Imagen que contiene Forma

Descripción generada automáticamente

**PROYECTO SGE**

**2ª EVALUACIÓN**

|  |
| --- |
| CFGS Desarrollo de Aplicaciones Multiplataforma  Informática y Comunicaciones |

|  |
| --- |
| **<Library of Ohara (Python API Version)>** |
| ***Año:*** *<2025>*  ***Fecha de presentación:*** *(10/02/2025)* |

|  |  |
| --- | --- |
| **Nombre y Apellidos: Jaime González Bravo**  **Email: jaime.gonbra@educa.jcyl.es** |  |

Índice:

Introducción:

El proyecto consiste en un (intento) de API funcional en el que se trabajen con una base de datos relacional (en este caso PosgreSQL) dando servicio a diferentes funcionalidades que se verán en el mismo.

Como resumen, quería comprobar mis conocimientos de Python para probarme a mi mismo y conseguir una API parecida a la que quiero hacer en mi TFG.

Estado del Arte:

Arquitectura de Microservicios (API):

Es un método de desarrollo de aplicaciones de software que funciona como un conjunto de pequeños servicios que se ejecutan de manera independiente y autónoma proporcionando una funcionalidad de negocio completa. Los microservicios se comunican entre sí a través de APIs y cuentan con un sistema de almacenamiento propio (una base de datos). De esta forma, se facilita el uso y desarrollo de estas, ya no es una aplicación de gran tamaño sino varios mini proyectos que se encargan de una cosa en específico.

API (Interfaz de Programación de Aplicaciones):

es un conjunto de protocoles que se usa para diseñar e integrar el software de las aplicaciones. Permiten que tus productos y servicios se comuniquen con otros, sin necesidad de saber cómo están implementados. Por tanto, simplifica el desarrollo y permite ahorrar tanto tiempo como dinero.

Estructura de una API:

La estructura básica de una API suele seguir ciertas convenciones para permitir la interacción de clientes y servidores.

Como protocolo, se utiliza el de HTTP/HTTPS que es el estándar para todas las comunicaciones, siendo el HTTPS el mejor ya que es más seguro.

Los métodos más utilizados son:

GET: petición para obtener la información del servidor.

POST: petición para enviar datos (se suele utilizar para crear nuevos recursos).

PUT: petición para enviar datos (se suele utilizar para actualizar recursos).

PATCH: similar que el PUT, pero para un solo campo.

DELETE: petición para eliminar un recurso.

Las partes más características de una API son:

El esquema que hace referencia al protocolo (https://).

El dominio que es la dirección del servidor donde se encuentra la API (localhost:8000)

La ruta que es especifica para cada recurso que la API maneja (/api/usuario/all)

Los parámetros, que ya sean los que se emplean en la propia ruta o bien, los parámetros que se utilizan en los cuerpos de las peticiones (body).

Formas de crear una API (FastAPI):

FastAPI es uno de los frameworks más populares y rápidos a la hora de crear APIs en Python debido a su rendimiento y facilidad. Para poder utilizarlo, requieres instalar el servidor ASGI (servidor de aplicaciones asíncronas) como el denominado Uvicorn. Además, este framework, implementa la documentación interactiva del api conocido como Swagger que explicará y nos dejará probar todas las APIs que hayamos creado.

Descripción general de proyecto:

Objetivo:

La idea principal del proyecto es crear una API similar a la que quiero utilizar para el TFG utilizando los conocimientos aprendidos durante las clases. Además de así practicar con Python.

Entorno de Trabajo:

Como Gestor de base de datos, he utilizado PosgreSQL ya que tiene buena sinergia trabajando con Python y, además, utilizo Docker para levantar el servicio. De esta manera, puedo ejecutar y lanzar el servicio en cualquier momento.

Para poder ver las tablas y los cambios realizados, utilizo PGAdmin que es la interfaz de usuario que se conecta a PosgreSQL para mostrar todos los campos de las tablas.

Como lenguaje de programación, he utilizado Python debido a su fácil uso además de Visual Studio Code como IDE para poder ejecutar el código. Como framework empleado para la creación de la API, empleo el ya denominado FastAPI.

Documentación técnica:

Análisis del sistema:

Debido a la falta de tiempo y en algunos casos, falta de conocimientos, no dispongo de aplicación móvil ni de sistema de autenticación. Lo que el usuario puede llegar a hacer es la de creación, modificación visualización y eliminación de Usuarios, Libros, Géneros y Autores. Es bastante básico, pero es lo que me ha dado tiempo a realizar.

Diseño de la BD:

Esta es la idea de la cual partí

Diagrama, Esquemático

El contenido generado por IA puede ser incorrecto.

Finalmente, acabo así.

Texto

El contenido generado por IA puede ser incorrecto.

Como se puede ver en la imagen, lo mas complicado fue el tema de las relaciones entre género, autor y libro respectivamente. Porque, aunque solo se guarde en la base de datos en libro los ids del autor y del género, cada vez que visualizo uno autor o un género, muestro también los libros a los que hace referencia dicha relación. (créeme, ha sido lo más complicado).

Implementación:

Texto

El contenido generado por IA puede ser incorrecto.

Aunque al principio parece todo muy enrevesado, es una organización “simple”.

Por un lado, tenemos la carpeta db con todo lo relacionado con la base de datos. En ella, tenemos el fichero models con el modelo de base de datos y el de database que configura todo lo necesario para la conexión a base de datos.

Texto

El contenido generado por IA puede ser incorrecto.

Utilizo la biblioteca de sqlachemy que proporciona herramientas para interactuar con bases de datos relacionales de manera eficiente y flexible (parecido a un ORM).

Texto

El contenido generado por IA puede ser incorrecto.

Siguiendo por orden, llega la carpeta modelos, en el que están como su nombre indica, las clases con los modelos que se van a utilizar para el correcto uso de la aplicación. Dichas clases son una imagen y semejanza con el de la base de datos añadiendo algunas cosas.

Pantalla de computadora con letras

El contenido generado por IA puede ser incorrecto.

Como podemos ver en la imagen, tenemos la clase usuario que es idéntica a la de la bd y la clase UsuarioDTO que me sirve a mi a la hora de insertar un nuevo usuario o modificar uno ya existente. Utilizo pydantic que es una biblioteca de Python que se utiliza principalmente para la validación de datos y para la creación de modelos de datos de manera sencilla, eficiente y segura.

En el fichero género, también tenemos una clase Genero que es idéntica al de la bd haciendo referencia a la lista con los libros relacionado y la del DTO para facilitarme el insertado y el modificado de datos.

Texto

El contenido generado por IA puede ser incorrecto.

El fichero de Autor es muy parecido al de género. En él, tenemos tanto la clase autor que hace referencia al de la base de datos y el DTO.

Texto

El contenido generado por IA puede ser incorrecto.

Por ultimo y no menos importante, tenemos el fichero de Libro con las clases de libro y libroDTO. En dicha clase, tenemos el id que hace referencia tanto al autor y al genero al que pertenece dicho libro.

Pantalla de computadora con letras

El contenido generado por IA puede ser incorrecto.

Siguiendo con la primera imagen, tendremos el archivo repository en el que se encapsula la lógica de acceso a la base de datos. En él, se debería de meter toda la funcionalidad que realiza nuestra API (cosa interesante de saber antes de realizar este proyecto). Para simplificar, todo ese código lo tengo en los routers.

Los routers están todos juntos y representan tanto la dirección como lo que van a realizar dichos microservicios. Para resumir, solo mostraré uno ya que el resto es bastante parecido la verdad.

Como podemos ver en la imagen, inicia con el router que sirve para indicarle a la aplicación que este fichero “controla” todas las acciones de autores. Es solo una manera de organizar y no tener todo en un mismo fichero. Tal y como hemos visto con anterioridad, tenemos cada uno de los métodos de las APIs (get,post,put,delete).

Como primer método, es el método get all que recibe todos los autores de la bd. (curiosidad, en este método siempre me mostraban los libros, en los otros siempre me daban problemas.) también esta el de getByID en el que me busca en base de datos un autor que tenga dicho id que pasamos por parámetro. Si no existe, le enviamos al usuario un mensaje de que no existe dicho autor y si existe, le pasamos el autor.

Texto

El contenido generado por IA puede ser incorrecto. Texto

El contenido generado por IA puede ser incorrecto.

Texto

El contenido generado por IA puede ser incorrecto.