

Elaboro	Documento	Versión	Descripción	Fecha
Luis Sánchez Martínez	Práctica Crear proyecto Spring Boot con OpenAPI y H2	1.0	Creación de documento	26/03/2024

Práctica Crear proyecto Spring Boot con documentación de OpenAPI y H2.

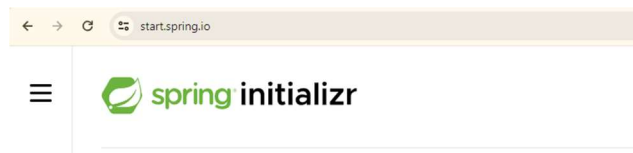
Tiempo estimado: 2 horas.

Uso de:

- Spring Initializr
- IntelliJ
- OpenAPI
- H2

Generar proyecto Spring Boot en sitio spring initializr

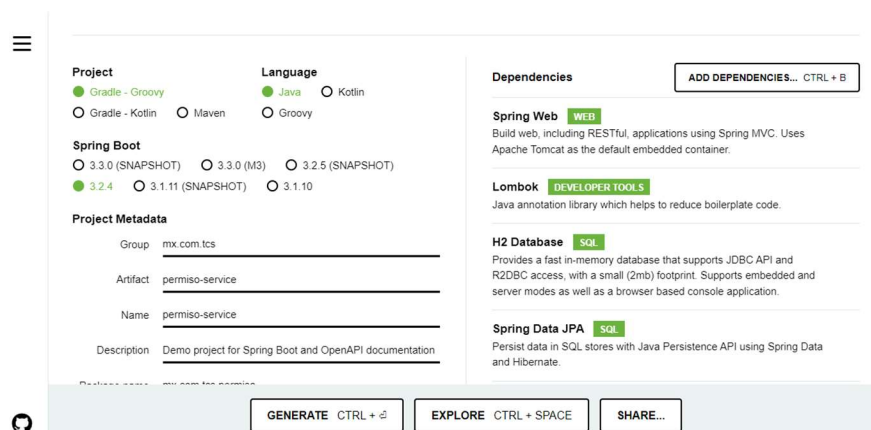
1. Consultar la página: <https://start.spring.io/>



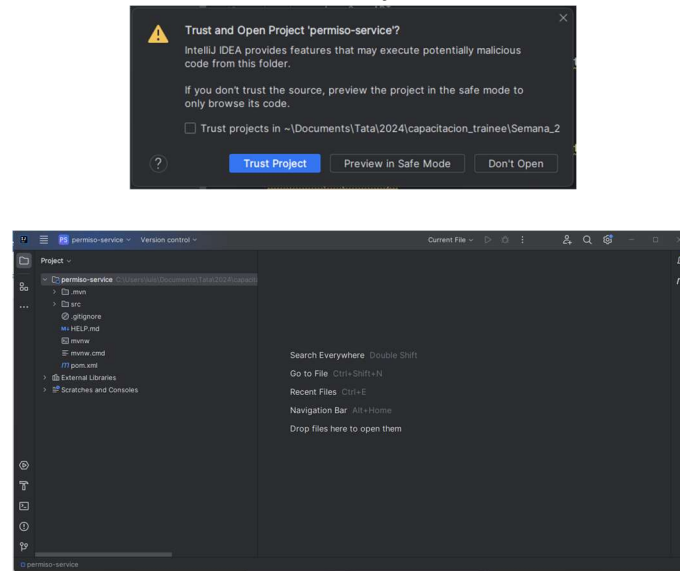
2. Configurar un proyecto Maven, seleccionando las librerías:

- a. Spring Web
- b. Lombok
- c. H2 Database
- d. Spring Data JPA

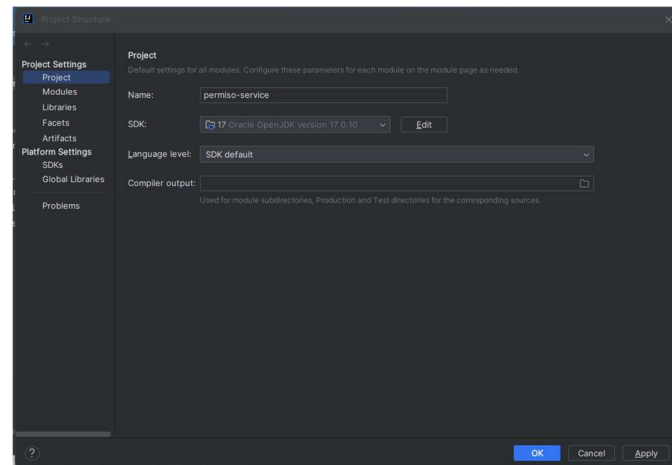
y dar clic en la opción *Generate*:



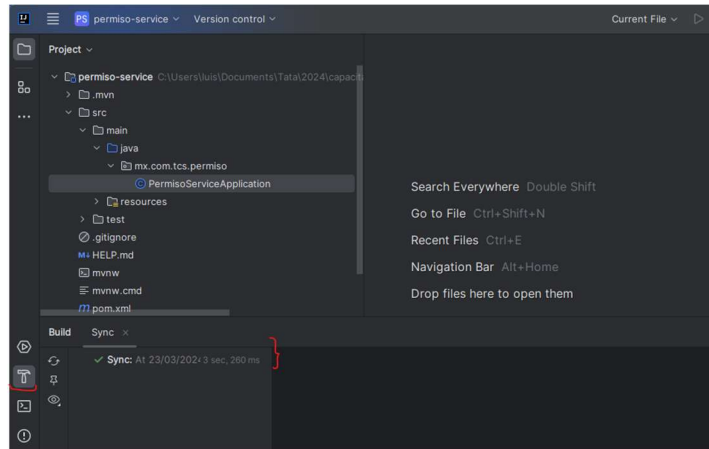
3. Abrir proyecto en IDE IntelliJ al dar clic en *Trust Project*:



4. Configurar SDK a Java 17 del proyecto al dar clic en el botón de settings y Project Structure:



5. Construir el proyecto dando clic en el botón martillo.



6. Agregar la clase Controller y el agregar el primer endpoint GET /permiso de la API catpermiso-service.

```

1  PermisoController.java
2
3  public PermisoController(IPermisoService permisoService) {
4      this.permisoService = permisoService;
5  }
6
7  /**
8   * Endpoint used to get all records in Permiso entity.
9   * @return a ResponseEntity of the list of Permiso object.
10 */
11
12 no usages
13
14 @RequestMapping("/permiso")
15 public ResponseEntity<List<Permiso>> getAll() {
16     return permisoService.listAll();
17 }
18
19

```

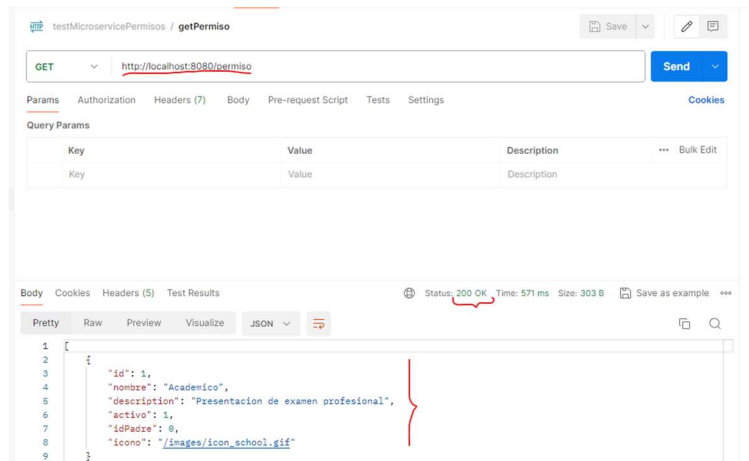
7. Construir el proyecto y ejecutarlo desde el IDE:

```

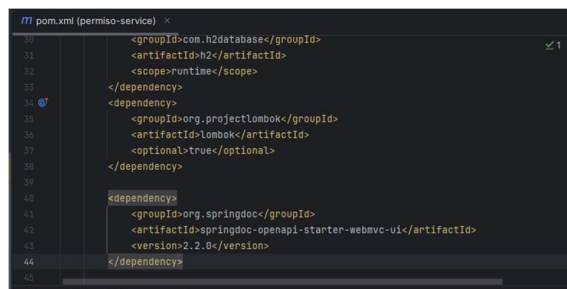
1  PermisoController.java
2
3  public PermisoController(IPermisoService permisoService) {
4      this.permisoService = permisoService;
5  }
6
7  /**
8   * Endpoint used to get all records in Permiso entity.
9   * @return a ResponseEntity of the list of Permiso object.
10 */
11
12 no usages
13
14 @RequestMapping("/permiso")
15 public ResponseEntity<List<Permiso>> getAll() {
16     return permisoService.listAll();
17 }
18
19

```

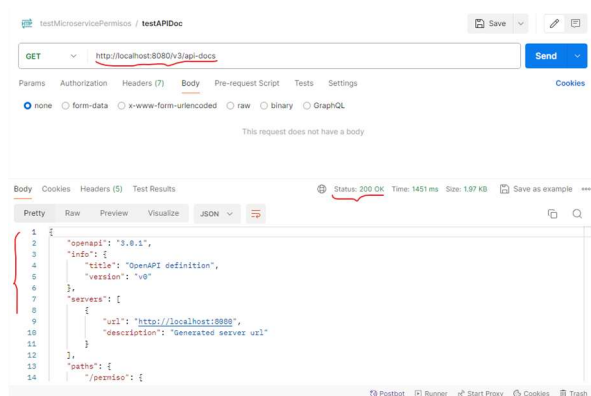
8. Validar el endpoint /permiso con la función GET ejecutado desde postman:



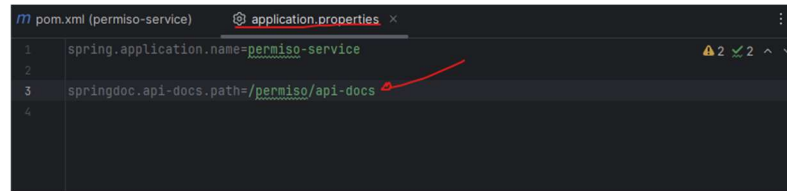
9. Integrar la librería springdoc en el proyecto:



10. Validar la información de OpenAPI consultando la URL: <http://localhost:8080/v3/api-docs>, después de ejecutar el proyecto:

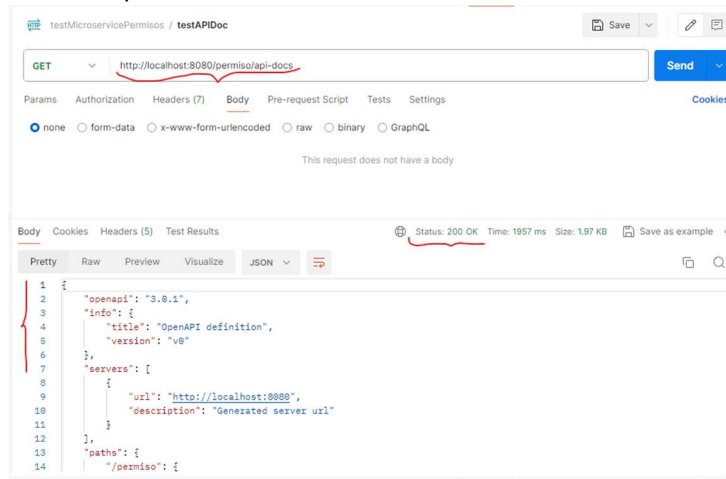


11. Actualizar el Path para acceder a la documentación de OpenAPI usando la propiedad springdoc.api-docs.path=/api-docs:

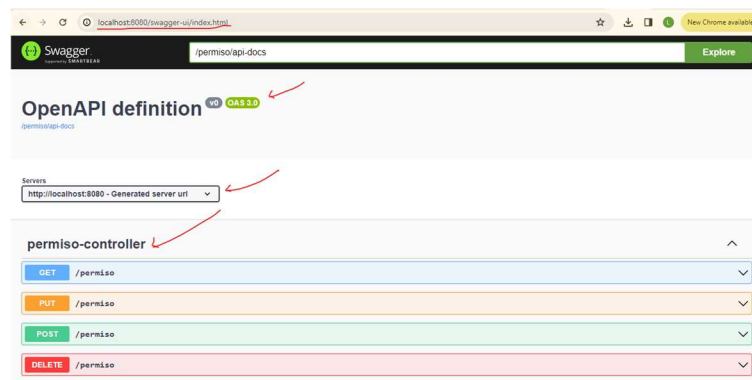


```
1 spring.application.name=permiso-service
2
3 springdoc.api-docs.path=/permiso/api-docs
4
```

12. Validar la información de OpenAPI:

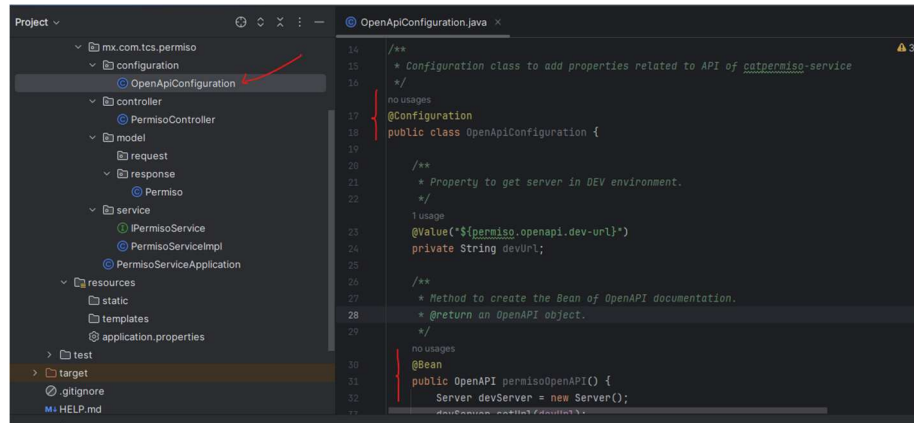


13. Para consultar el formato en HTML debe de ser desde un navegador web consultando la ruta:
<http://localhost:8080/swagger-ui/index.html>

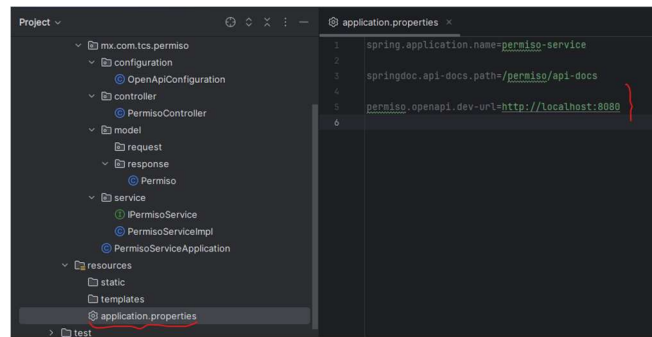


14. Crear la clase Configuration para integrar la documentación de OpenAPI en el proyecto agregando información de las propiedades:

- a. Título
- b. Versión
- c. Contacto
- d. Descripción
- e. Servers

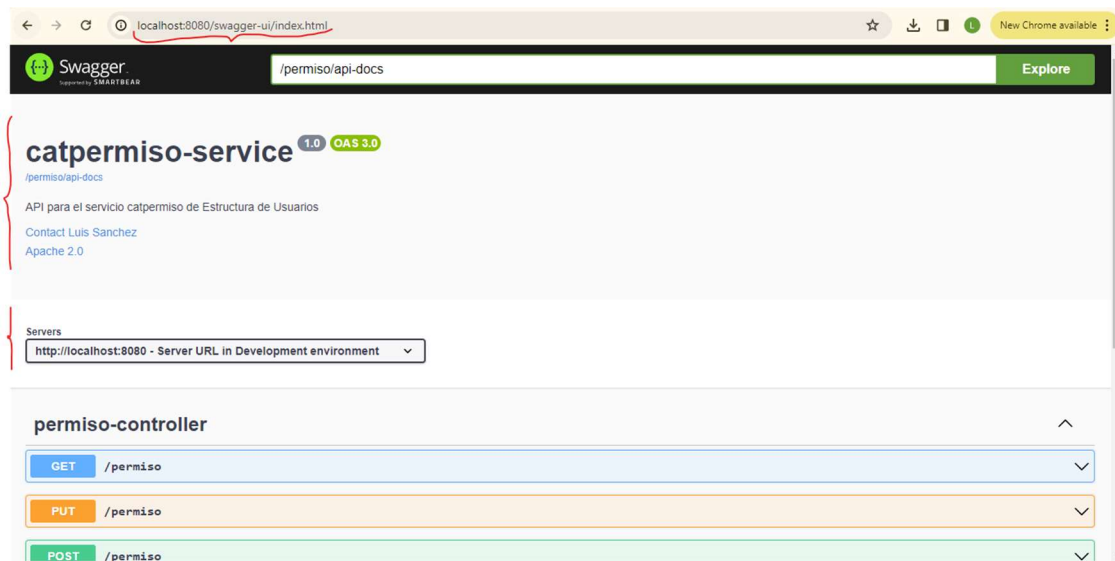


15. Validar la propiedad integrada con el valor del server en el ambiente DEV:



16. Consultar la documentación en formato html desde la ruta:

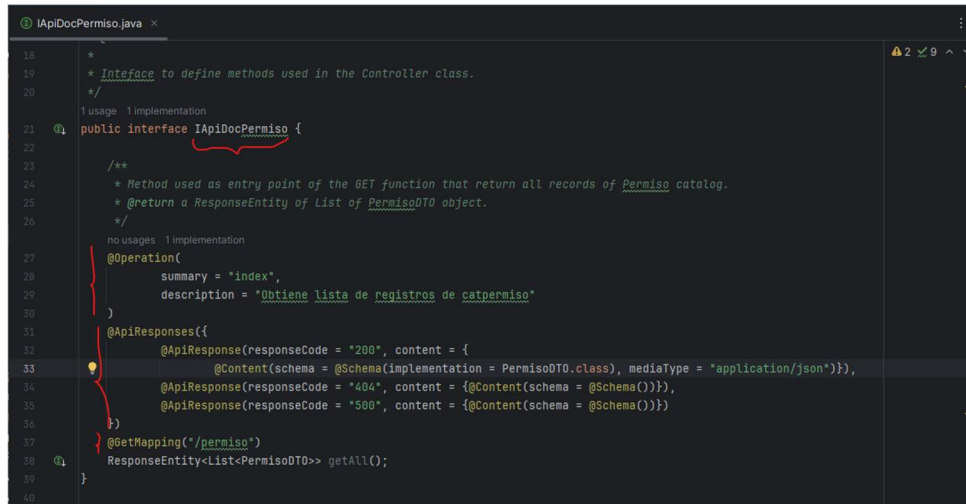
<http://localhost:8080/swagger-ui/index.html>



17. Integrar la documentación de OpenAPI en el método GET del *Controller* usado en el microservicio usando las notaciones:

- a. `@Operation`
- b. `@ApiResponse`
- c. `@Content`
- d. `@Schema`

Para este caso se creara una interface donde se define el método que se emplementa en el `Controller: getAll()`.



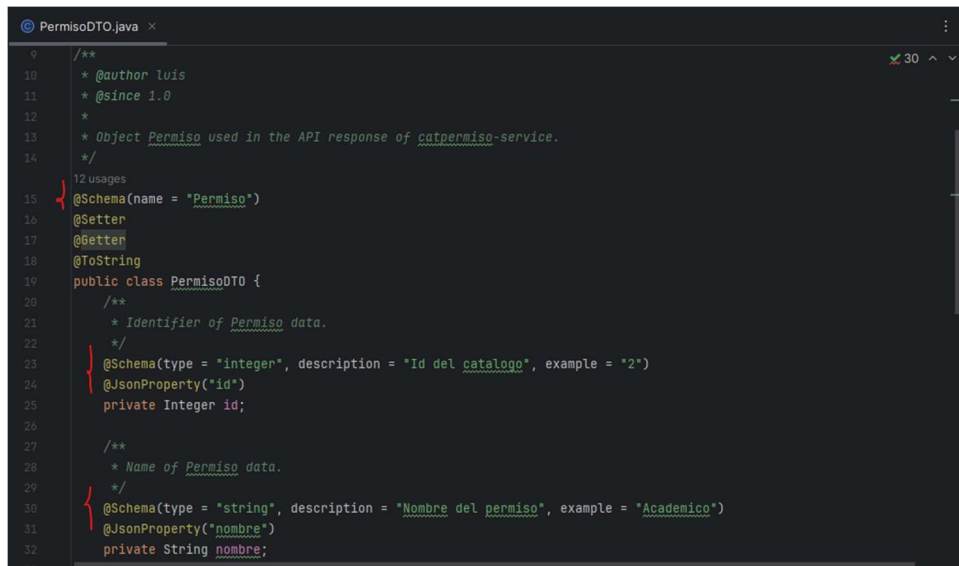
```

18  *
19  * Interface to define methods used in the Controller class.
20  */
21  public interface IApiDocPermiso {
22
23      /**
24       * Method used as entry point of the GET function that return all records of Permiso catalog.
25       * @return a ResponseEntity of List of PermisoDTO object.
26       */
27      @Operation(
28          summary = "index",
29          description = "Obtiene lista de registros de catpermiso"
30      )
31      @ApiResponse(
32          @ApiResponse(responseCode = "200", content = {
33              @Content(schema = @Schema(implementation = PermisoDTO.class), mediaType = "application/json")
34            },
35            @ApiResponse(responseCode = "404", content = {
36              @Content(schema = @Schema(), mediaType = "application/json")
37            },
38            @ApiResponse(responseCode = "500", content = {
39              @Content(schema = @Schema(), mediaType = "application/json")
40            })
41      )
42      @GetMapping("/permiso")
43      ResponseEntity<List<PermisoDTO>> getAll();
44  }

```

18. Agregar la documentación del esquema a la clase DTO usando las notaciones:

- a. `@Schema`
- b. `@JsonProperty`



```

9  /**
10   * @author luis
11   * @since 1.0
12   *
13   * Object Permiso used in the API response of catpermiso-service.
14   */
15   @Schema(name = "Permiso")
16   @Setter
17   @Getter
18   @ToString
19   public class PermisoDTO {
20
21       /**
22        * Identifier of Permiso data.
23        */
24       @Schema(type = "integer", description = "Id del catalogo", example = "2")
25       @JsonProperty("id")
26       private Integer id;
27
28       /**
29        * Name of Permiso data.
30        */
31       @Schema(type = "string", description = "Nombre del permiso", example = "Academico")
32       @JsonProperty("nombre")
33       private String nombre;
34   }

```

19. Agregar las propiedades de configuración:

- a. `springdoc.packages-to-scan`
- b. `springdoc.swagger-ui.tryItOutEnabled`
- c. `springdoc.swagger-ui.operationsSorter`

- d. springdoc.swagger-ui.tagsSorter
- e. springdoc.swagger-ui.filter

```

4  springdoc.api-docs.path=/permiso/api-docs
5  springdoc.packages-to-scan=mx.com.tcs.permiso.controller
6  springdoc.swagger-ui.tryItOutEnabled=false
7  springdoc.swagger-ui.operationsSorter=method
8  springdoc.swagger-ui.tagsSorter=alpha
9  springdoc.swagger-ui.filter=false
10

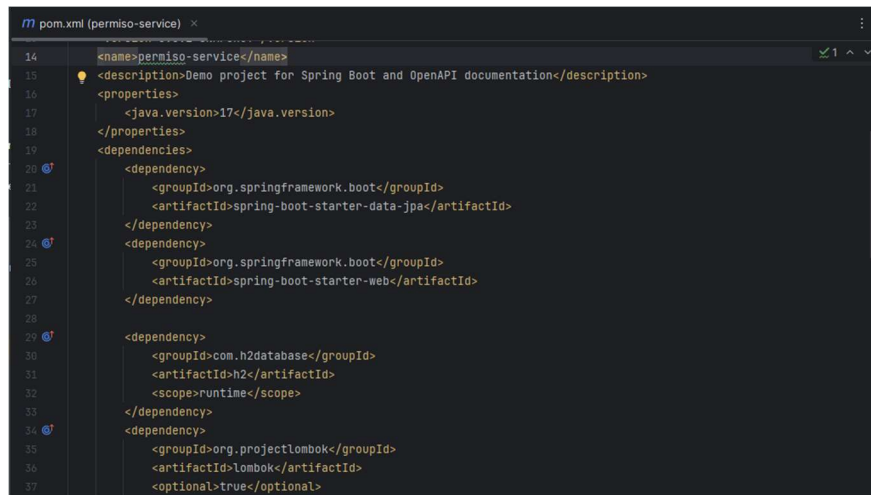
```

20. Consultar la documentación en formato HTML desde la ruta <http://localhost:8080/swagger-ui/index.html> para confirmar la actualización de los datos de las funciones:

The screenshot displays the Swagger UI for the 'Permiso' API. The main section shows the 'GET /permiso/index' endpoint with a description 'Obtiene lista de registros de catpermiso'. The 'Parameters' section is empty. The 'Responses' section shows a 200 OK response with a media type of 'application/json'. A red bracket on the left highlights the 'Responses' section. Below the main interface, a 'Schemas' section is visible, showing a schema for 'Permiso' with fields like 'id', 'numero', 'descripcion', 'activo', 'estado', and 'icono'.

21. Integrar en el *pom* del proyecto para el uso de la base de datos las librerías:
- a. *spring-boot-starter-data-jpa*
 - b. *h2*

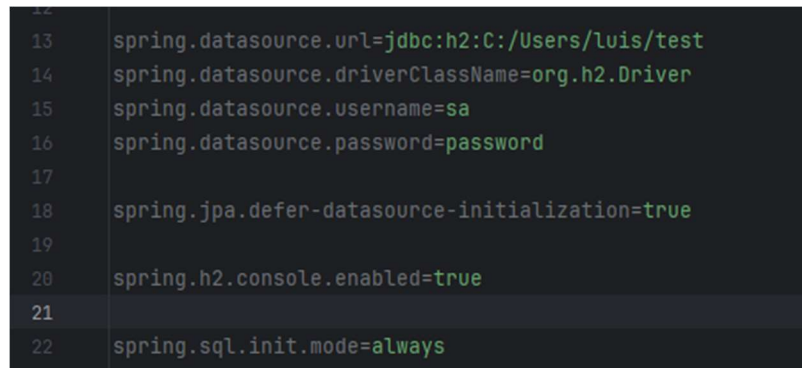
c. *modelmapper*



```
14 <name>permiso-service</name>
15 <description>Demo project for Spring Boot and OpenAPI documentation</description>
16 <properties>
17   <java.version>17</java.version>
18 </properties>
19 <dependencies>
20   <dependency>
21     <groupId>org.springframework.boot</groupId>
22     <artifactId>spring-boot-starter-data-jpa</artifactId>
23   </dependency>
24   <dependency>
25     <groupId>org.springframework.boot</groupId>
26     <artifactId>spring-boot-starter-web</artifactId>
27   </dependency>
28
29   <dependency>
30     <groupId>com.h2database</groupId>
31     <artifactId>h2</artifactId>
32     <scope>runtime</scope>
33   </dependency>
34   <dependency>
35     <groupId>org.projectlombok</groupId>
36     <artifactId>lombok</artifactId>
37     <optional>true</optional>
```

22. Configurar las propiedades usadas por la base de datos H2:

- a. `spring.datasource.url`
- b. `spring.datasource.driverClassName`
- c. `spring.datasource.username`
- d. `spring.datasource.password`
- e. `spring.jpa.defer-datasource-initialization`
- f. `spring.h2.console.enabled`
- g. `spring.sql.init.mode`



```
13 spring.datasource.url=jdbc:h2:C:/Users/luis/test
14 spring.datasource.driverClassName=org.h2.Driver
15 spring.datasource.username=sa
16 spring.datasource.password=password
17
18 spring.jpa.defer-datasource-initialization=true
19
20 spring.h2.console.enabled=true
21
22 spring.sql.init.mode=always
```

23. Generar los scripts SQL para crear la tabla e insertar los registros al iniciar el proyecto y colocarlos en la carpeta resources:

- a. `schema.sql`
- b. `data.sql`

```

schema.sql x
1 DROP TABLE IF EXISTS Permiso;
2 CREATE TABLE Permiso (
3     id INT AUTO_INCREMENT PRIMARY KEY,
4     nombre VARCHAR(25) NOT NULL,
5     descripcion VARCHAR(50) NOT NULL,
6     activo INT NOT NULL,
7     id_Padre INT NOT NULL,
8     icono VARCHAR(50) NOT NULL
9 );

```

```

data.sql x
1 INSERT INTO Permiso VALUES (1,'Academico', 'Categoria de permisos academicos', 1, 1, '/images/icon_school.gif');
2 INSERT INTO Permiso VALUES (2,'Enfermedad', 'Categoria de permisos por enfermedad', 1, 2, '/images/icon_disease.gif');
3 INSERT INTO Permiso VALUES (3,'Academico Profesional', 'Presentacion de exámenes', 1, 1, '/images/icon_school_quiz.gif');

```

24. Generar las clases *Entity* y *Repository* de la tabla Permiso.

```

Permiso.java x
10
11 /**
12  * @author luis
13  * @since 1.0
14  *
15  * Entity class of the table Permiso.
16  */
17 @Entity
18 @Table
19 @Getter
20 @Setter
21 @ToString
22 public class Permiso {
23
24     /**
25      * Identifier of record.
26      */
27     @Id
28     @Column
29     private Integer id;
30
31     /**
32      * Name column.
33      */
34     @Column

```

```

PermisoRepository.java x
1 package mx.com.tcs.permiso.model.repository;
2
3 > import ...
4
5
6
7 /**
8  * @author luis
9  * @since 1.0
10  *
11  * Repository of the Permiso entity.
12  */
13 @Repository
14 public interface PermisoRepository extends CrudRepository<Permiso,Integer> {
15 }

```

25. Modificar la clase *ServiceImpl* para consultar la información de la tabla usando la interfaz *Repository* y *ModelMapper* para convertir el *Entity* en *DTO*.

```
PermisoServiceImpl.java
46
47
48 /**
49  * Method to query from Permiso entity.
50  * @return a list of Permiso object.
51  */
52
53 1 usage
54 private List<PermisoDTO> getAllPermiso() {
55     List<PermisoDTO> permisoDTOList = new ArrayList<>();
56
57     List<Permiso> permisoList = (List<Permiso>) repository.findAll();
58
59     permisoDTOList = permisoList.
60         stream().
61         map(permiso -> modelMapper.map(permiso, PermisoDTO.class)).
62         collect(Collectors.toList());
63
64     return permisoDTOList;
65 }
```

26. Construir el proyecto y ejecutar el microservicio:

```
Run PermisoServiceApplication
[ice] [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH0000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to er
[ice] [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
[ice] [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be pe
[ice] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8090 (http) with context path ''
[ice] [main] m.c.t.permiso.PermisoServiceApplication : Started PermisoServiceApplication in 20.108 seconds (process running for 23.672)
[ice] [nio-8090-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
[ice] [nio-8090-exec-1] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
[ice] [nio-8090-exec-1] o.s.web.servlet.DispatcherServlet : Completed initialization in 3 ms
```

27. Probar el endpoint desde postman y desde el botón “Try it out” de la documentación generada desde OpenAPI.

testMicroservicePermisos / getPermiso

GET http://localhost:8090/permiso

Send

Params

Key	Value	Description
Key	Value	Description

Body

200 OK 1397 ms 588 B

Save as example

Query Params

Key	Value	Description
Key	Value	Description

Raw

```
1 {
2   {
3     "id": 1,
4     "nombre": "Academico",
5     "descripcion": "Categoría de permisos academicos",
6     "activo": 1,
7     "idPadre": 1,
8     "icono": "/images/icon_school.gif"
9   },
10  {
11    "id": 2,
12    "nombre": "Enfermedad",
13    "descripcion": "Categoría de permisos por enfermedad",
14    "activo": 1,
15    "idPadre": 2,
16    "icono": "/images/icon_disease.gif"
17  },
18  {
19    "id": 3,
20    "nombre": "Academico Profesional",
21    "descripcion": "Presentacion de exámenes",
22    "activo": 1,
23    "idPadre": 1,
```

No parameters

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8090/permiso' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:8090/permiso
```

Server response

Code	Details
------	---------

200	
-----	--

Response body

```
[
  {
    "id": 1,
    "nombre": "Academico",
    "descripcion": "Categoria de permisos academicos",
    "activo": 1,
    "idPadre": 1,
    "icono": "/images/icon_school.gif"
  },
  {
    "id": 2,
    "nombre": "Enfermedad",
    "descripcion": "Categoria de permisos por enfermedad"
```

Referencias

<https://www.baeldung.com/spring-rest-openapi-documentation>

<https://www.baeldung.com/spring-boot-json>

<https://www.bezkoder.com/spring-boot-swagger-3/>