BOOKIEGAMES

OBJETIVO DE LA API:

**Controlar una base de datos privada sobre videojuegos y libros mediante el uso de una api desarrollada en Python mediante la librería FLASK.**

**La aplicación cuenta con diversas rutas que nos permiten interactuar con la base de datos para poder recibir datos, insertarlos, modificarlos o eliminarlos.**

**Podremos acceder a estas rutas levantando el servidor, la base de datos y el servidor de apache (mediante herramientas como xampp).**

Cómo levantar la base de datos:

La api viene con un archivo sql para poder levantar de forma rápida la base de datos que esta api requiere.

Para hacerlo posible, basta con tener un controlador de bases de datos de tipo MySQL e importar el archivo “bookiegames.sql”, que se encuentra en la carpeta utils, en la raíz de src (esto puede hacerse mediante una consola MySQL o con un administrador de bases de datos como phpMyAdmin).

Para poder acceder a la base de datos, necesitamos realizar cambios en el archivo “config.py”, localizado en la raíz de la carpeta src:

Dentro de la clase DevelopmentConfig() encontraremos una serie de elementos de acceso a la base de datos que deberemos configurar para poder acceder con nustros datos a la base de datos. Por defecto estos son:

-MYSQL\_HOST = “localhost”

-MYSQL\_USER = “root”

-MYSQL\_PASSWORD = “”

-MYSQL\_DB = “bookiegames”

Deberemos acomodar estos parámetros entrecomillados a nuestros datos de acceso a la base de datos.

En este punto ya tendremos la base de datos levantada y la api podrá interactuar con ella.

La base de datos viene por defecto set de elementos insertados para mejor visualización de la base de datos desde el inicio.

Cómo funciona la base de datos:

La base de datos de bookiegames es muy sencilla y puede ser manejada por cualquier persona con conocimientos básicos de MySQL .

La base de datos por defecto cuenta con solo dos tablas:

-Libros

-Videojuegos

Ambas tablas contienen 4 columnas, ID (clave primaria), Nombre (clave única), nota, imagen.

* La columna ID funciona como un número entero autoincremental y en ningún caso deberá tenerse en cuenta a la hora de hacer cualquier tipo de consulta SQL, ya que esto podría producir fallos en la base de datos dada su naturaleza de clave primaria autoincremental.
* La columna Nombre funciona como cadena de texto de tamaño 50, lo que nos permitirá introducir cualquier nombre de cualquier libro o videojuego a nuestra base de datos. Dada su naturaleza de clave única, se nos impedirá realizar consultas SQL que hagan coincidir el nombre del elemento con otro que existiera previamente en la base de datos.
* La columna nota nos permite introducir una nota subjetiva sobre el libro o videojuego que nos interese. Es una columna de tipo entero con un rango de 0-10.
* La columna imagen nos permite introducir una URL a una imagen de internet, de forma que se pueda utilizar en el front-end para mostrar imágenes.

Cómo levantar la API:

Bookiegames viene con un entorno virtual ya creado, por lo que no es necesario crearlo, sin embargo, habrá que instalar todas las librerías necesarias para trabajar con flask. Para hacer esto, basta con lanzar el comando “pip install -r requirements.txt”, el cual leerá el archivo con dicho nombre e instalará las librerías solicitadas ahi. Es importante no modificar este archivo, a no ser que se necesite para ampliar la funcionalidad de la API.

Para levantar la api en un servidor local solo hay que ejecutar el siguiente comando en una consola, colocándonos en la ruta raíz de bookiegames: “-app api.py –debug run”.

En este punto la api ya estaría levantada y podríamos trabajar con ella.

**RECOMENDACIÓN:** Si no se está interesado en desarrollar un front-end, se aconseja la utilización de herramientas gratuitas como Postman o Insomnia para realizar las peticiones a la api.

Cómo Implementar un front-end para bookiegames:

Bookiegames cuenta por defecto con un front-end creado con templates de HTML, pero este se puede obviar o modificar.

Como ya se ha explicado anteriormente, el front-end no es necesario para trabajar con bookiegames, pero la api está preparada por si queremos implementarlo.

Para implementar un front-end en bookiegames necesitaremos crear una carpeta llamada “templates” en la raíz de la carpeta src. En esta carpeta es donde crearemos nuestros archivos HTML para ser mostrados mediante las rutas con método GET de bookiegames. Estos archivos HTML se comunicarán con la api mediante requests de JavaScript en los que debe especificarse el método que queremos utilizar (GET, POST, PUT, DELETE).

Una vez creada una template, tendremos que llamarla desde las rutas de bookiegames dentro de el método flask.render\_template(“nombre\_template.html”).

Rutas de bookiegames:

La api cuenta con una serie de rutas que nos permiten trabajar con el front-end, recuperar datos de este, e interactuar con ellos en la base de datos. A continuación se detalla una lista de las funciones de cada una:

-**“/”**

Devuelve el template del índice de la página.

-**“/videojuegos”**

Devuelve el template de la página inicial de videojuegos.

-**“/videojuegos/agregar”**

Devuelve el template de la página de agregar videojuego.

-**“/videojuegos/agregar”(POST)**

Obtiene los datos del front y conecta con la base de datos para agregar un nuevo videojuego.

-**“/videojuegos/eliminar”**

Devuelve el template de la página eliminar videojuego.

-**“/videojuegos/eliminar”(DELETE)**

Obtiene los datos del front y conecta con la base de datos para eliminar un videojuego.

-**“/videojuegos/ver\_todos”**

Devuelve el template de la página ver todos los videojuegos.

-**“/videojuegos/modificar”**

Devuelve el template de la página modificar videojuego.

-**“/videojuegos/modificar”(PUT)**

Obtiene los datos del front y conecta con la base de datos para modificar un videojuego.

-**“/videojuegos/correcto”**

Devuelve el template de la página que aparece al realizar con éxito un cambio sobre un videojuego en la base de datos. Algunos métodos de la api llaman a esta ruta al terminar.

-**“/libros”**

Devuelve el template de la página inicial de libros.

-**“/libros/agregar”**

Devuelve el template de la página de agregar libro.

-**“/libros/agregar”(POST)**

Obtiene los datos del front y conecta con la base de datos para agregar un nuevo libro.

-**“/libros/eliminar”**

Devuelve el template de la página eliminar libro.

-**“/libros/eliminar”(DELETE)**

Obtiene los datos del front y conecta con la base de datos para eliminar un libro.

-**“/libros/ver\_todos”**

Devuelve el template de la página ver todos los libros.

-**“/libros/modificar”**

Devuelve el template de la página modificar videojuego.

-**“/libros/modificar”(PUT)**

Obtiene los datos del front y conecta con la base de datos para modificar un libro.

-**“/libros/correcto”**

Devuelve el template de la página que aparece al realizar con éxito un cambio sobre un libro en la base de datos. Algunos métodos de la api llaman a esta ruta al terminar.