UNIVERSIDAD TECNOLOGICA DE XICOTEPEC DE JUAREZ

Investigación Swagger

DDI y ECBD

INTEGRANTES:

JOSSELINE ALVARADO VARGAS M_200687

CRYSTIAN ENRIQUE SUAREZ CUEVAS M_200527

Swagger

Swagger es de código abierto que fue creado por Tony Tam en 2011 y ha sido adoptado por muchas empresas líderes en tecnología como Google, Microsoft y IBM.

La herramienta principal de Swagger es el lenguaje de descripción de la API (API Description Language o ADL), que permite a los desarrolladores describir su API en un formato legible por máquina. Esto significa que la información sobre la API se puede compartir fácilmente entre diferentes herramientas y plataformas.

Además del lenguaje ADL, Swagger también incluye una serie de herramientas adicionales para ayudar a los desarrolladores a crear documentación efectiva. Estas herramientas incluyen:

- Swagger UI: Una interfaz web interactiva que muestra la documentación generada automáticamente a partir del lenguaje ADL.
- Swagger Editor: Un editor visual para crear y editar archivos ADL.
- Swagger Codegen: Una herramienta que genera código cliente o servidor a partir del archivo ADL.

¿Por qué usar Swagger?

Hay varias razones por las cuales los desarrolladores pueden optar por utilizar Swagger para documentar sus API:

Consistencia: Al utilizar un formato estandarizado como el lenguaje ADL, se asegura que la documentación sea coherente en toda la organización.

Compartibilidad: La información sobre la API se puede compartir fácilmente entre diferentes equipos y plataformas.

Automatización: Al utilizar herramientas como Swagger UI y Swagger Codegen, se puede generar documentación y código automáticamente, lo que ahorra tiempo y reduce errores.

Claridad: La documentación generada por Swagger es clara y fácil de entender, lo que facilita la adopción de la API por parte de los desarrolladores.

Entorno

Sistema Operativo:	Linux Mint
Hardware:	Laptop Toshiba Satellite AMD E1
Entorno de desarrollo:	Visual Studio Code

Cómo utilizar Swagger

Utilizaremos un Api rest con Mongo sencillo, para este ejemplo está cargado en GitHub ya realizado con Swagger como srzzuares/Practica_ApiNode

A continuación, ya creada la carpeta con los archivos esenciales donde tengamos la Api, abriremos la terminal de visual studio code para instalar dependencias para nuestra documentación Swwagger.

```
npm i swagger-jsdoc swagger-ui-express
```

Después de la instalación haya sido completa, requerimos los siguientes paquetes de Swagger y el path para saber la raíz de nuestro proyecto:

```
//Path
const path = require('path') //Raiz del proyecto
//Swagger
const swaggerUI = require('swagger-ui-express') // Interfaz del usuario de Swagger
const swaggerJsDoc = require('swagger-jsdoc') // Documentacion de swagger
```

Crearemos un objeto para darle especificaciones de swagger y en el código se describe cada cosa que se utiliza en el objeto:

```
//Object Swagger
const swaggerSpec = {
    definition: { //Definimos el objeto
        openapi: "3.0.0", //Que version de swagger utilizaremos
        info: { //La informacion del objeto
            title: "Api-Node JS, Mongo y Swagger", //Titulo del proyecto
            version: "1.0.0" //Version del proyecto
        },
        servers: [ //Servicios
            {
                url: "http://localhost:3030" //Servicio donde inicializamos el proyecto NodeJs
            }
        ]
    },
    apis: [ //Apis que creamos
        `${path.join(__dirname, './routes/obras.Routes.js')}` //Solo tenemos una llamada obras.Routes, asi que con el path podr
    ]
}
//Midleware
APP.use('/api-doc', swaggerUI.serve, swaggerUI.setup(swaggerJsDoc(swaggerSpec))) //creamos un middleware para el servicio de sw
/* Para esto primero es la ruta de acces, segundo inicializamos el servidor, tercero inicializamos el UI.setup y
dentro inicializamos el jsDoc y por ultimo el objeto ya especificado   */
```

Dentro de las Rutas creamos la documentación:

```
const { Router } = require('express')
const rooteable = Router()
const OCLL = require('../controllers/obrasControllers')

/**
 *  @swagger
 *  components:
 *      schemas:
 *       User:
 *        type: object
 *        properties:
 *         idObra:
 *          type: String
 *          description: Un identificador
 *         nombreObra:
 *          type: String
 *          description: Nombre de la obra
 *         tamanoObra:
 *          type: String
 *          description: Medidas de la obra
 *         tecnicaObra:
 *          type: String
 *          description: Tipo de tecnica
 *         materialesObra:
 *          type: String
 *          description: materiales que contiene la obra
 *         valorEconomicoObra:
 *          type: Number
 *          description: Costo de la obra o a la venta
 *         nombreAutor:
 *          type: String
 *          description: Nombre del autor de la obra
 *         telefonoAutor:
 *          type: Integer
 *          description: Numero telefonico
 *        telefonoAutor:
 *         type: Integer
 *         description: Numero telefonico
 *        correoAutor:
 *         type: String
 *         description: Correo del autor
 *       required:
 *        - idObra
 *        - nombreObra
 *        - nombreAutor
 *        - correoAutor
 *       example:
 *        idObra: 1
 *        nombreObra: La Soledad
 *        tamanoObra: Mediano 40x80
 *        tecnicaObra: Oleo con acrilico
 *        materialesObra: Pintura, Lienzo
 *        valorEconomicoObra: 1000
 *        nombreAutor: Crystian
 *        telefonoAutor: 7641112222
 *        correoAutor: crys@gmail.com
 */


/**
 *  @swagger
 *  /:
 *  get:
 *      summary: Retorna todas las obras creadas
 *      tags: [User]
 *      responses:
 *          200:
 *              description: Todas las obras creadas
 *              content:
 *                  application/json:
```

```
 *                     schema:
 *                         type: array
 *                         items:
 *                             $ref: '#/components/schemas/User'
 */
rooteable.get('/', OCLL.obtenerObra)

/**
 *  @swagger
 *  /obtAct/{idOb}:
 *  get:
 *      summary: Retorna una id de Obra
 *      tags: [User]
 *      parameters:
 *          - in: path
 *            name: idOb
 *            required: true
 *            schema:
 *                type: string
 *            description: Identificacion de la obra
 *      responses:
 *          200:
 *              description: Todas las obras creadas
 *              content:
 *                  application/json:
 *                      schema:
 *                          type: object
 *                          $ref: '#/components/schemas/User'
 *          404:
 *              description: Identificacion no encontrada
 */
rooteable.get('/obtAct/:idOb', OCLL.obtenerOneObra)
/**
 *  @swagger
 *  /crear:
 *      post:
 *          summary: Crear una obra de arte
 *          tags: [User]
 *          requestBody:
 *              required: true
 *              content:
 *                  application/json:
 *                      schema:
 *                          type: object
 *                          $ref: '#/components/schemas/User'
 *          responses:
 *              200:
 *                  description: Success created
 */
rooteable.post('/crear', OCLL.guardarObras)

/**
 *  @swagger
 *  /actualizar/{idOb}:
 *  put:
 *      summary: actualizar una obra
 *      tags: [User]
 *      parameters:
 *          - in: path
 *            name: idOb
 *            required: true
 *            schema:
 *                type: string
 *            description: Identificacion de la obra
 *      requestBody:
 *              required: true
 *              content:
```

```
 *                 application/json:
 *                     schema:
 *                         type: object
 *                         $ref: '#/components/schemas/User'
 *     responses:
 *         200:
 *             description: Obra actualizada
 *             content:
 *                 application/json:
 *                     schema:
 *                         type: object
 *                         $ref: '#/components/schemas/User'
 *         404:
 *             description: Identificacion no encontrada
 */
rooteable.put('/actualizar/:idOb', OCLL.actualizaObra)

/**
 *  @swagger
 *  /eliminar/{idOb}:
 *  delete:
 *      summary: Elimina una obra
 *      tags: [User]
 *      parameters:
 *          - in: path
 *            name: idOb
 *            required: true
 *            schema:
 *                type: string
 *            description: Identificacion de la obra
 *      responses:
 *          200:
 *              description: Obra Eliminada
 *              content:
 *                  application/json:
 *                      schema:
 *                          type: object
 *                          $ref: '#/components/schemas/User'
 *          400:
 *              description: Identificacion no encontrada
 */
rooteable.delete('/eliminar/:idOb', OCLL.eliminaObra)
```

Conclusión

Swagger es una herramienta poderosa para describir, diseñar y documentar las API. Al utilizar un formato estandarizado como el lenguaje ADL, se asegura que la documentación sea coherente en toda la organización y se puede compartir fácilmente entre diferentes equipos y plataformas.

Además, las herramientas adicionales como Swagger UI y Swagger Codegen permiten generar automáticamente documentación y código cliente o servidor a partir del archivo ADL. En resumen, si deseas crear una documentación efectiva para tu API, considera utilizar Swagger.

GitHub Srzzuares srzzuares/Practica ApiNode Completado