FUNDACION UNIVERDITARIA DE LA RIOJA

Faculta de Ingeniería

Especialización de Ingeniería de Software

Desarrollo de Aplicaciones Web

Jose Alexis Rozo Bahamon

14136457

Profesor: Javier Díaz Díaz

17/06/2024

Ibagué – Tolima

1. Introducción

En este proyecto, se desarrollará el Back-End de una aplicación web utilizando Java y el framework Spring Boot. El objetivo es implementar uno o dos microservicios que hagan uso de los verbos HTTP (GET, POST, DELETE, PUT) y permitan búsquedas por ID y por el nombre de un atributo. La persistencia de datos se realizará con MySQL.

2. Requisitos del Proyecto

- Lenguaje de programación: Java
- Framework: Spring Boot
- Base de datos: MySQL
- Microservicios: 1
- Verbos HTTP: GET, POST, DELETE, PUT
- Métodos de búsqueda: Por ID y por nombre de atributo

3. Arquitectura del Sistema

Describe la arquitectura de tu sistema, incluyendo diagramas de los componentes principales y cómo interactúan entre sí.

4. Configuración del Entorno de Desarrollo

Prerrequisitos:

- JDK 8 o superior
- Maven o Gradle
- Spring Boot
- MySQL

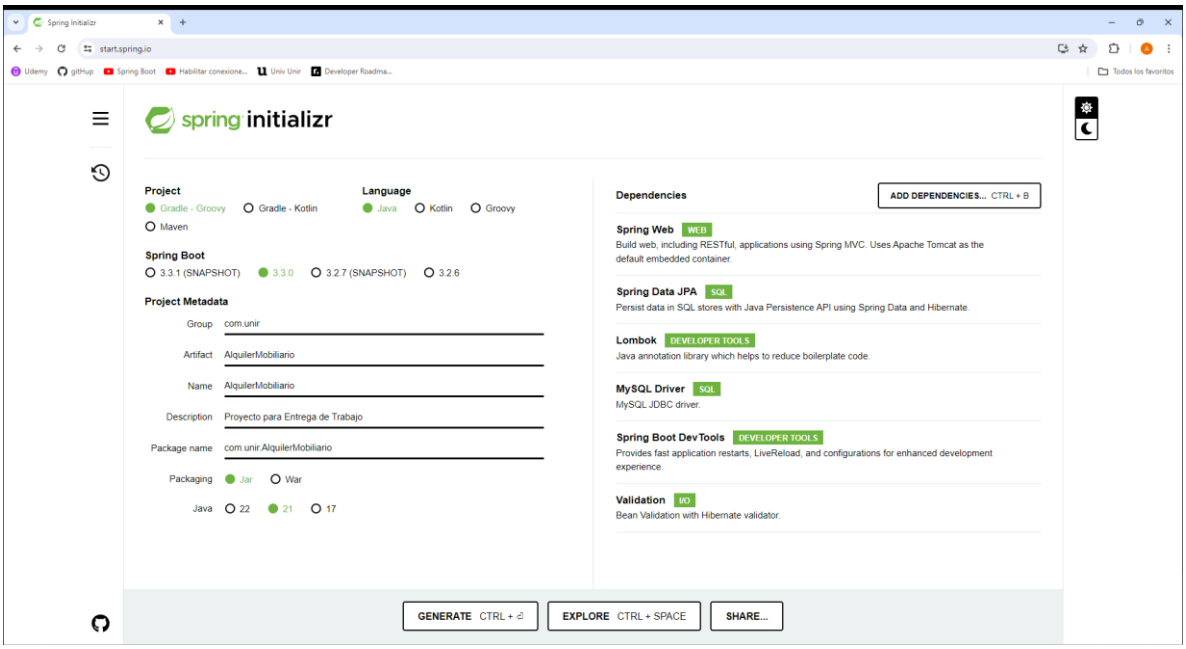
Configuración:

4.1 Instalación de Java y Maven/Gradle

Para esto ya haremos la instalación ya que primero, se entiende que debe de estar instalados estas herramientas y nos enfocaremos mas en el servicio.

4.2 Configuración de Spring Boot

Configuramos nuestro proyecto por **Spring Initializr** donde creamos un proyecto para alquiler de mobiliarios para fiestas.

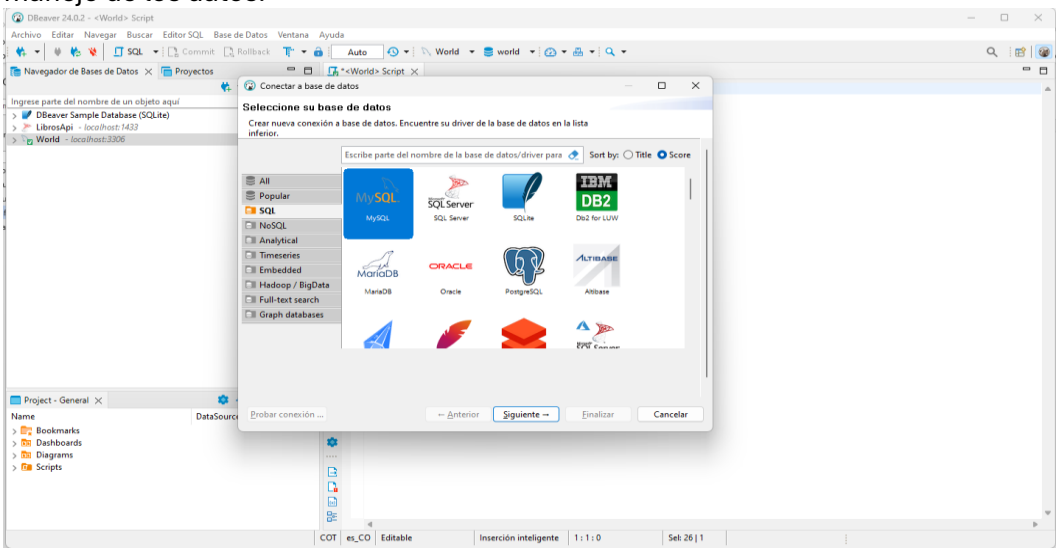


Como podemos ver, le agregamos las dependencias para generar nuestro microservicio que son:

- Spring Web
- Spring Boot DevTools
- Lombok
- Mysql Driver
- Spring Data JPA
- Validation

5. **Configuración de MySQL**

Para la interacción con la base de datos utilizaremos el IDE DBeaver para las consultas y manejo de los datos.



6. **Desarrollo del Microservicio**

6.1 Estructura del Proyecto

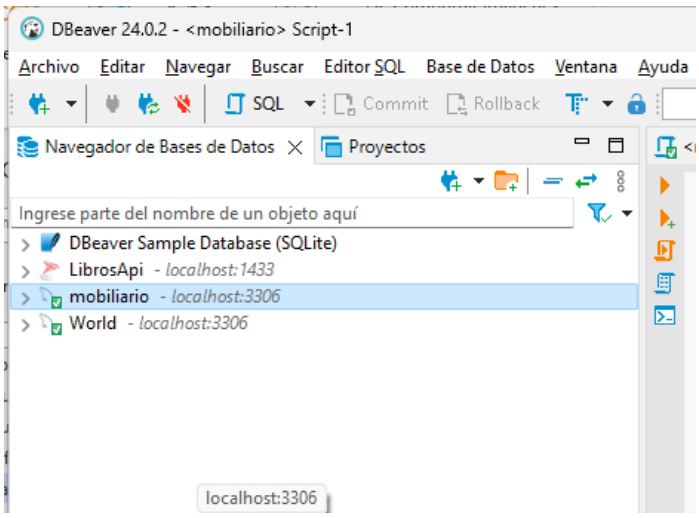
De acuerdo con la imagen proporcionada, la estructura del proyecto es:

```
src
└── main
    ├── java
    │   └── com
    │       └── ejemplo
    │           └── demo
    │               ├── controller
    │               ├── service
    │               ├── repository
    │               └── model
    └── resources
        ├── application.properties
```

6.2 Configuración de la Base de Datos MySQL

Spring Boot configuration files are supported by IntelliJ IDEA Ultimate

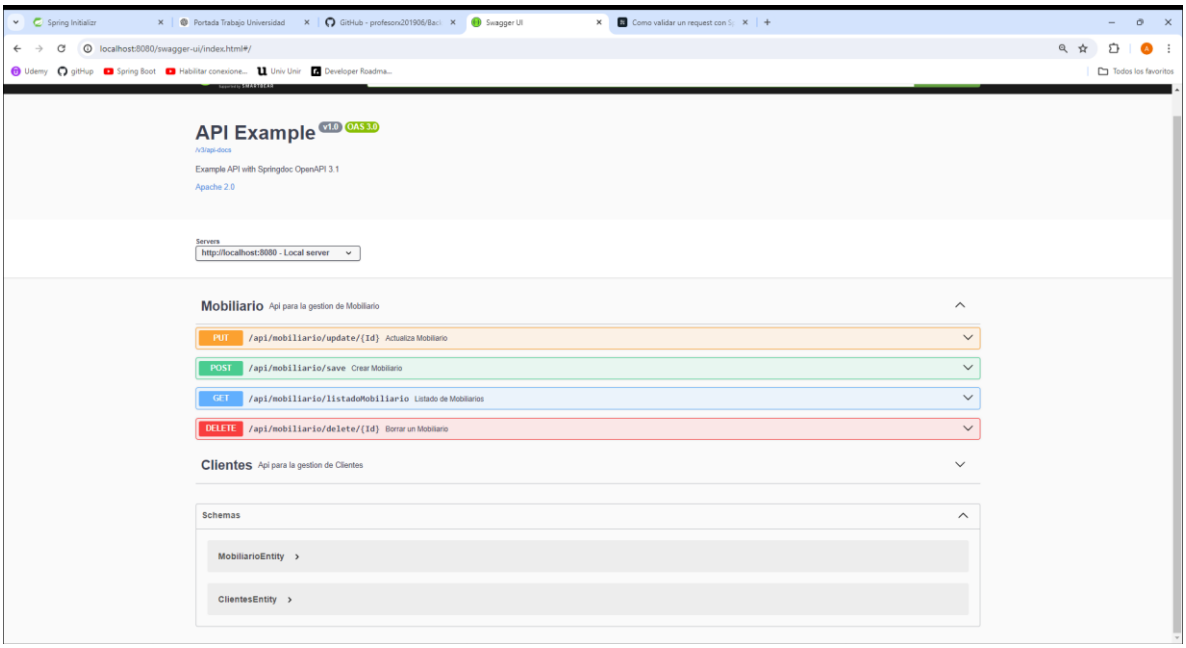
```
spring.application.name=mobiliario
spring.datasource.url=jdbc:mysql://localhost:3306/mobiliario?createDatabaseIfNotExist=true
spring.datasource.username=root
spring.datasource.password=12345678
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
```



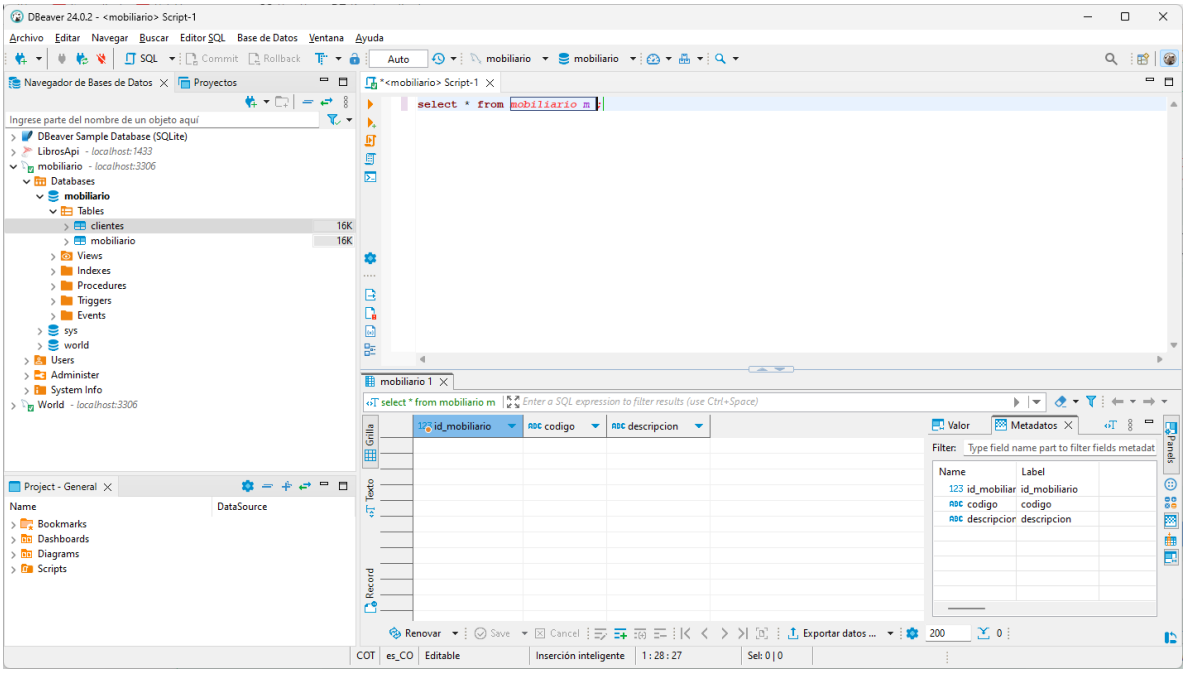
Se creo la base de datos con el comando **createDatabaseIfNotExist=true**

6.3 Detalle de CRUD para los Mobiliarios a alquilar

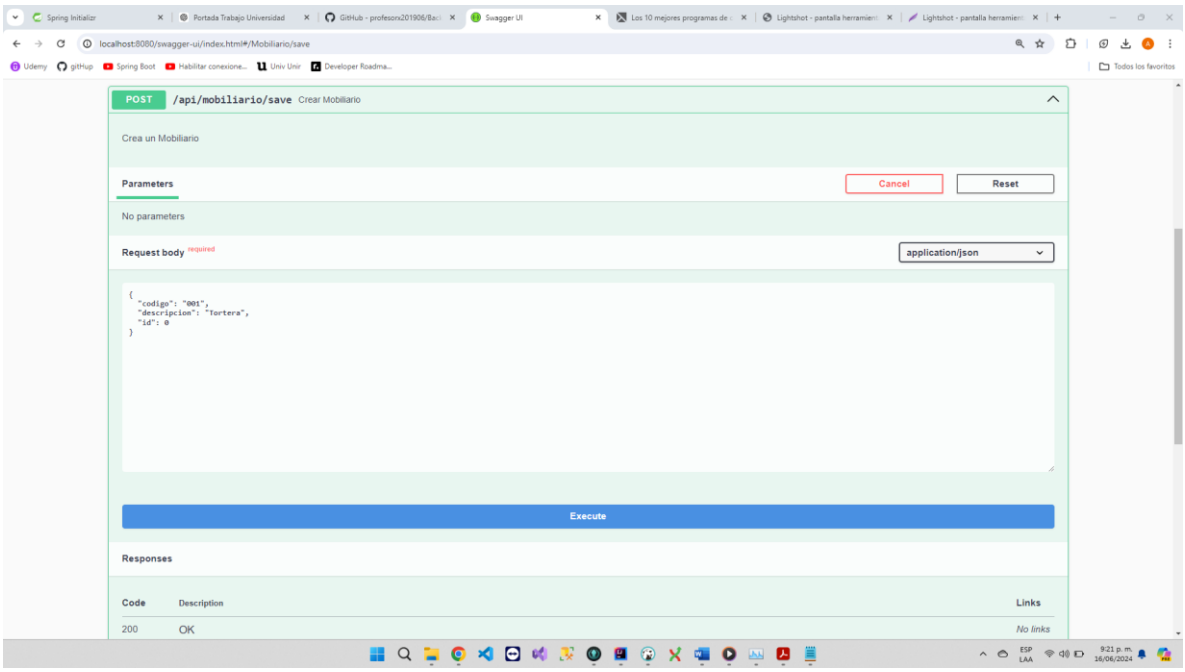
Crear



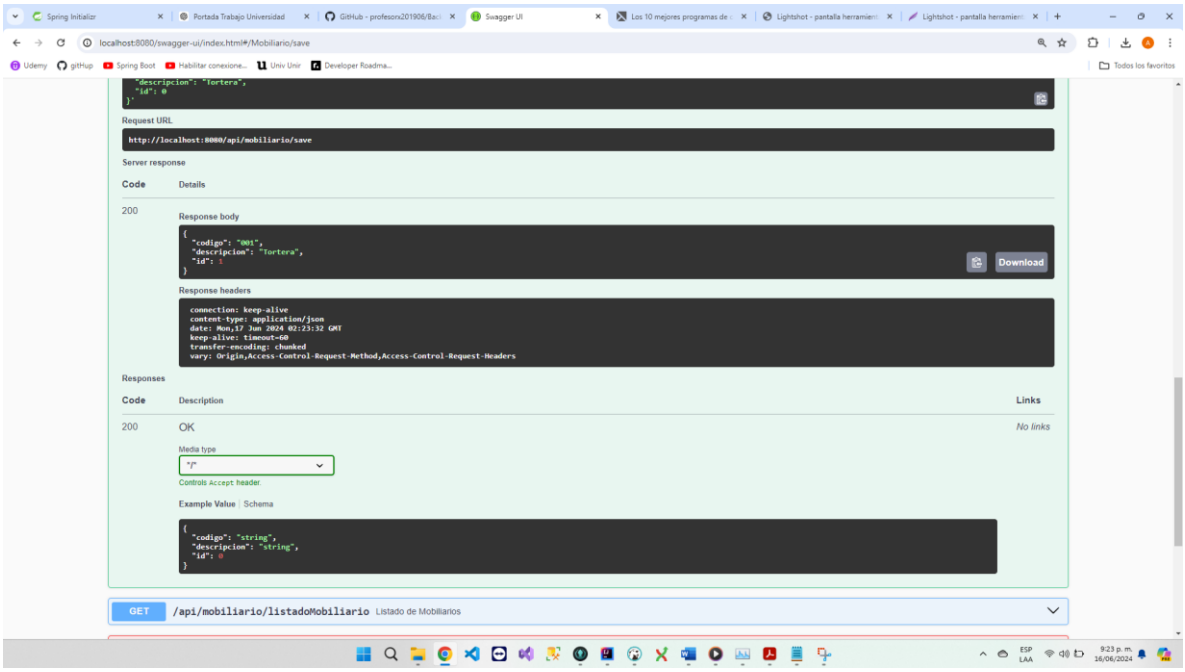
Validamos que al correr el proyecto se creó las tablas y así mismo vamos a hacer el CRUD



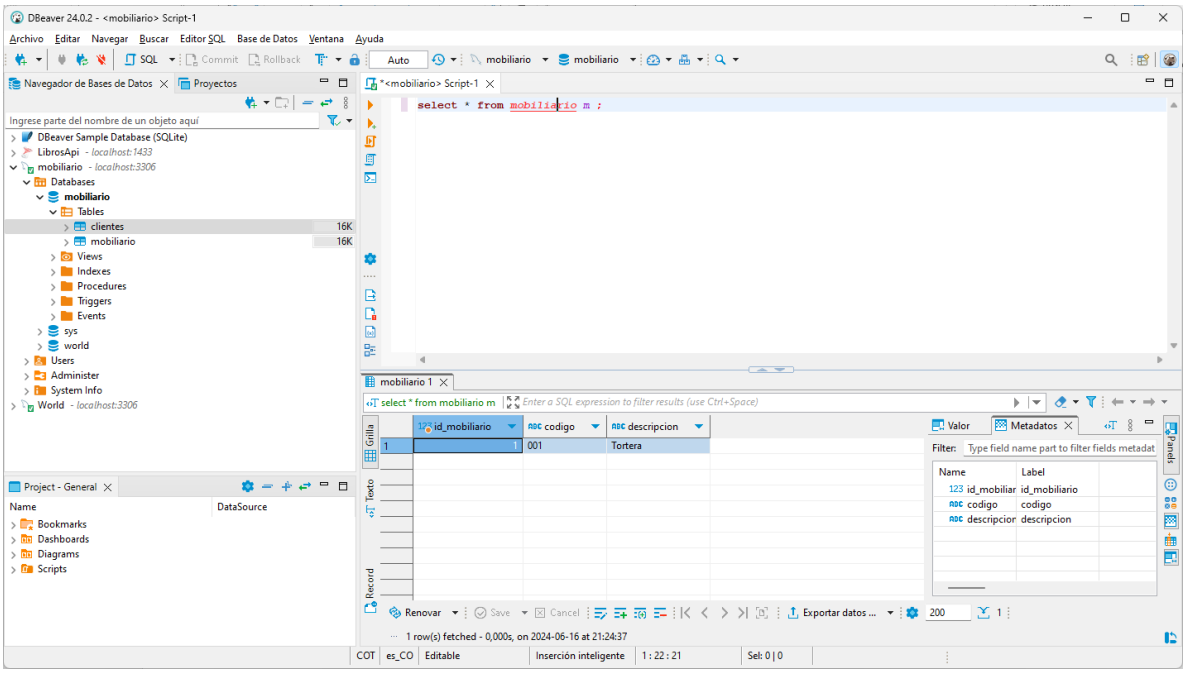
Ingresamos al servicio end-point



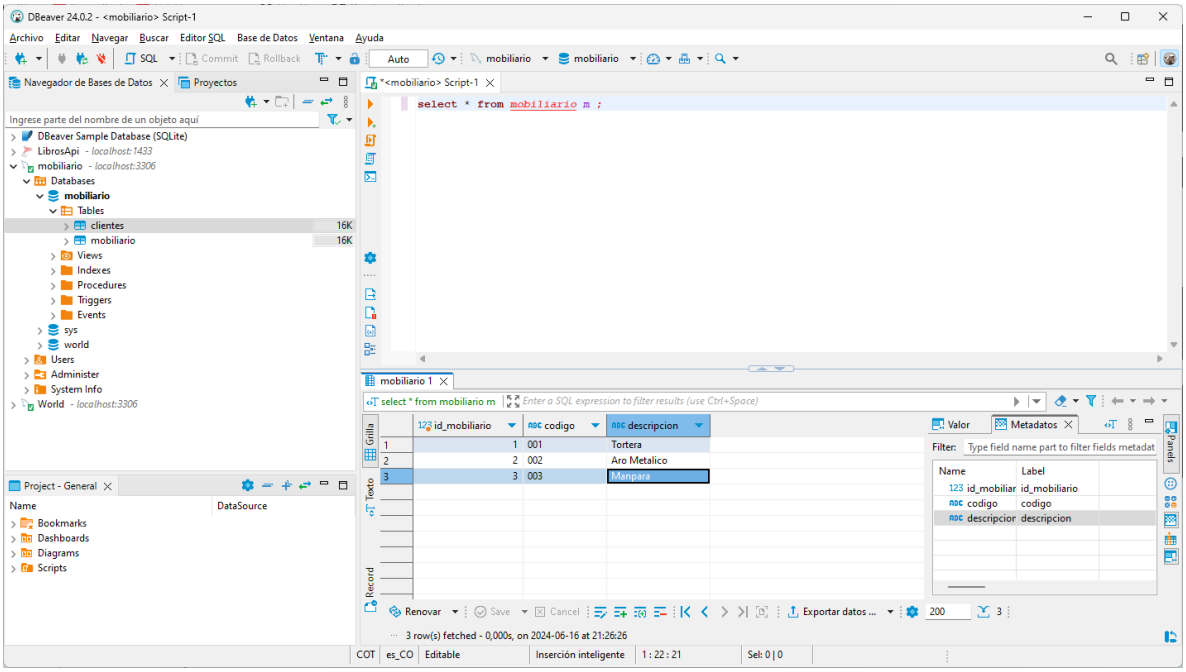
Modificamos los datos para ingresarlos a la base de datos, y por ultimo ejecutar



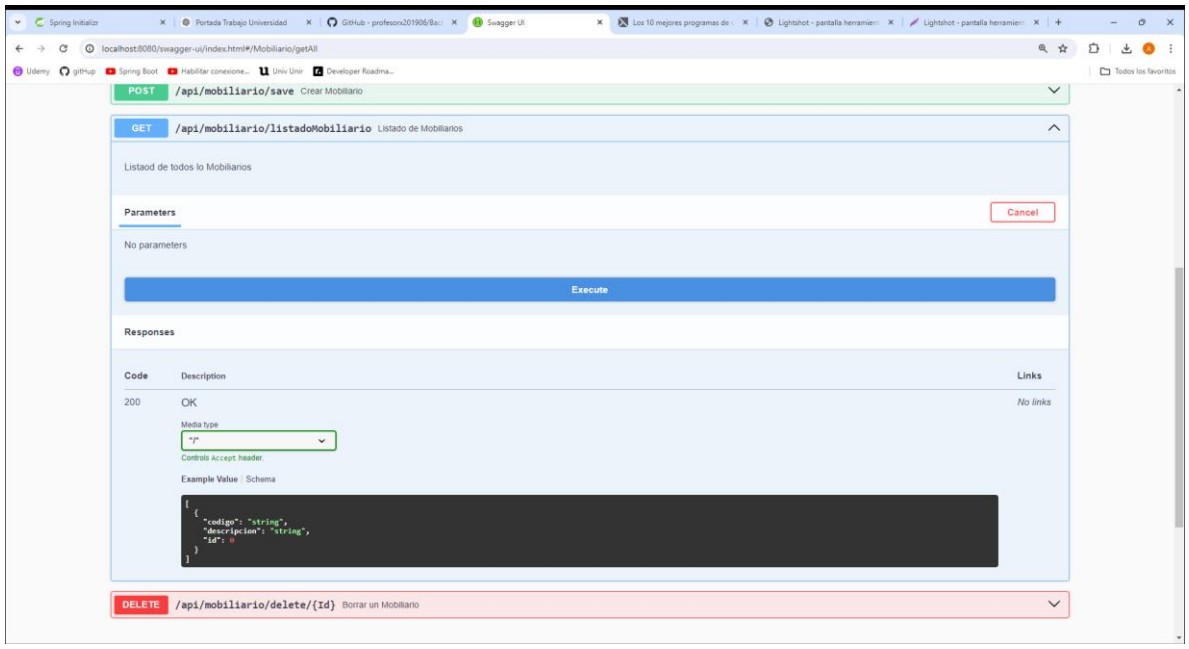
Validamos en la base de datos que se haya creado

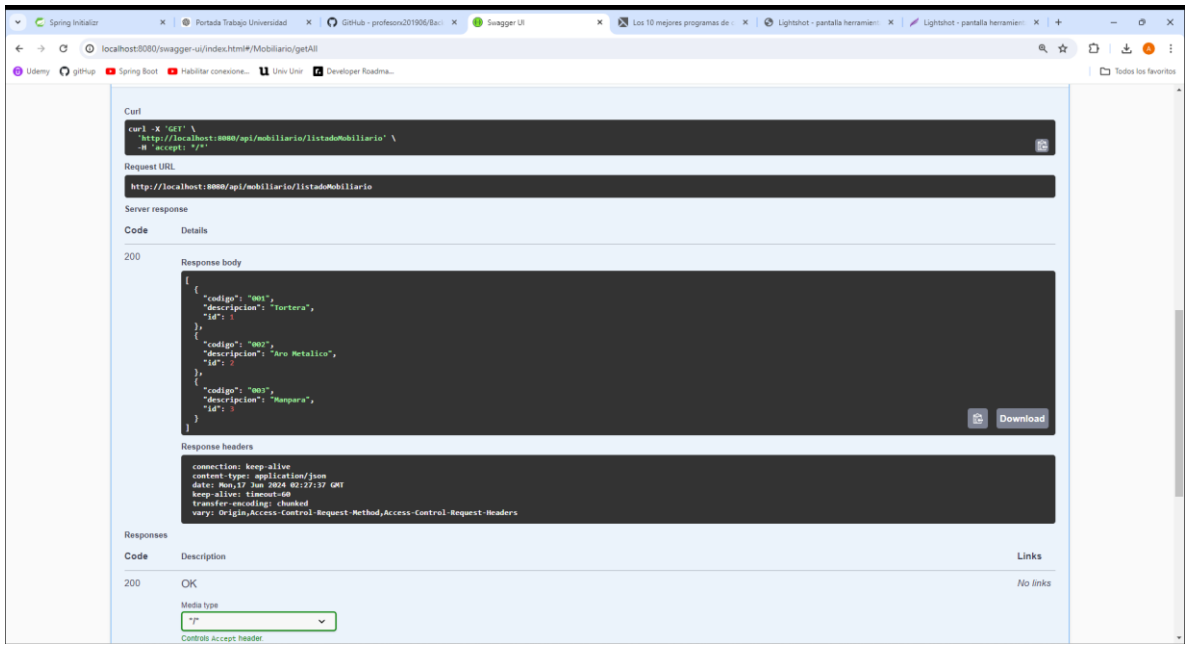


Vemos que se inserto sin problemas, haremos 3 registros mas y con eso haremos los demás procesos de CRUD

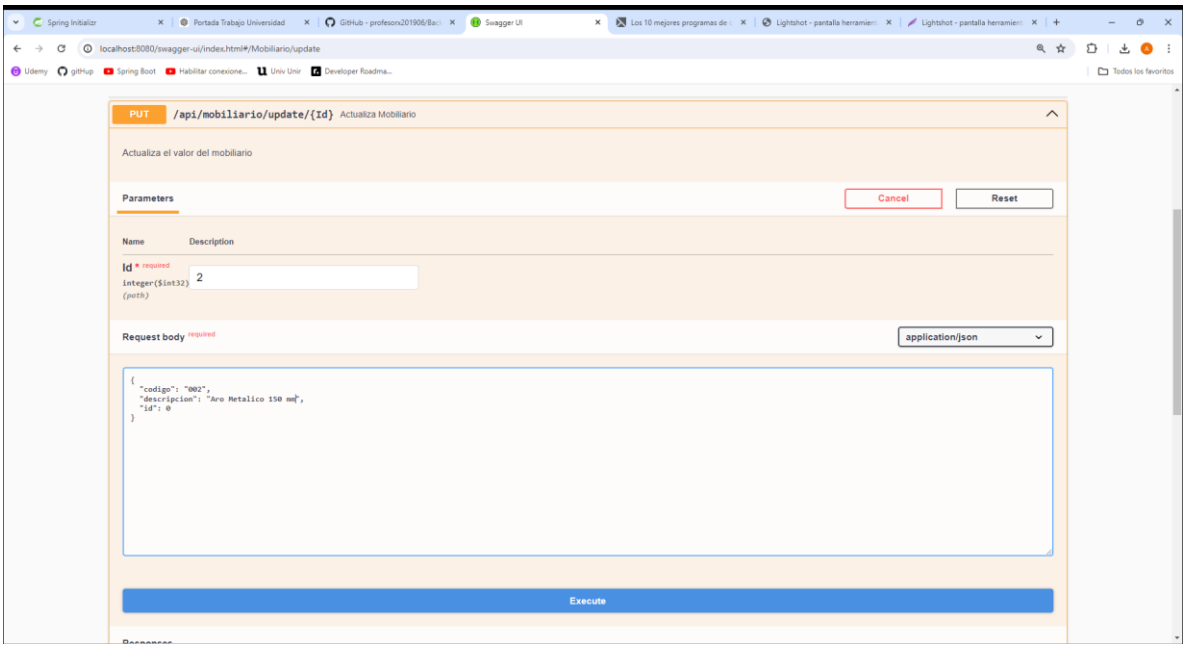


Ahora hacemos el proceso de Listar los mobiliarios

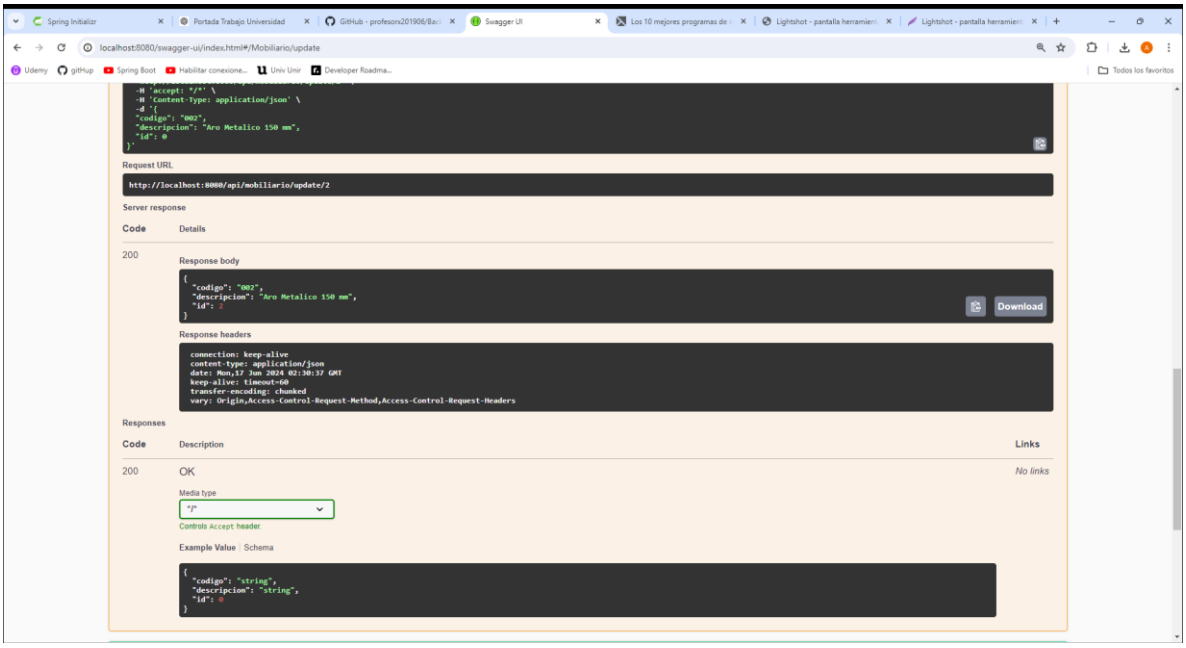


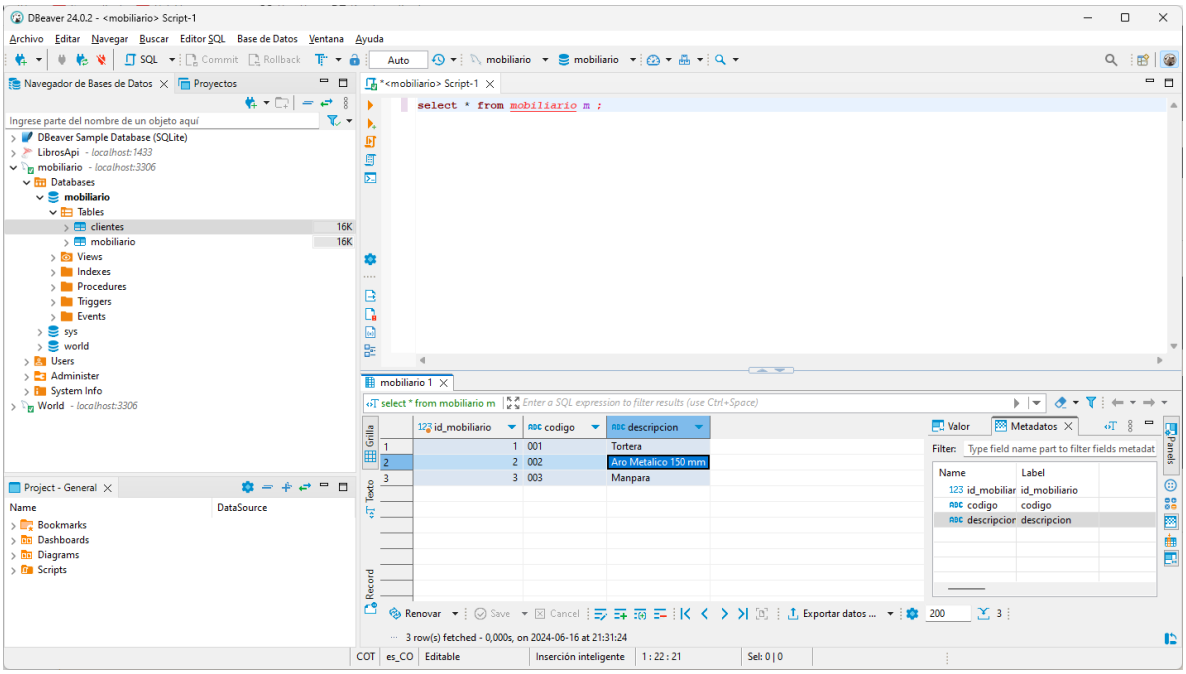


Ahora hacemos el proceso de Actualizar, modificaremos el Aro ingresándole las medidas

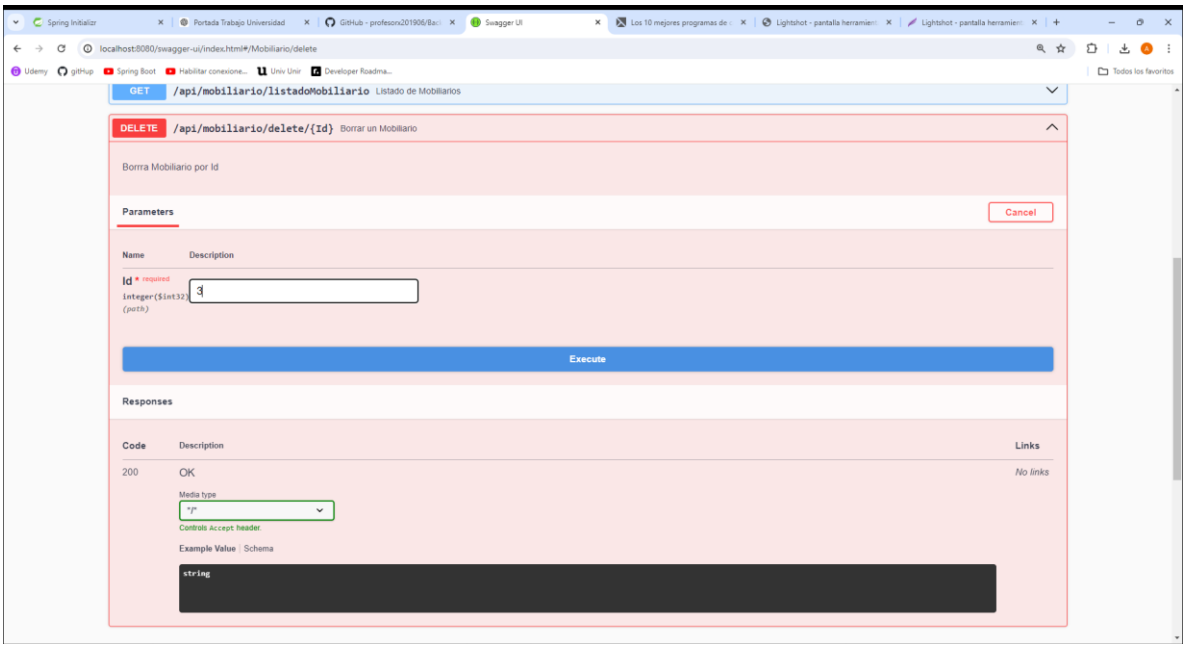


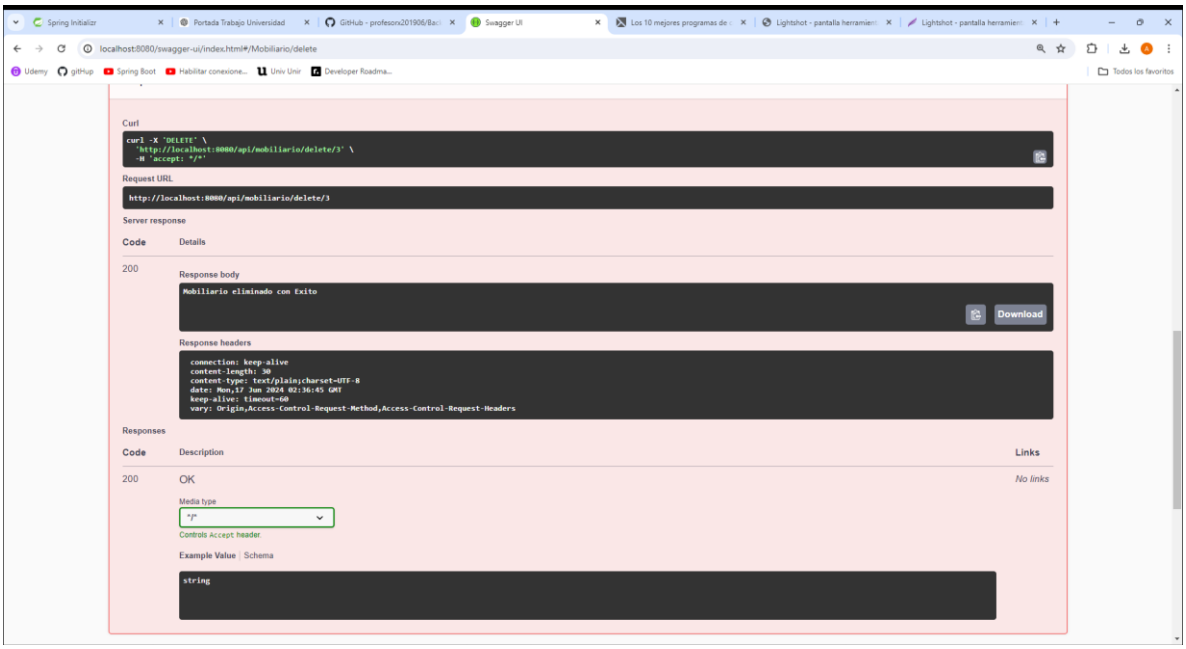
Vemos que se actualizo en la base de datos



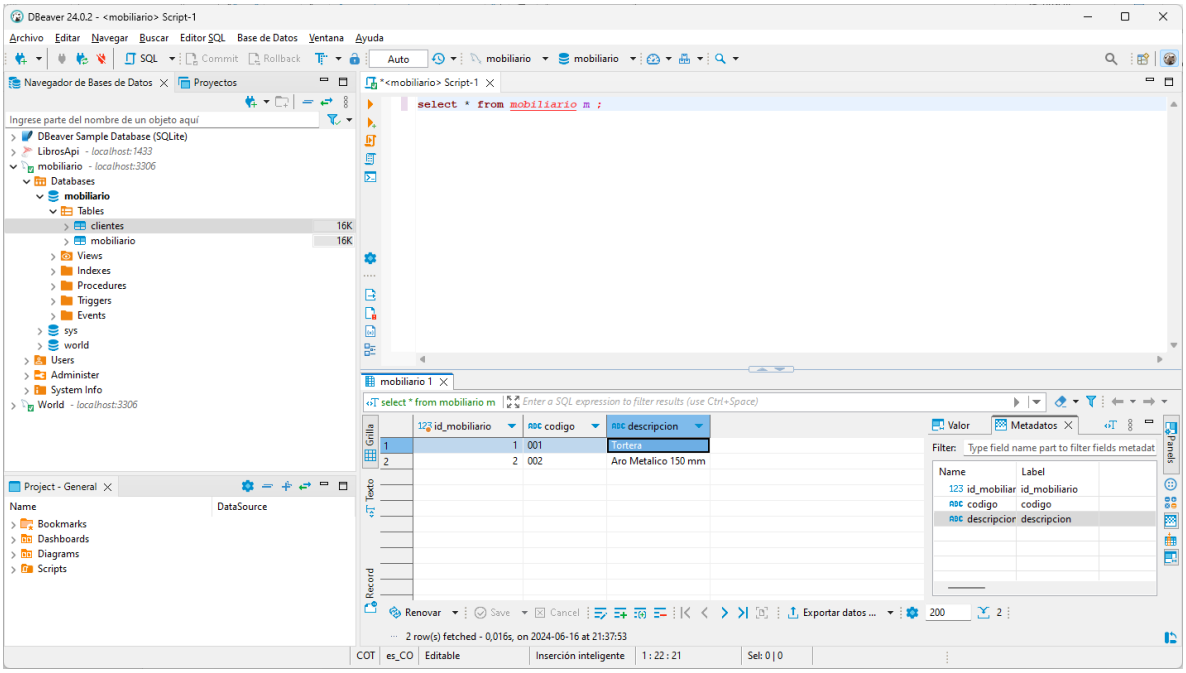


Ahora eliminamos el ultimo registro con el Id 3





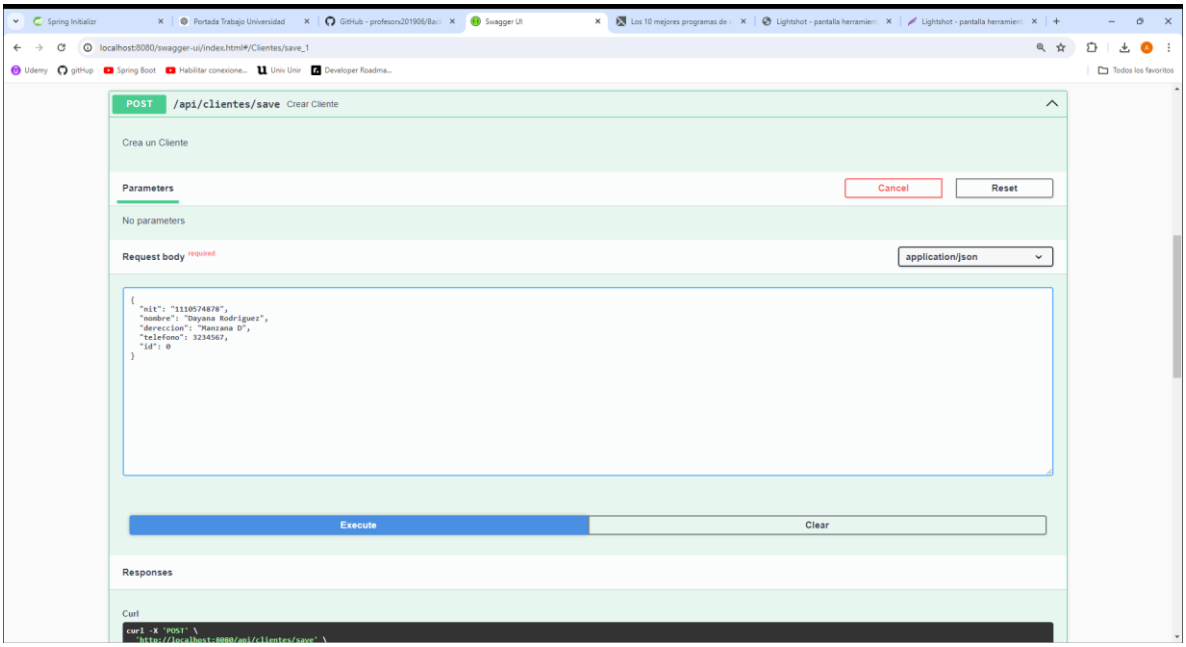
Consultamos en la base de datos para validar que si se haya eliminado.

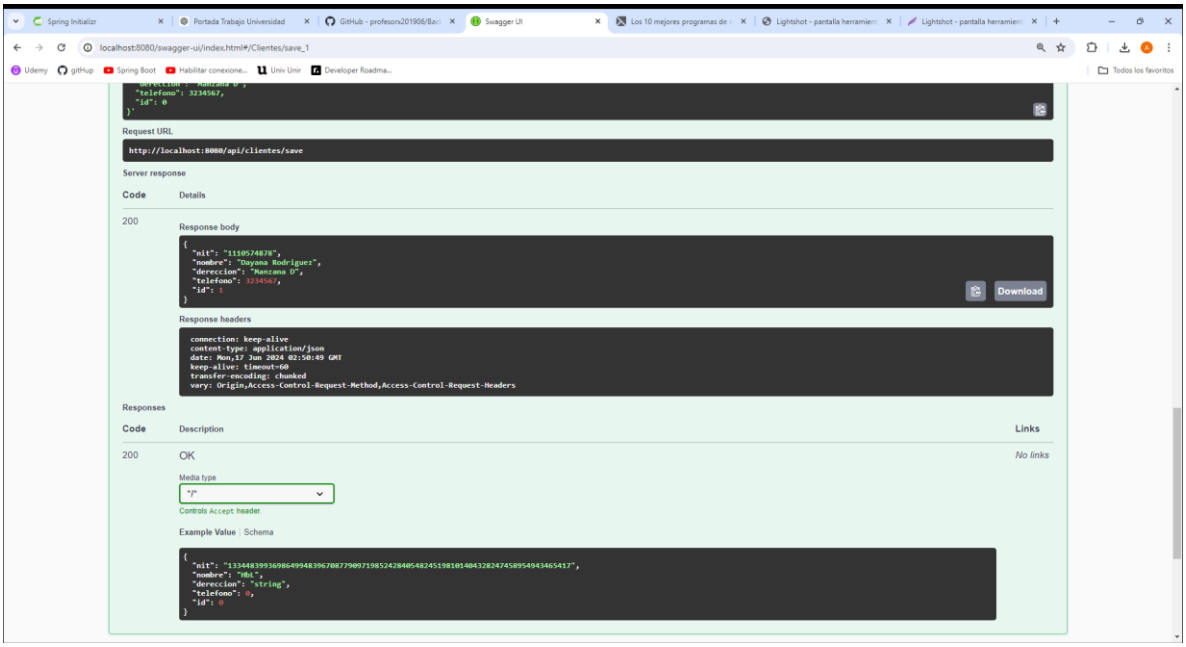


Con esto comprobamos que se esta ejecutando correctamente el proceso de nuestra Api con el Spring Boot, en el siguiente proceso haremos las validaciones de los datos antes de insertar un cliente.

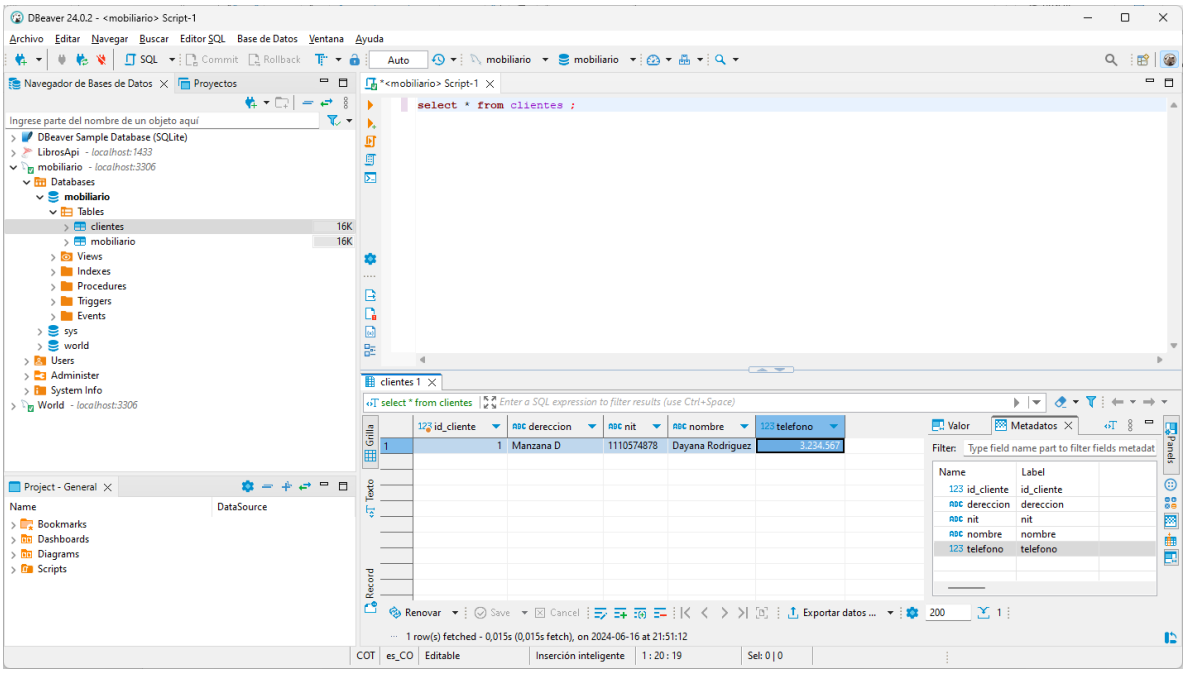
6.4 Detalle de CRUD para los Cliente a alquilar

Ingresamos por Swagger para la gestión del Api Rest de Clientes y tratamos de guardar nuestro primer cliente

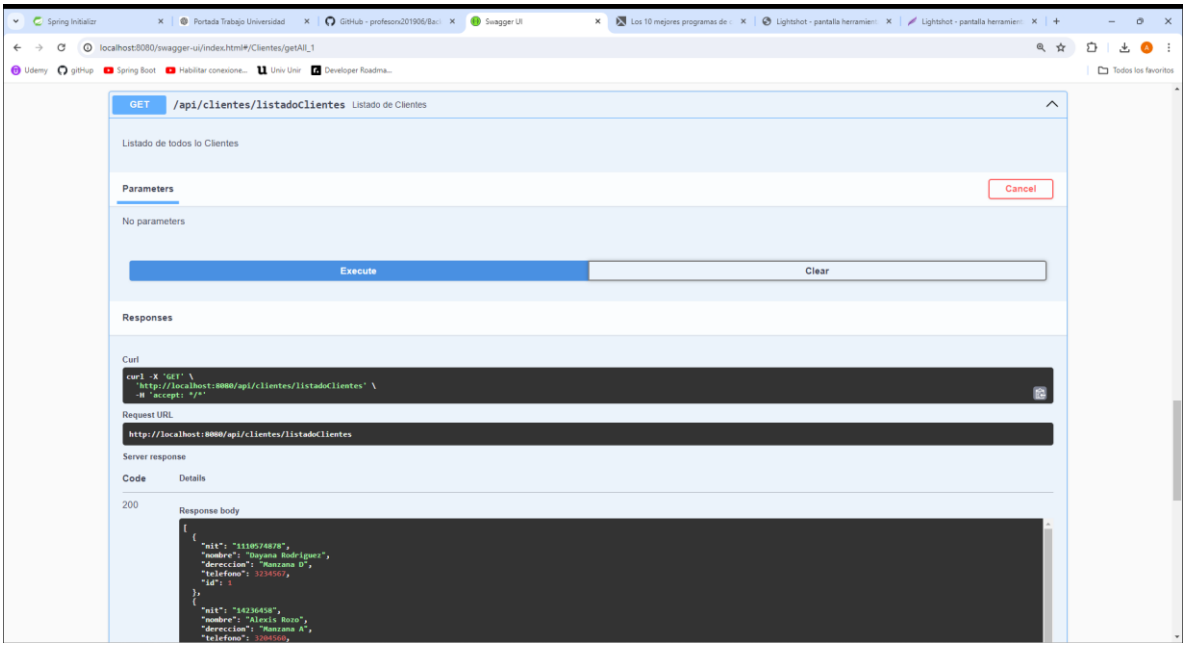


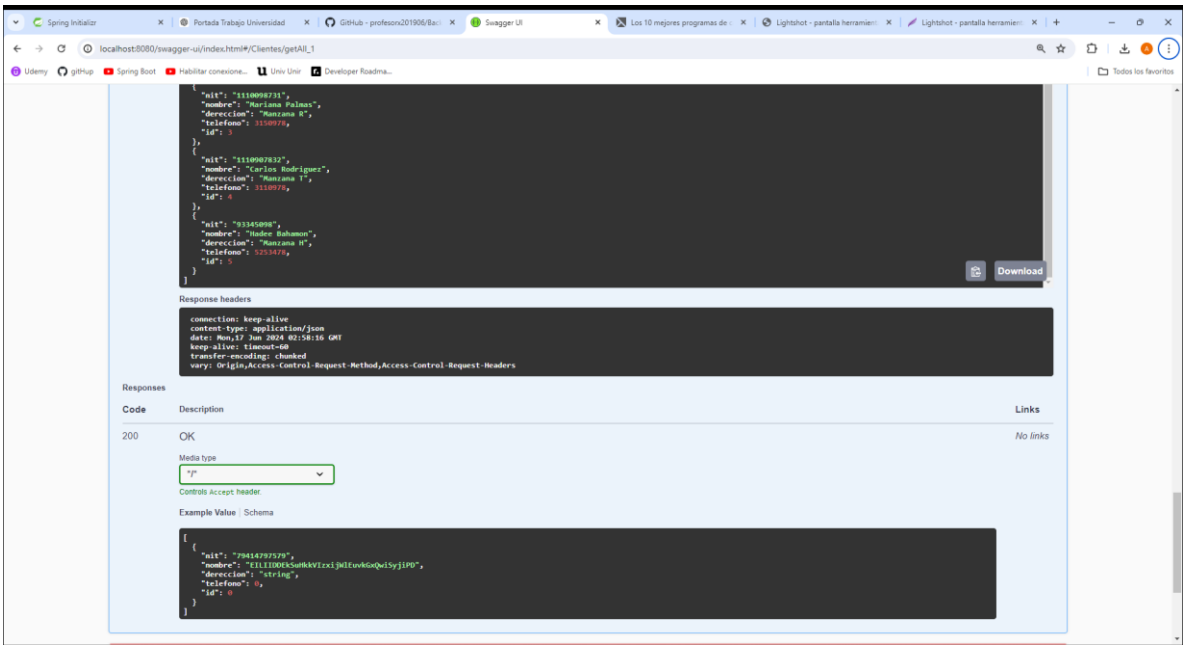


Confirmamos en la base de datos lo guardado

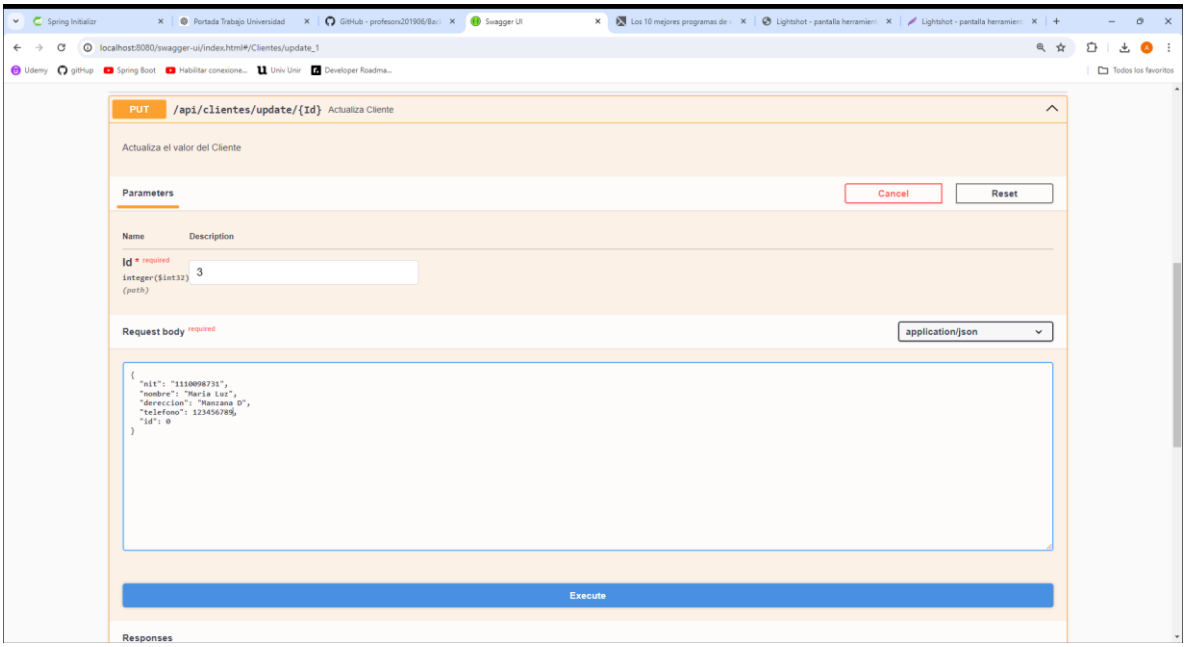


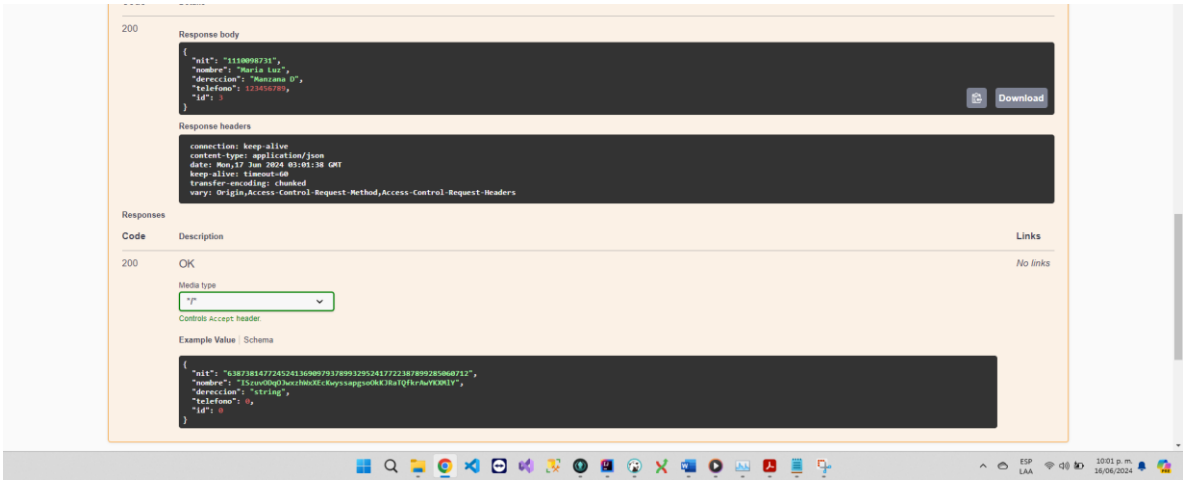
Ahora listamos los clientes



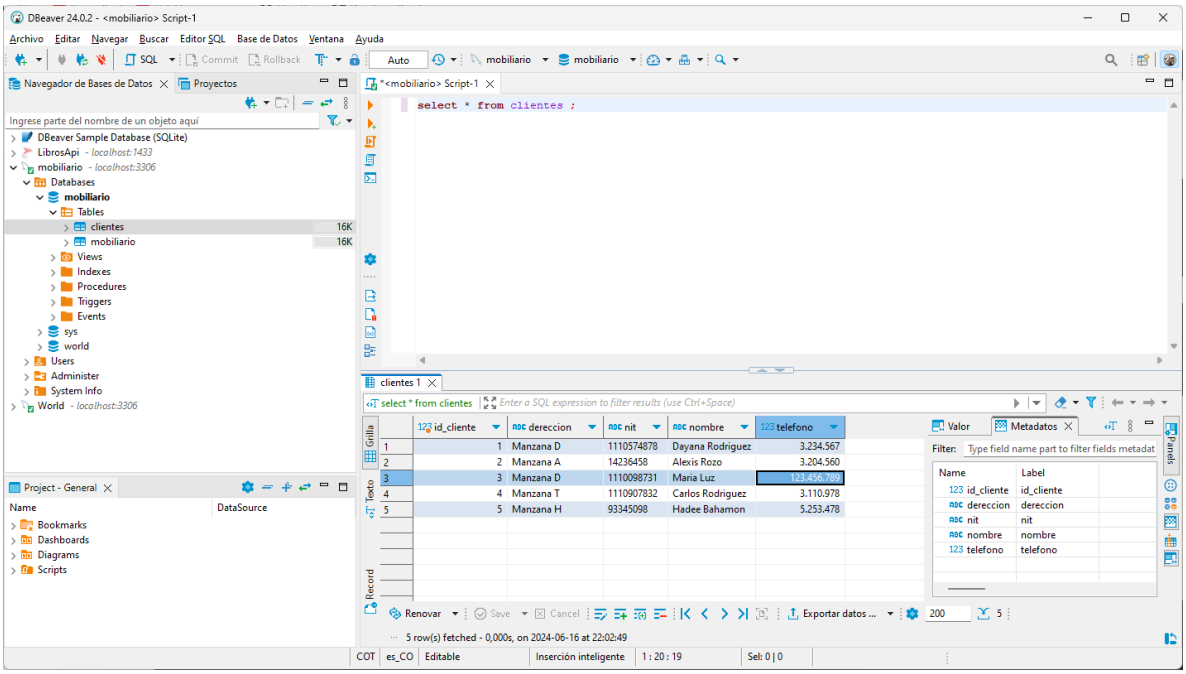


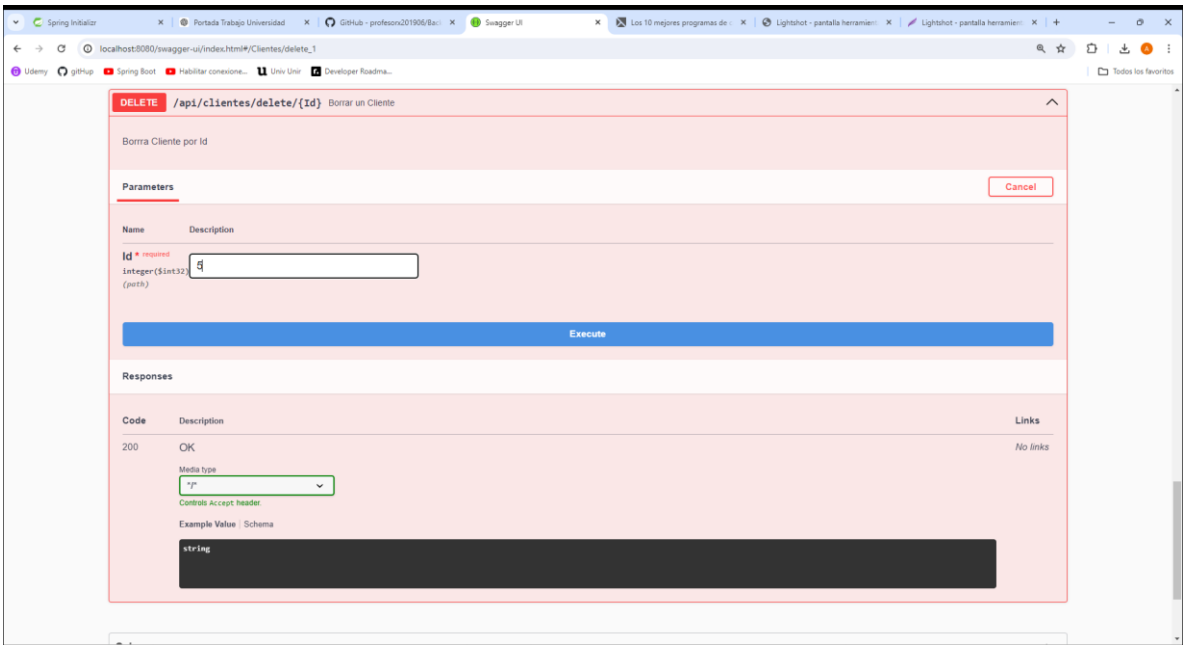
Actualizaremos un cliente

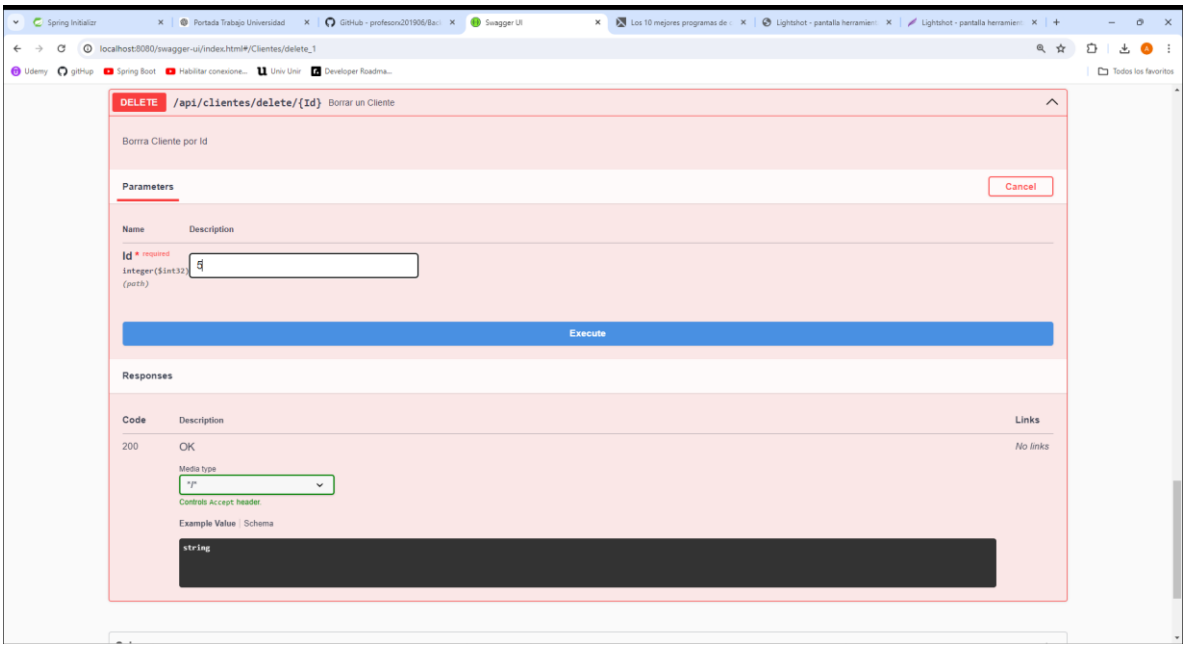




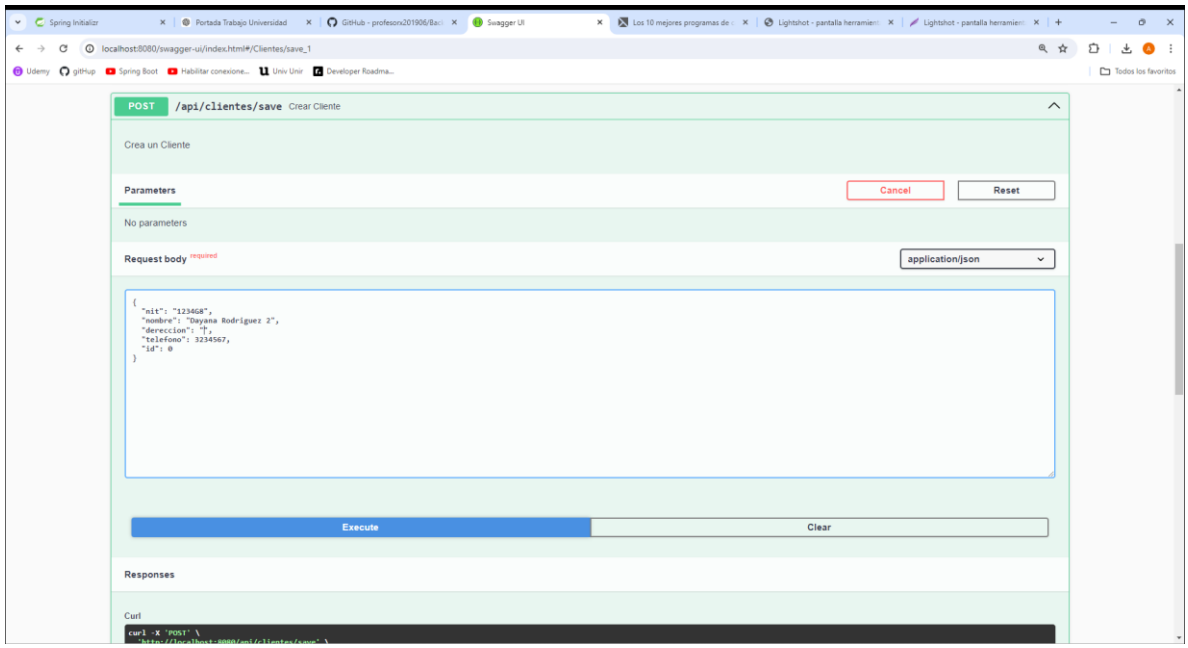
Revisamos en la base de datos




localhost:8080/swagger-ui/index.html#/Clientes/delete_1
DELETE
/api/clientes/delete/{Id}
Borrar un Cliente
Borra Cliente por Id
Parameters
Cancel
Name
Description
Id * required
integer($int32)
(path)
5
Execute
Responses
Code
Description
Links
200
OK
No links
Media type
/
Controls Accept header.
Example Value | Schema
string


localhost:8080/swagger-ui/index.html#/Clientes/delete_1
DELETE
/api/clientes/delete/{Id}
Borrar un Cliente
Borra Cliente por Id
Parameters
Cancel
Name
Description
Id * required
integer($int32)
(path)
5
Execute
Responses
Code
Description
Links
200
OK
No links
Media type
/
Controls Accept header.
Example Value | Schema
string

6.5 Validaciones en las Entidades



Validamos cada campo