Universidad de Cantabria

*Facultad de Ciencias*

# HERRAMIENTA WEB PARA COMPARAR LA DISPERSIÓN DE LUZ POR PARTÍCULAS ESFÉRICAS DE DIFERENTES MATERIALES MEDIANTE CÁLCULOS DE MIE

## (Web tool to compare light scattering by spherical particles of different materials using Mie calculations)

Trabajo de Fin de Grado
para acceder al

**GRADO EN INGENIERÍA INFORMÁTICA**


Autor: Sergio Salas Sánchez

Director: Alfonso de la Vega Ruiz

Co-Directora: Yael Gutiérrez Vela


Junio – 2025

# Resumen

Este proyecto consiste en el desarrollo de una herramienta web que permita comparar la respuesta de dispersión electromagnética de partículas esféricas hechas de diferentes materiales utilizando la teoría de Mie.

Dicha teoría, formulada a principios del siglo XX, describe de forma teórica cómo se dispersa la luz al interactuar con partículas esféricas de diferentes materiales con tamaño comparable a la longitud de onda. Este fenómeno es de gran relevancia en campos como la óptica, la nanofotónica, la ciencia de materiales y la nanotecnología, ya que permite predecir y analizar el comportamiento electromagnético de partículas nanométricas en distintos medios y bajo diversas condiciones del entorno.

El objetivo principal de la herramienta desarrollada es proporcionar una interfaz intuitiva y funcional que facilite la comparación visual de la dispersión de luz por partículas de diferentes materiales y tamaños a través de la generación de gráficas. Para esta generación, se utilizan datos reales de índices de refracción obtenidos de catálogos científicos, y se aplican los cálculos correspondientes para obtener los resultados de dispersión de la luz en base al material utilizado, además de otros factores.

Además de servir como herramienta de apoyo en entornos de investigación, este proyecto pretende ser una muestra de integración de conceptos físicos complejos con tecnologías de desarrollo web, facilitando así la comprensión y visualización de fenómenos científicos a través de una interfaz interactiva y accesible para el usuario.

El desarrollo del proyecto se ha realizado íntegramente en Python. Para la visualización interactiva de los resultados y la creación de la interfaz web, se han utilizado las librerías Panel y Bokeh. Además, los datos relativos a las propiedades ópticas de los materiales se han gestionado mediante una base de datos relacional SQLite.

# Abstract

This project involves the development of a web tool that enables the comparison of the electromagnetic scattering response of spherical particles made from different materials using Mie theory.

This theory, formulated in the early 20th century, provides a theoretical description of how light is scattered when it interacts with spherical particles, especially when their size is comparable to the wavelength of the light. This phenomenon is highly relevant in fields such as optics, nanophotonics, materials science, and nanotechnology, as it allows for the prediction and analysis of the electromagnetic behavior of nanometer-sized particles in different media and under varying environmental conditions.

The main objective of the developed tool is to provide an intuitive and functional interface that facilitates the visual comparison of light scattering by particles made of different materials and sizes through the generation of graphs. For this purpose, real refractive index data obtained from scientific catalogs are used, and the necessary calculations are applied to obtain the light scattering results based on the selected material and other influencing factors.

In addition to serving as a support tool in research environments, this project aims to demonstrate the integration of complex physical concepts with web development technologies, thus promoting the understanding and visualization of scientific phenomena through an interactive and user-friendly interface.

The entire project has been developed in Python. For the interactive visualization of the results and the creation of the web interface, the Panel and Bokeh libraries have been used. Additionally, data related to the optical properties of materials is managed through a relational SQLite database.

# Índice

# Índice de figuras

# 1-Introducción

Las ecuaciones de Maxwell describen el comportamiento de los campos electromagnéticos, así como su propagación en distintos medios. Estas ecuaciones forman la base de la teoría electromagnética clásica y permiten modelar de forma precisa una amplia variedad de situaciones físicas, entre ellas la dispersión de la luz por partículas.

En el campo de la nanotecnología, muchas partículas tienen un tamaño similar al de la longitud de onda de la luz (varios cientos de nanómetros). En estos casos, no basta con aproximaciones simples, ya que la luz interactúa de forma compleja con la partícula.

En este contexto, la teoría o solución de Mie [1] ofrece una solución analítica a las ecuaciones de Maxwell para la dispersión de la radiación electromagnética para el caso de partículas esféricas. Esta solución permite calcular los campos eléctricos y magnéticos producidos fuera y dentro de un objeto esférico sobre el que se hace incidir dicha radiación. En estos casos, debemos tener en cuenta que la luz incide en una partícula que tiene un determinado radio, que la partícula es de un material concreto con sus propiedades ópticas correspondientes, y que la luz viaja a través de un medio con un índice de refracción concreto.

Como resultado de esta interacción entre la luz y la partícula esférica, obtenemos una dispersión de la luz que se recoge en tres propiedades:

- Coeficiente de dispersión (qsca): Representa la parte de la energía que se desvía de la dirección original de propagación.
- Coeficiente de absorción (qabs): Indica la fracción de energía absorbida por la partícula.
- Coeficiente de extinción (qext): Cuantifica la pérdida total de intensidad de la onda incidente debido tanto a la dispersión como a la absorción.
  Es decir, $q_{\{ext\}} = q_{\{sca\}} + q_{\{abs\}}$

En la Figura 1 vemos un ejemplo de lo explicado anteriormente.

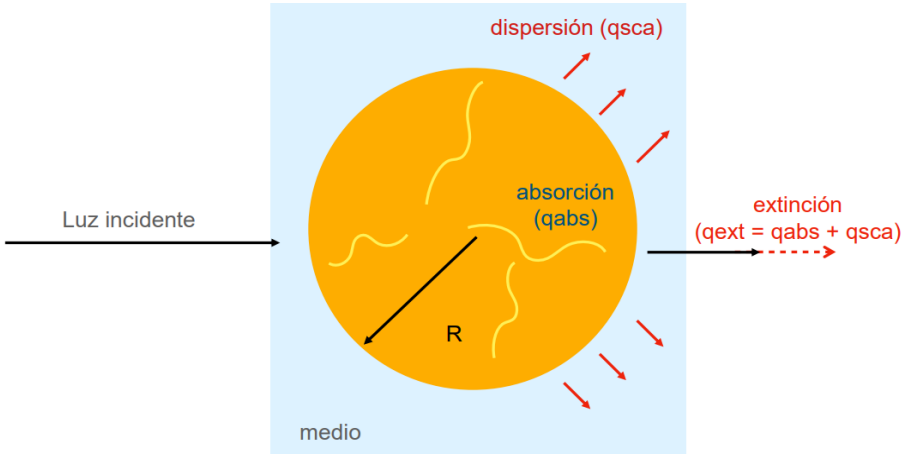


*Figura 1 – Ejemplo teoría de Mie*

## 1.1 Motivación

Las características del material, el tamaño de la esfera y el índice de refracción del medio afectan a la dispersión que se obtiene, siendo entonces posible obtener una dispersión deseada para una aplicación concreta mediante la selección del material adecuado y el tamaño de la partícula. No obstante, el uso de las fórmulas de Mie para esta selección puede ser tedioso. Esto se debe a que, aunque existen varias librerías software que implementan estas fórmulas, estas requieren de conocimientos de programación, tanto para la ejecución de las fórmulas, como para la obtención de las propiedades ópticas de los materiales a utilizar, normalmente extraídas de libros, artículos científicos, o de bases de datos. Es por esto por lo que surge la necesidad de crear una herramienta web interactiva y visual que simplifique este proceso.

## 1.2 Objetivos

El objetivo de este trabajo es desarrollar un producto software que permita a usuarios sin conocimientos técnicos en programación comparar la difracción obtenida al aplicar Mie sobre distintos materiales.

La herramienta, a grandes rasgos, permitirá a sus usuarios el siguiente proceso:

1. Seleccionar los materiales deseados de un catálogo.

2. Calcular la absorción y dispersión obtenida para cada material utilizando una implementación de Mie.

3. Mostrar gráficamente los resultados obtenidos para cada material, de forma que el usuario pueda compararlos.

4. Permitir exportar la información obtenida.

Además, el usuario de la herramienta podrá añadir o eliminar materiales de la comparativa en cualquier momento, y la herramienta deberá dinámicamente responder a estos cambios actualizando la información mostrada.

# 2-Background

En este apartado se presentan las distintas tecnologías y herramientas utilizadas a lo largo del desarrollo del proyecto, explicando el papel que desempeña cada una y su aportación al trabajo realizado.

## 2.1 Python

Python es el lenguaje de programación utilizado para realizar este proyecto debido a su sencillez y versatilidad. Su sintaxis sencilla y clara facilita la escritura y mantenimiento del código, mientras que su gran versatilidad permite abordar tanto tareas de cálculo numérico como la creación de interfaces gráficas o la gestión de datos. Además de contar con varias librerías útiles para el tipo de proyecto desarrollado, Python es uno de los lenguajes de programación con los que más familiarizados están los físicos, lo que lo convierte en una elección ideal pensando en la continuidad y mantenimiento de este trabajo.

## 2.2 PyCharm

PyCharm es el entorno de desarrollo integrado (IDE) elegido para la programación del proyecto. Esta herramienta ofrece una interfaz intuitiva y una serie de funcionalidades avanzadas que han sido de gran utilidad durante el desarrollo, como la detección de errores en tiempo real, la gestión de entornos virtuales y la integración con sistemas de control de versiones.

## 2.3 Git

Git es el sistema de control de versiones utilizado en este proyecto. Esta herramienta permite crear un repositorio donde ir almacenando el proyecto, así como trabajar con ramas que facilitan el desarrollo de nuevas funcionalidades sin afectar la versión principal del mismo. Git es útil, también, para mantener un historial de versiones con los cambios que se van realizando, permitiendo volver a versiones anteriores si fuese necesario. Además, hace posible compartir las actualizaciones del proyecto de forma cómoda e instantánea con otras personas.

## 2.4 SQLite y DB Browser

SQLite es el sistema de gestión de bases de datos relacional empleado para almacenar la información sobre los materiales y sus propiedades. Su integración con Python, a través del módulo sqlite3, nos permite realizar consultas desde el propio código del proyecto. Para su creación y gestión se ha utilizado DB Browser for SQLite, una herramienta gráfica que permite diseñar y modificar fácilmente la estructura de la base de datos, así como visualizar y editar los datos.

## 2.5 Panel

Panel [2] es el framework utilizado para el desarrollo de la interfaz gráfica del proyecto. Nos permite construir aplicaciones web interactivas directamente desde Python, sin necesidad de escribir código en HTML o CSS.

En el contexto de este trabajo, Panel ha sido utilizado para desarrollar la interfaz gráfica de usuario, permitiendo crear una experiencia interactiva y fácil de usar. Esto incluye, por ejemplo, la disposición de los elementos visuales, los desplegables o los botones. Panel nos permite, también, gestionar los eventos y procesos que se activan como respuesta a las acciones del usuario.

## 2.6 Bokeh

Bokeh [3] es una biblioteca de visualización interactiva en Python. Gracias a su integración con Panel, ha sido empleada para la representación visual de los resultados obtenidos con los cálculos de Mie. Bokeh es una herramienta muy potente que ofrece múltiples opciones interactivas para la visualización de gráficas, como el zoom, recortes y filtros.

## 2.7 MiePython

MiePython [4] es una biblioteca especializada en la implementación de la teoría de Mie. En este proyecto, ha sido utilizada para calcular los coeficientes de extinción, dispersión y absorción de distintos materiales en función de parámetros como la longitud de onda, el radio de la partícula y el índice de refracción complejo. Gracias a esta librería, ha sido posible obtener resultados precisos y rápidos sin necesidad de desarrollar desde cero los algoritmos matemáticos implicados.

## 2.8 RefractiveIndex.info

La fuente de datos utilizada en este proyecto es el repositorio online RefractiveIndex.info [5]. Esta plataforma colaborativa recoge datos experimentales del índice de refracción de una amplia variedad de materiales, mayoritariamente provenientes de publicaciones científicas.

En general, el índice de refracción de un material se representa mediante un número complejo formado por:

- Una parte real n, que indica la velocidad a la que se propaga la luz dentro del material.
- Una parte imaginaria k, que representa la absorción o pérdida de energía (coeficiente de extinción).

Estos valores no son fijos, sino que varían en función de la longitud de onda de la luz incidente, por lo que un índice de refracción se representa como una tabla con tres columnas: longitud de onda, n, y k.

RefractiveIndex.info organiza sus datos mediante una estructura jerárquica:

- Shelf (estantería): categoría general de los materiales.
- Book (libro): identifica un material específico dentro de una shelf (por ejemplo, Au para el oro).
- Page (página): contiene un conjunto concreto de datos experimentales para ese material, correspondientes a una fuente bibliográfica determinada. Un mismo material puede tener varias páginas si hay distintos conjuntos de medidas disponibles.

## 2.9 Otros

Además de las mencionadas anteriormente, han sido utilizadas en este trabajo otras herramientas como, por ejemplo:

- **Unittest**: Módulo estándar de Python para realizar pruebas unitarias y verificar el correcto funcionamiento del código.
- **Tempfile**: Permite crear archivos y directorios temporales de forma segura durante la ejecución del programa. Necesario para la funcionalidad de descarga o exportación de los datos.
- **Zipfile**: Utilizado para la compresión de archivos y su empaquetado en formato ZIP. Necesario para la funcionalidad de descarga o exportación de los datos.
- **Os**: Proporciona funciones para interactuar con el sistema operativo, como manipulación de rutas y archivos.
- **Abc**: Permite definir clases abstractas para estructurar el diseño del software de manera más clara y mantenible. Utilizado para la creación de interfaces.
- **Numpy**: Biblioteca para cálculo numérico. Utilizada para realizar los cálculos necesarios con la información de los materiales.
- **RefractiveSQLite**: Un módulo que nos permite acceder fácilmente a la base de datos de los materiales.

# 3- Metodología

En este apartado se describe la metodología empleada para la elaboración de este proyecto, así como la planificación seguida a lo largo del mismo.

En primer lugar, es importante señalar el rol desempeñado por cada uno de los dos directores del TFG a lo largo del desarrollo del proyecto:

- Yael Gutiérrez asumió el rol de Product Owner, es decir, fue la persona encargada de definir los objetivos funcionales del proyecto y sugerir mejoras. También es la persona experta en Óptica y en la Teoría de Mie utilizada en el proyecto.
- Alfonso de la Vega supervisó los aspectos técnicos y metodológicos del proyecto de Ingeniería del Software, asistiendo al alumno durante todo el transcurso del mismo.

Para el desarrollo de este trabajo se ha seguido una metodología de trabajo iterativa e incremental, inspirada en la aplicada durante el Proyecto Integrado realizado en el marco de las asignaturas “Calidad y Auditoría”, “Métodos de Desarrollo” y “Procesos de Ingeniería del Software”. Aunque dicha metodología sirvió como una base sólida, fue necesario simplificarla y adaptarla al contexto de un proyecto desarrollado de forma individual. Se optó por tomar este camino debido a que permite una evolución continua del producto, con ciclos de mejora basados en la retroalimentación obtenida a lo largo del proceso. Sin embargo, las primeras semanas fueron de toma de contacto con las herramientas, hasta que se obtuvo una aplicación inicial sobre la que poder iterar.

Desde el comienzo del proyecto, se realizaron reuniones periódicas con la Product Owner, quien actuaba como figura de referencia para validar el avance del proyecto y proponer mejoras. Estas reuniones no seguían una cadencia estrictamente fija, sino que se convocaban en función del progreso alcanzado o cuando surgían dudas relevantes en cuanto a diseño o implementación. En cada reunión se exponían los avances conseguidos y se definían las nuevas tareas o las correcciones a realizar.

Paralelamente a esto, se realizaban reuniones semanales con el profesor donde revisábamos que los avances fueran correctos desde un punto de vista técnico y se resolvían dificultades tanto de implementación como de diseño de la arquitectura.

Para organizar el trabajo y mantener un control preciso de la evolución del proyecto, se utilizó control de versiones mediante un repositorio en GitHub. Esto permitió documentar todos los cambios, mantener copias de seguridad de todas las versiones del producto y facilitar la detección de errores. Además, el uso de las ramas fue muy útil a la hora de añadir nuevas funcionalidades sin afectar a la versión principal del proyecto [6].

# 4-Análisis de requisitos

En este apartado, tras hacer una descripción general de la aplicación, se detallan los requisitos necesarios para su funcionamiento. Se distinguen entre requisitos funcionales, que describen las funcionalidades que debe ofrecer el sistema, y requisitos no funcionales, que definen las características de calidad que debe cumplir.

## 4.1 Descripción general

La aplicación cuenta con una vista de usuario única. En la Figura 2 vemos la interfaz inicial cuando se ejecuta la aplicación.

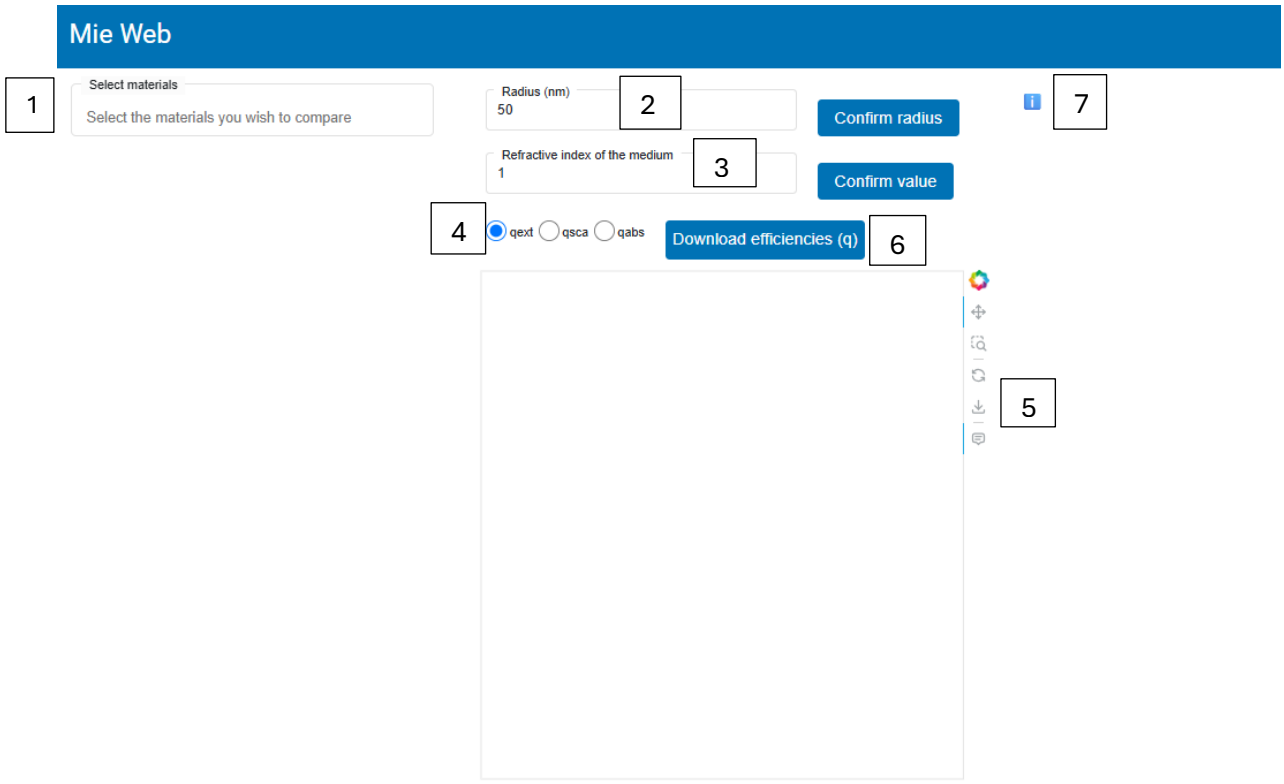


*Figura 2 – Estado inicial de la interfaz al abrir la aplicación*

Por defecto no hay ningún material seleccionado. El usuario debe seleccionar los materiales que desee comparar (1). Por cada material seleccionado aparece un desplegable en el que el usuario debe seleccionar que página (fuente de datos) desea elegir.

Por otro lado, se incluyen campos de texto para introducir los valores correspondientes al radio de la esfera (2) y al índice de refracción del medio (3). Siguiendo las indicaciones de la Product Owner, se establecieron como valores por defecto 50 nanómetros y 1, respectivamente.

El usuario puede, en todo momento, modificar tanto la selección de materiales como los valores introducidos para el radio de la esfera y el índice de refracción del medio. Una vez confirmados los cambios mediante los botones correspondientes, la gráfica se actualiza automáticamente para reflejar la nueva configuración.

Con la configuración previamente indicada, la aplicación está en condiciones de calcular y mostrar las gráficas correspondientes. El usuario tiene la posibilidad de alternar entre las distintas representaciones gráficas utilizando los botones etiquetados como “qext”, “qsca” y “qabs” (4), los cuales permiten visualizar, respectivamente, los coeficientes de extinción, dispersión y absorción calculados mediante la solución de Mie.

Adicionalmente, la aplicación ofrece otras funcionalidades que buscan mejorar la experiencia del usuario a la hora de analizar materiales:

- Descarga de la gráfica (5): permite exportar la gráfica generada en formato .svg, facilitando su incorporación en informes o presentaciones.
- Exportación de datos (6): posibilita la descarga de un archivo comprimido .zip que contiene, para cada material representado, un fichero de texto con los valores numéricos de los coeficientes calculados.
- Interacción con la gráfica: el usuario puede modificar la vista de la gráfica realizando zoom sobre una región específica para analizar con mayor detalle los resultados.
- Visualización de ayuda (7): al seleccionar el icono de información, se muestra una descripción general sobre el funcionamiento y propósito de la aplicación.

En la Figura 3, se muestra un ejemplo de una consulta realizada a través de la aplicación. En este caso, el usuario selecciona dos materiales de la lista disponible: Silver (Ag) y Aluminium (Al). Para cada uno, elige una página específica dentro de las opciones disponibles, en este caso “Johnson” y “McPeak” respectivamente.

Posteriormente, el usuario introduce un valor de radio de 100 nm para el cálculo de la solución de Mie y un índice de refracción del medio de 1. La aplicación realiza los cálculos correspondientes y muestra las curvas de extinción, dispersión y absorción para cada material, facilitando la comparación visual de sus propiedades ópticas.

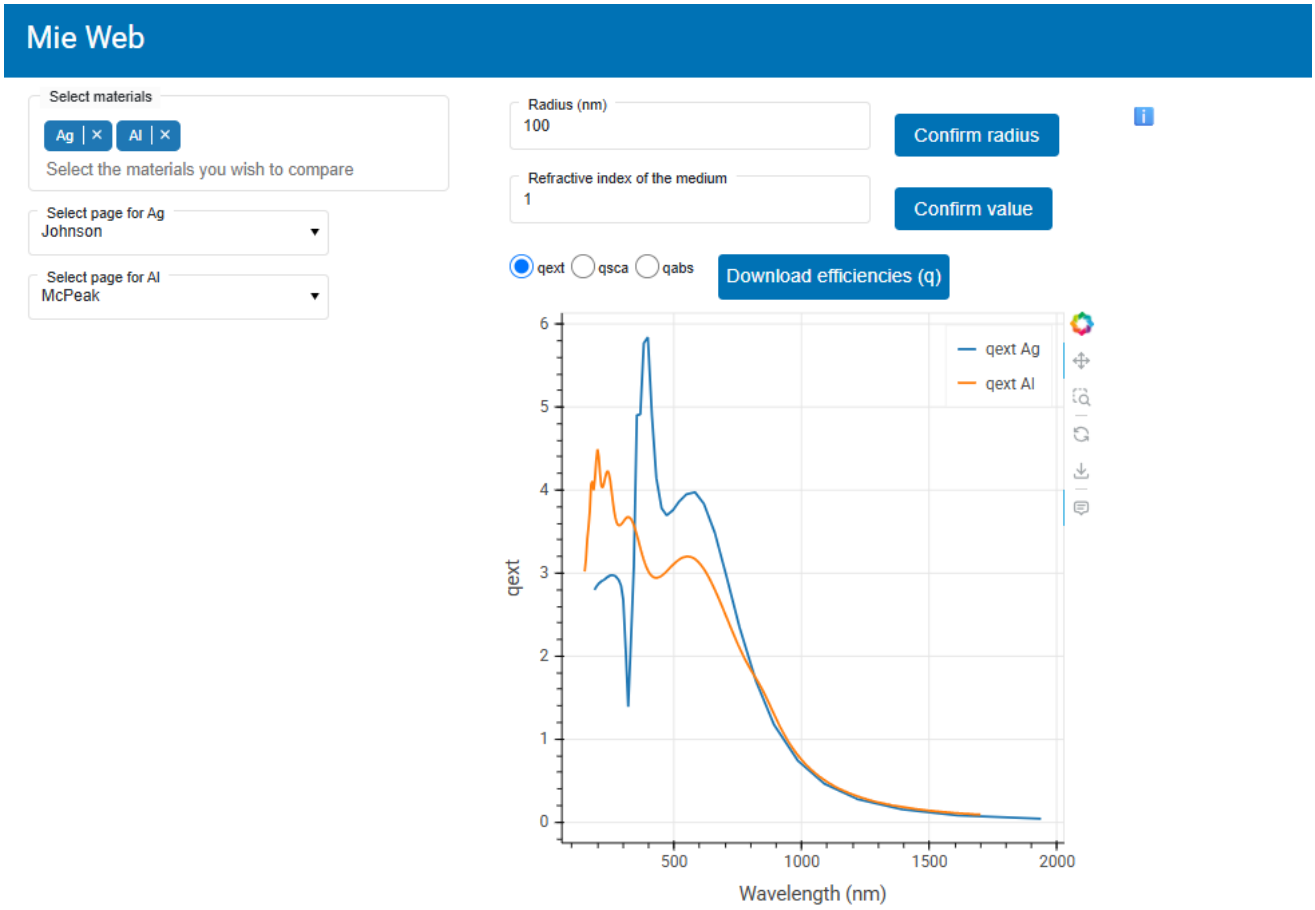


*Figura 3 – Ejemplo de uso de la aplicación*

## 4.2 Requisitos funcionales

Para abordar de forma estructurada todas las funcionalidades que se deseaban implementar en la aplicación, se elaboraron una serie de historias de usuario. Estas historias permiten definir los requisitos desde la perspectiva del usuario final facilitando así la identificación de las funcionalidades que hay que implementar.

| Identificador | RF01 |
|---|---|
| Requisito | Añadir material a la comparación |
| Descripción | Yo, como usuario, quiero poder seleccionar materiales de una lista de materiales disponibles,<br>de manera que pueda incluirlos en la comparación y analizar sus propiedades ópticas. |
| Flujo principal | 1- El usuario navega por la lista de materiales y selecciona uno o más materiales de interés.<br>2- Los materiales elegidos aparecen como seleccionados.<br>3- Se verifica que aparece, para cada material seleccionado, una opción para elegir la página que se desee. |

| Identificador | RF02 |
|---|---|
| Requisito | Eliminar material de la comparación |
| Descripción | Yo, como usuario, quiero poder eliminar materiales previamente seleccionados de manera que pueda modificar dicha selección |
| Flujo principal | 1- El usuario deselecciona los materiales que quiera eliminar de la selección.<br>2- Los materiales deseleccionados desaparecen de la selección.<br>3- Se verifica que la gráfica se actualiza eliminando los materiales deseleccionados. |

| Identificador | RF03 |
|---|---|
| Requisito | Seleccionar la página de un material |
| Descripción | Yo, como usuario, quiero poder seleccionar una página específica de un material dentro de los datos disponibles, de manera que pueda elegir la fuente de información más relevante para mi análisis. |
| Flujo principal | 1- El usuario selecciona la página que desee como fuente de datos para uno de los materiales que previamente ha sido añadido a la comparativa.<br>2- Se verifica que el sistema carga los datos de la página seleccionada. |
| Flujo alternativo | El radio y el índice de refracción del medio están establecidos<br>1- El usuario selecciona la página que desee como fuente de datos para uno de los materiales que previamente ha sido añadido a la comparativa.<br>2- Se verifica que el sistema carga los datos de la página seleccionada.<br>3- Se verifica que se actualiza la gráfica. |

| Identificador | RF04 |
|---|---|
| Requisito | Establecer el valor del radio para el cálculo de Mie |
| Descripción | Yo, como usuario, quiero poder introducir un valor de radio para el cálculo de Mie, de manera que pueda ajustar los parámetros del análisis a mis necesidades. |
| Flujo principal | 1- El usuario introduce un valor válido para el radio.<br>2- Se verifica que la gráfica se actualiza utilizando el nuevo valor del radio en los cálculos. |
| Flujo alternativo | Valor de radio no válido<br>1- El usuario introduce un valor no válido para el radio*.<br>2- Se verifica que se notifica el error al usuario solicitándole que introduzca un valor válido (>0).<br><br>*Nota: Solo se consideran válidos los valores numéricos mayores que cero. |

| Identificador | RF05 |
|---|---|
| Requisito | Establecer el valor del n del medio para el cálculo de Mie |
| Descripción | Yo, como usuario, quiero poder introducir un valor del índice de refracción del medio (n), de manera que pueda realizar los cálculos considerando el entorno en el que se encuentran los materiales. |
| Flujo principal | 1- El usuario introduce un valor válido para el índice de refracción del medio.<br>2- Se verifica que la gráfica se actualiza utilizando el nuevo valor del índice de refracción en los cálculos. |
| Flujo alternativo | 1- El usuario introduce un valor no válido para el índice de refracción.<br>2- Se verifica que se notifica el error al usuario solicitándole que introduzca un valor válido (>0)<br>Nota: Solo se consideran válidos los valores numéricos mayores que cero |

| Identificador | RF06 |
|---|---|
| Requisito | Descargar la gráfica como SVG |
| Descripción | Yo, como usuario, quiero poder descargar la gráfica generada como un archivo SVG,<br>de manera que pueda guardar y compartir los resultados del análisis fácilmente y sin perder calidad de imagen. |
| Flujo principal | 1- El usuario selecciona la opción de descargar la gráfica.<br>2- Se verifica que la aplicación genera una imagen de la gráfica en formato SVG y la descarga. |

| Identificador | RF07 |
|---|---|
| Requisito | Descarga de un ZIP con los datos de los materiales seleccionados |
| Descripción | Yo, como usuario, quiero poder descargar un archivo ZIP que contenga los datos de los materiales seleccionados de manera que pueda almacenarlos y consultarlos o analizarlos posteriormente. |
| Flujo principal | 1- El usuario selecciona la opción de descargar los datos de los materiales.<br>2- Se verifica que la aplicación genera un archivo comprimido con ficheros que contienen los datos de los materiales de la gráfica. |

| Identificador | RF08 |
|---|---|
| Requisito | Intercambio entre distintas gráficas (qext, qsca, qabs) |
| Descripción | Yo, como usuario, quiero poder alternar entre diferentes tipos de gráficos (extinción, dispersión y absorción),<br>de manera que pueda analizar cada uno de los posibles resultados de forma independiente. |
| Flujo principal | 1- El usuario selecciona la opción de gráfica que prefiere.<br>2- Se verifica que la aplicación alterna la gráfica que se muestra. |

| Identificador | RF09 |
|---|---|
| Requisito | Despliegue de información |
| Descripción | Yo, como usuario, quiero poder acceder a unas instrucciones básicas de uso de la aplicación, de manera que resulte sencilla la comprensión del funcionamiento de la herramienta. |
| Flujo principal | 1- El usuario selecciona el botón de información.<br>2- Se verifica que la aplicación muestra información con instrucciones de uso de la aplicación<br>Extra: Cerrar el diálogo de información. |

## 4.3 Requisitos no funcionales

Los siguientes requisitos no funcionales fueron impuestos al comienzo del desarrollo del proyecto.

| | |
|---|---|
| Identificador | RNF01 |
| Requisito | Idioma de la interfaz gráfica |
| Descripción | Se solicitó que la interfaz de la herramienta estuviese en inglés para facilitar la comprensión por parte de otros desarrolladores y usuarios. |

| | |
|---|---|
| Identificador | RNF02 |
| Requisito | Lenguaje de programación |
| Descripción | Se estableció como requisito utilizar Python como lenguaje principal, aprovechando sus capacidades para el procesamiento numérico y el conocimiento del lenguaje ya existente en la comunidad científica. |

| | |
|---|---|
| Identificador | RNF03 |
| Requisito | Fuente de datos |
| Descripción | Los datos ópticos utilizados en el proyecto debían obtenerse del repositorio RefractiveIndex.info, por su amplio catálogo de materiales y su uso común en trabajos científicos relacionados con óptica y fotónica. |

# 5- Diseño y arquitectura

En este apartado se presenta el diseño y la arquitectura de la aplicación desarrollada. La arquitectura define la estructura interna del sistema, organizando sus componentes de forma que se facilite la comprensión, el mantenimiento y la escalabilidad del software.

Para este proyecto se ha adoptado una arquitectura en 3 capas: Presentación, negocio y persistencia. Cada capa tiene unas funciones muy definidas, lo que permite una separación de responsabilidades que facilita el mantenimiento y la escalabilidad del código.

Además, esta estrategia nos ofrece un aislamiento e independencia entre las distintas capas que permite hacer modificaciones sin afectar al funcionamiento de las demás.

La comunicación entre las capas de presentación y negocio se ha gestionado mediante la aplicación del patrón Modelo-Vista-Presentador (Model-View-Presenter, MVP). En este patrón, la View se encarga únicamente de la interfaz gráfica y de mostrar los datos al usuario, mientras que el Presenter actúa como intermediario, gestionando la lógica de presentación y coordinando la comunicación entre la vista y la capa de negocio (Modelo).

En la Figura 4 vemos un esquema del diseño que se ha seguido para este proyecto.

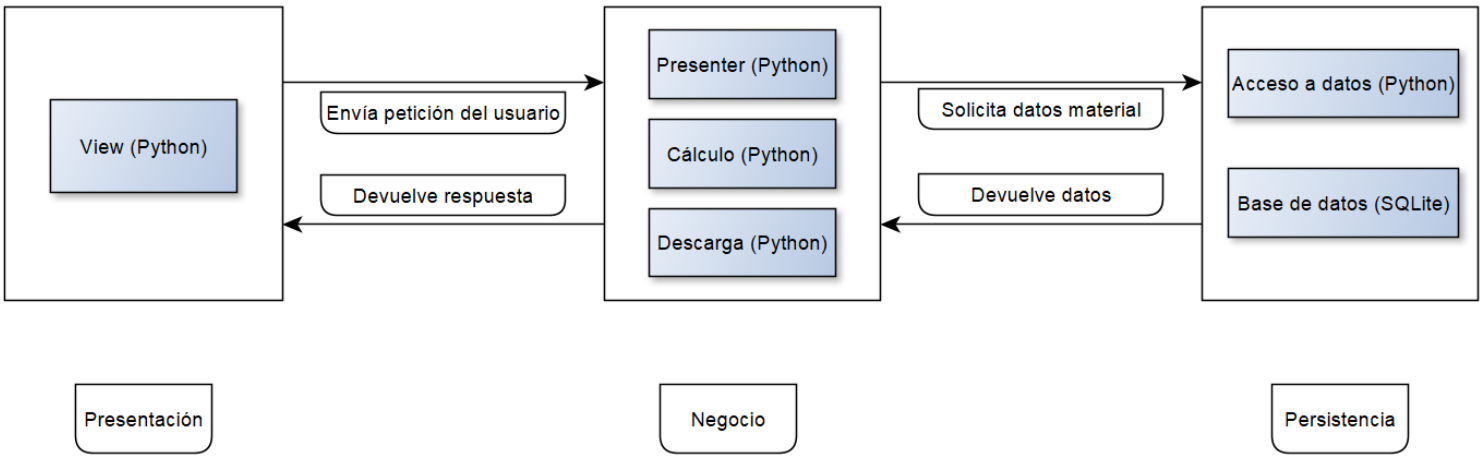


*Figura 4 – Diseño arquitectónico*

## 5.1 Capa de presentación

La capa de presentación constituye la interfaz entre el usuario y la aplicación. Es la responsable de mostrar la información y de responder a las interacciones con el usuario. El principal objetivo que se busca con esta capa es ofrecer una experiencia de uso clara, intuitiva y funcional que permita al usuario seleccionar materiales, visualizar resultados y exportar los datos obtenidos.

Para la implementación de esta capa se ha utilizado Panel [2], una biblioteca de alto nivel para Python que permite crear interfaces web interactivas de forma sencilla. Además, facilita la integración de los diferentes elementos visuales como botones, menús y gráficas, sin necesidad de utilizar tecnologías web externas como HTML o JavaScript. Esto ha permitido desarrollar una interfaz completamente funcional utilizando únicamente Python, lo cual simplificó el proceso.

La capa de presentación actúa como intermediaria entre el usuario y la lógica de la aplicación, interactuando directamente con la capa de negocio. Esto significa que cuando el usuario realiza una acción, la capa de presentación se encarga de solicitar a la capa de negocio lo que necesite. A continuación, la capa de negocio responde con la información solicitada y la capa de presentación se encarga de presentarle los datos al usuario. De este modo se mantiene una independencia entre la lógica de la aplicación y la visualización de los resultados.

## 5.2 Capa de negocio

La capa de negocio representa el núcleo funcional de la aplicación y actúa como intermediaria entre la capa de presentación y la capa de persistencia.

En este proyecto, la capa de negocio se encarga de coordinar la obtención de los índices de refracción de los materiales seleccionados y de la realización de los cálculos correspondientes utilizando la solución de Mie. También se encarga de preparar la descarga de datos. Para ello, recibe los materiales seleccionados y los parámetros establecidos de la capa de presentación y solicita a la capa de persistencia los datos necesarios. Una vez cuenta con toda esta información, realiza los cálculos necesarios para obtener los coeficientes de extinción, dispersión y absorción.

## 5.3 Capa de persistencia

La capa de persistencia es la encargada de gestionar el acceso a los datos almacenados en la base de datos. Su función principal es abstraer a la capa de negocio de las tareas relacionadas con el almacenamiento y organización de la información. Esto permite que el resto de la aplicación trabaje con datos ya preparados sin preocuparse por cómo se obtienen.

En el contexto de este proyecto, los datos se almacenan en una base de datos SQLite que permite:

- Obtener la lista de materiales disponibles.
- Obtener los valores de índice de refracción (n y k) y la longitud de onda para una página concreta de un material.

De esta forma, la capa de negocio no necesita saber cómo están guardados los datos ni cómo se accede a ellos, ya que todo eso lo gestiona la capa de persistencia. Esto también hace que sea más fácil modificar o ampliar la base de datos en el futuro sin que el resto de la aplicación se vea afectada por los cambios si no es necesario.

# 6- Implementación

En este apartado se describe como se ha llevado a cabo la implementación del proyecto, desde la idea original hasta el producto final. El código se encuentra disponible en un repositorio de código abierto alojado en GitHub [6].

## 6.1 Comprobación inicial librería miePython

Inicialmente, se realizó una prueba simple para comprobar el funcionamiento de la librería miePython [4]. Esta prueba consistía en calcular los coeficientes de extinción, dispersión y absorción utilizando valores simulados para el índice de refracción (n y k), la longitud de onda y el radio de la partícula. El objetivo era entender cómo se utilizaba la librería y verificar que los resultados obtenidos eran correctos.

## 6.2 Base de datos

Tras la realización de la prueba inicial, se procedió a la creación de la base de datos. Para ello, se utilizó la base de datos de índices de refracción disponible en *refractiveindex.info [5]*. El catálogo de datos se organiza en ficheros YAML, uno por cada página de material disponible. Dichos ficheros contienen, además de los datos, una serie de metadatos de las páginas de los materiales. De primeras se consideró poco práctico trabajar directamente contra estos ficheros YAML, siendo necesario procesarlos para cada página que se deseara utilizar. No obstante, alrededor de este catálogo de datos existen varios proyectos de código abierto realizados para facilitar su exploración y uso. Tras un análisis de dichos proyectos, se realizó una generación de una base de datos SQLite mediante un repositorio de Github [7] que permite convertir los datos de dicho catálogo en una base de datos en formato SQLite.

En la Figura 5 se muestra cómo se ve la base de datos:

| | pageid | shelf | book | page | filepath |
|---|---|---|---|---|---|
| | Filtro | Filtro | Filtro | Filtro | Filtro |
| 1 | 0 | main | Ag | Johnson | main\Ag\Johnson.yml |
| 2 | 1 | main | Ag | Yang | main\Ag\Yang.yml |
| 3 | 2 | main | Ag | McPeak | main\Ag\McPeak.yml |
| 4 | 3 | main | Ag | Babar | main\Ag\Babar.yml |
| 5 | 4 | main | Ag | Wu | main\Ag\Wu.yml |
| 6 | 5 | main | Ag | Werner | main\Ag\Werner.yml |
| 7 | 6 | main | Ag | Stahrenberg | main\Ag\Stahrenberg.yml |
| 8 | 7 | main | Ag | Windt | main\Ag\Windt.yml |
| 9 | 8 | main | Ag | Hagemann | main\Ag\Hagemann.yml |
| 10 | 9 | main | Ag | Ciesielski | main\Ag\Ciesielski.yml |
| 11 | 10 | main | Ag | Ciesielski-Ge | main\Ag\Ciesielski-Ge.yml |
| 12 | 11 | main | Ag | Ciesielski-Ni | main\Ag\Ciesielski-Ni.yml |
| 13 | 12 | main | Ag | Rakic-BB | main\Ag\Rakic-BB.yml |
| 14 | 13 | main | Ag | Rakic-LD | main\Ag\Rakic-LD.yml |
| 15 | 14 | main | Ag | Werner-DFT | main\Ag\Werner-DFT.yml |
| 16 | 15 | main | Al | Rakic | main\Al\Rakic.yml |
| 17 | 16 | main | Al | McPeak | main\Al\McPeak.yml |
| 18 | 17 | main | Al | Larruquert | main\Al\Larruquert.yml |
| 19 | 18 | main | Al | Ordal | main\Al\Ordal.yml |
| 20 | 19 | main | Al | Hagemann | main\Al\Hagemann.yml |
| 21 | 20 | main | Al | Mathewson | main\Al\Mathewson.yml |
| 22 | 21 | main | Al | Rakic-BB | main\Al\Rakic-BB.yml |
| 23 | 22 | main | Al | Rakic-LD | main\Al\Rakic-LD.yml |

*Figura 5 – Estructura de la base de datos*

Una de las tablas principales generadas en este proceso es la tabla “pages”.que contiene los metadatos de los ficheros de datos ópticos de los materiales. Esta tabla incluye, entre otros, los siguientes campos:

- **pageid**: identificador único de cada conjunto de datos o entrada.
- **shelf**: categoría general a la que pertenece el material (por ejemplo, "main").
- **book**: símbolo químico o nombre del material (por ejemplo, "Ag" para plata o "Al" para aluminio).
- **page**: nombre del autor o grupo de autores del conjunto de datos, que sirve para distinguir diferentes fuentes dentro del mismo material.
- **filepath**: ruta relativa del archivo .yml que contiene los datos de índice de refracción para ese autor y material específicos.

Esta estructura permite gestionar múltiples fuentes de datos para un mismo material, lo cual resulta especialmente útil a la hora de realizar comparativas entre estudios y seleccionar la información más adecuada para cada caso.

Por lo tanto, aunque los datos ópticos provienen originalmente de archivos .yml, en la aplicación no se accede directamente a dichos archivos. En su lugar, se utilizan métodos específicos como get_material_n_numpy y get_material_k_numpy, que permiten obtener directamente los valores de longitud de onda, índice de refracción real (**n**) y coeficiente de absorción (**k**) a partir del identificador (pageid) almacenado en la base de datos SQLite. Esto hace que el acceso a la información sea más sencillo para su uso dentro de la aplicación.

## 6.3 Desarrollo de prototipo:

Una vez completada la creación y validación de la base de datos, se optó por comenzar con la implementación de algunas funcionalidades básicas antes de estructurar la aplicación siguiendo una arquitectura en tres capas. Esta decisión se tomó con el objetivo de probar y consolidar progresivamente la lógica principal de la aplicación, permitiendo validar el comportamiento del sistema y agilizar el desarrollo inicial.

En esta primera etapa, se dio forma a aspectos esenciales como la lectura y carga de datos desde la base de datos, la visualización de resultados gráficos preliminares y la integración básica de los cálculos ópticos basados en la solución de Mie.

Esta serie de primeros pasos resultaron muy útiles para familiarizarse con las herramientas principales empleadas en el desarrollo, como la librería miepython (utilizada para realizar los cálculos de dispersión y absorción mediante la solución de Mie) y el framework Panel (empleado para construir la interfaz gráfica de usuario de forma sencilla e interactiva).

Además, estas primeras implementaciones permitieron construir una base funcional del sistema, probar los componentes esenciales y obtener una visión global del proyecto, facilitando así la planificación de las siguientes etapas de forma iterativa.

## 6.4 Implementación en capas

Una vez que se disponía de una base preliminar del proyecto, con las funcionalidades esenciales implementadas y en funcionamiento, se procedió a aplicar una arquitectura en tres capas con el objetivo de darle forma de producto software estructurado. Esta transición hacia una arquitectura en tres capas permitió que el producto fuera más escalable, facilitó su mantenimiento y aportó una independencia clara entre las capas de presentación, lógica de negocio y acceso a datos.

La implementación de la arquitectura en tres capas se organizó en tres paquetes principales dentro del proyecto: presentación, negocio y persistencia.

### 6.4.1 Capa de presentación:

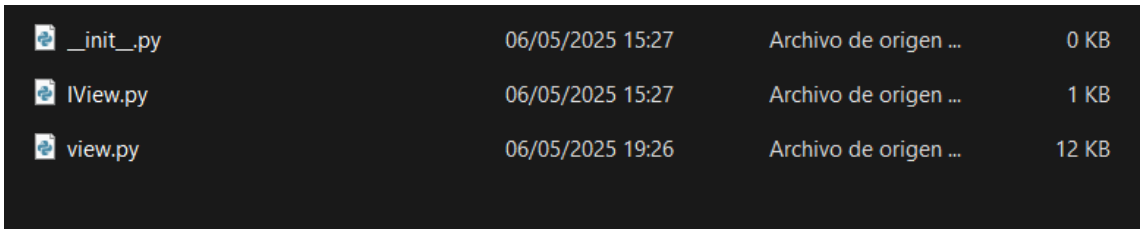


*Figura 6 – Estructura de la capa de presentación*

Como vemos en la Figura 6, dentro de esta capa se implementa una clase View, junto con su correspondiente interfaz, cuya responsabilidad principal es gestionar la interacción con el usuario y mostrarle la información de forma clara y estructurada. Esta clase se encarga de construir la interfaz gráfica utilizando Panel, disponiendo elementos como menús desplegables, botones, gráficas y otros componentes visuales.

Además, la View también se encarga de comunicarse con el Presenter de la capa de negocio. Cada vez que el usuario realiza una acción (por ejemplo, seleccionar un material), la View traslada esta solicitud al Presenter, quien se encarga de procesarla y devolver los resultados correspondientes para ser mostrados en pantalla. De esta forma, se mantiene una separación clara entre la interfaz de usuario y la lógica de negocio.

### 6.4.2 Capa de negocio

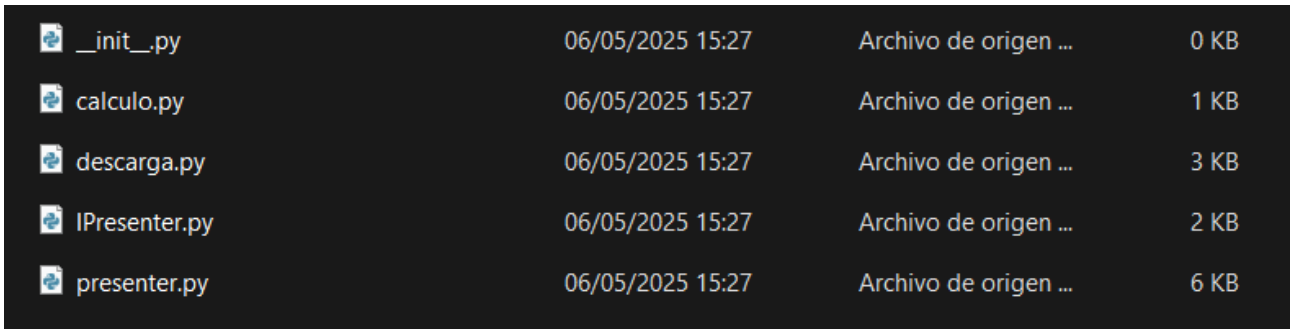


*Figura 7 – Estructura de la capa de negocio*

La capa de negocio constituye el núcleo funcional del sistema. Su principal objetivo es gestionar la lógica de la aplicación, coordinando la interacción entre la interfaz de usuario y los datos. Además, se encarga de realizar los cálculos ópticos necesarios y preparar la información para devolvérsela a la capa de presentación que presentará los resultados.

Como vemos en la Figura 7, esta capa se compone de tres módulos principales:

- Presenter: actúa como intermediario entre la View y el resto de la lógica de negocio. Recibe las solicitudes procedentes de la capa de presentación (por ejemplo, la selección de un material), accede a los datos necesarios a través de la capa de persistencia, realiza los cálculos correspondientes y devuelve a la View los resultados listos para ser mostrados.
- Cálculo: se encarga de realizar los cálculos ópticos mediante la solución de Mie, utilizando la librería miepython. Para ello, toma como entrada los valores de longitud de onda, el índice de refracción complejo (n - ik), el radio de la partícula y el índice de refracción del medio, y devuelve los coeficientes de extinción, dispersión y absorción.
- Descarga: permite generar archivos de texto plano en formato tabla (separadas por tabuladores) con los resultados obtenidos, para que el usuario pueda guardarlos o analizarlos externamente.

### 6.4.3 Capa de persistencia:

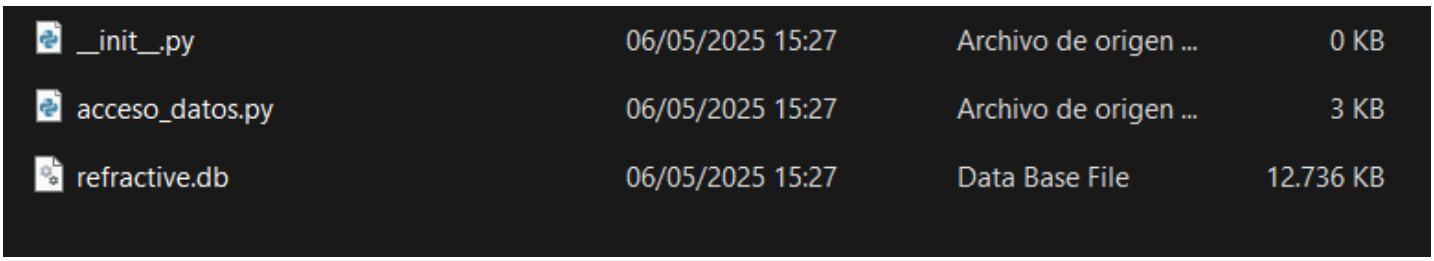


*Figura 8 – Estructura de la capa de persistencia*

La capa de persistencia es la encargada de gestionar el acceso a los datos almacenados en la base de datos SQLite, que contiene los valores del índice de refracción (n y k) de distintos materiales.

Su responsabilidad se limita a ofrecer acceso estructurado a los datos, permitiendo operaciones como:

- Obtener la lista de materiales disponibles en la base de datos.
- Para cada material, obtener sus páginas con los datos correspondientes.
- Recuperar los valores de n y k de una página concreta de un material para un rango dado de longitudes de onda.

Como vemos en la Figura 8 la capa de persistencia solo cuenta con la clase de acceso a datos. Se ha añadido también la base de datos en esta capa, ya que SQLite es un sistema relacional que carece de servidor en activo, y que simplemente almacena la base de datos en un fichero. La razón para almacenar la base de datos junto al código es facilitar el posterior despliegue público de la aplicación web.

## 6.5 Ejemplo de interacción View – Presenter:

En la Figura 9 podemos observar un diagrama con las interfaces de la View y el Presenter, mostrando los métodos que utilizan para comunicarse.

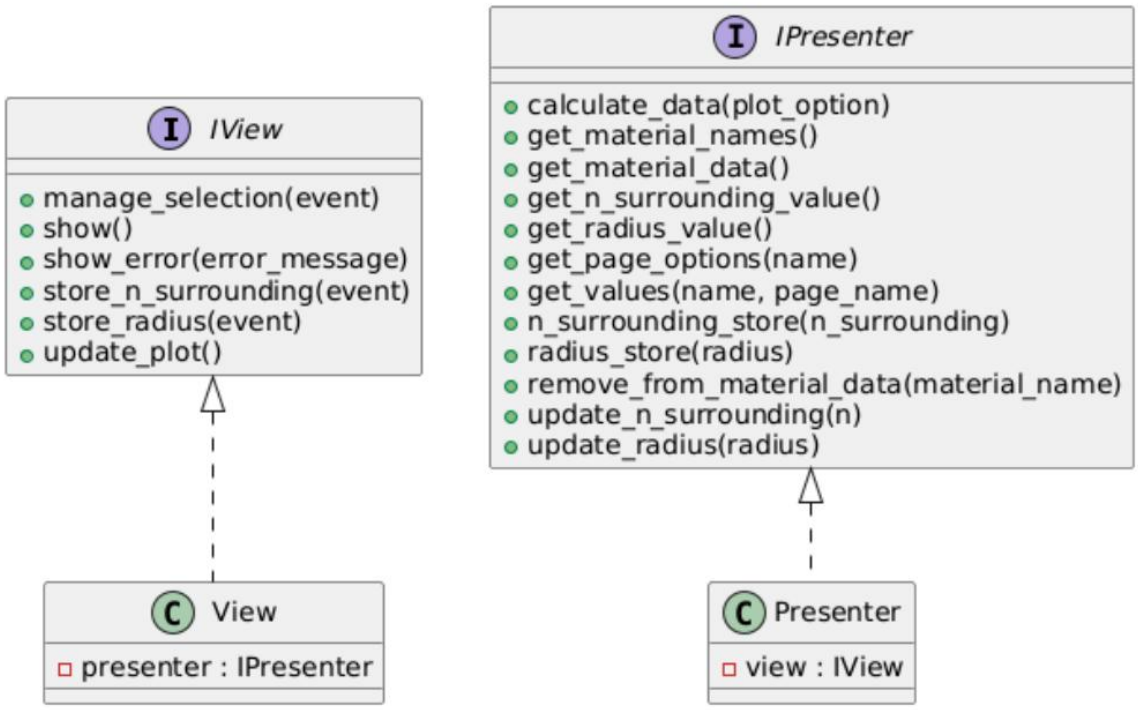


*Figura 9 – Diagrama de clases de la interacción View-Presenter*

Un ejemplo sencillo de la interacción entre capas es el caso en el que el usuario introduce el valor del radio de la esfera con la que desea realizar los cálculos.

El siguiente ejemplo ilustra cómo:

- La View se encarga únicamente de recoger la entrada del usuario y delegar la lógica al Presenter.
- El Presenter es quien, valida, almacena y decide las acciones a realizar en función del estado de la aplicación.
- La lógica de negocio está completamente desacoplada de la interfaz gráfica, cumpliendo con los principios de la arquitectura en capas.

1- Como vemos en la Figura 10, en la View se proporciona un campo de texto donde el usuario puede introducir el valor del radio en nanómetros, junto con un botón de confirmación para validar su entrada.

```
# Crear una entrada de texto para el radio
self.radius_input = pn.widgets.TextInput(
    name='Radius (nm)',
    placeholder='Enter the radius value in nanometers',
    value='50',  # Valor predeterminado
    width=300
)

# Botón para confirmar el radio
self.confirm_radius_button = pn.widgets.Button(
    name='Confirm radius',
    button_type='primary',
    width=50
)

# Adjuntar la función store_radius al evento del botón
self.confirm_radius_button.on_click(self.store_radius)
```

*Figura 10 – Fragmento de código para introducir el radio*

2- Como vemos en la Figura 11, cuando el usuario pulsa el botón, se ejecuta el método store_radius, definido en la View, que a su vez llama al método radius_store del Presenter, pasando el valor introducido como argumento.

```
# Función para manejar la entrada del radio
def store_radius(self,event):  2 usages  Sergio Salas
    self.presenter.radius_store(self.radius_input.value)
```

*Figura 11 – Fragmento de código manejo del valor del radio por la View*

Como vemos en la Figura 12, en el Presenter, el método radius_store intenta convertir el valor a float y valida que sea positivo:

- Si el valor es válido, se almacena mediante update_radius, se marca como correcto (valid_radius = True), se borra cualquier mensaje de error, y si el valor del índice del medio también es válido, se actualiza automáticamente la gráfica.
- Si el valor no es válido, se guarda igualmente (como string), se marca como inválido (valid_radius = False) y se muestra un mensaje de error en la View.

```
def radius_store(self, radius):  Sergio Salas *
    try:
        r = float(radius)
        if r <= 0:
            raise ValueError("Radius value must be greater than 0")

        self.update_radius(r)
        self.valid_radius = True
        self.view.show_error("")

        # Solo actualizamos la gráfica si ambos valores son válidos
        if self.valid_n_surrounding:
            self.view.update_plot()

        else:
            self.view.show_error("Enter a valid value for the refractive index of the medium")

    except ValueError:
        self.update_radius(radius)
        self.valid_radius = False
        self.view.show_error("Error: Enter a valid value for radius")
```

*Figura 12 – Fragmento de código manejo del valor del radio por el Presenter*

# 7- Pruebas

Para asegurar la fiabilidad y el correcto funcionamiento de la aplicación desarrollada, se han implementado diferentes tipos de pruebas durante el proceso de implementación. Estas pruebas se organizaron en tres niveles: unitarias, de integración y de aceptación, cubriendo así tanto el comportamiento de los módulos de forma individual como su integración con el resto del proyecto y la experiencia desde el punto de vista del usuario.

## 7.1 Pruebas unitarias

Las pruebas unitarias se centran en verificar el correcto funcionamiento de componentes individuales del código.

Para la realización de estas pruebas se utilizó “unittest” de Python, ya que nos permite ejecutar casos de prueba con facilidad y además nos ofrece aserciones para comprobar el comportamiento esperado del código.

En este proyecto, las pruebas unitarias se centraron en comprobar que la función “calculate_mie_arrays”, encargada del cálculo de los coeficientes de extinción (Qext), dispersión (Qsca) y absorción (Qabs), devolviera resultados correctos comparándolos con un conjunto de datos esperados previamente calculados y almacenados en archivos de texto proporcionados por la Product Owner.

Los tests siguen el siguiente esquema:

- Se extraen los datos experimentales esperados desde los archivos de texto plano.
- Se obtienen los datos ópticos del material a partir de la base de datos.
- Se realiza el cálculo usando la función de Mie.
- Se comparan los resultados calculados con los valores reales mediante assertAlmostEqual, con un margen de error pequeño (delta = 0.001) para compensar posibles imprecisiones numéricas.

En la Figura 13 tenemos la función read_tsv, la cual es utilizada para leer los archivos con valores de referencia de los coeficientes ópticos. Estos archivos están en formato TSV (valores separados por tabulaciones) y contienen las columnas: longitud de onda (Lambda), extinción (Qext), dispersión (Qsca) y absorción (Qabs).

```
def read_tsv(self, file_name):  5 usages  Sergio Salas

    base_dir = os.path.dirname(os.path.abspath(__file__))  # Carpeta 'test'
    file_path = os.path.join(base_dir, "..", "datos", file_name)  # Subir un nivel y entrar en 'datos'

    data = {
        "Lambda": [],
        "Qext": [],
        "Qsca": [],
        "Qabs": []
    }

    with open(file_path, mode='r') as file:
        reader = csv.reader(file, delimiter='\t')

        for row in reader:
            data["Lambda"].append(float(row[0]))
            data["Qext"].append(float(row[1]))
            data["Qsca"].append(float(row[2]))
            data["Qabs"].append(float(row[3]))

    return data
```

*Figura 13 – Método para leer datos de referencia de un archivo*

En la Figura 14 tenemos un test que verifica que los resultados obtenidos por el método calculate_mie_arrays para partículas de plata (Ag) con un radio de 25 nm en un medio con índice de refracción 1.0 coinciden con los datos esperados. Se comparan los valores calculados de extinción (Qext), dispersión (Qsca) y absorción (Qabs) con los de un archivo de referencia usando assertAlmostEqual, permitiendo un pequeño margen de error (delta=0.001).

```python
# Pruebas unitarias para el cálculo de Mie
def  test_calculo_mie_1(self):  Sergio Salas

    material_1 = get_pages_data( material_name: "Ag",  page_name: "Johnson")
    radius = 25  # Radio de la partícula en nanómetros
    n_surrounding = 1.0  # Índice de refracción del entorno (agua o aire, por ejemplo)

    results_material = calculate_mie_arrays(material_1, radius, n_surrounding)


    file_name = 'Q_Ag_Johnson_R_25_nmed_1.txt'
    real_data = self.read_tsv(file_name)

    for real_value, calc_value in zip(real_data["Qext"], results_material["qext"]):
        self.assertAlmostEqual(real_value, calc_value, delta=0.001)

    for real_value, calc_value in zip(real_data["Qabs"], results_material["qabs"]):
        self.assertAlmostEqual(real_value, calc_value, delta=0.001)

    for real_value, calc_value in zip(real_data["Qsca"], results_material["qsca"]):
        self.assertAlmostEqual(real_value, calc_value, delta=0.001)
```

*Figura 14 – Prueba unitaria*

Como dato anecdótico, es interesante destacar que la realización de estos tests de pruebas unitarias permitió detectar un fallo en la implementación de la fórmula, concretamente al respecto del cambio en el índice de refracción del medio. Tras revisar esta divergencia con el Product Owner, se localizó un operador faltante en una parte de la fórmula.

## 7.2 Pruebas de integración

Las pruebas de integración permiten comprobar cómo se comporta una parte del código cuando interactúa con otras, en lugar de probarla de forma aislada como se hace en las pruebas unitarias. El objetivo es asegurarse de que los diferentes componentes del sistema se comuniquen correctamente entre sí.

Para desarrollar estas pruebas se ha utilizado MagicMock, una herramienta que ofrece la librería unittest.mock de Python. Los mocks permiten simular el comportamiento de módulos o funciones sin necesidad de tener que ejecutarlas realmente. Esto es útil, por ejemplo, para no depender de una base de datos al hacer las pruebas. MagicMock permite crear objetos simulados a los que se les puede indicar qué valor deben devolver cuando se llaman, y también permite comprobar si han sido llamados y cuántas veces.

```
def test_datos_pagina(self):  Sergio Salas *
    mock_obtener_datos = MagicMock(return_value={
        "lambda": [500, 600, 700],
        "n": [1.5, 1.6, 1.7],
        "k": [0.1, 0.2, 0.3],
        "page_id": 2,
        "page_name": "Johnson"
    })

    # Instancia del Presenter
    presenter = Presenter()

    # Reemplazar la función con el mock
    presenter.get_page_data = mock_obtener_datos

    # Llamamos a la función que usa la función mockeada
    resultados = presenter.get_values( name: "Ag",  page_name: "Johnson")

    # Verificar que se haya llamado correctamente
    mock_obtener_datos.assert_called_once_with( *args: "Ag", "Johnson")

    # Verificar que los datos se hayan almacenado correctamente en material_data
    self.assertEqual(resultados['lambda'],  second: [500, 600, 700])
    self.assertEqual(resultados['n'],  second: [1.5, 1.6, 1.7])
    self.assertEqual(resultados['k'],  second: [0.1, 0.2, 0.3])
    self.assertEqual(resultados['page_id'],  second: 2)
    self.assertEqual(resultados['page_name'],  second: "Johnson")
```

*Figura 15 – Prueba de integración*

Como vemos en la Figura 15, esta prueba de integración tiene como objetivo verificar el correcto funcionamiento del método “get_values” de la clase Presenter. A diferencia de una prueba unitaria, esta prueba no evalúa el método de forma aislada, sino el cómo interactúa con otros componentes del sistema.

Para simular el comportamiento de la función “get_page_data”, que se trata de una función importada del módulo de persistencia y, por tanto, normalmente accedería a la base de datos para obtener el resultado, se emplea un *mock* utilizando MagicMock de la biblioteca unittest.mock.

Con esto, se verifica que:

- La función es invocada una única vez con los argumentos correctos ("Ag" y "Johnson").
- Los datos devueltos por el mock son devueltos correctamente por el método get_values.

## 7.3 Pruebas de aceptación

Las pruebas de aceptación consisten en la validación del sistema por parte del usuario final. En estas pruebas se verifica que se cumplen los requisitos funcionales y no funcionales definidos al inicio del proyecto.

En este caso, las personas encargadas de llevar a cabo estas pruebas fueron tanto la Product Owner como el director del trabajo. Durante el proceso, utilizó la aplicación como lo haría un usuario real, probando la interfaz, introduciendo distintos valores y observando los resultados.

A partir de estas pruebas, se hizo una lista con cambios y mejoras. Algunas de estas sugerencias se implementaron en la aplicación, mientras que otras quedaron pendientes como trabajos futuros, que se describirán más adelante en la memoria.

Lista de comentarios / "Tickets de mantenimiento":

- “Añadir un valor por defecto del radio, al igual que hay uno del medio. Hacer que este valor sea 50 nm.” (Implementado)
- “Que los dos valores por defecto estén confirmados de primeras, y no haya que darle al botón "confirm radius" y "confirm value" la primera vez que se usa la app.” (Trabajos futuros)
- “¿Cómo de fácil sería que, al cambiar el valor de los cuadros de texto, se "autoconfirme" el cambio?” (Trabajos futuros)
- “Cambiar el título "Panel Application" de la pestaña a "Mie Web" “(Implementado)
- “Quizá, añadir en alguna esquina una interrogación o una (i) de información, de manera que al darle salga un diálogo que muestre unas instrucciones generales de uso de la app.” (Implementado)
- “Cambiar nombre de botón "Download materials" a "Download efficiencies (q)" “(Implementado)

# 8- Trabajos futuros

Este trabajo sienta las bases de una herramienta web funcional que permite comparar la respuesta electromagnética de distintos materiales utilizando la solución de Mie. Sin embargo, dado que se trata de una primera versión y el desarrollo se ha realizado en un tiempo limitado, existe un amplio margen de mejora y expansión del proyecto. Al igual que ocurre en la gran mayoría de productos de software, las posibilidades de evolución y de incorporación de nuevas funcionalidades son prácticamente infinitas.

A continuación, se comentan algunas de las ideas que surgieron como posibilidad durante el desarrollo del proyecto y que, por cuestiones de tiempo y alcance, no llegaron a implementarse, pero podrían abordarse en futuras versiones:

- Tickets de cambio de la Product Owner: Como vimos en las pruebas de aceptación, quedaron algunas recomendaciones sin implementar como:

    - Que los dos valores por defecto estén confirmados de primeras, y no haya que pulsar en el botón de confirmación la primera vez que se use la app.
    - Que, al cambiar el valor de los cuadros de texto, se "autoconfirme" el cambio.

- Otras ideas que surgieron durante el desarrollo:
    - Desplegar el proyecto en una web para poder acceder de forma online
    - Mejora de la interfaz de usuario: Aunque la herramienta ya es funcional, se podrían introducir mejoras en la experiencia de usuario, como un diseño más intuitivo, soporte para temas claros/oscuros, y mayor personalización de la visualización de los resultados.
    - Integrar una funcionalidad que permita compartir los resultados. Al pulsar en el botón de compartir se podría generar una URL que contenga como parámetros los materiales seleccionados, el tamaño de la esfera y el índice de refracción. De esta forma, cualquier persona que acceda a dicha URL cargaría automáticamente la misma configuración en la aplicación.

# 9- Conclusiones

El principal objetivo de este trabajo fue el desarrollo de una herramienta web capaz de comparar la respuesta electromagnética de distintos materiales mediante la solución de Mie. Este objetivo se ha cumplido satisfactoriamente, ya que la aplicación resultante es completamente funcional y cumple con los requisitos planteados al inicio del proyecto.

Durante el desarrollo, se presentaron ciertas dificultades, especialmente en la implementación inicial de la arquitectura en tres capas. Sin embargo, una vez comprendido su funcionamiento, esta elección resultó ser muy beneficiosa, ya que permitió una mejor organización del código y favoreció su escalabilidad.

Pese a que no contaba con experiencia previa con este lenguaje de programación, considero que la elección de Python para desarrollar el proyecto fue especialmente acertada debido a la disponibilidad de librerías que ofrece, las cuales facilitaron considerablemente el trabajo. En particular, el uso de la librería Panel permitió construir una interfaz web interactiva de forma sencilla, mientras que miepython simplificó notablemente la realización de los cálculos asociados a la solución de Mie.

A lo largo del proyecto, he adquirido experiencia en un nuevo lenguaje de programación como es Python y he reforzado mis conocimientos sobre la arquitectura en tres capas que aprendimos durante la carrera. Además, he puesto en práctica lo aprendido sobre bases de datos. Todo este proceso de aprendizaje ha sido muy enriquecedor y, sin duda, me resultará de gran utilidad en el futuro.

En conclusión, considero que la elección de este trabajo fue muy acertada, ya que me ha permitido aplicar muchos de los conocimientos teóricos adquiridos a lo largo de la carrera, al mismo tiempo que he aprendido nuevas habilidades y tecnologías.

Por todo esto, me considero satisfecho con el trabajo realizado.

# 10- Referencias


[1] Mie, G. Beiträge Zur Optik Trüber Medien, Speziell Kolloidaler Metallösungen. Ann. Phys. 1908, 330 (3), 377–445. https://doi.org/10.1002/andp.19083300302.

[2] Panel: https://panel.holoviz.org/

[3] Bokeh: https://bokeh.org/

[4] MiePython: https://miepython.readthedocs.io/en/latest/

[5] RefractiveIndex.info: https://refractiveindex.info/

[6] Repositorio del proyecto

https://github.com/sergiiosalas/tfg

[7] RefractiveIndex.info-sqlite

https://github.com/HugoGuillen/refractiveindex.info-sqlite