Universidad Tecnologica de Xicotepec de Juárez

Ingeniería en Desarrollo y Gestión de Software

Materia: Extracción de Conocimiento en Bases de Datos

Como implementar swagger en python

Integrantes:

Álvarez Ramírez Daniela M-200644

Hernández Bonilla Lesly Yareth M-180247

Rosas González Marco Antonio M-200749

Solis Sanchez Alfredo M-200291

Docente: MTI. Marco Antonio Ramírez

9° “A”

Enero-Abril 2023

Para empezar, debemos saber que es Swagger y que es Python

¿Qué es Swagger?

Swagger es una especificación abierta para definir las API REST.

Un documento Swagger es el equivalente de la API REST de un documento WSDL para un servicio web basado en SOAP.

El documento Swagger especifica la lista de recursos que están disponibles en la API REST y las operaciones que se pueden llamar en esos recursos. El documento Swagger también especifica la lista de parámetros de una operación, incluido el nombre y el tipo de los parámetros, si los parámetros son necesarios o opcionales, e información sobre los valores aceptables para dichos parámetros. Además, el documento Swagger puede incluir el esquema JSON que describe la estructura del cuerpo de solicitud que se envía a una operación en una API REST, y el esquema JSON describe la estructura de los cuerpos de respuesta que se devuelven de una operación.

Los documentos Swagger deben estar en formato JSON con la extensión de archivo .json o en formato YAML con la extensión de archivo .yaml o .yml.

¿Qué es Swagger UI?

Swagger UI es una de las herramientas atractivas de la plataforma. Para que una documentación sea útil necesitaremos que sea navegable y que esté perfectamente organizada para un fácil acceso. Por esta razón, realizar una buena documentación puede ser realmente tedioso y consumir mucho tiempo a los desarrolladores.

¿Para qué se usa el Swagger UI y por qué?

Utilizando el Swagger UI para exponer la documentación de nuestra API, podemos ahorrarnos mucho tiempo. Con el Swagger UI podremos organizar nuestros métodos e incluso poner los ejemplos que necesitemos.

Swagger UI utiliza un documento JSON o YAML existente y lo hace interactivo. Crea una plataforma que ordena cada uno de nuestros métodos (get, put, post, delete) y categoriza nuestras operaciones. Cada uno de los métodos es expandible, y en ellos podemos encontrar un listado completo de los parámetros con sus respectivos ejemplos. Incluso podemos probar valores de llamada.

Sin duda, Swagger, Swagger UI y todas sus herramientas, son capaces de hacer el trabajo de nuestros desarrolladores mucho más fácil a la hora de documentar nuestras APIs. Unas APIs bien documentadas significa que son accesible por otros desarrolladores, y mejorando la accesibilidad de nuestras APIs podremos mejorarlas.

¿Qué es Python?

Python es un lenguaje de programación de código abierto, creado por Guido van Rossum en 1991. Se trata de un lenguaje orientado a objetos, fácil de interpretar y con una sintaxis que permite leerlo de manera semejante a como se lee el inglés. Es un lenguaje interpretado, esto significa que el código de programación se convierte en bytecode y luego se ejecuta por el intérprete, que, en este caso, es la máquina virtual de Python.

¿Para qué sirve Python?

La respuesta es breve: para todo. Python está en todo, desde programación de instrumentos hasta software de computadoras, desarrollo web y aplicaciones móviles. Incluso, te permite hacer comentarios para que tengas recordatorios para funciones futuras o indicar problemas en una línea de código.

Python es genial para casi cualquier necesidad de desarrollo, ya sea programación de servidores, operación de sistemas, software, juegos y mucho más. A continuación, repasamos los usos más comunes.

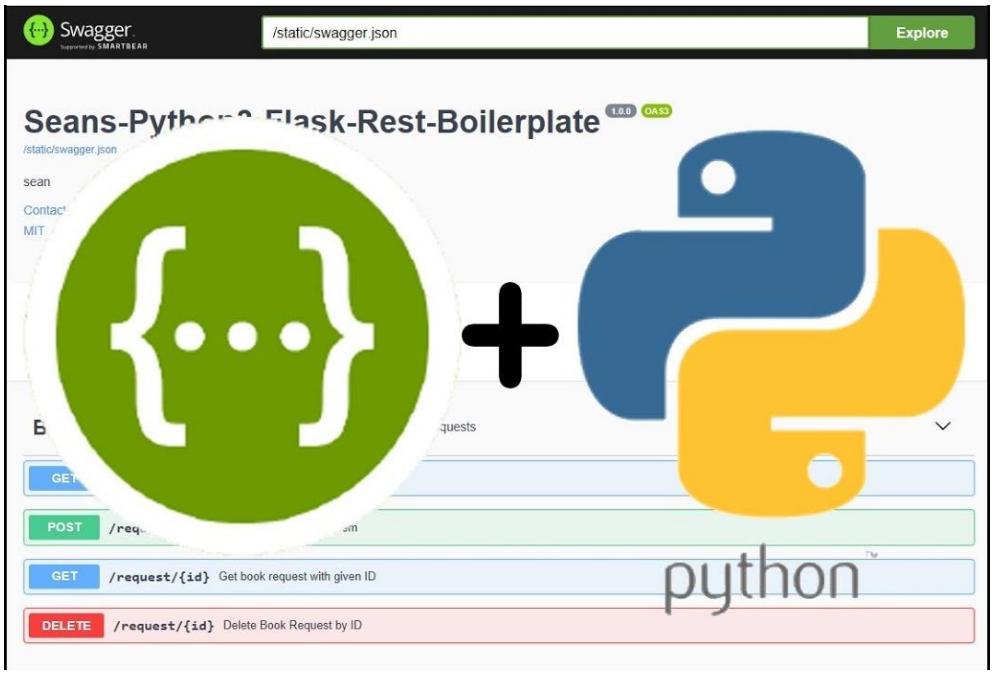

Ilustración 1 Swagger+Python

Cómo crear una API en Python

Como científico o ingenieros de datos, muchas veces tenemos que compartir nuestro trabajo para que cualquier otra persona de la empresa pueda utilizar los procesos o modelos que hemos creado. Claramente, compartir un script no es una opción, ya que todo el mundo necesitaría tener esos mismos programas que tú. Aquí es cuando entran en juego las APIs y hoy, vamos a ver qué son y cómo crear una API en Python. ¿Suena interesante? ¡Pues vamos a ello!

Conceptos básicos sobre APIs

Una API (Application Programming Interface) permite que dos sistemas informáticos interactuen entre sí. Por ejemplo, si creamos una automatización que genere un informe y lo mande por email, el envío de ese email no se hace de forma manual, lo hará el propio script. Para ello, Python (o el lenguaje que usemos), deberá pedirle a Gmail que mande ese correo, con ese informe adjunto a ciertas personas. La forma de hacerlo es mediante una API, en este caso la API de Gmail

Bien, ahora que sabes lo que es una API, veamos cuáles son las partes principales de una API:

Protocolo de transferencia HTTP: es la forma principal de comunicar información en la web. Existen diferentes métodos, cada uno de ellos utilizados para diferentes cuestiones:

GET: este método permite obtener información de la base de datos o de un proceso.

POST: permite mandar información, ya sea para añadir información a una base de datos o para pasar el input de un modelo de machine learning, por ejemplo.

PUT: permite actualizar información. Generalmente se usa para gestionar información en base de datos.

DELETE: este método se utiliza para eliminar información de la base de datos.

Url: es la dirección en la que podremos encontrar a nuestra API. Básicamente esta URL constará de tres partes:

Protocolo: como toda dirección, puede ser http:// o https://

Dominio: el host en el que está alojado, que va desde el protocolo hasta el final del .es o .com, o la terminación que sea. En mi web, por ejemplo, el dominio es anderfernandez.com.

Endpoint: al igual que una web tiene varias páginas (/blog), (/legal), una misma API puede incluir varios puntos y que cada uno haga cosas diferentes. Al crear nuestra API en Python nosotros indicaremos los endpoints, por lo que debemos asegurarnos de que cada enpoint sea representativo de lo que haga la API que está por detrás.

Bien, ahora que ya sabemos qué es una API y cuáles son sus partes principales, veamos cómo podemos crear una API en Python.

Cómo crear una API

Existen diferentes formas de crear una API en Python, siendo las más utilizadas FastAPI y Flask. Así pues, explicaré cómo funcionan las dos, de tal forma que puedas usar la forma de crear APIs en Python que más te guste. Empecemos con FastAPI.

Cómo crear una API en Python con FastAPI

Requisitos para usar FastAPI

FastAPI es una forma de crear APIs en Python que surgió a finales de 2018. Es muy rápida, aunque solo puede usarse con Python 3.6+ (en mi opinión esto no debería ser un problema, pero es importante.

Para utilizarlo deberás instalar dos librerías: fastapi y uvicorn.

pip install fastapi

pip install uvicorn

Tu primera API en Python con FastAPI

Ahora que ya hemos instalado los paquetes, simplemente debemos crear un fichero en Python donde definiremos nuestra API. En este fichero, deberemos crear una app, en donde iremos incluyendo las APIs, con sus endpoints, parámetros, etc.

Una vez tenemos la app, es ahí donde definimos la información que requiere la API: endpoint, método HTTP, argumentos de entrada y lo que hará la API por detrás.

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/my-first-api")

def hello():

  return {"Hello world!"}
```

Con esto ya tendríamos una API muy sencilla creada, que simplemente devuelve “Hello world!”. Como verás, en muy pocas líneas hemos definido: el método (get), el endpoint (“/”) y la función que debe correr esta API.

Incluso, podríamos pasar argumentos a nuestra API, para que los utilice en su función. Importante: siempre que pasemos un argumento a nuestra función debemos indicar el tipo de dato que debe ser (número, texto, etc.).

Importante: FastAPI realiza un check de que el tipo de dato que le pasamos en la llamada es el que hemos indicado que debe ser. Esto es algo fundamental para asegurarnos de que nuestra API funciona adecuadamente y, es algo, que otros frameworks de creación de APIs no incluyen.

Comprobar el funcionamiento de una API de FastAPI

Como decía al comienzo de esta sección, para crear una API en FastAPI debemos incluir nuestro código en un fichero de Python, preferiblemente main.py. Asimismo, debemos tener instalado uvicorn. Teniendo en cuenta esto, podemos correr nuestra API de una forma muy sencilla, con el siguiente código:

uvicorn main:app --reload

Esto ejecutará nuestra aplicación y ya podremos acceder a nuestra API, tanto a nivel de navegador como haciendo llamadas desde nuestro ordenador. La API aún no se ha subido a ningún sitio, por lo que será accesible desde el localhost. Puedes acceder al localhost tanto en http://127.0.0.1/ como en http://localhost/, aunque tendrás que hacerlo en el mismo puerto en el que se esté ejecutando la API. Para conocerlo, en mi opinión, lo más fácil es usar el enlace que el propio uvicorn te dará al ejecutar la API:

```
PS C:\Users\Ander\Google Drive\Web\Posts\Como-crear-APIs-en-Python> uvicorn main:app --reload
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [20564] using watchgod
INFO:     Started server process [20204]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

Ilustración 2 comprobación de que funciona la API

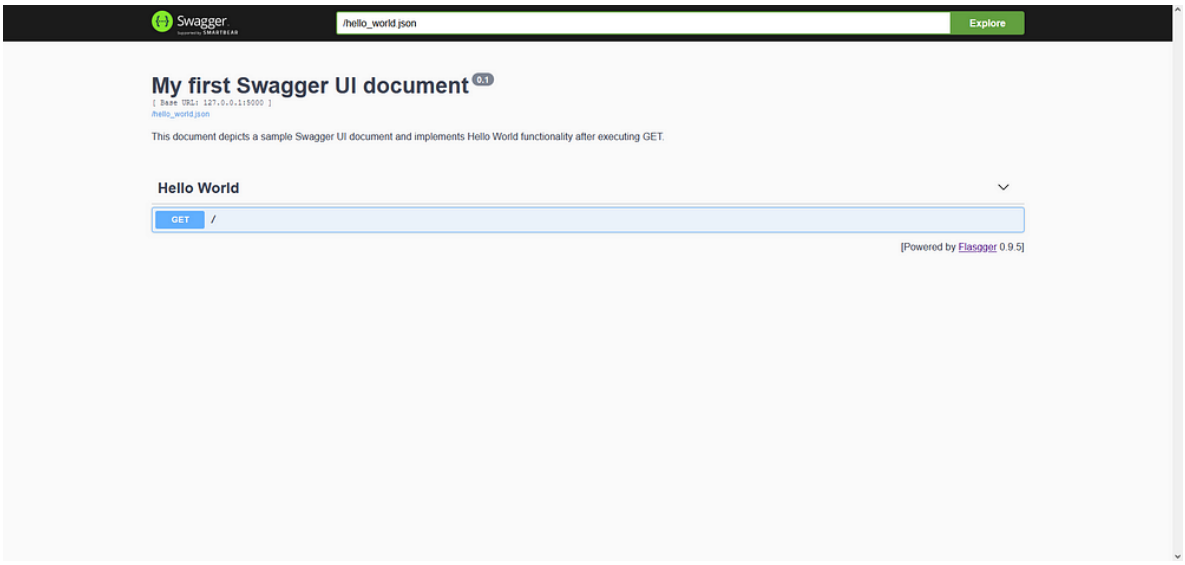

Ilustración 3 Hello world

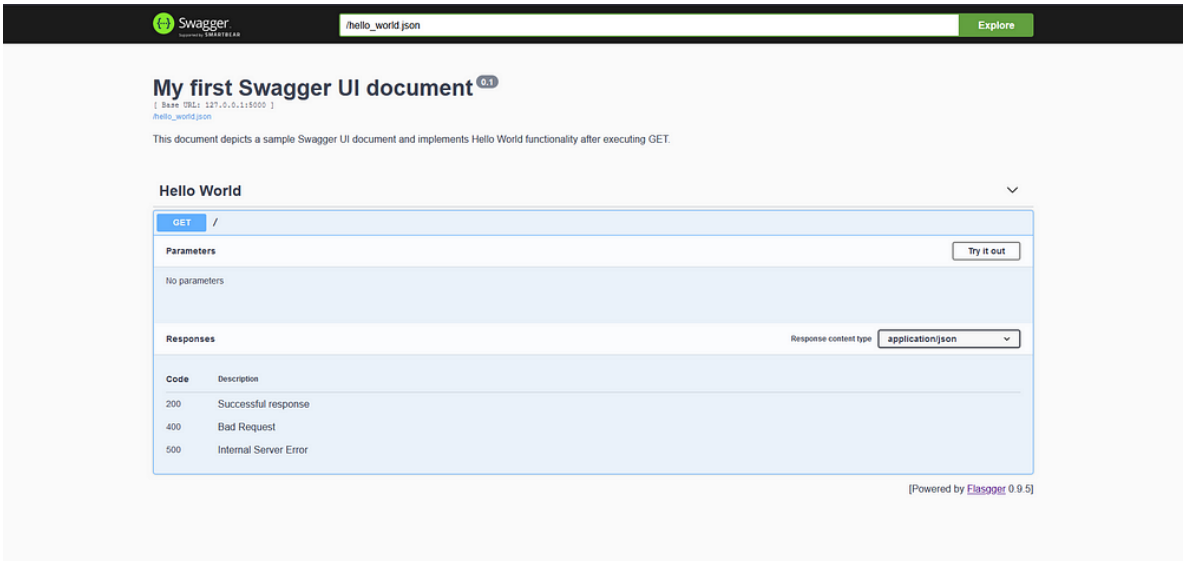

Ilustración 4 Clic en GET

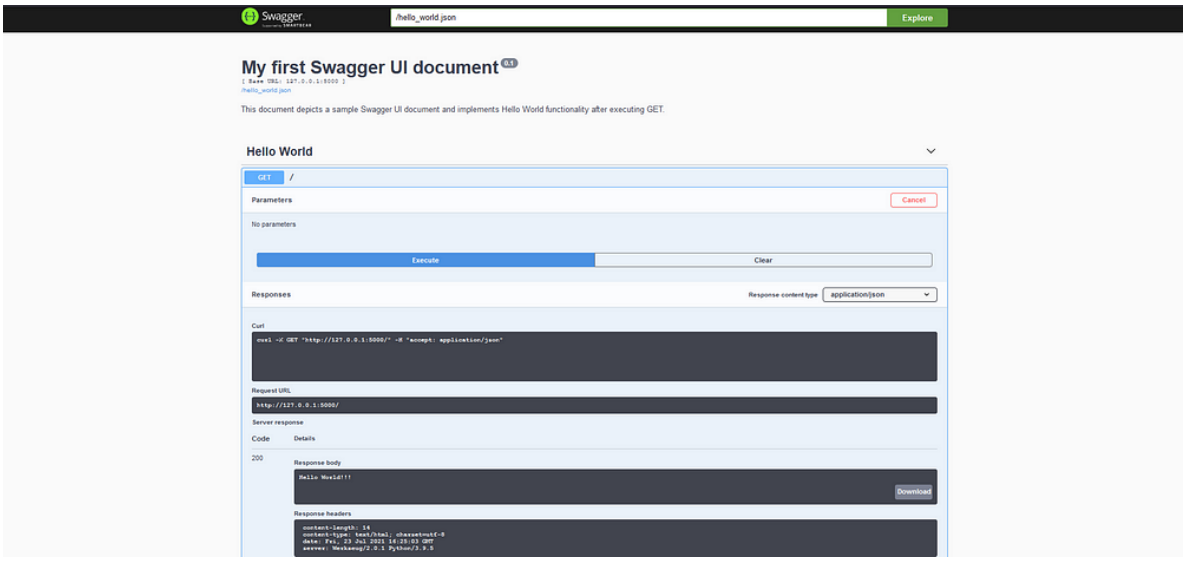

Ilustración 5 Resultado tras dar clic en execute, toda la información obtenida tras el GET

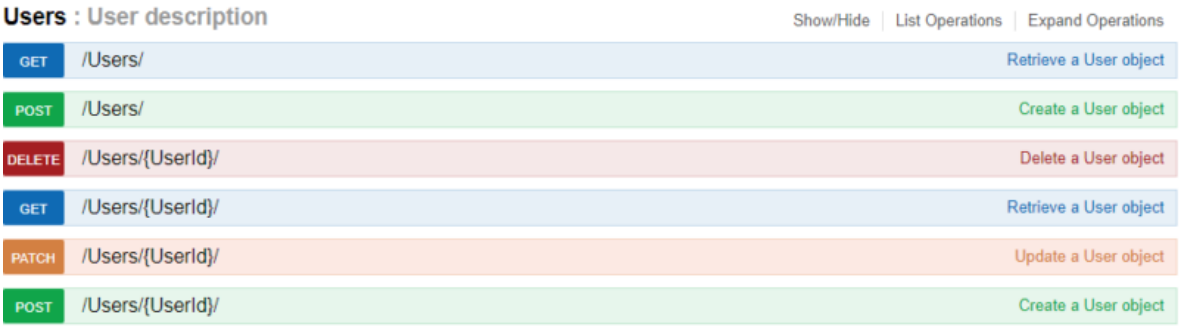

Ilustración 6 Diferentes tipos de rutas en Swagger