¿Qué es Swagger (y OpenAPI)? 📝

Swagger es un conjunto de herramientas de código abierto que se basan en la **Especificación OpenAPI (OAS)**. Su propósito principal es ayudarte a diseñar, construir, documentar y consumir servicios web RESTful. Imagina que es un lenguaje estándar que tanto humanos como computadoras pueden entender para describir las capacidades de una API sin necesidad de acceder al código fuente o a la documentación manual.

Los componentes clave que usualmente usamos con Node.js son:

- **Swagger-UI**: Una interfaz web interactiva que toma la especificación de tu API (escrita en OpenAPI) y la convierte en una documentación legible y fácil de usar. Puedes ver los endpoints, entender los parámetros y hasta hacer llamadas de prueba directamente desde el navegador.
- **Swagger-JS**: Una librería de JavaScript que puede parsear y mostrar documentos OpenAPI.
- **Swagger-Editor**: Un editor basado en navegador para escribir y validar las definiciones de la Especificación OpenAPI.

¿Por qué usar Swagger con Node.js REST? 🤔

Integrar Swagger/OpenAPI con tus APIs REST de Node.js ofrece varias ventajas importantes:

- **Documentación Interactiva**: Proporciona una interfaz web amigable donde los desarrolladores pueden explorar tus endpoints, entender los parámetros y realizar llamadas de prueba directamente desde el navegador. Esto elimina la necesidad de herramientas de documentación externas o de comunicación constante entre equipos de frontend y backend.
- **Diseño y Colaboración de API**: Funciona como una fuente única de verdad para el diseño de tu API, lo que facilita una mejor colaboración entre los miembros del equipo. Los cambios en la API se reflejan inmediatamente en la documentación.
- **Generación de Código**: Las herramientas pueden usar la Especificación OpenAPI para generar automáticamente SDKs de cliente (para varios lenguajes de programación) y stubs de servidor, ahorrando tiempo de desarrollo y reduciendo errores.

- **Pruebas**: Simplifica las pruebas de API al permitir a los desarrolladores realizar fácilmente solicitudes e inspeccionar las respuestas directamente desde la interfaz de Swagger UI.
- **Validación**: Asegura que las solicitudes y respuestas de la API cumplan con la especificación definida, mejorando la integridad y consistencia de los datos.

¿Cómo implementar Swagger en una API REST de Node.js? 🛠️

Implementar Swagger en una API REST de Node.js generalmente sigue estos pasos:

1. **Define tu Esquema de API**: Crea un archivo de Especificación OpenAPI (usualmente en formato YAML o JSON) que describa los endpoints de tu API, los parámetros, los cuerpos de solicitud, las respuestas y los modelos de datos.
2. **Integra Swagger UI**: Usa una librería para servir la interfaz de Swagger UI junto con tu aplicación Node.js, apuntándola a tu archivo de Especificación OpenAPI.
3. **Genera Documentación (Opcional pero Recomendado)**: Utiliza herramientas que puedan generar o actualizar automáticamente la Especificación OpenAPI basándose en tu código (por ejemplo, comentarios JSDoc).

Aquí te muestro un enfoque común usando el framework express y librerías populares de Swagger:

1. Instala los paquetes necesarios

Necesitarás express para tu API y swagger-ui-express junto con una herramienta para generar la especificación, como swagger-jsdoc.

Bash

npm install express swagger-ui-express swagger-jsdoc

2. Configura Swagger en tu aplicación Node.js

Crearás una configuración para swagger-jsdoc y luego usarás swagger-ui-express para servir la documentación generada.

JavaScript

```
// app.js (o tu archivo principal de la aplicación)
const express = require('express');
```

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

const app = express();
const port = 3000;

// Definición de Swagger
const options = {
 definition: {
  openapi: '3.0.0', // Versión de OpenAPI
  info: {
   title: 'Mi API REST de Node.js', // Título de tu API
   version: '1.0.0', // Versión de tu API
   description: 'Una API de ejemplo para propósitos de demostración', //
Descripción
  },
  servers: [
   {
    url: `http://localhost:${port}` , // URL base de tu API
    description: 'Servidor de desarrollo',
   },
  ],
 },
 // Rutas a los archivos que contienen los comentarios JSDoc
 // Estos comentarios se usarán para generar la especificación OpenAPI
 apis: ['./routes/*.js'], // Por ejemplo, si tus rutas están en la carpeta 'routes'
};

const specs = swaggerJsdoc(options);

// Sirve la interfaz de Swagger UI en la ruta /api-docs
app.use('/api-docs', swaggerUi.serve, swaggerUi.setup(specs));

// Ruta de API básica (ejemplo)
app.get('/', (req, res) => {
 res.send('¡Hola Mundo!');
});

// Inicia el servidor
app.listen(port, () => {
 console.log(`Servidor escuchando en http://localhost:${port}`);
 console.log(`Swagger UI disponible en http://localhost:${port}/api-docs`);
});
```

3. Agrega comentarios JSDoc a tus rutas (ejemplo en ./routes/users.js)

Para que swagger-jsdoc pueda generar la especificación, debes añadir comentarios JSDoc especiales a tus rutas.

JavaScript

```
// routes/users.js
const express = require('express');
const router = express.Router();

/**
 * @swagger
 * tags:
 * name: Usuarios
 * description: Operaciones de usuarios
 */

/**
 * @swagger
 * /users:
 * get:
 * summary: Obtiene todos los usuarios
 * tags: [Usuarios]
 * responses:
 * 200:
 * description: Lista de usuarios
 * content:
 * application/json:
 * schema:
 * type: array
 * items:
 * type: object
 * properties:
 * id:
 * type: integer
 * description: ID del usuario
 * name:
 * type: string
 * description: Nombre del usuario
 * 500:
 * description: Error del servidor
 */
router.get('/users', (req, res) => {
  // Lógica para obtener usuarios
  res.json([{ id: 1, name: 'Alice' }, { id: 2, name: 'Bob' }]);
});
```

```
/**
 * @swagger
 * /users/{id}:
 * get:
 * summary: Obtiene un usuario por ID
 * tags: [Usuarios]
 * parameters:
 * - in: path
 * name: id
 * required: true
 * schema:
 * type: integer
 * description: ID del usuario a obtener
 * responses:
 * 200:
 * description: Información del usuario
 * content:
 * application/json:
 * schema:
 * type: object
 * properties:
 * id:
 * type: integer
 * name:
 * type: string
 * 404:
 * description: Usuario no encontrado
 * 500:
 * description: Error del servidor
 */
router.get('/users/:id', (req, res) => {
  const userId = parseInt(req.params.id);
  // Lógica para obtener un usuario por ID
  if (userId === 1) {
    res.json({ id: 1, name: 'Alice' });
  } else {
    res.status(404).send('Usuario no encontrado');
  }
});

module.exports = router;
```

Recuerda: Deberás incluir estas rutas en tu archivo app.js:

JavaScript

```
// app.js
// ... (código anterior)
```

```
const usersRouter = require('./routes/users');
app.use('/', usersRouter); // O el prefijo que desees, por ejemplo, '/api'

// ... (resto del código)
```

Después de iniciar tu servidor Node.js (node app.js), podrás acceder a la documentación interactiva de tu API en http://localhost:3000/api-docs. ¡Ahí verás cómo Swagger te permite explorar y probar tus endpoints fácilmente!