Segundo parcial de Análisis y metodologías de sistemas

Materia: Análisis y metodologías de sistemas

Profesor: Florencio Elías Contrera

Alumno: Tobias Bruckner

Comisión: ACM4AP

Fecha de entrega: 20 de noviembre de 2025

Proyecto seleccionado: Librería Yenny

Da Vinci
PRIMERA ESCUELA DE ARTE MULTIMEDIAL

1. Modelo ambiental

Para mí proyecto decidí elegir una librería donde el usuario pueda loguearse navegar por la web y comprar libros. Para realizar esto usé el lenguaje de programación llamado Python. Una vez elegido el lenguaje implementé un microframework de Python llamado Flask que nos ayuda a desarrollar páginas web de forma rápida y sencilla. Para organizarme mejor utilicé modularización en las carpetas de mi proyecto, haciendo que cada módulo cumpla su función determinada y quede todo más ordenado. Para trabajar con base de datos en mi proyecto usé sqlite3, es un sistema de gestión de bases de datos de tipo relacional ligero y sin servidor, y para hashear las contraseñas utilicé CRYPT. Para separar la lógica de la presentación use un motor de plantillas HTML llamado Jinja2. Para la autenticación a la hora que el usuario inicie sesión en nuestra página use una API de Google, permitiendo que el usuario pueda iniciar sesión con su cuenta de Google, haciendo más fácil y rápido el login. Por último, cree un repositorio en GitHub para subir mi proyecto y usé ese mismo repositorio para desplegar mi proyecto en un hosting llamado Render.

2. Estructura del proyecto

Carpeta .venv: contiene un entorno virtual, que es una instalación de Python completamente aislada para un proyecto específico.

Carpeta app: aquí se encuentran los archivos __init__.py que nos sirve para marcar un directorio como un paquete de Python y ejecutar código de inicialización cuando se importa ese paquete. Y el routes.py que se utiliza para definir rutas que mapean las URLs a funciones específicas que manejan las solicitudes de los usuarios.

Carpeta models: contiene nuestro archivo bdd.py que crea la base de datos y las tablas. También podemos insertar, modificar, y eliminar datos en las tablas y hacer consultas en estas.

Carpeta Static: aquí podremos usar nuestro archivo style.css para darle estilo a nuestras páginas.

Carpeta templates: en esta carpeta se encuentra nuestros archivos HTML que serán la estructura de cada página.

Archivo .env: es un archivo de texto plano que almacena pares clave-valor para configurar una aplicación, como credenciales de base de datos o claves API, de forma segura.

Archivo .gitignore: archivo de texto que le dice a Git qué archivos y directorios debe ignorar, es decir, no rastrear ni incluir en los commits.

Archivo librería.db: se utiliza para almacenar nuestra base de datos, y con la extensión sqlite3 editor podemos visualizar las tablas de la base de datos.

Archivo requirements.txt: aquí podremos listar nuestras dependencias y sus versiones sin necesidad de ejecutar cada vez los comandos en la terminal.

Archivo run.py: este archivo nos permite correr nuestra aplicación. Contiene instrucciones para ser ejecutadas por el intérprete de Python.

3. Librerías y dependencias:

Flask ==3.0.3: sirve para instalar la última versión de Flask.

Flask-bcrypt: es una extensión de Flask que proporciona utilidades para usar el algoritmo de has bcrypt en una aplicación de Python.

Flask-login: es una extensión de Flask que simplifica enormemente la gestión de sesiones de usuario, como el inicio, cierre y mantenimiento de la sesión, y la protección de rutas para que solo los usuarios autenticados puedan acceder a ellas.

Google-auth: es una biblioteca que facilita la autenticación con las APIs de Google.

Google-auth-oauthlib: es una biblioteca de Python que se usa junto con la biblioteca google-auth para gestionar el protocolo OAuth 2.0, permitiendo que las aplicaciones de Python se autoricen en las APIs de Google.

Google-auth-httplib2: es una biblioteca de transporte para la biblioteca de autenticación de Google para Python (google-auth). Su propósito principal es permitir que las aplicaciones Python se autentiquen en las APIs de Google utilizando httplib2 como el mecanismo subyacente para realizar las solicitudes HTTP autenticadas.

Python-dotenv: es un paquete de Python que automatiza la carga de variables de entorno desde nuestro archivo .env al entorno de ejecución.

Requests: es una herramienta para hacer solicitudes HTTP de forma sencilla, permitiendo interactuar con APIs y recursos web.

Gunicorn: sirve para ejecutar aplicaciones web Python de producción al actuar como un servidor WSGI, que actúa como intermediario entre un servidor web y la aplicación web en sí.

4. Instrucciones de ejecución:

Creación del entorno virtual:

A. Ejecutar en la terminal: python -m venv .venv para crear el entorno.
B. Activar el entorno con el comando: .\.venv\Scripts\Activate.ps1

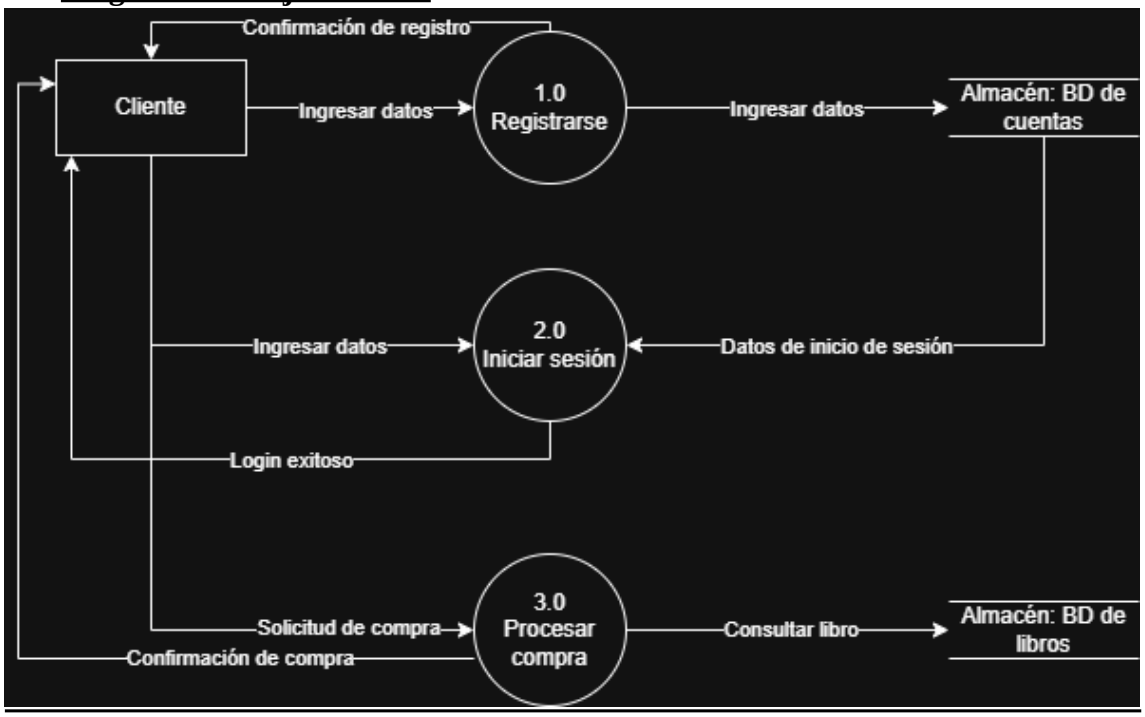
Instalación de dependencias: Ejecutar en la terminal: pip install -r requirements.txt para instalar las dependencias.

Ejecución del servidor: Ejecutar en la terminal python run.py para ejecutar el servidor.

5. Diagramas actualizados:

Diagrama de contexto

Diagrama de flujo de datos



6.Resultados y reflexión:

Se me presentaron algunas dificultades a la hora de instalar mi entorno virtual ya que mi computadora no soportaba la instalación por políticas de Windows. Por eso tuve que ejecutar los comandos del entorno virtual en la consola cmd de mi sistema. También tuve problemas a la hora de subir mi proyecto a un hosting porque no tenía la dependencia gunicorn instalada. Una mejora futura que tendría que implementar es corregir el inicio de sesión con Google cuando ejecuto mi proyecto desde render porque me da error, en cambio cuando lo ejecuto desde visual studio code me funciona perfectamente. También me gustaría añadir más elementos a mi interfaz de inicio ya que me quedó incompleta debido a que no llegué con los tiempos.