🐍 PROYECTO FINAL PYTHON 🐍

📚 App Web CRUD para gestión de BBDD Mongo 📚

Tabla de contenidos

En esta práctica, trabajaremos el **stack Flask-MongoDB** y aprenderemos a orquestar todo el ecosistema mediante **Docker**.

NOTA: el siguiente enunciado se basa en un supuesto de gestión de inventario de una tienda de Comics, pero solo es para explicar los pasos necesarios. El alumnado deberá crear aplicaciones basadas en sus propias ideas.

1. Infraestructura de Microservicios (docker-compose.yml)

Antes de picar código Python, debéis preparar vuestro entorno. Cread un archivo docker-compose.yml que levante:

- **Servicio db**: Imagen oficial de mongo:latest en el puerto 27017.
- **Servicio admin**: Imagen de mongo-express para visualizar vuestros documentos desde el navegador (puerto 8081).

2. Requisitos del Desarrollo

A. Backend y Datos (app.py y forms.py)

- **Conexión**: Usad pymongo para conectar Flask con el contenedor de Mongo. Recordad que el *host* en la cadena de conexión será el nombre del servicio en Docker (mongodb://db:27017).
- **Formularios con WTForms**: Nada de input simples en HTML. Definid una clase ComicForm en **forms.py** que valide:
  - Título (Obligatorio).
  - Autor (Obligatorio).
  - Precio (Número decimal positivo).
  - Stock (Número entero).
  - Categoría (Select con opciones: Manga, Marvel, DC, Europeo).

B. Interfaz de Usuario (Motor Jinja2)

Usa la herencia de plantillas para no repetir código. La estructura en **templates/** será:

- **base.html**: Contendrá el Navbar (con el logo de la tienda) y el bloque de contenido.
- **inicio.html**: Un dashboard de bienvenida con estadísticas rápidas (ej. "Tienes 50 cómics en catálogo").
- **read.html**: Lista detallada de todos los cómics.
- **add.html**: Formulario de alta.
- **update.html**: Edición de un cómic existente.
- **remove.html**: Ventana de confirmación para eliminar un registro.

3. Reto puntos Extra:

Añadir un sistema de **Autenticación**. Para ello:

1. Cread plantillas de **registro.html** y **login.html**.
2. Implementad la lógica en Flask para que solo los usuarios registrados puedan acceder a las funciones de creación, edición y borrado (CRUD). Los visitantes anónimos solo podrán ver el catálogo en **read.html**.

4. Despliegue con Docker

Vuestra aplicación no debe ejecutarse solo en "vuestra máquina".

- Cread un **Dockerfile** basado en **python:3.9-slim**.
- Añadid un archivo **requirements.txt** con las dependencias (flask, pymongo, flask-wtf, etc.).
- Actualizad el **docker-compose.yml** para que construya e incluya el servicio web (vuestra app).

5. Criterios de Evaluación

Criterio	Peso
Funcionamiento CRUD completo Mongo	**35%**
Uso correcto de WTForms y validaciones	**20%**
Diseño HTML y herencia de plantillas Jinja2	**20%**
Configuración de Docker y docker-compose	**10%**
Creación de sistema de login y registro	**15%**