UNIVERSIDAD TECNOLÓGICA DE XICOTEPEC DE JUÁREZ

TECNOLOGÍAS DE LA INFORMACIÓN

25 DE JUNIO DE 2023

SWAGGER Y PYTHON

DESARROLLO PARA DISPOSITIVOS INTELIGENTES

EQUIPO: NUTRIMADS

UNIVERSIDAD TECNOLÓGICA DE XICOTEPEC DE JUÁREZ

Arely Aguilar Farias

Sandra Aguilar Santos

Daniel Arroyo Méndez

Mauricio Ramírez López

9A_IDGS

Lo primero es definir los conceptos empecemos con ¿qué es Swagger?:

Swagger es una documentación online que se genera sobre una API. Por lo tanto, en esta herramienta podemos ver todos los endpoint que hemos desarrollado en nuestra API Swagger. Además, nos demuestra cómo son los elementos o datos que debemos pasar para hacer que funcione y nos permite probarlos directamente en su interfaz.

¿Qué es Python?

Python es un lenguaje de programación ampliamente utilizado en las aplicaciones web, el desarrollo de software, la ciencia de datos y el machine learning (ML). Los desarrolladores utilizan Python porque es eficiente y fácil de aprender, además de que se puede ejecutar en muchas plataformas diferentes. El software Python se puede descargar gratis, se integra bien a todos los tipos de sistemas y aumenta la velocidad del desarrollo.

Teniendo en cuenta lo anterior veremos que es lo que necesitamos para implementar Swagger con Python.

Para implementar Swagger con Python y Flask-RESTful-Swagger, necesitarás tener instalado lo siguiente:

- **Python:** Necesitamos tener instalado Python. Se puede descargar desde su sitio web oficial de Python (https://www.python.org) y seguir las instrucciones de instalación adecuadas para tu sistema operativo.
- **pip:** pip es el gestor de paquetes de Python. Por lo general, se instala automáticamente junto con Python a partir de la versión 3.4. Sin embargo, si no lo tienes instalado o necesitas actualizarlo, puedes seguir las instrucciones en la documentación oficial de Python para instalar pip (https://pip.pypa.io/en/stable/installing/).
- **Flask y Flask-RESTful:** Flask es un marco de desarrollo web ligero y Flask-RESTful es una extensión para crear APIs RESTful con Flask. Puedes instalar ambos paquetes utilizando pip con el siguiente comando:
    - pip install flask flask-restful
- **Flask-RESTful-Swagger:** Esta es la biblioteca que permite la integración de Swagger en Flask-RESTful. Puedes instalarlo utilizando pip con el siguiente comando:
    - pip install flask-restful-swagger

Una vez instalado todo lo anterior, ya contaras con todo lo necesario para implementar Swagger en tu aplicación Python utilizando Flask-RESTful-Swagger. Recuerda que también es importante tener un entorno virtual para tu proyecto, lo cual te permite aislar las dependencias y mantener un ambiente limpio.

Queda mas que claro que para implementar Swagger con Python, necesitarás tener una API existente o crear una nueva utilizando un marco de desarrollo web compatible con Flask (como Flask-RESTful). Luego, puedes agregar la documentación Swagger utilizando la biblioteca Flask-RESTful-Swagger.

Entonces ¿qué es un API?:

API significa “interfaz de programación de aplicaciones”. En el contexto de las API, la palabra aplicación se refiere a cualquier software con una función distinta. La interfaz puede considerarse como un contrato de servicio entre dos aplicaciones. Este contrato define cómo se comunican entre sí mediante solicitudes y respuestas. La documentación de su API contiene información sobre cómo los desarrolladores deben estructurar esas solicitudes y respuestas.

Las API son mecanismos que permiten a dos componentes de software comunicarse entre sí mediante un conjunto de definiciones y protocolos. Por ejemplo, el sistema de software del instituto de meteorología contiene datos meteorológicos diarios. La aplicación meteorológica de su teléfono “habla” con este sistema a través de las API y le muestra las actualizaciones meteorológicas diarias en su teléfono.

Ahora que sabemos lo que es una API, veremos 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.
    - **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.

Cómo crear una API en Python:

Existen diferentes formas de crear una API en Python, siendo las más utilizadas FastAPI y Flask, puedes checar como crear el API de las dos formas en el siguiente link: https://anderfernandez.com/blog/como-crear-api-en-python/

Pasos para crear un documento de interfaz de usuario de Swagger.

Seguiremos los siguientes pasos para crear un documento de interfaz de usuario de Swagger para una función de API:

Primero, crearemos la API utilizando el marco de la API web de Flask.

A continuación, crearemos un archivo JSON o YAML para implementar la funcionalidad API en SwaggerUI.

Finalmente, llamaremos al archivo JSON o YAML creado dentro del programa Python donde está presente la definición de la API.

Debe instalar los siguientes paquetes de Python pip install para ejecutar este ejemplo.

$ pip instalar Matraz

$ pip instalar marcador

Si desea actualizar el módulo mientras lo instala, puede usar pip install -U <module_name> o si está usando entornos virtuales de Python y desea instalar una versión específica del módulo para su proyecto, puede ejecutar, pip install <module_name>==<version> por ejemplo,pip install Flask==2.0.0

Pero le recomendaría que debe instalar las últimas versiones. El siguiente es el bloque de importación de nuestro script de Python.

```
de matraz importación Matraz, solicitud
desde flagsgger importar Swagger, LazyString, LazyJSONEncoder
desde flagger importar swag_from
```

Ilustración 1. bloque de importación de nuestro script de Python.

Defina la aplicación Flask usando el método Flask:

aplicación = Frasco (__nombre__)

Necesita el codificador JSON para crear JSON a partir de los objetos API para los que puede usar la clase LazyJSONEncoder.

app.json_encoder = LazyJSONEncoder

La plantilla y la configuración del documento de interfaz de usuario de Swagger se definen como objetos de diccionario de la siguiente manera:

```
swagger_template = dict(
información = {
    'título': LazyString(lambda: 'Mi primer documento de interfaz de
usuario de Swagger'),
    'versión': LazyString(lambda: '0.1'),
    'description': LazyString(lambda: 'Este documento muestra un
documento de muestra de la interfaz de usuario de Swagger e
implementa la funcionalidad Hello World después de ejecutar GET.'),
    },
    host = LazyString(lambda: solicitud.host)
)
arrogancia_config = {
    "encabezados": [],
    "especificaciones": [
        {
            "punto final": 'hola_mundo',
            "ruta": '/hola_mundo.json',
            "rule_filter": regla lambda: Verdadero,
            "model_filter": etiqueta lambda: Verdadero,
        }
    ],
    "static_url_path": "/flasgger_static",
    "swagger_ui": Cierto,
    "specs_route": "/apidocs/"
}
```

Ilustración 2.Objetos de diccionario.

La LazyString funcionalidad del flasgger módulo se utiliza para establecer valores predeterminados para ciertos parámetros durante el tiempo de ejecución. Los parámetros del documento de la interfaz de usuario de Swagger, como el título del documento, la versión, la descripción, etc., se definen dentro del info campo. La URL base para la API se especifica dentro del host campo. Los detalles de configuración de la interfaz de usuario de Swagger se definen dentro del swagger_config objeto JSON.

Finalmente, el objeto swagger se define utilizando el método Swagger importado del flasgger módulo de la siguiente manera:

```
swagger = Swagger(aplicación, plantilla=swagger_template,
config=swagger_config)
```

La función API se define y la ruta/dirección de la aplicación Flask se especifica mediante el @app.route decorador. El archivo YAML para el documento de la interfaz de usuario de Swagger también se llama mediante el @swag_fromdecorador especificando la ruta del archivo YAML y GET el método como argumentos (argumentos como POST, PUT, DELETE también se pueden usar según los requisitos).

```
@swag_from("hola_mundo.yml", métodos=['GET'])
@app.ruta("/")
def hola_mundo():
volver "¡¡¡Hola mundo!!!"
```

Las siguientes líneas deben agregarse dentro del hello_world.yml archivo para implementar la GET respuesta en la interfaz de usuario de Swagger:

```
openapi: 3.0.0
etiquetas:
  - nombre: Hola mundo
obtener:
  descripción: Ninguno
respuestas:
    '200':
      descripción: Respuesta exitosa
    '400':
      descripción: Solicitud incorrecta
    '500':
      descripción: Error interno del servidor
```

```
si __nombre__ == '__principal__':
    aplicación.ejecutar()
```

Ilustración 4. Código para ejecutar la aplicación de Flask.

Cuando se ejecute el código, se ejecutará localhost en modo de desarrollo y se mostrará el siguiente documento cuando se ejecute 127.0.0.1:5000/apidocs:

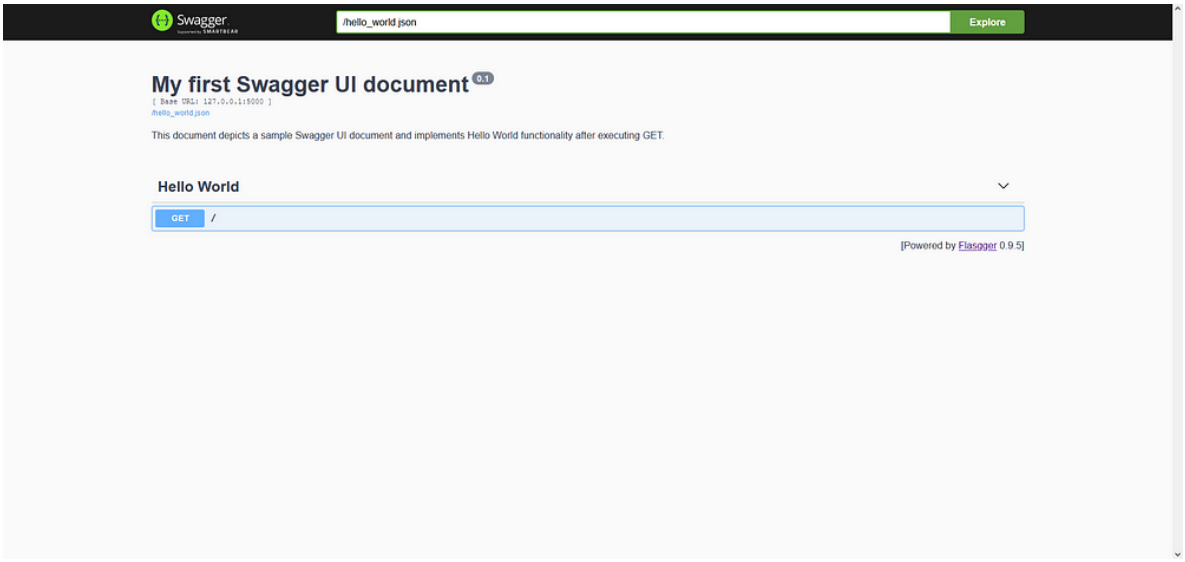

Ilustración 3.Documento obtenido después de ejecutar 127.0.0.1:5000/apidocs en el navegador.

Como puede ver, este documento contiene el título, la versión y la descripción de la API tal como se define en el archivo YAML. Una vez que se selecciona la respuesta GET en el documento de la interfaz de usuario de Swagger, el documento se expande como:

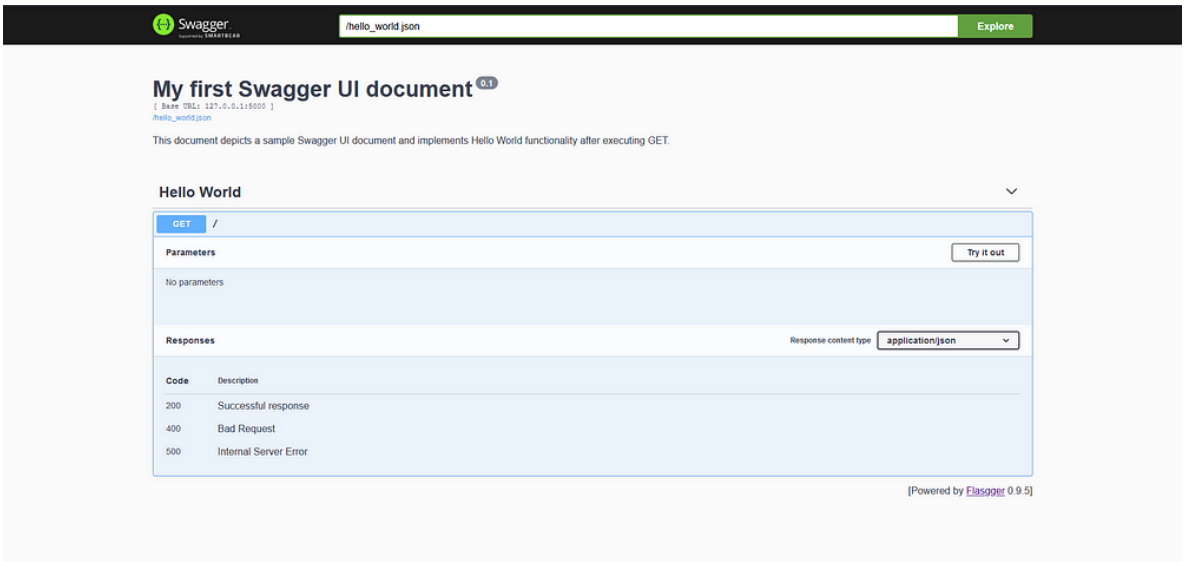

Ilustración 5. Resultado obtenido tras seleccionar GET respuesta del documento.

Cuando hace clic en el botón "Pruébelo", el documento muestra el botón "Ejecutar" de la siguiente manera:

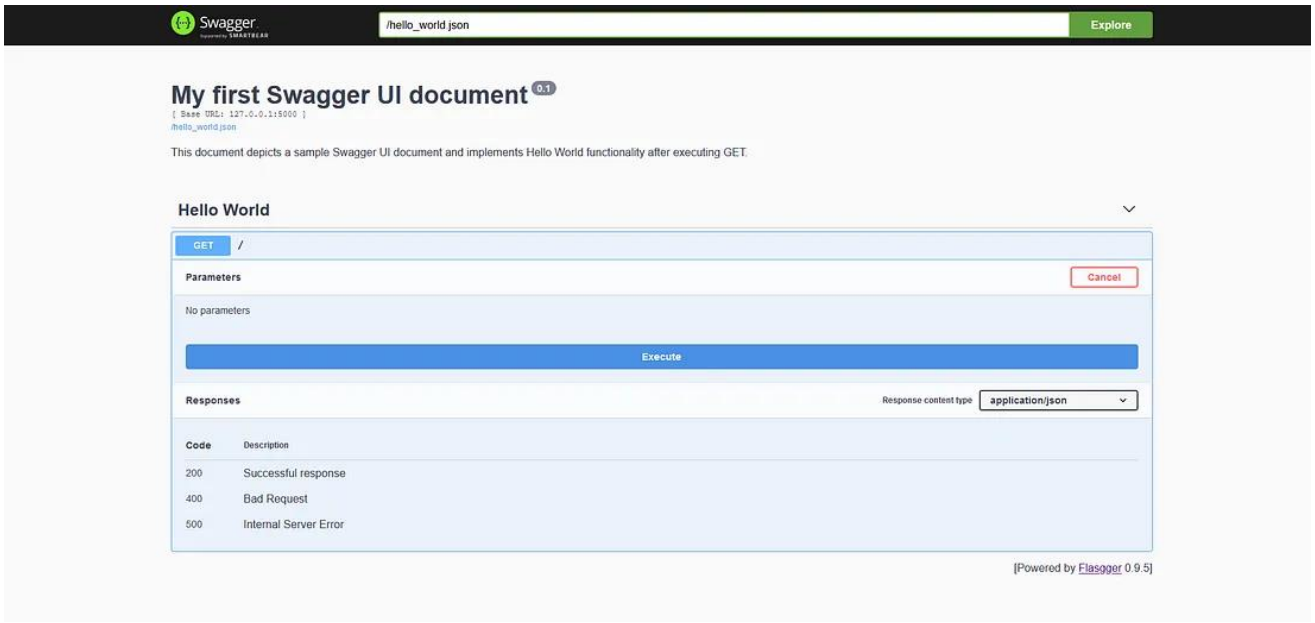

Ilustración 6. Resultado obtenido tras seleccionar el botón "Execute".

Una vez que se selecciona la opción "Execute" Hello World , la salida se muestra como se muestra a continuación:

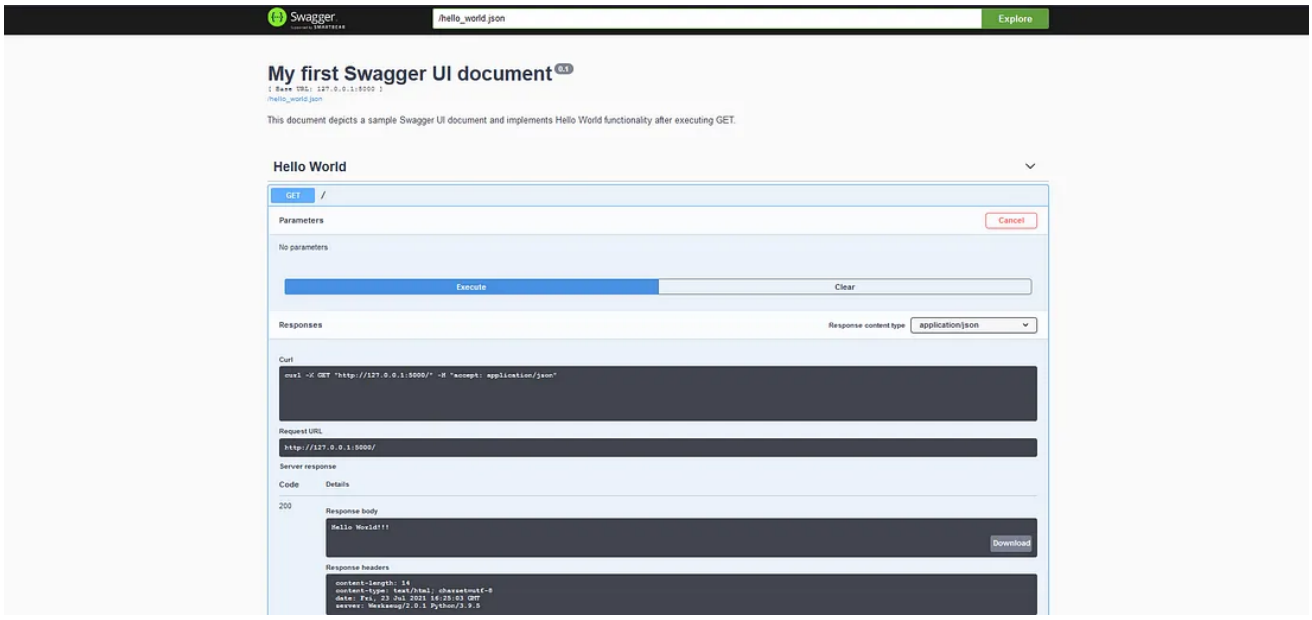

Ilustración 7.Resultado después de ejecutar la respuesta GET seleccionando la opción "Execute".

Como se ve, la respuesta de la API se devuelve dentro "Response body" del método GET. Contienen "Response headers" información básica de la respuesta, como la duración de la respuesta, el tipo de respuesta, la fecha y la hora en que se recibió la respuesta.