Examen Construcción y Evolución de Software

Nombre: Henry Ludeña

Fecha: 15/06/2024

* Compilar:

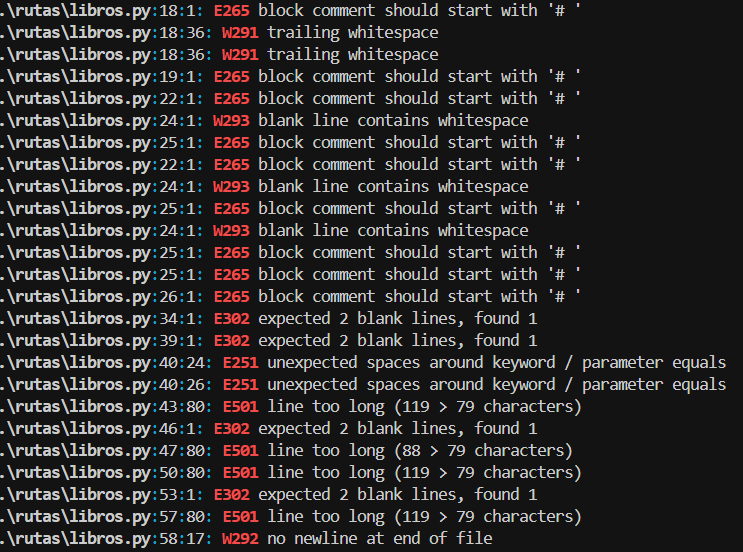
No hay compilación en Python

* Lint:

En Python se usa flake8, posterior a instalarlo mediante el comando: pip install flake8. Se procede a la ejecución de flake 8 para conocer que errores a advertencias de estilo se debe seguir según las guías de estilo de Python (PEP8).

Ejecución del comando: flake8 .

* Este comando permite analizar todos los archivos Python a partir del directorio actual (.) y muestra los errores de estilo y sintaxis leve. Por lo cual no corrige nada, solo muestra donde el código no cumple con las reglas.



A continuación, se describe el significado de las recomendaciones:

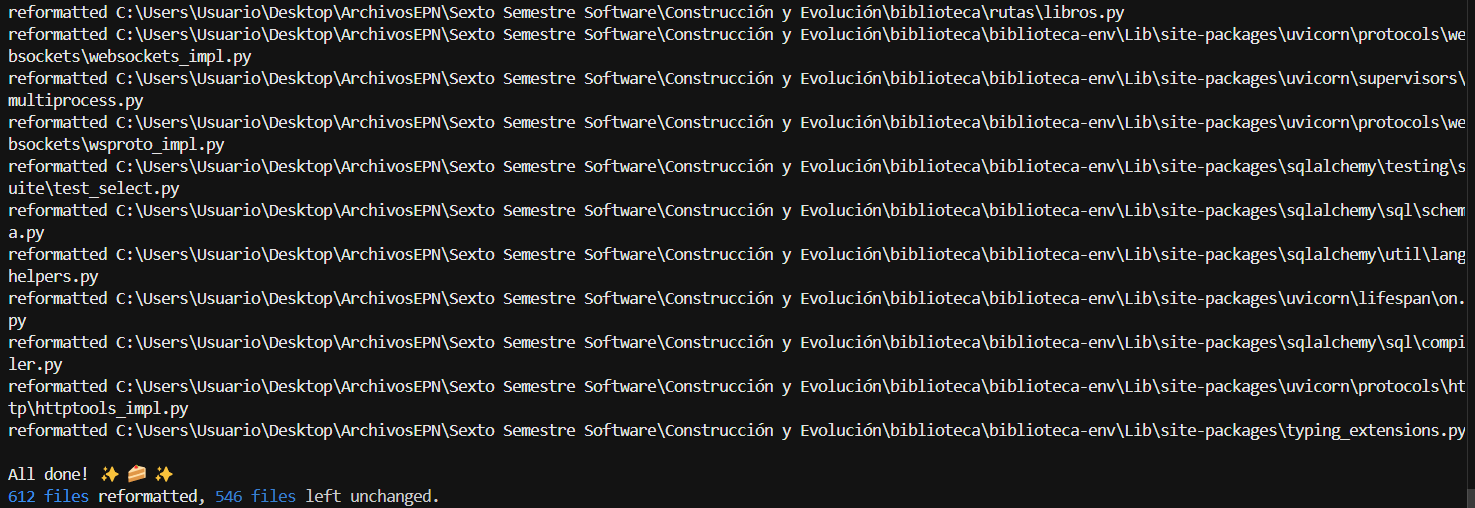
* E265: Comentario debe empezar con # seguido de un espacio
* E251: Hay espacios extra alrededor de = al definir argumentos
* E501: Línea demasiado larga (más de 79 caracteres)
* W292: El archivo no termina con una línea en blanco
* W293: Línea en blanco tiene espacios (debería estar vacía)
* E302: Se esperan 2 líneas en blanco antes de una función o clase

Para este caso existen varias opciones, pero las más sencillas son:

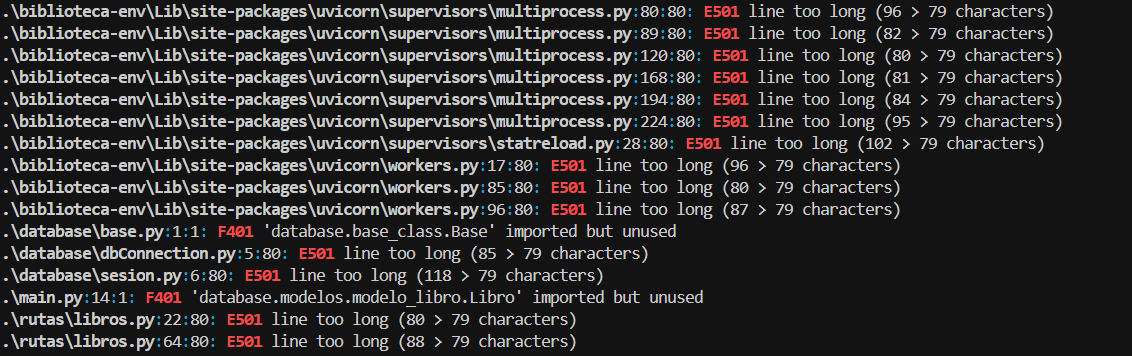
* Arreglar todo manualmente
* O, usar Black que en este caso ayuda a que el código cumpla con PEP8 que es lo que presenta flake 8

En el caso de Black se lo instala mediante: pip install black

Y se corre el comando: black .



Si se vuelve a correr flake 8, en su mayoría notifica faltas en el entorno virtual de Python más no en el proyecto en sí, además de algunas faltas por la longitud de algunas líneas de código.

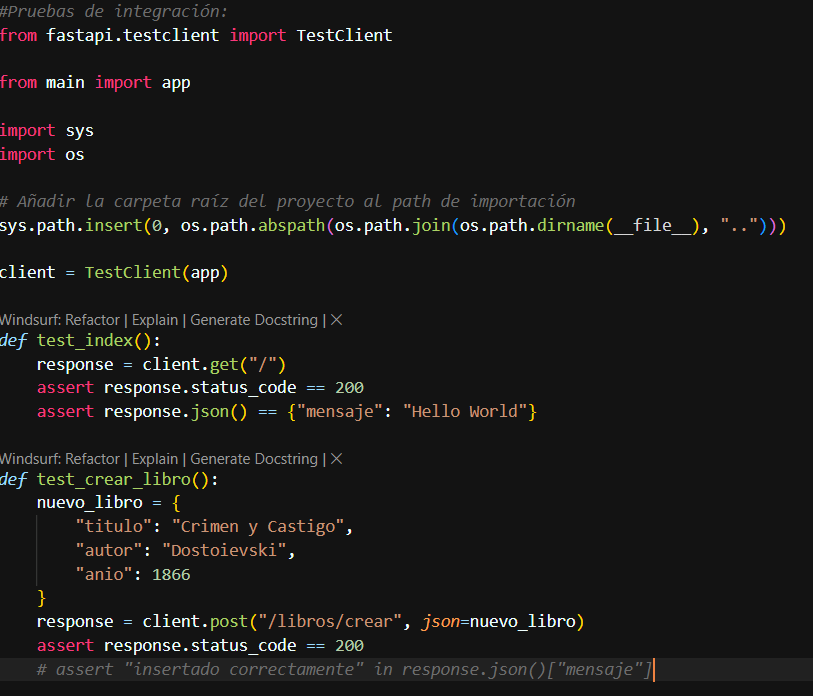


* Pruebas:

Para esta sección se van a realizar dos pruebas de integración.

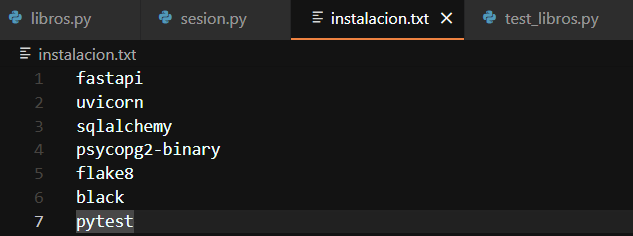
* La primera es para probar que la ruta raíz (/) funciona correctamente y arroja el mensaje esperado, que es el que se colocó por defecto.
* La segunda es para verificar que un libro se crea correctamente y el servidor devuelve un código 200, el cual indica que se ha devuelto una respuesta satisfactoria.

La ventaja al usar FastAPI es que las pruebas vienen integradas, por lo cual solo se enecsita importar TestClient

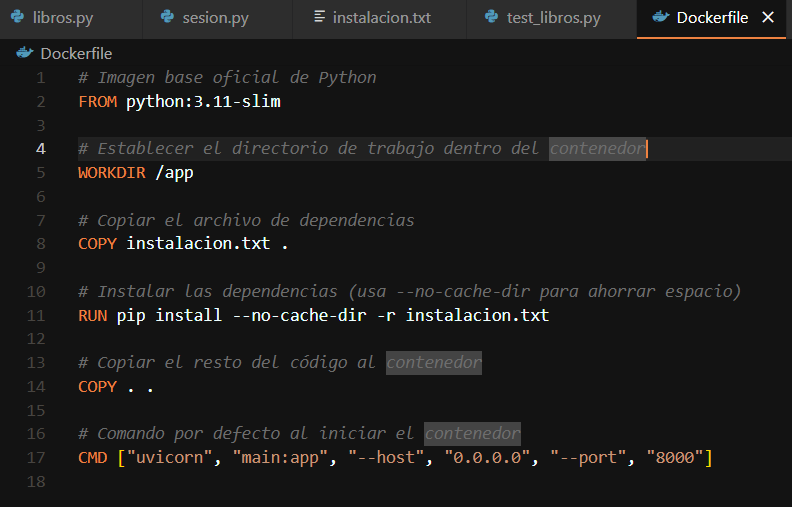


* Dockerfile

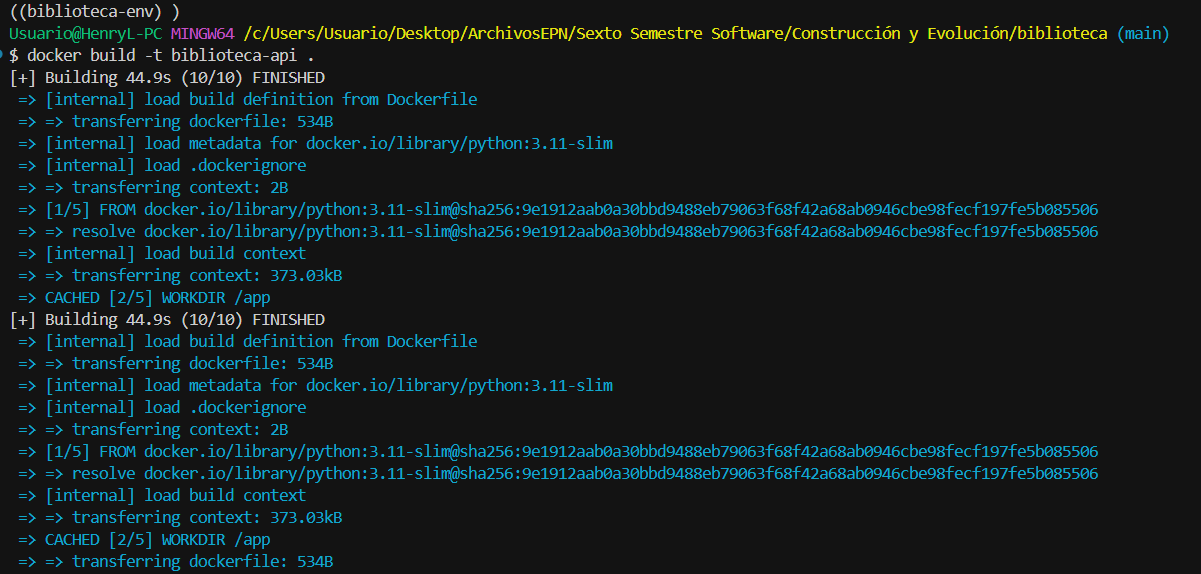
Para esta parte primero se crea el .txt que enlista las dependencias necesarias para correr el archivo:



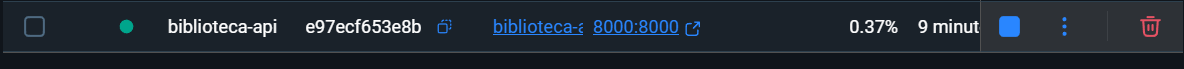
Posteriormente se crea el Dockerfile, indicando el archivo instalación.txt, la imagen de Python, estableciendo el directorio de trabajo e indicando el comando por defecto para iniciar el contenedor:



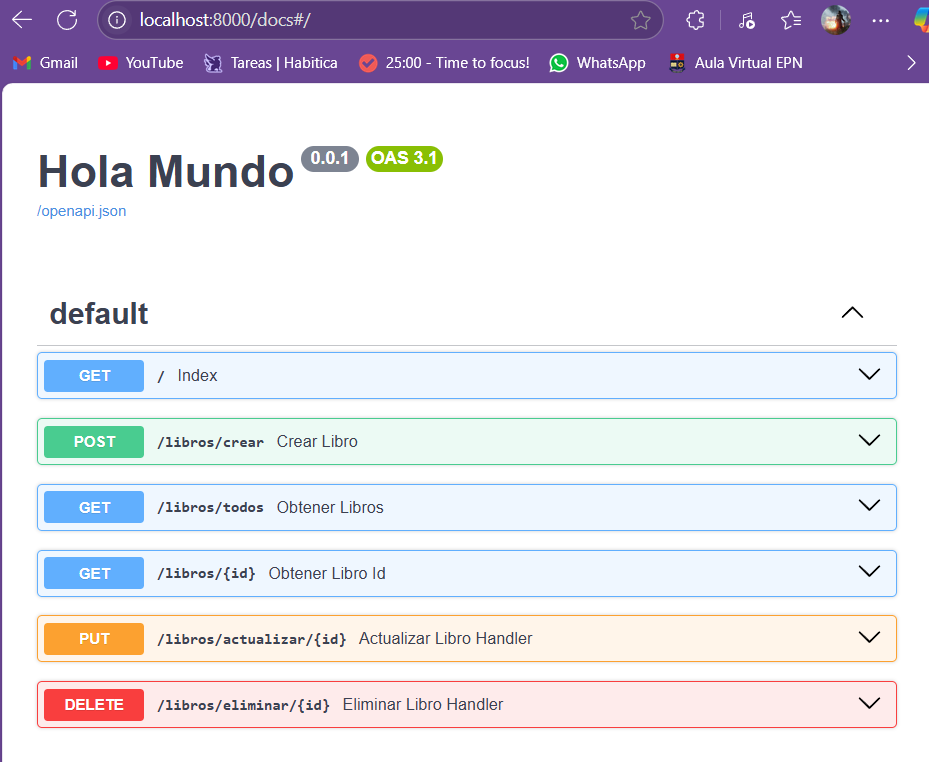
Luego se construye la imagen (con Docker desktop corriendo en segundo plano), mediante el comando: “docker build -t biblioteca-api .” , el resultado es el siguiente:



Luego se corre el contenedor con docker run -d -p 8000:8000 --name biblioteca-api biblioteca-api, y se puede verificar en Docker Desktop que se ejecuta correctamente:



Y al visitar localhost: 8000 se puede verificar que todo funciona correctamente:



Como extra se subió el archivo a Docker Hub para una instalación más sencilla en la nube

Texto

El contenido generado por IA puede ser incorrecto.

