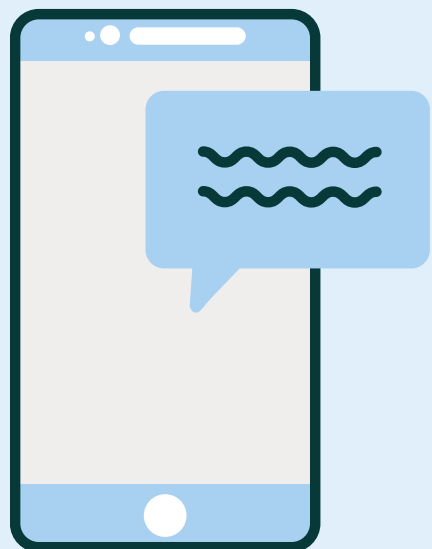




# TRANSFORMACIÓN DIGITAL EDUTECH INNOVATORS SPA

Benjamín Torrejón Soto – Alejandra Reyes Duque  
Desarrollo Fullstack I 002D

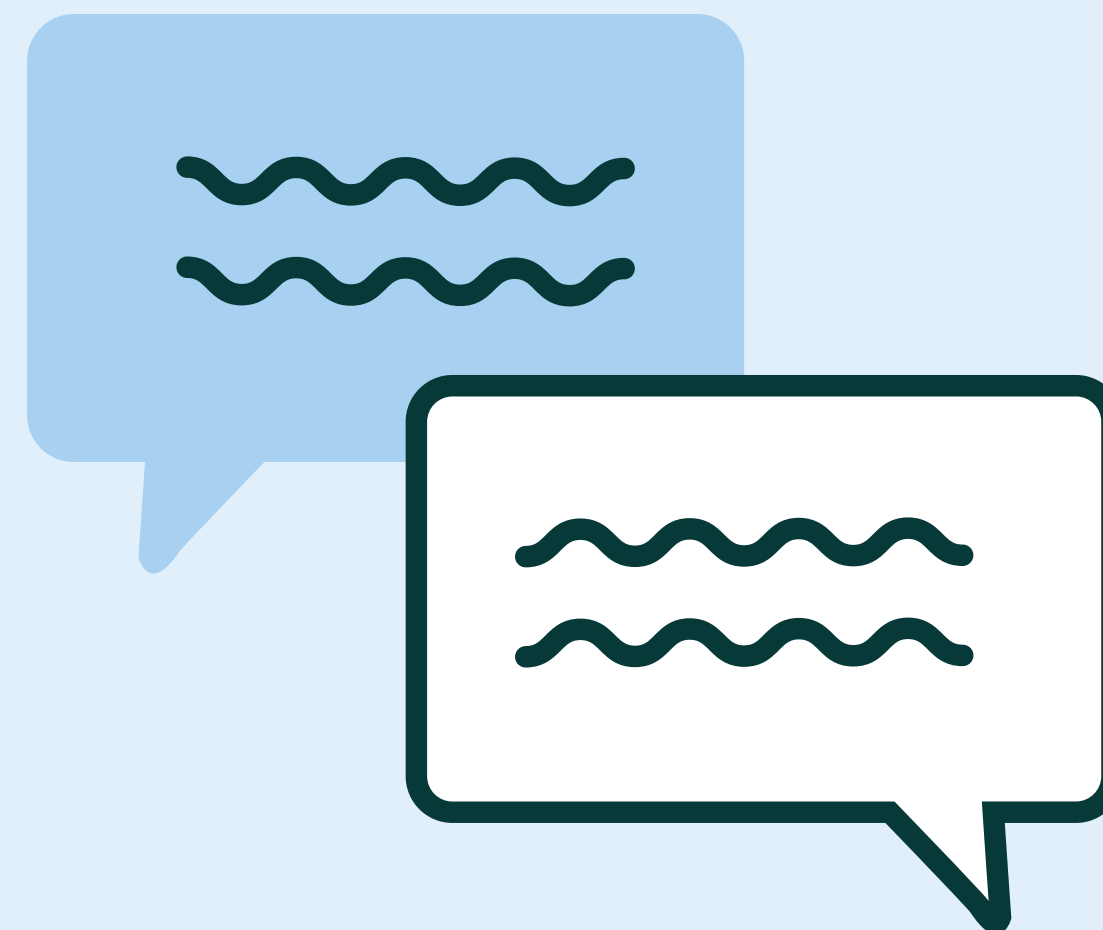


# INTRODUCCIÓN

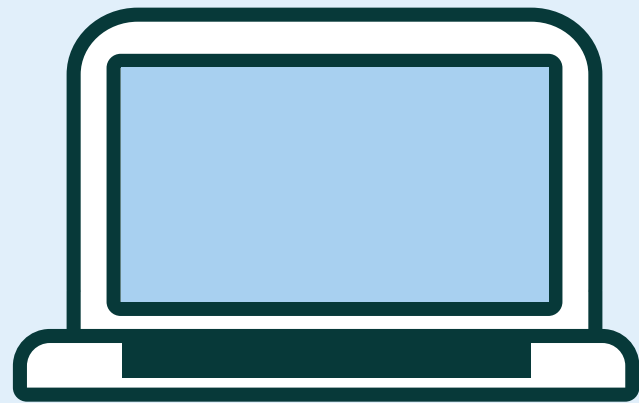
## Problema a abordar

### EDUTECH INNOVATORS SPA ENFRENTA PROBLEMAS CON SU SISTEMA MONOLÍTICO

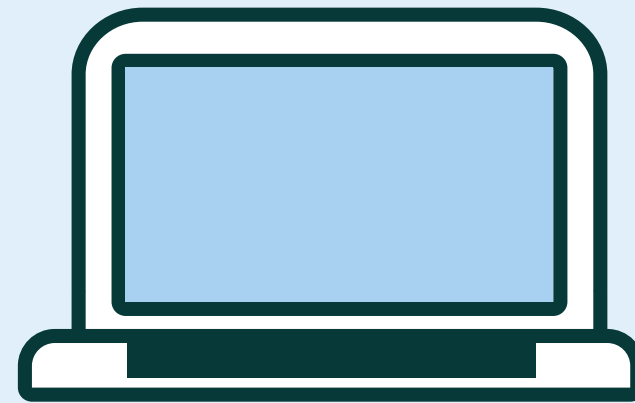
Empresa chilena emergente que desarrolla plataformas educativas en línea. Originada en Providencia, Santiago, ha crecido rápidamente con nuevas oficinas en Valparaíso y La Serena. Sin embargo, su software monolítico presenta limitaciones que afectan el rendimiento y la disponibilidad, impactando las operaciones y la satisfacción del cliente, lo que representa un riesgo para su expansión futura.



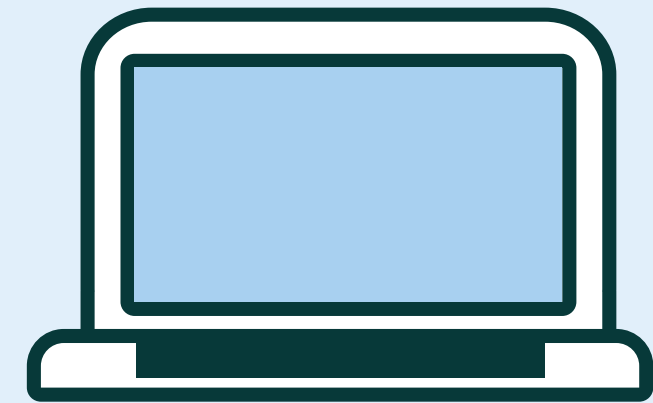
# EJECUCIÓN DE SERVICIOS



**USUARIO**



**EVALUACIONES**



**CURSOS**

# LÍNEA BASE Y HOJA DE RUTA

## LÍNEA BASE

- Commit inicial
  - Fecha:
  - Hash:
- Ramas:
  - feature/usuarios
  - feature/cursos
  - feature/evaluaciones
- Decisiones clave:
  - H2 embebida vs MySQL
  - Modelado JPA antes de controladores
  - CORS centralizado

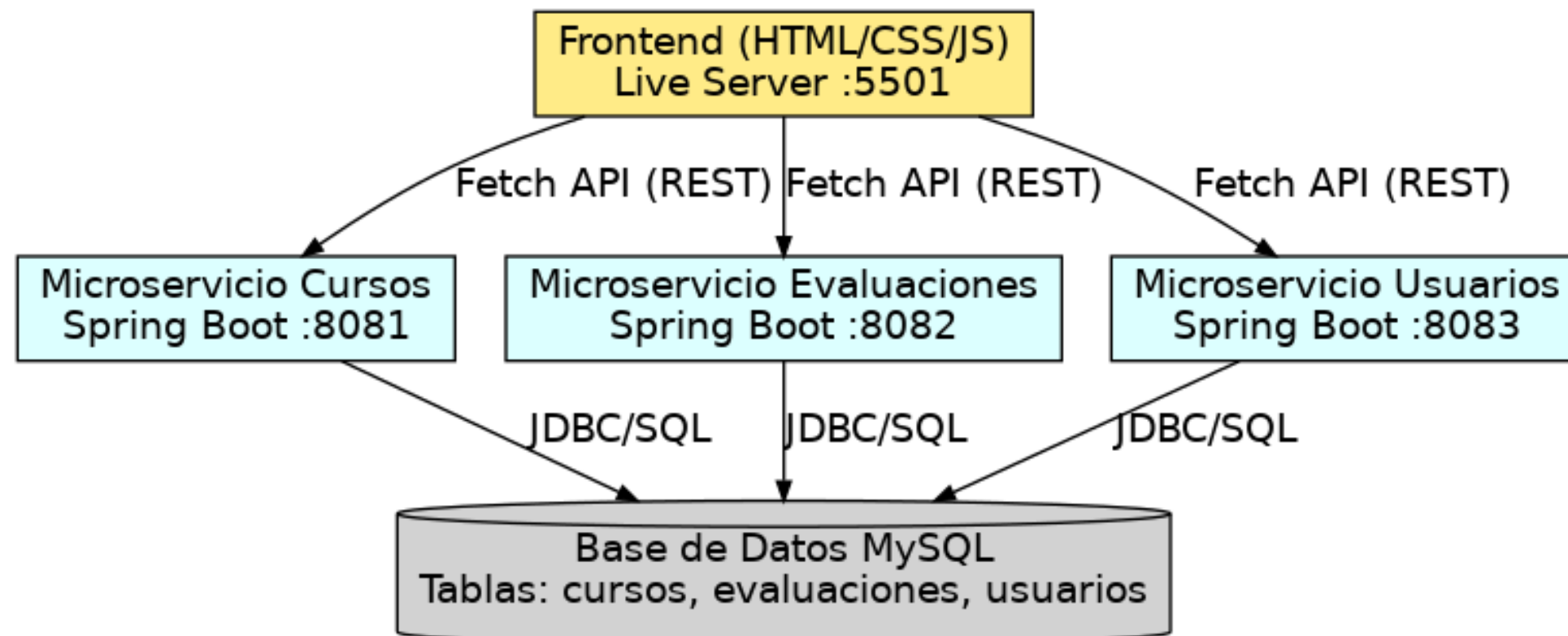
## HOJA DE RUTA

1. Diseño ER y entidades JPA
2. Repositorios y servicios CRUD
3. Controladores REST y pruebas con Postman
4. Integración frontend + ajustes CORS
5. Pruebas unitarias e integración continua (CI/CD)



# ARQUITECTURA INTERNA E INTEGRACIÓN FULL-STACK

Frontend servido con Live Server de VSCode en <http://localhost:5500/>



El index.html invoca:

`cursos.js → GET`

`http://localhost:8081/api/cursos`

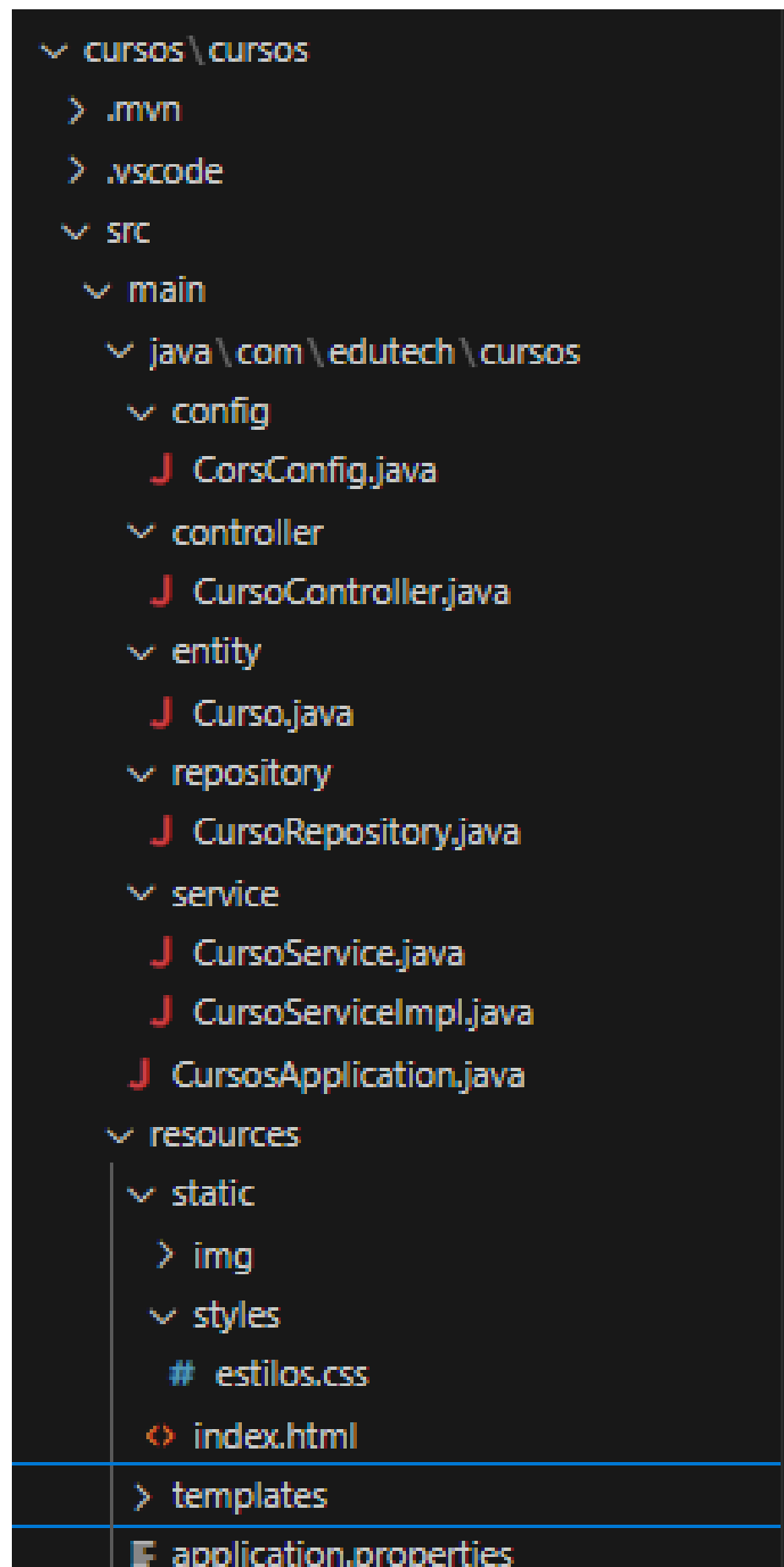
`usuarios.js → GET`

`http://localhost:8083/api/usuarios`

`evaluaciones.js → GET`

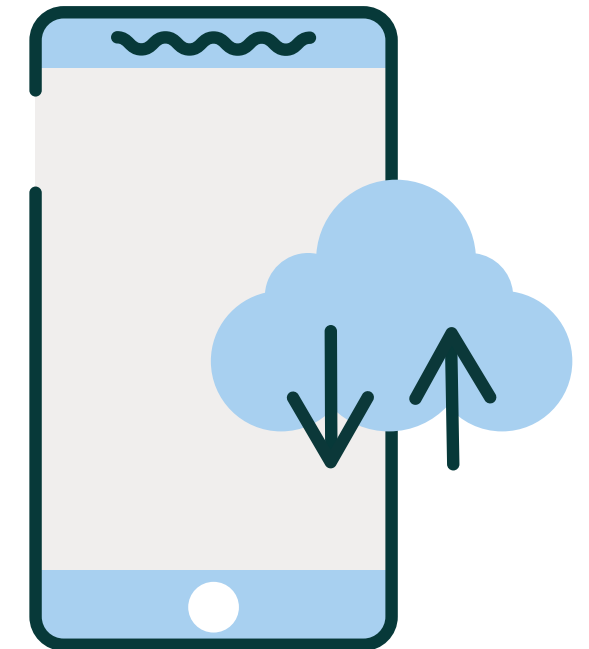
`http://localhost:8082/api/evaluaciones`

Luego procesa el JSON recibido e inyecta dinámicamente el contenido en el DOM.



# ESTRUCTURA

## Spring Boot



config

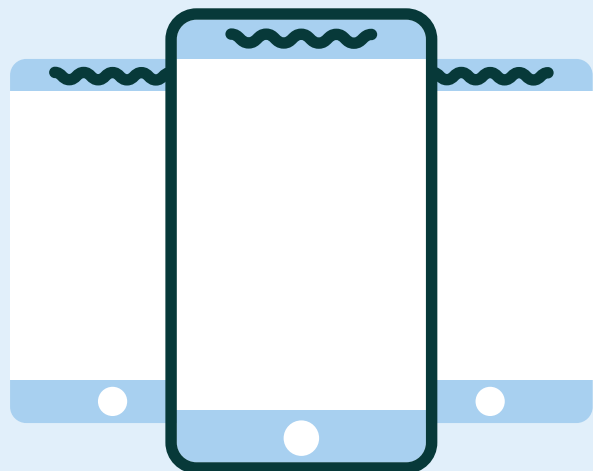
controller

entity

repository

service

main



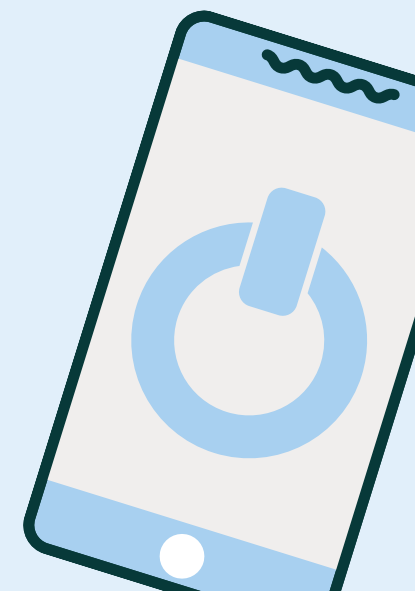
# MANEJO DE DEPENDENCIAS Y CICLO DE VIDA

Maven

Para gestionar las dependencias y facilitar la construcción del proyecto

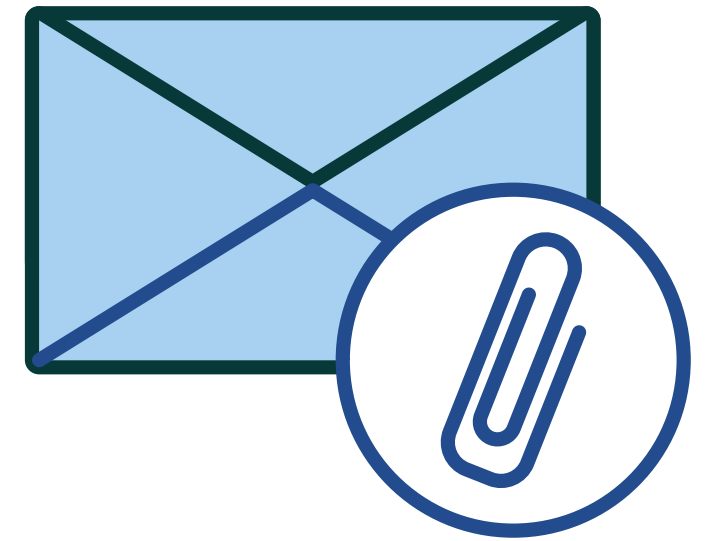
pom.xml

Ejecutamos comandos básicos





# BUENAS PRÁCTICAS Y ARQUITECTURA MODULAR



**01**

Patrón de arquitectura de software MVC

**03**

Inyección de dependencias

**02**

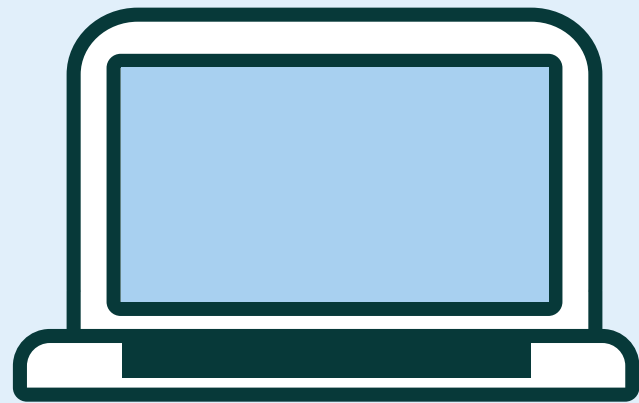
Separación de responsabilidades

**04**

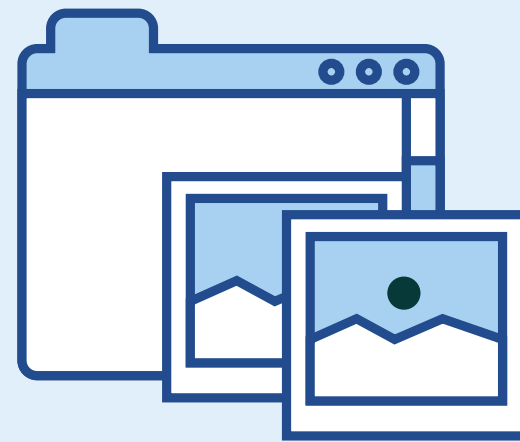
Código modular y reutilizable



# BASE DE DATOS Y CONEXIÓN



**MOTOR DE BASE DE  
DATOS**



**ESTRUCTURA DE TABLAS**



**CONEXIÓN Y PRUEBAS**

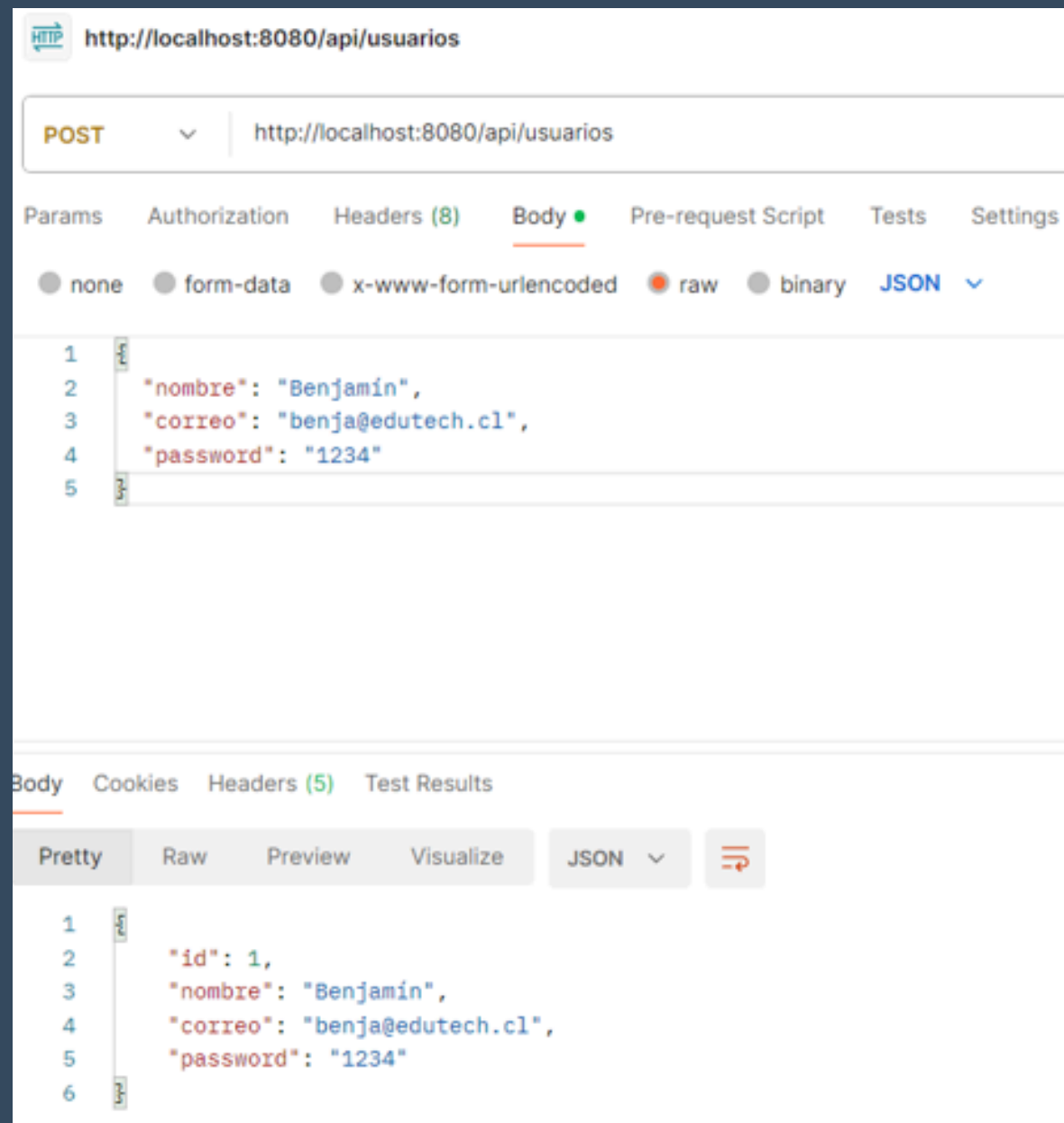
# POSTMAN

Validando la comunicación  
entre microservicios

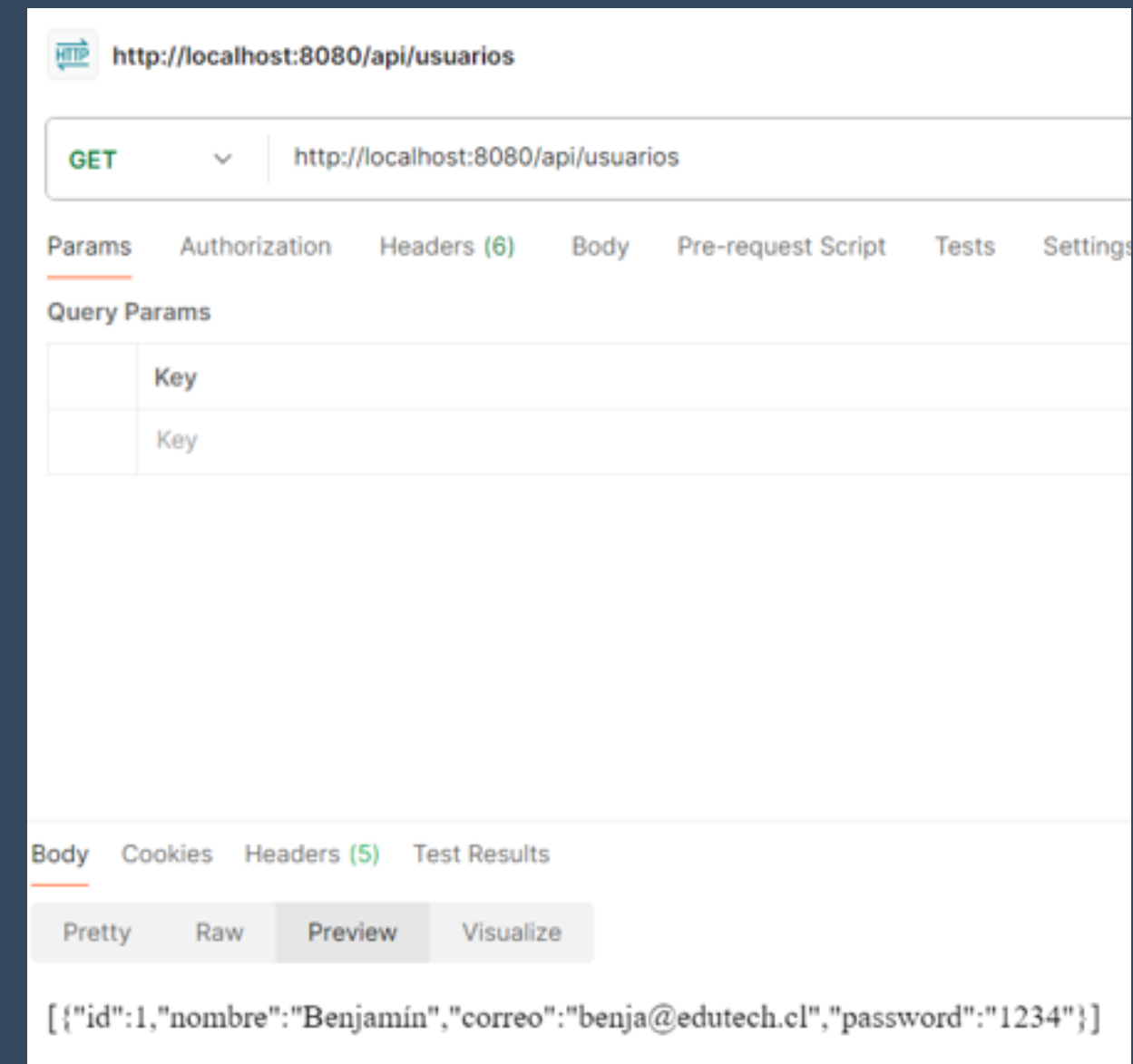


# Evidencias Postman “Usuario”

## POST

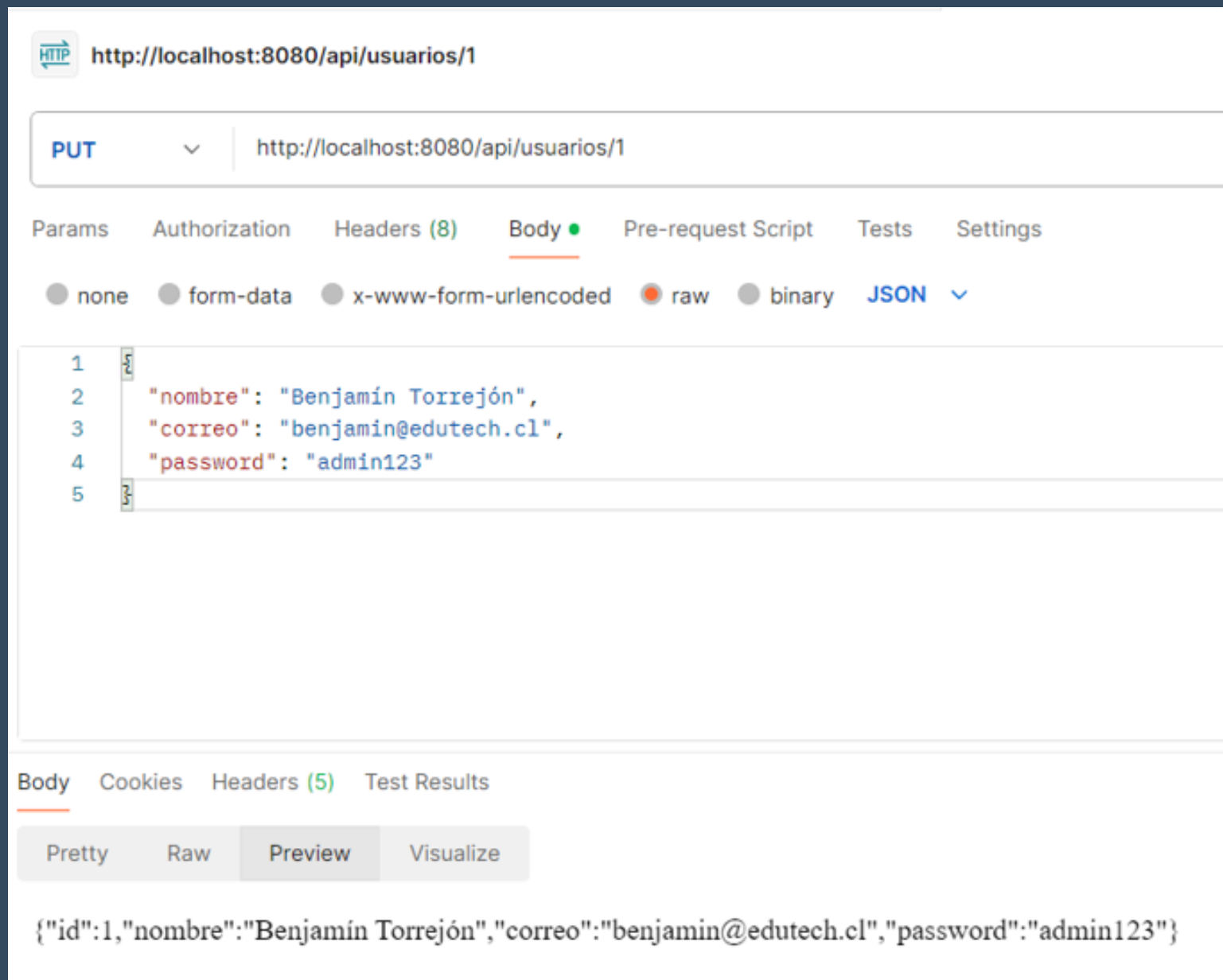


## GET



# Evidencias Postman “Usuario”

## PUT



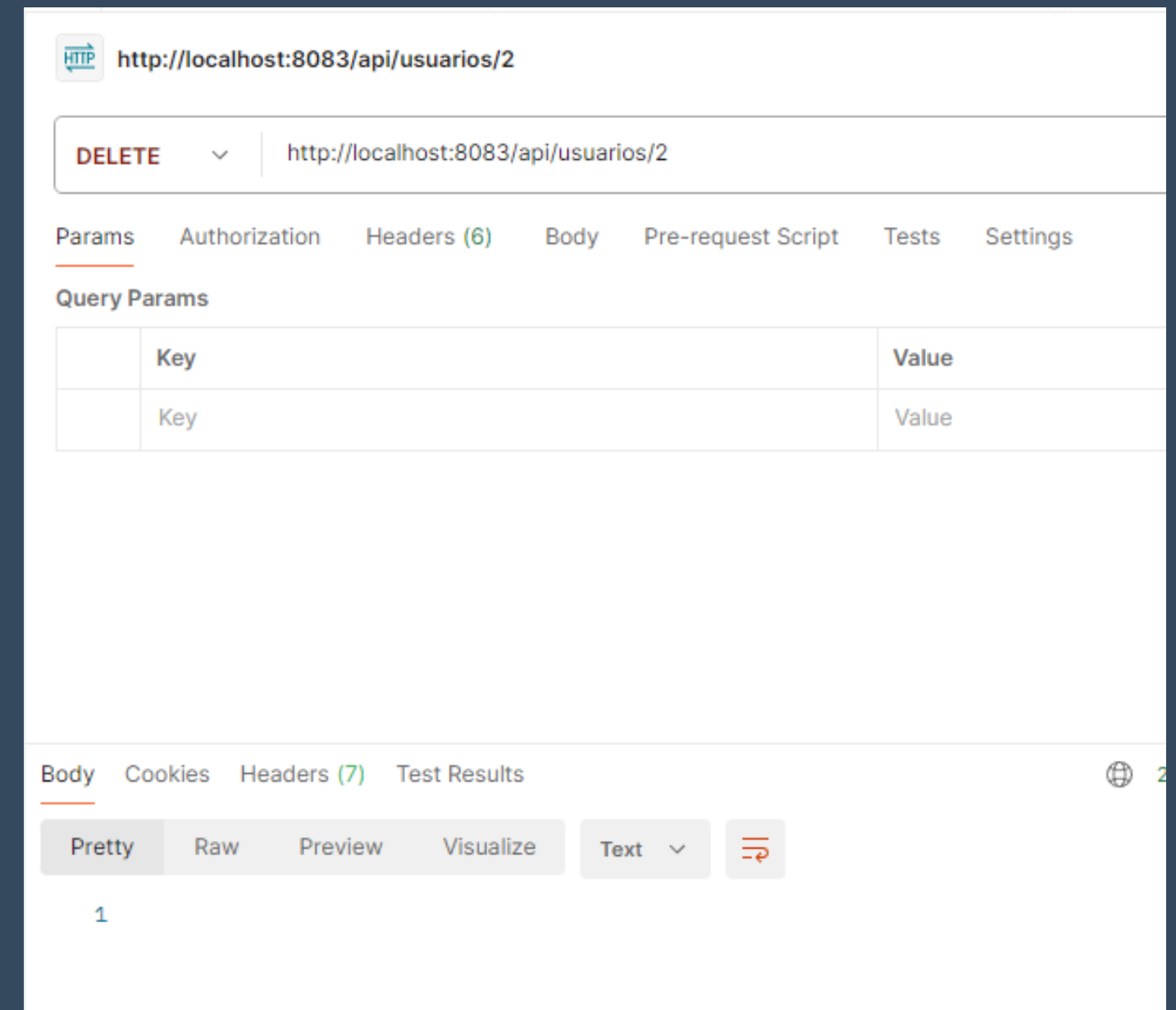
The image shows a Postman interface for a PUT request. The URL bar at the top displays `http://localhost:8080/api/usuarios/1`. Below the URL bar, the HTTP method is set to **PUT**. The interface includes tabs for Params, Authorization, Headers (8), Body, Pre-request Script, Tests, and Settings. The Body tab is selected, and the content type is set to **JSON**. The request body is a JSON object with the following fields:

```
1 {
2   "nombre": "Benjamín Torrejón",
3   "correo": "benjamin@edutech.cl",
4   "password": "admin123"
5 }
```

At the bottom, the response body is displayed in the Pretty view, showing the following JSON object:

```
{"id":1,"nombre":"Benjamín Torrejón","correo":"benjamin@edutech.cl","password":"admin123"}
```

## DELETE



The image shows a Postman interface for a DELETE request. The URL bar at the top displays `http://localhost:8083/api/usuarios/2`. Below the URL bar, the HTTP method is set to **DELETE**. The interface includes tabs for Params, Authorization, Headers (6), Body, Pre-request Script, Tests, and Settings. The Body tab is selected, and the content type is set to **Text**. The request body is empty. At the bottom, the response body is displayed in the Pretty view, showing the following JSON object:

```
1 {}
```

# Evidencias Postman “Cursos”

## POST

POST ⌵ http://localhost:8081/api/cursos

Params Authorization Headers (8) **Body** ● Pre-request Script Tests Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary **JSON** ⌵

```
1 {
2   "nombre": "Curso de Java Spring",
3   "descripcion": "Aprende a hacer APIs REST con Spring Boot",
4   "duracion": 60
5 }
```

Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize **JSON** ⌵

```
1 {
2   "id": 1,
3   "nombre": "Curso de Java Spring",
4   "descripcion": "Aprende a hacer APIs REST con Spring Boot",
5   "instructor": null,
6   "duracion": 60
7 }
```

## GET

http://localhost:8081/api/cursos

**GET** ⌵ http://localhost:8081/api/cursos

Params Authorization Headers (6) Body Pre-request Script Tests Settings

**Query Params**

	Key
	Key

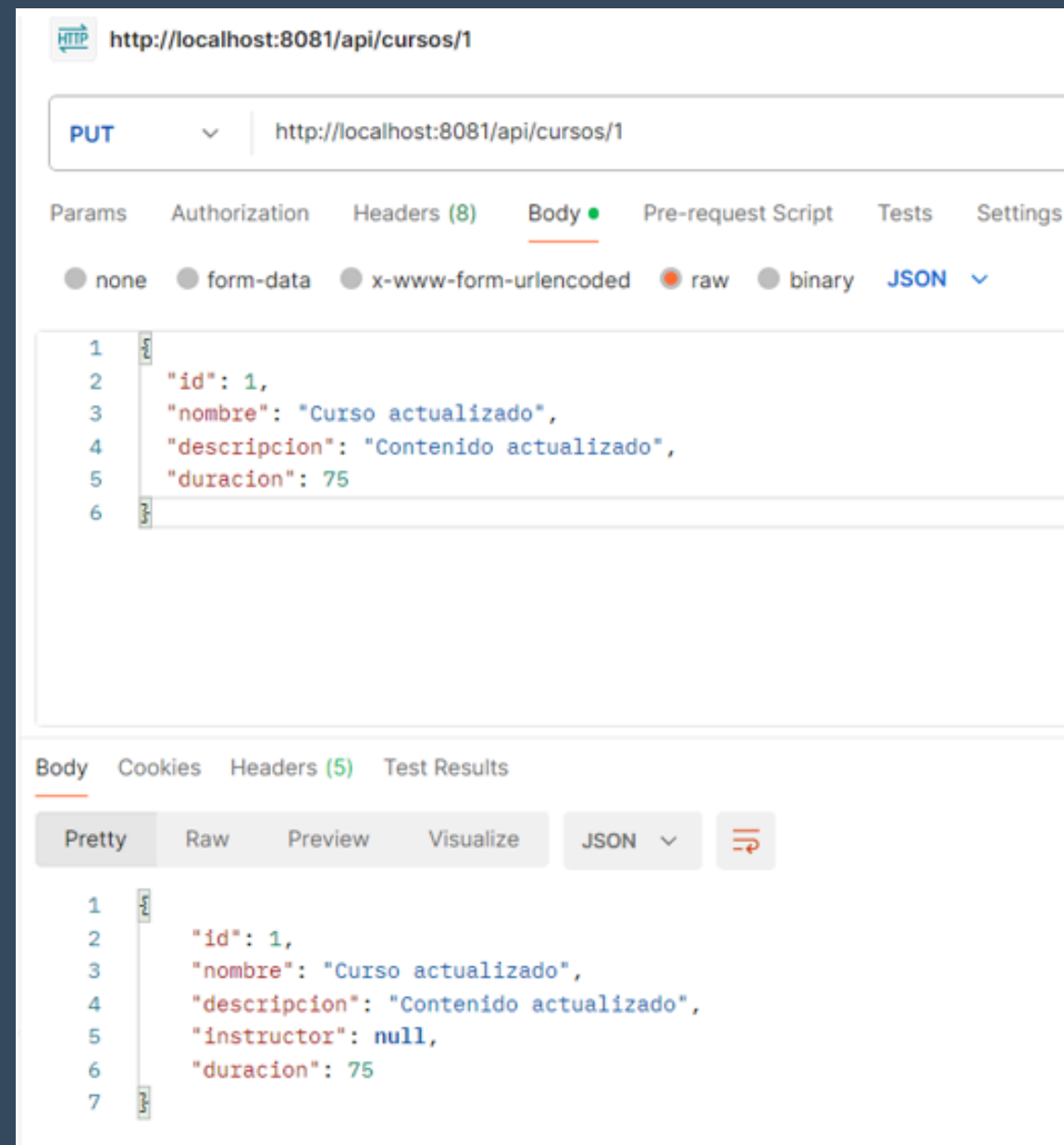
Body Cookies Headers (5) Test Results

Pretty Raw Preview Visualize **JSON** ⌵

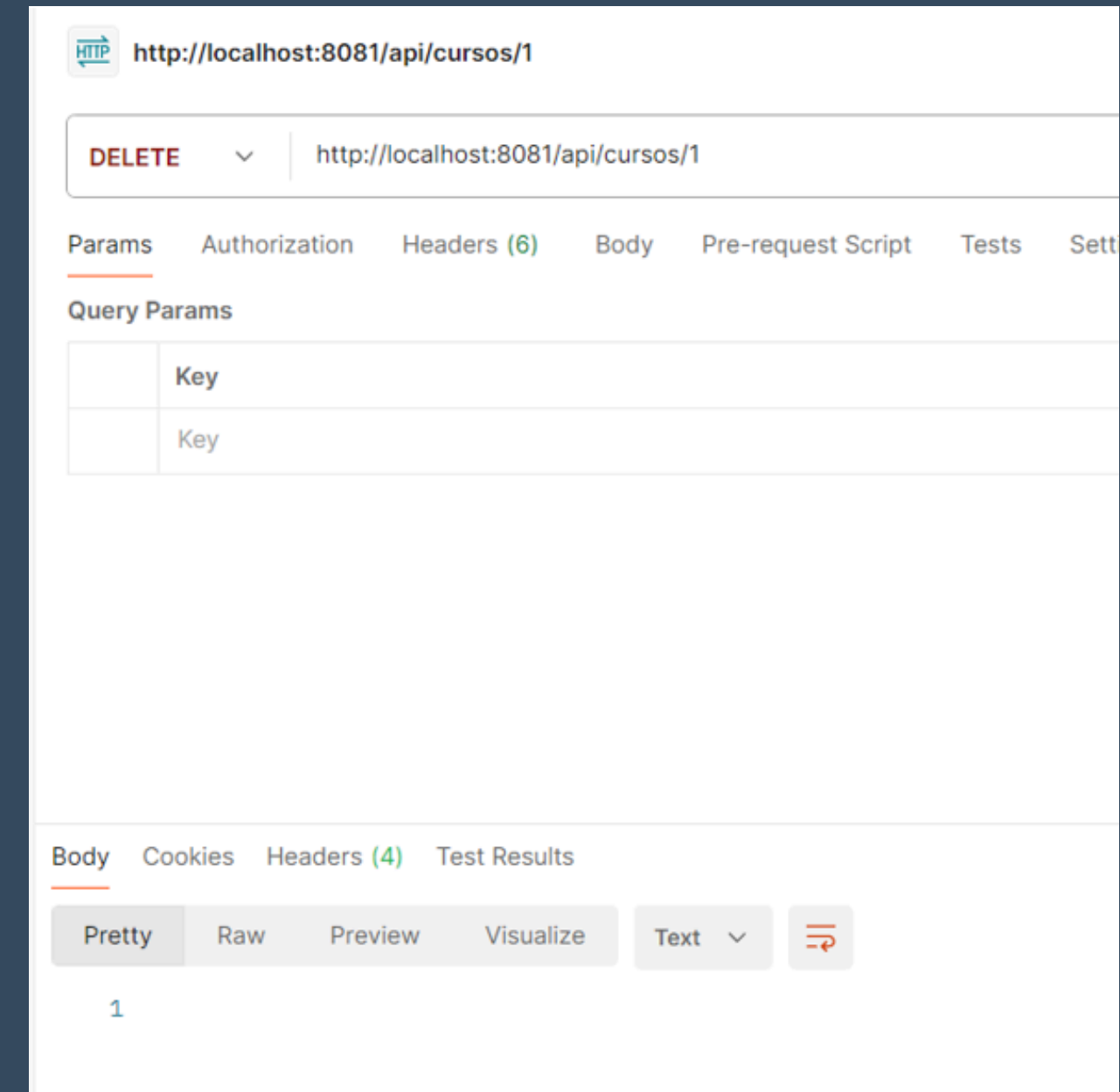
```
1 [
2   {
3     "id": 1,
4     "nombre": "Curso de Java Spring",
5     "descripcion": "Aprende a hacer APIs REST con Spring Boot",
6     "instructor": null,
7     "duracion": 60
8   }
9 ]
```

# Evidencias Postman “Cursos”

## PUT



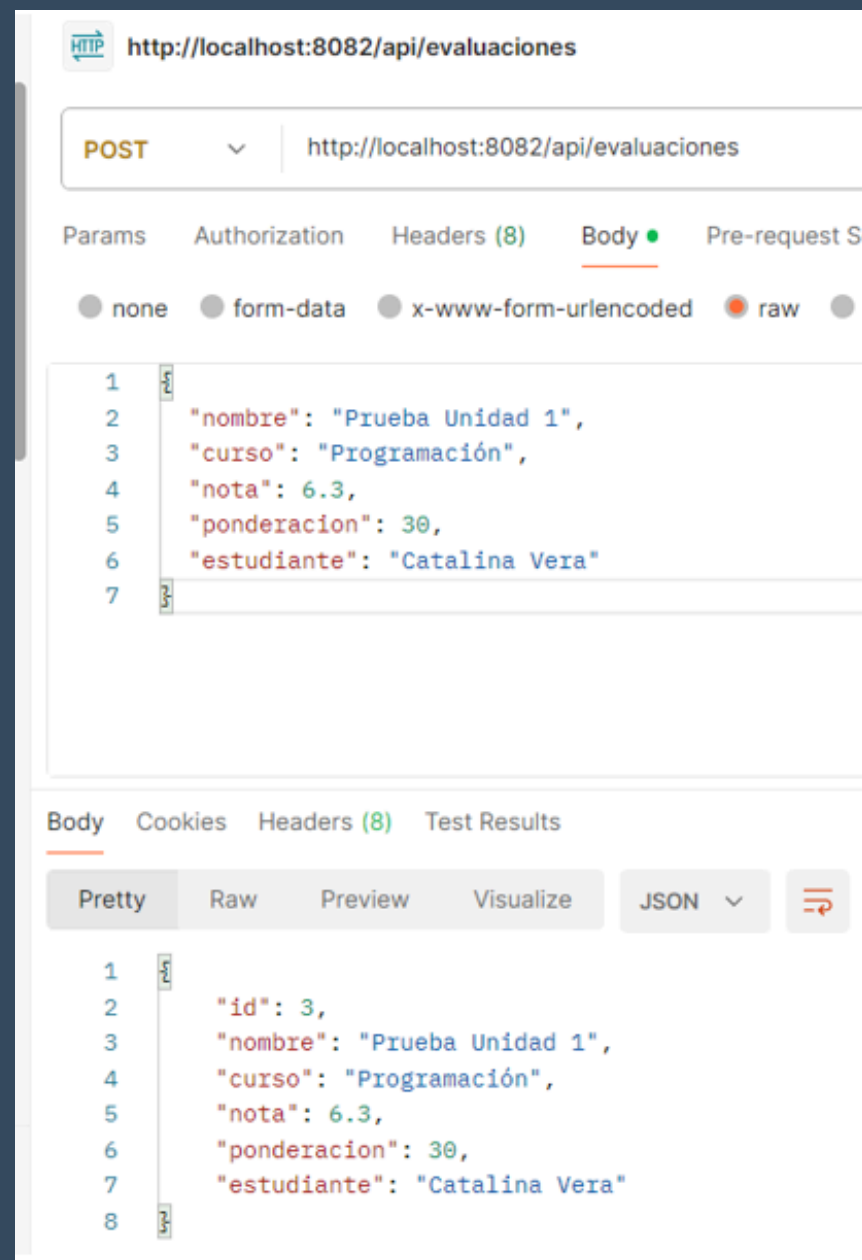
## DELETE



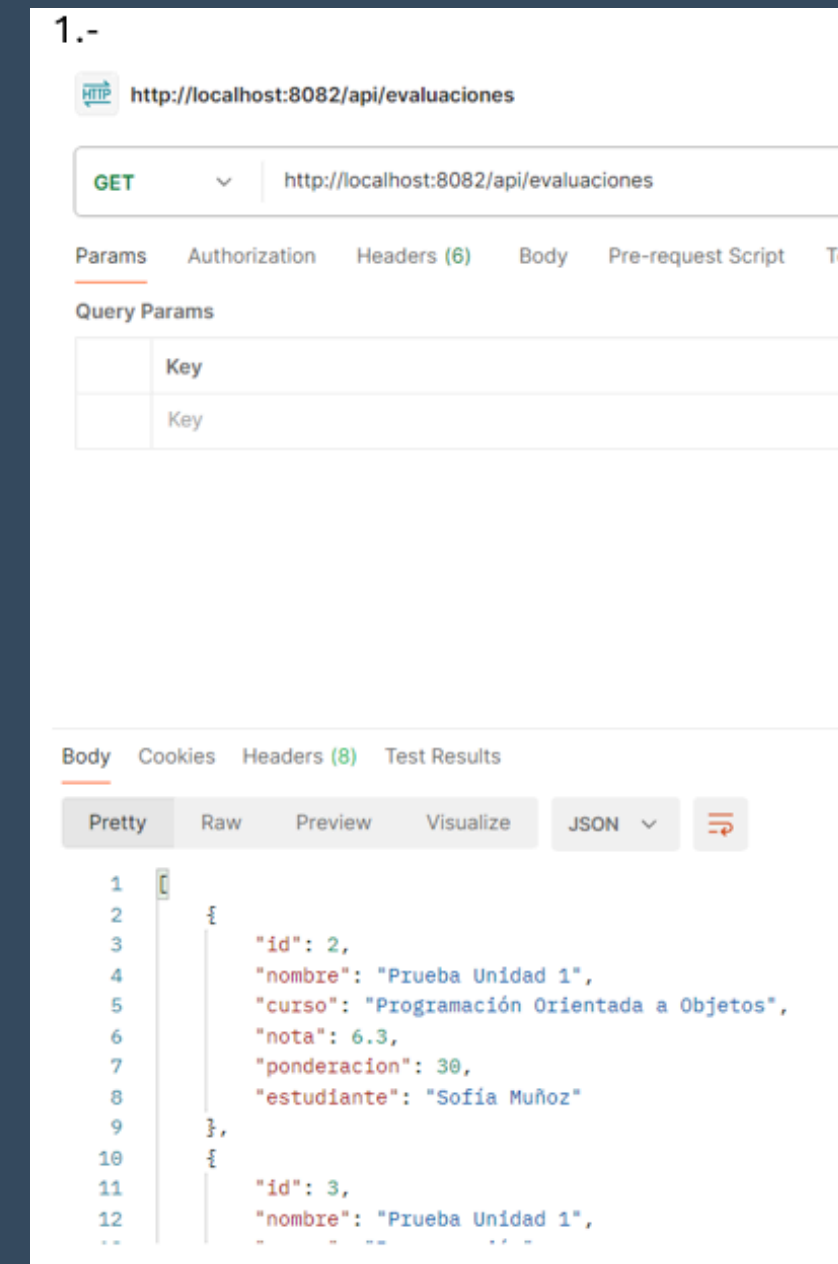


# Evidencias Postman “Evaluaciones”

## POST

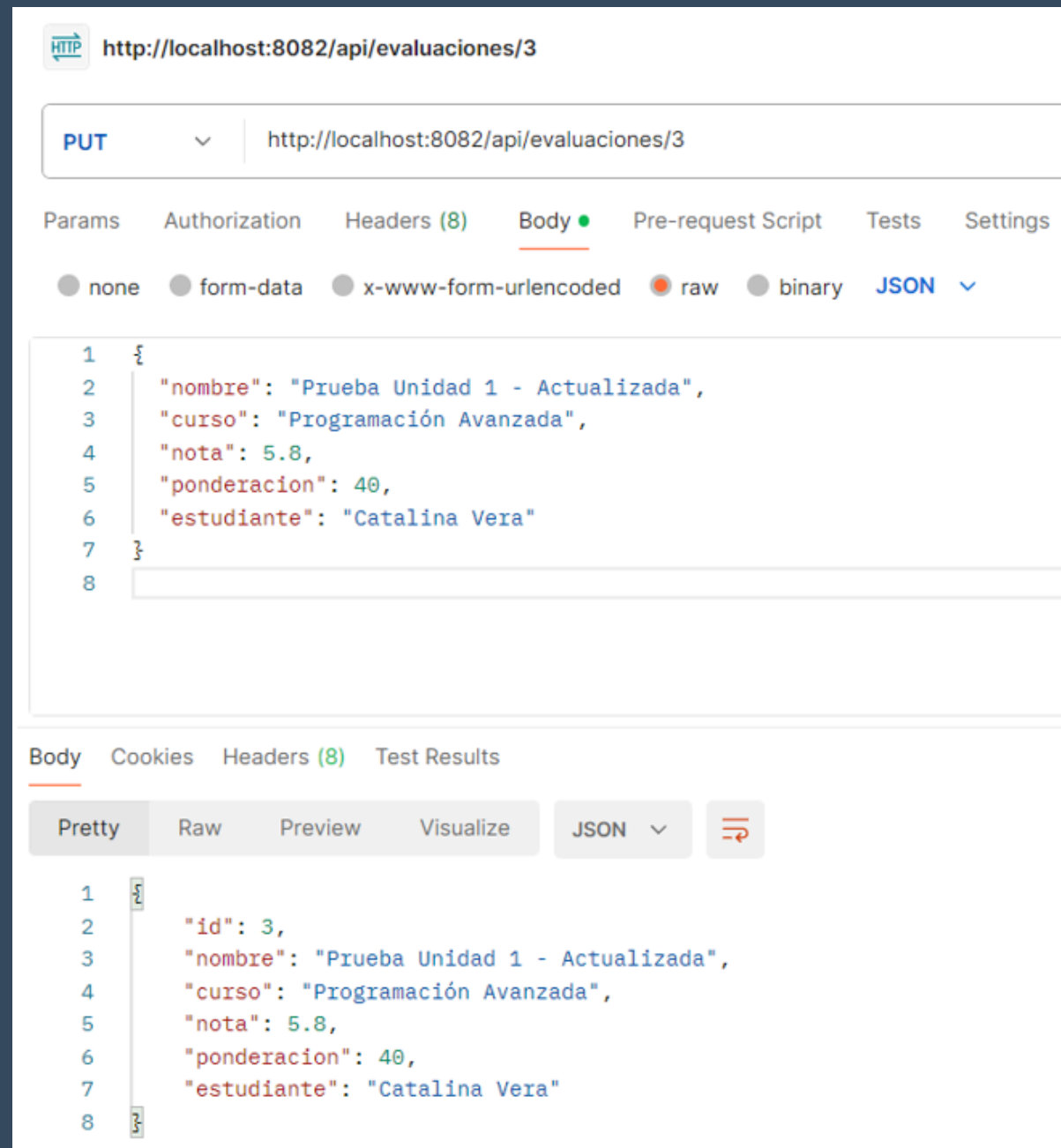


## GET

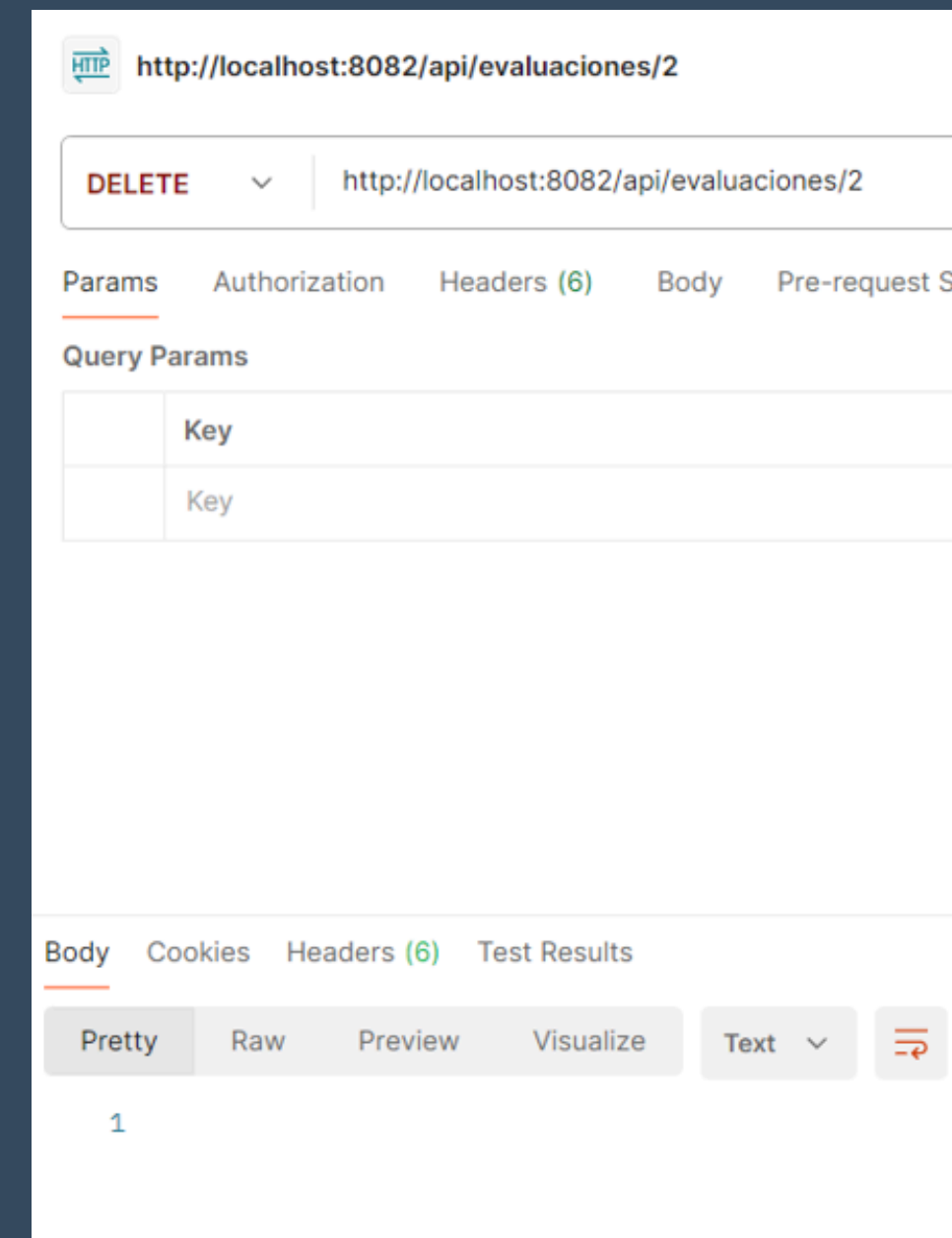


# Evidencias Postman “Evaluaciones”

## PUT



## DELETE



# TRABAJO COLECTIVO

Usamos **Git y GitHub** para coordinar el equipo. Git gestionó las versiones del código y facilitó la colaboración, mientras que GitHub centralizó y revisó las aportaciones. Además, complementamos este trabajo con comunicación constante mediante WhatsApp y reuniones presenciales. Esto garantizó un flujo de trabajo ordenado, mejorando la comunicación y la integración de los cambios de cada integrante.





**MUCHAS GRACIAS**  
**POR SU ATENCIÓN**