**Informe Desarrollo de Microservicios EduTech**

**Martín Cofré, Dantte García, Joaquín Molina**

**Prof. Eduardo Baeza**

**Desarrollo Fullstack I 011D**

**Introducción**

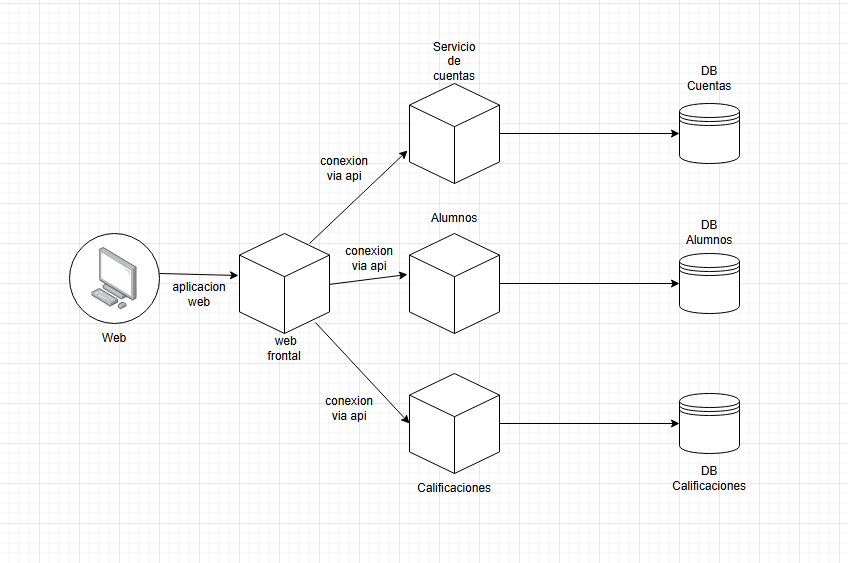
EduTech Innovators SPA es una empresa emergente enfocada en plataformas educativas en línea. Su arquitectura monolítica original dificulta la escalabilidad, disponibilidad y mantenimiento del sistema. En respuesta a estas limitaciones, se implementó una arquitectura basada en **microservicios**, donde cada funcionalidad del sistema se desarrolló de forma modular y desacoplada, con el objetivo de mejorar la eficiencia operativa, el rendimiento y la experiencia de usuario.

En esta entrega se desarrollaron tres microservicios clave:

* Gestión de usuarios
* Gestión de cursos
* Gestión de evaluaciones

Cada servicio se diseñó como una API REST independiente que se comunica con una **única base de datos MySQL**. Se utilizaron herramientas modernas como Java 23, Spring Boot, Visual Studio Code, Git y Postman para lograr un desarrollo completo, escalable y mantenible.

***Diagrama de arquitectura de microservicios***



**Estructura del proyecto**

El proyecto se construyó bajo una arquitectura de microservicios, donde cada servicio se desarrolla como una aplicación Spring Boot independiente. Para mantener la organización y buenas prácticas de desarrollo, se siguió una estructura basada en el patrón **MVC (Modelo - Vista - Controlador)**, aunque en este caso no se implementan vistas gráficas, sino que la "vista" corresponde a las respuestas JSON de las APIs.

Cada microservicio contiene sus propios archivos de configuración, su base de código separada y su lógica de negocio, permitiendo el despliegue y mantenimiento de forma independiente.

users-service/

├── src/

│ └── main/

│ ├── java/

│ │ └── com/edutech/users/

│ │ ├── controller/

│ │ │ └── UserController.java

│ │ ├── model/

│ │ │ └── User.java

│ │ ├── repository/

│ │ │ └── UserRepository.java

│ │ └── UsersServiceApplication.java

│ └── resources/

│ ├── application.properties

├── pom.xml

**Controller**: contiene las clases que exponen las rutas (endpoints) del servicio mediante anotaciones como @*RestController*. Cada clase representa una API que responde a peticiones HTTP (GET, POST, PUT, DELETE).

**Model**: define las entidades que se almacenan en la base de datos. Cada clase representa una tabla. Se usan anotaciones de JPA como @*Entity*, @*Id*, @*Column*, entre otras.

**Repository**: contiene las interfaces que extienden JpaRepository. Estas se encargan de comunicarse con la base de datos y realizar operaciones como guardar, buscar o eliminar registros.

**application.properties**: archivo de configuración donde se define la conexión con la base de datos MySQL, el puerto del servidor, y propiedades de JPA e Hibernate.

**UsersServiceApplication.java**: clase principal del microservicio, con la anotación @*SpringBootApplication*, que sirve como punto de entrada de la aplicación.

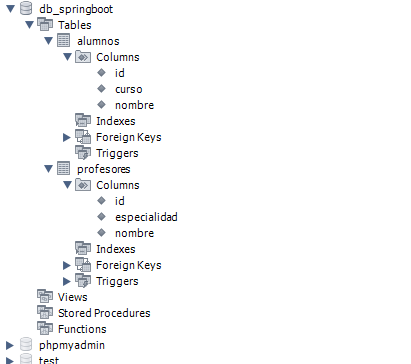
***Estructura de Base de datos***

En esta implementación se utilizó una única base de datos MySQL, compartida por los tres microservicios del sistema:

* users-service
* courses-service
* evaluations-service

El acceso a la base de datos fue gestionado mediante Spring Data JPA, lo que permitió generar automáticamente las tablas desde las entidades Java, sin necesidad de escribir consultas SQL manuales. Las tablas generadas para el proyecto fueron:

| **Tabla** | **Microservicio** | **Descripción** |
| --- | --- | --- |
| user | users-service | Almacena información de usuarios (nombre, correo, rol) |
| course | courses-service | Contiene los cursos disponibles en la plataforma |
| evaluation | evaluations-service | Registra evaluaciones asociadas a cursos |



**Implementación de los Servicios**

Como parte del proceso de transformación digital de EduTech Innovators SPA, se desarrollaron tres microservicios independientes que representan funcionalidades clave del sistema: ***gestión de usuarios****,* ***gestión de cursos*** *y* ***gestión de evaluaciones***.

Todos los microservicios comparten una estructura común que incluye:

* **Entidad (Model)**: define la estructura de los datos que se almacenan en la base de datos. Se anotan con @*Entity* para que Spring JPA las reconozca como tablas.
* **Repositorio (Repository)**: se define como una interfaz que extiende JpaRepository, lo que permite realizar operaciones CRUD (crear, leer, actualizar y eliminar) sin escribir consultas SQL.
* **Controlador (Controller)**: expone una API REST utilizando anotaciones como *@RestController y @RequestMapping*, permitiendo que otros sistemas o herramientas (como Postman o un futuro frontend) interactúen con el servicio.

Cada microservicio fue configurado con su propio puerto y archivo *application.properties*, en el que se definieron los datos de conexión a la base de datos MySQL, la configuración del dialecto, y otras propiedades necesarias para el funcionamiento del servicio.

***Microservicio de Gestión de Usuarios***

El microservicio de usuarios permite registrar, consultar, modificar y eliminar usuarios del sistema. La entidad definida fue User, con los atributos *id, name y email.*

Se implementaron los endpoints necesarios para cumplir con las operaciones básicas:

* GET para obtener todos los usuarios o uno específico por su ID,
* POST para crear nuevos usuarios,
* PUT para actualizar datos existentes,
* DELETE para eliminar usuarios.

*El servicio fue probado completamente con Postman, verificando el correcto almacenamiento y recuperación de los datos.*

***Microservicio de Gestión de Cursos***

Este microservicio permite administrar los cursos disponibles en la plataforma. La entidad correspondiente, Course, incluye los atributos *id, title, description e instructor*. A través del controlador se definieron los mismos métodos CRUD para la gestión de cursos.

Una vez desplegado el servicio, se validó que los cursos se almacenarán correctamente en la base de datos y que se pudiera acceder a ellos desde los distintos endpoints. Este microservicio será fundamental para futuras funcionalidades como la inscripción de usuarios o la asignación de evaluaciones.

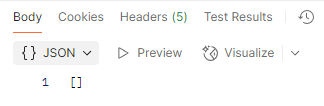
***Microservicio de Gestión de Evaluaciones***

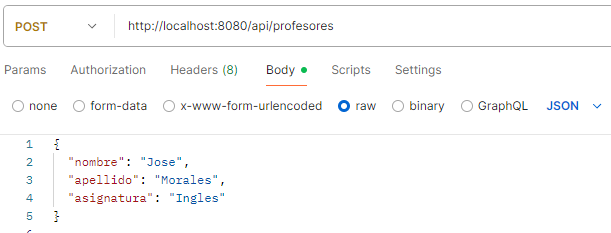
El tercer microservicio corresponde al módulo de evaluaciones. Se implementó la entidad Evaluation, que contiene los campos id, title, type (tipo de evaluación: prueba, tarea, etc.) y courseId, que representa una relación lógica con un curso.

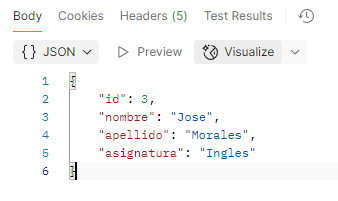
Este servicio también expone los endpoints necesarios para realizar las operaciones CRUD básicas. Permite registrar nuevas evaluaciones, consultarlas por ID, actualizarlas o eliminarlas. En versiones futuras se podría agregar validación cruzada con el microservicio de cursos para verificar que el courseId exista antes de guardar.

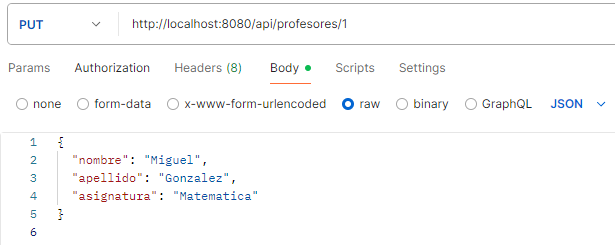
**Vistas de peticiones en Postman**

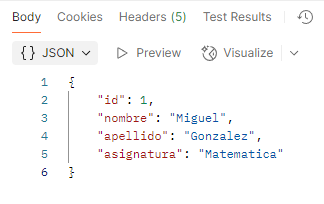
****

**→ **

****

**→ **

****

**→ **

****

**Git y GitHub**

Durante el desarrollo del proyecto se utilizó Git como sistema de control de versiones, y GitHub como plataforma de respaldo y colaboración en línea. Esto permitió mantener un historial claro de los avances del proyecto, facilitar el trabajo ordenado y asegurar la disponibilidad del código desde cualquier lugar.

El control de versiones es una práctica fundamental en el desarrollo profesional de software, ya que permite registrar cada cambio en el proyecto, deshacer errores y coordinar el trabajo en equipo. Git, en conjunto con GitHub, proporcionó una forma eficiente de cumplir con estos objetivos.

A continuación, se explican los comandos utilizados en el flujo de trabajo del proyecto.

***Inicialización del repositorio local:*** *git init*

Este comando se utilizó para convertir la carpeta del proyecto en un repositorio Git. Esto permitió comenzar a registrar cambios localmente.

### ***Configuración del usuario:***

### *git config user.name "DantteG"*

### *git config user.email "*[*dant.garcia@duocuc.cl*](mailto:dant.garcia@duocuc.cl)*"*

Se configuraron el nombre y el correo electrónico del autor, que se asocian a cada commit para llevar un registro de quién realizó cada cambio.

***Autenticación con GitHub:***

*git config user.password "ghp\_xxx..."*

Se configuró un token personal de acceso (PAT) en lugar de una contraseña, como lo exige GitHub actualmente por motivos de seguridad. Esto permite autenticar las operaciones de push sin errores.

***Conexión con el repositorio remoto:***

*git remote add origin* [*https://github.com/DanttEG/RepoEvaluacion2*](https://github.com/DanttEG/RepoEvaluacion2)

Este comando conectó el repositorio local con un repositorio remoto alojado en GitHub. A partir de aquí, fue posible subir el código desde el equipo local a la nube.

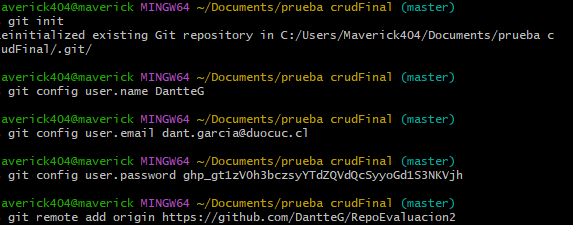
***Registro de cambios y código:***

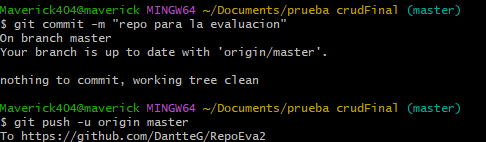
*git commit -m "repo para la evaluacion"*

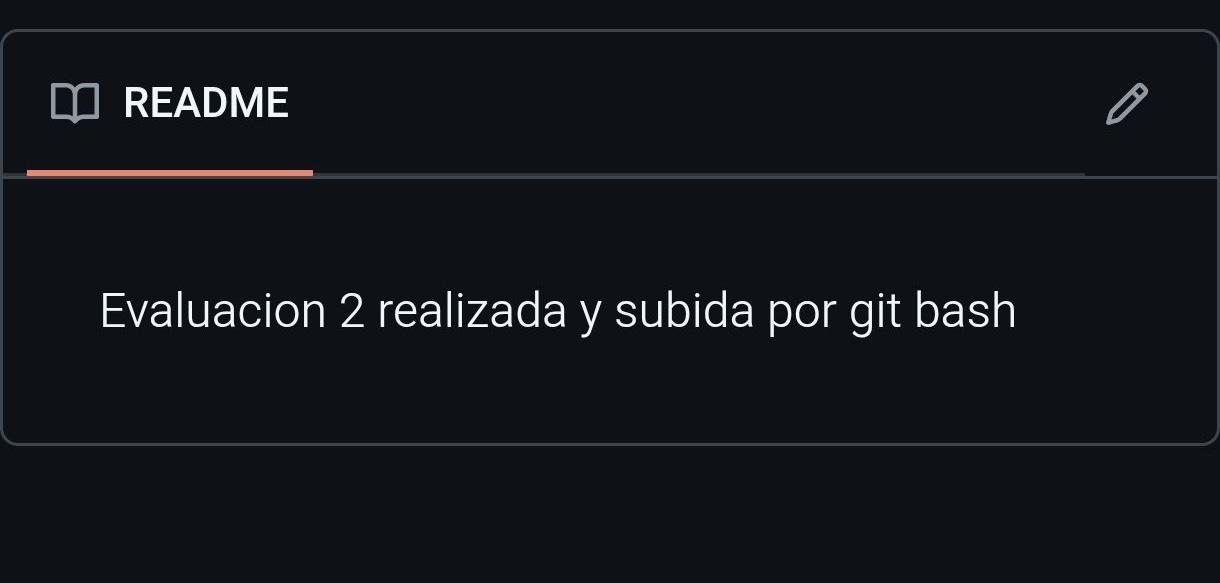
Se creó un commit que guarda una versión puntual del proyecto, incluyendo todos los archivos que se hayan modificado o agregado desde el último commit

*git push -u origin master*

Finalmente, se subió el proyecto al repositorio remoto en la rama principal (master) y se estableció como la rama por defecto para futuros envíos.







**Conclusión**

La implementación de microservicios para EduTech Innovators SPA permitió transformar una arquitectura monolítica en una solución moderna, modular y escalable. Se logró dividir el sistema en servicios independientes, facilitando la administración de código, pruebas y posibles futuras extensiones. Esta arquitectura no solo mejora el rendimiento y mantenimiento, sino que también sienta las bases para la integración de nuevos servicios como notificaciones, pagos o autenticación segura.