### **Enlaces:**

<https://miro.com/app/board/uXjVKnh9ULQ=/>

<https://trello.com/b/jjyEILZ8/animal-shelter>

### 

## 

### 

### **Nombre del Proyecto:**

Desarrollo del Backend para una Aplicación Web de Refugio de Animales utilizando Spring Boot

### **Miembros y Roles:**

* **Miembro 1:** (nombre) - **Rol:** Desarrollador Backend, Responsable de la implementación de la API REST.
* **Miembro 1:** (nombre) - **Rol:** Desarrollador Backend, Responsable de la implementación de la API REST.
* .
* **Miembro 3:** (nombre) - **Scrum Master**.
* **Miembro 4:** (nombre) - **Product Owner**.

### **Breve Explicación del Proyecto:**

El proyecto consiste en el desarrollo del backend de una aplicación web para un refugio de animales que tiene como objetivo principal facilitar la adopción de mascotas y la gestión de donaciones. Los usuarios pueden visualizar anuncios de mascotas, realizar donaciones, y los administradores pueden gestionar tanto las mascotas como las donaciones. Se ha implementado un sistema de autenticación y autorización con Spring Security y JWT para proteger las operaciones críticas.

### **Proyecto Visual:**

Aquí se presentará el código y las funcionalidades principales implementadas en el proyecto. Se hará una demostración de las principales características:

* **Listar Mascotas:** Mostrar un ejemplo de cómo se visualizan las mascotas disponibles para adopción.
* **Autenticación y Autorización:** Ejemplo de login y cómo se restringen ciertas funcionalidades a los administradores.
* **Gestión de Donaciones:** Visualización del sistema de donaciones.

### **Kanban/Jira:**

* **¿Qué son las metodologías ágiles?**Las metodologías ágiles, como Scrum y Kanban, son enfoques iterativos para la gestión de proyectos que se centran en la **entrega continua de valor**, la **adaptación a los cambios, y la colaboración constante con el cliente.**
* **¿Por qué nuestro equipo las implementa?**Utilizamos metodologías ágiles para asegurar una **organización eficiente**, f**acilitar una clara visibilidad del progreso del proyecto**, y **responder rápidamente a cambios y nuevos requisitos** del cliente.
* **herramientas de organización:** trello para la gestión y adjudicación de tareas, discord para las comunicaciones, miro para el pseudocódigo y las bases de datos.

### **Reuniones y Qué Hacemos en Cada una de Ellas:**

* **Sprint Planning:**Definimos las historias de usuario y las tareas que se abordarán en cada sprint, asegurando que los objetivos sean alcanzables dentro del tiempo estimado.
* **Daily:**Reunión diaria de 10-15 minutos para discutir el progreso, identificar obstáculos y ajustar el trabajo diario.
* **Sprint Review:**Revisamos el trabajo completado durante el sprint, mostrando los avances y recibiendo retroalimentación.
* **Sprint Retrospective:**Reflexionamos sobre el sprint para identificar áreas de mejora en el proceso de trabajo.

## **Herramientas Empleadas:**

* Spring Boot: Para el desarrollo del backend.
* Spring Web: Para manejar solicitudes web.
* Spring Data JPA: Para la interacción con la base de datos.
* Spring Security: para proteger la aplicación gestionando la autenticación y autorización de usuarios, asegurando el acceso a recursos y datos.
* My php admin: Para la conexión con la base de datos MySQL.
* Lombok: Para reducir el código repetitivo.
* JUnit 5 y Mockito:: Para pruebas unitarias.
* GitHub: alojar el repositorio.
* Git: Para el control de versiones.

### **Visualización del Código:**

**Estructura de la Base de Datos:**Explicación de la configuración de application.properties, incluyendo las credenciales de la base de datos y las opciones de spring.jpa.hibernate.ddl-auto.

* Crear la base de datos en MySQL utilizando phpMyAdmin.
* Modificar `application.properties` y añadir las credenciales de MySQL `jdbc:mysql://localhost:3306/**animalshelterdb**
* Configurar `spring.jpa.hibernate.ddl-auto` adecuadamente ( `create` durante el desarrollo inicial y luego cambiar a `update` o `validate`).

**Modelo-Vista-Controlador:**Descripción de cómo se implementó la arquitectura MVC, incluyendo la organización de las clases y la lógica de negocio.

* + El **Modelo Vista Controlador (MVC)** es un patrón de diseño que organiza el código en tres componentes principales:

1. **Modelo**: Gestiona la lógica de datos y las reglas de negocio. Se encarga de acceder y manipular la información, generalmente interactuando con la base de datos.
2. **Vista**: Presenta la información al usuario. Es la interfaz gráfica o la parte que muestra los datos del modelo en un formato visual.
3. **Controlador**: Actúa como intermediario entre el modelo y la vista. Recibe las entradas del usuario, las procesa, y actualiza el modelo y la vista según sea necesario.

Además de estos tres componentes principales, en aplicaciones más complejas con Spring se suelen incluir:

1. **Servicio**: Contiene la lógica de negocio de la aplicación. Es una capa intermedia entre el controlador y el repositorio, permitiendo que el controlador no tenga que manejar directamente la lógica de negocio.
2. **Repositorio**: Se encarga de la interacción directa con la base de datos. Es responsable de realizar operaciones de CRUD (Crear, Leer, Actualizar, Eliminar) sobre las entidades del modelo.

**Relaciones entre Entidades:**Explicación de las relaciones One-to-Many entre User y Pet, y User y Donation, mostrando cómo estas relaciones se han manejado en la base de datos y el código.

## Entidad USER:

* **Funciones del CRUD: crear, listar, modificar y eliminar.**

## Entidad PET:

* **Mostrar una lista de mascotas disponibles para adopción.**
  + Permitir a los usuarios visualizar todas las mascotas que están disponibles para adopción, incluyendo detalles como nombre, edad, raza, y estado de adopción.
* **Mostrar detalles de una mascota específica.**
  + Permitir a los usuarios consultar información detallada sobre una mascota específica.
* **Agregar nuevas mascotas al sistema.**
  + Permitir a los administradores añadir nuevas mascotas al refugio, incluyendo información relevante como descripción, fotos, y características.
* **Actualizar la información de las mascotas.**
  + Permitir a los administradores editar los detalles de una mascota existente en el sistema.
* **Eliminar mascotas del sistema.**
  + Permitir a los administradores eliminar una mascota cuando ya no esté disponible para adopción.

## Entidad DONATION:

* **Registrar donaciones al refugio.**
  + Permitir a los usuarios realizar donaciones llenando un formulario con sus datos y la cantidad donada.
* **Listar donaciones realizadas.**
  + Permitir a los administradores consultar un listado de todas las donaciones registradas en el sistema.
* **Consultar detalles de una donación específica.**
  + Permitir a los administradores revisar los detalles de una donación individual.
* **Editar o eliminar donaciones del sistema.**
  + Permitir a los administradores modificar o eliminar una donación registrada.

#### **Funcionalidades Extra:**

* **Cambiar el estado de adopción de una mascota.**
  + Permitir a los administradores marcar una mascota como adoptada o disponible.
* **Buscar mascotas adoptadas y no adoptadas.**
  + Permitir a los usuarios buscar listas de mascotas adoptadas y no adoptadas.
* **Identificar al usuario con mayor contribución en donaciones.**
  + Permitir a los administradores consultar el usuario que ha realizado la mayor cantidad de donaciones.

#### **Requisitos de Autenticación y Autorización:**

* **Implementar sistema de login con Spring Security y JWT.**
  + Permitir a los usuarios autenticarse en la aplicación utilizando un sistema seguro basado en Spring Security y JSON Web Tokens (JWT).
* **Definir roles de usuario y administrador.**
  + Los usuarios no autenticados podrán ver mascotas y realizar donaciones.
  + Los administradores tendrán permisos adicionales para gestionar mascotas y donaciones.

**Pruebas Automatizadas:**Explicación de cómo se han desarrollado y ejecutado pruebas automatizadas utilizando **JUnit y Mockito**, con ejemplos de pruebas unitarias. **Postman**: desarrollar, probar y documentar APIs (Interfaces de Programación de Aplicaciones). Enviar Solicitudes HTTP y Examinar Respuestas:

### **Desafíos y Soluciones en el Desarrollo del Proyecto:**

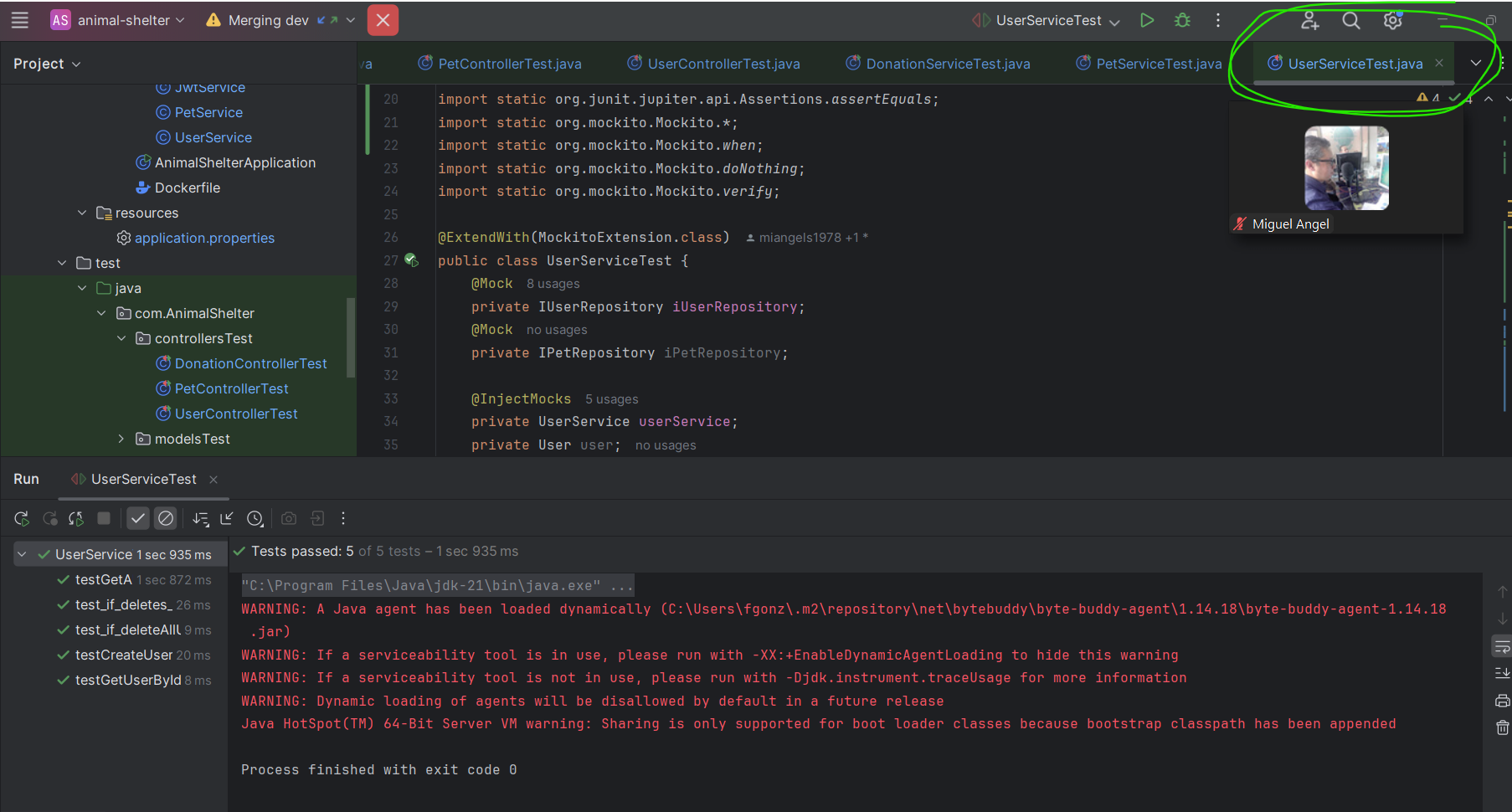
* **Integración de Spring Security y JWT:**Desafío en la configuración inicial de Spring Security y la implementación de JWT.  
  **Solución:** Implementación paso a paso con revisiones continuas para asegurar la seguridad en el manejo de tokens.
* **Optimización de Consultas a la Base de Datos:**Desafío en la gestión eficiente de las consultas para buscar mascotas y donaciones.  
  **Solución:** Optimización de las consultas con JPQL y ajustes en la configuración de Spring Data JPA.
* **Dockerización del Proyecto:**Dificultades en la configuración del contenedor y el despliegue en Docker Hub.  
  **Solución:** Documentación y uso de Docker Compose para gestionar múltiples servicios.
* Futuras implementaciones: opción de adopción o acogida.

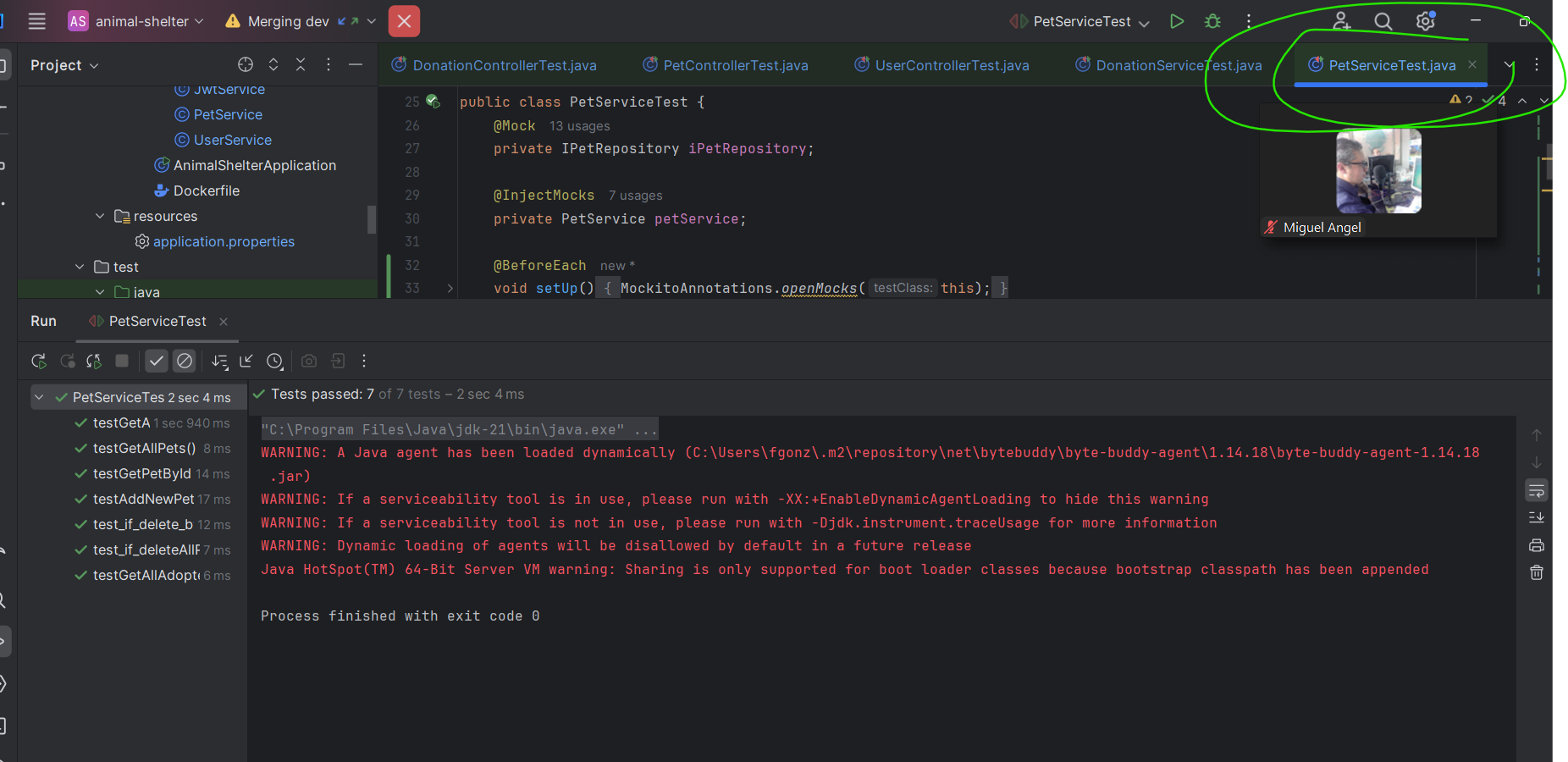
### **Conclusiones Finales:**

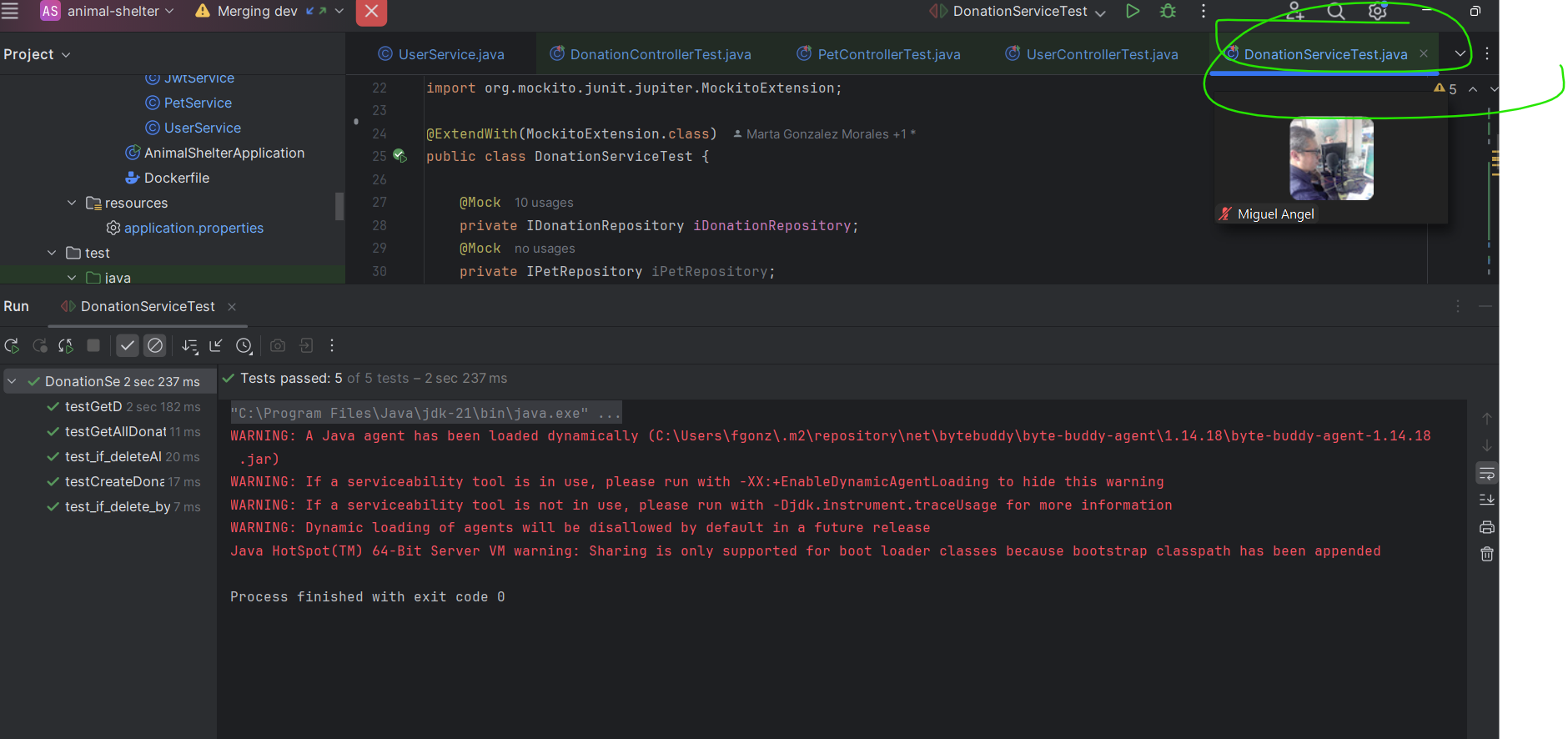
El proyecto ha permitido desarrollar una aplicación robusta y escalable para la gestión de un refugio de animales, utilizando Spring Boot y tecnologías relacionadas. El uso de metodologías ágiles ha sido clave para gestionar el proyecto de manera eficiente y adaptarse a los cambios. Las funcionalidades desarrolladas no solo cumplen con los requisitos iniciales, sino que también abren la puerta a futuras mejoras y expansiones, como la implementación de nuevas funcionalidades y la mejora continua de la seguridad.

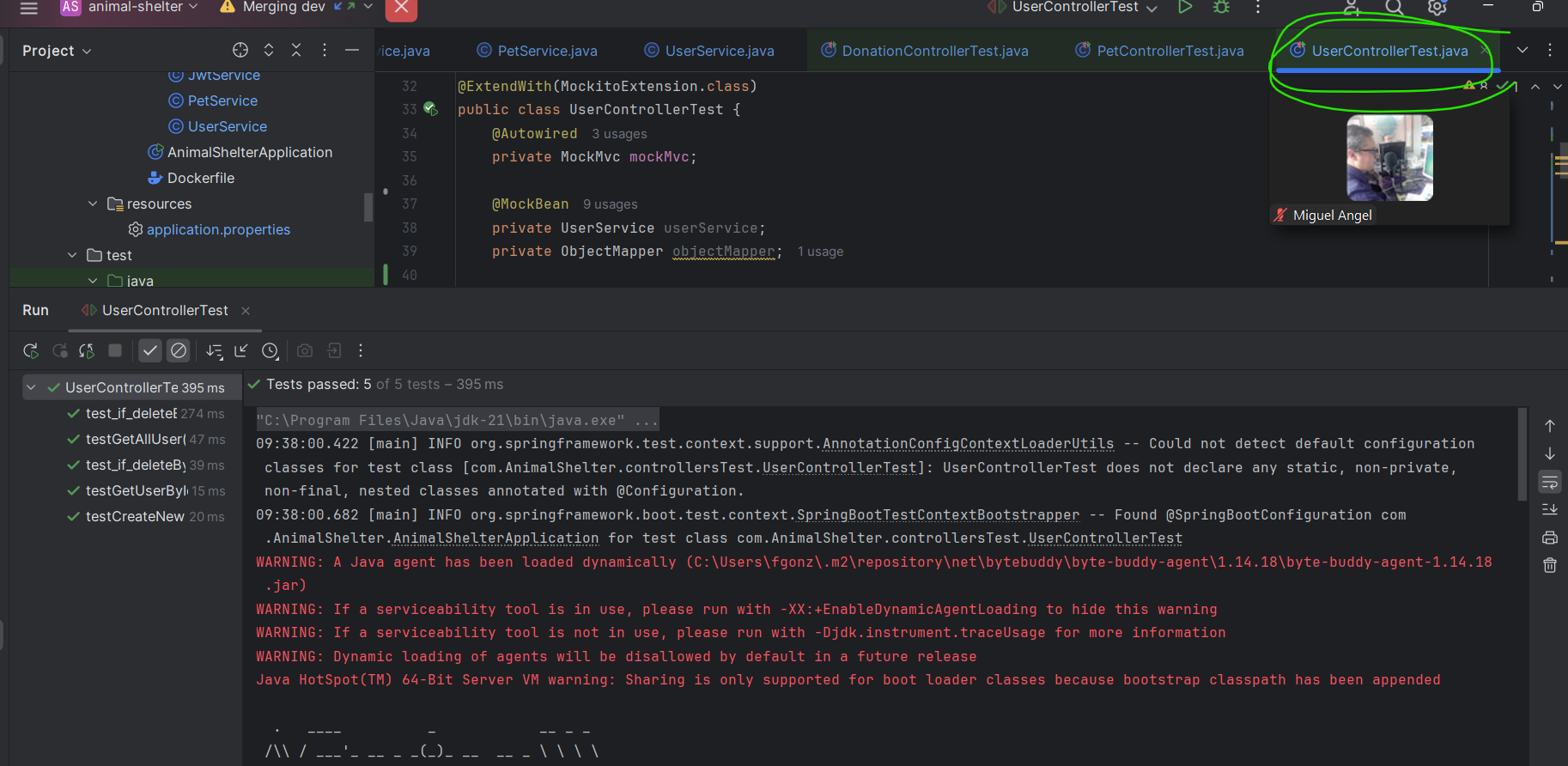
### **Turno de Preguntas:**

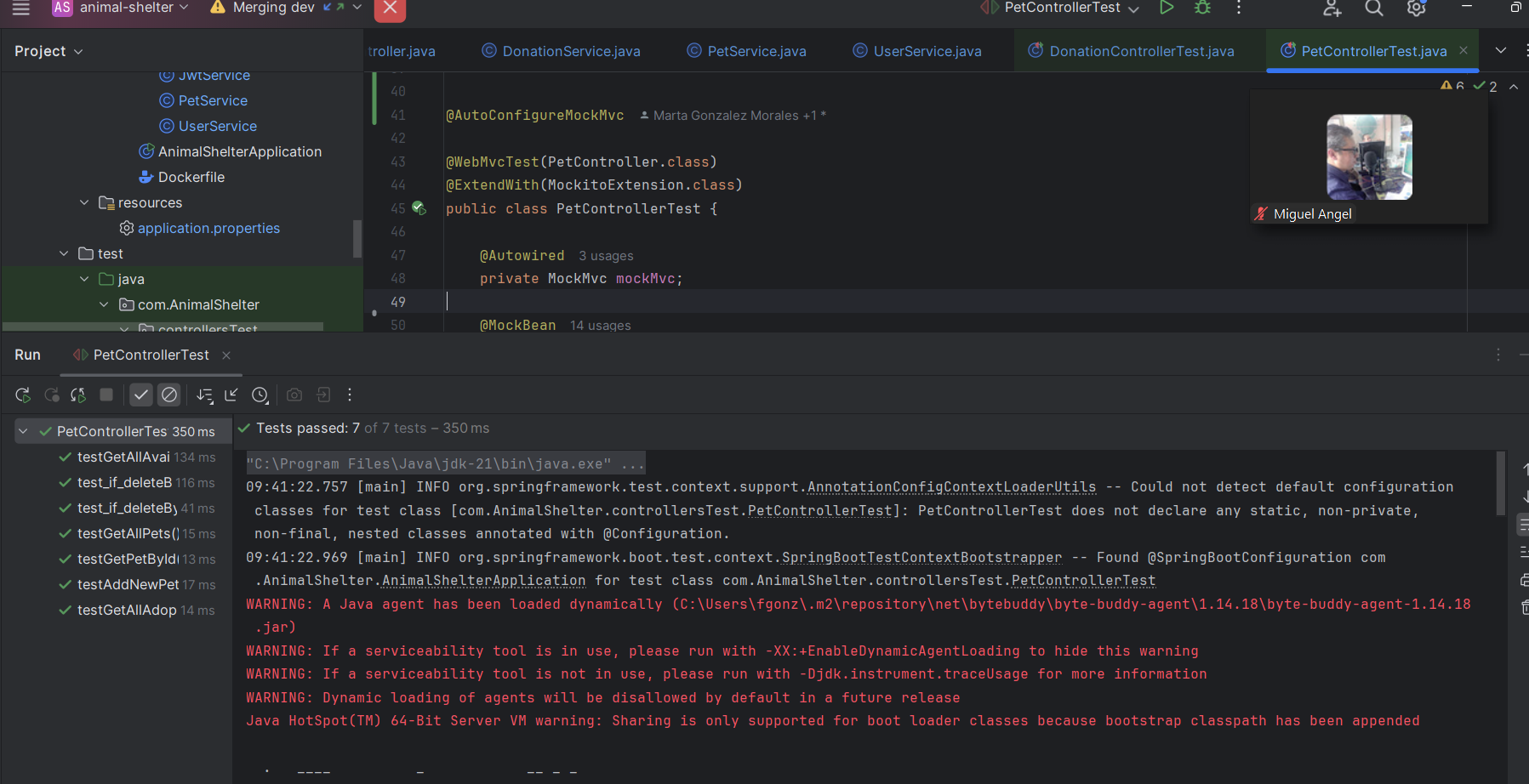
Se abrirá un espacio para responder cualquier duda o pregunta del público sobre el proyecto, su implementación y los próximos pasos a seguir.

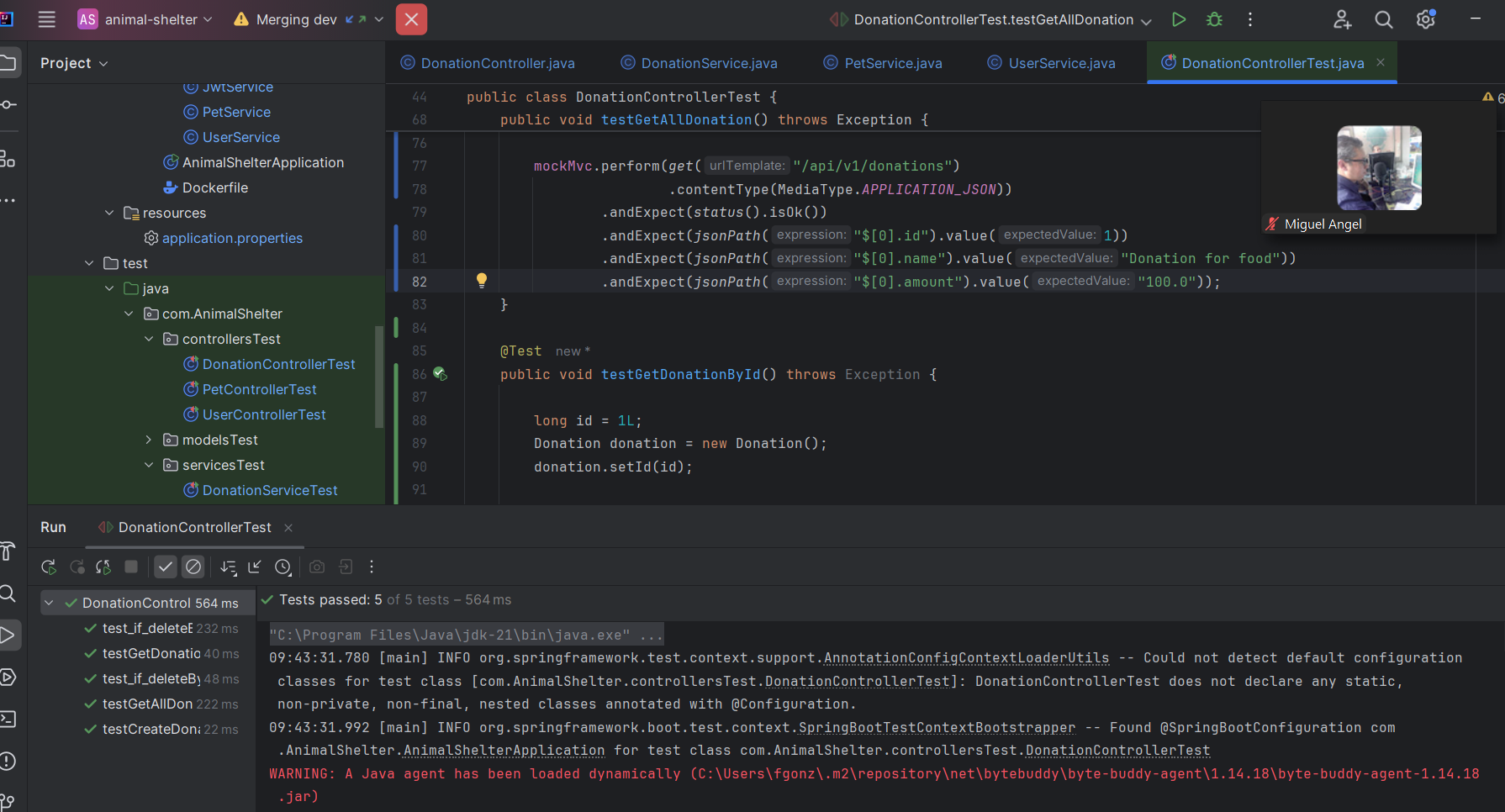








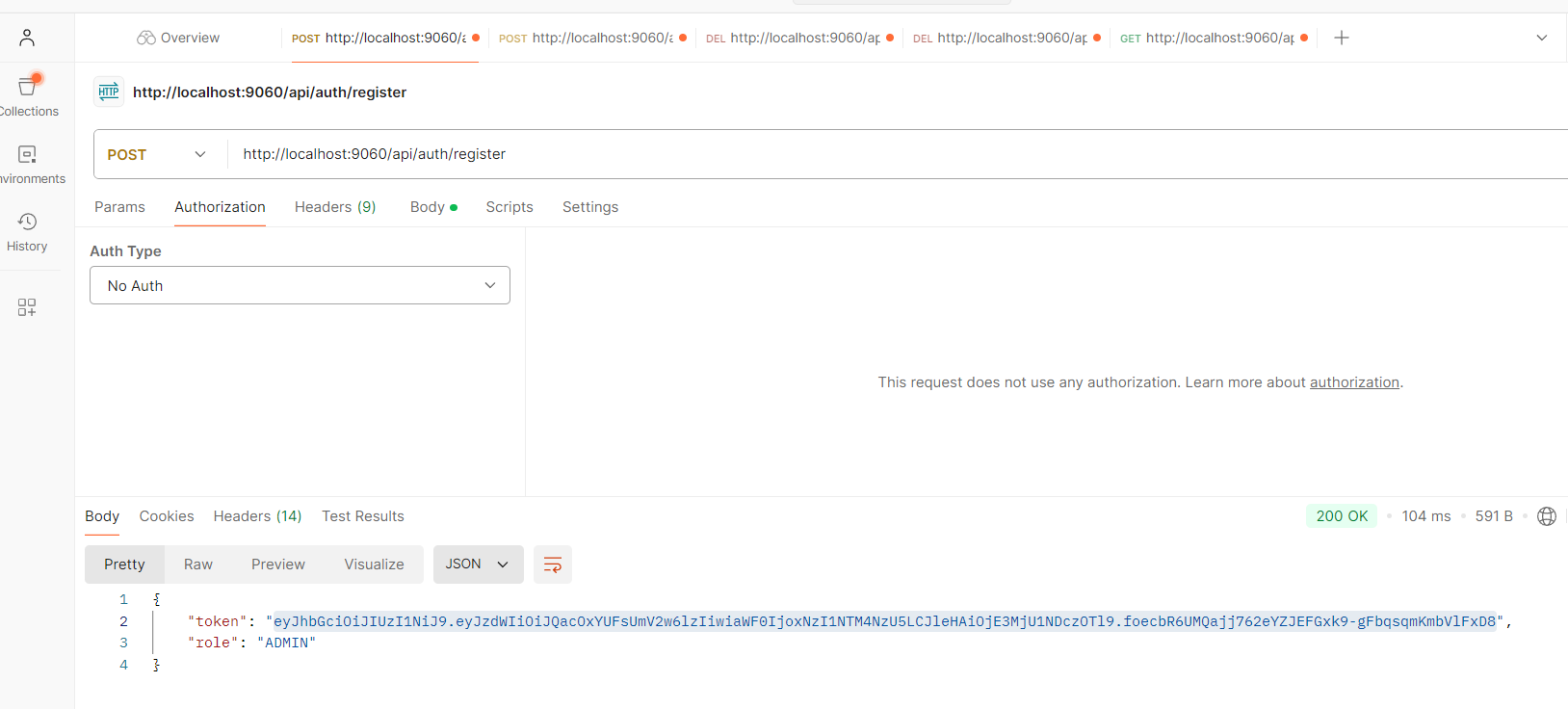




**PRUEBAS CON POSTMAN**

**REGISTER**

**endpoint:** [**http://localhost:9060/api/auth/register**](http://localhost:9060/api/auth/register)

****

**en body poned otro username:**

**{**

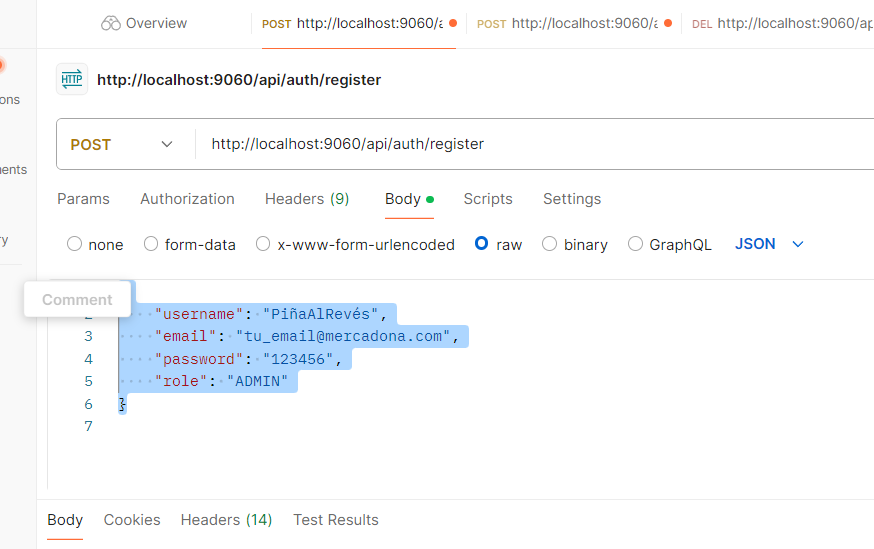
**"username": "PiñaAlRevés",**

**"email": "tu\_email@mercadona.com",**

**"password": "123456",**

**"role": "ADMIN"**

**}**

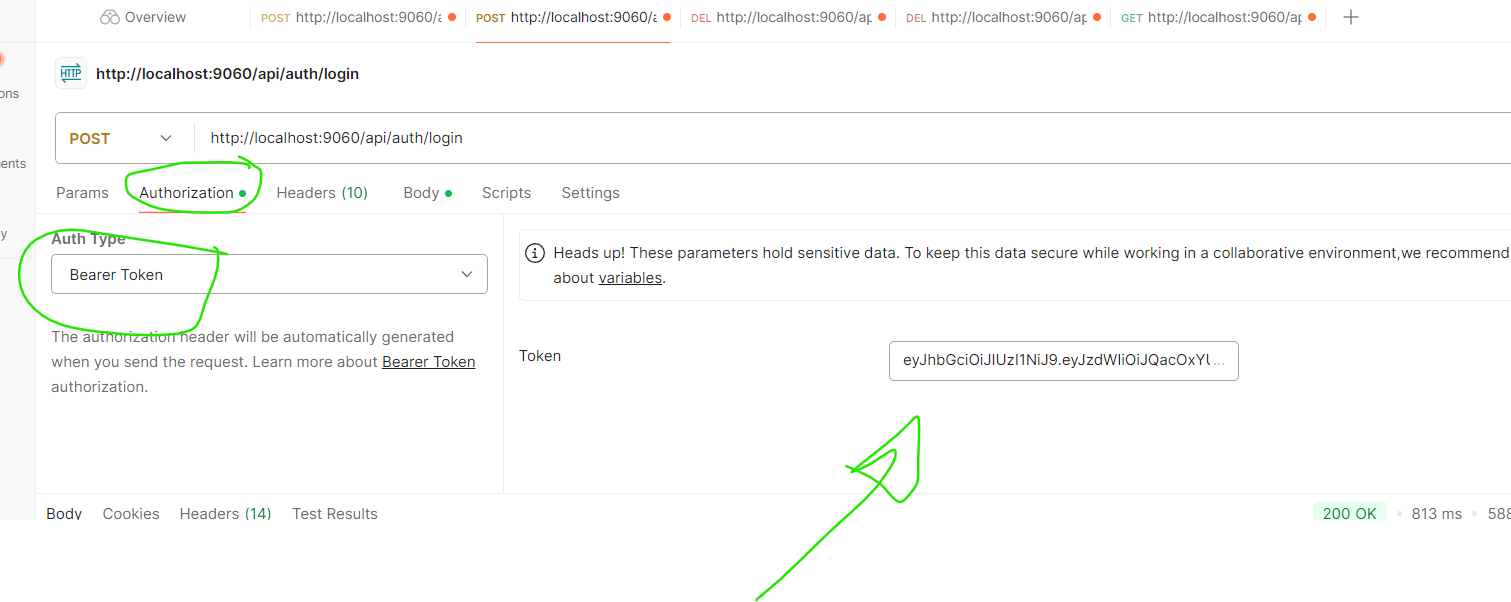


Copia el token q aparece abajo, sin comillas.

**LOGIN**

**endpoint:** <http://localhost:9060/api/auth/login>

en authorization, seleccionar bearer token y copiar el token de la pantalla anterior sin comillas



copiar el token que da como respuesta y pegarlo en las consultas que vienen a continuacion:

**USER**

**Primero de todo hay q crear el usuario para que no de fallo de MEDIATYPE**

endpoint: [http://localhost:](http://localhost:9060/api/v1/users)[:8080](http://localhost:9060/api/v1/donations)[/api/v1/users](http://localhost:9060/api/v1/users)

{

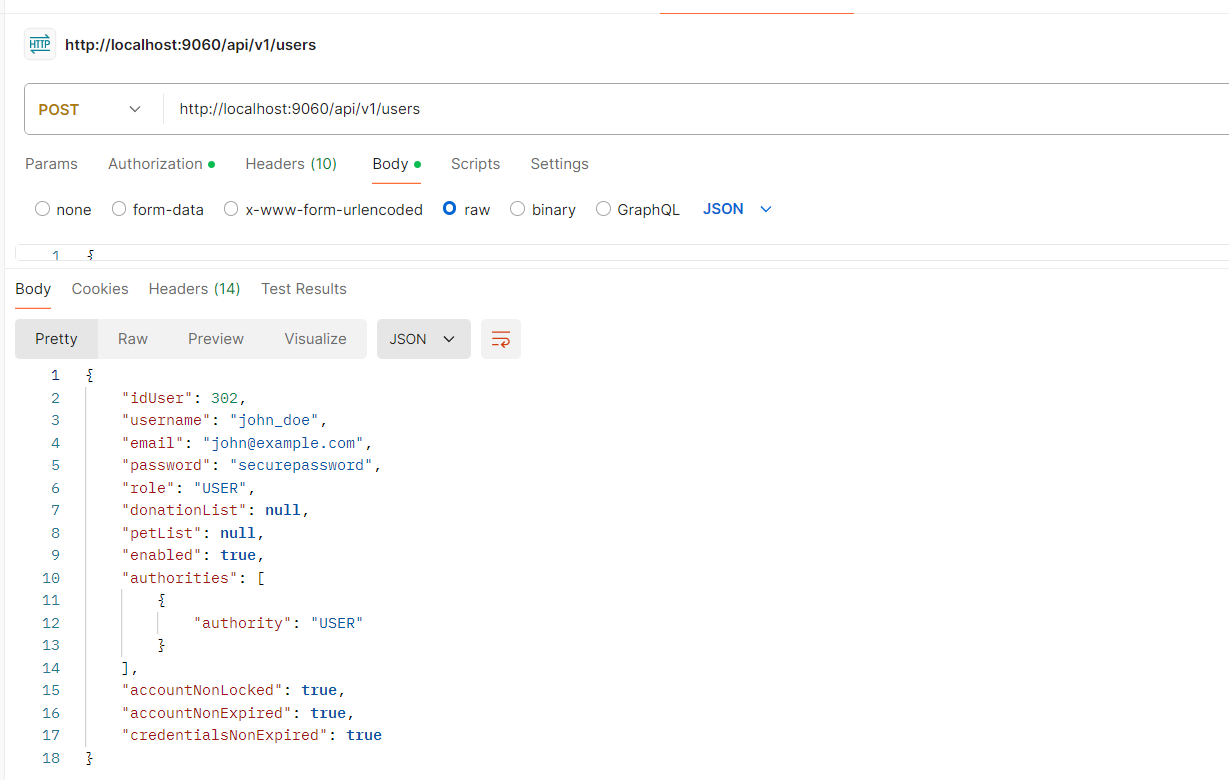
"username": "john\_doe",

"email": "john@example.com",

"password": "securepassword",

"role": "USER"

}



getall users

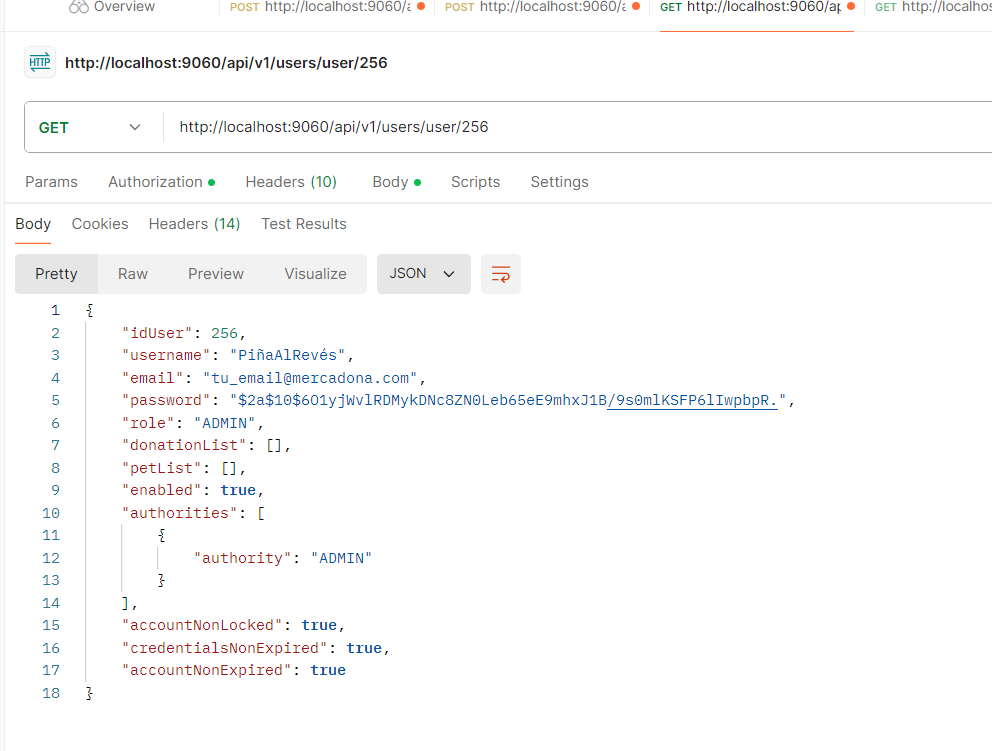
endpoint: http://localhost:[:8080](http://localhost:9060/api/v1/donations)/api/v1/users





get user by id

endpoint: http://localhost:[:8080](http://localhost:9060/api/v1/donations)/api/v1/users/user/302



**PET**

endpoint: http://localhost:[:8080](http://localhost:9060/api/v1/donations)/api/v1/pets

body:

{

"name": "Buddy",

"species": "Dog",

"age": 3,

"gender": "Male",

"description": "A friendly dog looking for a home.",

"isadopted": false,

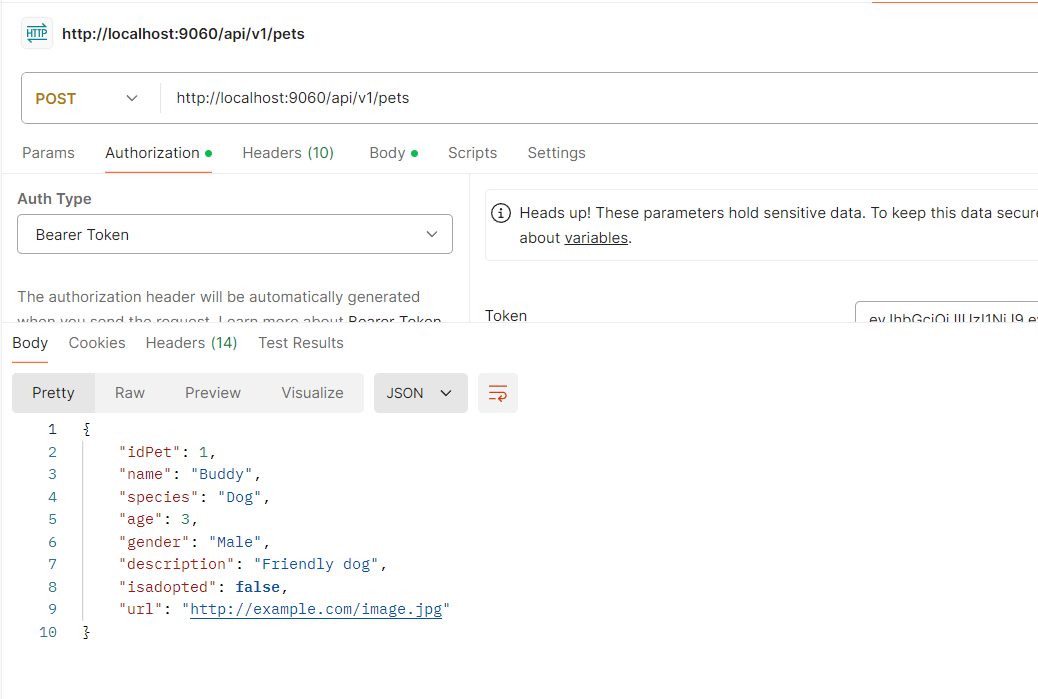
"url": "http://example.com/buddy.jpg",

"user": {

"idUser": 1

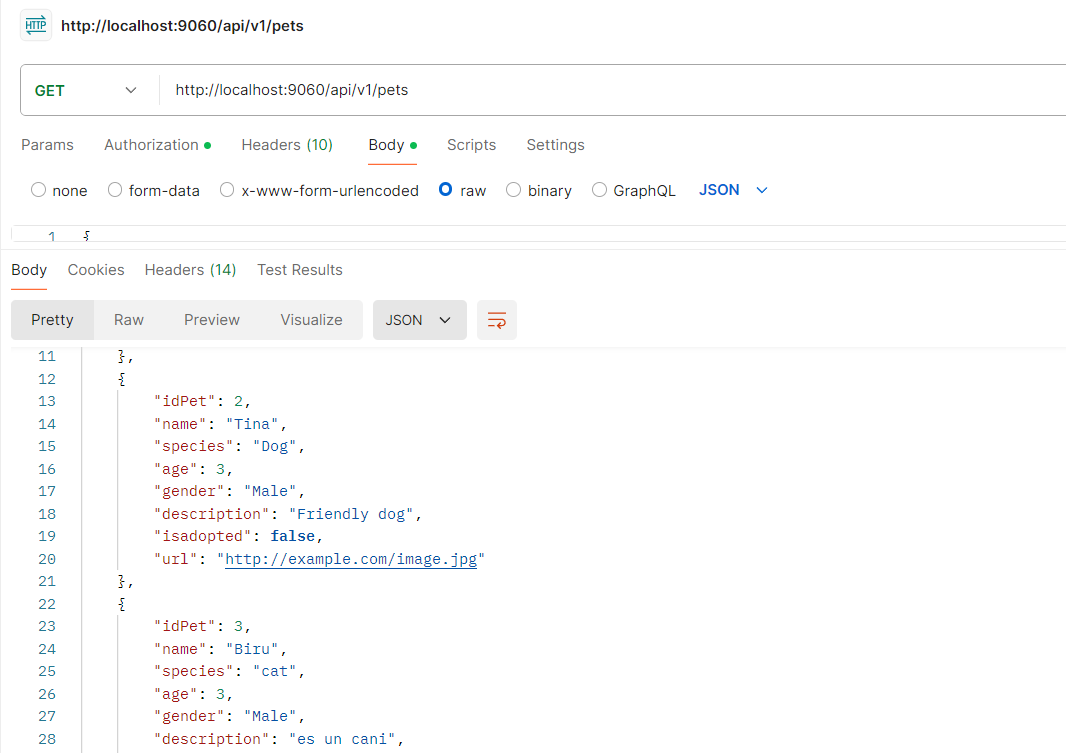
}

}



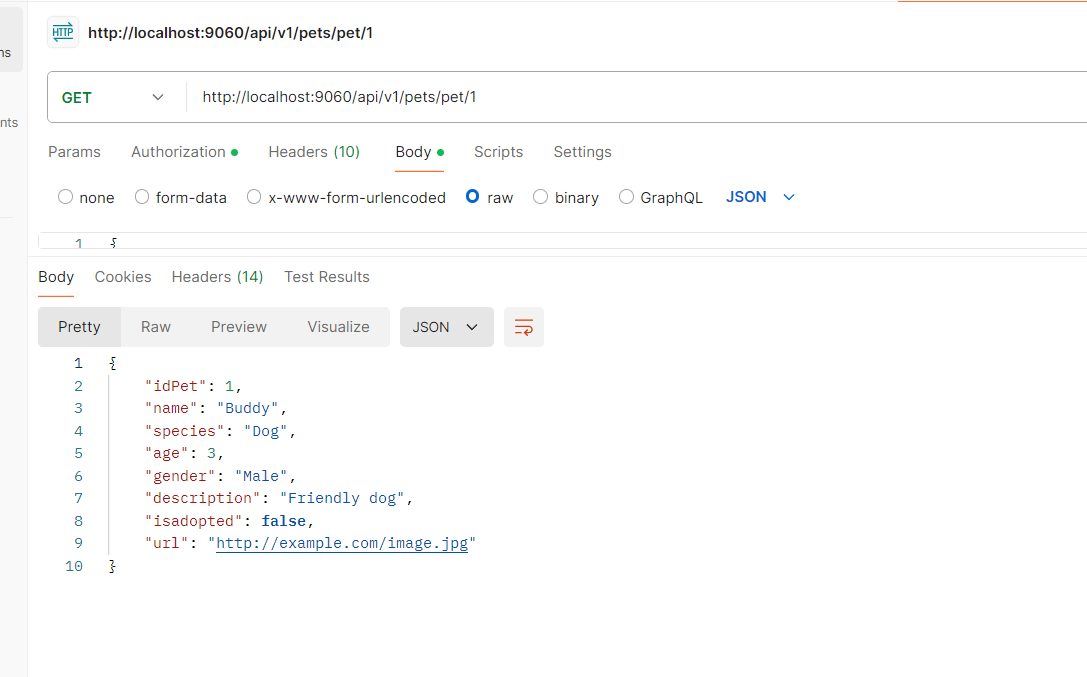
GETALL

endpoint: http://localhost:[:8080](http://localhost:9060/api/v1/donations)/api/v1/pets



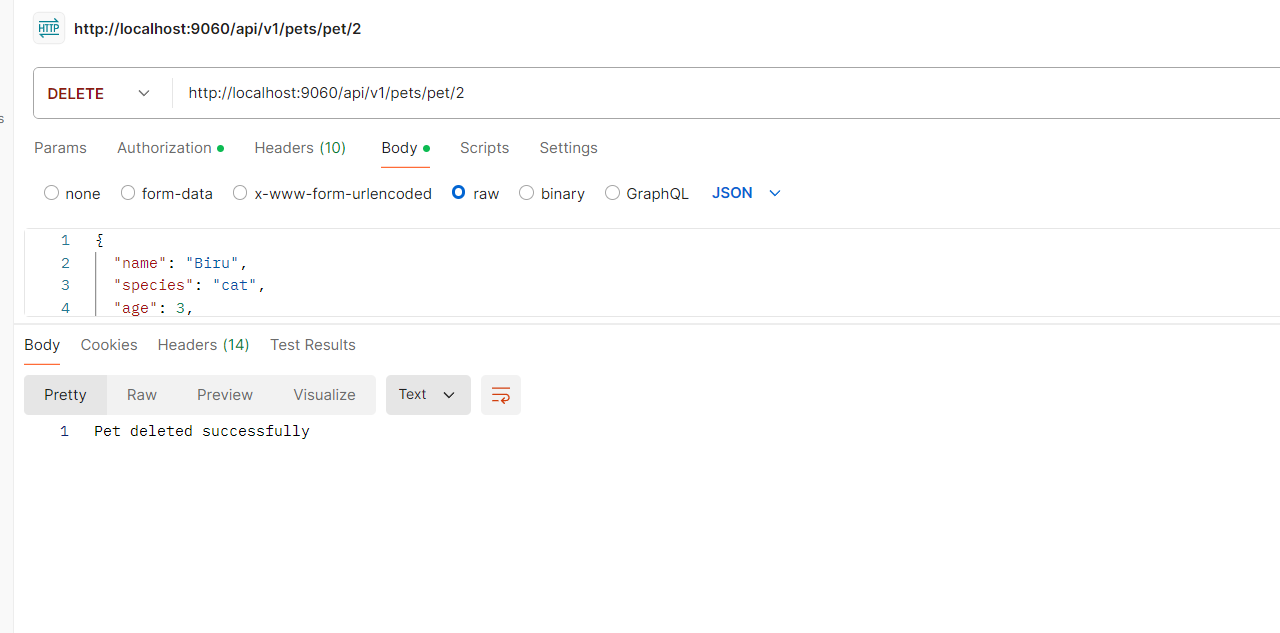
get by id

endpoint: http://localhost:[:8080](http://localhost:9060/api/v1/donations)/api/v1/pets/pet/1



delete by id

http://localhost:[:8080](http://localhost:9060/api/v1/donations)/api/v1/pets/pet/1



DONATION

endpoint [http://localhost::8080/api/v1/donations](http://localhost:9060/api/v1/donations)

{

"amount": 100.0,

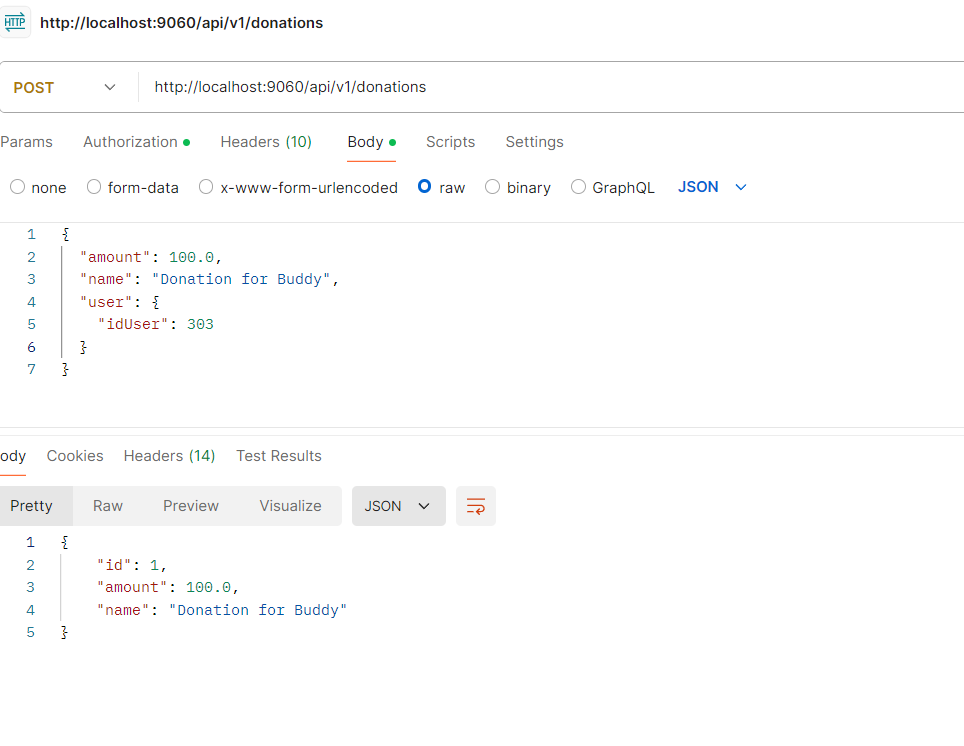
"name": "Donation for Buddy",

"user": {

"idUser": 303

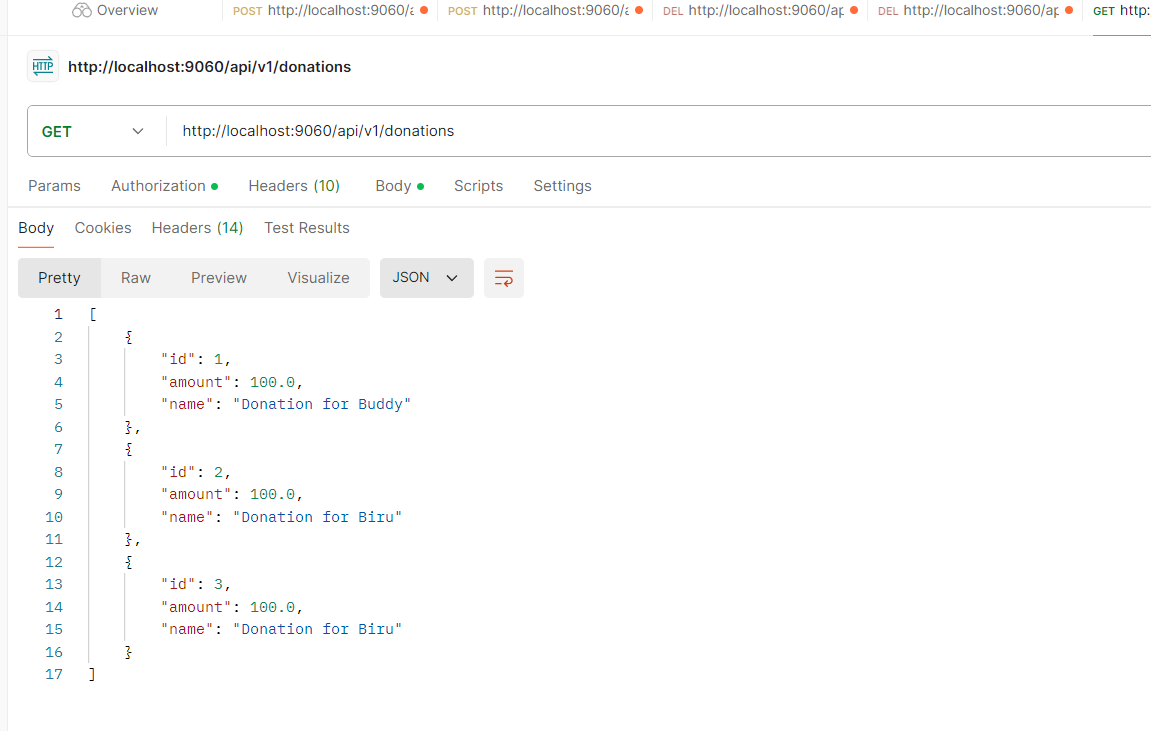
}

}



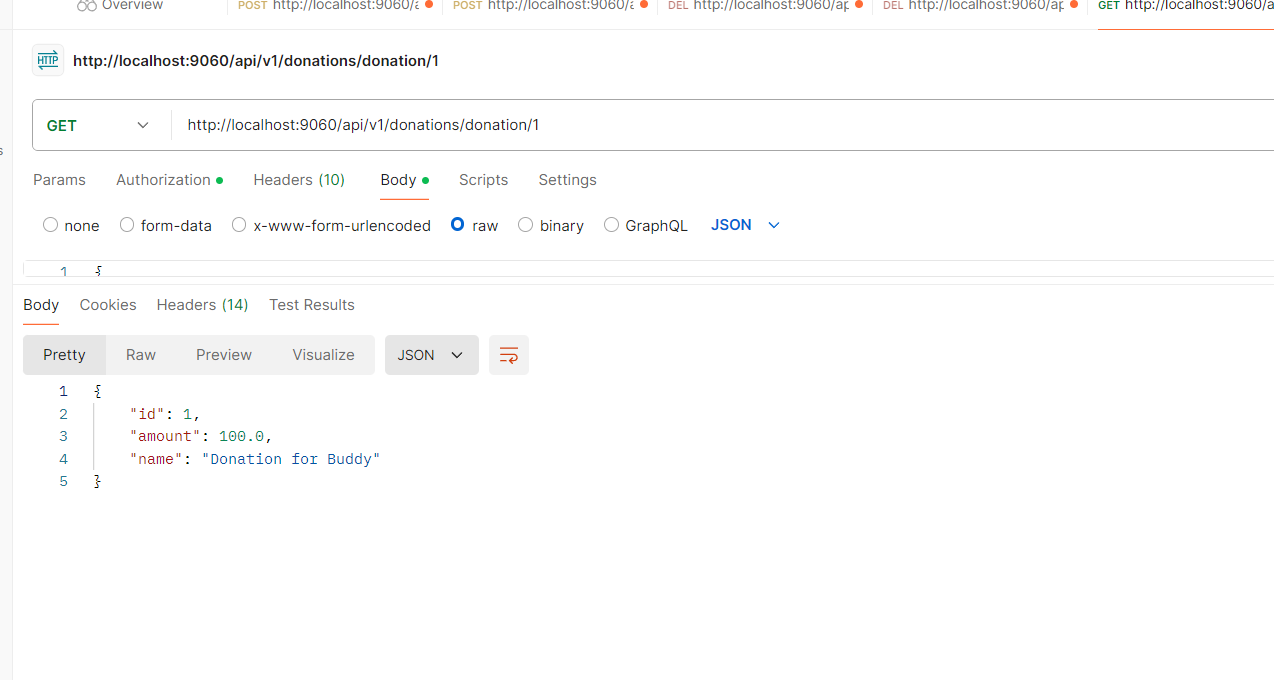
getall

endpoint [http://localhost:8080/api/v1/donations](http://localhost:9060/api/v1/donations)



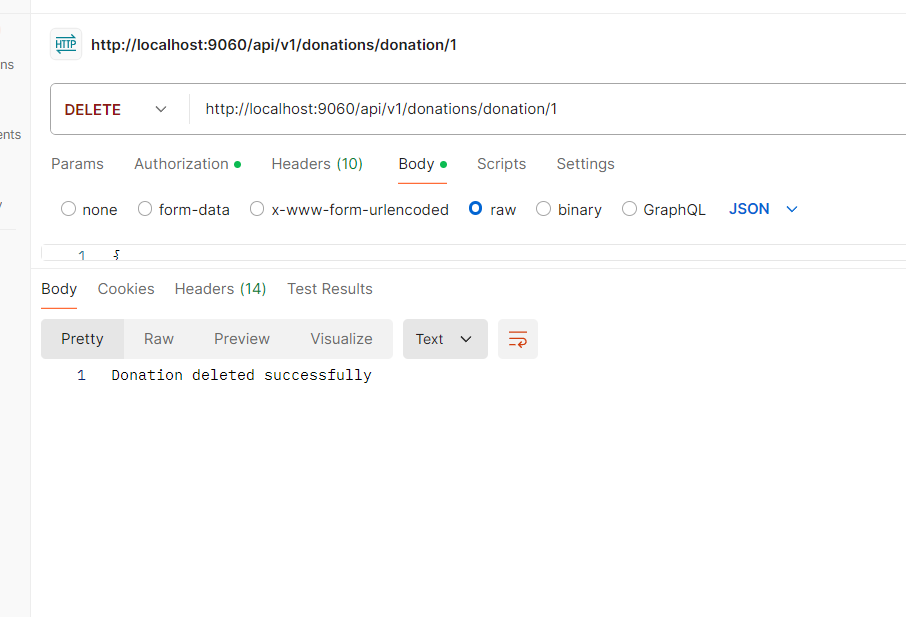
get by id

http://localhost:[:8080](http://localhost:9060/api/v1/donations)/api/v1/donations/donation/1



delete by id

http://localhost:[:8080](http://localhost:9060/api/v1/donations)/api/v1/donations/donation/2



Los test funcionan a 11.46 del 6/09

