

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.

Compatibilidad: 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 Swagger.

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

Img1.- Instalación de paquetes

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

Img2.- Requerimos los paquetes

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, así que con el path podr
  ]
}
//Middleware
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 */
```

Img3.- Estructura de Swagger

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'
*/
routeable.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
 */
routeable.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
 */
routeable.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:

```

```

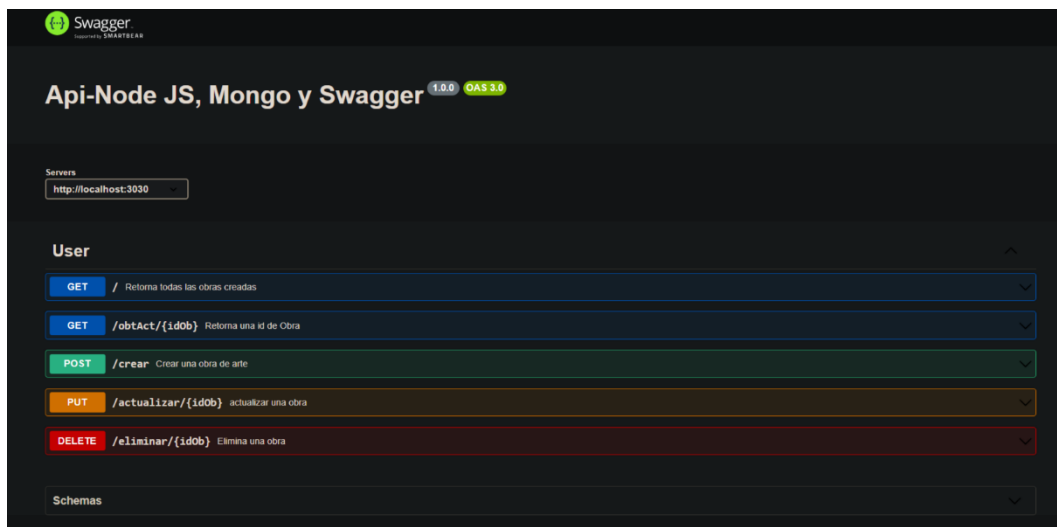
*           application/json:
*             schema:
*               type: object
*               $ref: '#/components/schemas/User'
*     responses:
*       200:
*         description: Obra actualizada
*         content:
*           application/json:
*             schema:
*               type: object
*               $ref: '#/components/schemas/User'
*       404:
*         description: Identificacion no encontrada
*     */
rootable.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
 *   */
rootable.delete('/eliminar/{idOb}', OCLL.eliminaObra)

```

Img4.- Código de implementación de swagger

Implementación de Swagger



Img5.- Vista de Swagger

User

GET / Retoma todas las obras creadas

Cancel

Parameters

No parameters

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:3030/' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:3030/
```

Server response

Code Details

200

Response body

```
{
  "id": "648295888c90b1bf54186c",
  "idObra": "12345",
  "nombreObra": "el sol",
  "tamañoObra": "50x40",
  "tecnicoObra": "oleo",
  "materialesObra": "mdf",
  "valorEconomicoObra": 1000,
  "nombreAutor": "crystian",
  "telefonoAutor": 764100333,
  "correoAutor": "skdkdj@gmail.com",
  "createdAt": "2023-06-08T23:32:20.042Z",
  "updatedAt": "2023-06-09T23:18:16.358Z"
},
  "id": "6482653908c90b1bf541870",
  "idObra": "64564",
  "nombreObra": "sdfs",
  "tamañoObra": "hola",
  "tecnicoObra": "dfs",
  "materialesObra": "dfs",
  "valorEconomicoObra": 123,
  "nombreAutor": "dfs",
  "telefonoAutor": 123,
  "correoAutor": "dfs",
  "createdAt": "2023-06-08T23:33:13.521Z",
  "updatedAt": "2023-06-09T22:55:03.469Z"
}
```

Response headers

```
access-control-allow-origin: *
content-length: 1888
content-type: application/json; charset=utf-8
date: Fri, 07 Jul 2023 20:45:57 GMT
etag: W/"758-qvIwa8c1QVU0BqGNBRd147dg"
x-powered-by: Express
```

Responses

Code	Description	Links
200	Todas las obras creadas	No links

Media type

application/json

Controla Accept header.

Example Value Schema

```
{
  "idObra": 1,
  "nombreObra": "la Soledad",
  "tamañoObra": "Mediano 40x80",
  "tecnicoObra": "Oleo con acrílico",
  "materialesObra": "Pintura, Lienzo",
  "valorEconomicoObra": 1000,
  "nombreAutor": "Crystian",
  "telefonoAutor": 764112222,
  "correoAutor": "crys@gmail.com"
}
```

Img6.- Vista de Swagger Obtiene todos los datos

GET /obAct/{idOb} Retorna una id de Obra

Parameters

Name	Description
idOb ^{required}	Identificación de la obra

string (path)

12345

Execute **Clear**

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:3030/obAct/12345' \
  -H 'accept: application/json'
```

Request URL

http://localhost:3030/obAct/12345

Server response

Code	Details
200	<p>Response body</p> <pre>{ "id": "6482650489e90b1bf54186c", "idObra": "12345", "nombreObra": "el sol", "tamañoObra": "50x40", "técnicaObra": "óleo", "materialesObra": "maf", "valorEconomicoObra": 1000, "nombreAutor": "crystian", "telefonoAutor": "764100333", "correoAutor": "chad@gmail.com", "createdAt": "2023-06-08T23:32:20.042Z", "updatedAt": "2023-06-09T23:18:16.358Z" }</pre> <p>Response headers</p> <pre>access-control-allow-origin: * connection: keep-alive content-length: 306 content-type: application/json; charset=utf-8 date: Fri, 07 Jul 2023 20:47:48 GMT etag: W/"146-551205jerru0CSgtu0cl/3jM" keep-alive: timeout=5 x-powered-by: Express</pre>

Responses

Code	Description	Links
200	Todas las obras creadas	No links

Media type

application/json

Control Accept header

Example Value **Schema**

```
{
  "idObra": 1,
  "nombreObra": "La Soledad",
  "tamañoObra": "Mediano 40x50",
  "técnicaObra": "Óleo con acrílico",
  "materialesObra": "Pintura, lienzo",
  "valorEconomicoObra": 1000,
  "nombreAutor": "Crystian",
  "telefonoAutor": "764112222",
  "correoAutor": "crys@gmail.com"
}
```

404

Identificación no encontrada

No links

Img7.- Vista de Swagger obtiene un dato por id

POST /crear Crear una obra de arte

Parameters

No parameters

Request body ^{required}

application/json

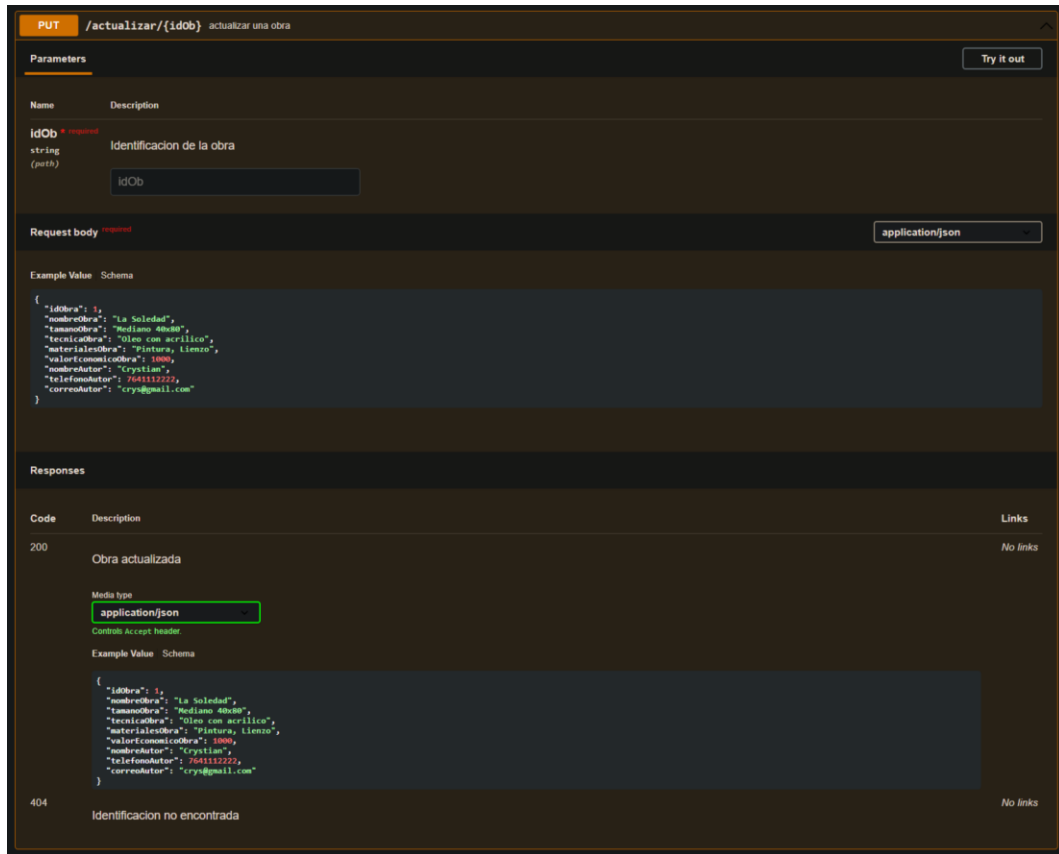
Example Value **Schema**

```
{
  "idObra": 1,
  "nombreObra": "La Soledad",
  "tamañoObra": "Mediano 40x50",
  "técnicaObra": "Óleo con acrílico",
  "materialesObra": "Pintura, lienzo",
  "valorEconomicoObra": 1000,
  "nombreAutor": "Crystian",
  "telefonoAutor": "764112222",
  "correoAutor": "crys@gmail.com"
}
```

Responses

Code	Description	Links
200	Success created	No links

Img8.- Vista de Swagger Guarda un dato



Img9.- Vista de Swagger Actualiza un dato por id



Img10.- Vista de Swagger Elimina un dato por id

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](#)