Challenge Python

Objetivo general

Se requiere desarrollar una API que permita interactuar con datos de prueba y realizar peticiones GET, POST y DELETE.

Características de los datos

Se brinda un ejemplo de la estructura de datos:

Character

```
{
        "id": 1,
        "name": "Luke Skywalker",
        "height": 172,
        "mass": 77,
        "hair_color": "blond",
        "skin_color": "fair",
        "eye_color": "blue",
        "birth_year": 1998,
}
```

Todos los campos de la tabla Character deberán ser de tipo string excepto id, height, mass y birth_year que deberán ser numéricos. El campo id se considera identificador de cada ítem y deberá ser único.

Funciones disponibles para el usuario

La API debe proporcionar las siguientes funcionalidades:

- Como usuario de la API puedo consultar todos los characters guardados.
- Como usuario de la API puedo consultar los datos de un character buscado por el campo id
- Como usuario de la API puedo insertar un nuevo character
- Como usuario de la API puedo eliminar un characters buscado por el campo id

Características técnicas

El servicio deberá tener las siguientes características:

- Se deberá usar framework de Python para crear la aplicación (FastAPI o Flask)
- Deberá implementarse sobre un ambiente virtual.
- Deberá implementarse en un repositorio GIT.
- Se requiere utilizar una base de datos SQLite para la persistencia de datos.
- Utilizar SQLAlchemy como ORM.
- Se deberá inicializar la base de datos desde la API creando los modelos necesarios.
- Retornar la lista de todos los ítems existentes.
    - La ruta deberá ser */character/getAll* [GET]
    - Deberá retornar la lista con todos los *characters*. Los datos de cada *character* deberán ser:
        - Id, name, height, mass, birth_year, eye_color

  Esta estructura de datos deberá ser validada al momento de retornarla.

- Retornar characters buscados por su id
    - La ruta deberá ser */character/get/{*id*} [GET]*
    - Deberá retornar todos los datos de los character. Esta estructura de datos deberá ser validada al momento de retornarla.
- Insertar un nuevo character
    - La ruta deberá ser */ character /add [POST]*
    - El nuevo character deberá cumplir las siguientes condiciones:
        - Deberán existir todos los campos del character nuevo
        - No puede tener valor nulo ninguno de los campos
        - Se deberán respetar los tipos de campos
        - El ítem no deberá existir (controlando por su id). Si el character ya existe deberá retornar un HTTP Code 400 – Bad Request con el detalle acorde al problema que se produjo.

  Esta estructura de datos deberá ser validada al momento de recibirla. Se deberá retornar el character con todos sus campos, estructura que también deberá verificarse al momento de retornar.

- Eliminar un character
    - La ruta deberá ser */character/delete/{*id*} [DELETE]*
    - Si el character existe se elimina y se deberá retornar un HTTP CODE 200 con un detalle acorde a la acción que se realizó
    - Si el character no existe deberá retornar un HTTP Code 400 – Bad Request con el detalle acorde al problema que se produjo.

Consideraciones

Algunos de los puntos que serán tenidos en cuenta son los siguientes:

- Aplicar buenas prácticas en organización del proyecto (clean architecture).
- Aplicar buenas prácticas de código y comentarios.
- Se deberá documentar cada ruta de la API e implementar una ruta para consulta.
- El proyecto deberá estar disponible en un repositorio GIT con la documentación necesaria para poder correr localmente el mismo.
- Se valorará la implementación de un Dockerfile para poder levantar la api en un contenedor de docker.
- Se valorará la la construcción de una colección en PostMan (o similar) para la prueba de la API.

Cualquier duda o consulta comunicarse a la dirección donde recibiste el challenge.