**Instalación de Swagger**

Swagger tiene varias herramientas y bibliotecas disponibles. Una de las opciones más comunes es Swagger UI para la documentación y Swagger Editor para diseñar la especificación de la API. Puedes instalar Swagger UI de la siguiente manera utilizando npm (Node Package Manager):

npm install -g swagger-ui-express

* + También puedes utilizar Swagger Editor, que es una herramienta en línea, accediendo a [Swagger Editor](https://editor.swagger.io/).

1. **Crear una especificación OpenAPI:**
   * Swagger utiliza la especificación OpenAPI (anteriormente conocida como Swagger Specification) para describir las API. Puedes crear un archivo YAML o JSON que describa tu API de acuerdo con la especificación de OpenAPI. A continuación, se muestra un ejemplo mínimo de una especificación OpenAPI en formato YAML:

openapi: 3.0.0

info:

title: Mi API

version: 1.0.0

paths:

/saludo:

get:

summary: Obtiene un saludo

responses:

'200':

description: Respuesta exitosa

* + Esta especificación describe una API con una ruta **/saludo** que admite un método GET.

1. **Configurar Swagger UI:**
   * Si estás utilizando Express.js para tu aplicación web, puedes configurar Swagger UI fácilmente. Aquí tienes un ejemplo de cómo hacerlo:

const express = require('express');

const swaggerUi = require('swagger-ui-express');

const swaggerDocument = require('./swagger.json'); // Reemplaza con la ubicación de tu archivo YAML/JSON

const app = express();

app.use('/api-docs', swaggerUi.serve, swaggerUi.setup(swaggerDocument));

// Resto de la configuración de tu aplicación Express

app.listen(3000, () => {

console.log('La aplicación está escuchando en el puerto 3000');

});

1. **Acceder a Swagger UI:**
   * Una vez que hayas configurado Swagger UI, puedes acceder a la documentación de tu API a través de tu navegador web navegando a **http://localhost:3000/api-docs** (si has configurado tu servidor para escuchar en el puerto 3000, ajusta el puerto según tu configuración).
2. **Documentar tu API:**
   * En Swagger UI, puedes comenzar a documentar tu API, agregar descripciones, especificar parámetros, definir esquemas de respuestas y más utilizando la interfaz de Swagger UI.
3. **Probar tu API:**
   * Además de la documentación, Swagger UI permite probar tu API directamente desde la interfaz. Puedes realizar solicitudes HTTP y ver las respuestas correspondientes.
4. **Guardar y compartir tu especificación:**
   * Una vez que hayas completado la documentación de tu API, asegúrate de guardar la especificación de OpenAPI (ya sea en formato YAML o JSON). Puedes compartir este archivo con tu equipo de desarrollo y otros interesados.

Swagger es una herramienta poderosa que facilita la documentación y el uso de tus API. Puedes personalizar aún más la documentación y aprovechar las características avanzadas de Swagger para documentar API más complejas. A medida que evoluciona tu API, actualiza la especificación de OpenAPI para mantener la documentación al día.

En mi ejemplo cree un archivo json que ayuda a swagger a crear el reporte:

const express = require('express');

const swaggerUi = require('swagger-ui-express');

const axios = require('axios');

const { exec } = require('child\_process');

const app = express();

const PORT = process.env.PORT || 3000;

const swaggerJsonURL = '/openapi.json';

// Configuración de Swagger UI

app.use('/playwright-report', swaggerUi.serve, swaggerUi.setup(null, {

  swaggerUrl: swaggerJsonURL, // Reemplaza con la URL de tu especificación Swagger JSON

}));

// Configuración de las pruebas de carga

app.get('/run-load-test', (req, res) => {

  // Ejecuta las pruebas de carga usando Artillery (o la herramienta de tu elección)

exec('artillery-config.yaml', (error, stdout, stderr) => {

    if (error) {

    console.error(`Error al ejecutar las pruebas de carga: ${error.message}`);

    res.status(500).send('Error al ejecutar las pruebas de carga.');

    return;

    }

    // Envía la salida de las pruebas de carga como respuesta

    console.log(`Salida de las pruebas de carga: ${stdout}`);

    res.send('Pruebas de carga completadas.');

});

});

// Inicia el servidor Express

app.listen(PORT, () => {

console.log(`Servidor Express en ejecución en el puerto ${PORT}`);

});

Tuve que instalar:

npm install axios y ejecutarlo con node report.js