Práctica Spring

1.- Introducción:

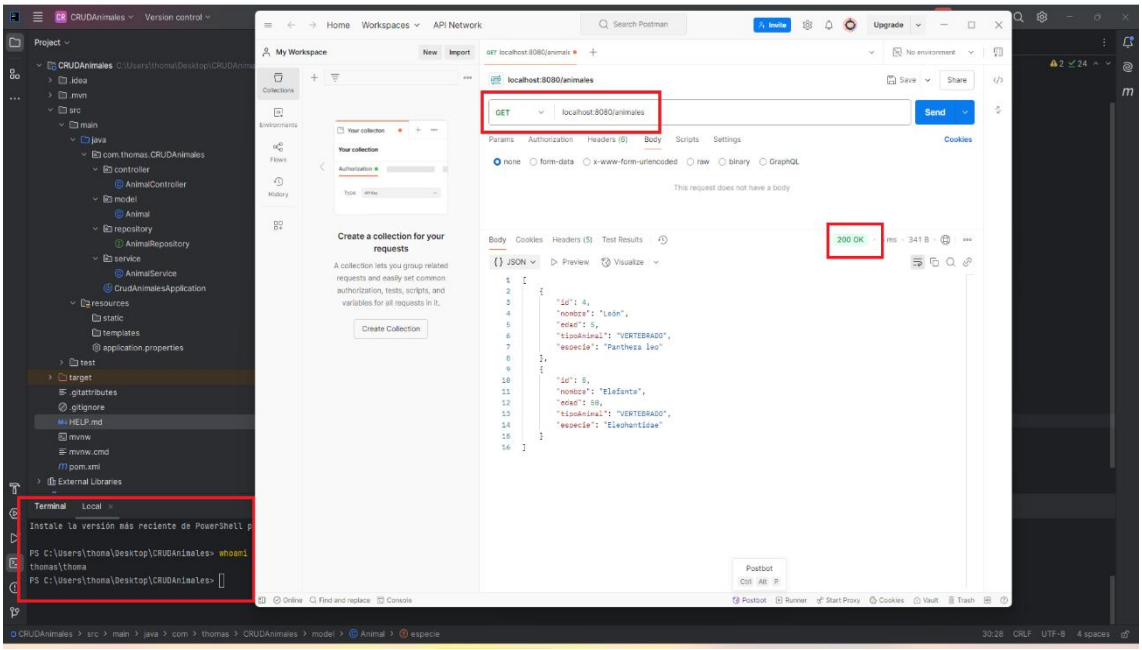
Este documento explica la funcionalidad de la API de gestión de animales desarrollada en Spring Boot. La API permite crear, leer, actualizar y eliminar (CRUD) animales, usando Spring Data JPA para persistencia y OpenAPI para documentación.

2. Explicación de la funcionalidad de la API

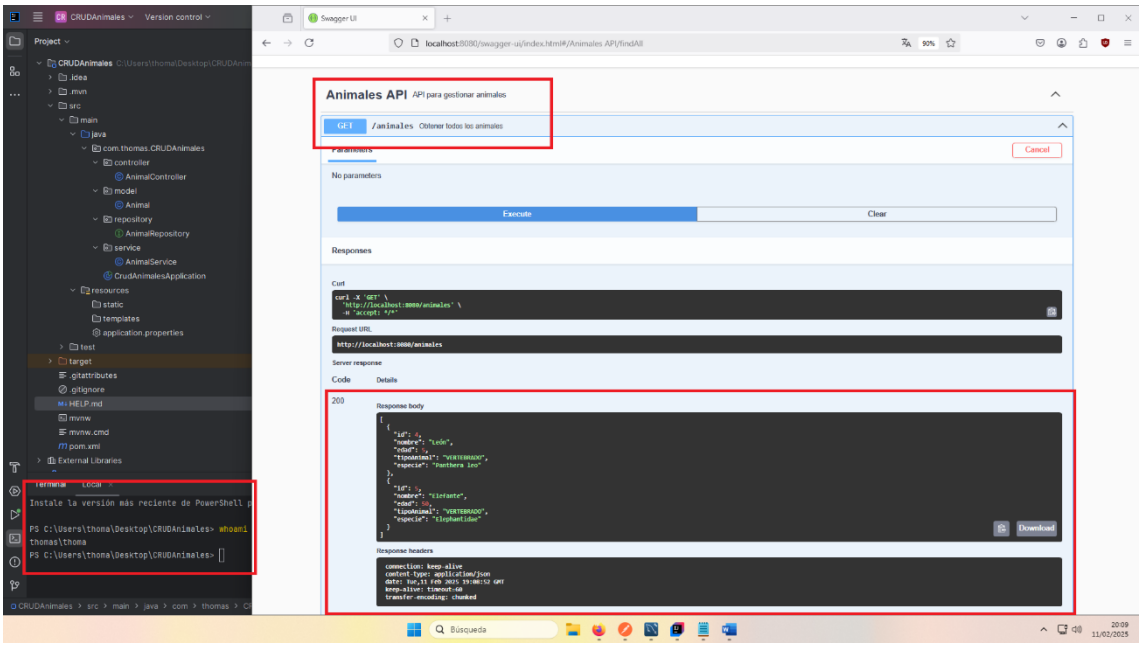
Para poder efectuar el CRUD, tenemos que usar los métodos HTTP como GET, POST, PUT y DELETE. Para probar los resultados añadiré capturas de pantalla del postman (con el intellij de fondo, para que en la esquina izquierda se vea mi nombre) y capturas del Swagger UI para poder acceder a swagger hay que usar el navegador y poner esto: **http://localhost:8080/swagger-ui.html**

2.1. Obtener todos los animales (GET)

- **Descripción**: Este método permite recuperar una lista de todos los animales almacenados en la base de datos.
- **Método HTTP**: GET
- **Ruta**: /animales
- **Respuesta esperada**: Un objeto JSON que contiene una lista de todos los animales, con sus atributos como id, nombre, tipoAnimal, especie y edad.
- **Captura de pantalla:** En Postman:

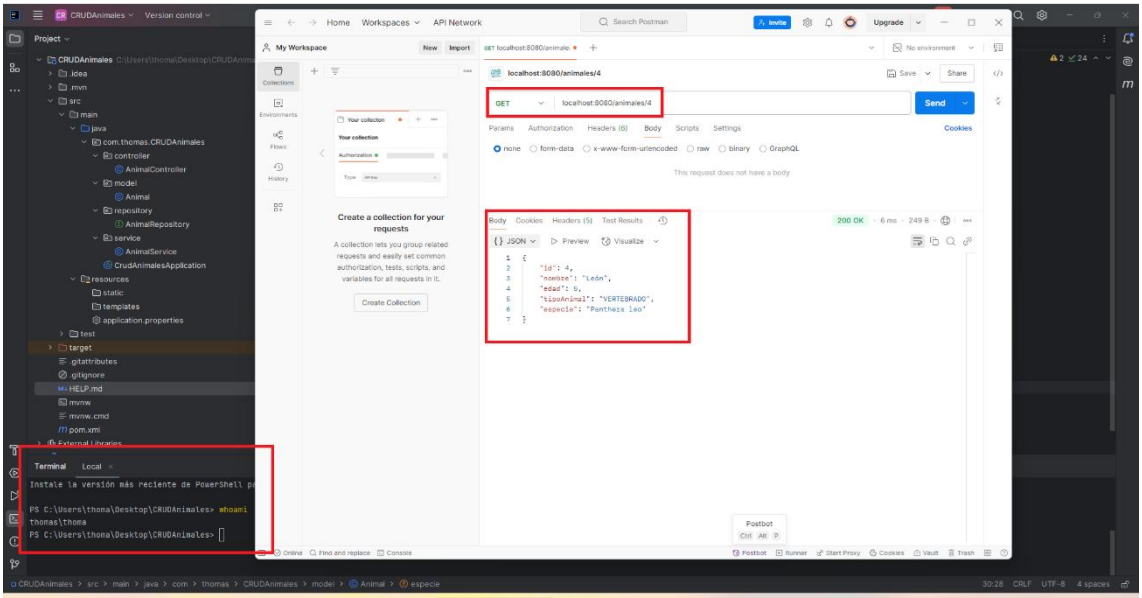


Y ahora en Swagger:

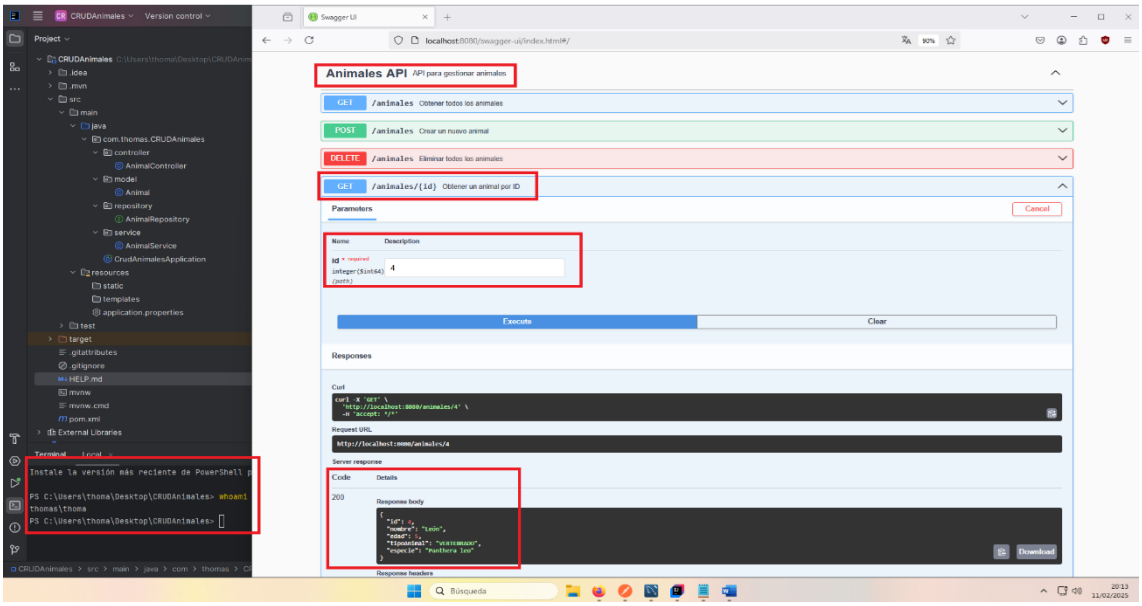


2.2 Obtener un animal por ID (GET)

- **Descripción**: Permite obtener los detalles de un animal específico utilizando su id.
- **Método HTTP**: GET
- **Ruta**: /animales/{id}
- **Parámetros**: El id del animal a consultar.
- **Respuesta esperada**: Un objeto JSON con la información del animal, o un mensaje de error si no se encuentra el animal con el ID proporcionado.
- **Captura de pantalla**: En Postman:



Y ahora en Swagger:



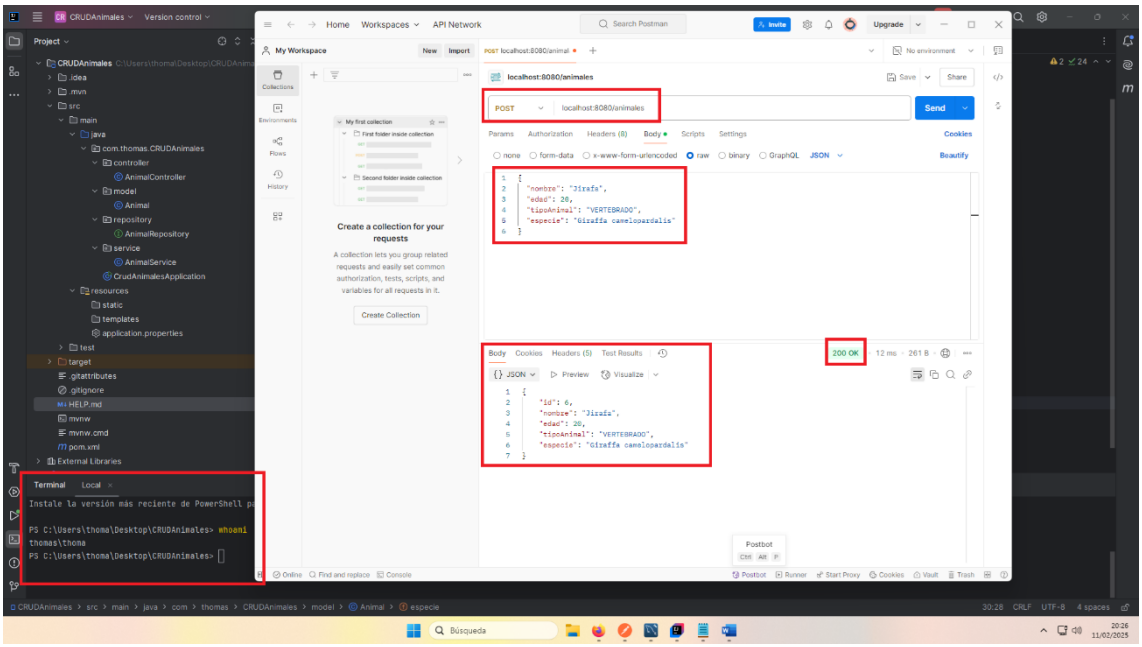
2.3. Crear un nuevo animal (POST)

- **Descripción**: Este método permite agregar un nuevo animal a la base de datos, proporcionando los detalles necesarios como nombre, tipoAnimal, especie, y edad.
- **Método HTTP**: POST
- **Ruta**: /animales
- **Cuerpo de la solicitud:** Un objeto JSON que incluye los atributos del animal.

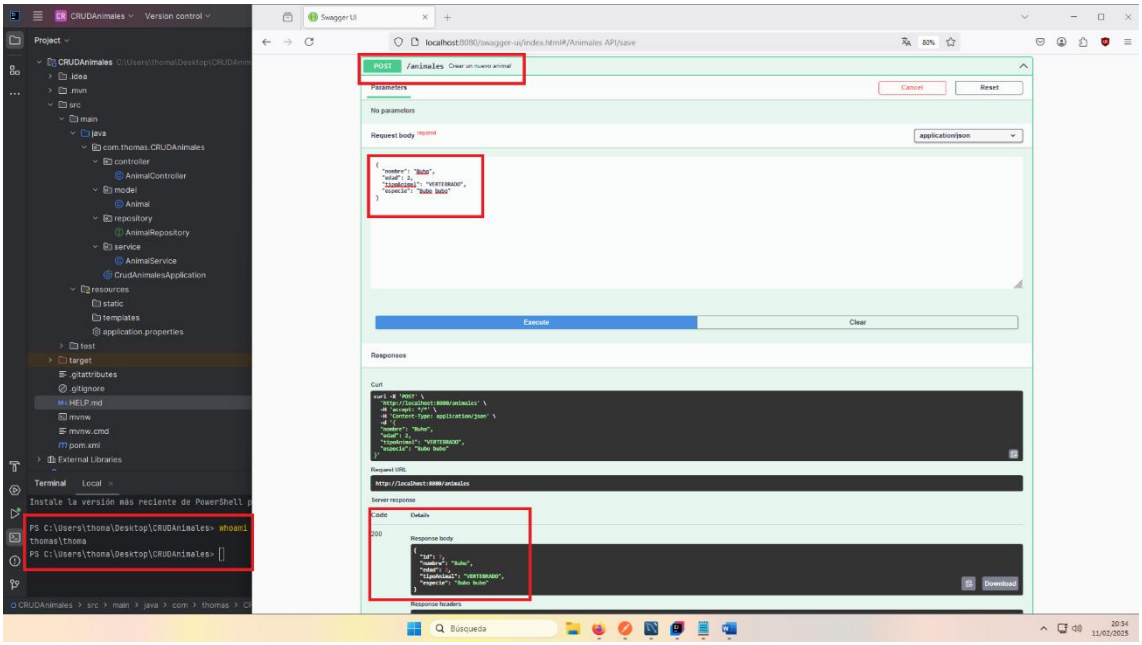
```
{
  "nombre": "Jirafa",
  "edad": 20,
  "tipoAnimal": "VERTEBRADO",
  "especie": "Giraffa camelopardalis"
}
```

- **Respuesta esperada: El animal recién creado, con un id asignado automáticamente.**

Captura de pantalla: En Postman:

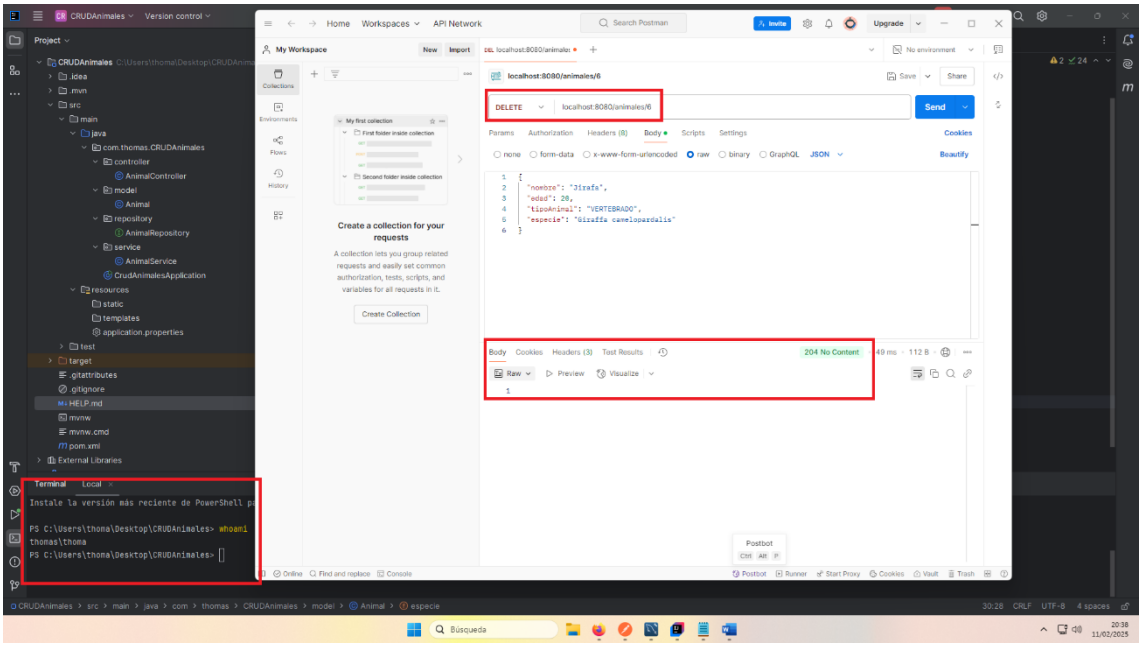


Y ahora en swagger:

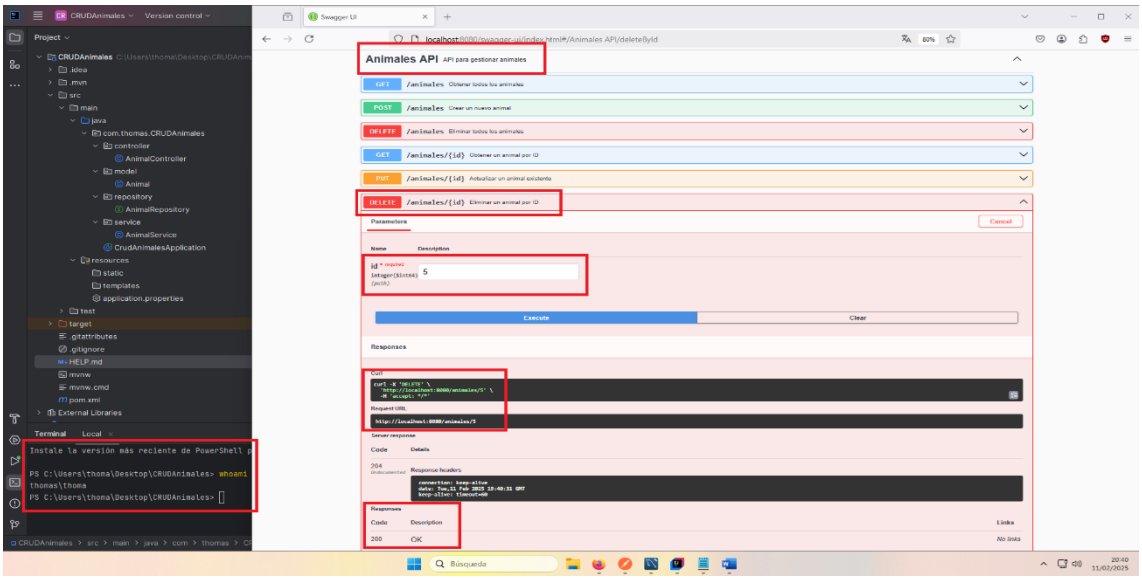


2.5. Eliminar un animal por ID (DELETE)

- **Descripción**: Este método elimina un animal específico de la base de datos utilizando su id.
- **Método HTTP**: DELETE
- **Ruta**: /animales/{id}
- **Parámetros**: El id del animal a eliminar.
- **Respuesta esperada**: Un mensaje que indica que el animal ha sido eliminado correctamente. En caso de que el animal no exista, se devolverá un error.
- **Captura de pantalla**: En Postman:

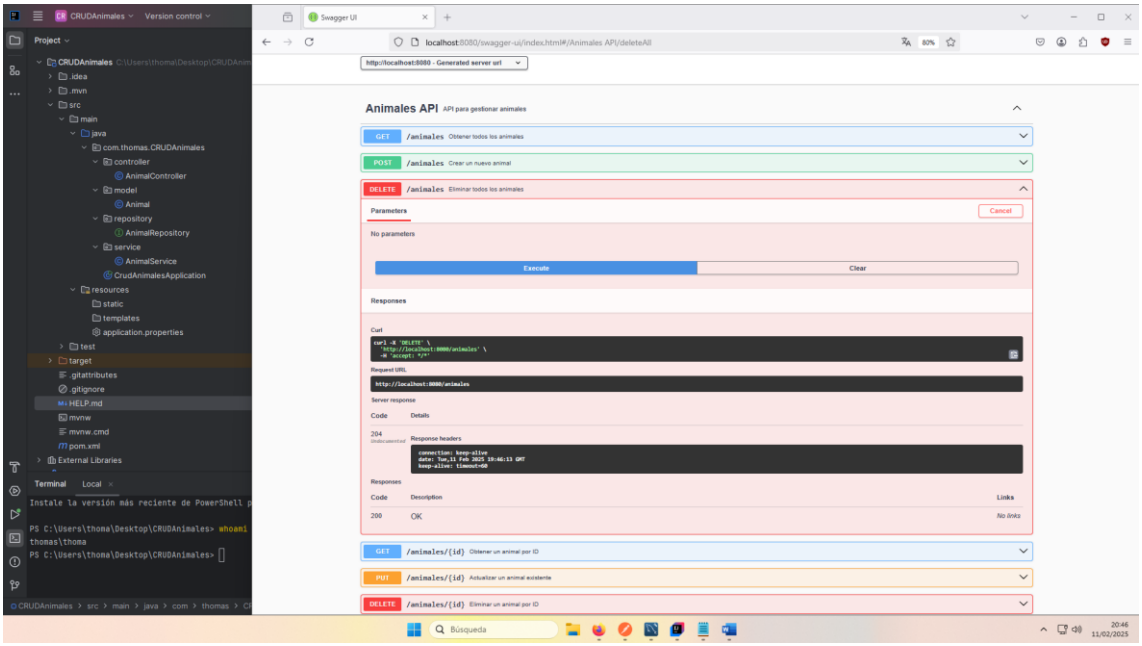


Y ahora en Swagger:



2.6. Eliminar todos los animales (DELETE)

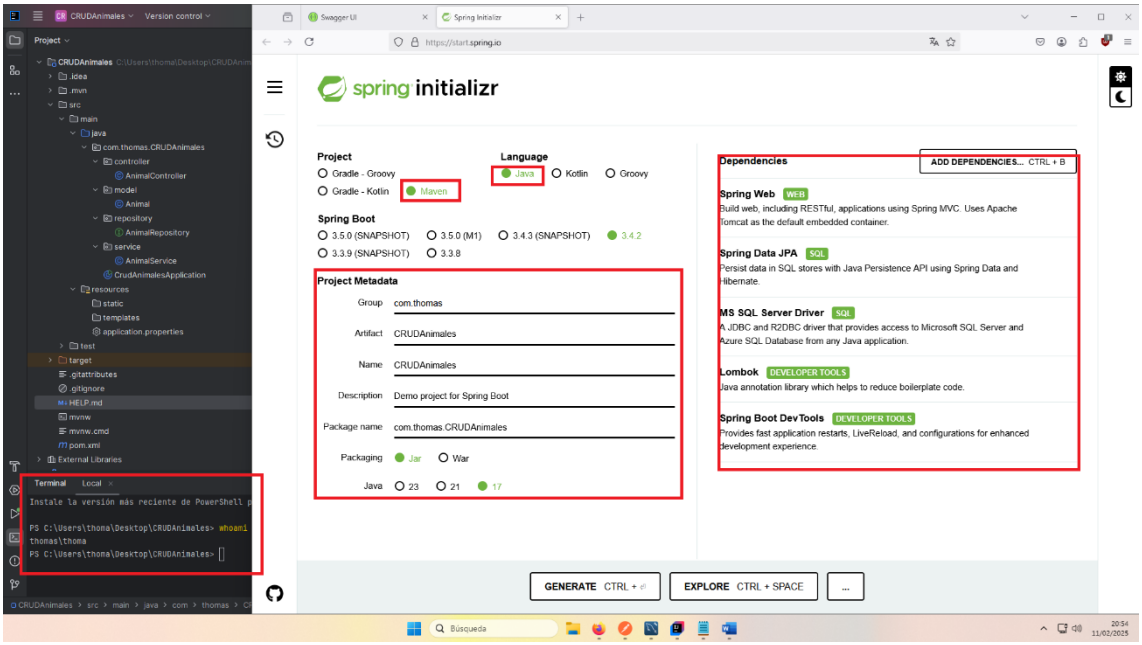
- **Descripción**: Este método permite eliminar una lista de todos los animales almacenados en la base de datos.
- **Método HTTP**: DELETE
- **Ruta**: /animales
- **Respuesta esperada**: Un mensaje que indica que los animales han sido eliminados correctamente. En caso de que los animales no existan, se devolverá un error.
- **Captura de pantalla**: En Swagger:



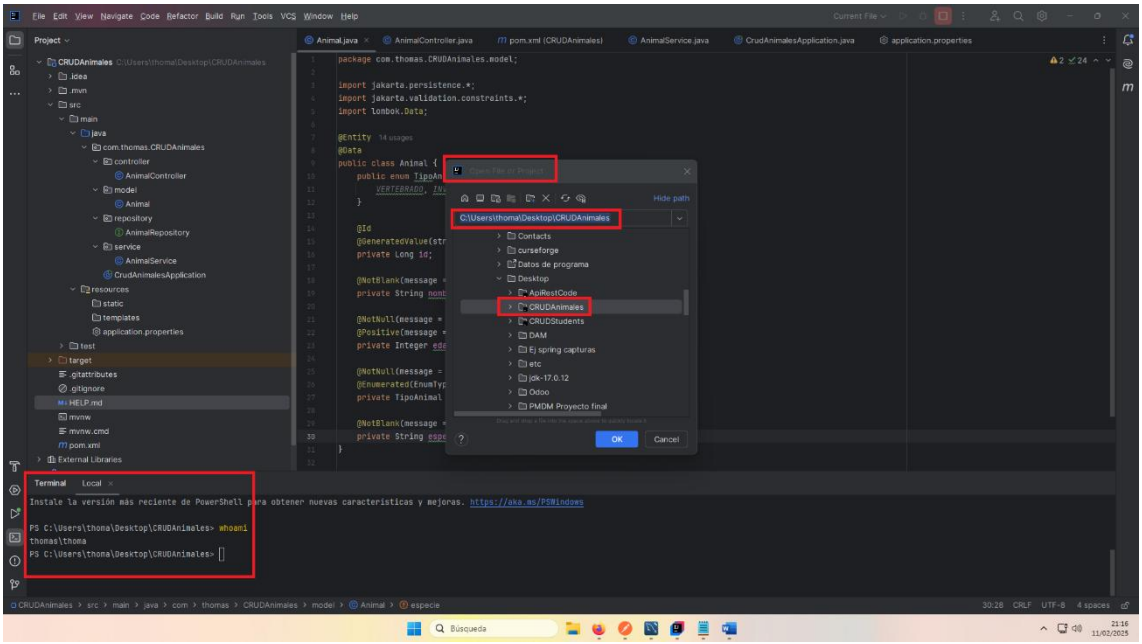
3. Pasos Seguidos en el Desarrollo

Paso 1: Configuración del proyecto:

Primero, configuré un proyecto Spring Boot usando Maven, el lenguaje Java, he puesto el nombre al proyecto “CRUDAnimales” y con las dependencias necesarias para trabajar con Spring Web, Spring Data JPA, MySQL, Lombok, Spring Boot DevTools y Swagger. He utilizado la página web: https://start.spring.io/



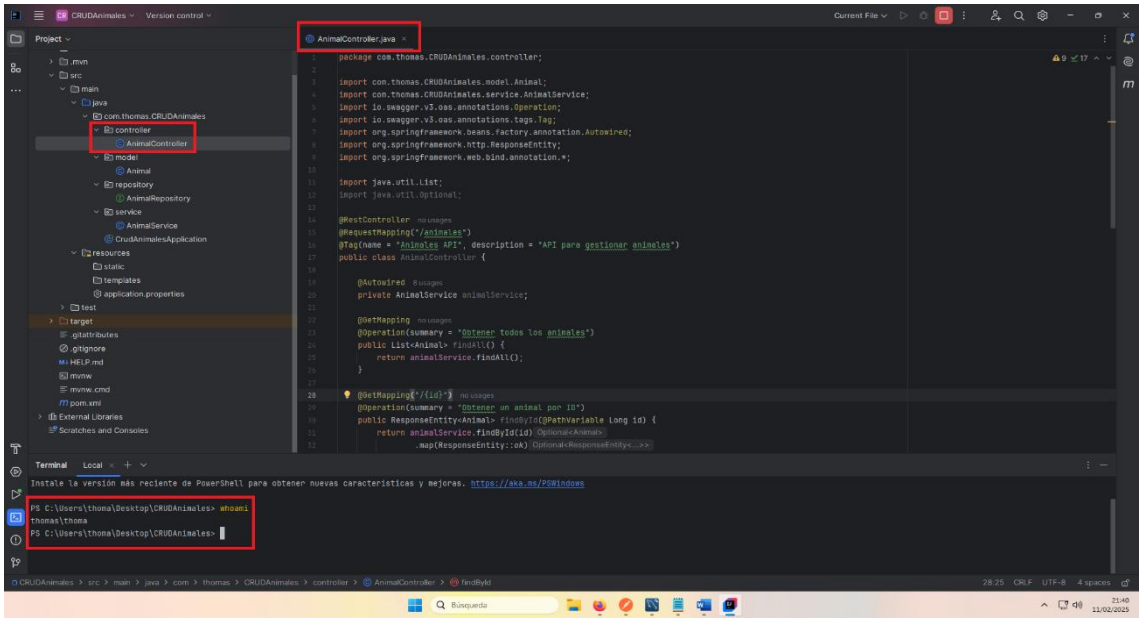
Le das a “GENERATE” y se te descargará un archivo .zip, hay que descomprimirlo en el escritorio y abrir el proyecto con el IDE, en este caso lo hemos hecho en Intellij

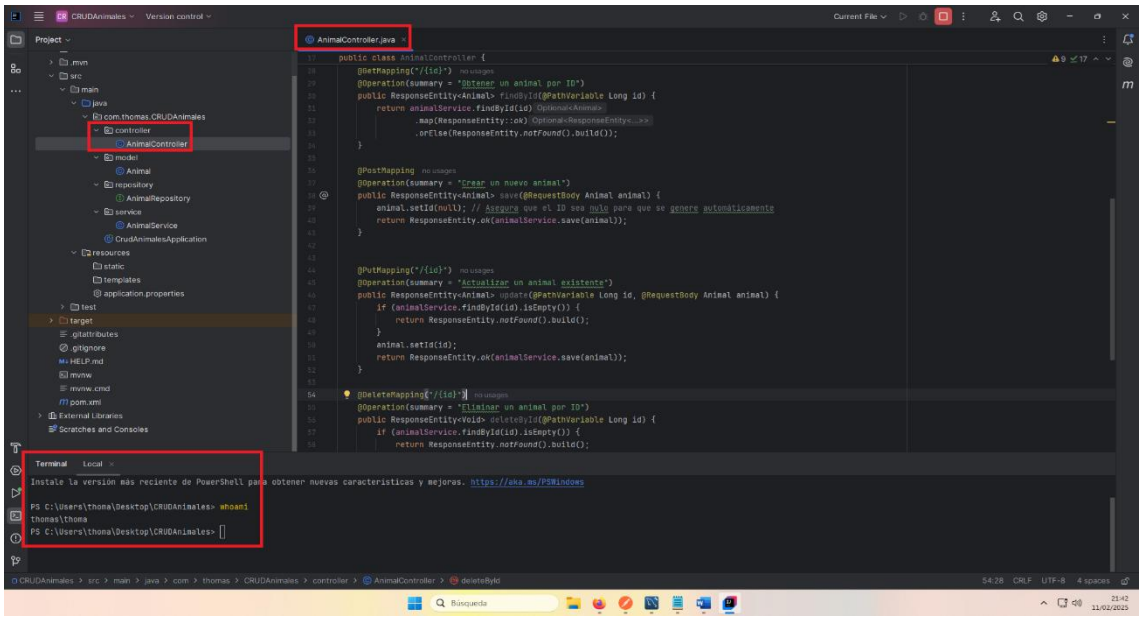


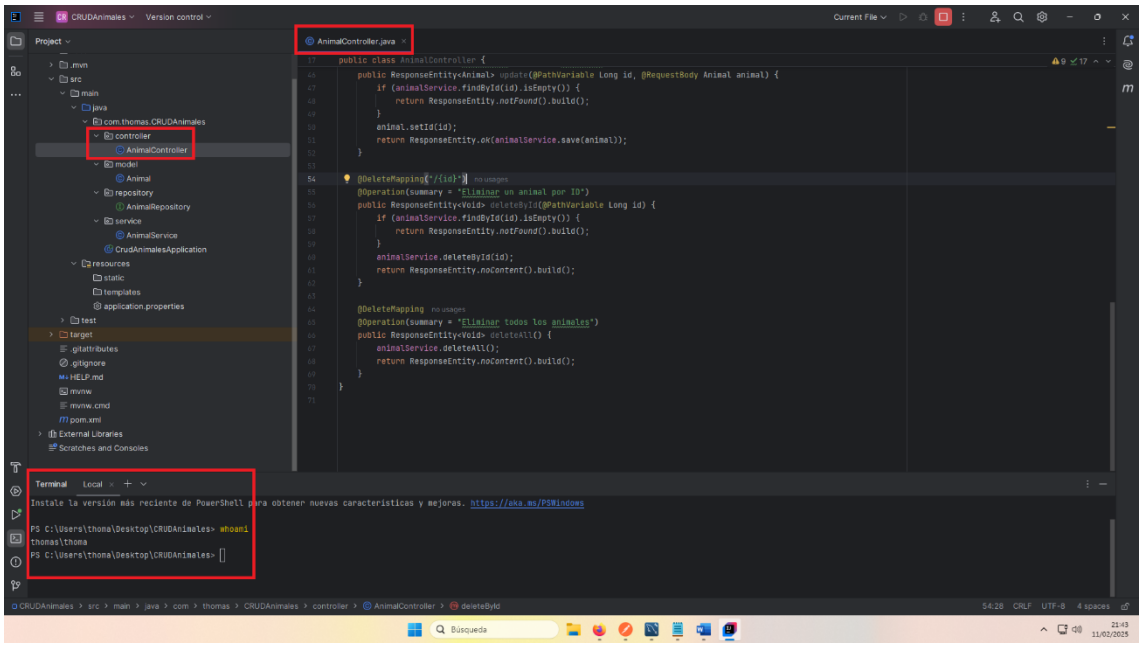
Paso 2: Estructura del Proyecto:

Organicé el proyecto usando los siguientes paquetes:

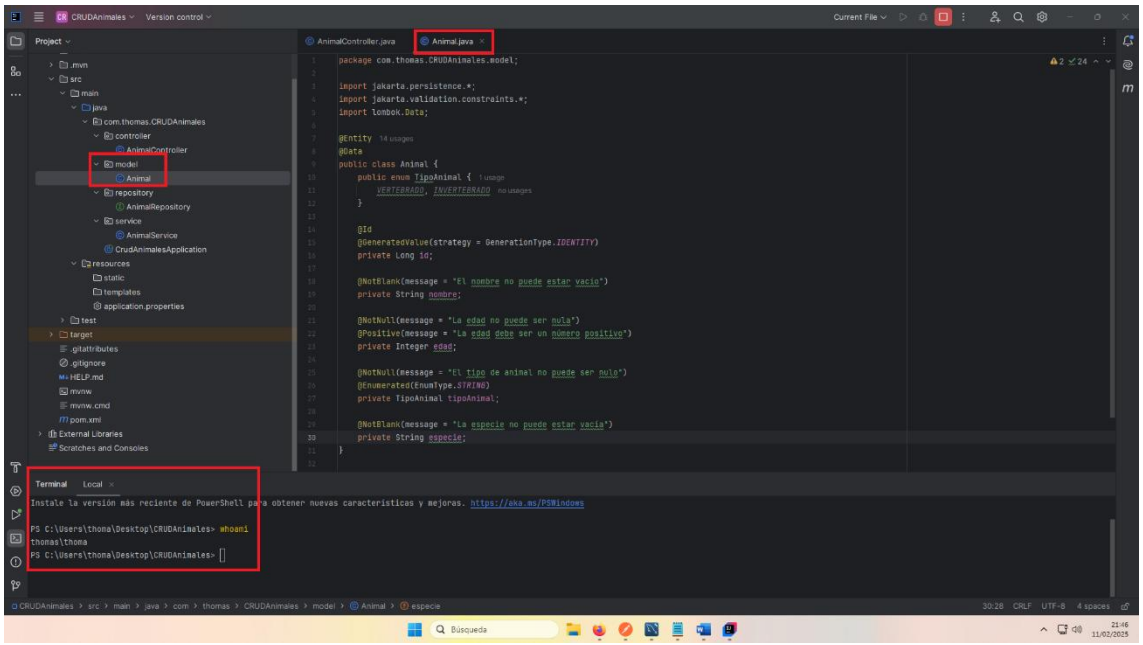
- **controller:** Contiene AnimalController, la clase AnimalController maneja las solicitudes HTTP relacionadas con los animales. Usa anotaciones como:
  @RestController y @RequestMapping("/animales") para definir los endpoints.
  @GetMapping obtiene todos los animales o uno por ID.
  @PostMapping crea un nuevo animal.
  @PutMapping("/{id}") actualiza un animal existente.
  @DeleteMapping("/{id}") elimina un animal por ID.
  Se usó **ResponseEntity** para manejar respuestas adecuadas en cada caso.



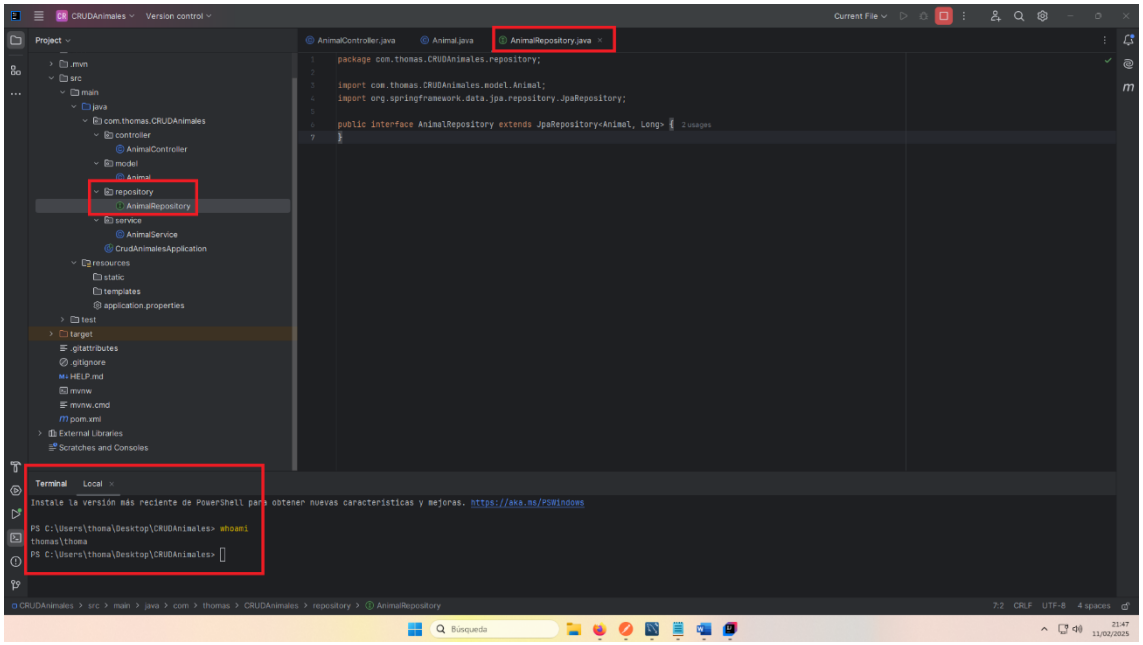




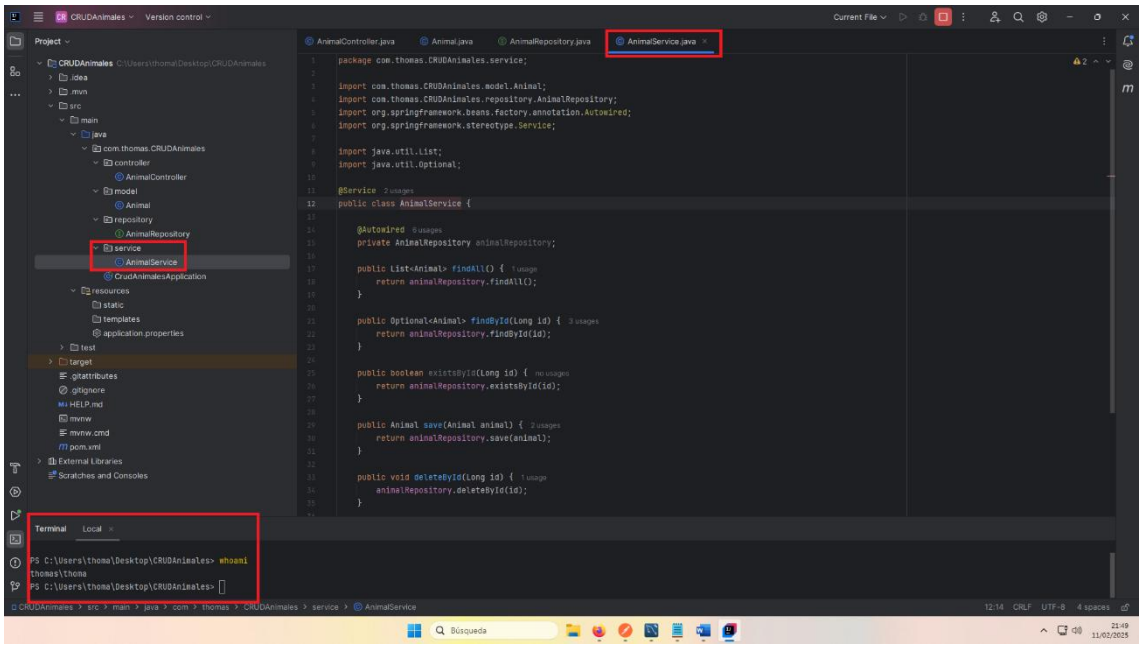
- **model:** Contiene Animal, la entidad **Animal** en la base de datos. Se usa **@Entity** para mapear la clase a una tabla, y sus atributos como nombre,edad, tipoAnimal y especie están anotados con validaciones **(@NotNull, @Positive, @NotBlank** y **@Enumerated**.). También tiene un **@Id** con **@GeneratedValue** para generar el ID automáticamente.



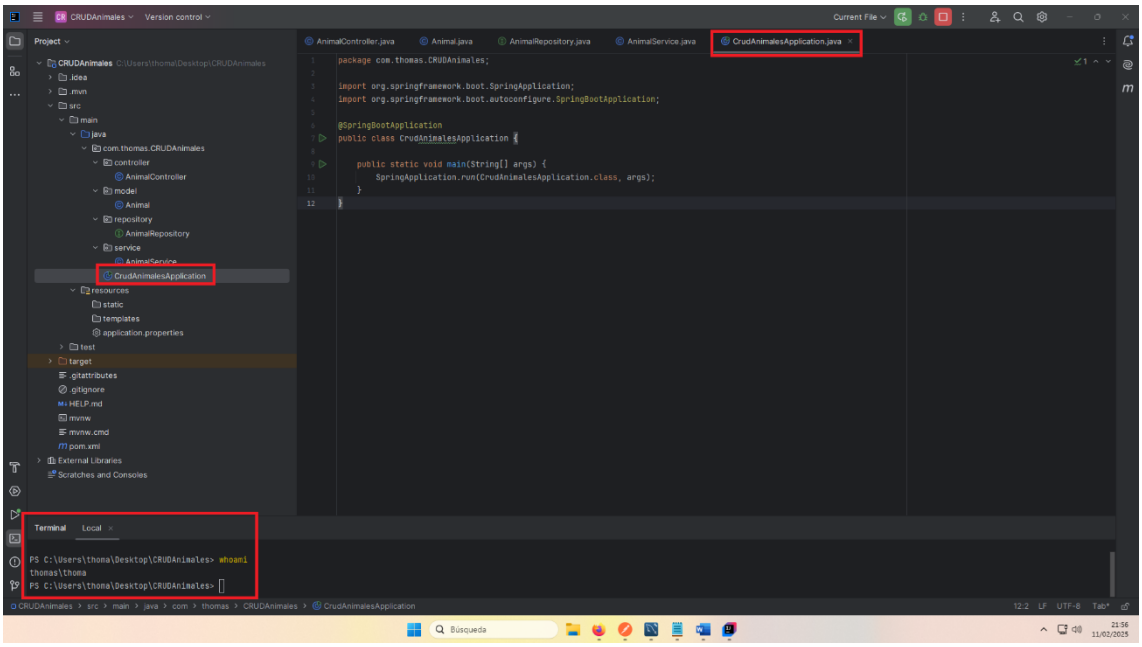
- **repository:** Contiene AnimalRepository, es una interfaz que extiende **JpaRepository<Animal, Long>,** lo que permite interactuar con la base de datos sin escribir consultas SQL. Proporciona métodos como **findById(id), findAll(), save(animal)** y **deleteById(id).**



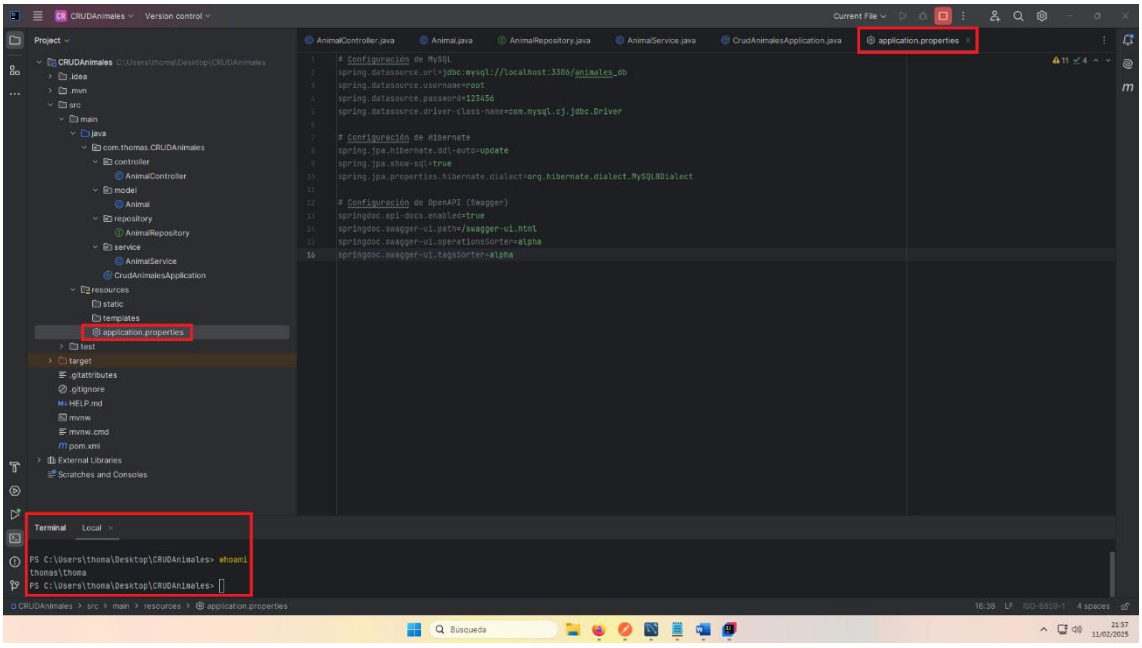
- **service:** Contiene AnimalService, se encarga de la lógica de negocio y de interactuar con el AnimalRepository. Contiene métodos como **findAll(), findById(id), save(animal)** y **deleteById(id)**, asegurando que la información se procese correctamente antes de enviarla al controlador.



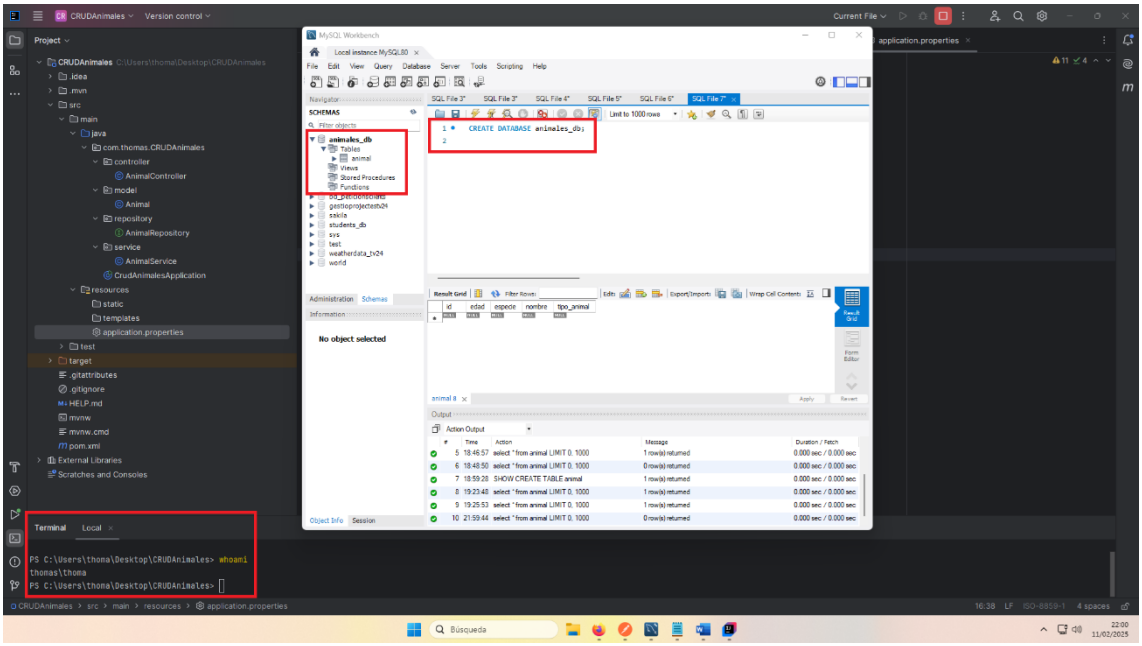
- **CrudAnimalesApplication (Clase Principal):** Es la clase principal que inicia la aplicación Spring Boot. Contiene la anotación **@SpringBootApplication**, que habilita automáticamente la configuración, el escaneo de componentes y la configuración de Spring Boot. Dentro de esta clase, el método **main(String[] args)** usa **SpringApplication.run(CrudAnimalesApplication.class, args);** para lanzar la aplicación. Sin esta clase, la aplicación no se ejecutaría, ya que actúa como el punto de arranque de toda la API.



- **application.properties (Configuración de la Base de Datos y Spring Boot):** Contiene la configuración de la base de datos (MySQL), el puerto del servidor y otros parámetros como **spring.jpa.hibernate.ddl-auto=update** para la gestión de la base de datos.



- **Creación de la BBDD (MySQL):** Hemos usado en este caso el MySQL Workbench, en la captura anterior mostramos el driver, user y root para poder hacer la conexión a la bbdd. Creamos en el Workbench la bbdd.



Nombre y apellidos: Thomas Van Eemeren.
Curso: CFGS DAM Semipresencial.
Módulo: Accés A Dades (ADA).