3-4-2025

Xoel Rivas Pérez

CIFP A Carballeira - Marcos Valcárcel

CICLO SUPERIOR DE DESARROLLO DE APLICACIONES MULTIPLATAFORMA

PROYECTO DE FIN DE CICLO

APLICACIÓN DE GESTIÓN DE LECTURAS PERSONALES

Contenido

[INTRODUCCIÓN 2](#_Toc199700557)

[OBJETIVOS DEL PROYECTO 2](#_Toc199700558)

[Objetivo principal 2](#_Toc199700559)

[Objetivos técnicos 2](#_Toc199700560)

[Objetivos académicos 2](#_Toc199700561)

[ANÁLISIS Y ESPECIFICACIÓN DE REQUISITOS 2](#_Toc199700562)

[Requisitos funcionales 2](#_Toc199700563)

[Requisitos no funcionales 3](#_Toc199700564)

[Casos de uso 4](#_Toc199700565)

[GESTIÓN DE LA INFORMACIÓN Y DATOS 5](#_Toc199700566)

[PLANIFICACIÓN 8](#_Toc199700567)

[DESARROLLO 8](#_Toc199700568)

[Estructura del proyecto 8](#_Toc199700569)

[Herramientas y librerías utilizadas 9](#_Toc199700570)

[POSIBLES MEJORAS 9](#_Toc199700571)

# INTRODUCCIÓN

Este proyecto consiste en el desarrollo de una aplicación de escritorio, diseñada para amantes de la lectura que deseen llevar un registro organizado de sus libros leídos. La herramienta permitirá no solo almacenar información básica (título, autor, género), sino también añadir valoraciones y reseñas personales, facilitando un seguimiento enriquecedor de sus hábitos de lectura.

El proyecto surge como requisito para aprobar el Ciclo Superior de Desarrollo de Aplicaciones Multiplataforma, por lo que probará diversos conocimientos de programación, gestión de bases de datos, diseño de interfaces gráficas y demás habilidades relacionadas con el desarrollo de aplicaciones.

# OBJETIVOS DEL PROYECTO

## Objetivo principal

Desarrollar una aplicación funcional e intuitiva que permita:

* Registrar los libros leídos por el usuario.
* Autocompletar determinados campos (vía API) como título, autor, género, año de publicación, etc.
* Rellenar determinados campos con información personal como la reseña o la puntuación del libro.
* Visualizar el historial de lectura con filtros avanzados (por año, género, autor, puntuación…)

## Objetivos técnicos

* Implementar una base de datos SQLite para almacenar libros.
* Garantizar la integridad de los datos con relaciones y consultas SQL optimizadas.
* Crear una interfaz gráfica vistosa con CustomTkinter.
* Conectar con una API para autocompletar datos de los libros.

## Objetivos académicos

Demostrar la competencia en el desarrollo de aplicaciones y aplicar metodologías de planificación y documentación propias del ciclo.

# ANÁLISIS Y ESPECIFICACIÓN DE REQUISITOS

## Requisitos funcionales

* Gestión de libros:
  + **RF-01**: Búsqueda de libros en una API externa.

El sistema permitirá buscar libros por título/autor en Open Library y mostrará resultados en un formato de lista.

* + **RF-02**: CRUD de libros en la biblioteca personal.

Los libros se añaden desde la API o manualmente (con campos: título, autor, género, etc.). También se pueden editar o eliminar libros existentes.

* + **RF-03**: Gestión de estados de lectura.

Marcado de libros con estados: “Leído”, “Leyendo”, “Pendiente”, “Abandonado”.

* + **RF-04**: Visualización de detalles.

Mostrar información extendida: sinopsis, editorial, año de publicación, etc.

* Interfaz de Usuario:
  + **RF-05**: Pantalla principal.

Aparece el listado de los libros en formato tarjeta (filtrable y ordenable). Cada tarjeta tendrá un indicador visual que mostrará el estado de la lectura del libro.

* + **RF-06**: Formularios interactivos.

Validación de campos obligatorios (título, autor) y autocompletado vía API.

* + **RF-07**: Sistema de filtrado.

Filtros combinables por: género, autor, año, estado de lectura, puntuación personal.

* Base de datos:
  + **RF-08**: Persistencia de datos en SQLite.

Almacenamiento local de la biblioteca personal.

## Requisitos no funcionales

* Usabilidad:
  + **RNF-01**: Interfaz gráfica sencilla y fácil de navegar.

Menú de navegación intuitivo con diseño responsivo.

* Compatibilidad:
  + **RNF-02**: App compatible con Windows, macOS y Linux.
* Eficiencia:
  + **RNF-03**: La base de datos debe manejar cientos de libros sin problemas.
* Seguridad:
  + **RNF-04**: Integridad de los datos.

Validación de campos para evitar errores y restricción de unicidad en campos críticos.

## Casos de uso



Tabla : Caso de uso 1: Añadir un libro.



Tabla : Caso de uso 2: Editar un libro.



Tabla : Caso de uso 3: Eliminar un libro.



Tabla : Caso de uso 4: Buscar libros.

# GESTIÓN DE LA INFORMACIÓN Y DATOS

La base de datos de la app estará formada por cuatro tablas: libros, autores, géneros y editoriales. Las tablas “autores”, “géneros” y “editoriales” están relacionadas con la tabla principal “libros”, de esta manera se evita la repetición de datos.

A continuación, se muestra el diagrama entidad-relación resultante:

Diagrama

El contenido generado por IA puede ser incorrecto.

Ilustración : Diagrama entidad-relación.

Y a partir de este diagrama se crearon las siguientes tablas, que muestran las restricciones de cada campo y el resultado de las relaciones entre entidades:

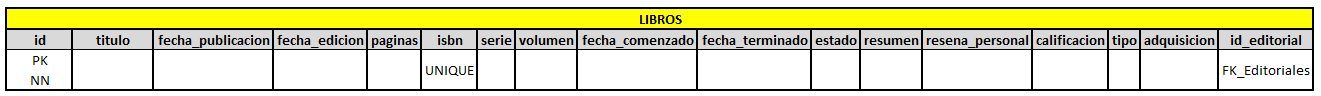


Tabla : Tabla "Libros".

* Contiene los datos principales de cada libro registrado.

Tabla

El contenido generado por IA puede ser incorrecto.

Tabla : Tabla "Autores".

* Almacena los nombres y apellidos de los autores.

Tabla

El contenido generado por IA puede ser incorrecto.

Tabla : Tabla "Géneros".

* Incluye los distintos géneros literarios disponibles.

Tabla

El contenido generado por IA puede ser incorrecto.

Tabla : Tabla "Editoriales".

* Lista las editoriales asociadas a los libros.

Tabla

El contenido generado por IA puede ser incorrecto.

Tabla : Tabla "Libros-Autores".

* Representa la relación muchos a muchos entre libros y autores.

Tabla

El contenido generado por IA puede ser incorrecto.

Tabla : Tabla "Libros-Géneros".

* Representa la relación muchos a muchos entre libros y géneros.

Tabla

El contenido generado por IA puede ser incorrecto.

Tabla : Leyenda.

Como se puede observar, al existir dos relaciones N:M, se crearon dos tablas adicionales para mostrar la relación entre las entidades.

Este diseño relacional permite mantener la integridad de los datos, evitar la redundancia y facilita la escalabilidad del sistema ante futuras ampliaciones.

# PLANIFICACIÓN



Tabla : Planificación del proyecto.

# DESARROLLO

## Estructura del proyecto

La aplicación está organizada en distintos archivos y carpetas que separan las responsabilidades principales del sistema. Esta modularidad facilita el mantenimiento y la comprensión del código. Su estructura es la siguiente:

* mi\_biblio\_app/ (carpeta raíz de la app)
  + ventana\_principal.py: contiene la clase principal que lanza la interfaz principal de la aplicación.
  + ventana\_anhadir\_libro.py: define la ventana donde se realiza la búsqueda y selección de libros para añadir a la biblioteca.
  + ventana\_editar\_libro.py: ventana en la que aparecen los diferentes campos editables de cada libro que se ha añadido a la biblioteca.
  + api.py: módulo encargado de gestionar la conexión con la API de OpenLibrary y recuperar los datos de libros.
  + database.py: define y crea la estructura de la base de datos SQLite, con sus tablas y relaciones.
  + imágenes/: carpeta donde se almacenan los iconos utilizados en la interfaz gráfica.

## Herramientas y librerías utilizadas

Para el desarrollo de la app se han utilizado las siguientes herramientas y librerías:

* **Python 3.9.13**: Lenguaje principal del desarrollo.
* **Visual Studio Code**: Editor de código.
* **Tkinter** (incluido en Python): librería estándar de Python para la creación de interfaces gráficas.
* **Tkcalendar 1.6.1**: complemento que permite incorporar calendarios funcionales en las interfaces.
* **CustomTkinter 5.2.2**: framework moderno basado en Tkinter que proporciona una interfaz gráfica más estilizada.
* **Pillow (PIL) 11.0.0**: utilizada para cargar y mostrar imágenes (iconos) dentro de los diferentes widgets de la interfaz.
* **Urllib.request** (incluido en Python): permite realizar solicitudes HTTP para descargar imágenes desde internet.
* **Requests 2.32.3**: permite realizar peticiones HTTP a la API de OpenLibrary para recuperar datos de libros.
* **Re** (incluido en Python): Módulo de expresiones regulares, utilizado para validar o procesar texto.
* **Io.BytesIO** (incluido en Python): facilita la lectura de datos binarios en memoria.
* **Threading** (incluido en Python): se utiliza para ejecutar tareas en segundo plano.
* **SQLite3** (incluido en Python): usado en database.py para crear y gestionar la base de datos local.
* **Datetime** (incluido en Python): Para manejar y formatear fecha, especialmente en combinación con tkcalendar.
* **Time** (incluido en Python): Utilizada en algunas funciones para añadir retardos o gestionar tiempos de espera.

# POSIBLES MEJORAS

* Añadir visualización de estadísticas mediante gráficos (gráficos de barras, circulares, etc.).