

Plataforma Educativa Interactiva - EduVibe



EduVibe

Nombre: Javier Pintado Navarro

Curso académico: (2023-2024)

Curso: 2º Desarrollo de aplicaciones web

Tutor: Antonio Gabriel González Casado



ÍNDICE

1. INTRODUCCIÓN.....	3
2. IDENTIFICACIÓN DE LAS NECESIDADES DE EDUVIBE.....	4
3. ANÁLISIS CON LAS ALTERNATIVAS DEL MERCADO.....	6
4. JUSTIFICACIÓN DEL PROYECTO.....	8
5. STACK TECNOLÓGICO.....	9
6. ESQUEMA E/R.....	10
1. Entidad: usuarios.....	10
2. Entidad: clases.....	10
3. Entidad: tarea.....	11
4. Entidad: usuario_clase.....	11
5. Relaciones entre Entidades.....	11
6. Descripción de las Relaciones.....	11
7. PROTOTIPADO APLICACIÓN WEB.....	12
8. DEFINICIÓN API REST.....	16
9. MANUAL DE DESPLIEGUE (DOCKER).....	19
Paso 1: Preparación del Entorno.....	19
Paso 2: Construcción y Despliegue del Backend.....	19
Paso 3: Construcción y Despliegue del Frontend.....	23
Paso 4: Ejecución de la Aplicación.....	25
Paso 5: Acceso a la Aplicación.....	25
10. CONCLUSIÓN.....	26
Posibles Mejoras.....	26
Viabilidad de Puesta en Marcha Real.....	27
11. ANEXOS.....	28
Dependencias de Spring Boot utilizadas en EduVibe:.....	28
Dependencias de Angular utilizadas en EduVibe.....	30



1. INTRODUCCIÓN

EduVibe es una plataforma en línea creada para hacer que el aprendizaje sea más interactivo y fácil tanto para profesores como para estudiantes. En un mundo donde cada vez hacemos más cosas en internet, la educación también ha dado ese salto al mundo digital. EduVibe está aquí para hacer que esa experiencia educativa en línea sea mucho mejor.

Lo que hace EduVibe es ayudar a los profesores a organizar sus clases y a interactuar de manera más fácil con sus alumnos. Esta plataforma tiene varias herramientas que hacen que aprender en línea sea más sencillo y divertido.

Los profesores pueden usar EduVibe para poner tareas, hacer seguimiento del progreso de los estudiantes y saber sus calificaciones de manera rápida y sencilla y para los estudiantes, es una forma genial de tener todo lo que necesitan para aprender en un solo lugar.

2. IDENTIFICACIÓN DE LAS NECESIDADES DE EDUVIBE

Gestión de Usuarios:

RF01. Permitir al Administrador realizar operaciones CRUD completas (Crear, Leer, Actualizar, Eliminar) sobre los usuarios, incluyendo la creación, edición y eliminación de cuentas de usuario, así como la capacidad de ver la información de los usuarios.

Roles de Usuario Diferenciados:

RF02. Administrador: Acceso total a la plataforma, incluyendo la gestión de usuarios, clases y tareas.

RF03. Profesor: Capacidad de editar las clases asignadas por el administrador, así como realizar operaciones CRUD completas sobre las tareas asignadas a esas clases.

RF04. Alumno: Acceso a sus clases asignadas y capacidad de ver las tareas asignadas por los profesores, además de poder agregar sus propias tareas para que los profesores las revisen.

Gestión de Clases:

RF05. Permitir a los profesores editar las clases que les han asignado los administradores, estos últimos son los que tienen la capacidad de agregar, editar y eliminar clases según sea necesario.

Gestión de Tareas:

RF06. Permitir a los profesores realizar operaciones CRUD completas sobre las tareas asignadas a sus clases, incluyendo crear nuevas tareas, editar tareas existentes, poner calificaciones a estas, eliminar tareas no deseadas y asignar tareas a los alumnos.

RF07. Permitir a los alumnos ver las tareas asignadas por los profesores y agregar sus propias soluciones para su revisión.

Interfaz de Usuario Intuitiva:

RF08. La aplicación debe tener una interfaz de usuario fácil de usar y amigable para los tres roles de usuarios, garantizando que los administradores, profesores y alumnos puedan navegar fácilmente por la plataforma y acceder a las funcionalidades relevantes de manera eficiente.



Seguridad y Autenticación:

RF09. Implementar un sistema de autenticación seguro para garantizar que solo los usuarios autorizados puedan acceder a la plataforma.

RF10. Establecer controles de acceso para asegurar que cada usuario solo pueda realizar las acciones permitidas según su rol asignado.

3. ANÁLISIS CON LAS ALTERNATIVAS DEL MERCADO

Para hacer un análisis breve y una comparativa con las alternativas del mercado para EduVibe, podemos considerar algunas de las plataformas educativas en línea más populares y compararlas en función de ciertos criterios clave:

Google Classroom:

Ventajas:

- Integración con el ecosistema de Google, lo que facilita el acceso para aquellos que ya utilizan otras herramientas de Google.
- Ofrece una variedad de funciones de gestión de clases y tareas, como la creación y distribución de asignaciones, la retroalimentación en línea y la comunicación entre profesores y alumnos.

Desventajas:

- Limitaciones en la personalización y flexibilidad de la plataforma en comparación con otras soluciones.

Moodle:

Ventajas:

- Plataforma de código abierto con una comunidad activa de desarrolladores y usuarios que ofrecen una amplia gama de complementos y extensiones.
- Ofrece una gran flexibilidad y personalización para adaptarse a las necesidades de diferentes tipos de instituciones educativas.

Desventajas:

- La interfaz de usuario puede parecer desactualizada y menos intuitiva en comparación con otras soluciones más modernas.
- Requiere conocimientos técnicos para la instalación, configuración y mantenimiento, lo que puede ser una barrera para algunos usuarios.

Schoology:

Ventajas:

- Ofrece una combinación equilibrada de funcionalidades de gestión de clases y herramientas de aprendizaje colaborativo.
- Interfaz intuitiva y fácil de usar tanto para profesores como para alumnos.

Desventajas:

- Algunas funcionalidades avanzadas pueden estar limitadas a suscripciones de pago.

En comparación con estas alternativas, EduVibe ofrece una plataforma centrada en proporcionar herramientas específicas para la gestión de clases y la interacción entre profesores y estudiantes.

Además, comparado con otras opciones, EduVibe destaca por su velocidad, lo que significa que los usuarios pueden hacer las cosas más rápido y sin demoras molestas. También puede adaptarse fácilmente a diferentes necesidades de las escuelas y colegios, lo que la hace perfecta para muchos tipos de instituciones educativas. Esto la convierte en una excelente opción para quienes buscan una herramienta que sea fácil de usar y se ajuste a sus necesidades específicas en el mundo de la educación en línea.



4. JUSTIFICACIÓN DEL PROYECTO

Después de años usando Moodle, me di cuenta de que tenía muchas limitaciones. No siempre era fácil de usar y no se adaptaba bien a lo que necesitaban tanto los profesores como los alumnos. Esto me llevó a querer crear algo mejor, algo que realmente cumpliera con las expectativas de los estudiantes.

La idea de EduVibe surgió de la necesidad de superar las barreras y dificultades que experimentaba con Moodle y otras plataformas similares. Quería algo más moderno, que fuera más fácil de usar y que se ajustara mejor a las necesidades de la educación en línea.



5. STACK TECNOLÓGICO

Para el desarrollo de EduVibe se han elegido las siguientes tecnologías:

Backend:

- Spring Boot

Frontend:

- Angular

Base de Datos:

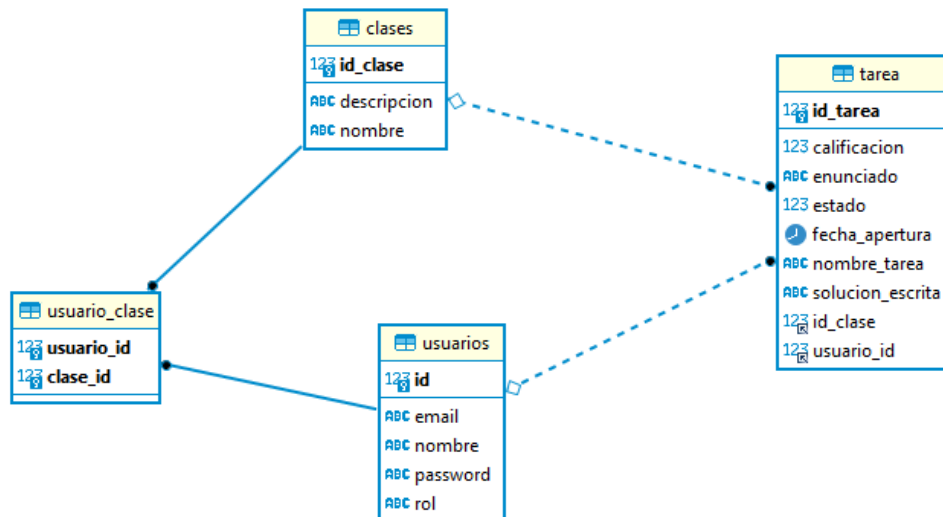
- MySQL

- Justificación:

He seleccionado estas tecnologías porque son las que mejor conozco y domino. Esto me permite desarrollar la aplicación de manera eficiente y garantizar un producto final de alta calidad.

Al estar familiarizado con estas herramientas, puedo aprovechar al máximo sus características y funcionalidades para crear una plataforma educativa interactiva que cumpla con los requisitos del proyecto. Además, al utilizar tecnologías con las que me siento cómodo, puedo resolver problemas de manera más rápida y efectiva durante el proceso de desarrollo.

6. ESQUEMA E/R



1. Entidad: usuarios

- id (Primary Key): Identificador único para cada usuario.
- email: Dirección de correo electrónico del usuario.
- nombre: Nombre del usuario.
- password: Contraseña del usuario.
- rol: Rol del usuario (alumno, profesor, admin.).

2. Entidad: clases

- id_clase (Primary Key): Identificador único para cada clase.
- descripcion: Descripción de la clase.
- nombre: Nombre de la clase.

3. Entidad: tarea

- id_tarea (Primary Key): Identificador único para cada tarea.
- calificacion: Calificación obtenida por el usuario en la tarea.
- enunciado: Enunciado de la tarea.
- estado: Estado de la tarea (pendiente, calificar).
- fecha_apertura: Fecha en que la tarea fue creada.
- nombre_tarea: Nombre de la tarea.
- solucion_escrita: Solución escrita proporcionada por el alumno para la tarea.
- id_clase (Foreign Key): Identificador de la clase a la que pertenece la tarea.
- usuario_id (Foreign Key): Identificador del usuario al que se asignó la tarea.

4. Entidad: usuario_clase

- usuario_id (Foreign Key): Identificador del usuario que está inscrito en la clase.
- clase_id (Foreign Key): Identificador de la clase en la que el usuario está inscrito.

5. Relaciones entre Entidades

- usuarios y clases están relacionados mediante la entidad usuario_clase, que actúa como una tabla de unión para representar una relación de muchos a muchos entre usuarios y clases.
- tarea tiene una relación de muchos a uno con clases mediante el campo id_clase.
- tarea también tiene una relación de muchos a uno con usuarios mediante el campo usuario_id.

6. Descripción de las Relaciones

- usuario_clase: Esta entidad intermedia se utiliza para gestionar la relación de muchos a muchos entre usuarios y clases. Cada registro en esta tabla representa la inscripción de un usuario en una clase.
- tarea: Cada tarea está asociada a una clase y a un usuario específico, lo que permite asignar tareas a los estudiantes y mantener un registro de sus calificaciones y soluciones.

7. PROTOTIPADO APLICACIÓN WEB

- Login de la aplicación:

EduVibe | Servicio De Autenticación

INICIAR SESIÓN

Nombre de usuario:
Escriba su nombre de usuario...

Contraseña:
Escriba su contraseña...

☐ Mostrar Contraseña

INICIAR SESIÓN

[¿Ha olvidado la contraseña?](#)

EduVibe - Plataforma Educativa

[Información legal](#) | [Política de privacidad](#)

- Creación de usuario (Admin):

EduVibe | Inicio Clases Calendario **Registro**

CREACIÓN DE USUARIO

Nombre de usuario:
Escriba su nombre de usuario...

Contraseña:
Escriba su contraseña...

☐ Mostrar Contraseña

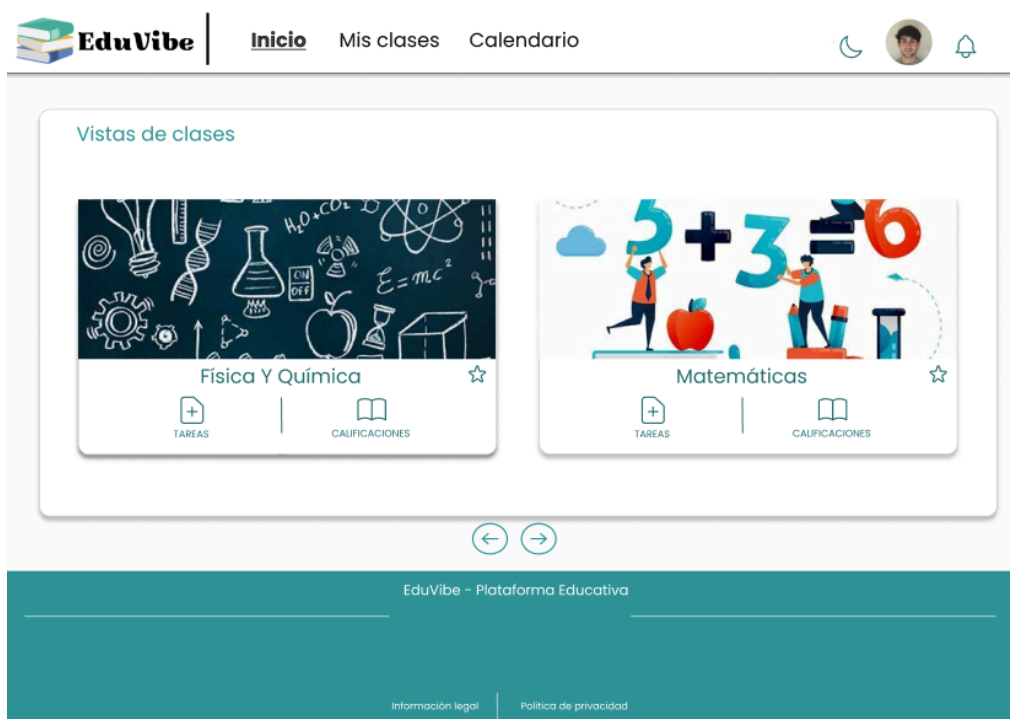
Tipo de Usuario:
☐ Administrador
☐ Profesor
☐ Alumno

CREAR USUARIO

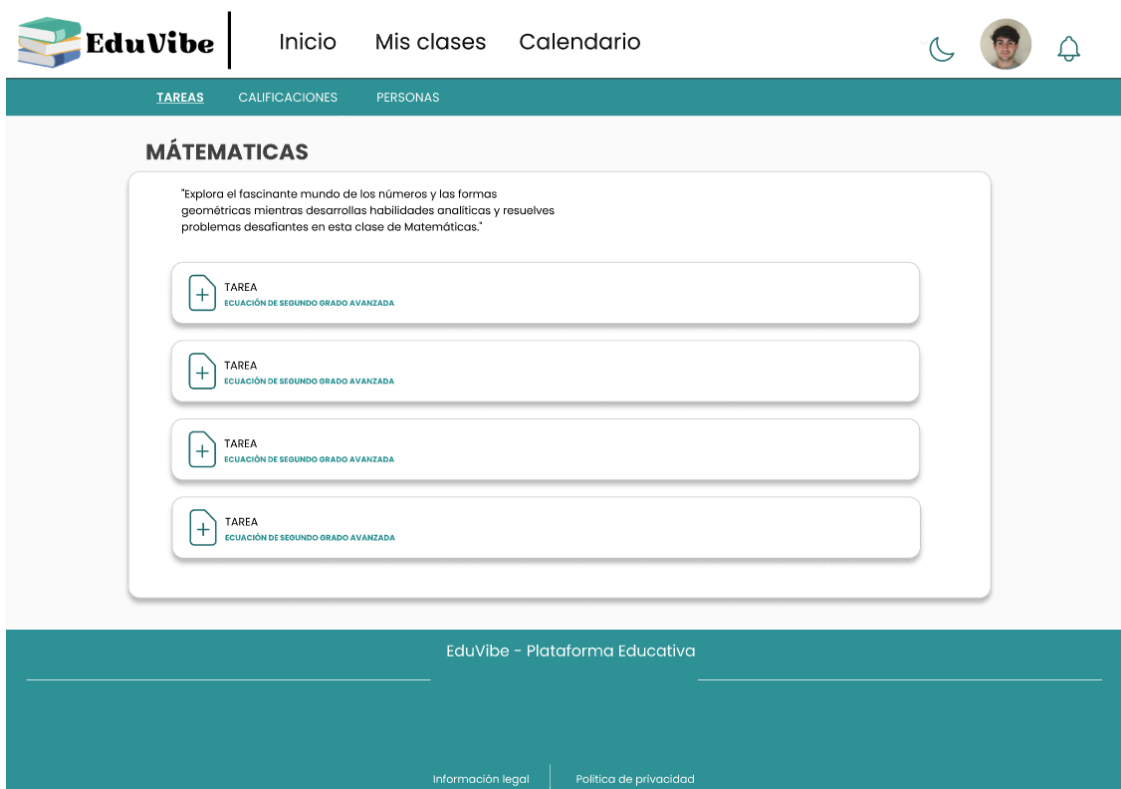
EduVibe - Plataforma Educativa

[Información legal](#) | [Política de privacidad](#)

- Inicio de la aplicación:




- Vista de las tareas de una clase (Alumno)









- Vista de tarea:



[Inicio](#) [Mis clases](#) [Calendario](#)



[TAREAS](#) [CALIFICACIONES](#) [PERSONAS](#)



TAREA ALGEBRA

Resuelve la siguiente ecuación cuadrática utilizando el método de factorización: $x^2 - 5x + 6 = 0$

2

-5x+6=0. Indica los valores de x que satisfacen la ecuación.


Agregar entrega

FECHA DE ENTREGA:	01/10/2024
ESTADO	ENTREGADO
CALIFICACIÓN	9.50




EduVibe - Plataforma Educativa

[Información legal](#) [Política de privacidad](#)

- Creación de tarea (Admin, Profesor)



[Inicio](#) [Mis clases](#) [Calendario](#)



[TAREAS](#) [CALIFICACIONES](#) [PERSONAS](#)

CREAR TAREA

Nombre:

Descripción:

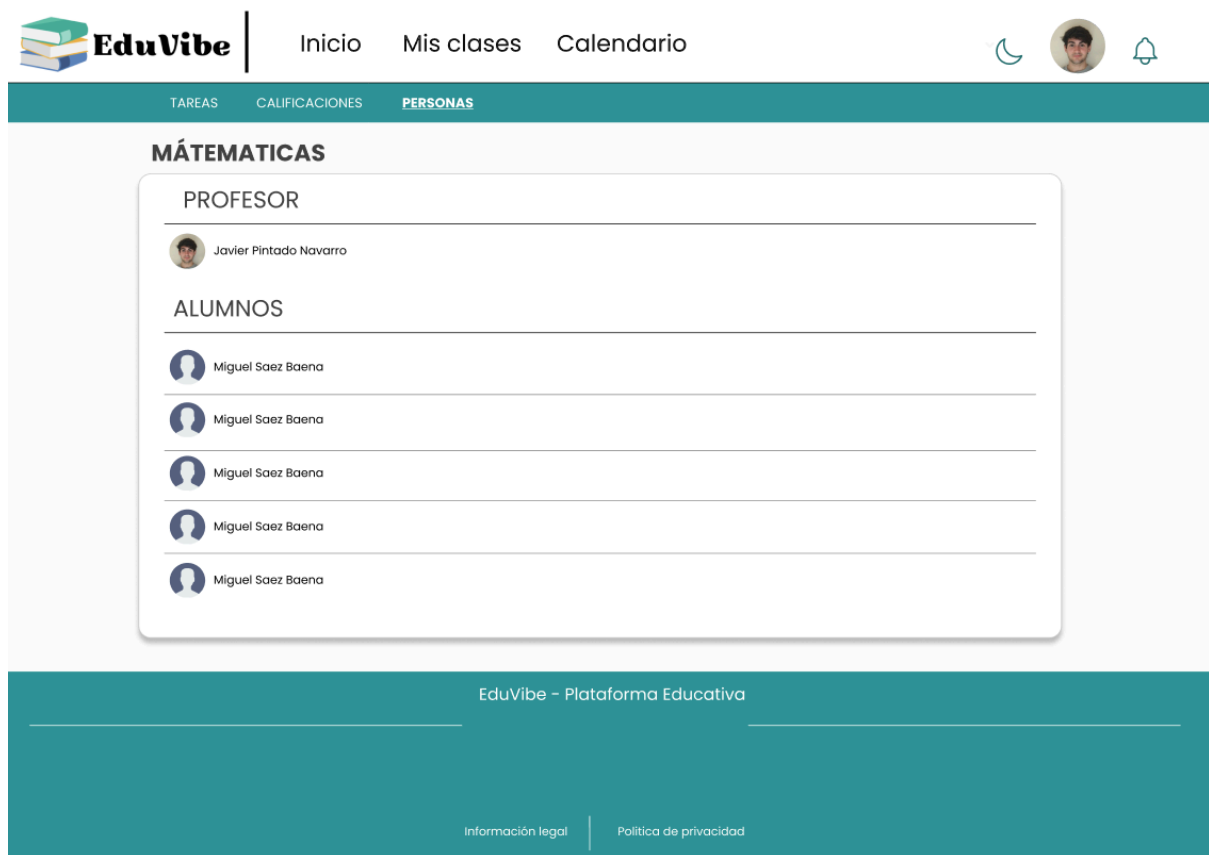
CREAR

EduVibe - Plataforma Educativa

[Información legal](#) [Política de privacidad](#)



- Lista de personas de una clase



Las demás vistas del prototipado puedes encontrarla en el figma de la aplicación, aquí el enlace público de este:

Enlace público de figma de EduVibe:

<https://www.figma.com/design/iTOjmqIpDJSsbTVeeCOtIH/EduVibe-Figma?node-id=38-3361&t=LOMlvrK5FbDHDOKU-0>

8. DEFINICIÓN API REST

- Tabla con todas las API desarrolladas para la aplicación, junto con la información relevante:

Endpoint	Tipo	Parámetros	Descripción
/clases/crear	POST	ClaseDto	Crea una nueva clase.
/clases/{id}	GET	id (PathVariable)	Obtiene una clase por su ID.
/clases/editar/{id}	PUT	id (PathVariable), ClaseDto	Edita una clase existente.
/clases/inscribir	POST	email (RequestParam), idClase (RequestParam)	Inscribir un usuario en una clase.
/clases/{id}/eliminar-usuario	POST	email (RequestParam), idClase (RequestParam)	Elimina a un usuario de una clase.
/clases/eliminar/{id}	DELETE	id (PathVariable)	Elimina una clase por su ID.
/clases/todos	GET	Authorization (RequestHeader)	Obtiene todas las clases.
/clases/usuario/{idUser}	GET	idUser (PathVariable)	Obtiene las clases asociadas a un usuario por su ID.
/registeruser	POST	RegistroUserDto	Registra un nuevo usuario.
/loginuser	POST	UserLogin	Autentica a un usuario y genera un token JWT.
/validate	GET	Authorization (RequestHeader)	Valida un token JWT.

/renew	GET	Authorization (RequestHeader)	Renueva un token JWT.
/tareas/crear	POST	TareaDTO	Crea una nueva tarea.
/tareas/{id}	GET	id (PathVariable)	Obtiene una tarea por su ID.
/tareas/editar/{id}	PUT	id (PathVariable), TareaDTO	Edita una tarea existente.
/tareas/eliminar/{id}	DELETE	id (PathVariable)	Elimina una tarea por su ID.
/tareas/todos	GET	-	Obtiene todas las tareas
/tareas/{id}/archivo	POST	id (PathVariable), archivo (RequestParam)	Agrega un archivo adjunto a una tarea.
/tareas/{id}/archivo/{ indice}	GET	id (PathVariable), indice (PathVariable)	Descarga un archivo adjunto de una tarea por su índice.
/tareas/clase/{idClas e}	GET	idClase (PathVariable)	Obtiene las tareas asociadas a una clase por su ID.
/tareas/clase/user/{c laselId}	GET	claselId (PathVariable)	Obtiene las tareas asociadas a una clase para el usuario actual.
/user/existeEmail	GET	email (RequestParam)	Verifica si un email ya está registrado.
/user/{id}	GET	id (PathVariable)	Obtiene un usuario por su ID
/user/changePassw ord	PUT	email (RequestParam), newPassword (RequestParam)	Cambia la contraseña de un usuario.



/usuarios	GET	-	Lista todos los usuarios registrados.
/usuarios/clase/{idClase}	GET	idClase (PathVariable)	Obtiene los usuarios asociados a una clase por su ID.

Documentación de postman de todos los endpoints:

Enlace: <https://documenter.getpostman.com/view/32189097/2sA3XMiNXD>

9. MANUAL DE DESPLIEGUE (DOCKER)

Requisitos Previos:

- Tener acceso a un entorno con Docker instalado.
- Conexión a internet para la descarga de imágenes y dependencias.

Paso 1: Preparación del Entorno

1.1. Asegúrate de que Docker esté instalado en tu entorno. Puedes instalar Docker siguiendo las instrucciones proporcionadas en su documentación oficial.

1.2. Clona el repositorio de tu aplicación en el servidor utilizando Git

Paso 2: Construcción y Despliegue del Backend

2.1. Crea el archivo Dockerfile para el backend en la raíz del directorio del proyecto. A continuación, agrega lo siguiente:

```
# Etapa de construcción

FROM maven:3-openjdk-17 as builder

COPY src /usr/src/app/src

COPY pom.xml /usr/src/app

RUN mvn -f /usr/src/app/pom.xml clean package -DskipTests


# Etapa de producción

FROM openjdk:17-alpine


# Variables de entorno

ENV BBDD_HOST="db"

ENV BBDD_NAME="eduvibe"

ENV APP_PORT=9090

ENV LOG_LEVEL="INFO"
```



```
ENV DLL_AUTO="update"

# Volumen para datos temporales y archivos multimedia

VOLUME /tmp

VOLUME /mediafiles

RUN mkdir -p /mediafiles

# Puerto expuesto

EXPOSE 9090

# Copiar el archivo JAR construido de la etapa de construcción

COPY --from=builder
/usr/src/app/target/EduvibeBackend-0.0.1-SNAPSHOT.jar /app/app.jar

# Establecer directorio de trabajo

# Comando de inicio

ENTRYPOINT ["java", "-jar", "/app/app.jar"]
```



2.2. Crea el archivo docker-compose.yml en la raíz del directorio del proyecto. Agrega lo siguiente:

```
version: '3.1'

services:

  db:

    image: mysql:latest

    container_name: bbdd-eduvibe

    restart: unless-stopped

    ports:

      - "3306:3306"

    environment:

      MYSQL_ROOT_PASSWORD: root

      MYSQL_DATABASE: eduvibe

  eduvibe_api:

    build:

      context: ./src-api/EduvibeBackend

      dockerfile: ./Dockerfile

    container_name: api-eduvibe

    restart: unless-stopped

    ports:

      - "9090:9090"

    environment:

      BBDD_HOST: db
```



```
BBDD_NAME: eduvibe

DATABASE_USERNAME: root

DATABASE_PASSWORD: root

APP_PORT: 9090

LOG_LEVEL: DEBUG

DLL_AUTO: create

volumes:

  - ./src/main/resources:/app/resources

depends_on:

  - db

eduvibe_front:

  build:

    context: ./src-frontend/EduVibeFront

    dockerfile: ./Dockerfile

  container_name: eduvibe_front

  restart: unless-stopped

  ports:

    - "80:4200"

  depends_on:

    - eduvibe_api
```



Paso 3: Construcción y Despliegue del Frontend

3.1. Crea el archivo Dockerfile para el frontend en el directorio del frontend de tu proyecto. A continuación, agrega el siguiente contenido:

```
# Imagen

FROM node:20.9.0 as build

# Directorio

WORKDIR /app

# Copia los archivos menos el dockerfile

COPY . .

# Instala las dependencias

RUN npm install

# Construye la aplicación Angular

RUN npm run build --prod

# Usa una imagen de Nginx para servir la aplicación Angular

FROM nginx:stable-alpine

# Copia los archivos

COPY --from=build /app/dist/edu-vibe-front/browser
/usr/share/nginx/html

EXPOSE 4200

# Comando para iniciar Nginx

CMD ["nginx", "-g", "daemon off;"]
```

3.2. Crea un archivo llamado nginx.conf en el mismo directorio que tu Dockerfile frontend. Agrega el siguiente contenido:

```
# Configuración Global

worker_processes 1;

events {

    worker_connections 1024;

}

http {

    include mime.types;

    default_type application/octet-stream;

    # Bloque del servidor para la aplicación Angular

    server {

        listen 80;

        server_name localhost;

        location / {

            root /usr/share/nginx/html;

            index index.html index.htm;

            try_files $uri $uri/ /index.html;

        }

    }

}
```




Paso 4: Ejecución de la Aplicación

4.1. En el directorio raíz del proyecto, ejecuta el siguiente comando para iniciar los contenedores Docker:

```
docker-compose up -d
```

Paso 5: Acceso a la Aplicación

5.1. Una vez que ambos contenedores estén en ejecución, podrás acceder a tu aplicación EduVibe a través de tu navegador web.

5.2. Abre tu navegador web y visita la dirección IP.

10. CONCLUSIÓN

Desarrollar esta aplicación me ha permitido aprender enormemente. He adquirido habilidades en la resolución de errores y he dedicado mucho tiempo a replantear y ajustar tanto el modelo de datos como la lógica de la aplicación.

Crear la base de datos fue un desafío, ya que al principio no tenía claro el modelo de datos. Después, desarrollé un backend básico con los métodos CRUD estándar para cada entidad. Tuve que modificar el sistema de seguridad varias veces hasta conseguir una versión funcional y fácil de configurar. Empecé implementando el inicio de sesión y el registro, y luego fui añadiendo las funcionalidades básicas de la aplicación, alternando entre el desarrollo del frontend y el backend.

Al final, la aplicación se parece mucho al prototipo, lo que me tiene bastante contento. Este proyecto ha sido crucial para darme cuenta de todo lo que implica crear una aplicación web.

Posibles Mejoras

Mejorar la Apariencia:

- Hacer la interfaz más atractiva y fácil de usar en cualquier dispositivo.

Añadir Funciones:

- Incluir un chat para mejorar la comunicación entre profesores y alumnos.
- Implementar notificaciones para avisar a los usuarios sobre nuevas tareas o cambios.
- Calendario con eventos

Pulir La lógica de la aplicació:

- Añadir una tabla más entre usuario y tareas, para que así para una tarea pueda tener más de un usuario y viceversa



Viabilidad de Puesta en Marcha Real

- Se podría usar EduVibe en una clase o escuela pequeña para obtener opiniones y hacer mejoras.
- Ofrecer formación a los administradores, profesores y alumnos para que puedan usar la plataforma de manera eficiente.
- Establecer un sistema de soporte técnico para resolver problemas y mantener la plataforma actualizada.
- Desarrollar una estrategia para dar a conocer EduVibe y sus ventajas.

11. ANEXOS

Dependencias de Spring Boot utilizadas en EduVibe:

1. *Spring Boot Starter Dependencies*

spring-boot-starter-data-jpa: Proporciona dependencias para trabajar con JPA (Java Persistence API), facilitando la configuración y el uso de bases de datos relacionales en aplicaciones Spring Boot.

spring-boot-starter-security: Ofrece herramientas y configuraciones para la seguridad de la aplicación, como autenticación y autorización.

spring-boot-starter-web: Proporciona las dependencias necesarias para crear aplicaciones web utilizando Spring MVC.

spring-boot-starter-mail: Facilita el envío de correos electrónicos desde la aplicación.

spring-boot-devtools: Herramientas de desarrollo que agilizan el ciclo de desarrollo, como la reinicialización automática de la aplicación.

2. *Additional Dependencies*

commons-lang3: Biblioteca útil que proporciona funcionalidades adicionales para trabajar con cadenas, matrices, números, etc.

lombok: Biblioteca que simplifica el desarrollo eliminando la necesidad de escribir código boilerplate, como getters, setters y constructores.

jjwt-api, jjwt-impl, jjwt-jackson: Implementación de JSON Web Tokens (JWT) que se utiliza para la autenticación y la creación de tokens seguros.

javafaker: Biblioteca para generar datos ficticios de manera realista, útil para pruebas y desarrollo.

3. *Spring Boot Test Dependencies*

spring-boot-starter-test: Proporciona dependencias para realizar pruebas unitarias y de integración en aplicaciones Spring Boot.

spring-security-test: Contiene clases y utilidades para realizar pruebas de seguridad en aplicaciones Spring.



4. Validation Dependencies

hibernate-validator: Implementación de las especificaciones de validación de Bean Validation, que permite definir restricciones de validación en los objetos del dominio.

jakarta.el: Implementación de Expression Language (EL), que se utiliza para la evaluación de expresiones en JSP, JSF y otros entornos.

5. WebFlux Starter Dependency

spring-boot-starter-webflux: Proporciona dependencias para desarrollar aplicaciones reactivas utilizando Spring WebFlux.

6. MySQL Connector Dependency

mysql-connector-java: Conector JDBC para MySQL, que permite a la aplicación conectarse y comunicarse con una base de datos MySQL.

7. H2 Database Dependency for Testing

h2: Base de datos en memoria que se utiliza para pruebas, permitiendo realizar pruebas de integración sin necesidad de configurar una base de datos externa.

8. ModelMapper Dependency

modelmapper: Biblioteca que facilita el mapeo de objetos de un tipo a otro, simplificando la conversión entre diferentes modelos de datos.



Dependencias de Angular utilizadas en EduVibe

1. Dependencias Principales

@angular/animations: Proporciona soporte para animaciones en Angular.

@angular/common: Contiene los módulos comunes de Angular, como el enlace de datos y las directivas básicas.

@angular/compiler: Proporciona el compilador de Angular, necesario para compilar los componentes de la aplicación.

@angular/core: Contiene los bloques de construcción básicos de Angular, como componentes, servicios, directivas, etc.

@angular/forms: Proporciona funcionalidades para trabajar con formularios en Angular.

@angular/platform-browser: Proporciona la implementación de Angular para el navegador web.

@angular/platform-browser-dynamic: Proporciona herramientas para cargar la aplicación Angular dinámicamente en un navegador web.

@angular/platform-server: Proporciona la implementación de Angular para el lado del servidor.

@angular/router: Proporciona enrutamiento para la aplicación Angular.

@angular/ssr: Proporciona soporte para el renderizado del lado del servidor (SSR) en Angular.

@fortawesome/angular-fontawesome: Integración de Font Awesome con Angular.

@fortawesome/fontawesome-svg-core: Core de Font Awesome para SVG.

@fortawesome/free-solid-svg-icons: Iconos sólidos gratuitos de Font Awesome.

@fullcalendar/angular, @fullcalendar/core, @fullcalendar/daygrid: Integración de FullCalendar con Angular.

@popperjs/core: Utilidades para posicionar elementos emergentes (popovers) de forma dinámica.



bootstrap: Framework de CSS para diseño responsivo.

express: Framework web de Node.js para el servidor backend.

file-saver: Biblioteca para guardar archivos desde el navegador.

jwt-decode: Decodificador de JSON Web Tokens (JWT).

ngx-pagination: Paginación para Angular.

rxjs: Biblioteca para programación reactiva en JavaScript.

sweetalert2: Biblioteca para mostrar mensajes emergentes (popups) personalizados.

tslib: Biblioteca para soporte de TypeScript.

zone.js: Biblioteca para la gestión de zonas en Angular.

2. Dependencias de Desarrollo

@angular-devkit/build-angular, @angular/cli, @angular/compiler-cli: Herramientas para compilar y construir aplicaciones Angular.

@types/express, @types/file-saver, @types/jasmine, @types/node: Definiciones de tipos TypeScript para librerías externas.

jasmine-core, karma, karma-chrome-launcher, karma-coverage, karma-jasmine, karma-jasmine-html-reporter: Herramientas de prueba para Jasmine y Karma.

typescript: Compilador TypeScript para el desarrollo Angular.