PRUEBA TÉCNICA PYTHON

Kerman Sanjuan

Introduccion

El trabajo se ha dividido en dos ficheros, cada uno correspondiendo al lenguaje de programación con el que se ha desarrollado la tarea. Para que sea mas sencilla la ejecución y la prueba de la API, he preparado un **docker** que prepara el entorno conecta con los puertos 5000(python) y 5005(nodejs) del ordenador respectivamente. Es decir, con runnear el docker ya es suficiente para que todo se ponga a funcionar.
docker-compose up en el directorio principal.

Python:

Modo de uso:

Con una solicitud get se manda el siguiente JSON en el body:

```
{
    "username":"X",
    "postal_code":"Y"
}
```

Si el usuario existe, se añadirá el código postal, puede ocurrir que varias localizaciones tengan ese código postal, es por ello que se añade solo una de esas posibles localizaciones. Dependiendo de lo que ocurra con el input, este endpoint devolverá diferentes status con sus correspondientes descripciones.

Explicaciones y decisiones tomadas:

He desarrollado una API-REST en Python como se menciona en el enunciado. He utilizado la librería Flask, ya que me parece que es mas simple de utilizar y requiere menos configuración (en contra de Django). He optado por estructurar de forma sencilla el proyecto, habiendo diferentes .py para repartir las funcionalidades y así modularizar un poco el código.

En el enunciado se menciona la posibilidad de implementar diferentes endpoints, es por ello que he creado un fichero donde programar cada endpoint (Sería mas optimo tenerlos en una carpeta, pero para este caso sirve así, ya que solo tenemos un endpoint).

Dentro de la programación del endpoint, he optado por separar lo relacionado con la llamada de la API de Geoname, y meter las cosas extras en un fichero llamado utils.py. En este fichero he gestionado la llamada a la API, y posteriormente, una llamada a la base de datos para añadir los datos obtenidos.

Para la base de datos he optado por utilizar sqlite debido a la simpleza y la portabilidad que esta tiene. Además de que para esta tarea lo óptimo es utilizar una base de datos relacional.

He creado dos tablas con un identificador de tipo incremento, para que cada usuario pueda realizar mas de una búsqueda. Estas tablas están relacionadas por este índice, siendo la tabla detalles dependiente de la id de la tabla Master.

Para realizar la conexión y la creación he utlizado SQLalchemy, una ORM que trata las tablas como objetos, manteniendo así un orden mas adecuado si se quiere trabajar en un futuro con mas tablas y endpoints, además de no tener que escribir querys (en tareas sencillas) directamente sobre el código. La base de datos se crea en caso de no existir, estando guardada en la carpeta database.

Los posibles errores generados por la API de Geoname o por problemas con la base de datos están tratadas en el código.

Como extra, en caso de visitar un endpoint no existente, se renderiza un html con 404 not found.

NodeJS/Typescript

Modo de uso:
Con una solicitud GET vacia se obtienen una lista de los recursos. Si en la URL especificamos el parámetro *postal_code* listamos los recursos que tienen ese código postal.

`http://127.0.0.1:5005/geolocate?postal_code=48992`

Lo mismo para el método DELETE, pero en ese caso se borraran los recursos.

Decisiones tomadas:

Se encuentra en la carpeta nodejs. Las librerías utilizadas han sido principalmente express y typeorm.

En esta implementación he mantenido una estructura de ficheros mas ordenada, ya que es mas sencillo ordenar los imports. De parecido modo al ejercicio anterior, se ha creado una carpeta llamada routes para gestionar todos los posibles enpoints. En este ejercicio se pedía la implementación de las funciones vinculadas con los protocolos GET y DELETE, es por ello que cada una de estas ha recibido un trato diferente.

Una vez mas la base de datos utilizada ha sido SQLite.

Para la base de datos he utilizado la misma estructura de datos, pero para evitar conflictos entre las dos implementaciones, utilizo dos bases de datos diferentes (Previamente inicializada con datos en su interior). También he utilizado una ORM para gestionar la conexión y las querys, permitiéndome así tener diferentes modelos relacionados con las tablas.

Nota: Si se quiere comprobar la funcionalidad, la base de datos viene inicializada con datos dentro. Recomiendo mirar cuales son los códigos postales que contiene para hacer las respectivas pruebas (Propongo ejecutar un get para listar los recursos, ejecutar el delete y después otra vez el get para ver los cambios)

Extra

Como extra y debido a no hacer la parte de Java. He creado un docker-compose para facilitar el uso de la herramienta. En mi caso personal he utilizado la aplicación Postman para comprobar y hacer los test pertinentes.