Actividad 8: Informe Final Entregado

EduTecnIA busca desarrollar una plataforma digital que adapte los cursos y contenidos al perfil y necesidades de cada usuario, a continuación voy a explicar la información que he recopilado, además del lenguaje que se va a implementar, la IDE que vamos a usar para el desarrollo y el gestor de versiones.

1. **Selección del Lenguaje y Paradigma de Programación**

   Para desarrollar esta plataforma el lenguaje empleado será java orientado a objetos, ya que su principal característica, la herencia con clases que heredan las características de otras clases gracias al método “super”, nos permite realizar un código mucho más estructurado, teniendo como clases al usuario, y que ésta herede a otra subclase.
   Otra de las razones por la que java orientado a objetos sería la mejor decisión es su seguridad, ya que el Java incluye características de seguridad integradas, además tiene herramientas para automatizar la gestión de memoria, y maneja un sistema de excepciones sobresaliente, además en una plataforma como EduTechIaA, la protección de datos de los usuarios es lo primordial.
   Tambien gracias a las maquina virtual de Java(JVM), se asegura que las aplicaciones desarrolladas en este lenguaje sean portables, y compatibles entre diferentes plataformas.
   Por último hay que nombrar a la extensa comunidad de desarrolladores de Java, que proporcionan acceso a recursos, documentación y soporte continuo.

   Y aunque Java sea el lenguaje ideal, no podemos dejar de lado un par de inconvenientes que pueden surgir a la hora de desarrollar el codigo.
   Lo primero sería su rendimiento, ya que al ser un lenguaje que corre sobre una máquina virtual, introduce una capa adicional de abstracción, que puede causar errores de rendimiento en comparación de otros lenguajes como C++.
   Y lo segundo sería su tiempo de desarrollo, ya que al tener que definir todas sus clases y emplear una sintaxis detallada y precisa hace que el tiempo de desarrollo sea mucho mayor, en comparación con otros lenguajes.

   En conclusión, hay que tener en cuenta todas las ventajas que nos va a ofrecer java a lo largo de todo el diseño y desarrollo de la plataforma EduTechIA, pero más en cuenta las desventajas, y si la plataforma se puede permitir un poco menos de rendimiento y el equipo dispone del suficiente tiempo para desarrollarla utilizando Java.

2. **Diseño de la Arquitectura del Software**

   En este apartado he diseñado una arquitectura ideal para la plataforma de EduTechIA.

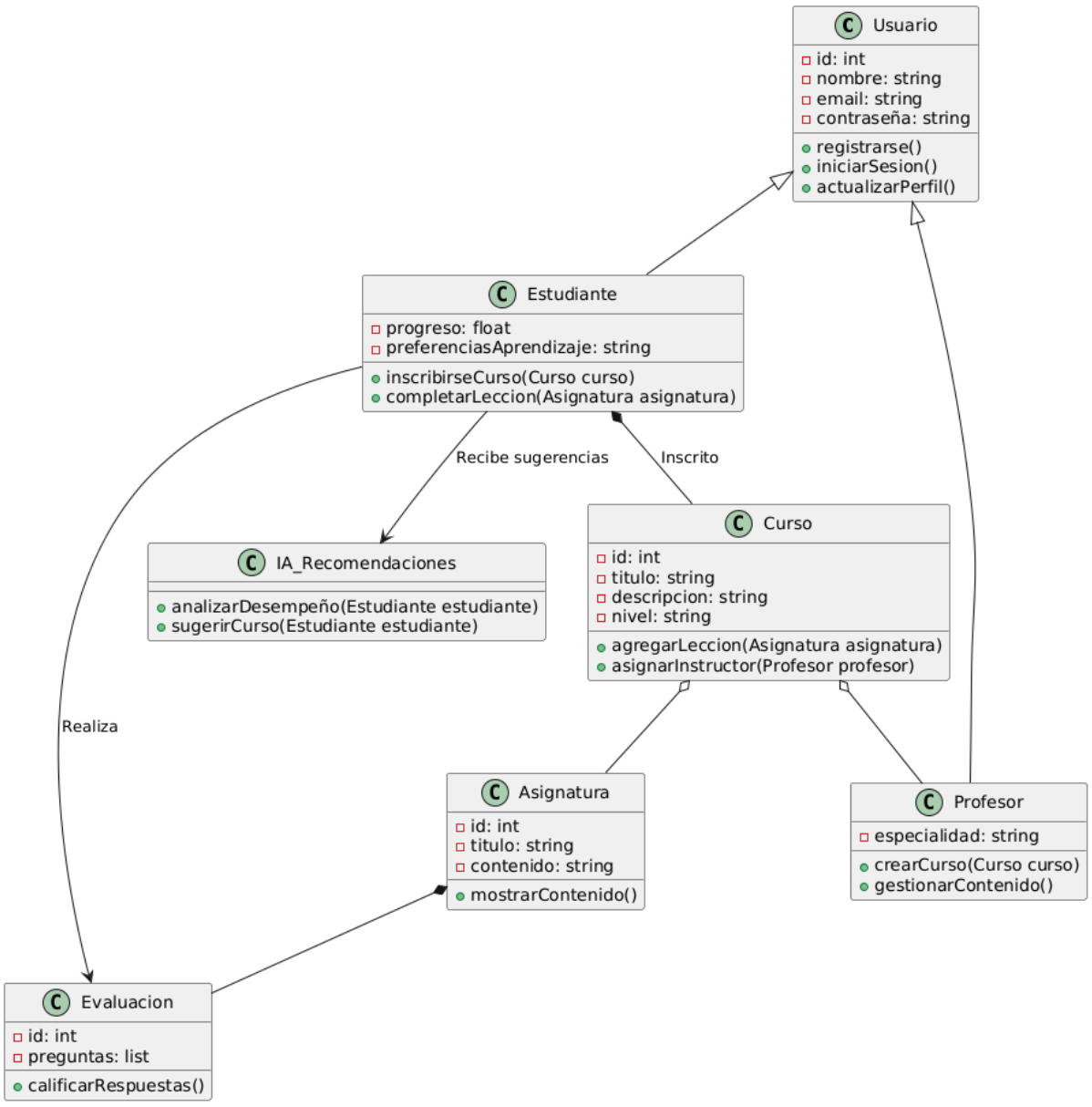

Imagen del diagrama de clases hecho en plantUML

En este diagrama he definido todas las clases necesarias para el desarrollo de la plataforma, siendo la clase principal usuario, que heredan la clase profesor y la clase estudiante, cada uno con sus propios atributos, pero mantienen en común los de la clase usuario.
En la clase curso se asigna a un profesor, y se inscribe un alumno.
Además en la clase evaluación se guardan las preguntas del estudiante, se califican y se envían a la clase asignatura.
Y en la clase IA Recomendaciones se analiza el desempeño del estudiante, y se le sugiere un curso.

En conclusión, el diagrama propuesto establece una estructura clara y organizada para el desarrollo de la plataforma, con una jerarquía de clases bien definida. La clase principal "Usuario" permite compartir atributos comunes entre "Profesor" y "Estudiante", optimizando la reutilización de código. Además, la relación entre

"Curso", "Evaluación" y "Asignatura" facilita la gestión académica y el seguimiento del aprendizaje. Finalmente, la implementación de "IA Recomendaciones" añade un valor adicional al sistema, personalizando la experiencia del estudiante mediante el análisis de su desempeño y la sugerencia de cursos adecuados.

3. Selección de Herramientas y Configuración del Entorno de Desarrollo

Para el desarrollo de la plataforma de EduTechIA, se utilizará el IDE IntellIJ IDEA, esto debido a su gran potencia, su facilidad a la hora de usarlo, y sus capacidades avanzadas para el desarrollo de aplicaciones. Gracias a este IDE se podrá garantizar la calidad del código desde el inicio del proyecto.
Su autocompletado inteligente del código será de ayuda a la hora de escribir de forma rápida y fluida, y reducirá posibles errores de sintaxis.
Asimismo cuenta con herramientas integradas, como las pruebas unitarias, generación de código como plantilla, y depuración.
Este IDE soporta múltiples frameworks y tecnologías, que permitirán una integración eficiente con bases de datos y servicios en la nube.
Y lo más importante, es que cuenta con una integración nativa con Git/GitHub, lo que facilita el trabajo en equipo, al permitir clonar, hacer commits, crear ramas y realizar pull requests sin necesidad de salir del IDE.

Para el control de versiones se utilizará Git y GitHub como plataforma de alojamiento y colaboración.
Al utilizar Git/GitHub, nos va a permitir tener un seguimiento detallado del código, lo que facilitará la gestión del desarrollo, también Git nos permitirá tener un respaldo seguro, que evite posibles pérdidas de información.

4. Planificación del Pipeline de Integración y Entrega Continua (CI/CD)

El desarrollo de la MVP requiere una arquitectura robusta, un pipeline de Integración y Entrega Continua (CI/CD) eficiente y un protocolo de seguridad bien definido. Este informe detalla cada una de estas fases, asegurando un desarrollo escalable, seguro y automatizado.

Lo primero será definir las Herramientas que se van a emplear a la hora de desarrollar el pipeline.

- Repositorio de código: GitHub
  GitHub será la plataforma a la que se subirá todo el código, y el control de versiones se hará con Git.

- Contenedorización: Docker
  Se utilizará docker para empaquetar la aplicación en contenedores.

- Gestión de dependencias: Marven/Gradle
  Automatiza compilaciones y gestión de librerias.

- Orquestación: Kubernetes
  Kubernetes administra los contenedores de producción.

- Pruebas unitarias: Junit
  Automatiza las pruebas de unidad del MVP.

- Análisis de calidad: SonarQube
  Detectará errores en el código como bugs, vulnerabilidades y estilo.

A continuación hay que definir las etapas del Pipeline CI/CD.
En el desarrollo y control de versiones los desarrolladores deberán subir a un repositorio Git con ramas de desarrollo(develop) y producción(main).
Para la integración continua lo primero será la activación con cada push o Pull Request en GitHub, para la compilación se construirá el proyecto con Maven/Gradle, también serán necesarias tanto pruebas unitarias, como de integración, y si todo es correcto se almacenará el JAR/WAR en un repositorio seguro (Cabe mencionar que si el código falla en alguna parte se avisará a los desarrolladores).

Para la construcción de la aplicación se generará una imagen docker con la versión del código y se almacenará en un docker registry(Docker Hub).

En cuanto al despliegue en el entorno Staging, se hará de forma automática con Docker, de forma que se despliegue la aplicación en un entorno de pruebas de aceptación, después pase a pruebas end-to-end (Cypress, Selenium), y por último a pruebas de carga y seguridad.

Si las pruebas Staging resultan exitosas, un revisor lo aprueba y pasa al despliegue de producción.

En el despliegue a producción el objetivo es automatizar el despliegue del software en entornos de staging y producción, asegurando la estabilidad y la rapidez.
Para empezar, se activa con una compilación exitosa en CI o con un merge a main, después se descarga el JAR/WAR, y se genera una imagen docker del servicio actualizado, en el despliegue staging se realizan pruebas funcionales, y en el despliegue de producción se realizarán baja aprobación manual para evitar fallos. Después del despliegue se ejecutan ciertas pruebas de sanity check y monitoreo, y para finalizar, si se detectan fallos, se activará una reversión automática a la versión estable anterior.

Todo el pipeline descrito, nos facilitará mucho la producción de esta plataforma, ya que se basa en una arquitectura sólida que integrará herramientas y procesos automatizados para así garantizar la eficiencia, seguridad y escalabilidad. Desde la gestión del código en GitHub hasta la orquestación con Kubernetes, cada etapa del pipeline CI/CD ha sido diseñada para optimizar la integración y entrega continua. La implementación de pruebas en múltiples niveles, junto con un monitoreo constante, asegura la calidad y estabilidad del software. Finalmente, el despliegue automatizado, con mecanismos de aprobación y reversión, permite minimizar riesgos y garantizar una transición segura a producción.

5. Desarrollo de un Protocolo de Seguridad en el Desarrollo

En este punto se busca garantizar la seguridad del MVP, este protocolo establecerá medidas y mejorará las prácticas en cada fase del desarrollo, minimizando la vulnerabilidad y protegiendo los datos

Para diseñar un protocolo de seguridad, lo primero será planificarlo y diseñarlo, para ello, se realizará un análisis de riesgos en el que se identificarán los activos más vulnerables del sistema(datos de usuarios, algoritmos de IA, infraestructura, etc.), una evaluación de amenazas, en la que se identifiquen posibles ataques, y una clasificación de riesgos según su impacto y probabilidad.
Asimismo debe tener un diseño Seguro, haciendo que cada componente tenga solo los permisos necesarios, estableciendo todas las opciones de seguridad por defecto, utilizar una arquitectura de cero confianza(Zero Trust) donde ningún usuario o sistema sea confiable por defecto, y diseñar un mecanismo para encriptar datos sensibles en reposo y tránsito.

La fase del desarrollo se dividirá en dos, por una parte las prácticas de Codificación seguras, en las que se hará un uso de librerías y frameworks actualizados y de buena reputación, además de evitar mostrar información sensible en mensajes de error.
Por otra parte se encargaran de revisar el código y hacer pruebas, aquí se realizarán revisiones para identificar vulnerabilidades y malas prácticas, y pruebas de penetración en etapas tempranas para identificar vulnerabilidades.

A continuación en la fase de implementación se realizarán dos cosas, una configuración segura del entorno, asegurándose la configuración de servidores y servicios, además de implementar sistemas de monitoreo y logging para detectar actividades sospechosas.
Y un despliegue seguro, asegurándose de que el pipeline de CI/CD esté configurado de forma segura para impedir que se inserte código malicioso

Por último se realizará una fase de operación y mantenimiento en la que se tendrá un monitoreo continuo con detección y prevención de intrusos (IDS, IPS), y con un sistema de respuestas a incidentes, por si en caso de brecha actuar con rapidez. Además contará con actualizaciones y parches, que se aplicaran de forma continua, y que mantenga las dependencias del sistema siempre actualizadas.

En conclusión, el desarrollo de un protocolo de seguridad sólido es esencial para garantizar la protección del MVP en todas sus fases. Desde la planificación y el análisis de riesgos hasta la implementación y mantenimiento, cada etapa está diseñada para minimizar vulnerabilidades y reforzar la seguridad del sistema. La adopción de una arquitectura de Zero Trust, el uso de prácticas de codificación seguras, y la integración de monitoreo continuo y respuesta a incidentes aseguran un entorno robusto y confiable. Con este enfoque, se protege la integridad de los datos, se previenen ataques y se garantiza la estabilidad del software a largo plazo.

6. Estrategia de Gestión y Optimización de Entornos

Para gestionar los entornos de desarrollo, preproducción y producción de la plataforma de EduTech IA, utilizaremos una arquitectura optimizada basada en contenedores, automatización CI/CD y buenas prácticas de seguridad y escalabilidad en la nube.

En la fase de desarrollo cada desarrollador trabajará en un entorno local con Docker. Se usará Spring Boot para el backend, junto con PostgreSQL como base de datos. Se adoptará el flujo GitHub Flow, con ramas feature, desarrollo y main, asegurando un código limpio con pruebas unitarias y pre-commit hooks

En la fase de producción se validaran cambios antes de que sean enviados a producción, además se implementara una arquitectura parecida a la de producción permitiendo la detección de errores antes de que los sufran los usuarios finales.

- Se desplegará automáticamente mediante GitHub Actions, donde cada push a develop activará pruebas automatizadas y un despliegue en preproducción.
- Se utilizarán contenedores Docker orquestados con Kubernetes, asegurando una infraestructura replicable y escalable.
- La base de datos en este entorno será una réplica de la producción, pero con datos anonimizados para cumplir con normativas de seguridad.

Este entorno lo utilizaran los usuarios finales y deberá garantizar una alta disponibilidad, gran seguridad, y un buen rendimiento, para ello, se utilizarán los siguientes métodos.

- La base de datos PostgreSQL será gestionada por un servicio cloud para garantizar redundancia y backups automáticos.
- Se utilizarán balanceadores de carga para distribuir el tráfico y optimizar el rendimiento del sistema.
- Para la autenticación y autorización de usuarios, se usará OAuth2, asegurando sesiones seguras y control de accesos adecuado.
- Se implementará cifrado en tránsito y en reposo con TLS y almacenamiento cifrado para proteger los datos sensibles.
- Se ejecutarán backups automáticos y estrategias de disaster recovery para garantizar la continuidad del servicio en caso de fallos críticos.

En conclusión, la gestión de los entornos de desarrollo, preproducción y producción de la plataforma EduTech IA se basa en una arquitectura optimizada que prioriza la escalabilidad, seguridad y automatización. Mediante el uso de Docker y Kubernetes, junto con un flujo de trabajo eficiente en GitHub Actions, se garantiza una integración y entrega continua confiable. La implementación de buenas prácticas de seguridad, como OAuth2 para autenticación, cifrado de datos y estrategias de recuperación ante desastres, asegura la protección y disponibilidad del sistema. Con este enfoque, se optimiza el desarrollo y despliegue, minimizando riesgos y mejorando la experiencia del usuario final.

https://youtu.be/AGDEHXXqBhc (enlace al video de la actividad 6)