10-6-2025

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

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 (ordenados alfabéticamente). Cada tarjeta tendrá un indicador visual que mostrará el estado de la lectura del libro:
    Verde → Leído
    Naranja → Leyendo
    Morado → Pendiente
    Rojo → Abandonado
  - **RF-06**: Formularios interactivos.
    Autocompletado vía API.

- Base de datos:
  - **RF-07**: 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.

- Eficiencia:
  - **RNF-02**: La base de datos debe manejar cientos de libros sin problemas.

- Seguridad:
  - **RNF-03**: Integridad de los datos.
    Validación de campos para evitar errores y restricción de unicidad en campos críticos.

Casos de uso

CASO DE USO 1: INICIAR APLICACIÓN	
Campo	**Descripción**
Actor	Usuario.
Descripción	Al arrancar la aplicación, se crea la base de datos si no existe, y se muestra la ventana principal con el listado de libros guardados (en el caso de haberlos).
Precondición	Ninguna.
Flujo principal	Ejecución script principal → Se crea o se abre la base de datos → Se instancian los componentes de la ventana principal → Se consultan las diferente tablas de la base de datos para obtener el listado → Se muestran los libros (de haberlos) o un mensaje "No hay libros guardados" (de no haberlos)
Postcondición	Se muestra la ventana principal.

Tabla 1: Caso de uso 1: Iniciar aplicación.

CASO DE USO 2: VER LISTADO DE LIBROS	
Campo	**Descripción**
Actor	Usuario.
Descripción	Se muestran todos los libros guardados en la base de datos, con su título, autor, año, estado y portada.
Precondición	La ventana principal está iniciada y la base de datos creada.
Flujo principal	Se obtienen los libros guardados en la base de datos → Se itera sobre cada libro → Para cada libro se determina el color de fondo según su estado y se crea un frame con el título, el autor, el año y la portada → Se muestran todos los frames
Flujos alternativos	Si no hay libros se muestra: "No hay libros guardados". Si falla la carga de portada, se usa una imagen predeterminada "Sin portada".
Postcondición	Se muestra la lista en pantalla y el usuario puede interactuar.

Tabla 2: Caso de uso 2: Ver listado de libros.

CASO DE USO 3: BUSCAR LIBROS GUARDADOS	
Campo	**Descripción**
Actor	Usuario.
Descripción	Se realiza una búsqueda de los libros guardados por título o autor.
Precondición	Ventana principal abierta y listado inicial cargado.
Flujo principal	El usuario introduce texto en el campo de búsqueda y pulsa el botón de buscar → La app recoje el texto → Se ejecuta sentencia SQL para hacer búsqueda por título o autor → Si hay resultados se muestran los libros filtrados, si no se muestra: "No se encontraron libros que coincidan" → Si el buscador está vacío y se pulsa el botón de búsqueda se recarga el listado completo
Flujos alternativos	Error en la consulta: mostrar messagebox de error.
Postcondición	Vista refrescada con los libros filtrados o con todos si la búsqueda está vacía.

Tabla 3: Caso de uso 3: Buscar libros guardados.

CASO DE USO 4: AÑADIR UN LIBRO	
Campo	**Descripción**
Actor	Usuario.
Descripción	El usuario busca un libro por título, autor o ISBN, se muestran los resultados y se guarda el libro seleccionado en la base de datos.
Precondición	El usuario accede a la opción de "Añadir libro" desde la interfaz principal.
Flujo principal	El usuario pulsa el botón "+" → Se abre una ventana de elección (Crear o Añadir) → Selecciona "Añadir libro" → Se abre la ventana de añadir libro → El usuario escribe el texto de búsqueda y pulsa buscar → Se hace una llamada a la API de OpenLibrary y se extraen los datos de búsqueda correspondientes → Se muestra cada resultado en una tarjeta con título, autor, año y portada (si los datos existen) → El usuario selecciona un libro cuyos datos se insertan en la BD → Se muestra: "Libro guardado correctamente" → Se recarga el listado de la ventana principal → El usuario cierra la ventana de búsqueda o sigue buscando
Flujos alternativos	Si el libro ya existe (ISBN duplicado) se ignora. El usuario puede cancelar sin seleccionar un libro.
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 4: Caso de uso 4: Añadir un libro.

CASO DE USO 5: CREAR UN LIBRO	
Campo	**Descripción**
Nombre	Crear un libro.
Actor	Usuario.
Descripción	Permite al usuario crear un libro manualmente (todos los campos editables) cuando la búsqueda no arroja los resultados deseados.
Precondición	Ventana principal abierta.
Flujo principal	El usuario pulsa el botón "+" y elige "Crear libro" en la ventana de elección → Se abre la ventana de editar libro con todos los campos del formulario vacíos → El usuario rellena los campos que crea convenientes y puede insertar una imagen en la portada → El usuario guarda los cambios → Se muestra: "Libro creado correctamente" → Se recarga el listado principal
Flujos alternativos	Error al insertar: se muestra el error. El usuario puede pulsar "Cancelar": la ventana se cierra sin guardar nada.
Postcondición	Si guardó, aparece en el listado principal.

Tabla 5: Caso de uso 5: Crear un libro.

CASO DE USO 6: EDITAR UN LIBRO	
Campo	**Descripción**
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.
Flujo principal	El usuario selecciona un libro del listado principal → Se abre la ventana de edición → Se cargan los datos del libro → El usuario modifica uno o varios campos → El usuario guarda los cambios → Se muestra: "Libro actualizado correctamente" → Se cierra la ventan y se recarga el listado principal
Flujos alternativos	Error en la BD: se muestra el error. El usuario pulsa "Eliminar libro" → Se solicita confirmación → Se eliminan datos y relaciones → Se muestra: "Libro eliminado" → Se cierra la ventana y se recarga el listado principal. El usuario pulsa "Cancelar": descarta cambios y cierra la ventana.
Postcondición	Los cambios realizados por el usuario se guardan correctamente en la base de datos, actualizando el registro del libro.

Tabla 6: Caso de uso 6: Editar un libro.

CASO DE USO 7: ELIMINAR UN LIBRO	
Campo	**Descripción**
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.
Flujo principal	El usuario selecciona un libro del listado principal → Se abre la ventana de edición → Se cargan los datos del libro → El usuario pulsa "Eliminar libro" → Aparece diálogo de confirmación → Si confirma se eliminan los datos y relaciones del libro → Se muestra: "Libro eliminado correctamente" → Se cierra la ventana y se recarga el listado principal → Si el usuario cancela, se cierra el diálogo y no hace nada
Flujos alternativos	Error durante la eliminación: mostrar error.
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 7: Caso de uso 7: Eliminar un libro.

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:

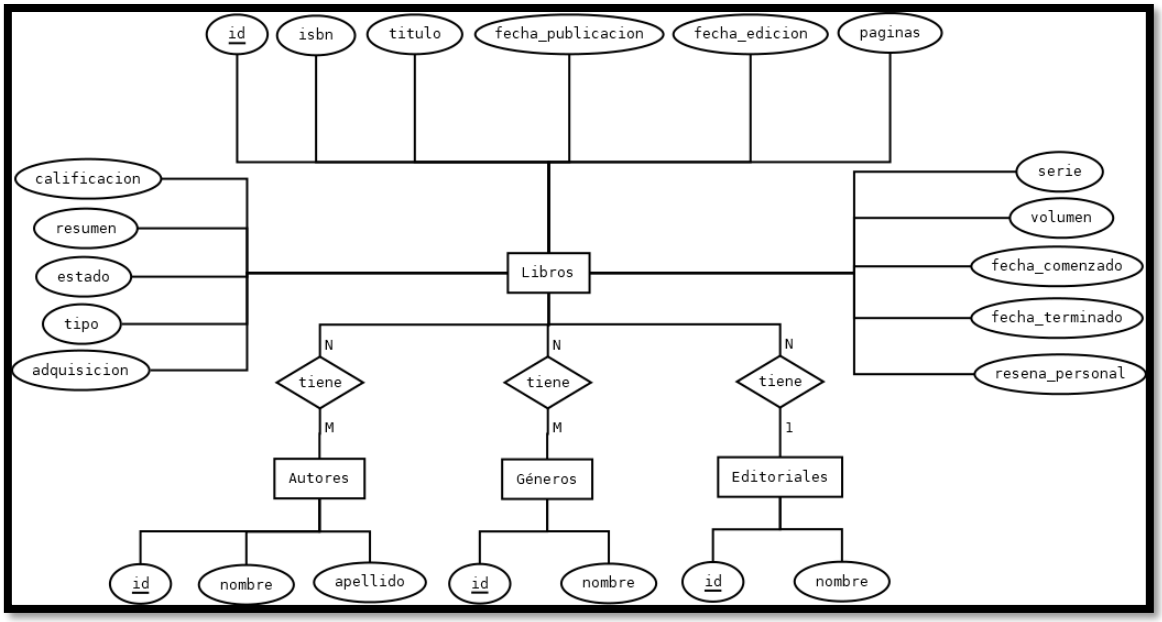

Ilustración 1: Diagrama entidad-relación.

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 8: Tabla "Libros".

- Contiene los datos principales de cada libro registrado.

AUTORES		
id	nombre	apellido
PK	NN	
NN	UNIQUE	

Tabla 9: Tabla "Autores".

- Almacena los nombres y apellidos de los autores.

GÉNEROS	
id	nombre
PK	NN
NN	UNIQUE

Tabla 10: Tabla "Géneros".

- Incluye los distintos géneros literarios disponibles.

EDITORIALES	
id	**nombre**
PK	NN
NN	UNIQUE

Tabla 11: Tabla "Editoriales".

- Lista las editoriales asociadas a los libros.

LIBROS_AUTORES	
id_libro	**id_autor**
PK	
FK_Libros	FK_Autores

Tabla 12: 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 13: 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 14: 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	0,5h	04-abr-25	05-abr-25
Redacción de Introducción y Objetivos	0,5h	05-abr-25	07-abr-25
Análisis y especificación de requisitos	1h	08-abr-25	10-abr-25
Diagrama ER y diseño de la base de datos	2h	15-abr-25	20-abr-25
Creación del script de la base de datos	1h	19-abr-25	20-abr-25
Planificación del proyecto	0,5h	21-abr-25	21-abr-25
Diseño de casos de uso	1h	22-abr-25	24-abr-25
Diseño para la gestión de la información y datos	1h	22-abr-25	25-abr-25
Implementación de la base del sistema	3h	26-abr-25	02-may-25
Implementación del formulario de añadir libros	3h	03-may-25	10-may-25
Integración de relaciones N:M (autores y géneros)	2,5h	11-may-25	18-may-25
Desarrollo de la interfaz gráfica básica (CustomTkinter)	3h	11-may-25	18-may-25
Búsqueda de libros	2h	19-may-25	25-may-25
Implementación de guardado y carga de datos	1,5h	26-may-25	30-may-25
Mejora de la interfaz y validaciones	2h	31-may-25	04-jun-25
Pruebas funcionales y depuración	2h	05-jun-25	08-jun-25
Finalización de funcionalidades (detalles extra)	1h	09-jun-25	10-jun-25
Redacción y finalización de la documentación	2h	11-jun-25	13-jun-25
Entrega final y revisión	0,5h	14-jun-25	15-jun-25
	30h		

Tabla 15: 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.
  - ventana_elegir_modo.py: ventana que sirve para preguntarle al usuario si desea crear un libro nuevo o añadir uno ya existente.
  - 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.

ESTUDIO DE MERCADO

Introducción

La gestión personal de libros es una necesidad creciente para los aficionados a la lectura. Muchos lectores buscan soluciones digitales para organizar su biblioteca personal de manera cómoda y accesible. En un contexto donde la mayoría de las aplicaciones de gestión de libros están diseñadas para móviles o

como servicios web, existe una necesidad poco cubierta de herramientas de escritorio orientadas a usuarios que prefieren una solución local, privada y sin dependencia de servicios en la nube. Este estudio analiza el mercado el mercado actual de apps de gestión de libros, identifica los principales competidores y detecta oportunidades de mejora que justifican el desarrollo de esta aplicación.

Análisis de la competencia

Existen diversas aplicaciones en el mercado que ofrecen funcionalidades similares. Algunas de las más conocidas son:

- Goodreads: Popular por su red social para lectores. Permite registrar libros, valorarlos y recibir recomendaciones. Sin embargo, depende mucho de conexión online y no está completamente orientada a una gestión privada de libros.
- LibraryThing: Más enfocada a coleccionistas serios y bibliotecarios. Ofrece herramientas potentes para catalogar libros, pero su interfaz es poco intuitiva y puede resultar abrumadora para el usuario medio.
- Calibre: Es una de las herramientas más completas para la gestión de ebooks. Permite convertir formatos, gestionar metadatos, sincronizar con dispositivos electrónicos y más. No obstante, está más enfocada en libros electrónicos y carece de una experiencia de usuario sencilla o atractiva para quienes solo desean registrar sus lecturas físicas.
- Libib: Aplicación web que permite catalogar libros, películas, videojuegos y música. Es muy visual y fácil de usar, pero también depende de conexión online y cuenta con funciones limitadas en su versión gratuita.

Oportunidades de mejora

Según la competencia nombrada anteriormente, existen varias áreas en las que existe una oportunidad para desarrollar una herramienta diferente:

- Privacidad y uso local: La mayoría de las apps existentes requieren conexión a internet o guardan los datos en la nube. Esta app se orienta a usuarios que valoran la privacidad y prefieren guardar sus datos localmente.
- Flexibilidad en la edición: Mientras que muchas apps imponen valores fijos en los diferentes campos de los libros, *MiBiblio* permite al usuario modificar los valores de todos ellos a su gusto, incluyendo el título, autor, portada, etc.
- Simplicidad y enfoque: Se busca una aplicación liviana, sin funcionalidades innecesarias, pensada para el uso rápido y directo, ideal

para lectores que solo quieren llevar un registro simple y personal de sus libros leídos.

- Cración manual: Permite crear libros desde cero en caso de que no se encuentren resultados mediante la API, lo cual da control total al usuario.

Conclusión

Este proyecto está enfocado a lectores que quieren gestionar su biblioteca personal de forma local, flexible, sencilla y sin depender de servicios externos. Frente a la complejidad de algunas soluciones actuales o a la orientación social de otras, esta aplicación destaca por su enfoque minimalista y privado.

POSIBLES MEJORAS

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

ANEXO

Presupuesto

Costes de desarrollo			
Tarea	Horas	Tarifa/h	Subtotal (€)
Inicio y organización	0,50	20 €	10,00 €
Redacción de Introducción y Objetivos	0,5	20 €	10,00 €
Análisis y especificación de requisitos	1	20 €	20,00 €
Diagrama ER y diseño de la base de datos	2	20 €	40,00 €
Creación del script de la base de datos	1	20 €	20,00 €
Planificación del proyecto	0,5	20 €	10,00 €
Diseño de casos de uso	1	20 €	20,00 €
Diseño para la gestión de la información y datos	1	20 €	20,00 €
Implementación de la base del sistema	3	20 €	60,00 €
Implementación del formulario de añadir libros	3	20 €	60,00 €
Integración de relaciones N:M (autores y géneros)	2,5	20 €	50,00 €
Desarrollo de la interfaz gráfica básica (CustomTkinter)	3	20 €	60,00 €
Búsqueda de libros	2	20 €	40,00 €
Implementación de guardado y carga de datos	1,5	20 €	30,00 €
Mejora de la interfaz y validaciones	2	20 €	40,00 €
Pruebas funcionales y depuración	2	20 €	40,00 €
Finalización de funcionalidades (detalles extra)	1	20 €	20,00 €
Redacción y finalización de la documentación	2	20 €	40,00 €
Entrega final y revisión	0,5	20 €	10,00 €
Total	**30,00**	**20 €**	**600,00 €**

Tabla 16: Costes de desarrollo.

Costes de software y herramientas		
Herramienta	Licencia/Coste	Subtotal (€)
Visual Studio Code	Gratuito	0 €
Python	Gratuito	0 €
SQLite	Gratuito	0 €
OpenLibrary (API)	Gratuito	0 €
Total		0 €

Tabla 17: Costes de software y herramientas.

Costes de hardware	
Recurso	Subtotal (€)
Equipo de trabajo (ordenador, monitores, periféricos, etc.)	800 €
Total	800 €

Tabla 18: Costes de hardware.

Costes de electricidad/internet			
Recurso	Meses	Tarifa/mes	Subtotal (€)
Electricidad	3	62,00 €	186 €
Internet	3	65,00 €	195 €
Total			381 €

Tabla 19: Costes de electricidad/internet.

Total presupuesto	
Concepto	Subtotal
Desarrollo	600 €
Sofware y herramientas	0 €
Hardware	800 €
Electricidad/internet	381 €
Total estimado	1.781 €

Tabla 20: Total presupuesto.

Webgrafía

- Documentación Tkinter
- Documentación Customtkinter
- Documentación Tkcalendar
- Documentación Pillow
- Documentación Urllib request

- Documentación Requests
- Documentación Re
- Documentación Io
- Documentación Threading
- Documentación Sqlite3
- Documentación Datetime
- Documentación Time