# PRÁCTICA 5

## APLICACIÓN PARA GESTIÓN DE UNA LIBRERÍA

Equipo:
Beltrán Saucedo Axel Alejandro
Cerón Samperio Lizeth Montserrat
Higuera Pineda Angel Abraham
Lorenzo Silva Abad Rey
4BV1.

ESCUELA SUPERIOR DE CÓMPUTO

**TECNOLOGÍAS PARA EL DESARROLLO DE APLICACIONES WEB**

31/10/2025

# Índice

# 1. Introducción

## Planteamiento del problema

Se busca crear una API web cuyo propósito sea gestionar un inventario y optimizar envíos con ayuda del algoritmo genético simple. Para la gestión del inventario, se debe garantizar lo siguiente:

- Permitir crear, leer, actualizar y borrar items, categorías y envíos en una base de datos.

Para la optimización de envíos, se debe garantizar lo siguiente:

- Ofrecer un endpoint que utilice el AGS para resolver el problema de la mochila. Que calcule la combinación de items que maximiza la ganancia total de un envío sin exceder una capacidad de peso determinada.

## Propuesta de solución

La solución propuesta es el desarrollo de una API con FastAPI para la lógica de negocio y SQLModel para la gestión de la base de datos. Componentes clave de la solución:

- **SQLModel:** Para definir la estructura de la base de datos estableciendo categoría, item y envío como entidades principales.
- **endpoints (URLs):** Para crear, leer, actualizar y borrar items, categorías y envíos.
- **Algoritmo Génetico:** Realizará el calculo. Simulando un proceso de evolución para encontrar la mejor combinación de items que da la mayor ganancia total sin superar la capacidad.

# 2. Fundamentos Teóricos

En el desarrollo de esta práctica, se pondrán en uso conceptos y teconologías ya vistas; refinandolos de manera que sea entendible para el cliente, esto en forma de un inventario de biblioteca.

## Entorno de Trabajo y Herramientas Principales

- **Python:** Es el lenguaje de programación sobre el que se construye toda la lógica de la aplicación. Su sencilla sintaxis y su amplia cantidad de librerías lo hacen ideal para el desarrollo web.[2]
- **FastAPI:** Framework web moderno para construir APIs con Python. Sus características más destacadas son la rapidez, la validación de datos automática mediante Pydantic y la generación de documentación interactiva , que fue crucial para probar los endpoints de nuestra API.
- **Uvicorn:** Es un servidor ASGI ultrarrápido, utilizado para ejecutar la aplicación FastAPI. Permite que la API maneje múltiples peticiones de forma asíncrona, mejorando el rendimiento.[3]

## Persistencia de Datos

La persistencia de datos es la capacidad de un sistema para poder conservar la información posterior de la duración de una sola ejecución. En nuestra practica pasada, los datos eran volátiles, ya que se almacenaban en una lista en memoria. En esta práctica, se implementa la persistencia a través de los siguientes componentes:

- **SQLite:** Es un motor de base de datos relacional, autocontenido y que no requiere un servidor. Almacena toda la base de datos en un único archivo (en nuestro caso, `database.db`). Es ideal para desarrollo y aplicaciones de pequeña a mediana escala por su simplicidad y portabilidad.
- **SQLAlchemy:** Es una librería que funciona como un Mapeador Objeto-Relacional (ORM). Un ORM es una técnica que actúa como un "traductor.^entre el código orientado a objetos y las tablas de una base de datos relacional. Permite manipular la base de datos utilizando código Python en lugar de escribir consultas SQL directamente.
- **SQLModel:** Es una librería que combina **SQLAlchemy** y **Pydantic**, creada por el mismo autor de FastAPI. Permite definir la estructura de los datos, las validaciones y el esquema de la base de datos en una sola clase, reduciendo la duplicación de código y simplificando el desarrollo. En nuestra práctica, las clases como `Item` son modelos de SQLModel que representan tanto la tabla en la base de datos como los datos que la API recibe y envía.[4]

## Relaciones en Bases de Datos

- **Relación Unívoca:** Cada valor de clave primaria se relaciona con sólo un registro en la tabla relacionada.
- **Uno a varios:** La tabla de claves primaria sólo contiene un registro que se relaciona con ninguno, uno o varios registros en la tabla relacionada.
- **Varios a varios:** Cada registro en ambas tablas puede estar relacionado con varios registros en la otra tabla. Este tipo de relaciones requieren una tercera tabla, denominada tabla de enlace o asociación, porque los sistemas relacionales no pueden alojar directamente la relación. [5]

# API CRUD

Una API CRUD es una interfaz de programación que permite Crear, Leer, Actualizar y Borrar datos. [6]

## Búsqueda por parámetros.

La búsqueda por parámetros consiste en consultar datos específicos en la base de datos utilizando ciertos "filtros.º çriterios". En la API se implementará mediante los endpoints:

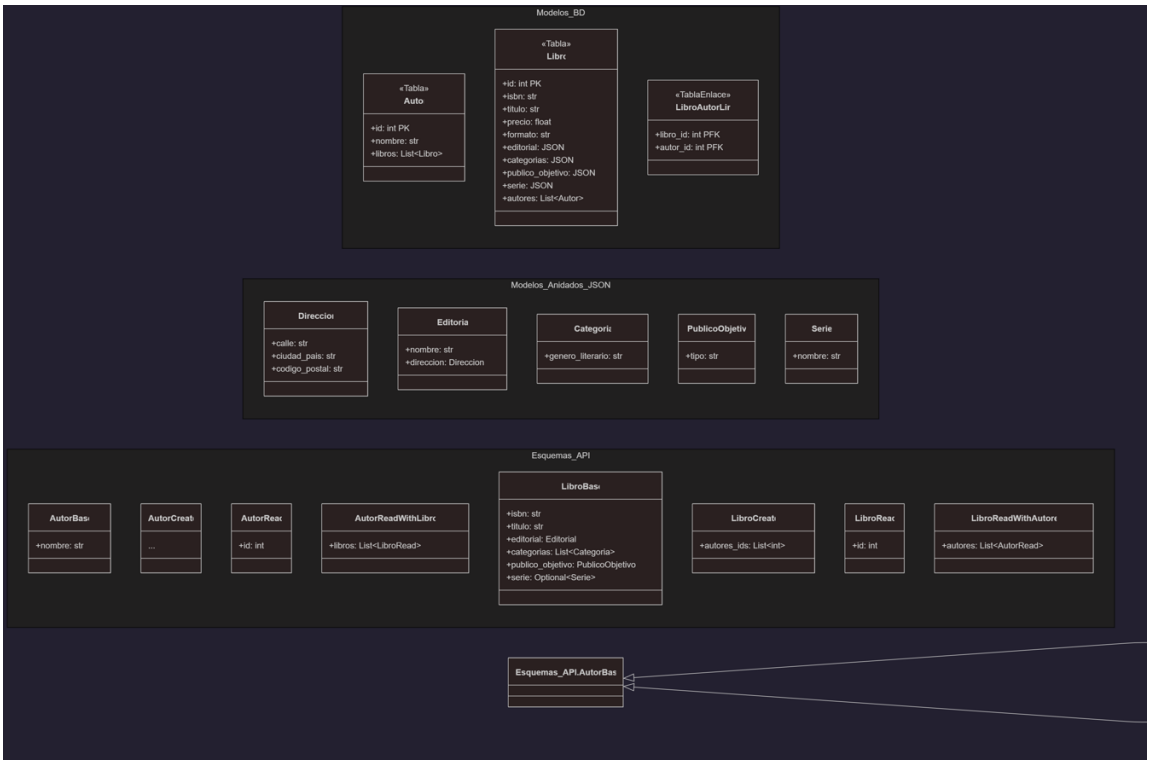
- **Libros del mismo autor.** Que permitirá obtener todos los libros escritos por un mismo autor.
- **Libros por categoría.** Que nos permitirá obtener todos los libros que pertenecen a una categoría específica.
- **Libros por serie.** Que nos permitirá obtener todos los libros pertenecientes a una serie.
- **Libros por público objetivo.** Que nos permitirá obtener los libros según el público al que van dirigidos.

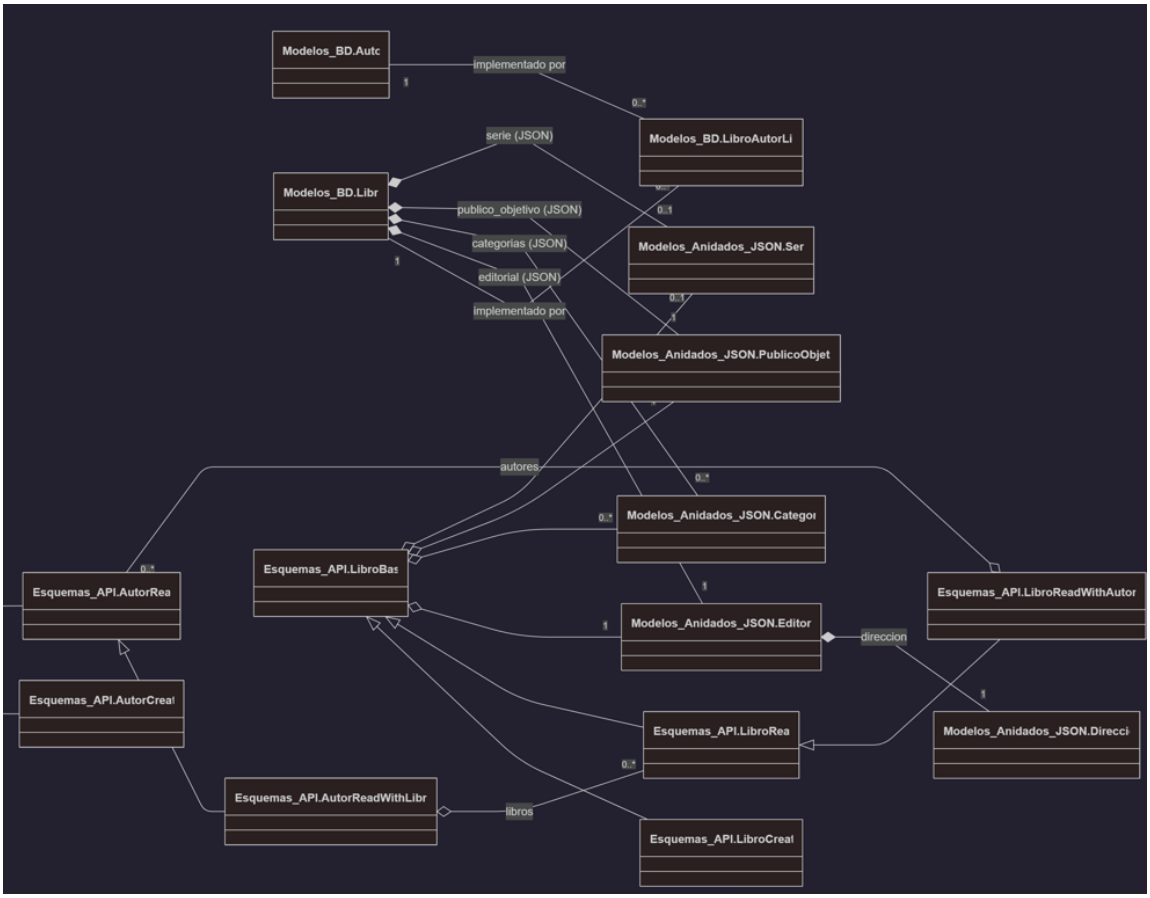
## Digitalización de un catálogo.

Se trata de un proceso para la conversión de un catálogo de libros físicos a una búsqueda por consultas en un formato digital. Tomando en cuenta los siguientes aspectos de cada libro:

- Titulo
- ISBN
- Autor(es)
- Editorial
- Año de publicación
- Páginas
- Categorías
- Precio
- Formato (físico o digital)
- Público objetivo (Mayores de edad o menores de edad)
- Serie (si pertenece a una serie de libros)

# 3. Diagrama UML






# 4. Implementación

## Código:

En esta sección, explicaremos las modificaciones y adiciones clave que transforman la API. El cambio fundamental es la introducción de **relaciones** en la base de datos, pasando de un solo modelo 'Item' a un sistema de tres modelos interconectados: Item, Categoria, y Envio.

### `Servicios/database.py`: Gestión de la Base de Datos

Este módulo es fundamental para la persistencia de datos.

- **Configuración**: Establece la conexión con la base de datos. Se define la `DATABASE_URL` para utilizar un archivo local de **SQLite** (`libreria.db`).

- **Motor (`engine`)**: Se crea el `engine` (motor) de SQLModel, que sirve como el punto de acceso centralizado y el gestor de la comunicación entre la aplicación y la base de datos.
- **Gestión de Sesiones (`get_session`)**: Expone la función `get_session` como una **dependencia** de FastAPI. Su responsabilidad es ïnyectarüna sesión de base de datos activa en cada *endpoint* que la requiera.

### `Esquemas/esquemas.py`: Definición de Modelos de Datos

Este archivo define la estructura y las reglas de validación para todos los datos que maneja la API, utilizando las clases de `SQLModel`.

- **`...Base` (Ej. `CategoriaBase`)**: Define los atributos comunes y los tipos de datos de una entidad.
- **`...Crear` (Ej. `AutorCreacion`)**: Esquemas utilizados para validar los datos JSON que **ingresan** a la API durante las operaciones de creación (`POST`).
- **`...Leer` (Ej. `AutorLeer`)**: Esquemas que definen el formato de los datos que la API **devuelve** como respuesta. Comúnmente añaden el `id` generado por la BD.
- **`...Actualizar` (Ej. `EditorialActualizar`)**: Esquemas específicos para operaciones `PATCH`, donde todos sus campos son `Optional` para permitir actualizaciones parciales.

### `Rutas/*.py`: Controladores de Endpoints de la API

Estos archivos definen los *endpoints* (URLs) públicos de la aplicación utilizando `APIRouter` de FastAPI para organizar las rutas de manera modular.

- **`Rutas/autores.py`**: Establece el prefijo `/Autores` e implementa las operaciones `POST` (crear), `GET` (leer) y `DELETE` (eliminar) para los autores.
- **`Rutas/editoriales.py`**: Gestiona el CRUD completo para `/Editoriales`, incluyendo la ruta `PATCH` para actualizaciones parciales y la lógica `DELETE` para borrar la editorial y su dirección.
- **`Rutas/publico_objetivo.py` y `Rutas/series.py`**: Exponen las rutas `POST` y `GET` para crear y leer, respectivamente, las entidades de público objetivo y series.

# Referencias


[1] IBM. (s.f.). ¿Qué es una API REST? Recuperado el 14 de octubre de 2025, de https://www.ibm.com/mx-es/topics/rest-apis

[2] Python Software Foundation. (s.f.). Acerca de Python™. Resumen Ejecutivo. Recuperado el 14 de octubre de 2025, de https://www.python.org/doc/essays/blurb-es/

[3] Ramírez, S. (s.f.). FastAPI. Recuperado el 14 de octubre de 2025, de https://fastapi.tiangolo.com/es/

[4] Ramírez, S. (s.f.). SQLModel. Recuperado el 14 de octubre de 2025, de https://sqlmodel.tiangolo.com/es/

[5] IBM Control Desk. (s. f.). Relaciones en Bases de Datos. Recuperado el 31 de octubre de 2025, de https://www.ibm.com/docs/es/control-desk/7.6.1?topic=structure-database-relationships

[6] Jain, A. (2024, 8 septiembre). What is CRUD API? DEV Community. Recuperado el 31 de octubre de 2025, de https://dev.to/ankitjaininfo/what-is-crud-api-502i

[7] Tema 2. Algoritmos genéticos. (s. f.). Departamento de Ciencias de la Computación E Inteligencia Artificial. Recuperado el 31 de octubre de 2025, de http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t2geneticos