**DOCUMENTACIÓN PROYECTO**

* **integración Flyway para manejo de script de BD**

En el proyecto Spring Boot con Maven, Flyway permite manejar la versión del esquema de base de datos.

Al iniciar la aplicación Spring Boot, Flyway detectará automáticamente los scripts en el directorio db/migration y aplicará las migraciones necesarias a la base de datos. La primera vez que se ejecute la aplicación, Flyway creará una tabla especial llamada flyway\_schema\_history en la base de datos para revisar qué migraciones se han aplicado.

* **Consola Base de datos H2**

H2 viene con una consola web integrada para visualizar y ejecutar consultas directamente en base de datos durante el desarrollo. Para habilitar y acceder a esta consola en una aplicación Spring Boot:

En el properties, se incuyen :

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console

Así queda disponible la consola de manejo de la BD H2:

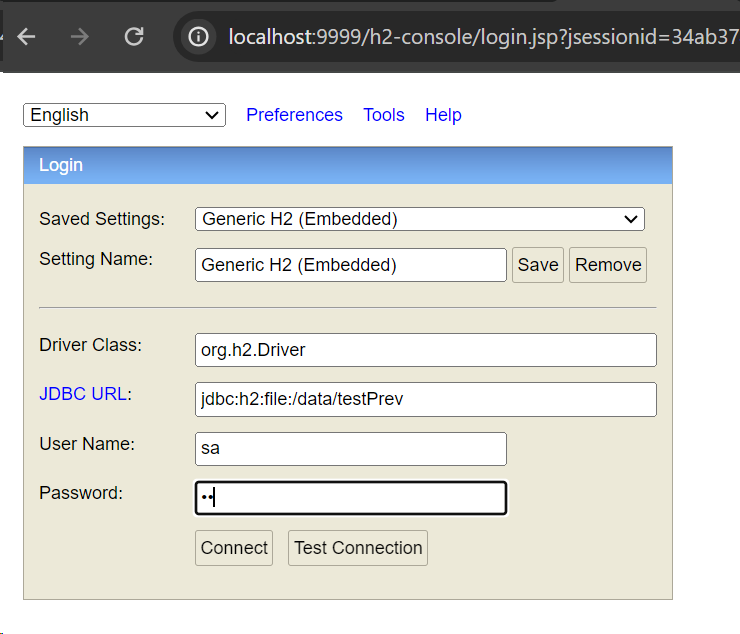
Url en el properties:

server.port=9999

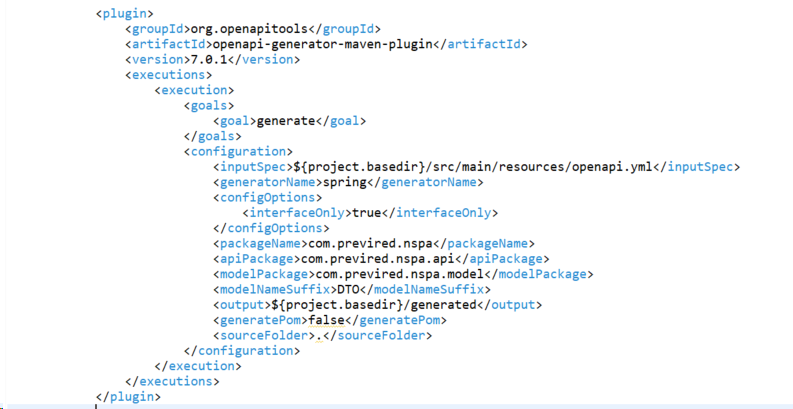
Cuando la aplicación esté ejecución, navega a http://localhost:8080/h2-console en el navegador web (ajustar el puerto dependiendo de la configuración).

En la pantalla de login de la consola, asegúrate de que la configuración de conexión JDBC coincida con la de tu application.properties. Introducir credenciales de usuario y contraseña (config. en el properties) si las configuraste; de lo contrario, el usuario por defecto suele ser "sa" sin contraseña.

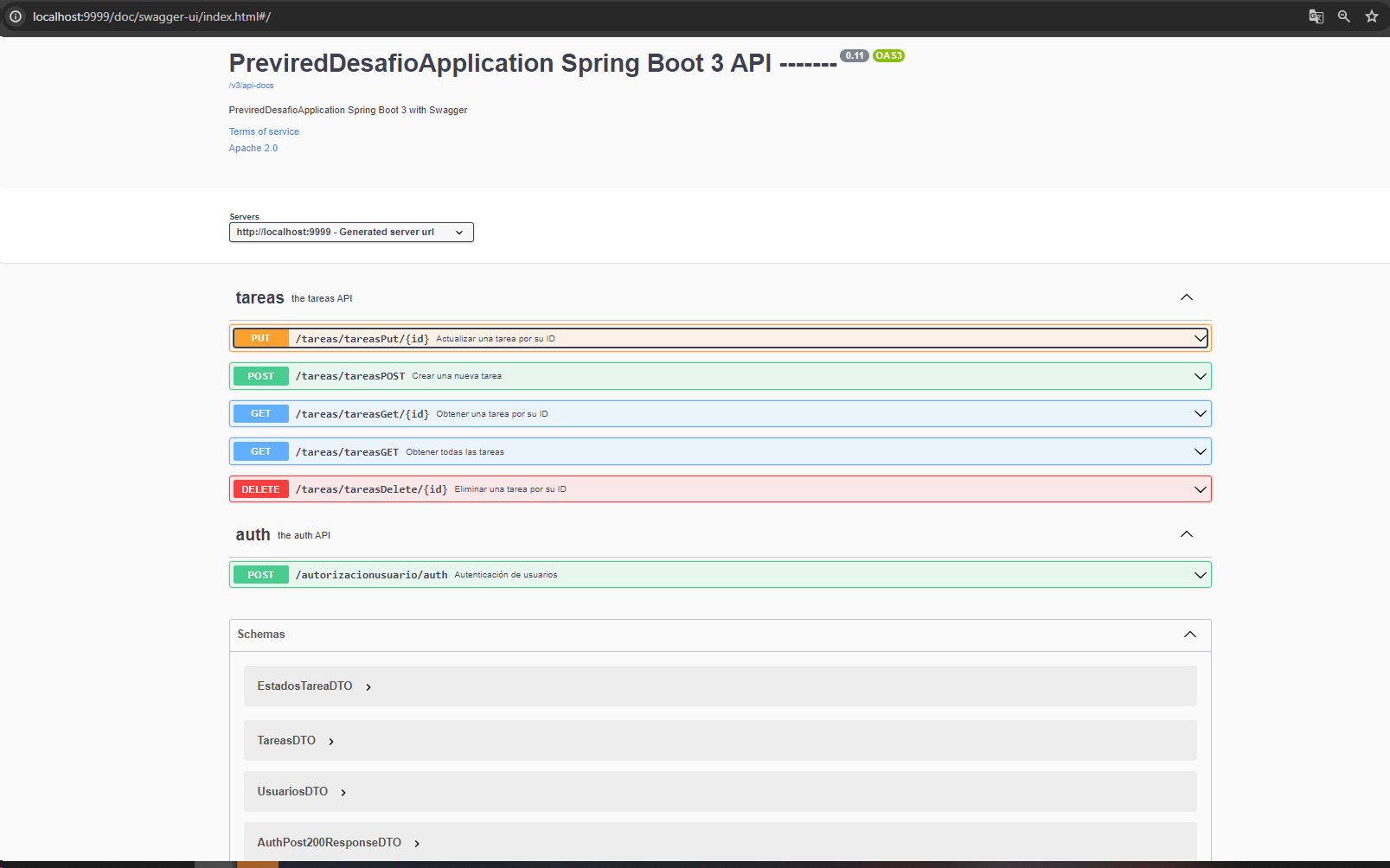
la siguiente propiedad disponibiliza una "BD" a través de un archivo, para este caso el archivo lo llamé "testPrev", entonces la url para la BD es:

spring.datasource.url=jdbc:h2:file:/data/testPrev;DB\_CLOSE\_DELAY=-1;DB\_CLOSE\_ON\_EXIT=FALSE  
  
  


* **Documentación con OpenAPI y Swagger**

Se ha ocupado OpenApi y swagger para la documentación de las apis y las clases model asociadas a las tablas TAREA, ESTADO\_TAREA y USUARIOS.  
Para la generación se ocupó un archivo “openapi.yml” con la estructura de las apis y los componentes (Models, clases DTO en este caso) y se generó a partir de este yml con el generador openapi plugin:  


Para acceder a la plataforma con las vistas documentadas de las apis en swagger, acceder a la url siguiente web (ajustar el puerto dependiendo de la configuración):

<http://localhost:9999/doc/swagger-ui/index.html>  
  


A mi entender cumplo con un correcto orden de la codificación, la aplicación de buenas prácticas y con las mayoría de los principios SOLID, me atreverá a decir que con todos, tal vez con el Principio de Inversión de Dependencias esté bajo, pero a juicio general, cumplo.

* **JSON de pruebas**

Si bien Swagger entrega el formato de los json, dejaré adjunto el formato de las apis que requieren JSON.

* + **Autenticación Usuarios**  
    /autorizacionusuario/auth  
    **Requestbody**

*{*

*"idUsuario": 0,*

*"nombreUsuario": "string",*

*"correoElectronico": "string",*

*"contrasenaHash": "string"*

*}*

* + **Actualizar Tarea**

/tareas/tareasPut/{id} **Requestbody**

*{*

*"idTarea": 0,*

*"usuario": {*

*"idUsuario": 0,*

*"nombreUsuario": "string",*

*"correoElectronico": "string",*

*"contrasenaHash": "string"*

*},*

*"titulo": "string",*

*"descripcion": "string",*

*"fechaCreacion": "2024-04-02T17:49:34.634Z",*

*"fechaLimite": "2024-04-02T17:49:34.634Z",*

*"estadoTarea": {*

*"idEstado": 0,*

*"nombreEstado": "string"*

*}*

*}*

* + **Crear Tarea**  
    /tareas/tareasPOST

**Requestbody**

*{*

*"idTarea": 0,*

*"usuario": {*

*"idUsuario": 0,*

*"nombreUsuario": "string",*

*"correoElectronico": "string",*

*"contrasenaHash": "string"*

*},*

*"titulo": "string",*

*"descripcion": "string",*

*"fechaCreacion": "2024-04-02T17:50:15.475Z",*

*"fechaLimite": "2024-04-02T17:50:15.475Z",*

*"estadoTarea": {*

*"idEstado": 0,*

*"nombreEstado": "string"*

*}*

*}*

* + **Obtener Tarea**/tareas/tareasGet/{id}  
    **parámetro:***“id” de tipo entero*
  + **Obtener todas las Tarea**

/tareas/tareasGET

* + **Eliminar Tarea**

/tareas/tareasDelete/{id}

**parámetro:***“id” de tipo entero*