Informe:

“Desarrollo de Microservicios con Java y Spring Boot”

**Integrantes:**

* Javiera Curin
* Vicente Quiroz
* Paula Rojas

**Nombre del docente:** Juan Carlos Molina

**Asignatura** : Ingeniería Software

# 1. Introducción

## Introducción

Este informe presenta el desarrollo de tres microservicios para la plataforma educativa EduTech, como parte de una migración desde una arquitectura monolítica hacia una arquitectura basada en microservicios. Este cambio permitió transformar el sistema en una solución web más escalable, flexible y fácil de mantener.

El microservicio de Gestión Académica se encarga de la creación de cursos, evaluaciones, preguntas y respuestas con sus respectivos puntajes. El microservicio de Gestión de Usuario administra los datos y roles de estudiantes, docentes y administradores. Finalmente, el microservicio de Gestión de Pago registra y controla las transacciones económicas realizadas dentro de la plataforma.

Cada microservicio fue desarrollado con Java y Spring Boot, utiliza MySQL como base de datos y se probó con Postman para garantizar su correcto funcionamiento. Gracias a esta estructura modular, EduTech ahora cuenta con un sistema web más robusto, adaptable y preparado para su crecimiento futuro.

# ● 1.1 Objetivo del proyecto

Desarrollar una plataforma educativa web para EduTech basada en una arquitectura de microservicios, que permita gestionar de forma independiente y eficiente los módulos de Gestión Académica, Gestión de Usuario y Gestión de Pago, con el fin de mejorar la escalabilidad, flexibilidad y mantenimiento del sistema, optimizando la administración de cursos, usuarios y transacciones dentro del entorno educativo.

# ● 1.2 Tecnologías utilizadas

Para el desarrollo de los microservicios del sistema EduTech, se utilizaron las siguientes tecnologías:

* **Java 17:** Lenguaje de programación principal para el desarrollo de los microservicios.
* **Spring Boot** : Framework que facilita la creación de aplicaciones Java con configuración mínima, ideal para microservicios REST.
* **MySQL**: Sistema de gestión de bases de datos relacional, utilizado para almacenar la información de usuarios, cursos, evaluaciones y pagos.
* **Postman:** Herramienta utilizada para realizar pruebas de las API REST, verificando el correcto funcionamiento de los endpoints.
* **Maven:** Gestor de dependencias y construcción del proyecto.
* **Lombok:** Librería para reducir la escritura de código repetitivo (como getters, setters y constructores).
* **Spring Data JPA:** Abstracción para facilitar el acceso a datos mediante repositorios basados en entidades.
* **XAMPP:** Servidor local utilizado para ejecutar MySQL de forma local durante el desarrollo.
* **Git y GitHub:** Control de versiones y almacenamiento del proyecto en repositorio remoto.
* **Transbank Webpay (modo integración/test)**: Pasarela de pago utilizada para simular pagos reales mediante tarjetas de prueba, permitiendo validar el flujo completo de pago e inscripción.

# ● 1.3 Alcance general del sistema

El sistema desarrollado para la plataforma educativa EduTech abarca la creación de una solución web modular basada en microservicios, permitiendo una gestión eficiente y organizada de los procesos clave del entorno académico. El sistema contempla tres áreas funcionales principales:

* **Gestión Académica:** Permite la creación y administración de cursos, evaluaciones, preguntas y respuestas, incluyendo la definición de puntajes. Esta funcionalidad facilita la planificación y seguimiento del proceso de enseñanza-aprendizaje.
* **Gestión de Usuario y Tickets:** Administra el registro, autenticación y control de acceso de los distintos perfiles que interactúan con la plataforma, tales como estudiantes, docentes y administradores. Además, permite la asignación de roles, el manejo básico de información personal y la gestión de tickets de soporte, facilitando la creación, seguimiento y resolución de incidencias o consultas por parte de los usuarios.
* **Gestión de Pago e inscripciones:** Incluye la generación de cupones de pago y el registro de pagos por parte de los estudiantes. Este microservicio permite llevar un control de las transacciones asociadas a los cursos ofrecidos por la plataforma, asegurando una gestión ordenada y trazable de los ingresos.

**2. Diseño y Configuración del Proyecto**

# ● 2.1 Estructura del proyecto con Spring Boot

Detalles por carpeta:

* **configuración**: Contiene clases dedicadas a la configuración del sistema, como la inicialización de componentes, beans y otras dependencias necesarias para el funcionamiento de la aplicación.
* **controller:** Contiene las clases anotadas con **@RestController,** que exponen los endpoints para interactuar con el sistema a través de HTTP (GET, POST, PUT, DELETE).
* **model/entity/request/response**: Contiene las clases que representan las entidades del dominio, anotadas con **@Entity**, que se corresponden con tablas de la base de datos.
* **repository**: Contiene interfaces que extienden **JpaRepository** o

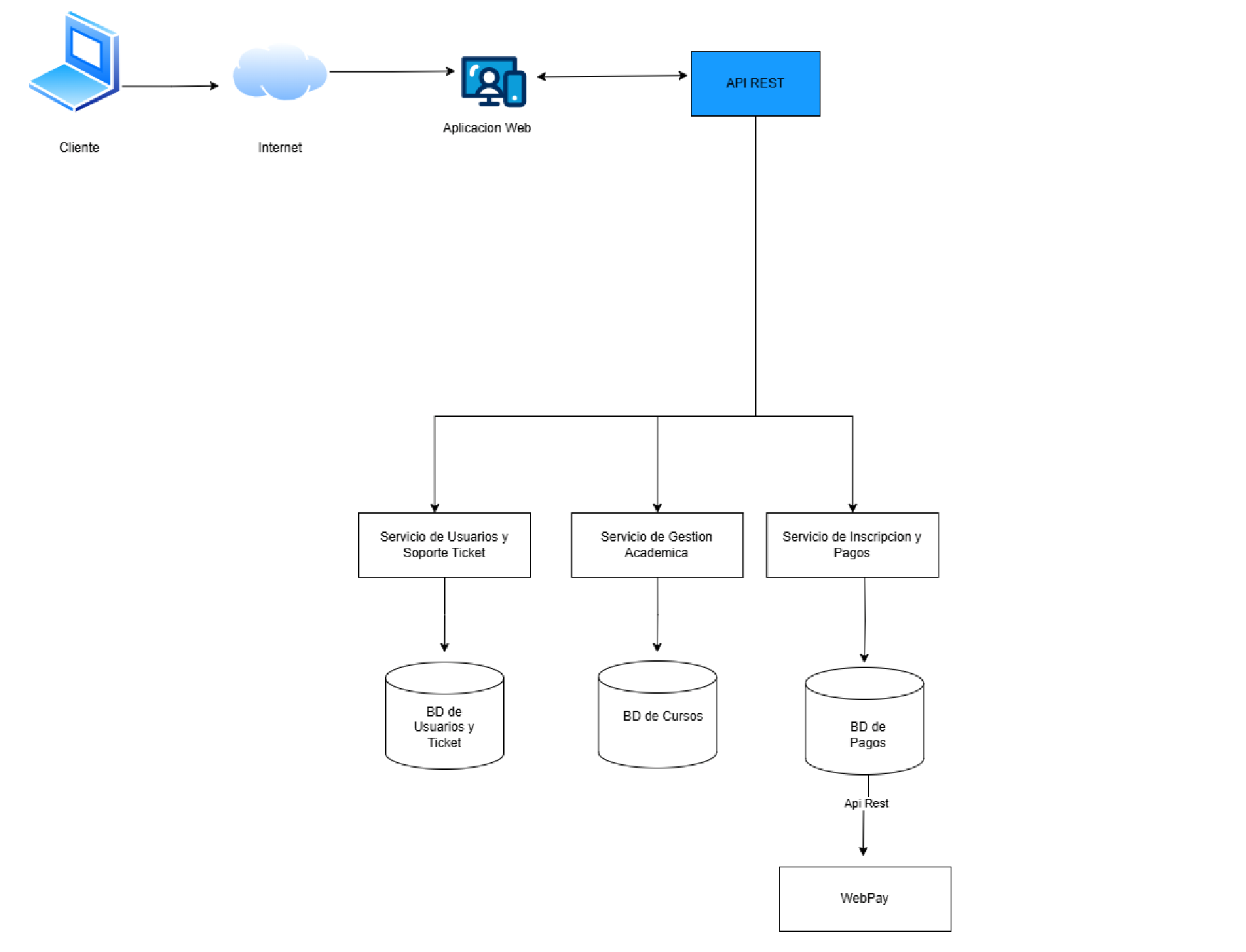
**CrudRepository**, encargadas de acceder a la base de datos sin necesidad de escribir SQL manualmente.

* **service:** Define la lógica de negocio de cada módulo. Se divide en interfaces **(IEntidadService)** y clases de implementación **(EntidadServiceImpl).**
* **application.properties**: Archivo donde se configuran las propiedades del proyecto como el puerto del servidor, la conexión a la base de datos, el nombre del microservicio, etc.

**Herramientas utilizadas para gestionar esta estructura:**

* Spring Initializr: Generador de proyectos Spring Boot desde cero.
* Maven: Herramienta de gestión de dependencias y compilación del proyecto.

**2.2 Diagrama general de arquitectura del sistema**



# ● 2.3 Organización de capas y componentes

El proyecto EduTech está diseñado bajo una arquitectura por capas, lo que permite separar responsabilidades, facilitar el mantenimiento y garantizar una estructura lógica y ordenada del código. Cada microservicio (Gestión Académica, Gestión de Usuario y Gestión de Pago) mantiene esta misma organización, permitiendo trabajar de manera independiente sobre cada funcionalidad.

A continuación, se describe cada una de las capas:

**Capa de Configuración (config)**

Incluye clases encargadas de la configuración del sistema, tales como la inicialización de componentes, gestión de beans y otras dependencias esenciales para el correcto funcionamiento de la aplicación.

**Capa de Controlador (Controller)**

Define los endpoints REST que permiten interactuar con el sistema desde el exterior. Utiliza anotaciones como @RestController, @GetMapping, @PostMapping, etc., y recibe o entrega datos en formato JSON.

Ejemplo: CursoController, UsuarioController, PagoController.

**Capa de Entidad (Model/Entity)**

Contiene las clases que representan las entidades del dominio, las cuales están mapeadas a las tablas de la base de datos mediante anotaciones de JPA (@Entity, @Table, @Id, etc.). Estas clases definen los atributos y relaciones necesarias.

Ejemplo: Curso, Evaluación, Usuario, Pago.

**Capa de Repositorio (Repository)**

Está compuesta por interfaces que extienden JpaRepository o CrudRepository. Estas interfaces permiten realizar operaciones CRUD (crear, leer, actualizar, eliminar) sin necesidad de escribir SQL manualmente.

Ejemplo: CursoRepository, UsuarioRepository, PagoRepository.

**Capa de Servicio (Service)**

Contiene la lógica de negocio. Esta capa actúa como intermediaria entre los controladores y los repositorios. Se organiza normalmente en una interfaz (NombreService) y su clase de implementación (NombreServiceImpl).

Ejemplo: Validar el puntaje de una respuesta, registrar un pago, asociar un usuario a un rol.

**Capa de Configuración (resources/application.properties)**

* Nombre del microservicio
* Puerto del servidor
* Datos de conexión a la base de datos
* Configuraciones específicas de Spring Boot

Esta organización modular y bien estructurada permite que cada capa cumpla con una función específica, facilitando la escalabilidad, pruebas unitarias y reutilización de componentes dentro de la plataforma EduTech.

# ● 2.4 Configuración de dependencias con Maven

Para la gestión de dependencias y la automatización del proceso de compilación de los microservicios del sistema EduTech, se utilizó Apache Maven, una herramienta ampliamente adoptada en el ecosistema Java. Su archivo principal de configuración es el pom.xml (Project Object Model), el cual define las dependencias, plugins, propiedades del proyecto y configuraciones específicas.

Cada microservicio (Gestión Académica, Gestión de Usuario y Gestión de Pago) cuenta con su propio **pom.xml**, permitiendo así que las dependencias se mantengan organizadas y específicas por funcionalidad.

Estructura básica del pom.xml:

* parent: Hace referencia al padre **spring-boot-starter-parent,** lo cual simplifica la gestión de versiones y configuración de plugins.
* groupId / artifactId / version: Identificadores únicos del microservicio dentro del proyecto.
* java.version: Define la versión de Java utilizada (en este caso, Java 17).
* Dependencias: Lista de bibliotecas requeridas por cada microservicio.
* build/plugins: Configurar los plugins de Maven necesarios para compilar, empaquetar y ejecutar el proyecto.

Principales dependencias utilizadas:

**Dependencia Descripción**

**spring-boot-starter-web** Permite crear aplicaciones RESTful usando Spring MVC.

**spring-boot-starter-data-jpa** Soporte para persistencia con JPA e Hibernate.

**spring-boot-starter-validation** Validación de datos de entrada mediante anotaciones.

**mysql-connector-j** Driver JDBC para conexión a MySQL.

|  |  |
| --- | --- |
| **lombok** | Simplifica el código mediante anotaciones para generar getters/setters, constructores, etc. |
| **spring-boot-starter-test** | Incluye herramientas de prueba unitaria e integración. |
| **spring-security-core** | (Solo en Gestión de Usuario) Proporciona componentes básicos de seguridad. |
| **spring-boot-starter-webflux** | (Solo en Gestión de Pago) Permite usar programación reactiva. |
| **transbank-sdk-java** | (Solo en Gestión de Pago) SDK oficial para integración con Webpay. |

Plugins utilizados:

* maven-compiler-plugin: Configura la compilación del código y permite procesar anotaciones (como las de Lombok).
* spring-boot-maven-plugin: Facilita el empaquetado y la ejecución del proyecto Spring Boot directamente desde Maven.

**Repositorios externos:**

En el microservicio de Gestión de Pago se incluye el repositorio **jitpack.io** para poder importar el SDK de Transbank, ya que no se encuentra disponible en el repositorio oficial de Maven Central.

# 3. Control de Versiones y Trabajo Colaborativo

Para el desarrollo de los microservicios de la plataforma EduTech, se utilizó Git como sistema de control de versiones distribuido, permitiendo un seguimiento detallado de los cambios realizados en el código fuente a lo largo del tiempo. Esto facilitó tanto la organización interna como el trabajo colaborativo entre integrantes del equipo.

El proyecto fue alojado en la plataforma GitHub, lo que permitió centralizar el repositorio, gestionar ramas, colaborar de manera remota y mantener un historial completo de los avances.

Beneficios del control de versiones aplicado:

* Rastreo de cambios: Cada modificación quedó registrada con un mensaje descriptivo, lo que permite retroceder a versiones anteriores en caso de errores o conflictos.
* Trabajo en ramas: Se crearon ramas independientes por microservicio o funcionalidad (por ejemplo, **gestion-academica, gestion-usuario, pago**), permitiendo a cada integrante trabajar de forma aislada sin afectar el código principal.
* Fusión de código (merge): Una vez finalizada una funcionalidad, esta se integraba a la rama principal (**main o develop**) mediante pull requests, lo cual permitía revisión y validación del código antes de su incorporación.
* Resolución de conflictos: Gracias a las herramientas de Git y GitHub, fue posible identificar y resolver conflictos de forma ordenada durante el proceso de integración.

**Herramientas utilizadas:**

* Visual Studio Code + Git integrado: Para escribir y subir código fácilmente desde el entorno de desarrollo.
* GitHub Desktop / Línea de comandos Git Bash: Para sincronizar, clonar repositorios y gestionar commits y push/pull de forma eficiente.
* GitHub: Plataforma central para alojar el repositorio remoto, realizar seguimiento de tareas y organizar el trabajo por ramas.

# 3.1 Configuración inicial del repositorio Git

**Configuración realizada mediante Git Bash:**

En esta etapa se configuró el repositorio del proyecto utilizando comandos ejecutados en Git Bash. Se inicializó el repositorio, se agregaron los archivos necesarios, y se realizaron los primeros commits para asegurar el control de versiones desde el inicio del desarrollo:

* cd /ruta/a/tu/proyecto
* git init ● git add .
* git commit -m "first commit"
* git branch -M main
* git remote add origin https://github.com/CatPino/DesarrolloFullstack\_Microservicios.git ● git push -u origin main

## 3.2 Organización de ramas y flujos de trabajo

Durante el desarrollo del proyecto EduTech, se adoptó una estrategia de trabajo basada en ramas para organizar y distribuir el desarrollo de los distintos microservicios, asegurando un flujo de trabajo ordenado y colaborativo. Esta organización permitió que cada integrante pudiera trabajar de forma paralela sin generar conflictos en el código principal.

**Estructura de ramas utilizada:**

* main: Rama principal del proyecto. Contiene el código más estable y funcional, listo para despliegue o pruebas integradas.

* Ramas por microservicio:

Cada microservicio se desarrolló de forma independiente en su propia rama, permitiendo mantener una separación clara entre funcionalidades y evitar interferencias durante el trabajo en

**Flujo de trabajo aplicado:**

1. Creación de rama: Cada integrante del equipo creaba una rama específica para la funcionalidad o módulo en el que iba a trabajar.

1. Desarrollo local: Se realizaban los cambios, pruebas y commits de forma local, utilizando mensajes claros y descriptivos.

1. Sincronización con el repositorio remoto: Se usaban comandos git push para subir los cambios a GitHub.

1. Revisión y Merge: Se revisaban los cambios, se resolvían posibles conflictos y, si todo estaba correcto, se aprobaba y fusionaba el código a la rama principal.

1. Eliminación de ramas temporales: Luego de completar y fusionar, se eliminaban las ramas que ya no eran necesarias para mantener limpio el repositorio.

**Beneficios de este enfoque:**

* Favorece la colaboración simultánea.

* Permite identificar errores rápidamente antes de integrarlos al proyecto general.

* Facilita pruebas por separado de cada microservicio.

* Mejora la trazabilidad del desarrollo.

## 3.3 Estrategias de commits y merges

Para asegurar un desarrollo claro, organizado y colaborativo en el proyecto EduTech, se definieron estrategias específicas para realizar commits (registro de cambios) y merges (fusión de ramas), siguiendo buenas prácticas de control de versiones con Git y GitHub.

Estrategias de Commits

Los commits fueron utilizados para registrar avances significativos en el desarrollo del código, manteniendo un historial limpio y comprensible. Se aplicaron las siguientes buenas prácticas:

* Commits frecuentes pero relevantes: Se realizaron commits cada vez que se completaba una funcionalidad específica o una corrección importante, evitando cambios masivos en un solo paso.

* Mensajes claros y descriptivos: Se utilizó un formato consistente en los mensajes.

## 3.4 Coordinación y trabajo en equipo

El desarrollo del sistema EduTech fue realizado en equipo, lo que implicó una constante necesidad de comunicación, coordinación y colaboración. Si bien trabajar en conjunto permitió dividir responsabilidades y avanzar en paralelo en los distintos microservicios, también presentó algunos desafíos, especialmente en las etapas iniciales del proyecto.

**Dificultades iniciales:**

Al comienzo, costó organizarnos adecuadamente como equipo, principalmente por falta de experiencia en el trabajo colaborativo con control de versiones y la gestión simultánea de distintas ramas. Hubo momentos en que se generaron conflictos de código, confusiones en los roles y retrasos por falta de planificación.

Además, la coordinación de horarios y la asignación clara de tareas no siempre fue buena en las primeras semanas, lo que llevó a una menor productividad y a tener que rehacer ciertos avances por falta de comunicación.

**Mejora progresiva:**

A medida que avanzamos en el proyecto, fuimos adoptando estrategias más claras para organizarnos:

* Utilizamos ramas por funcionalidad, lo que permitió que cada integrante trabajara en su parte sin interferir con los demás.
* Aprendimos a utilizar herramientas como Git, GitHub y Postman de forma colaborativa, lo que mejoró nuestra eficiencia.
* Fuimos desarrollando una mayor comunicación y responsabilidad compartida, lo que fortaleció el trabajo en equipo y permitió cumplir con los objetivos propuestos.

**Lecciones aprendidas:**

Este proyecto nos permitió comprender la importancia de planificar bien el trabajo colaborativo, documentar los avances, compartir conocimientos entre compañeros y apoyarse mutuamente para resolver errores. A pesar de las dificultades iniciales, logramos consolidarnos como equipo y completar con éxito el desarrollo de los tres microservicios de la plataforma.

**4. Base de Datos y Modelo de Datos:**

Cada microservicio de la plataforma EduTech gestiona su propia base de datos de forma independiente, siguiendo el principio de desacoplamiento de la arquitectura basada en microservicios. A continuación, se detallan las entidades y relaciones clave de cada microservicio.

### 4.1 Microservicio de Gestión Académica

Base de datos: **gestión\_académicabd**

Este microservicio se encarga de administrar los cursos y sus evaluaciones, incluyendo las preguntas y respuestas asociadas.

**Entidades principales:**

* **Curso:** Título, descripción, duración, categoría, estado, precio.
* **Evaluacion:** Se vincula a un curso. Contiene título, tipo y porcentaje.
* Pregunta: Se asocia a una evaluación. Representa el enunciado de una pregunta.
* **Respuesta:** Se asocia a una pregunta. Contiene el contenido de la opción, si es correcta o no, y su puntaje.

**Relaciones:**

* Un curso tiene muchas evaluaciones.
* Una evaluación tiene muchas preguntas. ● Una pregunta tiene muchas respuestas.

### 4.2 Microservicio de Gestión de Usuario y Soporte

Base de datos: **usuariosdb**

Este microservicio gestiona los datos personales y de acceso de los usuarios, sus roles dentro de la plataforma y la atención a través de tickets de soporte.

**Entidades principales:**

* **Usuario:** Nombre, correo, contraseña, tipo, otros datos personales.
* **Rol:** Define el tipo de usuario (estudiante, docente, administrador).
* **Ticket:** Solicitud de soporte generada por un usuario. Incluye asunto, estado y fecha de creación.
* **MensajeTicket:** Representa un mensaje dentro de un ticket. Contiene el contenido, fecha, autor y el ticket al que pertenece.

**Relaciones:**

* Un usuario tiene un rol.
* Un rol puede estar asignado a varios usuarios.
* Un usuario puede generar muchos tickets.
* Un ticket puede contener muchos mensajes (Mensaje Ticket).
* Cada mensaje está relacionado con un usuario (autor) y con un ticket.

### 4.3 Microservicio de Gestión de Pago

Base de datos: **pagoBD**

Microservicio encargado de realizar pagos a través de Webpay Transbank. Además, este microservicio gestiona la inscripción a los cursos adquiridos, la aplicación de cupones, el registro de pagos y la creación de inscripciones.

**Entidades principales:**

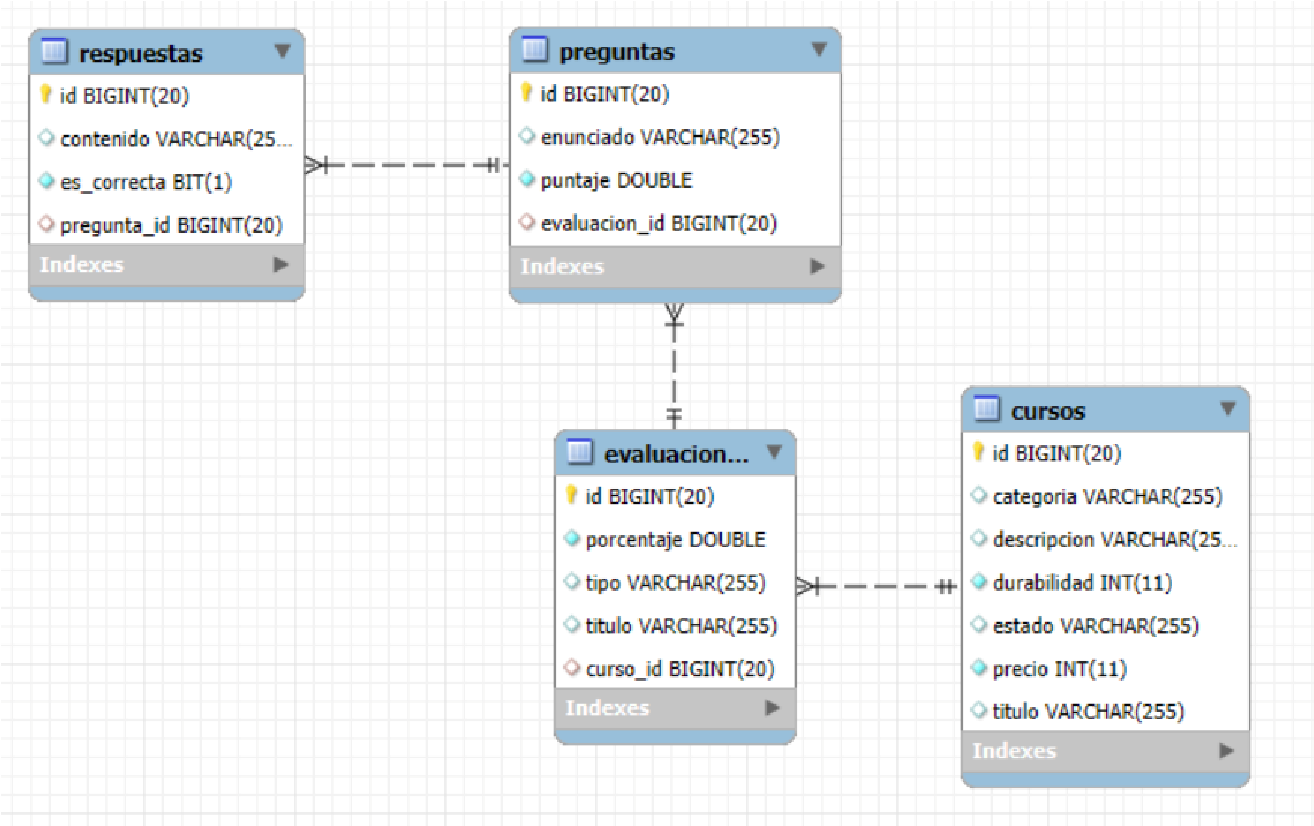
* **Inscripción:** Representa el vínculo entre un usuario y un curso.
* **Cupón:** Representa un beneficio aplicado a una compra o inscripción, que permite otorgar descuentos o promociones a los usuarios.
* **Pago:** Registro de una transacción realizada, contienen datos de los pagos.

**Relaciones:**

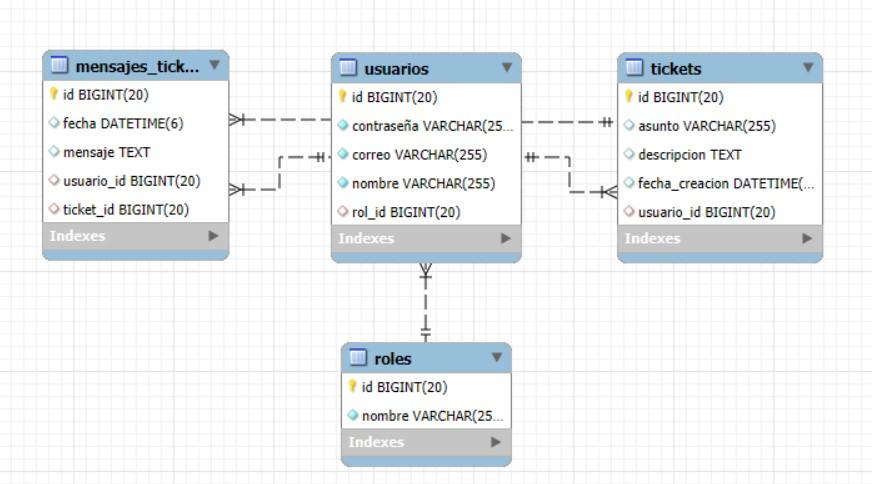
* Un usuario puede tener varias inscripciones.
* Una inscripción puede tener varios pagos (cuotas). ● Muchos pagos pueden usar un mismo cupón

# 4.1 Modelo de datos conceptual: entidades y relaciones

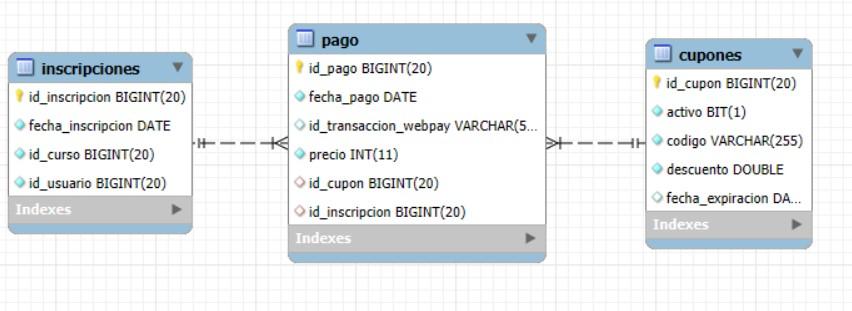
**Diagrama de microservicio de gestión académica:**



**Diagrama de microservicio gestión de usuarios:**



**Diagrama de microservicio de pagos e inscripciones :**



## 4.2 Modelo físico de base de datos (adjunto y explicado)

### Modelo Físico – Microservicio de Gestión Académica

El modelo físico del microservicio de Gestión Académica está compuesto por cuatro tablas principales: **cursos, evaluaciones, preguntas y respuestas**. Estas tablas se relacionan entre sí mediante claves foráneas, respetando la integridad referencial de la base de datos.

**cursos:**

* id (clave primaria)
* categoria, descripcion, durabilidad, estado, precio, titulo ● Se relaciona con la tabla **evaluaciones**. **evaluaciones:**
* id (clave primaria)
* porcentaje, tipo, titulo
* curso\_id (clave foránea) → referencia a **cursos.id** ● Se relaciona con la tabla **preguntas. preguntas:**
* id (clave primaria)
* enunciado, puntaje
* evaluacion\_id (clave foránea) → referencia a **evaluaciones.id** ● Se relaciona con la tabla **respuestas.**

**respuestas:**

* id (clave primaria)
* contenido, es\_correcta
* pregunta\_id (clave foránea) → referencia a **preguntas.id**

Este modelo permite crear cursos con evaluaciones estructuradas y automatizadas, manteniendo los datos organizados, normalizados y fácilmente escalables.

**Modelo Físico – Microservicio de Gestión de Usuario y Soporte:**

Este microservicio utiliza cinco tablas principales: usuarios, roles, tickets y mensajes\_ticket, organizadas para manejar la autenticación de usuarios, sus roles, solicitudes de soporte y los mensajes asociados a cada ticket.

**usuarios:**

* id (clave primaria)
* nombre, correo, contraseña
* rol\_id (clave foránea) → referencia a **roles.id**
* Se relaciona con **roles, tickets** y **mensajes\_ticket.**

**Rol:**

* id (clave primaria)
* nombre
* Se relaciona con **usuarios.** **tickets:**
* id (clave primaria)
* asunto, descripcion, fecha\_creacion
* usuario\_id (clave foránea) → referencia a **usuarios.id** ● Se relaciona con **mensajes\_ticket.** **mensajes\_ticket:**
* id (clave primaria)
* fecha, mensaje
* usuario\_id (clave foránea) → referencia a **usuarios.id** ● ticket\_id (clave foránea) → referencia a **tickets.id**

Este modelo permite manejar tanto el sistema de autenticación de usuarios con roles, como la gestión de soporte mediante tickets y mensajes internos, manteniendo relaciones claras y una estructura normalizada.

**Modelo Físico – Microservicio de gestión de pagos e inscripciones:**

El modelo físico del sistema de inscripciones y pagos está compuesto por tres tablas principales: inscripciones, pago y cupones. Estas tablas se relacionan mediante claves foráneas, garantizando la integridad referencial de la base de datos.

**inscripciones:**

* id\_inscripcion (PK)
* fecha\_inscripcion
* id\_curso (FK → cursos.id) (tabla "cursos" no mostrada)
* id\_usuario (FK → usuarios.id) (tabla "usuarios" no mostrada)

**pago:**

* id\_pago (PK)
* fecha\_pago
* id\_transaction\_webpay
* precio
* id\_cupon (FK → cupones.id, opcional)
* id\_inscripcion (FK → inscripciones.id\_inscripcion) **cupones:**
* id\_cupon (PK)
* activo
* codigo
* descuento
* fecha\_expiracion

Este modelo permite gestionar de manera integral el proceso de inscripciones a cursos y su flujo de pagos asociados. Registra cada inscripción vinculada a un usuario y curso (aunque estas tablas no se muestran en el diagrama), y controla los pagos realizados, incluyendo transacciones con Webpay y la aplicación opcional de cupones de descuento.

**7. Entorno de Pruebas Local**

**7.1 Estrategia de pruebas funcionales con Postman**

El entorno de pruebas locales con Postman es una fase crítica en el desarrollo del sistema, diseñada para garantizar que todas las funcionalidades operen correctamente antes de su implementación en producción.

**1. Objetivo Principal:**

* Las solicitudes HTTP (GET, POST, PUT, DELETE) reciban las respuestas esperadas.
* Los datos enviados y recibidos cumplan con la estructura definida (ej.: JSON válido). Los códigos de estado HTTP (200, 201, 400, 404, 500, etc.) sean los adecuados para cada caso.
* Las reglas de negocio se aplican correctamente (ej.: validación de cupones, permisos de usuario).
* La integridad de los datos se mantenga (ej.: relaciones entre entidades, claves foráneas).

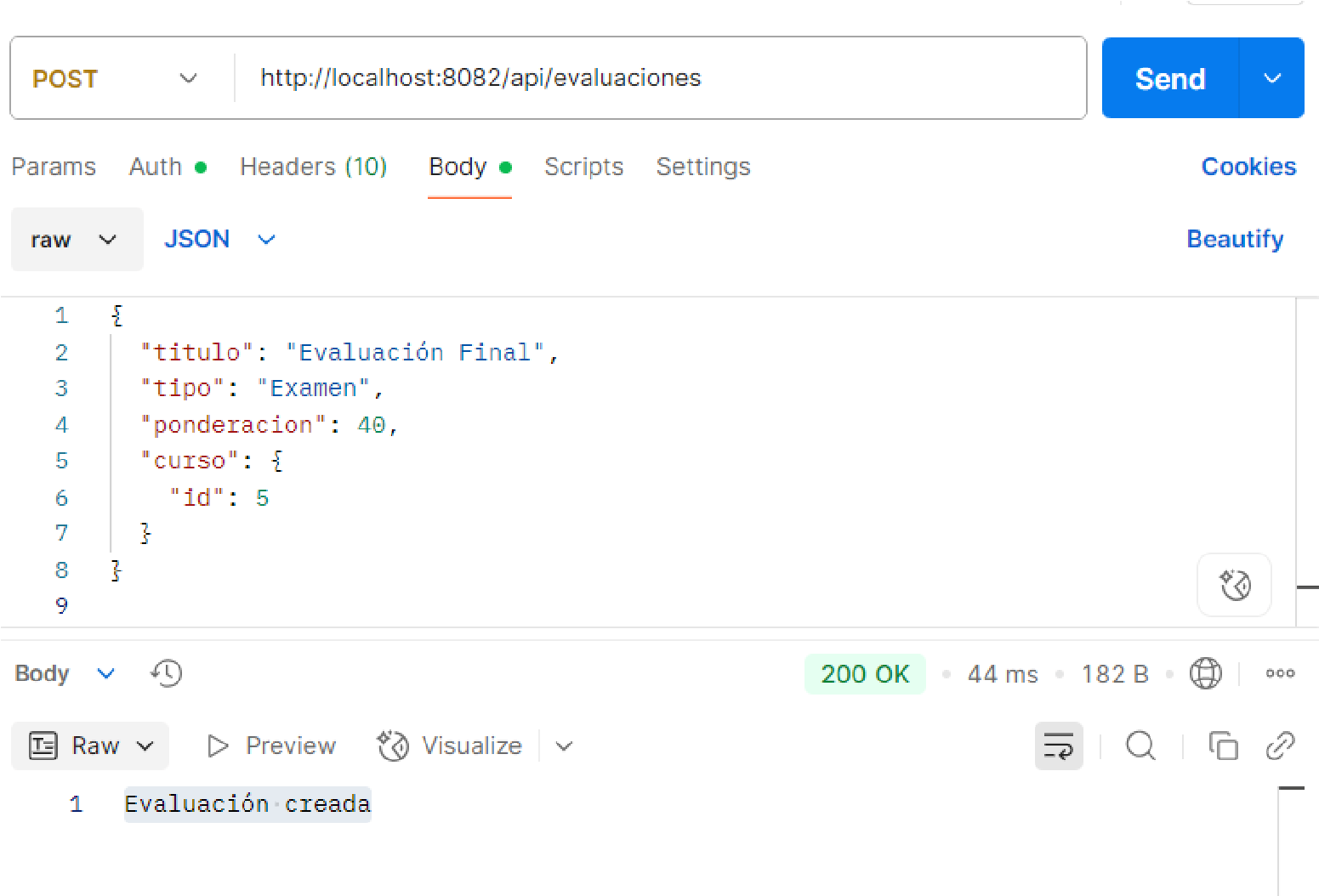
* **7.2 Evidencia del funcionamiento de los servicios:**

**Microservicio de Gestión académica:**

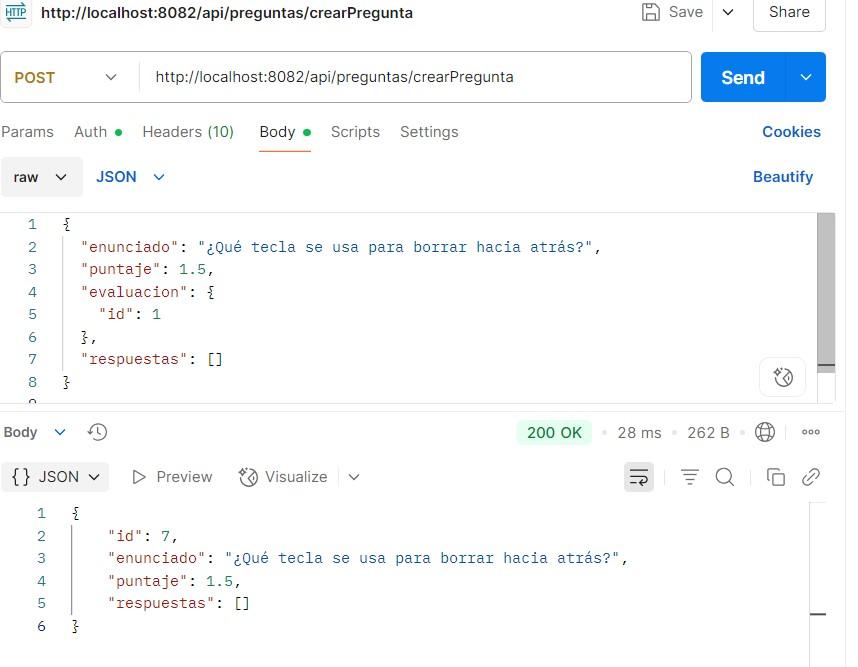
**Endpoint para crear curso:**



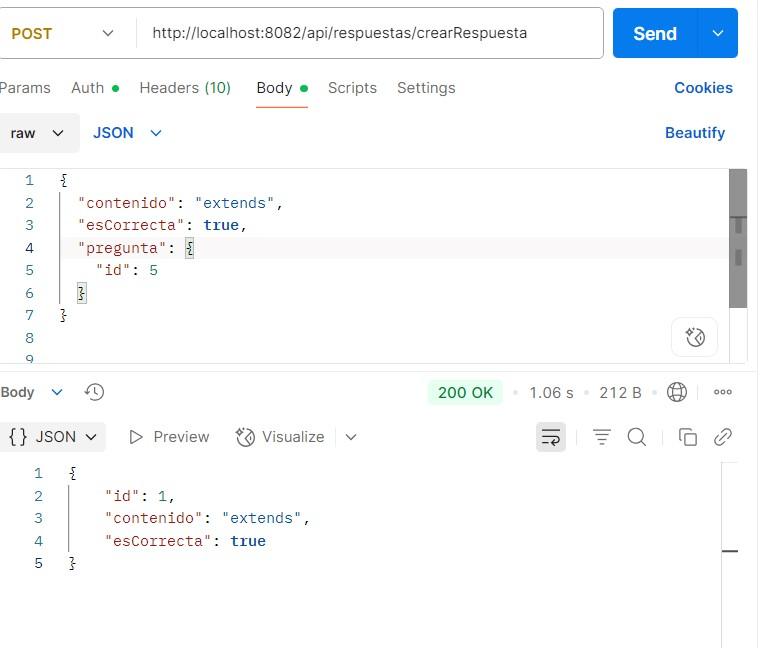
**Endpoint para crear evaluación:**



**Endpoint para crear pregunta:**

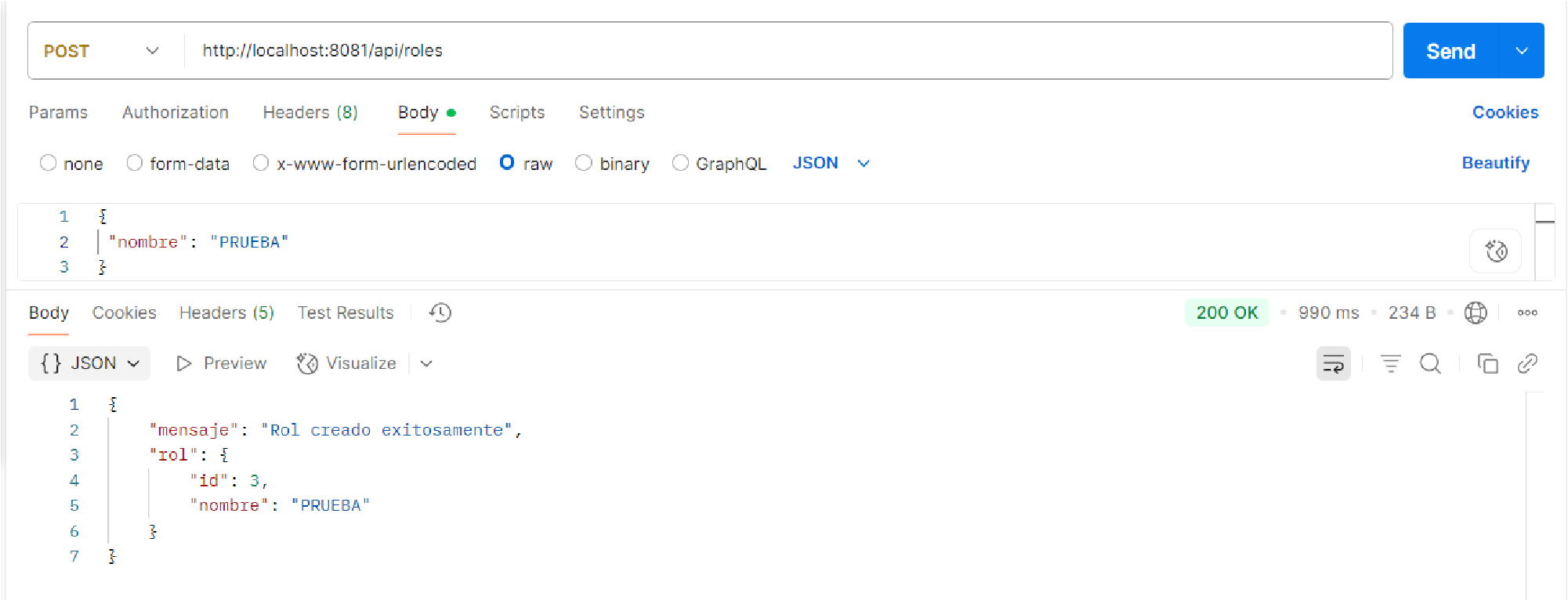


**Endpoint para crear respuestas:**

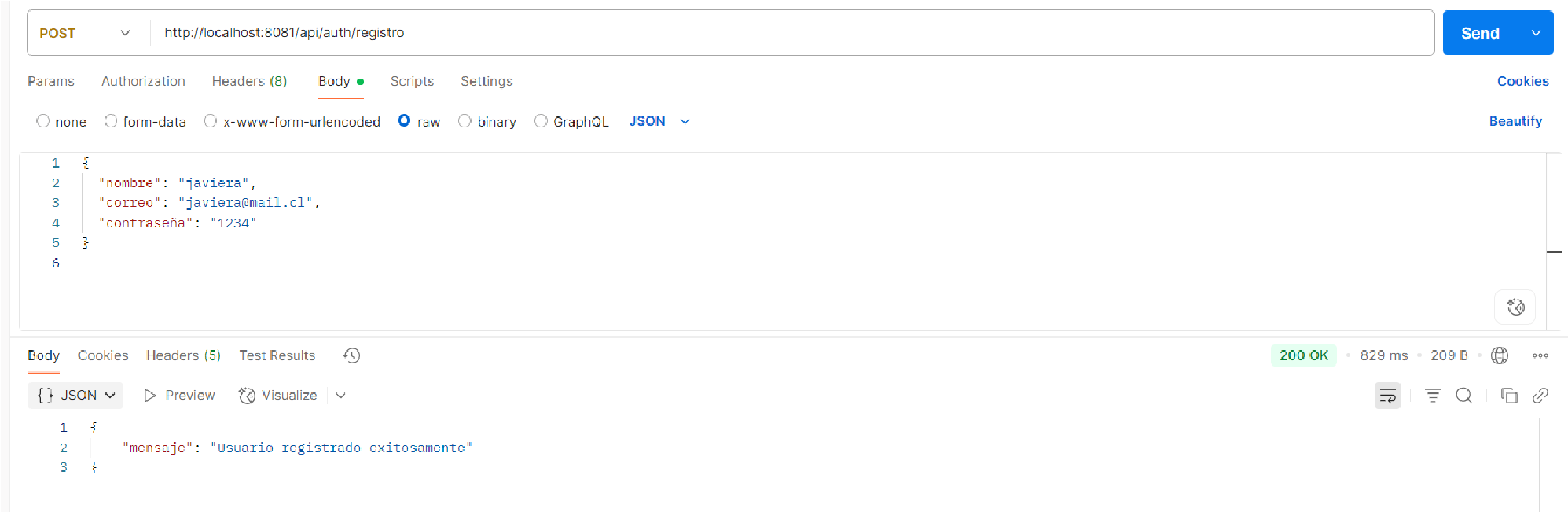


**Microservicio de gestión de usuarios y roles:**

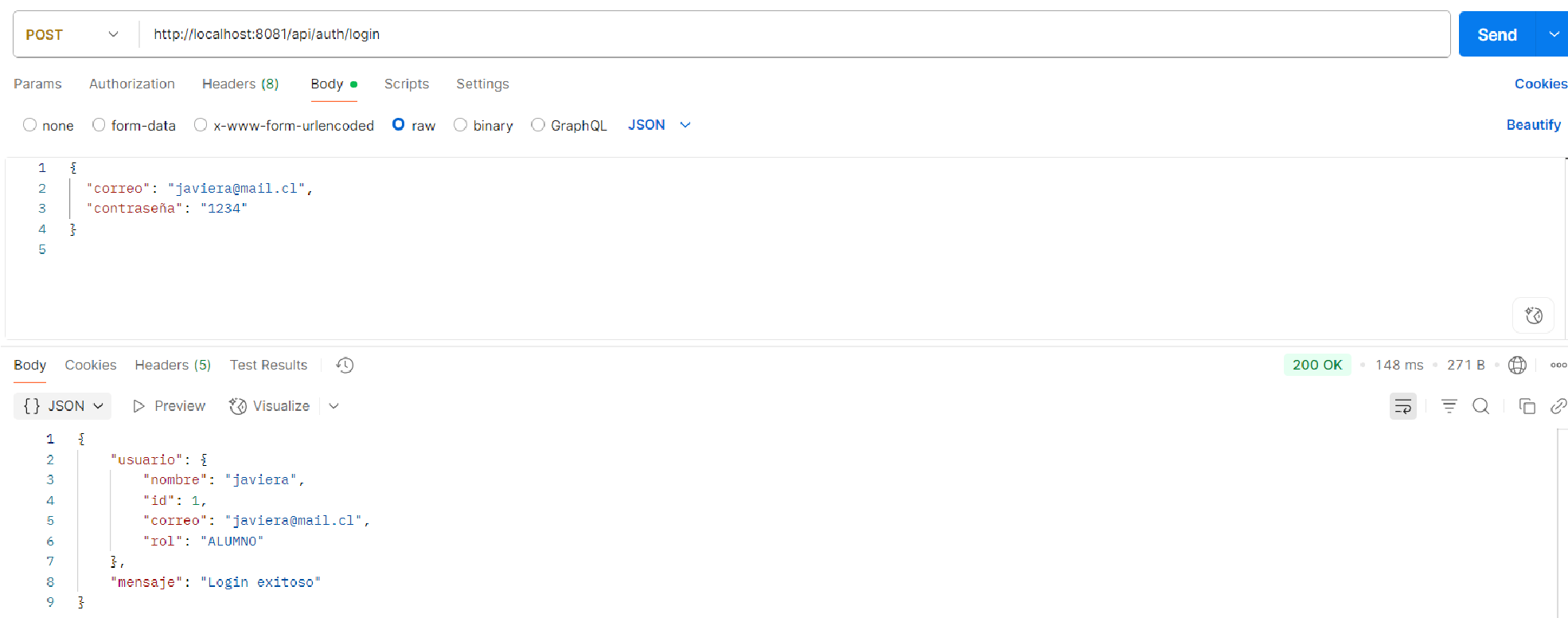
**Endpoint de creación de roles:**



**Endpoint de creación de usuarios:**

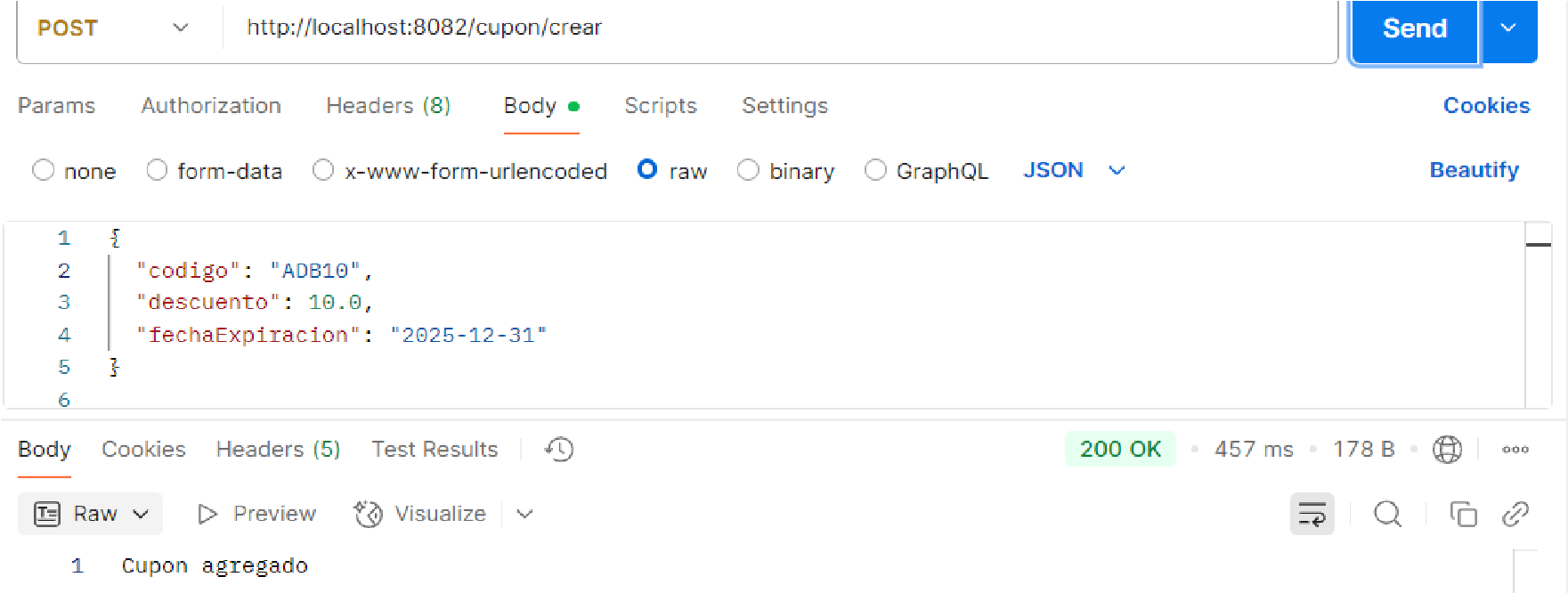


**Endpoint de Login:**

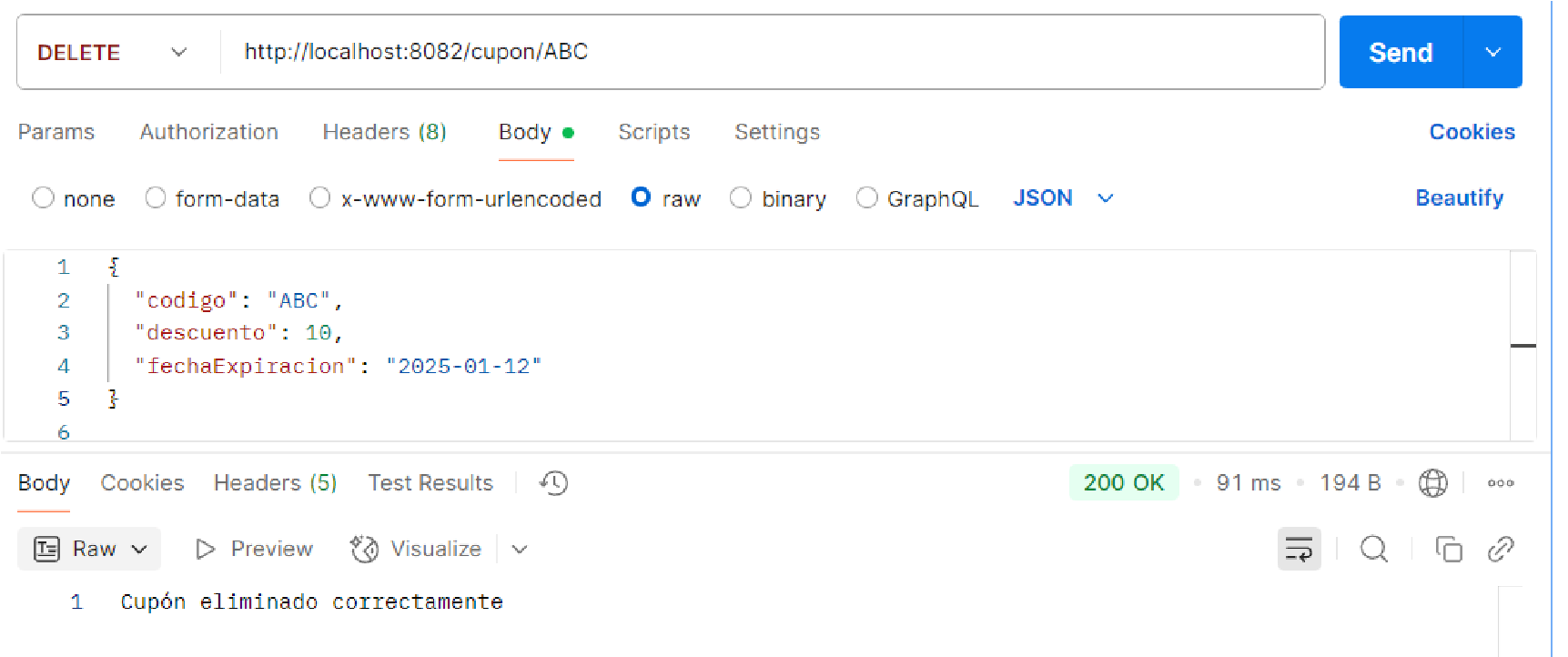


**Microservicio de gestión pagos e inscripción:**

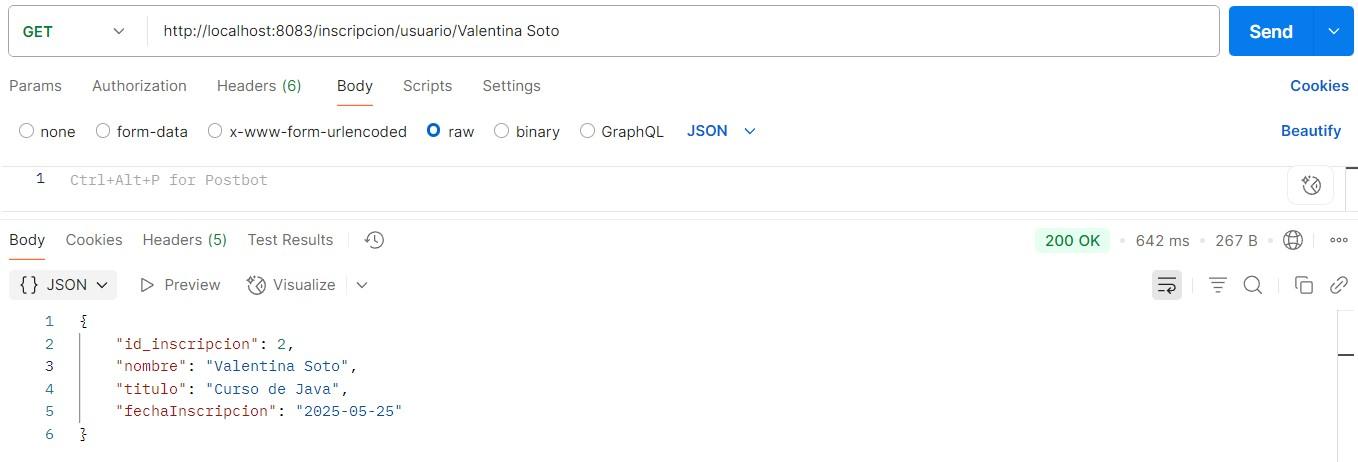
**Endpoint de creación de cupón:**



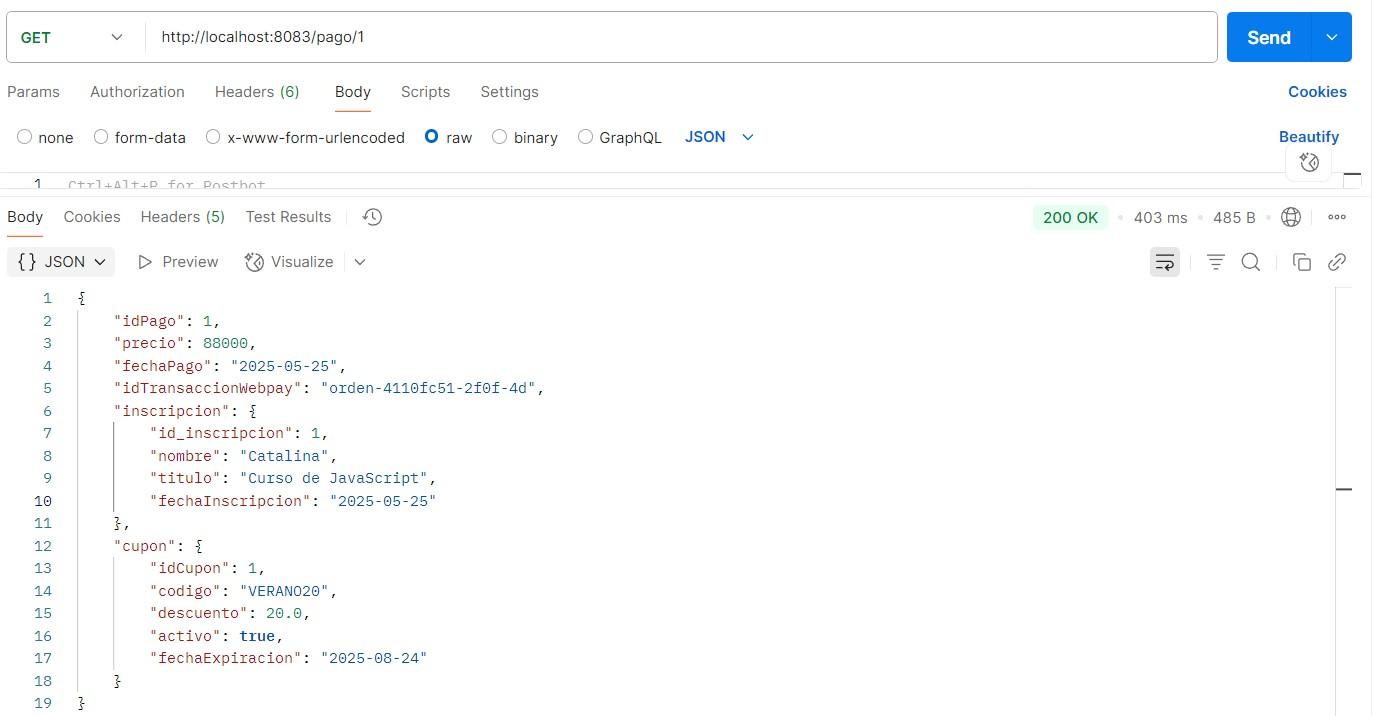
**Endpoint para eliminar cupón:**



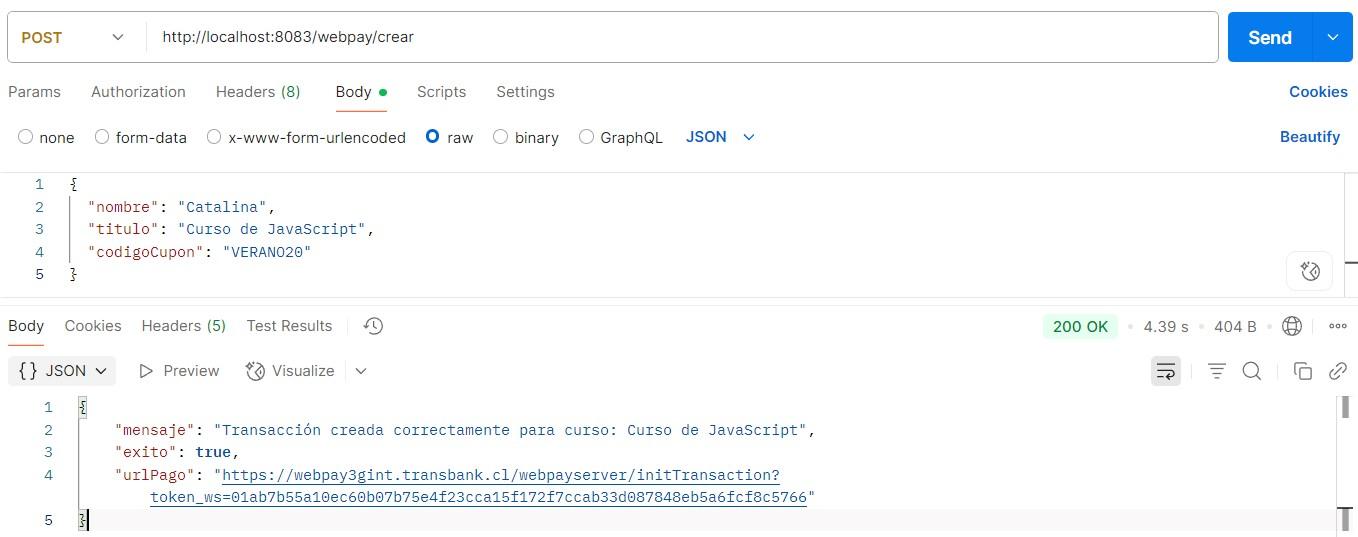
**Endpoint para obtener información por nombre de usuario:**



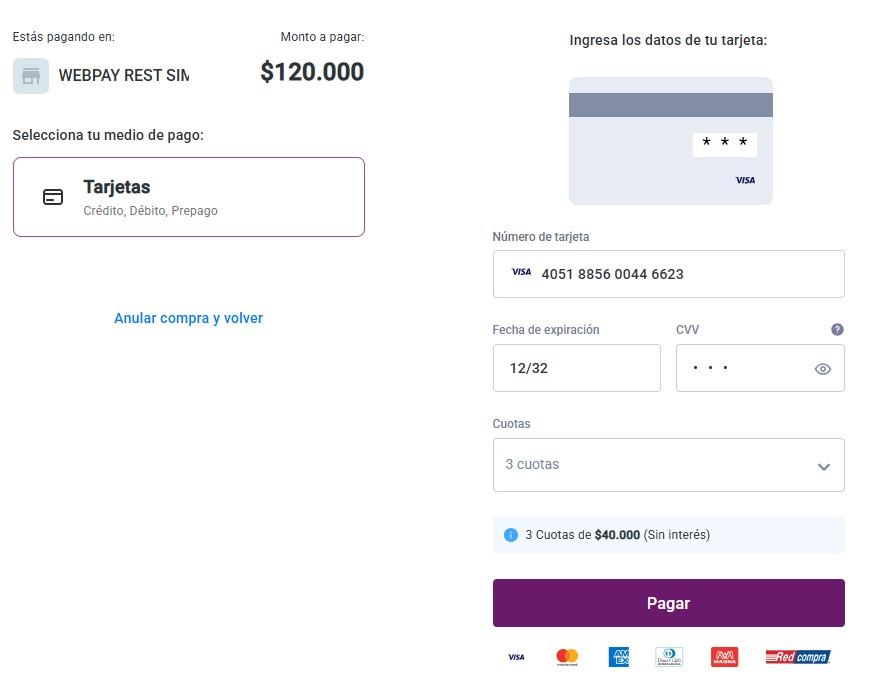
**Endpoint para obtener información del pago a través del id pago:**



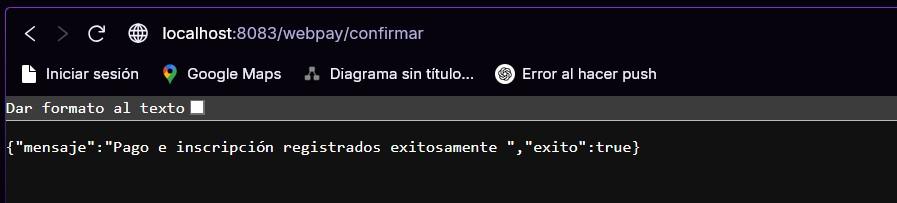
**Endpoint para realizar transacción a través de transbank:**



**Link de transbank:**



**Finalización de pago:**



**8. Conclusiones**

**● 8.1 Lecciones aprendidas**

Durante el desarrollo del proyecto EduTech aprendimos muchísimo, no solo sobre programación y microservicios, sino también sobre cómo trabajar mejor en equipo. Al principio no fue fácil organizarnos: nos costó repartir bien las tareas, coordinar los horarios y usar herramientas como Git sin pisarnos el trabajo unos a otros. Pero con el tiempo fuimos mejorando, aprendimos a comunicarnos más, a usar ramas y commits de forma ordenada y a confiar en el aporte de cada integrante.

En la parte técnica, trabajar con Spring Boot, MySQL y Postman nos permitió entender cómo construir APIs REST reales y cómo hacer que cada microservicio funcione por separado. También vimos lo importante que es probar cada cosa que hacemos, ya que muchas veces algo que parecía estar bien, fallaba cuando se probaba con datos reales.

Otra cosa que nos marcó fue darnos cuenta de que equivocarse es parte del proceso. A veces algo no funcionaba, nos frustramos, pero luego encontrábamos la solución juntas o aprendíamos algo nuevo. Al final, lo más valioso fue ver cómo todo el esfuerzo se convirtió en un sistema completo, funcional y hecho por nosotros mismos.

Este proyecto no solo nos ayudó a crecer como futuros desarrolladores, sino también como personas que saben trabajar en equipo, resolver problemas y no rendirse fácilmente.

**● 8.2 Dificultades enfrentadas y cómo se resolvieron**

Como en todo proyecto real, durante el desarrollo de EduTech también tuvimos varios momentos complicados que nos pusieron a prueba. Algunas de las principales dificultades que enfrentamos fueron:

1. **Organización del equipo**

**Dificultades iniciales:**

Al inicio del proyecto enfrentamos problemas de coordinación. La falta de planificación clara y de una estrategia de trabajo definida provocó una pérdida significativa de tiempo y esfuerzo.

**¿Cómo lo resolvimos?**

Implementamos el uso de ramas en Git para trabajar de forma paralela sin conflictos, distribuimos las tareas de manera más organizada y mejoramos la comunicación dentro del equipo. Estos cambios fortalecieron la colaboración y permitieron un avance más eficiente en el desarrollo.

1. **Manejo de Git y conflictos de código**

**Dificultades iniciales:**

Durante las primeras etapas del proyecto, tuvimos dificultades para utilizar correctamente Git y GitHub. Nos costó entender el flujo de trabajo con ramas, así como aprender y aplicar los comandos necesarios para subir cambios al repositorio remoto. Además, la falta de experiencia generó conflictos de código al momento de fusionar ramas, lo que complicó el desarrollo colaborativo.

**¿Cómo lo resolvimos?**

Nos capacitamos en el uso básico de Git, aprendiendo a utilizar comandos. También adoptamos buenas prácticas como la creación de ramas por funcionalidad y la revisión del código antes de integrarlo. A través de una mejor comunicación y la división clara de tareas, logramos minimizar los conflictos y trabajar de manera más ordenada y eficiente.

1. **Errores en las relaciones entre tablas**

**Dificultades iniciales:**

Algunos endpoints fallaban porque no estaban bien definidas las relaciones entre las entidades en la base de datos (por ejemplo, claves foráneas mal mapeadas).

**¿Cómo lo resolvimos?**

Revisamos la estructura de las entidades y sus anotaciones en Java con JPA, y validamos todo con Postman y la base de datos en MySQL.

1. **Dificultad para dividir los microservicios**

**Dificultades iniciales:**

Al principio era difícil pensar cómo separar correctamente las funciones en microservicios sin que se cruzaran.

**¿Cómo lo resolvimos?**

Entendimos que cada microservicio debía tener su propia base de datos y lógica independiente, y nos enfocamos en mantenerlos lo más desacoplados posible.

1. **Validaciones y pruebas**

**Dificultades iniciales:**

A veces dábamos por hecho que algo funcionaba sin probarlo completamente, y después nos encontrábamos con errores inesperados.

**¿Cómo lo resolvimos?**

Usamos Postman de forma más sistemática, probando cada endpoint con distintos datos y escenarios.

### ● 8.3 Valoración del trabajo colaborativo y herramientas utilizadas

El trabajo en equipo fue uno de los mayores desafíos del proyecto **EduTech**, especialmente al inicio. Hubo dificultades para coordinarnos y repartir bien las tareas, pero con el tiempo aprendimos a organizarnos mejor, a comunicarnos más y

a confiar en el aporte de cada integrante.

Gracias a las herramientas que usamos, pudimos avanzar de forma más ordenada:

* **Git y GitHub**: Para controlar versiones, trabajar por ramas y fusionar el trabajo de todos.
* **Visual Studio Code**: Editor de código principal, también útil para resolver conflictos.
* **Postman**: Ideal para probar los endpoints de los microservicios y verificar respuestas.
* **XAMPP y MySQL**: Para crear y gestionar nuestras bases de datos localmente.
* **Spring Boot**: Base del proyecto para crear APIs REST con estructura clara.
* **Trello**: Nos ayudó a organizarnos, dividir tareas y llevar un control visual del avance del equipo.

A pesar de las dificultades iniciales, el trabajo colaborativo nos permitió crecer como equipo y lograr un sistema funcional, integrando lo aprendido en programación, bases de datos y gestión de proyectos.

**9. Anexos:**

**Creación de base de datos:**

**Microservicio de gestión académica:**



**Microservicio de gestión de roles y usuarios:**



**Microservicio de gestión de pagos e inscripciones:**



**enlace a repositorio de github:**

https://github.com/Paula-rojaso/IngSoft.git