APPEDUCAONLINE

Raquel Cerdá

2º DAM

UNIR

Tabla de contenido

INTRODUCCION

En el dinámico y en constante evolución ámbito de la tecnología educativa, la implementación de sistemas basados en microservicios representa una tendencia creciente y altamente efectiva. Este anteproyecto tiene como objetivo el desarrollo de un innovador sistema educativo en línea, utilizando tecnologías modernas como Spring Boot, Spring Data JPA e Hibernate. El sistema está diseñado para proporcionar una experiencia educativa integral y optimizada, facilitando la administración y el acceso a cursos estructurados en módulos interactivos, así como la gestión de evaluaciones y retroalimentación inmediata.

El proyecto se centra en la creación de una plataforma educativa en línea que permite a los usuarios registrarse, acceder a cursos, realizar exámenes y recibir respuestas instantáneas a sus evaluaciones. Cada curso está compuesto por módulos con contenido multimedia y cuestionarios asociados, cuyos resultados serán almacenados y analizados para mejorar continuamente la experiencia educativa.

MODULOS FORMATIVOS APLICADOS AL TRABAJO

Acceso a Datos

Acceso a Datos es una disciplina fundamental para la gestión eficiente de la persistencia de datos en aplicaciones empresariales. En este proyecto, se han utilizado tecnologías como **Spring Data JPA** y **Hibernate** para manejar las operaciones de acceso a bases de datos.

Spring Data JPA simplifica la implementación de repositorios JPA, permitiendo realizar operaciones CRUD (Crear, Leer, Actualizar, Eliminar) de manera declarativa sin necesidad de escribir código SQL explícito. Este enfoque no solo reduce significativamente el código necesario, sino que también mejora la mantenibilidad y escalabilidad del sistema.

Hibernate es un framework de mapeo objeto-relacional que facilita la interacción entre los objetos de la aplicación y la base de datos. Hibernate gestiona automáticamente las transacciones y las consultas SQL, proporcionando un nivel de abstracción que simplifica el desarrollo de aplicaciones. Además, Hibernate soporta el caching de segundo nivel, lo que mejora el rendimiento al reducir el número de consultas a la base de datos.

Desarrollo de Interfaces

Desarrollo de Interfaces se centra en la creación de interfaces de usuario intuitivas y atractivas para mejorar la interacción del usuario con el sistema. La experiencia del usuario (UX) y la interfaz de usuario (UI) son componentes cruciales para el éxito de cualquier aplicación.

En este proyecto, se ha utilizado **Angular**, un framework de desarrollo frontend que permite la creación de aplicaciones web dinámicas y de alto rendimiento. Angular facilita la construcción de interfaces de usuario mediante componentes reutilizables, lo que mejora la consistencia y la mantenibilidad del código. Además, Angular proporciona herramientas integradas para pruebas, compilación y despliegue, lo que acelera el proceso de desarrollo.

La utilización de Angular permite consumir microservicios y mostrar la información de manera interactiva y eficiente. Esto garantiza una experiencia de usuario fluida y agradable, mejorando la satisfacción y el compromiso de los usuarios con la plataforma educativa.

Programación de Servicios y Procesos

Programación de Servicios y Procesos es esencial para el desarrollo de microservicios independientes y eficientes que gestionan diversas funcionalidades del sistema. La arquitectura de microservicios permite dividir la aplicación en componentes más pequeños y manejables, cada uno responsable de una funcionalidad específica.

En este proyecto, se han desarrollado microservicios utilizando **Spring Boot** y **Spring Cloud**. **Spring Boot** facilita la creación de aplicaciones basadas en microservicios mediante una configuración predeterminada que simplifica la configuración y el despliegue. Con Spring Boot, cada microservicio puede ser empaquetado como una aplicación independiente, lista para ejecutarse en cualquier entorno.

Spring Cloud proporciona un conjunto de herramientas para gestionar la configuración distribuida, el descubrimiento de servicios, el balanceo de carga, las llamadas remotas y la tolerancia a fallos. Estas herramientas son cruciales para construir sistemas distribuidos que puedan escalar horizontalmente y manejar la alta disponibilidad de manera eficiente. Los microservicios en este proyecto se comunican entre sí utilizando API REST, lo que permite una integración sencilla y flexible.

HERRAMIENTAS/LENGUAJES UTILIZADOS

Para el desarrollo de esta plataforma educativa en línea, se han seleccionado cuidadosamente una serie de tecnologías que garantizan la robustez, escalabilidad y facilidad de mantenimiento del sistema. A continuación, se ofrece una descripción detallada de cada una de estas tecnologías y sus beneficios.

Spring Boot

Spring Boot es un framework que facilita la creación de aplicaciones empresariales en Java. Ofrece una configuración predeterminada que permite a los desarrolladores concentrarse en la lógica de negocio en lugar de los detalles de configuración. Con Spring Boot, se pueden crear aplicaciones independientes, que son fáciles de ejecutar y desplegar, gracias a su capacidad de empaquetar el servidor web dentro de la aplicación. Además, Spring Boot proporciona una serie de características integradas como métricas, salud del sistema y configuración externa que ayudan a gestionar aplicaciones en producción de manera eficiente.

Spring IoC

Spring Inversion of Control (IoC) es un patrón de diseño que permite la gestión de la configuración y creación de objetos de manera eficiente. Con IoC, los desarrolladores pueden desacoplar la configuración y la dependencia de los objetos, lo que facilita la construcción de aplicaciones modulares y escalables. Spring IoC utiliza contenedores que gestionan el ciclo de vida de los objetos, inyectando las dependencias necesarias en tiempo de ejecución. Esto no solo mejora la mantenibilidad del código, sino que también promueve la reutilización de componentes.

Spring Data JPA e Hibernate

Spring Data JPA y **Hibernate** son herramientas cruciales para el acceso y gestión eficiente de bases de datos relacionales como MySQL. Spring Data JPA simplifica la implementación de repositorios basados en JPA (Java Persistence API), permitiendo realizar operaciones de persistencia de manera declarativa. Hibernate, por otro lado, es un framework de mapeo objeto-relacional que facilita la interacción entre los objetos de la aplicación y la base de datos, gestionando automáticamente las transacciones y las consultas SQL. Juntos, estos componentes permiten una gestión eficiente y optimizada de la base de datos, reduciendo significativamente el código boilerplate y mejorando la productividad del desarrollo.

Maven

Maven es una herramienta de gestión de proyectos y comprensión que facilita la construcción y gestión de dependencias. Maven utiliza un archivo de configuración XML llamado pom.xml para gestionar la configuración del proyecto, las dependencias y las tareas de construcción. Con Maven, los desarrolladores pueden automatizar la

compilación, el empaquetado y la prueba de aplicaciones Java, lo que mejora la eficiencia y la consistencia en el ciclo de vida del desarrollo del software.

API REST

API REST (Representational State Transfer) es un estilo de arquitectura para el desarrollo de servicios web escalables y flexibles. Utilizando métodos HTTP estándar, como GET, POST, PUT y DELETE, las API REST permiten la comunicación entre el cliente y el servidor de manera sencilla y eficiente. Las API RESTful son independientes de la plataforma y del lenguaje, lo que facilita la integración con diversas tecnologías y sistemas. Además, su enfoque en la comunicación stateless mejora la escalabilidad y la simplicidad del diseño del sistema.

Spring Cloud

Spring Cloud proporciona un conjunto de herramientas para la construcción de sistemas distribuidos y escalables en la nube. Con Spring Cloud, los desarrolladores pueden gestionar la configuración distribuida, el descubrimiento de servicios, el balanceo de carga, las llamadas remotas y la tolerancia a fallos. Estas herramientas son esenciales para construir microservicios que puedan escalar horizontalmente y manejar la alta disponibilidad de manera eficiente.

Eureka Netflix y Eureka Client

Eureka Netflix y **Eureka Client** son componentes clave para el registro y descubrimiento de servicios en una arquitectura de microservicios. Eureka Server actúa como un registro donde todos los microservicios se registran y descubren entre sí. Eureka Client, por otro lado, se utiliza para registrar los servicios en el servidor Eureka y para buscar otros servicios registrados. Este mecanismo permite la comunicación eficiente entre microservicios y mejora la resiliencia del sistema al permitir la detección automática de servicios fallidos.

Spring Cloud Gateway

Spring Cloud Gateway es una herramienta que proporciona enrutamiento y administración centralizada de solicitudes en una arquitectura de microservicios. Funciona como un proxy inverso que dirige las solicitudes a los microservicios adecuados basándose en diversas reglas de enrutamiento. Además, ofrece características avanzadas como el balanceo de carga, la limitación de tasa, la reintento de solicitudes y la manipulación de rutas, lo que mejora la seguridad y la eficiencia de la gestión de tráfico en el sistema.

Spring Cloud LoadBalancer

Spring Cloud LoadBalancer es una herramienta que permite el balanceo de carga entre microservicios registrados en Eureka. Distribuye las solicitudes entrantes entre varias instancias de servicio para asegurar que ninguna instancia individual se sobrecargue. Esto mejora la disponibilidad y la capacidad de respuesta del sistema, garantizando que los servicios puedan escalar horizontalmente para manejar mayores cargas de trabajo.

MySQL, MongoDB y PostgreSQL

MySQL, **MongoDB** y **PostgreSQL** son sistemas de gestión de bases de datos utilizados para el almacenamiento de datos. MySQL y PostgreSQL son bases de datos relacionales que ofrecen integridad referencial, transacciones ACID y soporte para SQL. MongoDB, en cambio, es una base de datos NoSQL que permite almacenar datos en un formato de documento flexible, ideal para aplicaciones que requieren escalabilidad horizontal y un esquema dinámico. La combinación de estas bases de datos permite al sistema aprovechar las fortalezas de cada una para diferentes casos de uso.

Postman

Postman es una herramienta poderosa para probar y consumir microservicios. Permite a los desarrolladores enviar solicitudes HTTP a sus APIs y ver las respuestas, facilitando el proceso de depuración y prueba. Con Postman, se pueden crear y gestionar colecciones de solicitudes, automatizar pruebas y documentar APIs de manera efectiva, lo que mejora significativamente la eficiencia en el desarrollo y mantenimiento de servicios web.

Angular

Angular es un framework utilizado para el desarrollo de la parte frontend de la aplicación. Angular permite la creación de aplicaciones web dinámicas y de alto rendimiento mediante la utilización de componentes reutilizables y la inyección de dependencias. Con Angular, los desarrolladores pueden consumir microservicios y mostrar la información de manera interactiva y eficiente, mejorando la experiencia del usuario final. Además, Angular ofrece un ecosistema robusto con herramientas integradas para pruebas, compilación y despliegue, lo que facilita el desarrollo y la entrega continua de aplicaciones.

En resumen, la combinación de estas tecnologías proporciona una base sólida para el desarrollo de una plataforma educativa en línea robusta, escalable y fácil de mantener. Cada tecnología aporta características específicas que, en conjunto, aseguran un rendimiento óptimo, una gestión eficiente de los recursos y una experiencia de usuario superior.

COMPONENTE DEL EQUIPO

. Jorge Ruiz Martínez : Elección de tipo de proyecto a realizar, selección de tecnologías que se van a utilizar y realización de todo el desarrollo del proyecto completo.

FASES DEL PROYECTO

. Estudio del mercado:

Viendo el auge de la formación online decimos crear un educativa para poder almacenar alumnos, cursos, exámenes y los resultados de esos exámenes.

. Modelo de datos:

El diseño de la base de datos para el sistema educativo en línea basado en microservicios se describe a continuación. Se utilizarán bases de datos relacionales (MySQL) para la mayoría de los microservicios, y bases de datos NoSQL (MongoDB) para algunos casos específicos donde sea necesario.

Base de Datos Relacional (MySQL y PostgreSQL)

Tablas y Relaciones

1. **Tabla usuarios**
   - id: INT, Primary Key, Auto Increment
   - nombre: VARCHAR(255)
   - apellido: VARCHAR(255)
   - email: VARCHAR(255), Unique
   - createAt: TIMESTAMP
   - foto: BLOB
2. **Tabla cursos**
   - id: INT, Primary Key, Auto Increment
   - nombre: VARCHAR(255)
   - createAt: TIMESTAMP
3. **Tabla** exámenes
   - id: INT, Primary Key, Auto Increment
   - nombre: VARCHAR(255)
   - createAt: TIMESTAMP
   - curso_id: INT, Foreign Key, References cursos(id)
4. **Tabla preguntas**
   - id: INT, Primary Key, Auto Increment
   - texto: TEXT
   - examen_id: INT, Foreign Key, References exámenes (id)
5. **Tabla asignaturas**
   - id: INT, Primary Key, Auto Increment
   - nombre: VARCHAR(255)
   - examen_id: INT, Foreign Key, References exámenes (id)

Relaciones entre Tablas

- Un curso puede tener muchos exámenes.
- Un examen puede tener muchas preguntas.
- Una pregunta puede tener muchas respuestas.
- Un usuario puede tener muchas respuestas.
- Un examen puede tener una asignatura.

Base de Datos NoSQL (MongoDB)

En algunos casos, utilizaremos MongoDB para almacenar datos que requieren una mayor flexibilidad en la estructura.

1. **Microservicio respuestas**

```
{
    "_id": ObjectId,
    "texto": String,
    "pregunta_id": ObjectId,
    "usuario_id": ObjectId
}
```

. Diagramas UML:

Diagrama de Casos de uso

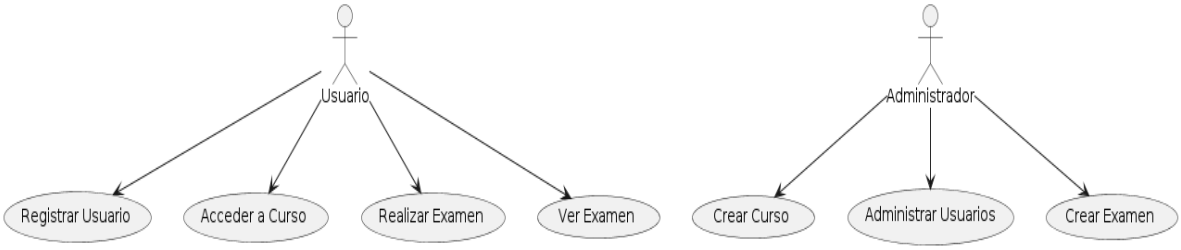
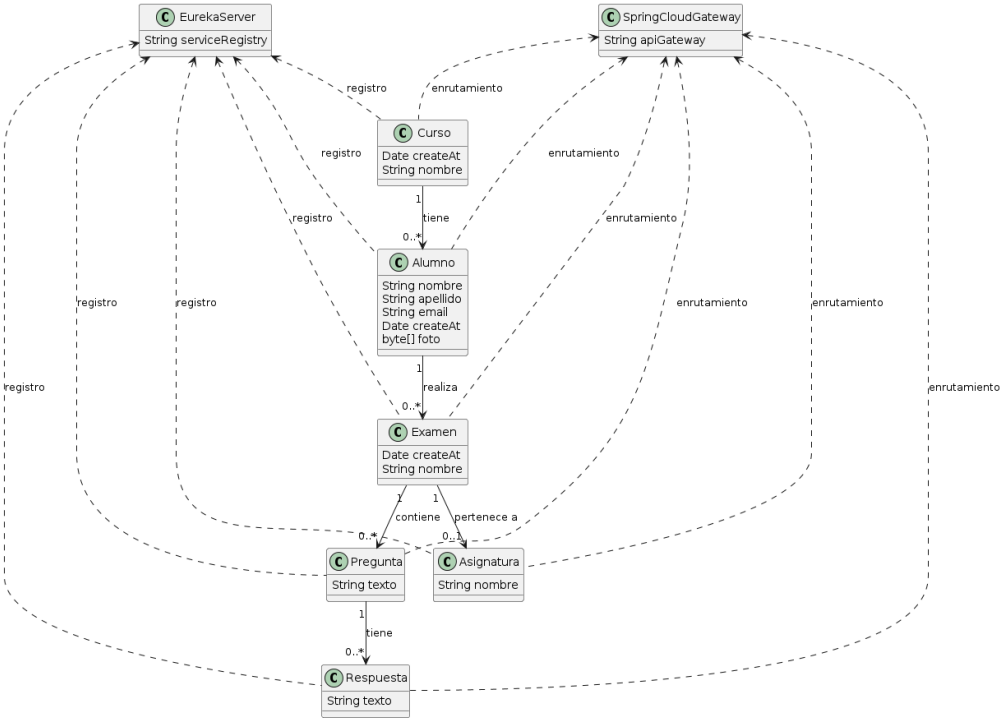
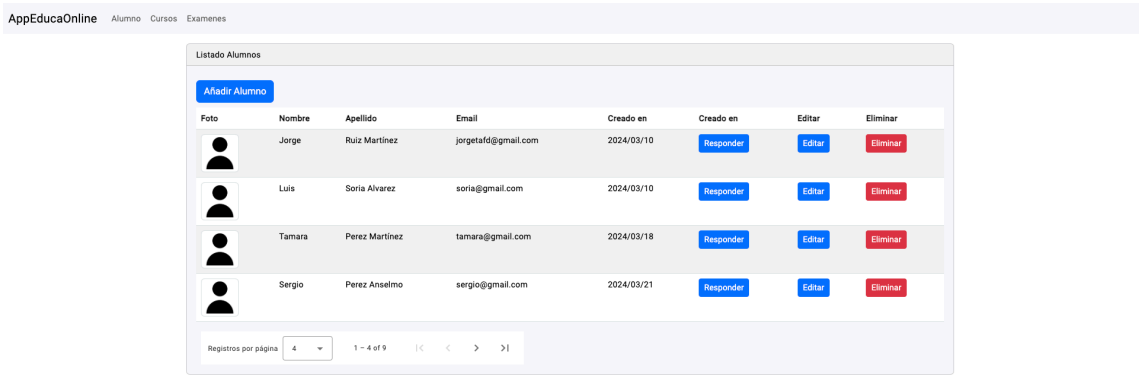


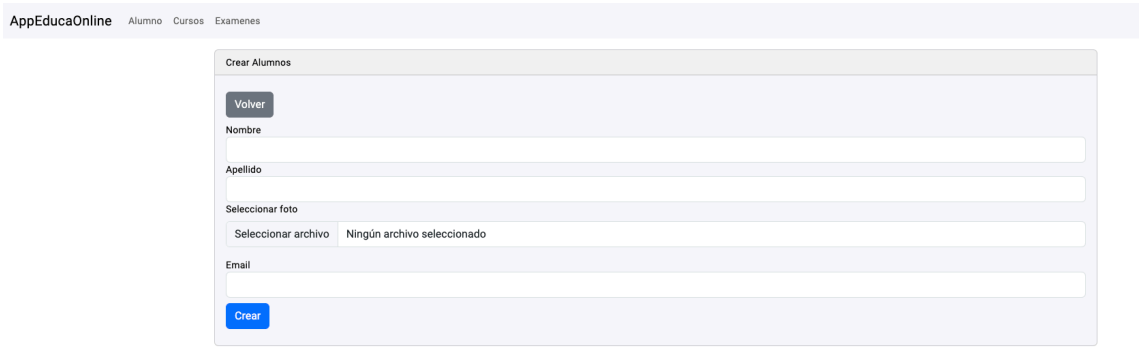
Diagrama de Clases



. Diseños de interfaces:

Todas las pantallas de la aplicación tendrán la misma apariencia utilizando angular material y bootstrap.





. Planificación del proyecto:

Fase	Actividades	Duración
Planificación y Análisis	Definición de requisitos, diseño de arquitectura, UML	2 semanas
Desarrollo del Backend	Configuración, desarrollo de microservicios, integración	10 semanas
Desarrollo del Frontend	Configuración, desarrollo de UI, integración	11 semanas

. Funcionalidad:

El proyecto se enfoca en la implementación de una plataforma educativa en línea basada en microservicios, utilizando tecnologías como Spring Boot, Spring Data JPA, y Hibernate. A continuación, se describen las funcionalidades principales del proyecto:

1. Registro de Usuarios/Alumnos

Descripción: Permite a los usuarios registrarse en la plataforma proporcionando información básica como nombre, apellido, correo electrónico y una fotografía opcional.
Microservicio: Microservicio de Usuarios.
Endpoints: POST /usuarios - Registra un nuevo usuario.

2. Gestión de Cursos

Descripción: Permite a los administradores crear, editar y eliminar cursos. Los usuarios pueden acceder a los cursos disponibles.
Microservicio: Microservicio de Cursos.
Endpoints:
POST /cursos - Crea un nuevo curso.
PUT /cursos/{id} - Edita un curso existente.
DELETE /cursos/{id} - Elimina un curso.
GET /cursos - Lista todos los cursos.

3. Gestión de Exámenes

Descripción: Permite la creación, edición y eliminación de exámenes asociados a los cursos. Los exámenes contienen preguntas y asignaturas.
Microservicio: Microservicio de Exámenes.
Endpoints:
POST /examenes - Crea un nuevo examen.
PUT /examenes/{id} - Edita un examen existente.
DELETE /examenes/{id} - Elimina un examen.
GET /examenes - Lista todos los exámenes.
POST /preguntas - Añade una pregunta a un examen.
DELETE /preguntas/{id} - Elimina una pregunta de un examen.

4. Realización de Exámenes

Descripción: Permite a los usuarios realizar exámenes y enviar sus respuestas. Las respuestas se almacenan para su posterior análisis.
Microservicio: Microservicio de Exámenes y Microservicio de Respuestas.
Endpoints:
POST /respuestas - Envía respuestas de un examen.
GET /respuestas/{usuarioId}/{examenId} - Obtiene las respuestas de un usuario para un examen específico.

CONCLUSION Y MEJORAS DEL PROYECTO

Conclusiones

El desarrollo de la plataforma educativa en línea basada en microservicios ha sido un proyecto desafiante y gratificante. A través de este proyecto, hemos logrado implementar una arquitectura moderna y escalable que facilita la gestión de cursos, exámenes y usuarios de manera eficiente. Algunas de las conclusiones clave del proyecto son:

1. **Arquitectura de Microservicios**:
    - La elección de una arquitectura de microservicios ha permitido una mayor flexibilidad y escalabilidad. Cada componente del sistema puede desarrollarse, desplegarse y escalarse de manera independiente, lo que facilita el mantenimiento y la evolución del sistema a lo largo del tiempo.
2. **Uso de Tecnologías Modernas**:
    - La utilización de tecnologías como Spring Boot, Spring Data JPA, Hibernate, y Angular ha demostrado ser eficaz para construir aplicaciones robustas y mantenibles. Estas tecnologías han facilitado la creación de APIs RESTful, la gestión de bases de datos y la implementación de una interfaz de usuario intuitiva.
3. **Gestión Eficiente de Datos**:
    - La integración de bases de datos relacionales (MySQL y PostgreSQL) y NoSQL (MongoDB) ha proporcionado un equilibrio entre estructura y flexibilidad, permitiendo un almacenamiento eficiente y escalable de la información.

Mejoras Sugeridas

1. **Optimización del Rendimiento**:
    - Aunque la arquitectura de microservicios proporciona escalabilidad, se pueden implementar técnicas adicionales de optimización, como el uso de caching y la optimización de consultas de base de datos, para mejorar el rendimiento del sistema.
2. **Seguridad Avanzada**:
    - Incluir mecanismos de seguridad más avanzados, como la autenticación multifactor (MFA) y la encriptación de datos en reposo, para mejorar la protección de la información del usuario.
3. **Experiencia de Usuario**:
    - Continuar mejorando la interfaz de usuario con retroalimentación de los usuarios finales. Realizar pruebas de usabilidad y aplicar mejoras en la interfaz para hacerla más intuitiva y accesible.

Resumen de los Objetivos Conseguidos

1. **Desarrollo de una Plataforma Educativa Robusta**:
    - Se ha creado una estructura basada en microservicios que garantiza escalabilidad y flexibilidad. La plataforma permite la gestión de cursos, exámenes y usuarios de manera eficiente.
2. **Gestión Eficiente de Usuarios y Cursos**:
    - Se han implementado microservicios específicos para la administración de usuarios, cursos, exámenes y respuestas, proporcionando una gestión centralizada y eficiente.
3. **Optimización de la Experiencia de Usuario**:
    - La interfaz de usuario ha sido diseñada para ser intuitiva y proporcionar respuestas instantáneas, mejorando significativamente la satisfacción del usuario.
4. **Análisis y Mejora Continua**:
    - Los datos recopilados a través de la plataforma se analizan para proporcionar retroalimentación y mejorar los contenidos educativos de manera continua.

Resultados Obtenidos

1. **Registro y Gestión de Usuarios**:
    - Implementación exitosa de un sistema de registro de usuarios que permite la gestión eficiente de la información de los usuarios.
2. **Gestión de Cursos y Exámenes**:
    - Creación y administración efectiva de cursos y exámenes, con funcionalidades para la creación, edición y eliminación de estos elementos.
3. **Evaluaciones y Retroalimentación**:
    - Sistema de evaluaciones que permite a los usuarios realizar exámenes y recibir retroalimentación instantánea, mejorando la experiencia de aprendizaje.

En resumen, el proyecto ha alcanzado sus objetivos principales, proporcionando una plataforma educativa robusta, escalable y eficiente. Las mejoras sugeridas y las prácticas de mantenimiento continuo asegurarán que el sistema siga evolucionando y mejorando, ofreciendo una experiencia educativa de alta calidad a los usuarios.

BIBLIOGRAFÍA

Documentación de Software

- Spring Framework. (n.d.). **Spring Boot Reference Documentation**. Recuperado de https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/
- Hibernate. (n.d.). **Hibernate ORM Documentation**. Recuperado de https://hibernate.org/orm/documentation/
- Angular. (n.d.). **Angular Documentation**. Recuperado de https://angular.io/docs

Herramientas y Tecnologías Utilizadas

- Spring Boot: https://spring.io/projects/spring-boot
- Hibernate: https://hibernate.org/
- Angular: https://angular.io/
- Docker: https://www.docker.com/
- PostgreSQL: https://www.postgresql.org/
- MongoDB: https://www.mongodb.com/
- Kubernetes: https://kubernetes.io/
- Postman: https://www.postman.com/