Universidad “Mayor de San Andrés”

Facultad de Ciencias Puras y Naturales

PROYECTO-DESARROLLO WEB BACKEND

PLATAFORMA DE ENCUESTAS Y ANÁLISIS DE OPINIÓN

Universitarios(a): Callizaya Rosas Carlos Javier
Canaviri Huaranca Carlos Javier
Loza Fernandez Joaquin Humberto
Mamani Cordero Iver Pedro
Picavia Saravia Erick Julio

Carrera: Informatica

Docente: Rosalia Lopez Montalvo

Grupo: 6

Fecha: 06 de mayo de 2025

La Paz - Bolivia

Documento de Proyecto – Desarrollo Web Backend

2. Introducción

· Objetivo del sistema.

El sistema de encuestas permite gestionar usuarios, encuestas, preguntas (simples y de opción múltiple) y respuestas, facilitando la recolección y análisis de información de manera segura y estructurada.

Alcance funcional

- Registro y autenticación de usuarios con diferentes roles (SUPER_ADMIN, ADMIN, USER).
- Creación, edición, consulta y eliminación de encuestas.
- Gestión de preguntas y respuestas asociadas a encuestas.
- Soporte para preguntas de opción múltiple.
- Seguridad basada en roles y JWT.
- Exposición de endpoints RESTful para integración con otros sistemas o clientes.

Tecnologías utilizadas

- Backend: Spring Boot 3.x, Spring Data JPA, Spring Security, Spring Session JDBC
- Base de datos: PostgreSQL
- Seguridad: JWT (JSON Web Token)
- Pruebas: Postman

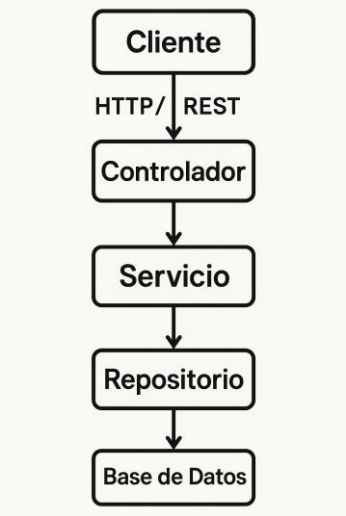
- Frontend: Next Js, Tailwind, Next-Auth
- Otros: Lombok, Redis

2. Arquitectura del Sistema

2.1. Arquitectura Monolítica *(Completar solo si aplica)*

Descripción de capas

- Controlador: Maneja las solicitudes HTTP y expone los endpoints REST (controller).
- Servicio: Contiene la lógica de negocio y validaciones adicionales (service).
- Repositorio: Acceso a datos y operaciones CRUD sobre la base de datos (repository).
- Entidad/DTO: Modelos de datos persistentes y de transferencia (entity, dto).
- Seguridad: Configuración de autenticación y autorización (security, config).·



3. Seguridad: Autenticación y Autorización

· Método utilizado (JWT, API externa, etc.).

· Roles y permisos.

- ROL_SUPERADMIN: Acceso total a todos los recursos.
- ROL_ADMIN: Acceso a casi todos los recursos.
- ROL_USER: Acceso a recursos públicos limitados.

· Protección de rutas.

Autenticación (/api/auth)

Método	Ruta	Descripción	Rol necesario
POST	/api/auth/login	Login de usuario	Público
POST	/api/auth/signup	Registro de usuario	Público
GET	/api/auth/session-info	Info de sesión actual	USER
POST	/api/auth/logout	Cerrar sesión	USER

Usuarios (/api/usuarios)

Método	Ruta	Descripción	Rol necesario
GET	/api/usuarios	Listar todos los usuarios	ADMIN
GET	/api/usuarios/{id}	Obtener usuario por ID	ADMIN

POST	/api/usuarios	Crear usuario	ADMIN
PUT	/api/usuarios/{id}	Actualizar usuario	ADMIN
DELETE	/api/usuarios/{id}	Eliminar usuario	SUPERADMIN

Encuestas (/api/encuestas)

Método	Ruta	Descripción	Rol necesario
GET	/api/encuestas	Listar encuestas	USER
GET	/api/encuestas/{id}	Obtener encuesta por ID	USER
POST	/api/encuestas	Crear encuesta	ADMIN
PUT	/api/encuestas/{id}	Actualizar encuesta	ADMIN
DELETE	/api/encuestas/{id}	Eliminar encuesta	ADMIN

Preguntas (/api/preguntas)

Método	Ruta	Descripción	Rol necesario
GET	/api/preguntas	Listar preguntas	USER
GET	/api/preguntas/{id}	Obtener pregunta por ID	USER
POST	/api/preguntas	Crear pregunta	ADMIN
PUT	/api/preguntas/{id}	Actualizar pregunta	ADMIN
DELETE	/api/preguntas/{id}	Eliminar pregunta	ADMIN

Preguntas Múltiples (/api/preguntas-multiples)

Método	Ruta	Descripción	Rol necesario
GET	/api/preguntas-multiples	Listar preguntas múltiples	USER
GET	/api/preguntas-multiples/{id}	Obtener pregunta multiple	USER
POST	/api/preguntas-multiples	Crear pregunta multiple	ADMIN
PUT	/api/preguntas-multiples/{id}	Actualizar pregunta multiple	ADMIN
DELETE	/api/preguntas-multiples/{id}	Eliminar pregunta multiple	ADMIN

Respuestas (/api/respuestas)

Método	Ruta	Descripción	Rol necesario
GET	/api/respuestas	Listar respuestas	ADMIN
GET	/api/respuestas/{id}	Obtener respuesta por ID	ADMIN
POST	/api/respuestas	Crear respuesta	USER
PUT	/api/respuestas/{id}	Actualizar respuesta	ADMIN
DELETE	/api/respuestas/{id}	Eliminar respuesta	ADMIN

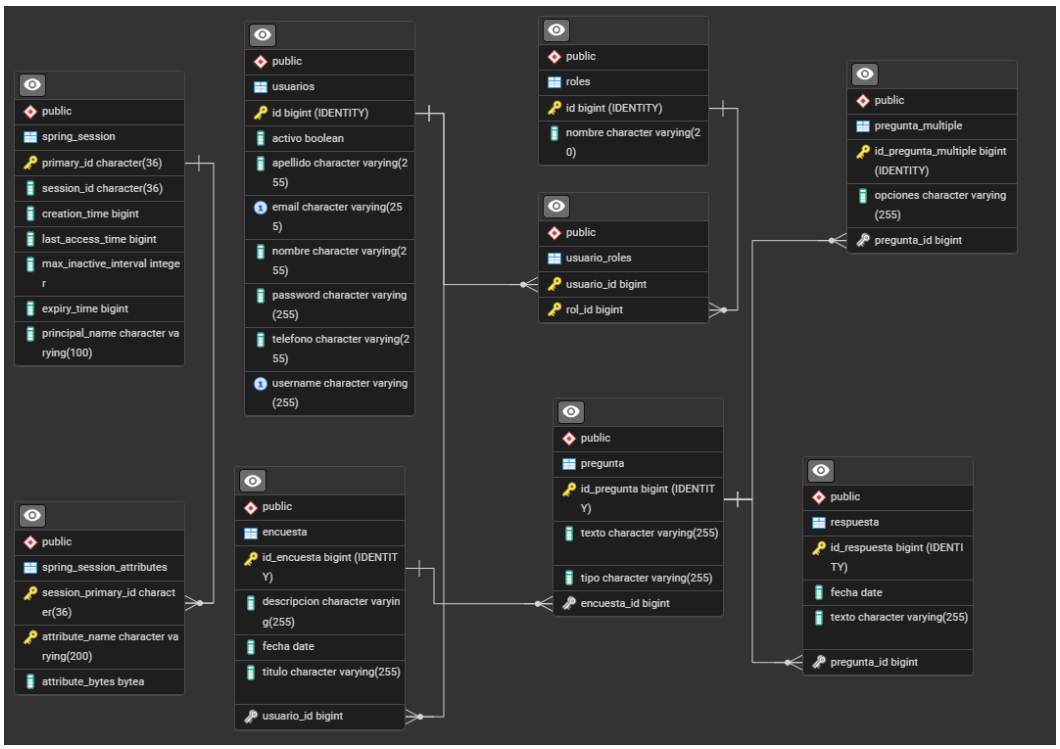
Endpoints de prueba/públicos

Método	Ruta	Descripción	Rol necesario
GET	/api/public/test	Test público	Público

GET	/api/estudiantes/test	Test para usuarios	USER
GET	/api/docentes/test	Test para admins	ADMIN
GET	/api/admin/test	Test para superadmins	SUPERADMIN

4. Base de Datos y Validaciones

· Diagrama ER o relacional.



· Descripción de entidades principales.

- Usuario: username, nombre, apellido, email, password, activo, roles.
- Rol: nombre (ROL_SUPERADMIN, ROL_ADMIN, ROL_USER).
- Encuesta: usuario, título, fecha, descripción.
- Pregunta: encuesta, texto, tipo.
- PreguntaMultiple: pregunta, opciones.

- Respuesta: pregunta, fecha, texto.

· Validaciones de campos en modelos (anotaciones, restricciones).

- Uso de anotaciones como @NotBlank, @Email, @Size, @NotNull en los DTOs.
- Restricciones de unicidad en username y email.

· Validaciones a nivel de servicio y controladores.

- Validación de existencia de entidades relacionadas antes de crear/editar (por ejemplo, usuario, encuesta, pregunta).
- Manejo de errores y respuestas claras en caso de datos inválidos.

· Consideraciones de concurrencia (transacciones, sincronización de accesos críticos).

- Uso de transacciones (@Transactional) en operaciones críticas de escritura.
- Sincronización de accesos críticos a nivel de servicio si se requiere.

5. Manejo de Repositorio (Operaciones CRUD)

· Descripción de los endpoints REST por entidad.

Usuarios (/api/usuarios)

- GET /api/usuarios - Listar usuarios (ADMIN)

- GET /api/usuarios/{id} - Obtener usuario por ID (ADMIN)

- POST /api/usuarios - Crear usuario (ADMIN)

- PUT /api/usuarios/{id} - Actualizar usuario (ADMIN)

- DELETE /api/usuarios/{id} - Eliminar usuario (ADMIN)

Encuestas (/api/encuestas)

- GET /api/encuestas - Listar encuestas

- GET /api/encuestas/{id} - Obtener encuesta por ID

- POST /api/encuestas - Crear encuesta

- PUT /api/encuestas/{id} - Actualizar encuesta

- DELETE /api/encuestas/{id} - Eliminar encuesta

Preguntas (/api/preguntas)

- GET /api/preguntas - Listar preguntas
- GET /api/preguntas/{id} - Obtener pregunta por ID

- POST /api/preguntas - Crear pregunta
- PUT /api/preguntas/{id} - Actualizar pregunta
- DELETE /api/preguntas/{id} - Eliminar pregunta

Preguntas Múltiples (/api/preguntas-multiples)

- GET /api/preguntas-multiples - Listar preguntas múltiples
- GET /api/preguntas-multiples/{id} - Obtener pregunta múltiple por ID
- POST /api/preguntas-multiples - Crear pregunta múltiple
- PUT /api/preguntas-multiples/{id} - Actualizar pregunta múltiple
- DELETE /api/preguntas-multiples/{id} - Eliminar pregunta múltiple

Respuestas (/api/respuestas)

- GET /api/respuestas - Listar respuestas
- GET /api/respuestas/{id} - Obtener respuesta por ID
- POST /api/respuestas - Crear respuesta
- PUT /api/respuestas/{id} - Actualizar respuesta
- DELETE /api/respuestas/{id} - Eliminar respuesta

· Ejemplos de peticiones (GET, POST, PUT, DELETE). ·

Crear usuario

POST /api/usuarios

{
"username": "nuevo_usuario",
"nombre": "Juan",
"apellido": "Pérez",
"email": "juan.perez@example.com",
"telefono": "123456789",
"password": "contraseñaSegura123",
"activo": true,
"roles": ["ROL_ESTUDIANTE"]

Crear encuesta

POST /api/encuestas

{
"usuarioId": 1,
"titulo": "Encuesta de Satisfacción",
"fecha": "2025-06-01",
"descripcion": "Encuesta para medir la satisfacción de los usuarios."
}

Crear pregunta

POST /api/preguntas

{

"encuestaId": 1,

"texto": "¿Qué tan satisfecho estás con el servicio?",

"tipo": "opcion_multiple"

}

Crear respuesta

POST /api/respuestas

{

"preguntaId": 1,

"fecha": "2025-06-01",

"texto": "Muy satisfecho"

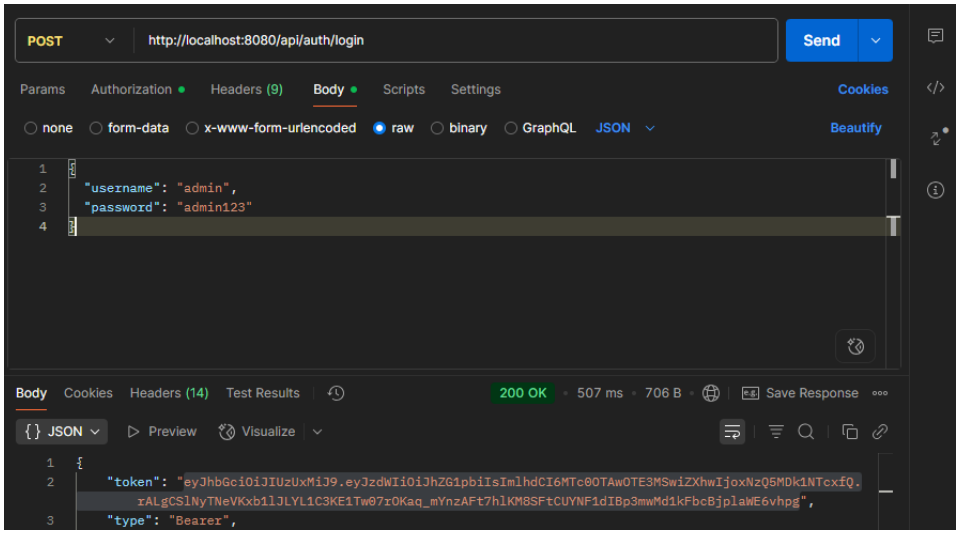
}

Consultas personalizadas

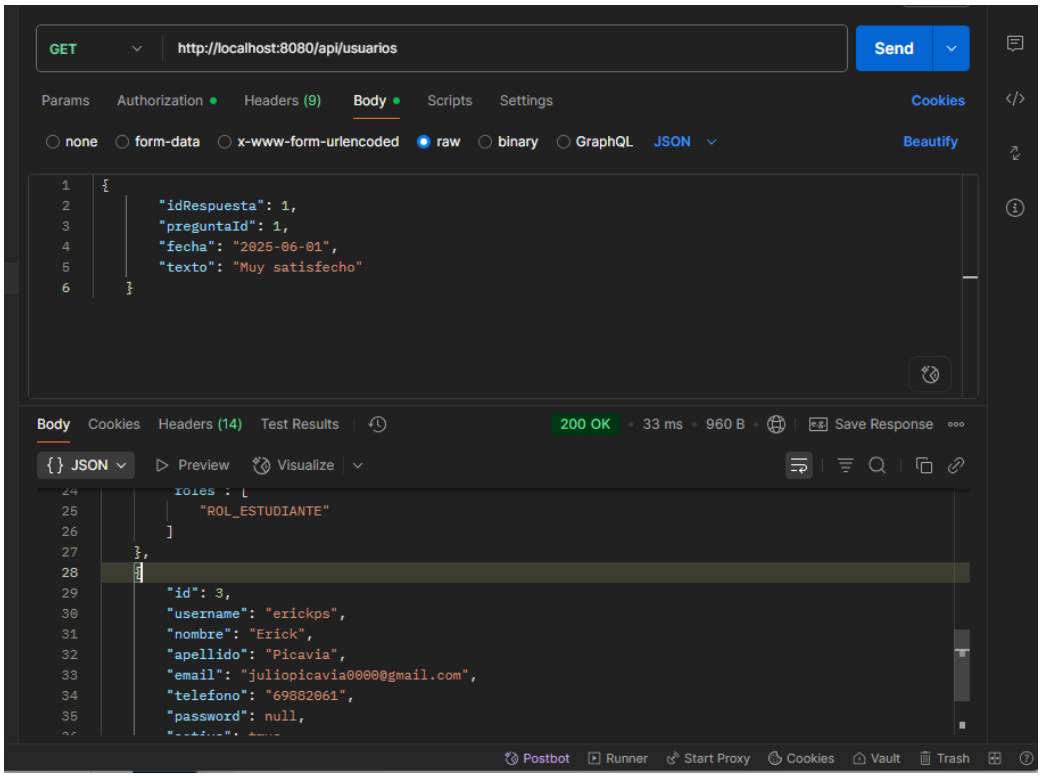
6. Pruebas y Depuración

· Pruebas funcionales con Postman.

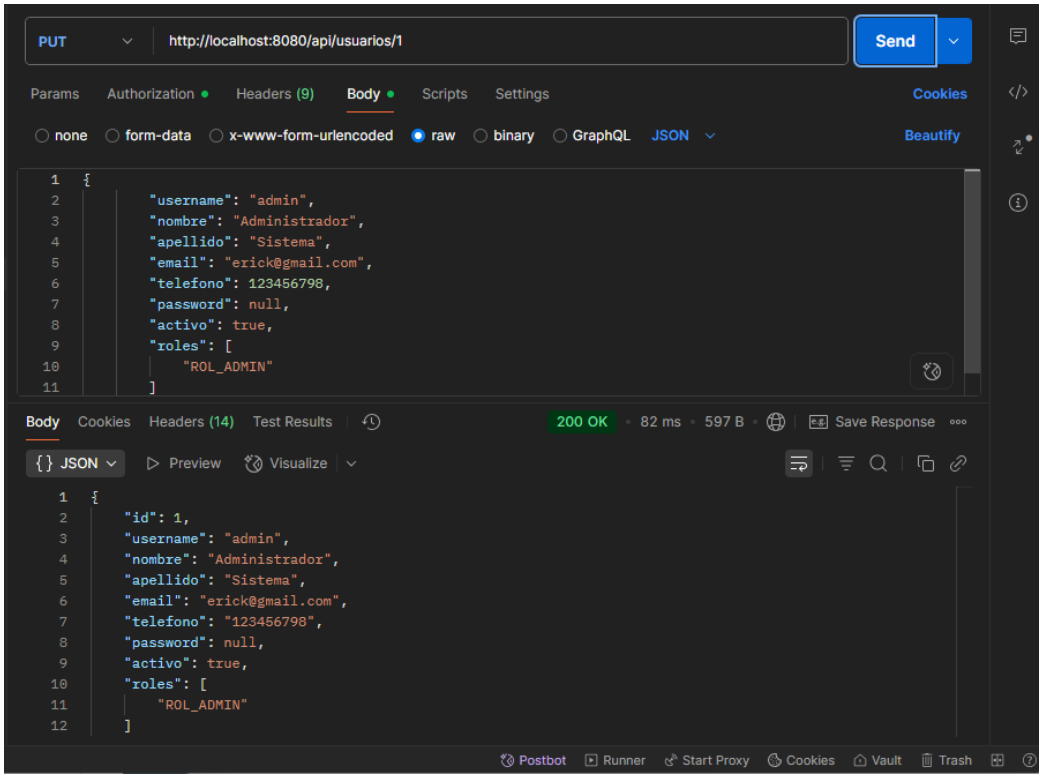
Aca entramos al usuario por defecto con el rol de ADMIN



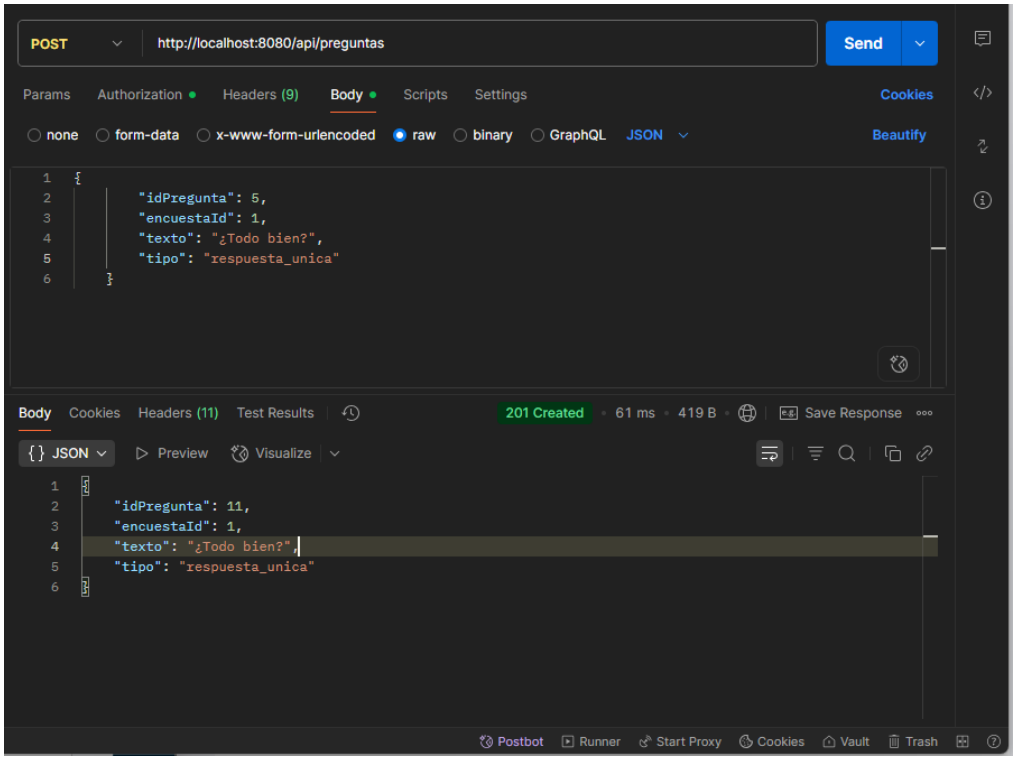
Luego entrando con la autorizacion de bearer token, podemos ver diferentes rutas pero en esta en especifico se mostrar el GET de Usuarios:



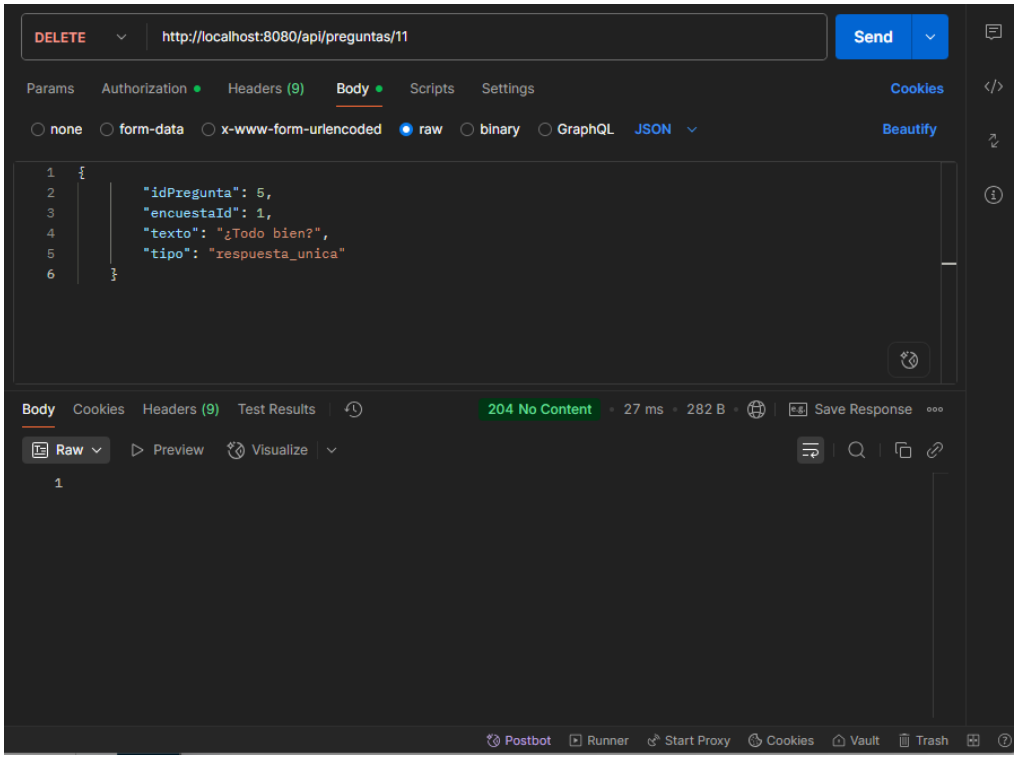
Luego lo editamos teniendo el rol logueado usando el metodo PUT:



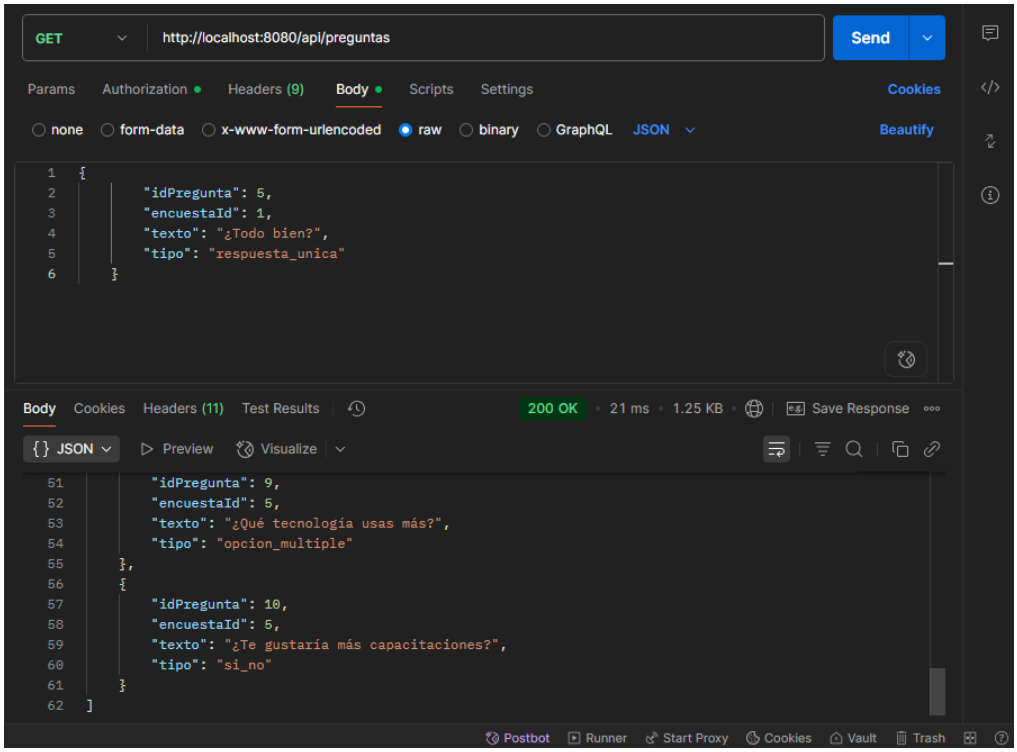
luego creamos una nueva pregunta usando el POST



ahora la borramos para probar el DELETE:



ahora corroboramos que la pregunta ya no este con un GET:



vemos que todo se cumplio correctamente asi que ahi termina la prueba.

7. DevOps y Entorno de Ejecución (No Aplica)

8. Reportes y Funcionalidades Extra(No Aplica)

9. Organización del Equipo

INTEGRANTE	ROL
Carlos Javier Callizaya Rosas	Desarrollo del Backend, rutas, y modelos
Carlos Javier Canaviri Huaranca	Desarrollo del Backend, documentación
Joaquin Humberto Loza Fernandez	Desarrollo de la Base de Datos, crear entidades y relaciones
Iver Pedro Mamani Cordero	Desarrollo del Backend, seguridad, y autenticación, Desarrollo del Frontend en Next Js
Erick Julio Picavia Saravia	Desarrollo del backend, rutas, modelos, DTOs, y desarrollo de la base de datos, elaboración de los scripts para entidades y relaciones

Estrategias de Colaboracion en Github: Usamo una rama principal Main para subir el resultado final, y ramas para subir propuestas que se fueron haciendo durando el desarrollo del proyecto.

10. Evaluación y Conclusiones

Se pudo trabajar de buena manera, debido a la comunicación y la división de tareas, para avanzar de forma mas acelerada, los principales retos fueron a la hora de unir el trabajo de todos ya que cada uno cuenta con perspectivas diferentes a la hora de resolver el problema.

11. Anexos

Script de la BD:

```
-- Secuencias
CREATE SEQUENCE answers_id_seq;
CREATE SEQUENCE question_options_id_seq;
CREATE SEQUENCE questions_id_seq;
CREATE SEQUENCE survey_responses_id_seq;
CREATE SEQUENCE surveys_id_seq;
CREATE SEQUENCE usuarios_id_seq;

-- Tabla: roles
CREATE TABLE roles (
    id BIGINT GENERATED BY DEFAULT AS IDENTITY PRIMARY KEY,
    nombre VARCHAR(20)
);
```

```
-- Tabla: usuarios
CREATE TABLE usuarios (
    id BIGINT PRIMARY KEY DEFAULT nextval('usuarios_id_seq'),
    activo BOOLEAN NOT NULL,
    apellido VARCHAR(255),
    email VARCHAR(255) NOT NULL,
    nombre VARCHAR(255),
    password VARCHAR(255) NOT NULL,
    username VARCHAR(255) NOT NULL
);

-- Tabla: usuario_roles (relación muchos a muchos entre usuarios y roles)
CREATE TABLE usuario_roles (
    usuario_id BIGINT NOT NULL,
    rol_id BIGINT NOT NULL,
    PRIMARY KEY (usuario_id, rol_id),
    FOREIGN KEY (usuario_id) REFERENCES usuarios(id),
    FOREIGN KEY (rol_id) REFERENCES roles(id)
);

-- Tabla: surveys
CREATE TABLE surveys (
    id BIGINT PRIMARY KEY DEFAULT nextval('surveys_id_seq'),
    title VARCHAR(255) NOT NULL,
```

```
    description VARCHAR(255),
    created_at DATE,
    updated_at DATE,
    active BOOLEAN DEFAULT true,
    user_id BIGINT,
    FOREIGN KEY (user_id) REFERENCES usuarios(id)
);

-- Tabla: questions
CREATE TABLE questions (
    id BIGINT PRIMARY KEY DEFAULT nextval('questions_id_seq'),
    survey_id BIGINT NOT NULL,
    text VARCHAR(255) NOT NULL,
    question_type VARCHAR(255) NOT NULL,
    order_index INTEGER,
    required BOOLEAN,
    FOREIGN KEY (survey_id) REFERENCES surveys(id)
);

-- Tabla: question_options
CREATE TABLE question_options (
    id BIGINT PRIMARY KEY DEFAULT nextval('question_options_id_seq'),
    question_id BIGINT NOT NULL,
    text VARCHAR(255) NOT NULL,
```

```
    order_index INTEGER,
    FOREIGN KEY (question_id) REFERENCES questions(id)
);

-- Tabla: survey_responses
CREATE TABLE survey_responses (
    id BIGINT PRIMARY KEY DEFAULT nextval('survey_responses_id_seq'),
    survey_id BIGINT NOT NULL,
    respondent_email VARCHAR(255),
    submitted_at TIMESTAMP,
    FOREIGN KEY (survey_id) REFERENCES surveys(id)
);

-- Tabla: answers
CREATE TABLE answers (
    id BIGINT PRIMARY KEY DEFAULT nextval('answers_id_seq'),
    question_id BIGINT NOT NULL,
    survey_response_id BIGINT NOT NULL,
    text_answer TEXT,
    numeric_answer DOUBLE PRECISION,
    date_answer DATE,
    selected_option_id BIGINT,
    FOREIGN KEY (question_id) REFERENCES questions(id),
    FOREIGN KEY (survey_response_id) REFERENCES survey_responses(id),
```

```
    FOREIGN KEY (selected_option_id) REFERENCES question_options(id)
);

-- Tabla: spring_session
CREATE TABLE spring_session (
    primary_id CHAR(36) PRIMARY KEY,
    session_id CHAR(36) NOT NULL,
    creation_time BIGINT NOT NULL,
    last_access_time BIGINT NOT NULL,
    max_inactive_interval INTEGER NOT NULL,
    expiry_time BIGINT NOT NULL,
    principal_name VARCHAR(100)
);

-- Tabla: spring_session_attributes
CREATE TABLE spring_session_attributes (
    session_primary_id CHAR(36) NOT NULL,
    attribute_name VARCHAR(200) NOT NULL,
    attribute_bytes BYTEA,
    FOREIGN KEY (session_primary_id) REFERENCES spring_session(primary_id)
);
```

Codigo Fuente del Proyecto:

https://github.com/Iver011/proyecto-TAW/tree/master