Universidad Tecnologica de Xicotepec de Juárez

Ingeniería en Desarrollo y Gestión de Software

Materia: Extracción de Conocimiento en Bases de Datos

Implementación y configuración de Swagger en Python

Integrantes:

Fabiola Bautista Joaquín M-200325

Gabriel Guzmán García M-200448

Uriel Maldonado Cortéz M-200931

Mateo Maldonado Tolentino M-200906

Docente: MTI. Marco Antonio Ramírez

9° “A”

Enero-Abril 2023

Implementación y configuración de Swagger en Python

Introducción:

Swagger es una herramienta que ayuda a documentar y consumir API. Se puede usar en proyectos de Python para crear documentación detallada y accesible desde el código fuente de la API. Swagger ofrece una interfaz interactiva llamada Swagger UI, la cual permite a los desarrolladores probar y explorar la API directamente desde el navegador.

Swagger proporciona beneficios los cuales puede ahorrar tiempo y esfuerzo al generar documentación automáticamente, también puede ayudar a mejorar la colaboración entre los equipos de desarrollo y los consumidores de API. La interfaz de usuario de Swagger facilita la comprensión de la estructura de la API, los parámetros de entrada, los métodos disponibles y las respuestas esperadas. Swagger es una poderosa herramienta que puede ayudar a mejorar el desarrollo y consumo de API. Es fácil de usar y ofrece una serie de ventajas, como la generación automática de documentación y una interfaz interactiva.

I. ¿Qué es Swagger?

Python es un lenguaje de programación de código abierto desarrollado por Guido van Rossum en 1991. Es un lenguaje orientado a objetos que se destaca por su facilidad de interpretación y una sintaxis que se asemeja al lenguaje inglés, lo que facilita su lectura y comprensión. A diferencia de otros lenguajes, Python es interpretado, lo que significa que el código fuente se traduce en byte-code y se ejecuta a través de la máquina virtual de Python.

2. ¿Qué es Swagger?

Swagger es una herramienta que ayuda a documentar, visualizar y consumir API RESTful. La cual, proporciona un conjunto de reglas, especificaciones y herramientas que permiten a los desarrolladores crear documentación clara y concisa para sus API. Ofrece una interfaz interactiva llamada Swagger UI, esta facilita la comprensión de la documentación y la prueba de la API.

Una de las principales ventajas de Swagger es que es ampliamente entendido tanto por desarrolladores como por no desarrolladores. Esto se debe a que la interfaz de usuario de Swagger es fácil de usar y entender, incluso si no tiene experiencia en programación. Además, Swagger se puede integrar fácilmente con otras herramientas, como WSO2, lo que lo hace aún más potente.

En general, Swagger es una herramienta valiosa para cualquiera que desarrolle o consuma API RESTful. Proporciona una forma clara y concisa de documentar las API, y facilita su comprensión y prueba.

Estos son algunos beneficios adicionales de Swagger:

- Documentación automatizada: Swagger puede generar automáticamente documentación a partir del código fuente de su API. Esto le ahorra tiempo y esfuerzo, y garantiza que su documentación esté siempre actualizada.
- Interfaz interactiva: La interfaz de usuario de Swagger proporciona una interfaz interactiva que facilita la exploración de la API y la prueba de los puntos finales. Esto puede ayudarle a identificar cualquier problema potencial con su API antes de lanzarla a producción.
- Ampliamente apoyado: Swagger es ampliamente compatible con una variedad de herramientas y marcos. Esto facilita la integración de Swagger en su entorno de desarrollo existente.

III. Preparación del entorno de desarrollo:

Instalar Python: se descarga e instala la versión adecuada para el sistema operativo desde el sitio web oficial de Python (https://www.python.org/).

Crear un entorno virtual (opcional): Se recomienda crear un entorno virtual para mantener separadas las dependencias del proyecto. Se usa la herramienta virtualenv para crear un entorno virtual en tu directorio de proyecto. Ejecuta el siguiente comando en la terminal:

- **python -m venv myenv**

Activar el entorno virtual: En la terminal, debes activar el entorno virtual recién creado.

Instalar dependencias: Para utilizar Swagger con Python, se necesita instalar las siguientes dependencias:

FastAPI: Un framework web rápido para crear APIs en Python.

- **pip install fastapi**

Uvicorn: Un servidor ASGI para ejecutar la aplicación FastAPI.

- **pip install uvicorn**

Swagger UI: Una interfaz de usuario para visualizar y probar la documentación generada por Swagger.

- **Pip install swagger-ui**
- **pip install fastapi[all]**

Pydantic: Una biblioteca para la validación de datos y generación de esquemas.

- **pip install pydantic**

IV. Integración de Swagger en un proyecto Python:

1. Importar los módulos necesarios: En el archivo Python principal, generalmente llamado "main.py" o similar, se debe importar los módulos necesarios para utilizar Swagger y FastAPI. Asegurarse de tener las siguientes importaciones:
2. Configurar la aplicación FastAPI: Crear una instancia de la clase FastAPI y configúrala según las necesidades. Se pueden agregar rutas, definir modelos de datos, etc.
3. Configurar Swagger en la aplicación
4. Ejecutar la aplicación: Asegurarse de estar en el directorio raíz del proyecto y ejecutar el siguiente comando para iniciar el servidor Uvicorn:

- **uvicorn main:app –reload**

5. Acceder a la documentación Swagger: Abrir el navegador web y visitar la siguiente URL http://127.0.0.1:8000/docs.

V. Configuración de Swagger:

- Importar la biblioteca. Esta biblioteca proporciona las clases y funciones que necesita para configurar Swagger en Python
- Crear un objeto. Este objeto representa la configuración de la API
- Establecer la propiedad del objeto. Esta propiedad especifica el título, la descripción y la versión de la API.
- Guardar el objeto en un archivo. Este archivo se puede abrir en un visor JSON para ver la configuración de su API
- Ejecutar la interfaz de usuario de Swagger. Esto abrirá la interfaz de usuario de Swagger en su navegador. A continuación, puede usar la interfaz de usuario de Swagger para explorar la API y probar los puntos de conexión.

```python
import swagger

swagger_object = swagger.Swagger()

swagger_object.info.title = "My API"
swagger_object.info.description = "This is my API"
swagger_object.info.version = "1.0.0"

swagger_object.paths.add_path("/hello", swagger.PathItem(
    get=swagger.Operation(
        summary="Returns a hello message",
        responses={200: swagger.Response(
            description="A hello message",
            content={"application/json": swagger.MediaType(
                schema=swagger.Schema(type="string"))}
        )}
    )
))

with open("swagger.json", "w") as f:
    f.write(swagger_object.json())
```

Ilustración 1 Ejemplo de configuración Swagger

V. Generación de la documentación de la API:

En el caso de Python, Swagger puede ser utilizado para generar la documentación de una API. Existen varias bibliotecas disponibles que pueden ayudar en este proceso:

swagger-ui: Permite generar una interfaz de usuario basada en web para explorar y probar una API. Puede utilizarse para visualizar de manera interactiva la documentación generada por Swagger.

swagger-codegen: Permite generar bibliotecas de cliente en diferentes lenguajes de programación a partir de la definición de Swagger. Estas bibliotecas facilitan la interacción con la API desde el lado del cliente.

swagger-spec: es la biblioteca principal de Swagger y proporciona la especificación central de Swagger. Esta especificación puede ser utilizada para generar documentación en otros formatos, más allá del formato JSON original.

Para generar la documentación de una API con Swagger en Python, se deben seguir estos pasos:

Crear un archivo de definición de Swagger que contenga la información relevante de la API, como los puntos finales, los parámetros y las respuestas esperadas.

Utilizar una biblioteca de Swagger que se ajuste a las necesidades del proyecto para generar la documentación a partir de la definición.

Proporcionar la documentación generada a los usuarios para que puedan consultarla y comprender el funcionamiento de la API.

```yaml
swagger: '2.0'
info:
  title: My API
  description: This is my API
  version: 1.0.0
paths:
  /hello:
    get:
      summary: Returns a hello message
      responses:
        200:
          description: A hello message
          content:
            application/json:
              schema:
                type: string
```

Ilustración 2 Ejemplo de un archivo de definición de Swagger

Referencias

(https://swagger.io/docs/, s.f.)

(https://fastapi.tiangolo.com/, s.f.)

(https://swagger.io/blog/api-development/automatically-generating-swagger-specifications-wi/, s.f.)

(https://pypi.org/project/swagger-ui-py/, s.f.)

(https://swagger.io/docs/open-source-tools/swagger-ui/usage/configuration/, s.f.)

(https://swagger.io/docs/open-source-tools/swagger-ui/usage/installation/, s.f.)