemas ya que había métodos mínimos que siempre debían ejecutarse por dependencia de datos, para solucionar esto se implementó el concepto de métodos mínimos, los cuales están declarados en el Workflow y estos se van a ejecutar siempre, se implementaron mecanismos para calcular los métodos opcionales seleccionables por el usuario, mediante el cálculo de la diferencia del conjunto de todos los métodos (mapa de método del procesador) y el conjunto de métodos mínimos del Workflow, continuación se adjunta una imagen con el código que realiza la operación.

```python
def generate_optional_methods(self) -> dict:
    """…
    optional_methods = {}

    # Iterate over minimum_methods_by_step keys to handle each configuration
    for step_key, step_config in self.minium_methods_by_step.items():
        step_name = step_config.get("step_name", "")
        processor = step_config.get("processor", "")
        minimum_methods = step_config.get("methods", [])
        minimum_method_ids = {method["METHOD_ID"] for method in minimum_methods}

        # Find the corresponding step instance in listOfSteps
        step_instance = None
        for step_dict in self.listOfSteps:
            if step_name in step_dict:
                step_instance = step_dict[step_name]
                break

        if step_instance is None:
            print(f"\033[33mWarning: Step instance not found for {step_name}\033[0m")
            continue

        # Get all available methods from the step
        all_methods = step_instance.get_method_map()
        all_method_ids = set(all_methods.keys())

        # Calculate optional methods (difference between all methods and minimum ones)
        optional_method_ids = all_method_ids - minimum_method_ids

        # Create optional methods structure for each unique step_key
        if optional_method_ids:  # Only if there are optional methods
            optional_methods[step_key] = {
                "step_name": step_name,
                "processor": processor,
                "methods": [
                    {
                        "METHOD_ID": method_id,
                        "METHOD_PARSER_FILTERS": {}
                    }
                    for method_id in optional_method_ids
                ]
            }
        else:
            # Even if there are no optional methods, create empty structure
            optional_methods[step_key] = {
                "step_name": step_name,
                "processor": processor,
                "methods": []
            }

    return optional_methods
```

Figura 42. Código del método de creación de métodos opcionales.

El Workflow además implementa funciones para combinar datos de diferentes fuentes, un sistema de generación de un informe en formato JSON que contiene la información final que se envía al Font para su representación y por último un sistema de staging para gestionar los diferentes estados en los que está el Workflow.

El informe final se genera en el Workflow mediante el uso del módulo JSONFactory, este ofrece mecanismos para crear la estructura del informe final junto al ordenamiento del contenido de las fuentes.

El Workflow hace también uso del módulo DataframesUtils que contiene funciones para facilitar el manejo de Dataframes, conversiones de datos, transformaciones y filtrado de estos.

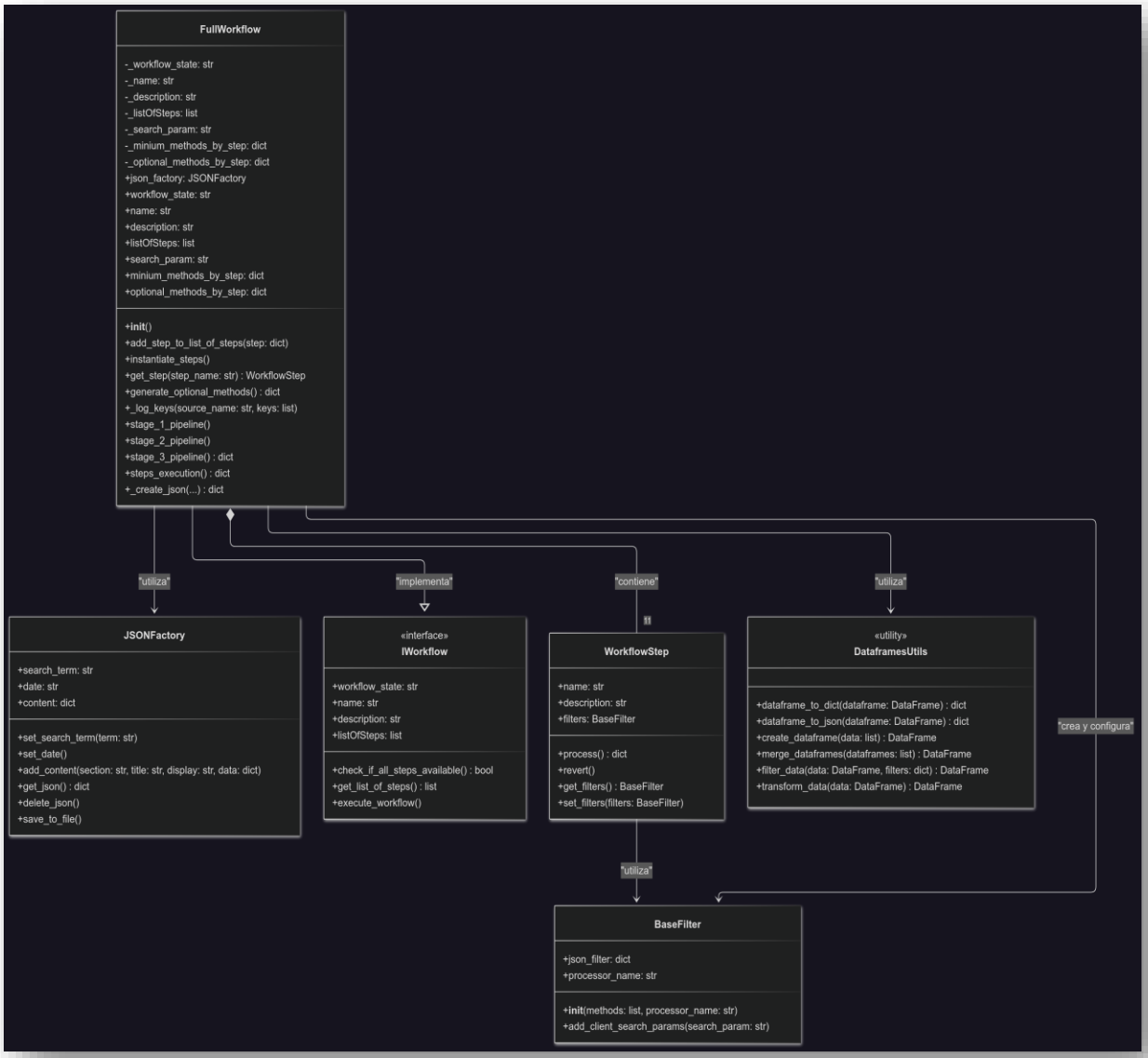



Figura 43. Diagrama de clases de Workflow y sus dependencias.

### 4.4.3. Sistema de Stages y por qué es necesario en el sistema

El concepto de Stage surge ante la necesidad de poder controlar el estado del Workflow, debido a que se tiene acceso al sistema mediante API, se tiene que controlar el estado en el que está el Workflow para evitar que se hagan llamadas sin un campo de búsqueda o cuando el Workflow está en un estado de procesamiento.

El sistema puede estar en 3 estados diferentes, Stage 1, Stage 2 y Stage 3

- **Stage 1**: Es el estado en el que el Workflow establece los métodos mínimos, y crea los filtros JSON de cada Step, una vez creados se mantienen en el estado. Desde el Orquestador se pueden hacer peticiones GET para saber el estado del Workflow, los métodos mínimos, opcionales, filtros…, además de ver el estado de disponibilidad de las fuentes de este.

- **Stage 2**: En este estado el Workflow espera recibir cambios desde el Front, en este caso, va a actualizar sus filtros, en el caso de que vengan métodos opcionales, filtros de métodos, o el usuario establezca el campo de búsqueda.
- **Stage 3**: En este estado el Workflow empieza a procesar, se ejecuta la combinación de datos junto y se exporta a un archivo JSON que contiene la información del informe, la estructura de este JSON se explicará en el punto 5.1.1
-

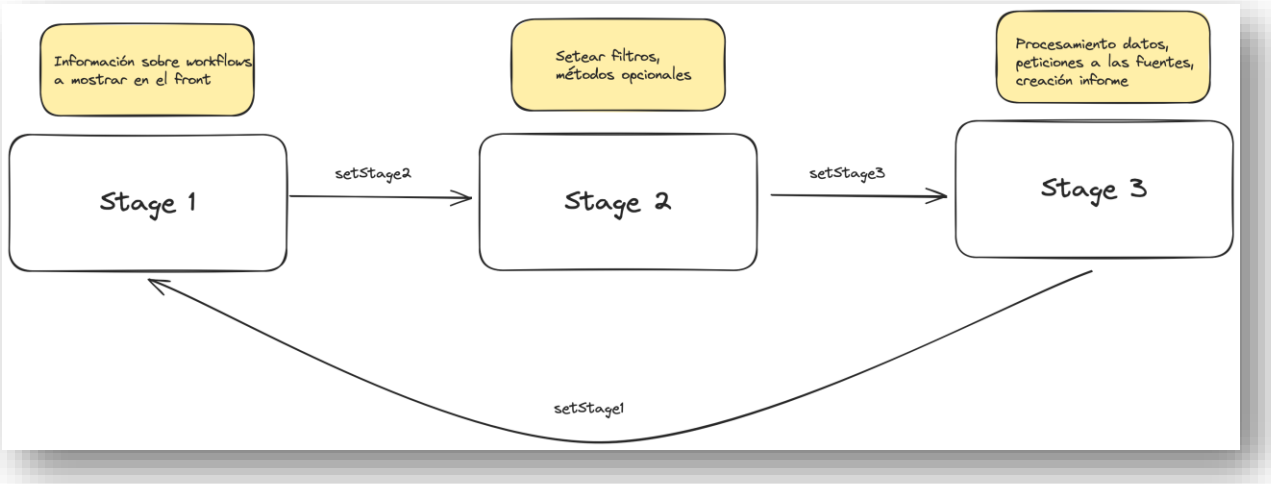



Figura 44. Diagrama flujo conceptual de Stages.

# 5. Implementación de la solución

En este capítulo se va a hablar sobre los entregables frutos del proyecto, uno de propia construcción, la API del proyecto y otra realizada en colaboración con el Grupo de Aplicación de Telecomunicaciones Visuales (GATV) de la UPM, la interfaz gráfica.

### 5.1.1. Implementación de la API en Flask

Para implementar la API del proyecto se optó por utilizar Flask, un micro-framework ligero y flexible que facilitó su desarrollo.

La API cuenta con tres módulos de endpoints principales, cada uno relacionado con las acciones específicas de los Stages descritos en la sección 4.5.3.

Además, la API sigue la especificación OpenAPI, lo que permite una mejor documentación, validación y mantenimiento de los servicios expuestos.

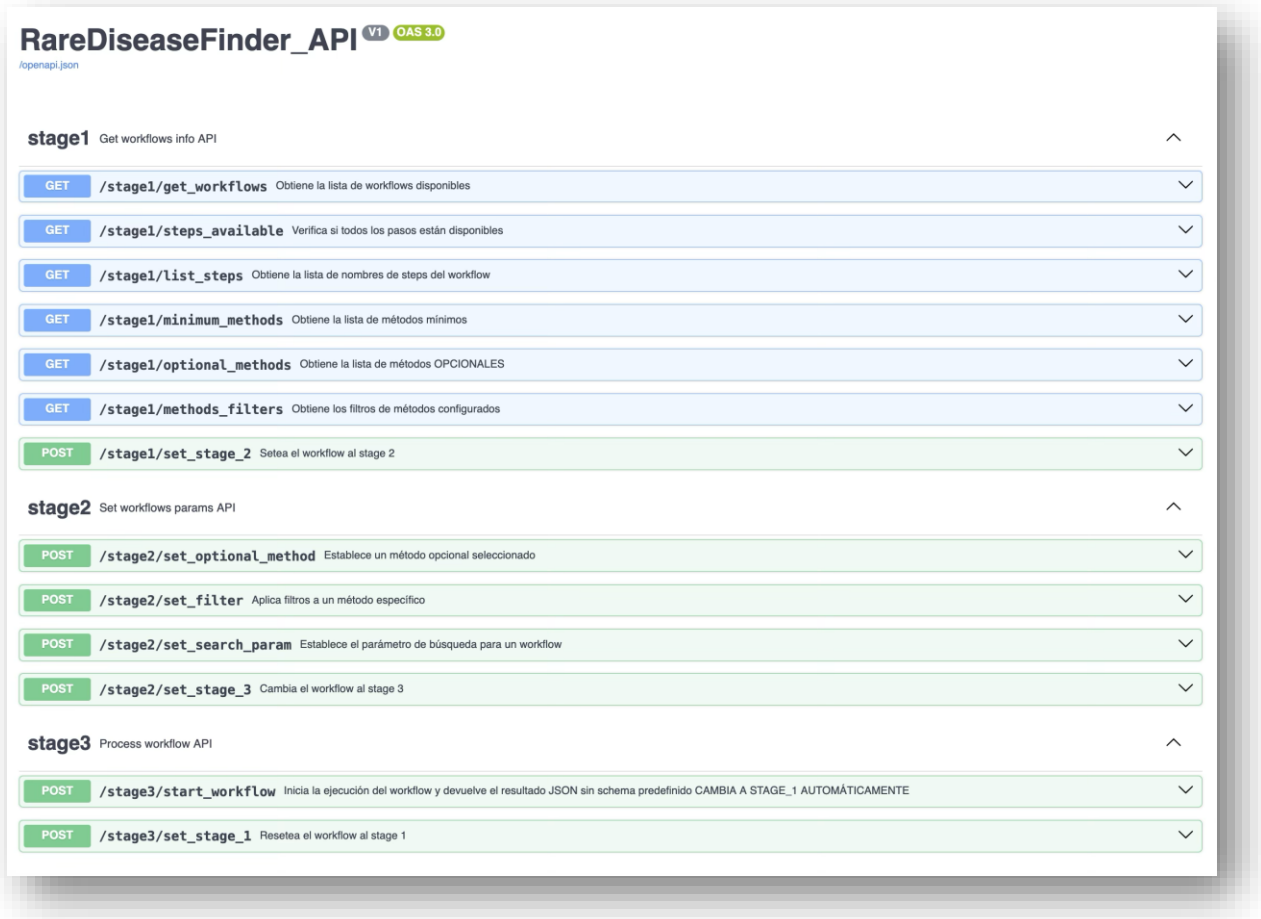



Figura 45. Swagger del proyecto.

#### 5.1.1.1. Endpoints Stage 1

En los endpoints del módulo Stage 1 tenemos diferentes acciones disponibles, como habíamos comentado el objetivo del estado Stage 1 es recibir peticiones de tipo GET, donde se envíen los Workflows disponibles, el estado de disponibilidad de los

Steps, la lista de Steps asociados a un Workflow especifico, la lista con métodos mínimos y opcionales y los filtros para un método de un Step en específico.

Además, cuenta con un endpoint de tipo POST por el cual se cambia el estado del Workflow a Stage 2.

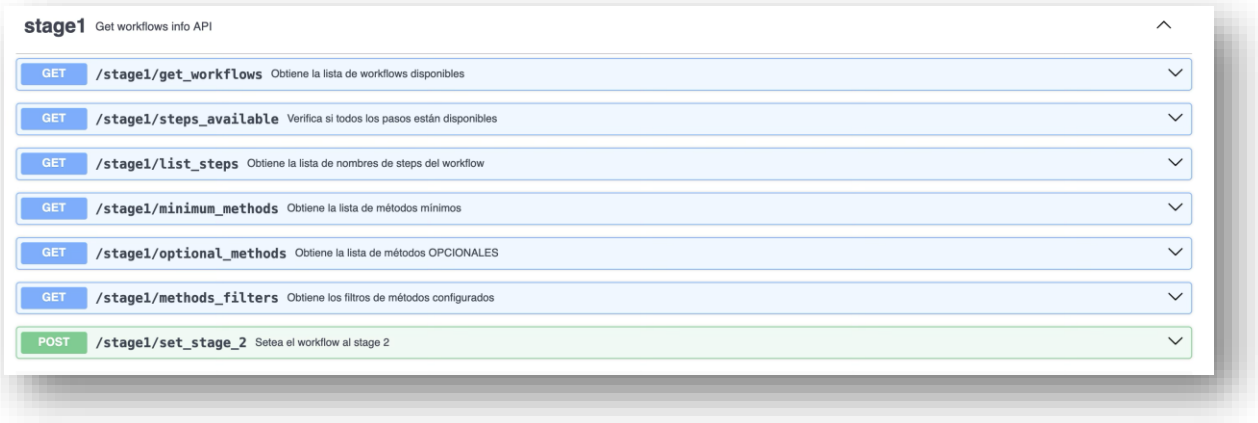



Figura 46. Endpoints del Stage 1.

Un ejemplo de petición con su respuesta se detalla en la imagen siguiente.

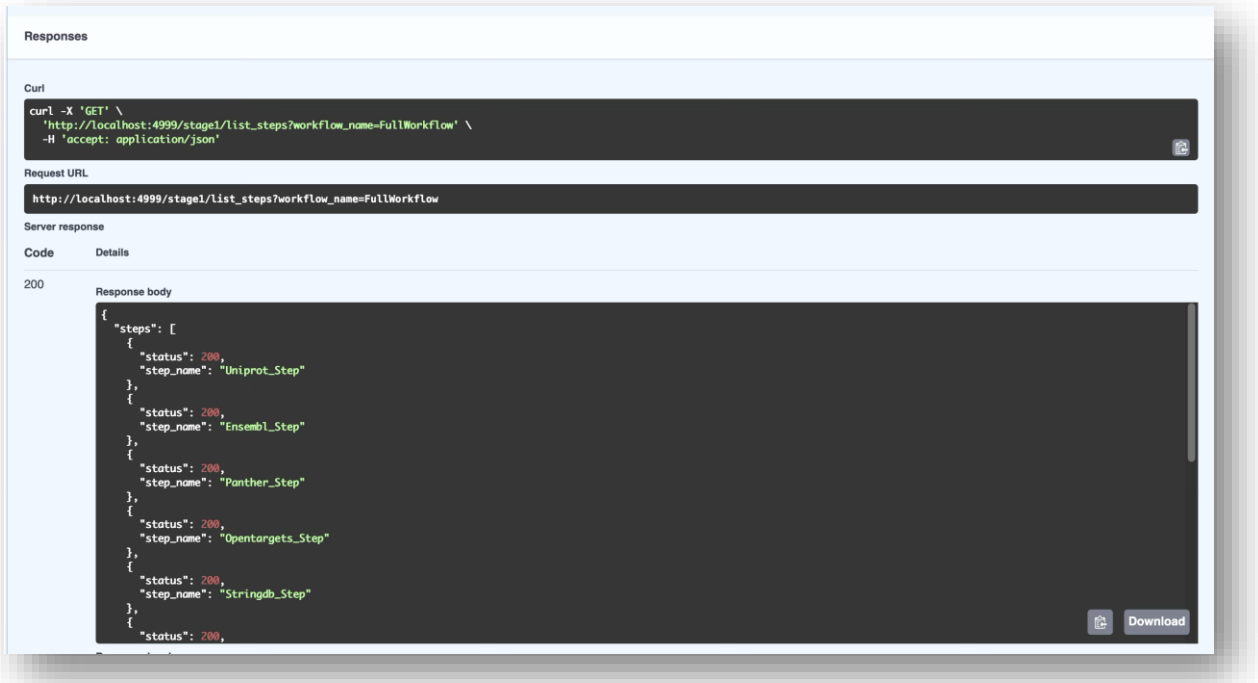



Figura 47. Respuesta a la petición list_steps.

#### 5.1.1.2. Endpoints Stage 2

Los endpoints del Stage 2 tienen el objetivo de modificar el estado del Workflow, cuenta con endpoints de tipo POST por los cuales se consigue establecer métodos opcionales que el usuario elije desde la interfaz gráfica, establecer filtros para métodos y establecer el parámetro de búsqueda.

Además, cuenta con un endpoint de tipo POST por el cual se cambia el estado del Workflow a Stage 3.

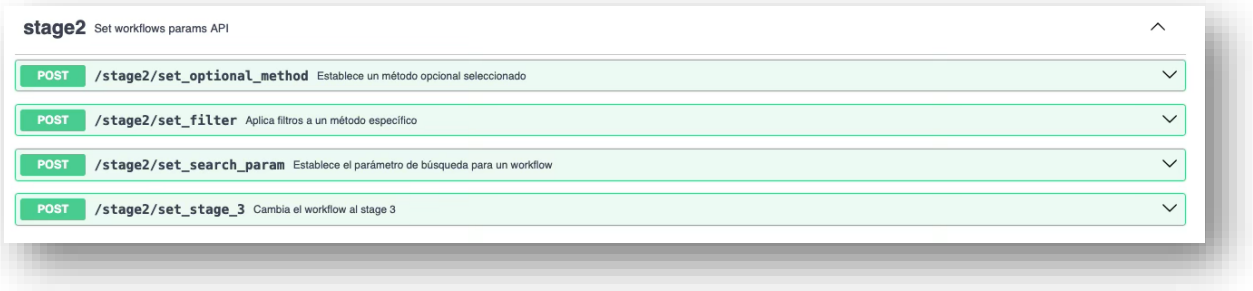



Figura 48. Endpoints del Stage 2.

#### 5.1.1.3. Endpoints Stage 3

Por último, los endpoints del Stage 3 tienen el objetivo de empezar el Workflow, el workflow al recibir la petición start_workflow, comienza a procesar, ejecutando la búsqueda con cada Step, procesando la salida de las distintas fuentes y generando el informe final.

Además, cuenta con un endpoint de tipo POST por el cual se cambia el estado del Workflow a Stage 1, Aunque al llamar a start_workflow, si se realiza correctamente la acción cambia automáticamente a Stage 1.

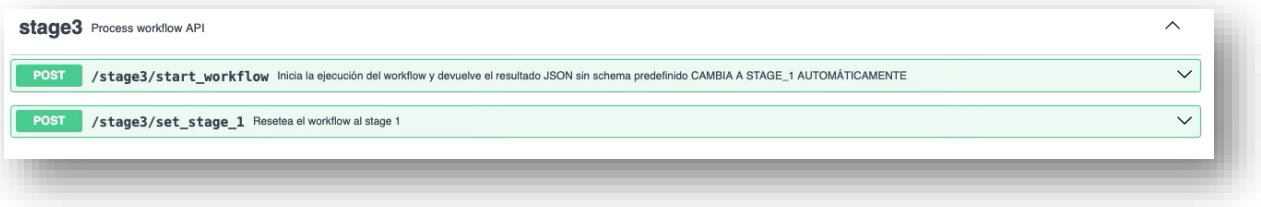



Figura 49. Endpoints del Stage 3.

El resultado de start_workflow es un documento en formato JSON que contiene la información del informe, la estructura del documento se compone de "search_term", el campo de búsqueda que se ha usado para realizar la búsqueda, "date", la fecha de generación del informe, y "categories", este último contiene una lista de objetos compuestos por diferentes secciones en el campo "section" y el contenido de las diferentes fuentes dentro de content.


Figura 50. Estructura del documento del informe.

El objeto “content” se compone de 3 diferentes campos, “title” donde se indica la Fuente y el método que se ha llamado, “display” usado para que el FrontEnd sepa si tiene que mostrar los datos en forma de tabla o sheet (“ficha”), por último, esta dispone de un campo “data” donde está la información recibida por la fuente, en este caso no sigue un schema predefinido.

```
{
  "section": "PATHWAYS",
  "content": [
    {
      "title": "Panther: Pathways",
      "display": "table",
      "data": [
        {
          "Pathway": "⚠️ No se han encontrado datos.",
          "Link": "⚠️ No se han encontrado datos."
        }
      ]
    },
    {
      "title": "OpenTargets: Pathways",
      "display": "sheet",
      "data": [
        {
          "Link": "https://reactome.org/content/detail/R-HSA-6783310",
          "Vía": "Fanconi Anemia Pathway",
          "Término de nivel superior": "DNA Repair"
        },
        {
          "Link": "https://reactome.org/content/detail/R-HSA-9833482",
          "Vía": "PKR-mediated signaling",
          "Término de nivel superior": "Immune System"
        }
      ]
    }
  ]
}
```


Figura 51. Estructura del objeto content.

### 5.1.2. Diagrama de secuencia de Stages

En esta sección se detallan los diagramas de secuencia del Stage respecto a la API:

En la guía del anexo se describe en detalle el intercambio de información en los distintos Stages.

- **Stage 1:**

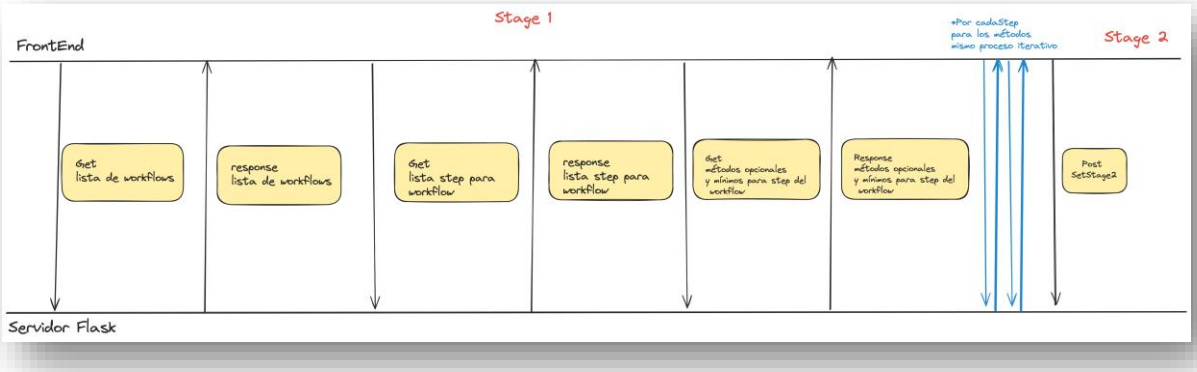



Figura 52. Diagrama de secuencia explicativo del Stage 1.

- **Stage 2:**

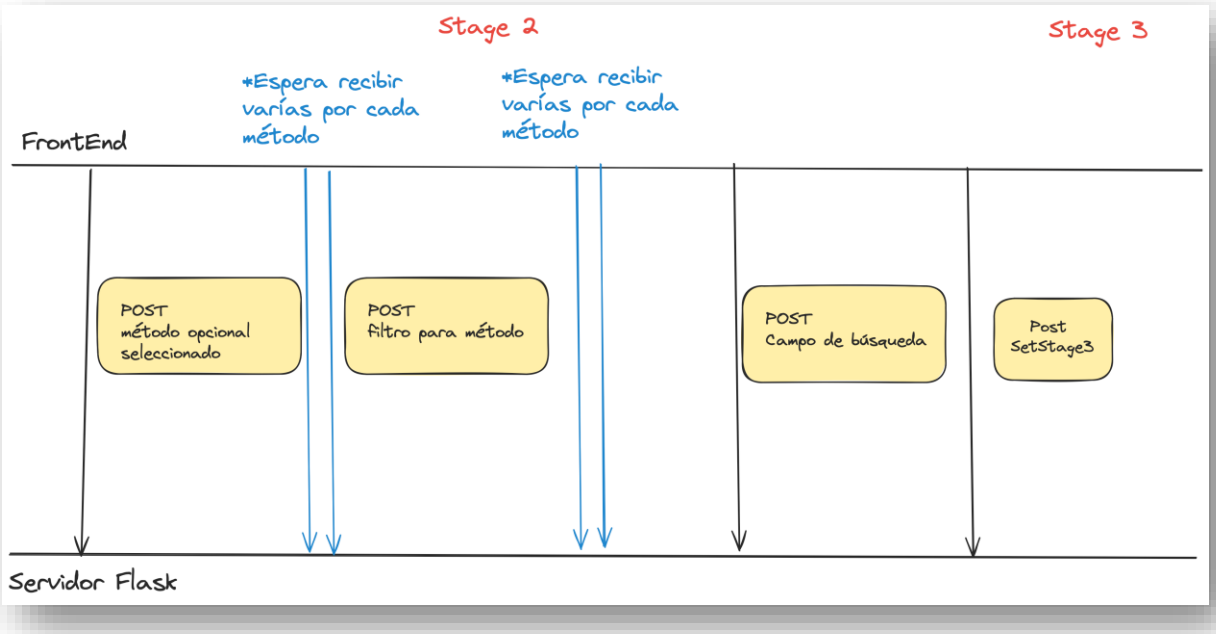



Figura 53. Diagrama de secuencia explicativo del Stage 1.2

- Stage 3:

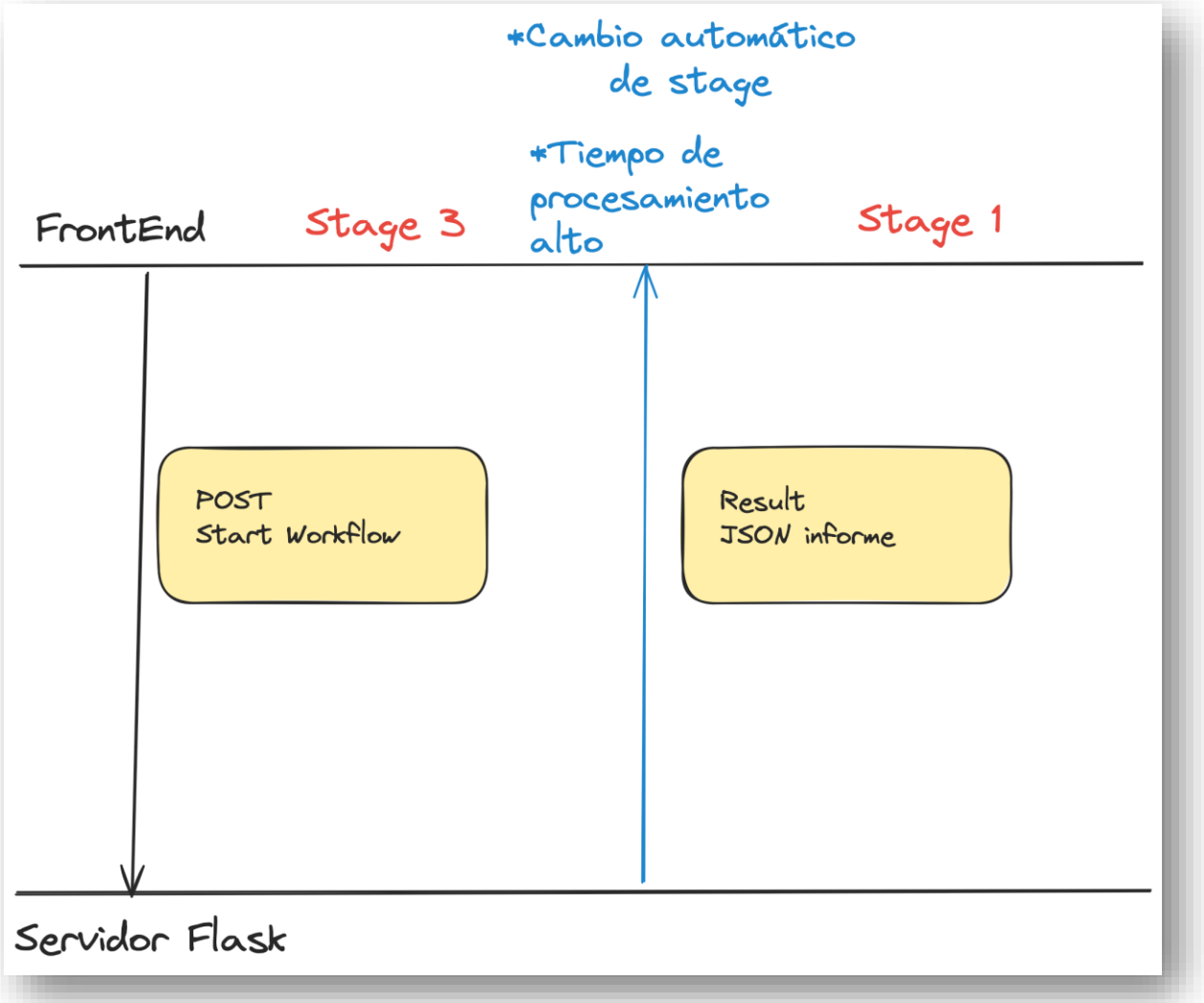



Figura 54. Diagrama de secuencia explicativo del Stage 1.3

### 5.1.3. Frontend del sistema

En esta sección se detalla el proceso de construcción de la interfaz gráfica del sistema. Se adjunta un vídeo como demostración y guía audiovisual de usuario en el anexo: Demo audiovisual del Frontend

#### 5.1.3.1. Construcción Interfaz propia con Tkinter

En una fase inicial del desarrollo, se planteó la creación de una interfaz gráfica de usuario (GUI) mediante la librería Tkinter, incluida en la biblioteca estándar de Python. Esta opción se valoró por su simplicidad de uso, portabilidad y por ofrecer

herramientas suficientes para construir tanto interfaces básicas como más complejas sin necesidad de dependencias externas.

El objetivo de esta interfaz era actuar como un mockup funcional, permitiendo interactuar con los distintos componentes del sistema desde una capa visual. El diseño de la interfaz se abordó utilizando el patrón Modelo-Vista-Controlador (MVC), lo que facilitaba una separación clara entre la lógica de negocio, la presentación y el control de eventos. De este modo, se esperaba poder conectar fácilmente esta interfaz con el orquestador del sistema, permitiendo ejecutar acciones como la recolección, procesamiento y visualización de datos mediante botones y formularios interactivos.

El desarrollo de esta interfaz se realizó en paralelo con la construcción de la arquitectura del sistema, con la idea de integrar ambos componentes en etapas posteriores.

Sin embargo, conforme el proyecto avanzaba, se reconsideraron los requisitos y el contexto de uso de la aplicación. Se concluyó que una interfaz web ofrecía más flexibilidad, escalabilidad y accesibilidad, permitiendo a múltiples usuarios acceder al sistema desde diferentes dispositivos sin necesidad de instalar software

Local. Además, una solución web facilitaba el despliegue en entornos remotos o en la nube.

Por estas razones, se tomó la decisión de abandonar el desarrollo con Tkinter y reorientar los esfuerzos hacia el diseño de una aplicación web, más alineada con los objetivos y necesidades del proyecto.

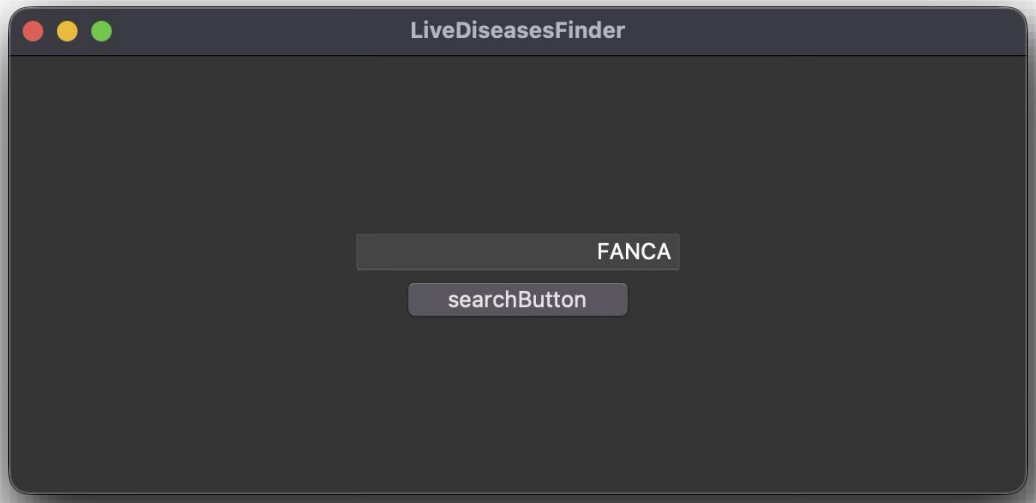



Figura 55. Interfaz gráfica de construcción propia usando Tkinter.

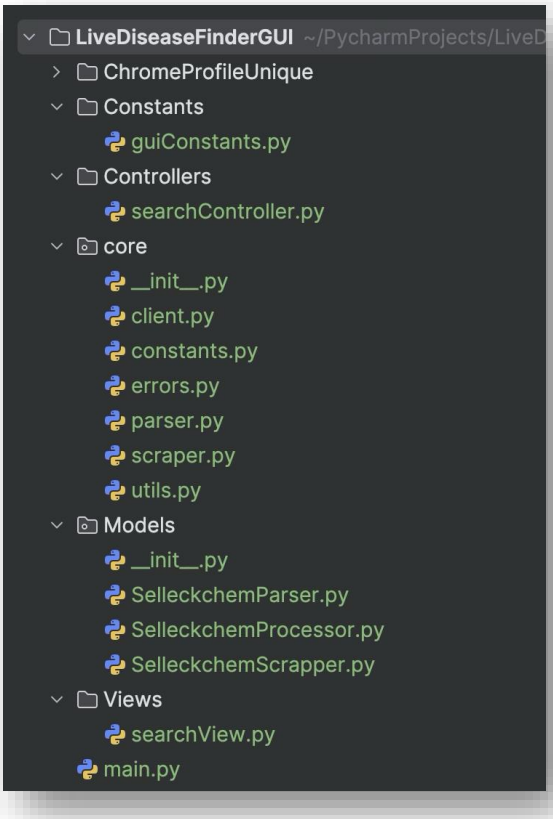



Figura 56. Raíz del proyecto

#### 5.1.3.2. Colaboración con el Grupo de Aplicación de Telecomunicaciones Visuales (GATV)

Se realizó un trabajo conjunto con el GATV y el cliente para el diseño de mockups de la interfaz gráfica, mediante reuniones de forma reiteradas. Por parte del equipo de desarrollo, se diseñaron mockups que se presentaron a las investigadoras y posteriormente a la experta en interfaces de usuario (UI), que a su vez refinó los diseños adaptándolos a nuestros requisitos de desarrollo y usabilidad para los usuarios en este caso los investigadores del IIER.

A continuación, se ilustran los diferentes mockups de la interfaz gráfica realizados por los desarrolladores:


Figura 57. Mockup de interfaz gráfica propuesta por desarrolladores Nº1.

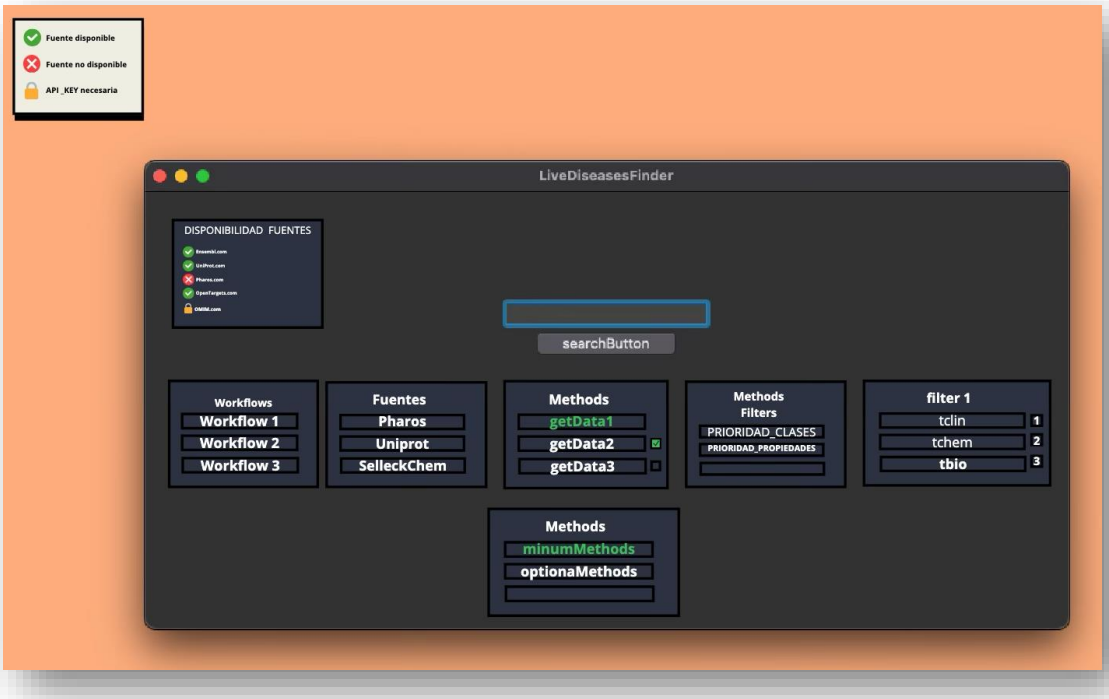



Figura 58. Mockup de interfaz gráfica propuesta por desarrolladores Nº2.

Realizados por la experta en UI:

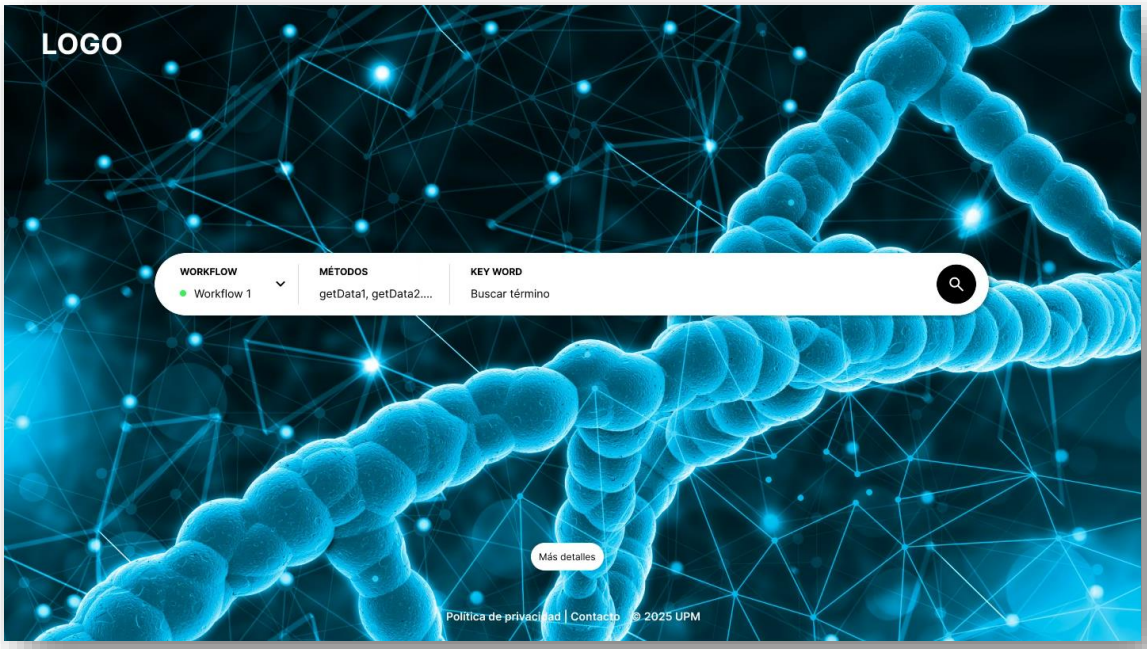



Figura 59. Mockup de interfaz gráfica propuesta por la experta en UI Nº1.

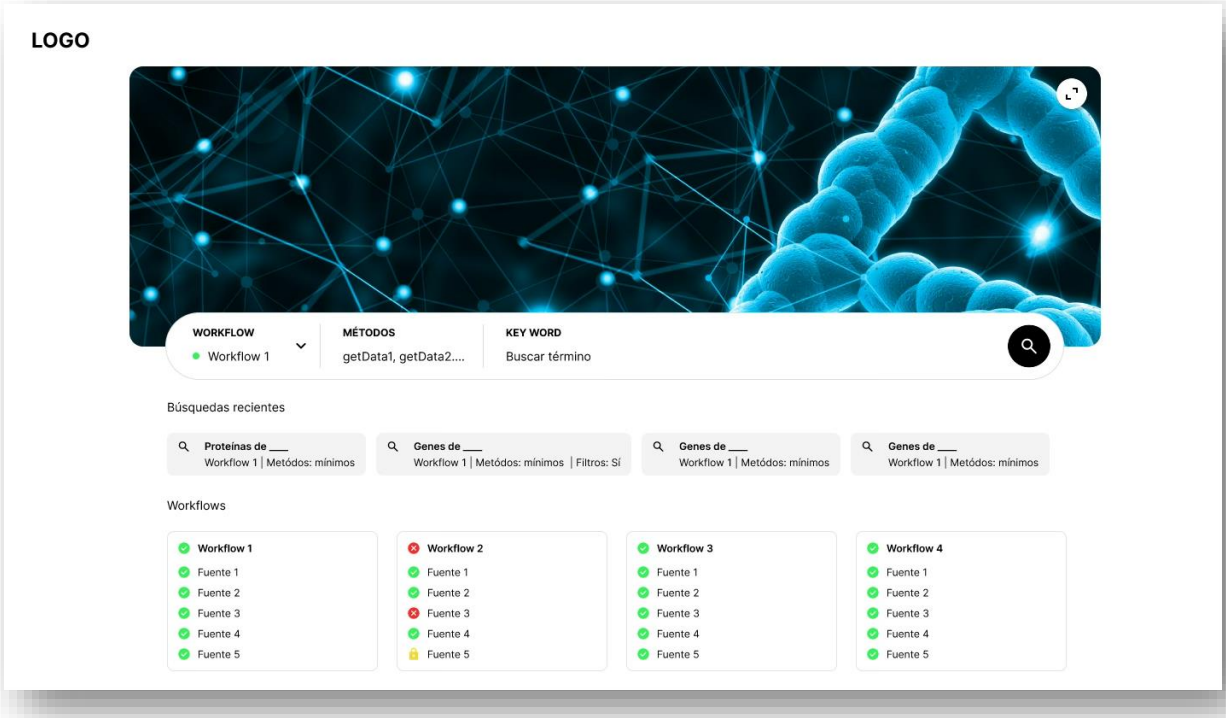



Figura 60. Mockup de interfaz gráfica propuesta por la experta en UI Nº2.

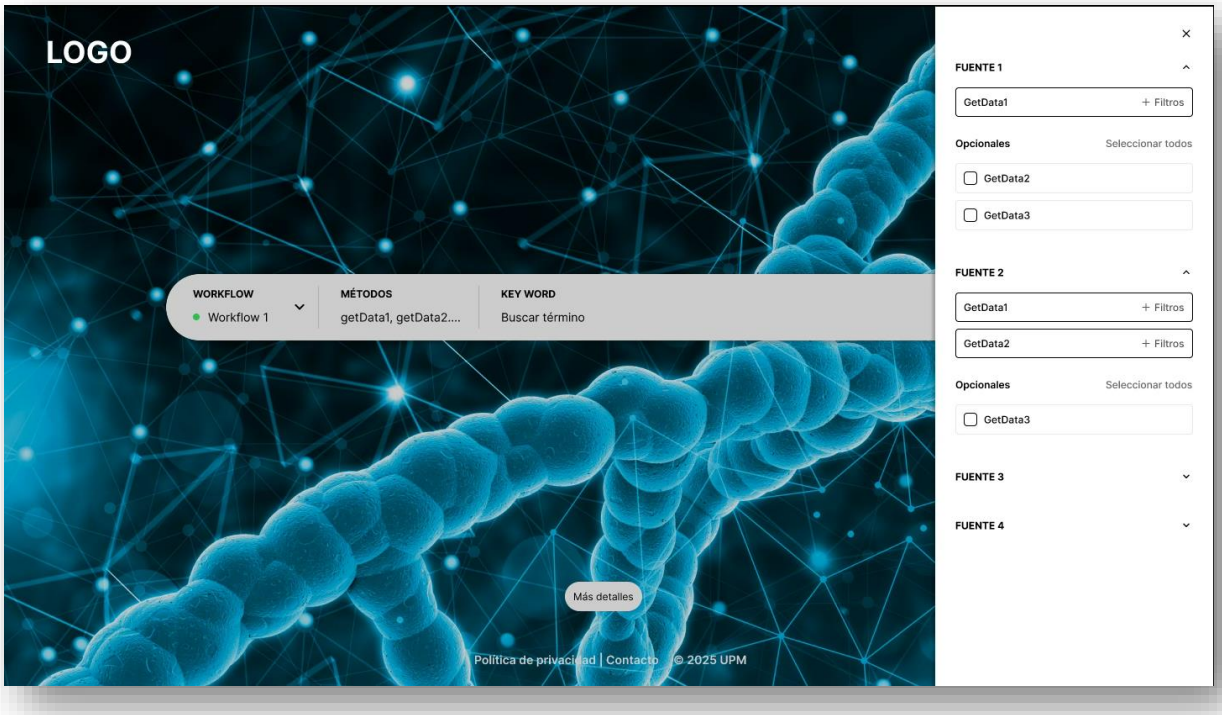



Figura 61. Mockup de interfaz gráfica propuesta la experta en UI Nº3.

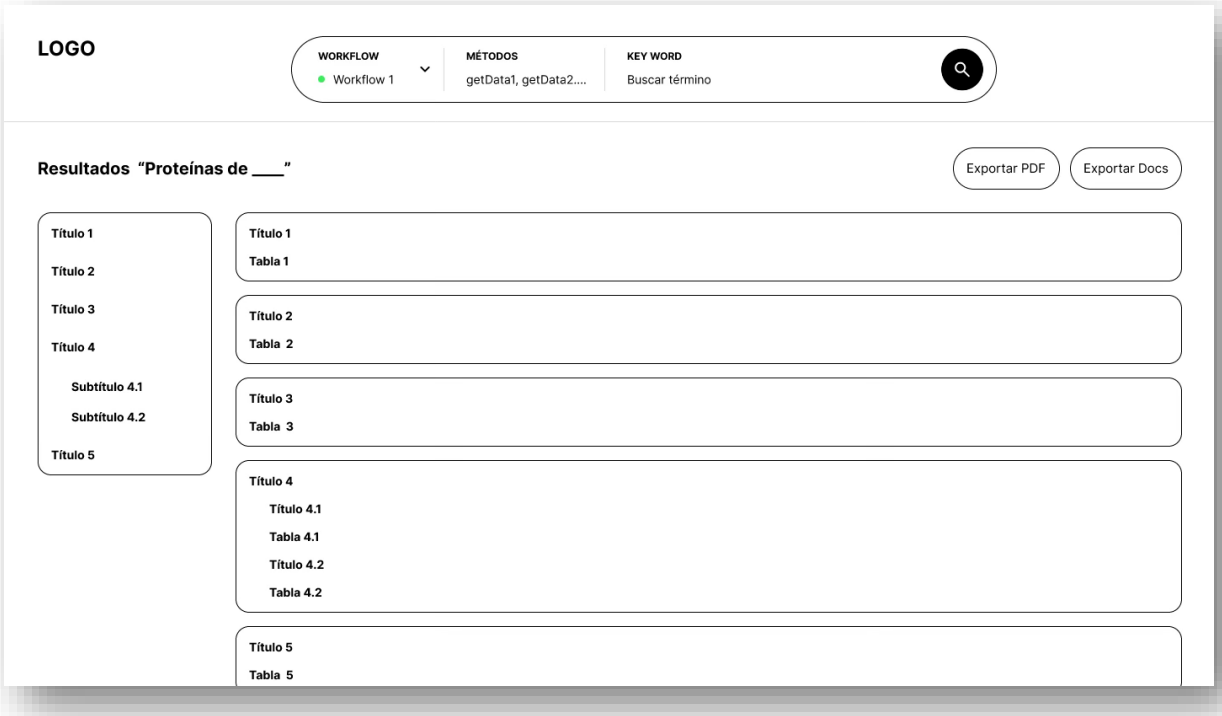



Figura 62. Mockup de interfaz gráfica propuesta por la experta en UI Nº4.

Tras semanas de desarrollo por ambas partes y sucesivas revisiones, el trabajo culminó en una interfaz refinada y funcional, centrada en la usabilidad y en facilitar el cumplimiento de los requisitos establecidos por el cliente. El resultado es una solución intuitiva y eficaz, alineada con los objetivos del proyecto.

La interfaz gráfica está realizada con el framework React y se ejecuta sobre un entorno Node.js.

En la página principal del buscador nos encontramos con distinta información, en el menú de búsqueda se puede seleccionar el Workflow que se desea ejecutar, este dispone de un indicador de disponibilidad, en caso de que alguna fuente no esté disponible, no se permitirá ejecutar el Workflow, debajo del menú se incluyen los Workflows con los Steps Detallados, aquí se puede observar la disponibilidad de las fuentes. Como se puede observar en la figura se dispone de 3 Workflows, Uno con todas las fuentes, uno sin la fuente Pharos y otro sin la fuente Phanter, después de realizar estudios internos, estas eran las fuentes que más suelen no tener disponibilidad. Siguiendo, en el menú podemos observar otro apartado llamado Métodos, en este al pulsar saldrá un desplegable lateral donde se permite la selección de métodos opcionales y selección de filtros. Por último, tenemos el campo de búsqueda donde se introduce el término a consultar y el botón de buscar (icono lupa).

A continuación, adjuntamos ilustraciones de la interfaz de usuario final del sistema:

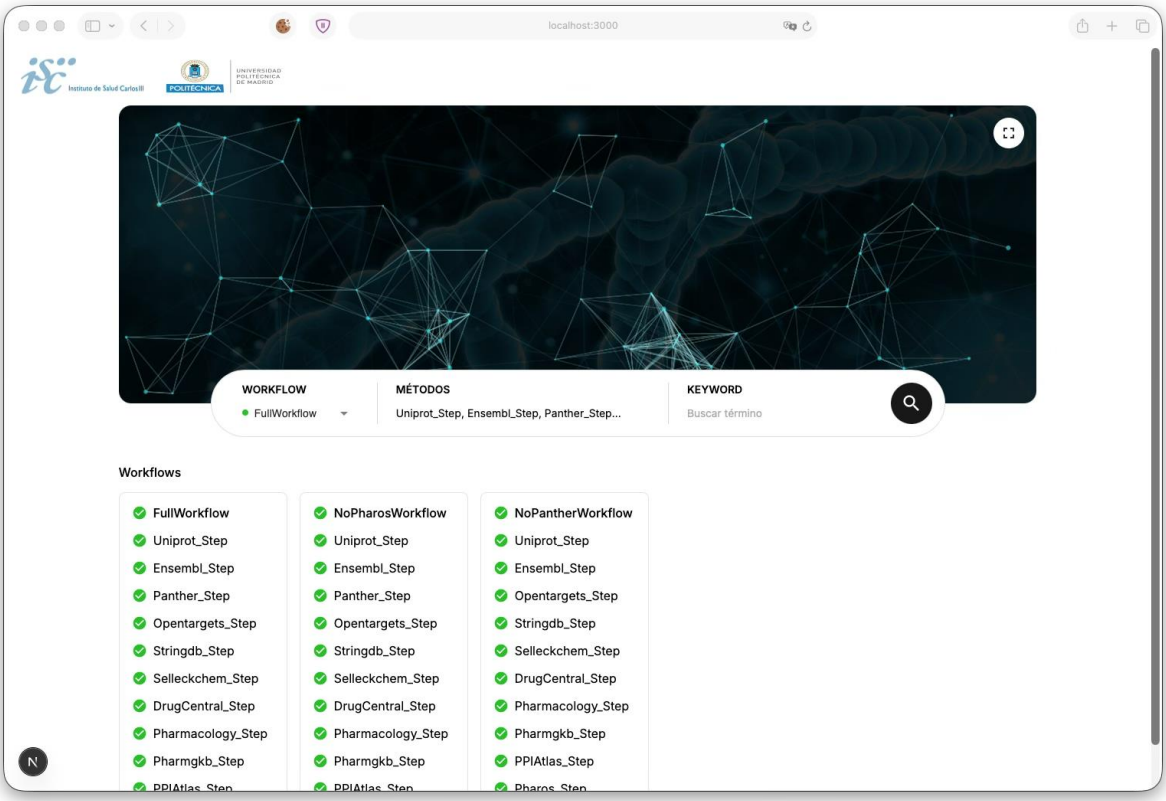



Figura 63. Página principal del buscador.

La vista del buscador dispone además de una versión alternativa que se activa pulsando el icono rectangular de la esquina superior derecha.

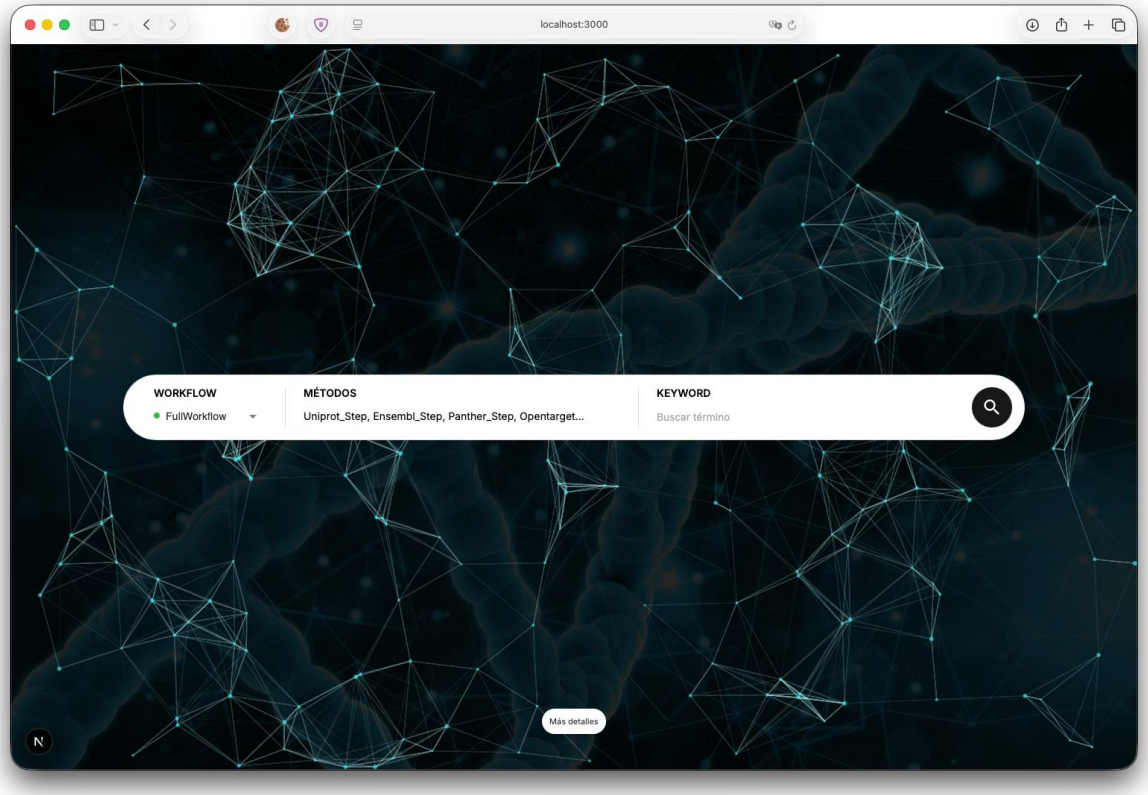



Figura 64. Versión alternativa de la vista del buscador

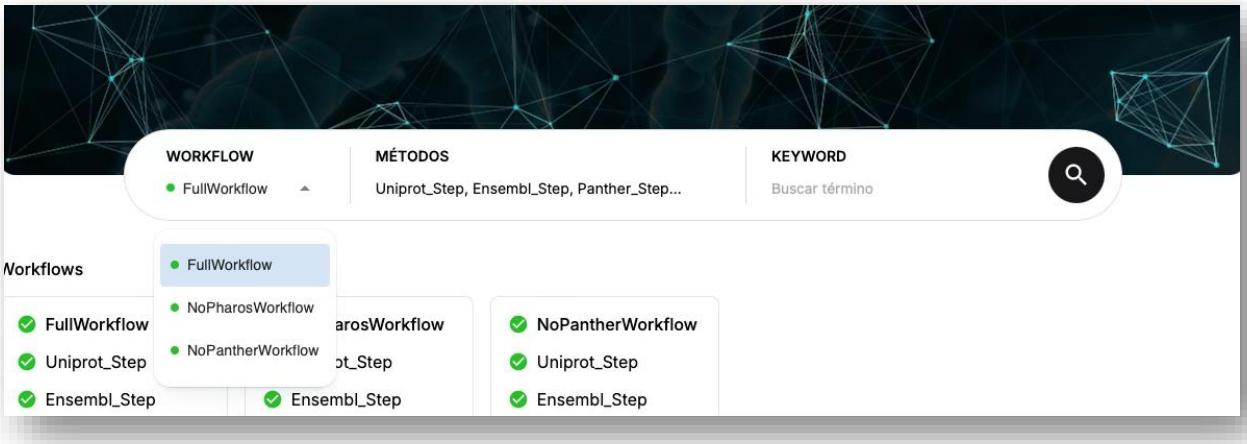



Figura 65. Desplegable de selección de Workflow.

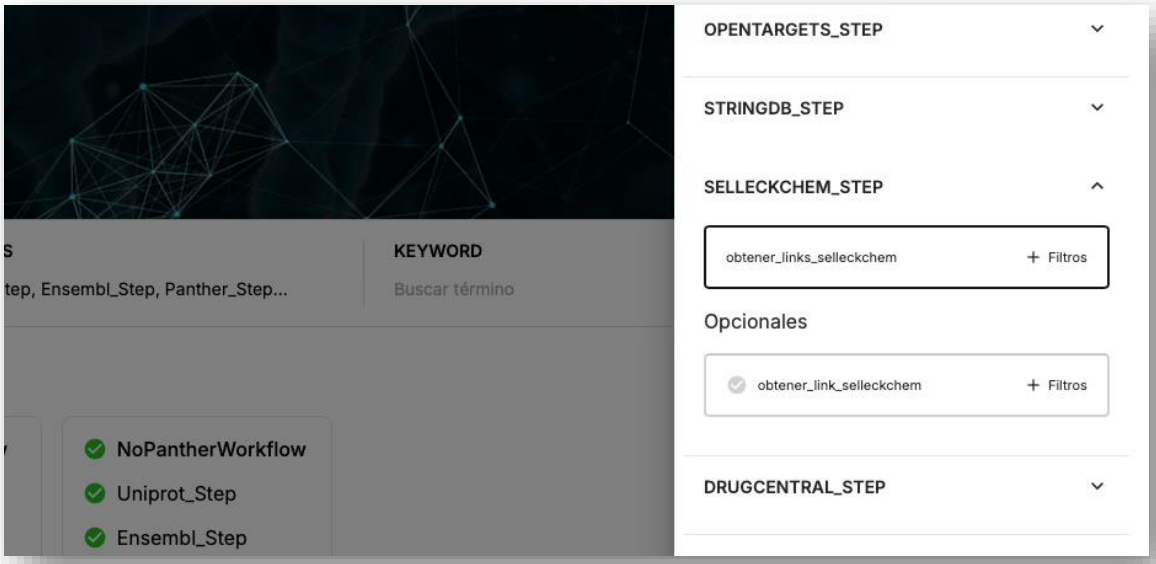



Figura 66. Desplegable lateral para la selección de métodos opcionales.

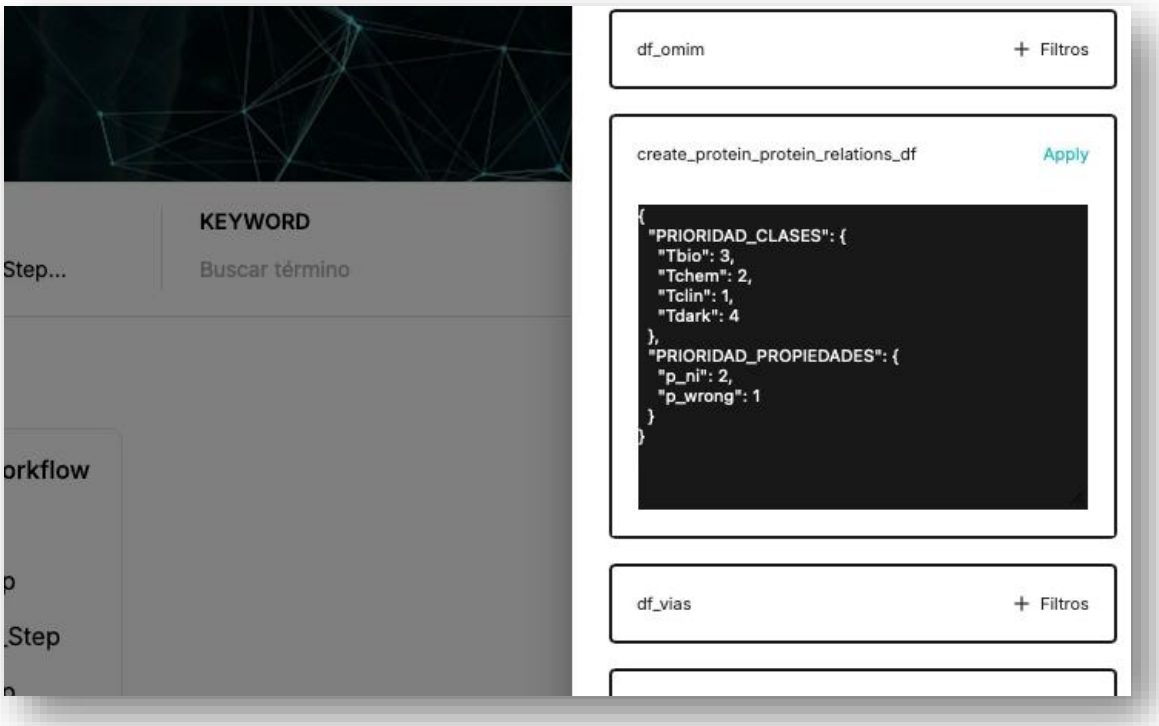



Figura 67. Desplegable lateral para la edición de filtros de método.

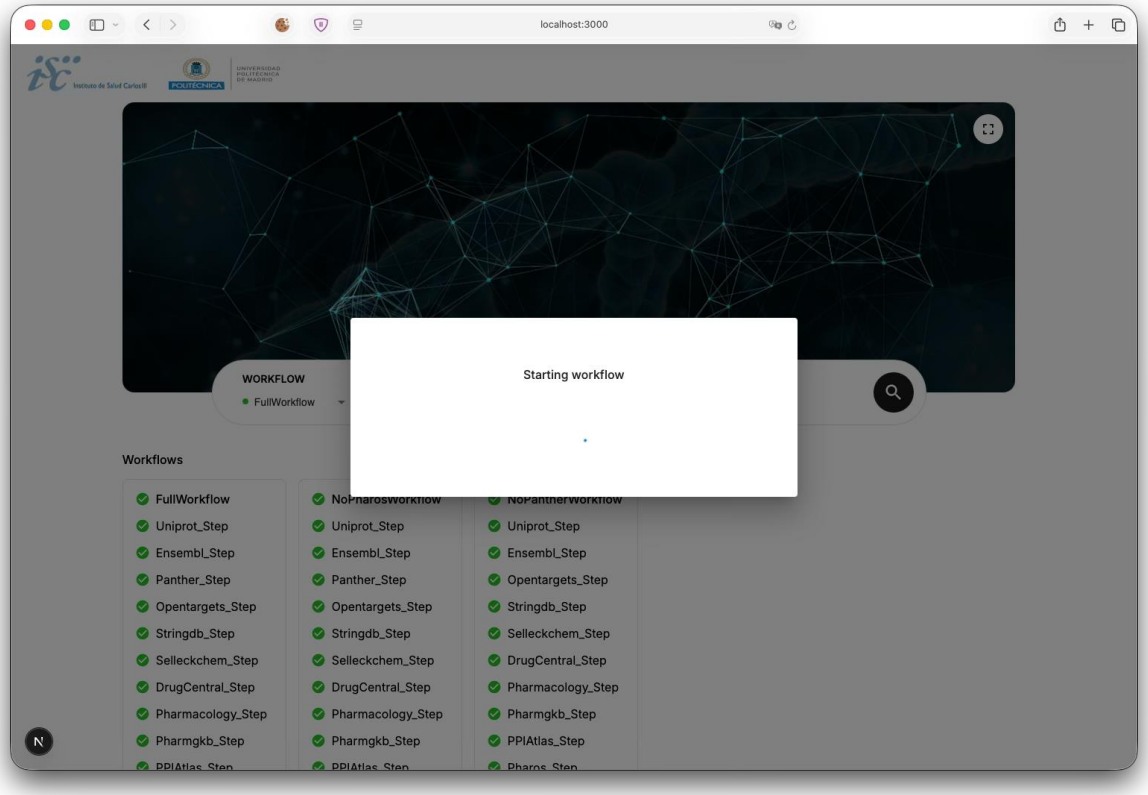



Figura 68. Pop up indicando que el Workflow ha empezado a ejecutarse.

Al pulsar el botón Buscar, se inicia en el backend el workflow de procesamiento. Durante su ejecución, se muestra un pop-up informativo con el estado actual del proceso.

Una vez generado el informe, se presenta una nueva ventana con un índice interactivo y la información estructurada en tablas. Estas pueden ordenarse de forma ascendente, descendente o alfabética.

Además, se incluye un botón de Exportar, que permite descargar el informe en formato HTML, elegido por su mayor compatibilidad con contenidos interactivos frente a las limitaciones del PDF.

Finalmente, un botón de "Nueva Búsqueda" permite al usuario volver a la pantalla del buscador para realizar otra consulta.

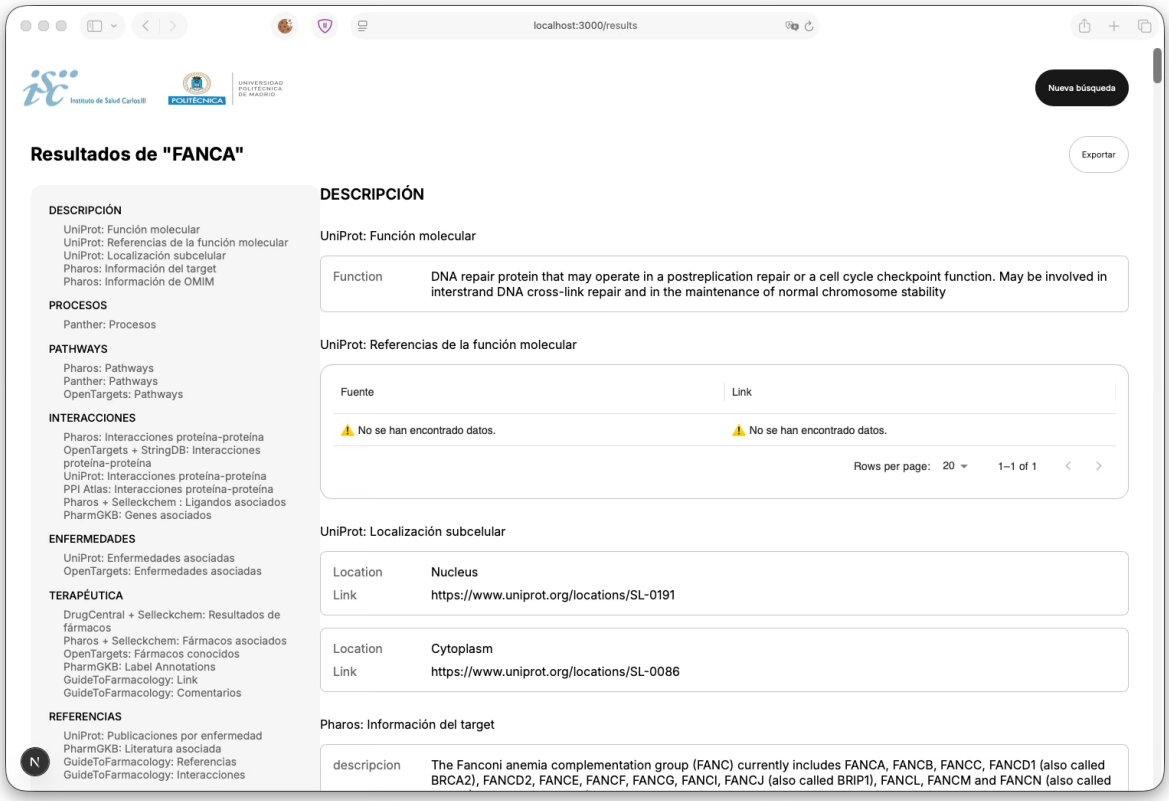



Figura 69. Ventana del Informe generado.

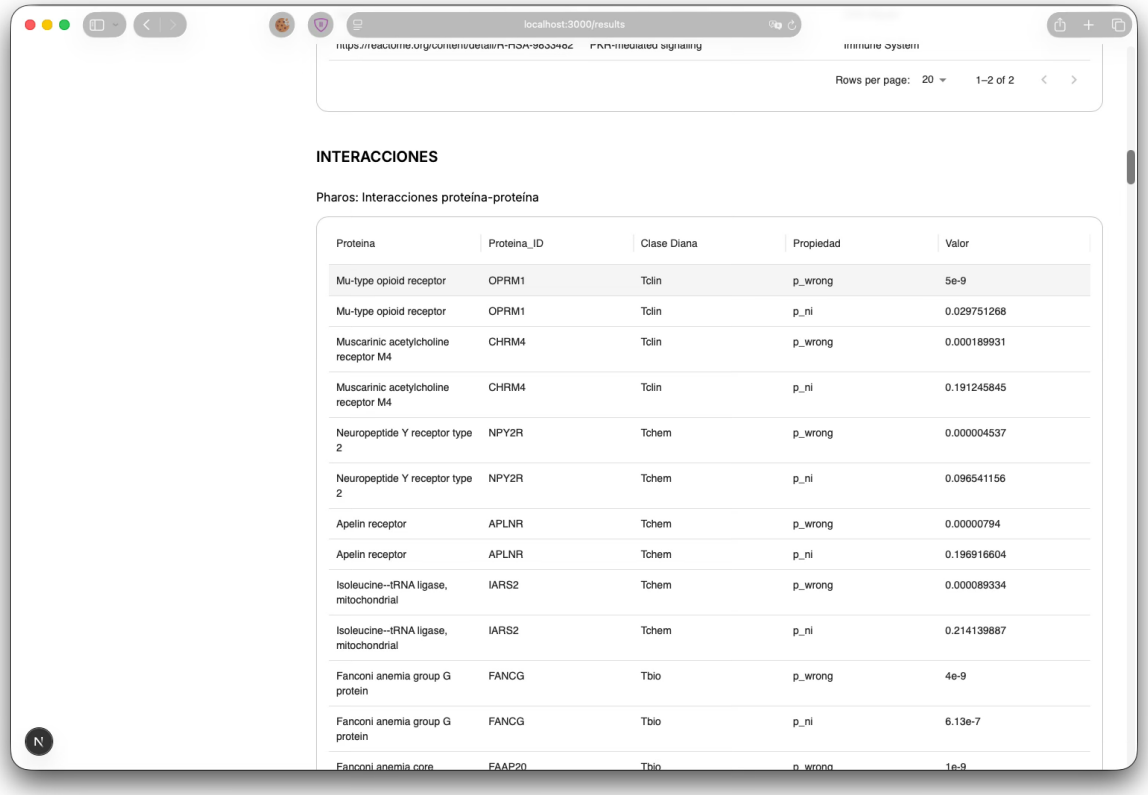



Figura 70. Ventana del informe generado representación datos tabulares.


Figura 71. Índice interactivo del informe.

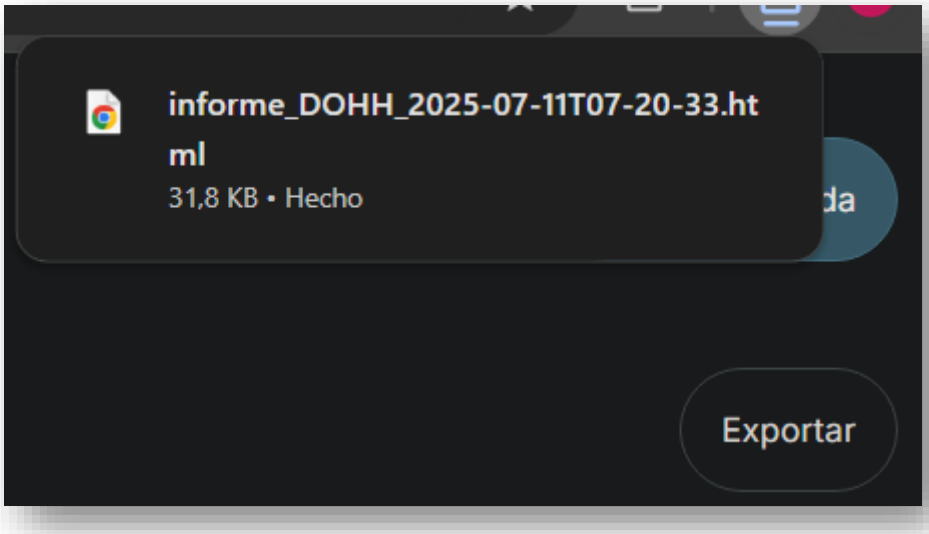



Figura 72. Archivo del informe descargado en formato HTML.

# 6. Validación y verificación de la solución

## 6.1. Verificación y validación del desarrollo

La verificación y validación han sido aspectos clave en el desarrollo de nuestro sistema, ya que permiten asegurar que tanto la lógica de procesamiento como la interfaz cumplen con los requisitos funcionales y técnicos establecidos.

La verificación de cada componente nos permitió detectar y corregir errores durante el desarrollo, mientras que la validación mediante escenarios reales aseguró que el sistema se comporta de forma adecuada ante las necesidades del usuario final.

Dado que el sistema implica múltiples etapas automatizadas como la búsqueda, el procesamiento y la generación de informes, estos procesos resultaron fundamentales para garantizar la fiabilidad, coherencia y usabilidad a lo largo de todo el flujo.

Durante el desarrollo, se implementaron distintos mecanismos para realizar pruebas unitarias e integradas, aplicadas en cada capa del sistema. Estas incluían

la ejecución de diversas consultas y casos de uso para asegurar el correcto funcionamiento general.

No obstante, debido a las limitaciones de tiempo, no fue posible aplicar una metodología de testeo exhaustiva, por lo que se priorizaron aquellas pruebas más críticas para el flujo principal del sistema.

```
298  if __name__ == "__main__":
299      wokflows = [FullWorkflow(),NoPharosWorkflow()]
300      orchestrator = Orchestrator(wokflows)
301
302
303      print(orchestrator.get_workflows())
304      print(orchestrator.get_list_of_steps_names("FullWorkflow"))
305      print(orchestrator.get_minium_methods_for_step_from_workflow("Pharos_Step","FullWorkflow"))
306      print(orchestrator.get_optional_methods_from_workflow("Pharos_Step","FullWorkflow"))
307      print(orchestrator.get_method_filters("df_numero_vias_por_fuente","Pharos_Step","FullWorkflow"))
308
309      print(orchestrator.workflows_list[1].workflow_state)
310
311      print("All step available? " + str(orchestrator.get_if_all_steps_available("FullWorkflow")))
312
313      print(orchestrator.workflows_list[1].workflow_state)
314
315      orchestrator.set_stage_2("FullWorkflow")
316      print(orchestrator.workflows_list[1].workflow_state)
317
318      orchestrator.set_workflow_search_param("FANCA","FullWorkflow")
319      print(orchestrator.workflows_list[1].workflow_state)
320
321      orchestrator.set_stage_3("FullWorkflow")
322      print(orchestrator.workflows_list[1].workflow_state)
323
324      orchestrator.start_workflow("FullWorkflow")
325      print(orchestrator.workflows_list[1].workflow_state)
326
327      orchestrator.set_stage_1("FullWorkflow")
328      orchestrator.workflows_list[1].workflow_state = "stage_1"
329      print(orchestrator.workflows_list[1].workflow_state)
330
```

Figura 73. Pruebas para verificar el funcionamiento del módulo Orquestador.

En cuanto a la validación del sistema, esta se llevó a cabo de forma iterativa durante las reuniones semanales, donde se presentaban los avances y se registraban las peticiones de cambio por parte del cliente.

Además, gracias a la comunicación constante por correo electrónico, se recibían solicitudes de ajuste o modificaciones funcionales de manera continua, lo que

permitió adaptar el desarrollo a las expectativas del cliente en tiempo real a través de la presentación de los prototipos interactivos.

Una vez concluida la fase de desarrollo, se realizaron varias reuniones de prueba en las que se evaluó el comportamiento del sistema en distintos escenarios. Estas sesiones finalizaron con la validación formal por parte del cliente, donde las investigadoras confirmaron que el sistema cumplía con los requisitos planteados.

## 6.2. Verificación y validación de los objetivos del proyecto

Aquí se mostrarán las tablas de verificación y validación desde el punto de vista del cliente y desde el punto de vista técnico.

Los objetivos se corresponden a los descritos en la Introducción de la memoria.

### 6.2.1. Objetivos del cliente

| ID OBJETIVO | DESCRIPCIÓN DEL OBJETIVO | VALIDADO/ VERIFICADO | JUSTIFICACIÓN |
|---|---|---|---|
| OBJ-C-01 | Sistematizar y dar robustez a la búsqueda de información biomédica. | SÍ | Sistema con arquitectura modular Client-Parser-Processor que estandariza el acceso a 11 fuentes de datos biomédicas. |
| OBJ-C-02 | Armonizar la secuencia de extracción de datos. | SÍ | Workflows predefinidos que establecen secuencias consistentes de extracción (Full, No Pharos, No Panther). |
| OBJ-C-03 | Acortar el tiempo de recuperación de los datos más relevantes. | SÍ | Reducción del tiempo de extracción de información de semanas a minutos mediante automatización. |
| OBJ-C-04 | Homogeneizar y complementar los informes emitidos. | SÍ | Generación de informes HTML estructurados con formato homogéneo para todos los genes consultados. |

### 6.2.2. Objetivos específicos del sistema

| ID OBJETIVO | DESCRIPCIÓN DEL OBJETIVO | VALIDADO/ VERIFICADO | JUSTIFICACIÓN |
|---|---|---|---|
| OBJ-ES-05 | Identificar y analizar las necesidades de investigación. | SÍ | Colaboración directa con el IIER mediante reuniones periódicas y validación iterativa de requisitos. |
| OBJ-ES-06 | Evaluar y caracterizar fuentes de datos biomédicos. | SÍ | Análisis e integración de 11 fuentes especializadas (UniProt, OpenTargets, Ensembl, Panther, etc.). |
| OBJ-ES-07 | Desarrollar un sistema de integración de datos biomédicos. | SÍ | Plataforma automatizada con procesadores específicos para cada fuente de datos heterogénea. |
| OBJ-ES-08 | Implementar una arquitectura robusta y escalable. | SÍ | Arquitectura modular con tolerancia a fallos, sistema de filtros configurable y workflows extensibles. |
| OBJ-ES-09 | Facilitar el acceso a información biomédica compleja. | SÍ | API REST con 3 stages que simplifica el acceso y generación de informes estructurados. |
| OBJ-ES-10 | Crear una solución interoperable. | SÍ | Backend con API REST documentada en Swagger que permite integración con múltiples frontends. |
| OBJ-ES-11 | Garantizar el cumplimiento normativo. | SÍ | Acceso exclusivo a bases de datos públicas y respeto de términos de uso de cada fuente. |
| OBJ-ES-12 | Validar la utilidad práctica del sistema. | SÍ | Validación continua con investigadores del IIER y casos de uso reales con genes específicos. |

# 7. Consideraciones ambientales, sociales y éticas

Es fundamental que la solución creada respete las normativas y regulaciones correspondientes tanto en el sector de la informática como en el de la salud. Garantizar el uso correcto de los datos en el mundo del software es una recomendación; pero en el sector de la salud es toda una obligación, puesto que se tratan datos sensibles continuamente, y un software que sea efectivo pero que no cumpla con los criterios de legalidad invalidaría por completo la solución propuesta.

En este capítulo, se analizarán los aspectos ambientales, sociales y éticos implicados en la creación y uso de la solución, así como su posible impacto en el mundo de la investigación de enfermedades raras.

## 7.1. Aspectos legales y éticos

Para el desarrollo de este proyecto o cualquiera que recopile datos biomédicos, es fundamental cumplir con las normativas de la bioinformática correspondientes. La regulación acerca de los mismos puede variar dependiendo del país o región donde estos datos se encuentren: la legislación respecto a estos datos puede no ser la misma en Europa que en América o Asia, y el carácter heterogéneo del proyecto obliga a atender en mayor detalle las distintas implicaciones legales que la recopilación de datos del sistema pueda tener en las distintas regiones donde se encuentran las fuentes de datos a acceder.

En el caso de que el proyecto incluyera elementos protegidos por derechos de propiedad intelectual, tales como patentes, marcas registradas o diseños, es imprescindible respetar dichos derechos y gestionar las autorizaciones pertinentes a la hora de usarlas en el sistema. En el caso del proyecto, algunas de las páginas candidatas a ser fuentes de información presentaban la posibilidad de obtener datos relevantes a partir de ellas a través de técnicas como el web scraping. Aunque desde el punto de vista técnico esto era posible por parte del equipo de desarrolladores, la legislación específica de la página negaba esta posibilidad al prohibir el uso de técnicas de web scraping dentro de su dominio. Es el caso, por ejemplo, de GeneCards:

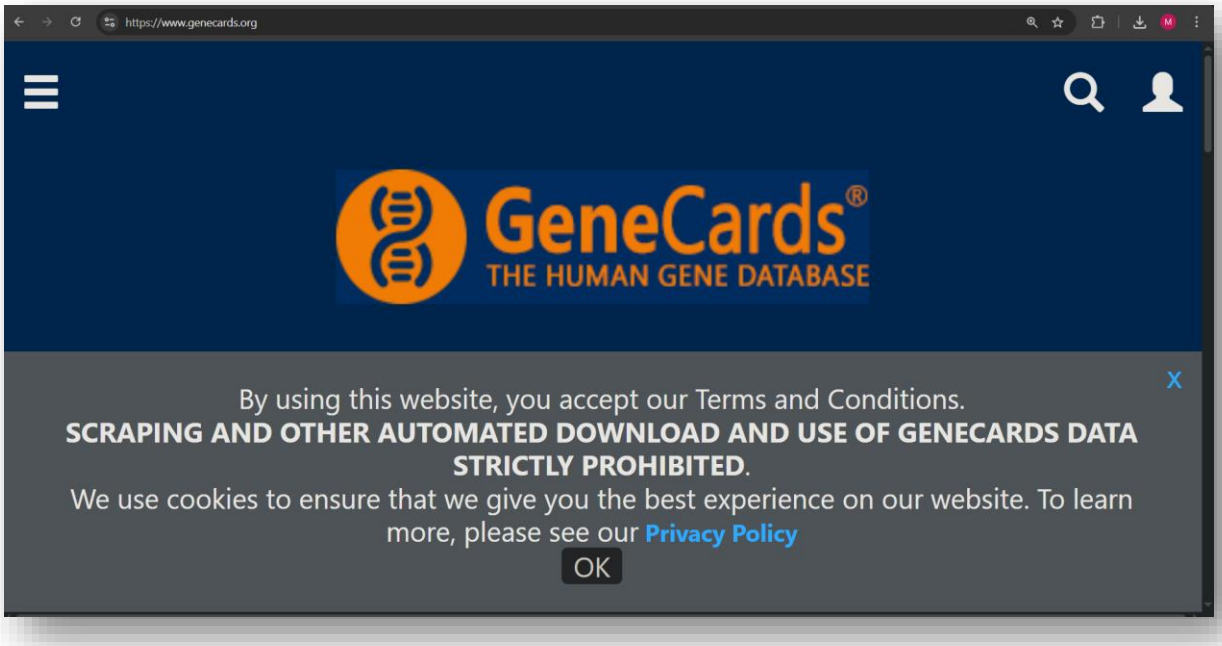



Figura 74. Mensaje de política de privacidad de la fuente de datos GeneCards. Fuente: https://www.genecards.org/

Esto demuestra que, a pesar de que es un requisito que pedía el cliente, y que desde el punto de vista de la informática era plausible, en el desarrollo del sistema impera el respeto por las reglas establecidas por encima de la eficiencia o la implementación de funcionalidades adicionales; para el correcto desarrollo del proyecto es más importante el mantenerse en el marco legal, aunque implique sacrificar algunos aspectos importantes de la solución.

Se destaca también que la solución creada es accesible para todo el mundo puesto que se encuentra subida a un repositorio público. El sistema se ha diseñado como una herramienta que pueda ser utilizada por cualquier investigador independientemente de su afiliación institucional o localización geográfica. Esta característica demuestra el compromiso con la equidad en el acceso al conocimiento científico; se aspira a que la solución sea de acceso libre para fomentar en su máxima expresión la difusión de conocimiento.

Por último, hay que mencionar que la plataforma del proyecto, al integrar datos de múltiples fuentes internacionales, promueve un enfoque global para abordar los problemas de salud e investigación biomédica más allá de las fronteras, recogiendo de esta manera todo el conocimiento de investigación biomédica y siguiendo la idea de que el conocimiento científico no pertenece a ningún país en concreto.

## 7.2. Aspectos sociales

El desarrollo de este proyecto tiene una trascendencia mayor que la de ser un proyecto de fin de carrera. Este proyecto, al tratarse de una colaboración con el Instituto de Salud Carlos III, aspira a crear una nueva línea de colaboración con el IIER.

Esta iniciativa se enmarca dentro de la estrategia consolidada de la Universidad Politécnica de Madrid de establecer vínculos sólidos con el sector biomédico, sumándose a las múltiples colaboraciones existentes entre la UPM e instituciones de investigación sanitaria. El proyecto representa una nueva línea férrea y duradera de trabajo entre el Instituto de Salud Carlos III y los grupos de investigación de la UPM, aprovechando la experiencia previa de la universidad en proyectos interdisciplinares que combinan ingeniería y biomedicina.

El éxito de este proyecto piloto está diseñado para fortalecer aún más la sinergia entre la experiencia biomédica del IIER y las capacidades tecnológicas consolidadas de la UPM, ampliando el precedente ya establecido por otras colaboraciones exitosas y facilitando futuras iniciativas interdisciplinares. La consolidación de esta alianza institucional específica con el IIER permitirá el desarrollo de una agenda de investigación conjunta, enriqueciendo el marco ya existente de intercambio de conocimientos y recursos entre la UPM y el sector biomédico.

Esta línea de colaboración continuada contribuirá al ecosistema de innovación sostenible que la Universidad Politécnica de Madrid ha venido desarrollando en el ámbito de la salud digital, generando beneficios a largo plazo para el conjunto del sistema de investigación e innovación en salud de España.

## 7.3. Contribución a los ODS

El desarrollo del proyecto contribuye significativamente a varios Objetivos de Desarrollo Sostenible (ODS) establecidos por las Naciones Unidas en la Agenda 2030.

- **ODS 3: Salud y Bienestar** constituye el objetivo principal al que contribuye este proyecto. Al facilitar el acceso a información biomédica integrada sobre enfermedades raras, el sistema contribuye a acelerar los procesos de investigación y diagnóstico, reduciendo potencialmente los tiempos de

espera que actualmente superan los 4 años en España y mejorando la situación de pacientes con condiciones poco frecuentes.

- **ODS 4: Educación de Calidad** se fortalece mediante la creación de una plataforma que proporciona acceso libre y estructurado a conocimiento biomédico especializado. La naturaleza de código abierto garantiza que las instituciones educativas puedan incorporar estas herramientas en sus programas formativos sin barreras económicas.
- **ODS 9: Industria, Innovación e Infraestructura** se beneficia del desarrollo de infraestructuras tecnológicas innovadoras que facilitan la investigación científica. La arquitectura modular del sistema representa una innovación en la integración de datos biomédicos que puede servir como base para futuras aplicaciones en el sector sanitario.
- **ODS 10: Reducción de las Desigualdades** se ve impactado al eliminar barreras económicas en el acceso a herramientas de investigación biomédica avanzadas. Mientras que plataformas similares requieren suscripciones costosas, el sistema proporciona acceso gratuito, especialmente relevante para investigadores con recursos limitados.
- **ODS 17: Alianzas para lograr los objetivos** se fortalece mediante la colaboración entre la Universidad Politécnica de Madrid y el Instituto de Salud Carlos III, sentando precedentes para futuras asociaciones que combinen experiencia en el sector tecnológico y biomédico en beneficio de la sociedad.

# 8. Conclusiones

El proyecto se ha completado satisfactoriamente, no solamente desde la perspectiva del equipo de desarrollo, sino fundamentalmente desde el punto de vista del cliente, quien ha expresado su satisfacción con los resultados obtenidos y las funcionalidades implementadas. Esta validación externa constituye el indicador más importante del éxito del proyecto, ya que demuestra que la solución desarrollada cumple con las expectativas y necesidades reales del IIER.

La experiencia adquirida durante el desarrollo ha sido especialmente enriquecedora al trabajar con un cliente real, lo que ha proporcionado una perspectiva práctica y profesional del proceso de desarrollo de software. A diferencia de proyectos académicos tradicionales, esta colaboración ha implicado la

gestión de requisitos cambiantes, la comunicación con stakeholders especializados, y la adaptación continua a las necesidades específicas del dominio biomédico.

El trabajo en equipo ha sido otro aspecto fundamental del proyecto. Al desarrollarse como un proyecto de fin de grado conjunto entre dos personas, se ha fortalecido la capacidad de colaboración, división de tareas, y coordinación de esfuerzos. Adicionalmente, la colaboración con la especialista en interfaces de usuario para el desarrollo del frontend ha enriquecido la experiencia multidisciplinar, desarrollando habilidades de comunicación técnica y trabajo con profesionales de diferentes áreas.

Una de las competencias más valiosas adquiridas ha sido la capacidad de análisis y comprensión de documentación técnica externa. Cada fuente de datos biomédica integrada (UniProt, Pharos, OpenTargets, StringDB, entre otras) presenta su propia arquitectura de APIs, sistemas de consulta, y formatos de datos. El proceso de lectura, interpretación y aplicación de estas documentaciones técnicas, especialmente en el caso de APIs REST y sistemas GraphQL, ha desarrollado habilidades de investigación técnica y adaptación a tecnologías diversas que serán fundamentales en el desarrollo profesional futuro.

## 8.1. Líneas de Trabajo Futuras

Las siguientes líneas de trabajo futuro no son especulaciones teóricas, sino especificaciones concretas identificadas y comentadas en colaboración directa con el cliente durante las sesiones de evaluación y feedback del proyecto.

Estas líneas de trabajo futuro reflejan el carácter iterativo y evolutivo del proyecto, así como el compromiso de mejora continua basado en el feedback real de usuarios especializados en el dominio de aplicación.

### 8.1.1. Mejoras en la presentación de resultados

Se ha identificado la necesidad de implementar funcionalidades de exportación que permitan generar informes en formato PDF directamente desde el HTML generado, o alternativamente, crear un mecanismo de exportación directa de informes a PDF. Esta funcionalidad facilitaría el uso académico y profesional de los resultados obtenidos, permitiendo su distribución e inclusión en documentos de investigación.

### 8.1.2. Personalización avanzada de workflows

El cliente ha expresado interés en desarrollar un mecanismo para seleccionar secciones específicas de información, implementando un "Manual Workflow" que permita a los usuarios personalizar exactamente qué tipos de datos desean obtener. Complementariamente, se requiere la creación de nuevos filtros especializados, como por ejemplo un filtro para establecer umbrales de puntuación en las interacciones proteína-proteína de OpenTargets, proporcionando mayor control sobre la relevancia de los resultados.

### 8.1.3. Optimización de búsquedas y navegabilidad

Se ha detectado que en la fuente de datos DrugCentral existen oportunidades de mejora en los algoritmos de búsqueda. Por ejemplo, mientras que la búsqueda directa del gen "FANCA" no produce resultados, la búsqueda por el término "Fanconi anemia" sí los genera. Esta variabilidad en los resultados sugiere la necesidad de implementar estrategias de búsqueda más sofisticadas y tolerantes a diferentes nomenclaturas. Adicionalmente, el cliente ha sugerido la implementación de un índice navegable al inicio de los informes que anticipe el contenido de las secciones, indicando incluso cuáles contienen datos y cuáles no, mejorando significativamente la experiencia de usuario y la eficiencia en la consulta de información.

### 8.1.4. Investigación de WikiPathways

Se ha identificado la necesidad de explorar e integrar WikiPathways como fuente adicional de datos de vías metabólicas y de señalización. Esta fuente utiliza formatos de datos RDF (Resource Description Framework) y consultas mediante lenguaje SPARQL, tecnologías del web semántico que requieren un enfoque de integración diferente al de las APIs REST y GraphQL actualmente implementadas. La incorporación de WikiPathways ampliaría significativamente las capacidades del sistema en el análisis de pathways biológicos, proporcionando información complementaria especializada en rutas metabólicas y vías de señalización celular específicamente relevantes para enfermedades raras.

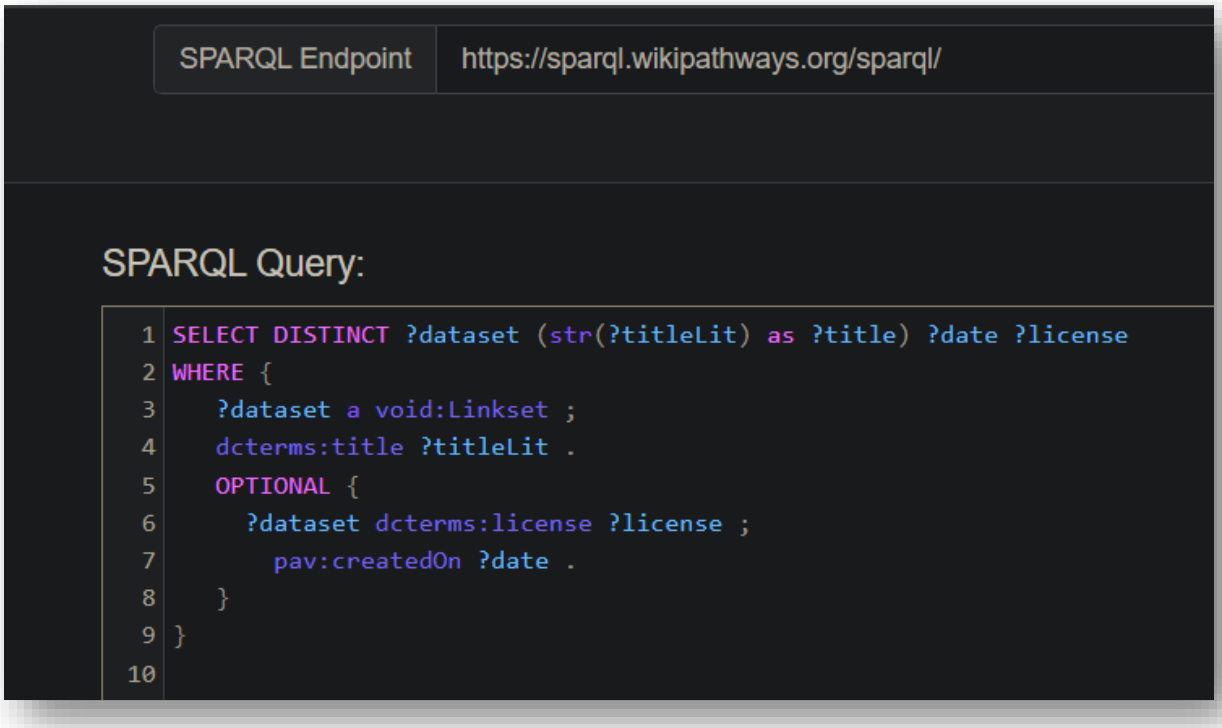



Figura 75. Representación de una consulta de ejemplo en el lenguaje SPARQL. Fuente: https://sparql.wikipathways.org/

### 8.1.5. Integración de modelos extensos de lenguaje para la generación de Workflows más personalizables.

La arquitectura modular del sistema permite una futura incorporación al sistema de búsqueda de un **LLM** (siglas en inglés para *Large Language Model*), esto se debe al mecanismo de comunicación que usan los Workflows y los Procesadores, al usar un documento JSON con una estructura clara, la generación de este documento mediante herramientas LLM facilitaría la creación de Workflows de forma automatizada, facilitando así la creación de Workflows adaptados a diferentes tipos de investigaciones permitiendo así llegar a una base de usuarios más amplia.

# 9. Referencias


1. **Revista Médica Clínica Las Condes.** Las enfermedades raras. *Revista Médica Clínica Las Condes.* 2015.

2. **FEDER, Página oficial:.** *https://www.enfermedades-raras.org/.*

3. **Federación Española de Enfermedades Raras.** La odisea de las personas con enfermedades raras: una media de más de cuatro años para obtener un diagnóstico. *Federación Española de Enfermedades Raras.* 28 de Febrero de 2023, págs. https://www.enfermedades-raras.org/actualidad/noticias/la-odisea-de-las-personas-con-enfermedades-raras-una-media-de-mas-de-cuatro-anos-para-obtener-un-diagnostico.

4. **SpringerLink.** Preparing Data at the Source to Foster Interoperability across Rare Disease Registries. *Data Integration in the Life Sciences.* 2017.

5. **IIER, Página oficial:.** *https://iier.isciii.es/.*

6. **KELLEHER, Karina D., et al.** Pharos 2023: an integrated resource for the understudied proteins of the human proteome. *Nucleic Acids Research [en línea].* Disponible en: https://academic.oup.com/nar/article/51/D1/D1405/6831351, 2023, Vols. vol. 51, no. D1, pp. D1405-D1416.

7. **PUN, Feng Ren, et al.** Identification of therapeutic targets for amyotrophic lateral sclerosis using PandaOmics - an AI-enabled biological target discovery platform. *Frontiers in Aging Neuroscience [en línea].* 2022, Vols. vol. 14, pp. 914017, Disponible en: https://www.frontiersin.org/journals/aging-neuroscience/articles/10.3389/fnagi.2022.914017/.

8. **Institute, EMBL's European Bioinformatics.** *Página oficial: https://www.ebi.ac.uk/.*

9. **Bioinformatics, SIB Swiss Institute of.** *Página oficial: https://www.sib.swiss/.*

10. **Resource, Protein Information.** *Página oficial: https://proteininformationresource.org/.*

11. **365, Microsoft.** OneNote. *Página oficial: https://www.microsoft.com/en-us/microsoft-365/onenote/digital-note-taking-app.*

12. **BELSHE, M., PEON, R. y THOMSON, M.** Hypertext Transfer Protocol Version 2 (HTTP/2). RFC 7540, Internet Engineering Task Force (IETF). s.l. : https://tools.ietf.org/html/rfc7540., 2015.

13. **BYRON, Lee, O'SHANNESSY, Dan y HARTIG, Bryn.** *GraphQL: A Query Language for APIs.* s.l. : The GraphQL Foundation, 2021. https://graphql.org/learn/.

14. **CHACON, Scott y STRAUB, Ben.** Pro Git [en línea]. s.l. : https://git-scm.com/book, 2nd ed. Berkeley: Apress, 2014 [consulta: 7 julio 2025].

15. **VAN ROSSUM, Guido y DRAKE, Fred L.** *Python 3 Reference Manual.* s.l. : CreateSpace, 2009.

16. **McKINNEY, Wes.** *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython.* s.l. : 2nd ed. Sebastopol, 2017.

17. **Reitz, Kenneth.** *requests.* 2025. https://pypi.org/project/requests/.

18. **GRINBERG, Miguel.** *Flask Web Development: Developing Web Applications with Python.* s.l. : 2nd ed. Sebastopol, 2018.

19. **RICHARDSON, Leonard.** *Beautiful Soup Documentation.* s.l. : Crummy, 2022. https://www.crummy.com/software/BeautifulSoup/bs4/doc/.

20. **BURNS, David.** *Selenium 2 Testing Tools: Beginner's Guide.* s.l. : Birmingham: Packt Publishing, 2012.

21. **META PLATFORMS, INC.** React – A JavaScript library for building user interfaces [en línea. s.l. : Disponible en: https://reactjs.org/.

22. **OPENJS FOUNDATION.** Node.js [en línea]. *Disponible en: https://nodejs.org.*

23. **PYTHON SOFTWARE FOUNDATION.** Tkinter — Python interface to Tcl/Tk [en línea]. *Disponible en: https://docs.python.org/3/library/tkinter.html.*

24. **GITHUB, INC.** GitHub Docs: About GitHub [en línea]. s.l. : Disponible en: https://docs.github.com/en/get-started/learning-about-github/about-github, San Francisco: GitHub, 2024 [consulta: 7 julio 2025].

25. **Santander Open Academy.** Metodologías de desarrollo software. *Santander Open Academy.* s.f.

26. **Royce, W.W.** Managing the Development of Large Software Systems. *Proceedings of IEEE WESCON.* 1970, Vol. 26, 1-9.

27. **Kruchten, P.** *The Rational Unified Process: An Introduction.* s.l. : 3ª ed. Addison-Wesley Professional, 2004.

28. **Mathur, S. y Malik, S.** Advancements in the V-Model. *International Journal of Computer Applications.* 1(12), 2010, pp 29-34.

29. **Cockburn, A.** Agile Software Development. *Addison-Wesley Professional.* 2002.

30. **Schwaber, K. y Sutherland, J.** The Scrum Guide: The Definitive Guide to Scrum. *Disponible en: https://scrumguides.org/.* 2020.

31. **Beck, K. y Andres, C.** *Extreme Programming Explained: Embrace Change.* s.l. : 2ª ed. Addison-Wesley Professional., 2004.

32. **Anderson, D.J.** Kanban: Successful Evolutionary Change for Your Technology Business. *Blue Hole Press.* 2010.

33. **Cockburn, Alistair.** *Agile Software Development.* s.l. : Addison-Wesley Professional, 2002. p.81.

34. *Apache Airflow Recomendación uso.* s.l. : Pagina oficial: https://airflow.apache.org/docs/apache-airflow/stable/index.html#why-not-airflow.

# 10. Anexos

La guía de instalación, así como los detalles del código fuente, pueden encontrarse en el README.md del repositorio del proyecto:

https://github.com/emilpintilie/RareDiseaseFinder

https://github.com/MarioBravoCuadro/RareDiseaseFinder

## 10.1. Guía de desarrollador Backend

Link al documento:

https://github.com/MarioBravoCuadro/RareDiseaseFinder/blob/main/docs/guides/Guia_Implementar_Proveedor_Datos_Biologicos_RareDiseaseFinder.docx

## 10.2. Guía de desarrollador Frontend

Link al documento:

https://github.com/MarioBravoCuadro/RareDiseaseFinder/blob/main/docs/guides/Gu%C3%ADa_Desarrollador_Front_End_API_REST_RareDiseaseFinder.docx

## 10.3. Inventario de extracción de información

Link al documento:

https://github.com/MarioBravoCuadro/RareDiseaseFinder/blob/main/docs/guides/Inventario_de_extraccion_de_informacion_de_fuentes_de_datos_biomedicas_COMENTARIOS%20SECCs%20Y%20WFs.docx

## 10.4. Excel de seguimiento de fuentes

Link al documento:

https://github.com/MarioBravoCuadro/RareDiseaseFinder/blob/main/docs/guides/SeguimientoFuentes.xlsx

## 10.5. Demo audiovisual del Frontend

Link a la demo:

https://youtu.be/ESQv-lsty60