Trabajo Practico Integrador

IA3.5 Redes de Datos

Tecnicatura Universitaria en Inteligencia Artificial

Tapia, Fabrizio

26/6/2025

Universidad Nacional de Rosario Facultad de Ciencias Exactas, Ingeniería y Agrimensura

INTRODUCCIÓN

Este trabajo consiste en el desarrollo de una API cliente-servidor la cual nos permite gestionar una colección de libros. La aplicación fue construida con Python utilizando el módulo FastAPI del lado del servidor. El objetivo principal es implementar funcionalidades básicas como agregar, buscar, actualizar y eliminar libros, con un sistema de autenticación básico para restringir el acceso a ciertas acciones.

La información de los libros se almacena en un archivo JSON que actúa como base de datos simple para el proyecto. Desarrollamos un programa en Python capaz de descargar, leer y realizar consultas sobre la información contenida en dicho archivo.

El desarrollo del cliente también se realizó en Python. Este cliente fue programado para interactuar con el servidor, enviando las solicitudes de consulta y modificación, y verificando el correcto funcionamiento de la comunicación.

DESCRIPCIÓN DEL ENTORNO DE TRABAJO DEL SERVIDOR

Configuramos nuestro servidor en Python. Lo armamos para que, ni bien se inicie, ya cargue la base de datos de libros, así está lista para usarse. Pensamos en la seguridad desde dos puntos de vista. Por un lado, le pusimos un sistema de usuario y contraseña para cuando alguien quiere hacer cambios en la base de datos (como agregar, modificar o borrar libros). Si solo van a consultar o ver la información, el acceso es libre. Por otro lado, para protegernos de posibles ataques que podrían saturar el servidor, configuramos un "límite de velocidad" para todas las veces que el cliente y el servidor se comunican.

Cuando estábamos armando todo el entorno, tuvimos que pensar en cómo íbamos a manejar la validación de credenciales. Decidimos que las validaciones se hicieran tanto del lado del cliente como del servidor.

Durante el desarrollo del servidor, nos encontramos con varios obstáculos que fuimos resolviendo. Uno de los primeros problemas que tuvimos fue que los datos que se ingresaban al querer agregar un libro no se estaban validando correctamente. Por ejemplo, se podían cargar libros sin título. Esto nos trajo bastantes inconveniente después, porque si el título estaba incompleto o faltaba, era imposible buscar, actualizar o borrar ese libro, ya que el título era nuestro identificador principal. Por eso, nos dimos

cuenta de que era fundamental agregar validaciones en ambos lados, tanto en el cliente como en el servidor, para asegurarnos de que el título estuviera siempre completo.

Finalmente, al principio no estábamos manejando los errores del servidor de la mejor manera. Si pasaba algo inesperado, el cliente recibía una respuesta un poco confusa o nada clara. Para que esto no pasara, agregamos HTTPException con mensajes personalizados en todos los lugares donde podía surgir un problema. De esta forma, los errores son mucho más informativos y el usuario sabe exactamente qué fue lo que ocurrió.

DESCRIPCIÓN DEL ENTORNO DE TRABAJO DEL DEL CLIENTE

Como ha sido mencionado previamente, decidimos que la validación de credenciales se hicieran tanto del lado del cliente como del servidor.

Inicialmente lidiamos con un error que nos aparecía, relacionado con la autenticación. Al principio, la verificación de si el usuario tenía permiso para hacer algo se hacía casi al final de la función. Esto significaba que el usuario podía pasarse un rato largo completando un formulario (por ejemplo, para actualizar un libro), y recién al final del proceso, el sistema le avisaba que no tenía los permisos necesarios para hacer eso. Para que la experiencia del usuario fuera mejor y más directa, cambiamos el orden y decidimos que las credenciales se verifiquen bien al principio, antes de que el usuario empiece a cargar cualquier dato relacionado con la función que quiere usar.

DESCRIPCIÓN DE LA BASE DE DATOS ELEGIDA

La base de datos utilizada es: “100 best books: Lista de los mejores 100 libros”. La misma se encuentra en formato JSON, y posee 100 objetos, cada uno de ellos correspondiente a un libro. Cada objeto libro tiene a su vez, 8 propiedades: author (autor), country (país), imageLink (link de la imagen, foto de portada), language (lenguaje en el que está escrito), link (link de Wikipedia), pages (número de páginas), title (título), year (año de publicación). Todas las propiedades son de tipo character a excepción de pages e year que son de tipo int.

PROCEDIMIENTO

Para llevar a cabo el desarrollo de la API, primero se definieron las operaciones

principales que debía soportar el sistema: agregar libros, buscar uno en particular, listar todos, actualizar un libro existente y eliminar uno. Estas operaciones la implementamos como endpoints en FastAPI, utilizando métodos HTTP apropiados (Post, Get, Put, Delete).

Se utilizó un archivo books.json como base de datos, lo cual facilitó la manipulación de los datos. La lectura y escritura se manejó con funciones auxiliares, asegurando que los cambios persistieran correctamente.

A los endpoints PUT, DELETE Y POST se les agregó control de acceso mediante autenticación básica, ya que son los que modifican la base de datos. Para ello, se pidió usuario y contraseña desde el cliente y se validó en el servidor con una función que compara contra credenciales predefinidas. Si no se ingresa correctamente, no se permite continuar con operaciones sensibles como eliminar, agregar o actualizar libros.

Además, se incorporó un sistema de rate limiting (limitación de peticiones) para evitar que el servidor se sature en caso de una sobrecarga de solicitudes.

En la parte del cliente, se creó una interfaz por consola para ir pidiendo al usuario los datos necesarios según la acción que quiera realizar. Se validaron campos como el título del libro, el año y la cantidad de páginas, para evitar mandar datos inconsistentes al servidor. También se mostró por pantalla la respuesta del servidor, tanto en casos exitosos como en errores.

Durante el desarrollo, fuimos probando cada funcionalidad paso por paso, corrigiendo errores comunes como rutas mal escritas, datos vacíos o credenciales inválidas. También se hicieron mejoras progresivamente, como asegurarse de que los títulos coincidan sin importar mayúsculas o espacios, y que no se pueda actualizar o eliminar un libro sin título.

Como un “plus”, también se hizo un intento de armar una interfaz gráfica con tkiner. No se llegó a un sistema completamente funcional ni reemplazó a la consola del cliente, pero sirvió como una prueba para ver cómo se podrían integrar botones, campos de entrada y respuestas del servidor en una GUI más amigable. Aunque quedó en etapa experimental, fue útil para practicar cómo conectar la API con una interfaz más visual.

INSTRUCCIONES DE USO DEL CLIENTE

Para ejecutar el lado del cliente de la API, se utiliza el script Python (cliente.py) que realiza solicitudes al servidor. El cliente necesita tener disponible la librería requests

para realizar las peticiones al servidor, y tener el valor adecuado en la variable API_URL, modificando la IP para que se corresponda con la IP del host del servidor

Una vez levantado el servidor, el cliente puede acceder a funcionalidades como el listado de libros completo, revisar la información de un libro en específico, acceder a la página de wikipedia del mismo o visualizar su portada. Estas funcionalidades son de acceso libre, por lo que el usuario puede acceder a las mismas sin necesidad de loguearse en el sistema. Sin embargo, dentro del menú de inicio, también se le presentan al cliente las opciones de agregar/quitar libros de la lista y de actualizar libros existentes. Para estas tres últimas funcionalidades, se le solicitará al cliente que ingrese su usuario y contraseña, para determinar si el mismo posee autorización para interactuar con dichas funcionalidades. En caso de acceso denegado, el usuario puede continuar simplemente visualizando la información del servidor, mientras que si el acceso es concedido, el usuario recibe acceso a la base de datos y se le permite modificarla.

Inicialmente, debe estar corriendo el servidor para que se pueda conectar el cliente. Para ejecutar el servidor: uvicorn server:app --reload. Esta terminal debe permanecer abierta (y por ende, el servidor corriendo) para que el cliente pueda conectarse.

Luego se ejecuta el script del cliente, el cual le va a proponer un menú con las funcionalidades previamente descritas, ante lo cual el cliente debe presionar el número de la opción que desee ejecutar, y completar los campos que se le soliciten.

El cliente y el servidor se montan en hosts distintos.

El puerto 8000 (puerto por defecto) debe estar abierto en el firewall del servidor, para que pueda permitir la conexión del cliente. Se considera que ambos hosts están dentro de la misma red.

REFERENCIAS

1. **Python Software Foundation.** *webbrowser – Convenient Web-browser controller*.

   Disponible en: https://docs.python.org/3/library/webbrowser.html

   (Usado para abrir enlaces como Wikipedia directamente desde el cliente.)

2. **Python Software Foundation**. *tkinter — Python interface to Tcl/Tk*. Python Docs.

   Disponible en: https://docs.python.org/3/library/tkinter.html

   (Para la prueba de interfaz gráfica que se implementó como plus.)

3. **SlowAPI Contributors.** *slowapi – Rate limiting for FastAPI*.
   Disponible en: https://pypi.org/project/slowapi/

   Utilizado para limitar el número de solicitudes al servidor y prevenir abusos.

4. **Python Software Foundation.** *secrets – Generate secure random numbers*.
   Disponible en: https://docs.python.org/3/library/secrets.html

   (Usado para comparar credenciales de forma segura.)

5. **Python Software Foundation.** *getpass – Portable password input*.

   Disponible en: https://docs.python.org/3/library/getpass.html

   (Usado en el cliente para ocultar la contraseña del usuario al escribir.)

6. **FastAPI Team.** *fastapi.security – HTTPBasic*.
   Disponible en:
   https://fastapi.tiangolo.com/advanced/security/http-basic-auth/

   (Autenticación básica HTTP mediante usuario y contraseña.)