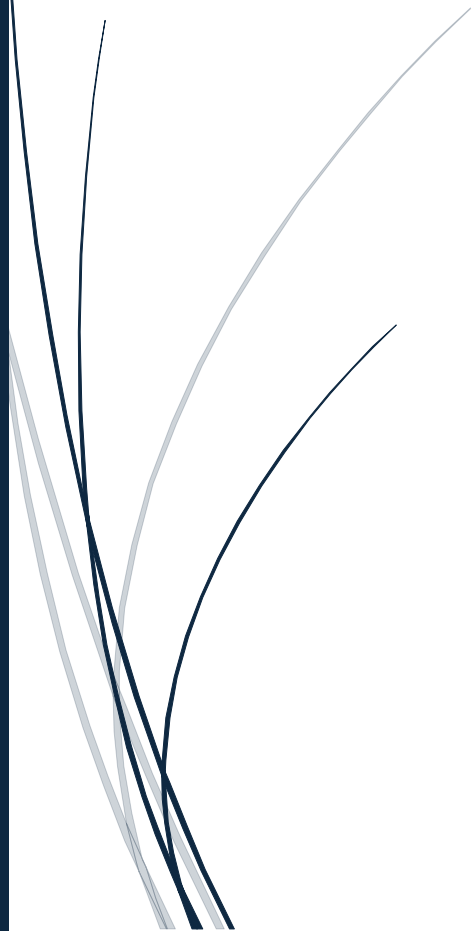
3-4-2025

PROYECTO DE FIN DE CICLO

APLICACIÓN DE GESTIÓN DE LECTURAS PERSONALES



Xoel Rivas Pérez
CIFP A CARBALLEIRA - MARCOS VALCÁRCEL
CICLO SUPERIOR DE DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Contenido

INTRODUCCIÓN 2
OBJETIVOS DEL PROYECTO 2
Objetivo principal 2
Objetivos técnicos 2
Objetivos académicos 2
ANÁLISIS Y ESPECIFICACIÓN DE REQUISITOS 2
Requisitos funcionales 2
Requisitos no funcionales 3
Casos de uso 4
GESTIÓN DE LA INFORMACIÓN Y DATOS 5
PLANIFICACIÓN 8
DESARROLLO 8
Estructura del proyecto 8
Herramientas y librerías utilizadas 9
POSIBLES MEJORAS 9

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

CASO DE USO 1: AÑADIR UN LIBRO	
Campo	**Descripción**
Nombre	Añadir un libro.
Actor	Usuario.
Descripción	El usuario pulsa el botón "+" para añadir un libro. Introduce el título, autor o ISBN en el buscador y se muestran las posibles coincidencias. Al seleccionar un libro, este se añade a su biblioteca personal con los datos recuperados automáticamente desde la API (si están disponibles).
Precondición	El usuario accede a la opción de "Añadir libro" desde la interfaz principal.
Postcondición	El libro queda registrado en la biblioteca del usuario. A nivel interno, si el autor, género o editorial no existiera previamente en la base de datos, se crean automáticamente para mantener las relaciones correctas.

Tabla 1: Caso de uso 1: Añadir un libro.

CASO DE USO 2: EDITAR UN LIBRO	
Campo	**Descripción**
Nombre	Editar un libro.
Actor	Usuario.
Descripción	El usuario accede a los detalles de un libro previamente añadido y modifica uno o varios campos: fechas de lectura, reseña personal, calificación, estado, etc.
Precondición	El libro ya debe estar registrado en la biblioteca del usuario.
Postcondición	Los cambios realizados por el usuario se guardan correctamente en la base de datos, actualizando el registro del libro.

Tabla 2: Caso de uso 2: Editar un libro.

CASO DE USO 3: ELIMINAR UN LIBRO	
Campo	**Descripción**
Nombre	Eliminar un libro.
Actor	Usuario.
Descripción	El usuario selecciona un libro de su biblioteca personal y lo elimina permanentemente.
Precondición	El libro ya debe estar registrado en la biblioteca del usuario.
Postcondición	El libro se elimina de la base de datos. Las relaciones con autores, géneros o editorial también se eliminan si ya no están vinculadas a otros libros.

Tabla 3: Caso de uso 3: Eliminar un libro.

CASO DE USO 4: BUSCAR LIBROS	
Campo	**Descripción**
Nombre	Buscar libros.
Actor	Usuario.
Descripción	Desde la vista principal (vista en la que se muestran los libros añadidos por el usuario a su biblioteca), el usuario introduce un término de búsqueda (título, autor o ISBN) en el buscador para localizar libros específicos.
Precondición	Deben existir libros previamente añadidos por el usuario.
Postcondición	La vista principal se actualiza mostrando únicamente los libros que coinciden con el criterio de búsqueda.

Tabla 4: 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:

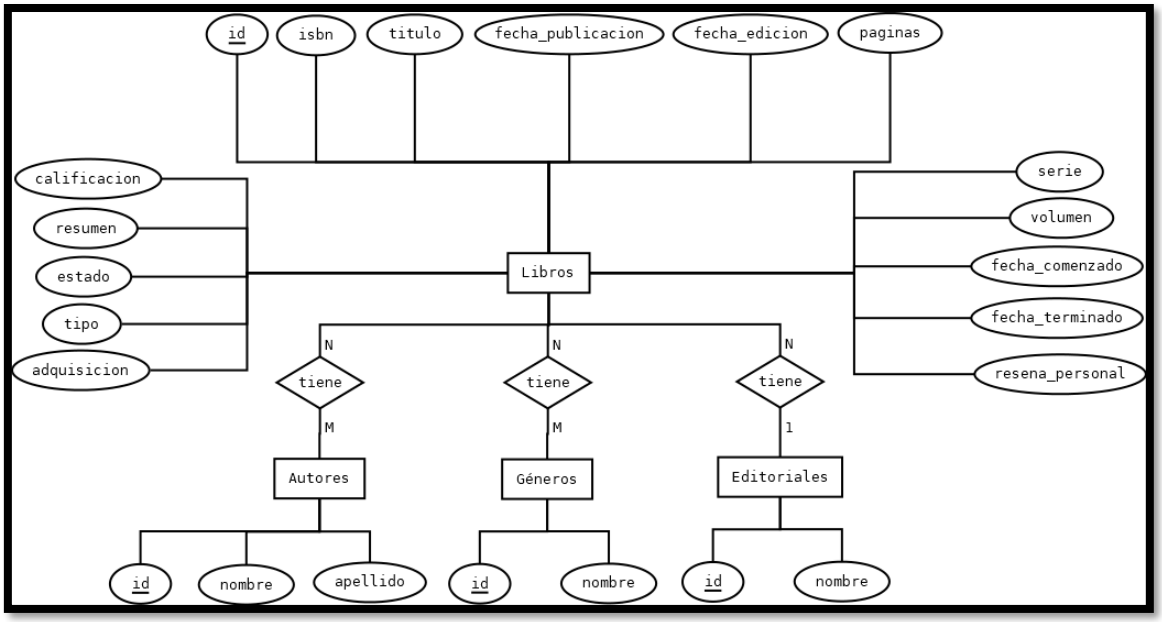


Ilustración 1: Diagrama entidad-relación.

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:

LIBROS															
id	titulo	fecha_publicacion	fecha_edicion	paginas	isbn	serie	volumen	fecha_comenzado	fecha_terminado	estado	resumen	resena_personal	calificacion	tipo	adquisicion	id_editorial
PK NN					UNIQUE											FK_Editoriales

Tabla 5: Tabla "Libros".

- Contiene los datos principales de cada libro registrado.

AUTORES		
id	nombre	apellido
PK	NN	
NN	UNIQUE	

Tabla 6: Tabla "Autores".

- Almacena los nombres y apellidos de los autores.

GÉNEROS	
id	nombre
PK	NN
NN	UNIQUE

Tabla 7: Tabla "Géneros".

- Incluye los distintos géneros literarios disponibles.

EDITORIALES	
id	**nombre**
PK	NN
NN	UNIQUE

Tabla 8: Tabla "Editoriales".

- Lista las editoriales asociadas a los libros.

LIBROS_AUTORES	
id_libro	**id_autor**
PK	
FK_Libros	FK_Autores

Tabla 9: Tabla "Libros-Autores".

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

LIBROS_GÉNEROS	
id_libro	**id_genero**
PK	
FK_Libros	FK_Géneros

Tabla 10: Tabla "Libros-Géneros".

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

LEYENDA	
PK	Primary Key
FK_NombreTabla	Foreign Key y tabla de referencia
NN	Not Null

Tabla 11: 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

Fase	Tiempo estimado	Inicio	Fin
Inicio y organización	4h	04-abr-25	05-abr-25
Redacción de Introducción y Objetivos	5h	05-abr-25	07-abr-25
Análisis y especificación de requisitos	6h	08-abr-25	10-abr-25
Diagrama ER y diseño de la base de datos	6h	15-abr-25	20-abr-25
Creación del script de la base de datos	4h	19-abr-25	20-abr-25
Planificación del proyecto	2h	21-abr-25	21-abr-25
Diseño de casos de uso	4h	22-abr-25	24-abr-25
Diseño para la gestión de la información y datos	3h	22-abr-25	25-abr-25
Implementación de la base del sistema	8h	26-abr-25	02-may-25
Implementación del formulario de añadir libros	12h	03-may-25	10-may-25
Integración de relaciones N:M (autores y géneros)	10h	11-may-25	18-may-25
Desarrollo de la interfaz gráfica básica (CustomTkinter)	12h	11-may-25	18-may-25
Búsqueda y filtrado de libros	8h	19-may-25	25-may-25
Implementación de guardado y carga de datos	6h	26-may-25	30-may-25
Mejora de la interfaz y validaciones	8h	31-may-25	04-jun-25
Pruebas funcionales y depuración	10h	05-jun-25	08-jun-25
Finalización de funcionalidades (detalles extra)	6h	09-jun-25	10-jun-25
Redacción y finalización de la documentación	10h	11-jun-25	13-jun-25
Entrega final y revisión	4h	14-jun-25	15-jun-25

Tabla 12: 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.).