Informe Final

Proyecto MVP – EduTech IA

1.Resumen Ejecutivo

El informe final integrado resume las elecciones y procedimientos implementados en el proyecto de EduTech AI, concebido para mejorar la enseñanza por medio de una plataforma en línea que ajusta los cursos y materiales al perfil y requerimientos individuales de cada estudiante.

Se ha optado por utilizar Java como lenguaje de programación debido a su experiencia previa en el campo de la programación y a su capacidad para adaptarse y crecer en complejidad. Además de esto se decidió seguir el enfoque de Programación Orientada a Objetos para facilitar la aplicación del patrón Modelo-Vista-Controlador (MVC).

Al mismo tiempo se estableció un entorno de desarrollo utilizando IntelliJ IDEA como Entorno de Desarrollo Integrado (IDE), junto a Git y GitHub para la gestión de versiones y la colaboración entre los programadores. La automatización de la integración continua se llevó a cabo a través de un flujo de trabajo CI / CD implementado en contenedores Docker mediante Jenkins; este flujo cubría desde el compromiso en GitHub hasta la ejecución de pruebas y el empaquetado y despliegue en un entorno de pruebas (staging).

Se creó también un plan de seguridad que integra prácticas recomendadas en cada fase del proceso de desarrollo para reducir vulnerabilidades y asegurar la protección de la información. Por último se implementó una estrategia de gestión y optimización de entornos enfocada en mantener la estabilidad y eficiencia del sistema mediante la automatización, normalización y supervisión constante para facilitar la identificación temprana de problemas y la expansión de la infraestructura.

2.1 Razones para seleccionar Java

Java ha mantenido su posición sólida en el mundo empresarial debido a su fiabilidad y solidez; por lo tanto es una elección confiable para aplicaciones fundamentales.
Amplia red de conexiones y comunidad vibrante:

La amplia variedad de marcos y utilidades disponibles en el mercado actualmente -como Spring y Hibernate- hace posible un progreso veloz y de gran calidad en los proyectos de desarrollo de software. Esta gama diversificada de recursos simplifica la solución de desafíos y potencia la efectividad del equipo de programadores.

Portabilidad y adaptabilidad:
La JVM permite que las aplicaciones se ejecuten en diferentes plataformas sin necesidad de cambios importantes, lo que garantiza que el sistema pueda ajustarse fácilmente a futuras necesidades de escalabilidad.

Integrando en los flujos de CI / CD:
La capacidad de Java para trabajar junto a herramientas y procesos de integración y despliegue continuos facilita la automatización de pruebas y despliegues, lo que a su vez mejora la eficiencia en el desarrollo y reduce la probabilidad de errores.
Enfoque de Programación Orientada a Objetos y el modelo MVC.

Modularidad y facilidad de mantenimiento:
La Programación Orientada Objetos (POO) brinda la oportunidad de crear sistemas donde cada clase u objeto contiene información y funciones particulares en su interior. Este enfoque modular simplifica la administración y mejora del código al permitir su actualización y crecimiento de manera efectiva.

Implementación del Modelo-Vista-Controlador (MVC):
El enfoque de la Programación Orientada al Objeto facilitando la implementación del patrón Modelo-Vista-Controlador al dividir de manera clara la funcionalidad del negocio (Model), la presentación visual para el usuario (View), y el control de la interacción entre ellos(Control). Esta división no solo beneficia la estructura del código sino que también promueve la capacidad de reutilización y mantenimiento efectivo en el futuro del sistema.

2.2. Selección de Herramientas y Configuración del Entorno de Desarrollo
Selección de herramientas y configuración del entorno de desarrollo: Elección de herramientas y ajustes del ambiente de desarrollo
Para la creación del Producto Mínimo Viable (MVP) de EduTech IA se implementó un ambiente técnico sólido y colaborativo que asegure la efectividad y la cohesión en cada fase del

procedimiento. Se eligieron las herramientas considerando la urgencia de contar en el entorno de soluciones completas y contrastadas que simplifiquen tanto el progreso del programa como su fusión y lanzamiento constante.
Elección de la herramienta de desarrollo integrado (IDE) y gestión de versiones
Se decidió utilizar IntelliJ IDEA como el IDE principal para el desarrollo del proyecto en lugar de otras opciones disponibles en el mercado debido a sus funciones avanzadas de depuración y refactorización que simplifican el proceso de programación en Java y mejoran la calidad del código final. Además de estas características sobresalientes se destaca su compatibilidad nativa integrada para trabajar de forma conjunta y eficiente junto al sistema Git que facilitará la gestión de versiones del proyecto y promoverá la colaboración efectiva entre los miembros del equipo de desarrollo. Para almacenar el código de manera remota se emplea GitHub. Esta plataforma permite llevar a cabo tareas como solicitudes de incorporación de código y revisiones del mismo para garantizar un flujo de trabajo colaborativo y eficiente.

Jenkins ha sido designado como el servidor de integración continua responsable de automatizar el proceso de construcción y despliegue del proyecto. Se estableció para conectarse a GitHub y llevar a cabo tareas de compilación y validación en respuesta a cada confirmación o solicitud de extracción. Estas acciones permiten identificar posibles errores tempranamente y asegurar que las versiones estables del software estén disponibles. Además se recurre a Docker para generar contenedores que mantengan la coherencia en los entornos de desarrollo, pruebas y producción. También se utiliza Docker para ejecutar el servidor Jenkins en un contenedor para hacer más sencilla su implementación y mantenimiento y reducir las disparidades entre los diversos entornos de ejecución.

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

El proceso comienza al hacer un commit en el repositorio de GitHub para activar un webhook y notificar a Jenkins. Posteriormente Jenkins clona el repositorio y compila el proyecto en Java para generar los artefactos necesarios. Se llevan a cabo pruebas unitarias y de integración para validar la funcionalidad del código. Si todas las pruebas son exitosas, se prepará el software y se crea una imagen Docker que se guarda en un registro de contenedores.

Después de eso la imagen se muestra en un entorno de pruebas (pruebas de etapa), donde se realizan evaluaciones adicionales —ya sean automáticas o manuales— para confirmar que la

aplicación funcione correctamente en condiciones similares al entorno de producción. Luego de que la etapa de pruebas en staging sea completada satisfactoriamente, la imagen es implementada en el entorno de producción. Durante todo este proceso, Jenkins envía notificaciones por correo electrónico o través de integraciones con herramientas de mensajería, y se establece un sistema de monitoreo para evaluar el rendimiento y la estabilidad del sistema.

La automatización total del proceso reduce de manera significativa los errores humanos y acelera la entrega de resultados. En cada fase del proceso que va desde el inicio del desarrollo hasta la implementación final se llevan a cabo pruebas exhaustivas y revisiones automáticas para garantizar la calidad del código. Estas medidas aseguran que solo el código que cumple los estándares establecidos sea desplegado en diferentes entornos para mantener la coherencia y fiabilidad del sistema durante todo el ciclo de desarrollo.

2.4. Protocolo de Seguridad en el Desarrollo

Protocolos de seguridad durante el proceso de desarrollo
Las medidas de seguridad específicas se implementan desde el comienzo de la recopilación de requisitos para garantizar la protección de la información.
Realizamos una evaluación de riesgos empleando técnicas como OWASP Threat Modeling para descubrir posibles puntos débiles y establecer acciones preventivas que reduzcan los riesgos.

Seguridad desde el Diseño y Defensa en Profundidad:
La seguridad se integra desde el inicio del proceso de diseño mediante la creación de sistemas de autenticación y autorización seguros y el manejo adecuado del cifrado y errores para garantizar la protección de la información sensible en diferentes entornos como desarrollo o producción.

Prácticas para Garantizar la Seguridad en la Codificación:
Se respetan las pautas de OWASP y se utilizan diseños que previenen vulnerabilidades comunes de forma efectiva sin dejar lugar para errores peligrosos en el sistema.

Se asegura de que las bibliotecas y marcos se mantengan al día, y se emplean herramientas para detectar vulnerabilidades en las dependencias.

Establecimiento y Control de Ambientes Seguros:
Se establecen ajustes seguros en servidores y contenedores mediante el enfoque de “Privilegio Mínimo” y los principios de “Confianza Cero". En el proceso de contenerización se emplean imágenes Docker simples y se realizan análisis periódicos. Además, se despliega un sistema de supervisión para detectar rápidamente posibles incidentes de seguridad.

Se llevan a cabo actualizaciones periódicas y se aplican parches para garantizar la seguridad del sistema en todo momento. Además se realizan auditorías tanto internas como externas para evaluar la efectividad de las medidas implementadas.

2.5. Estrategia de Gestión y Optimización de Entornos

Ambiente de desarrollo:
Se establecen contenedores de Docker que replican la configuración de producción en el entorno local para que los desarrolladores puedan trabajar en un entorno coherente y simular de manera precisa la ejecución real del software. Además de eso se llevan a cabo procesos de integración y entrega continuas a través de Jenkins, los cuales realizan pruebas y construcciones automáticas tras cada confirmación, asegurando una respuesta rápida y la detención temprana de posibles fallos.

Ambiente de Pruebas en Etapa de Desarrollo:
Se replica el entorno de producción para aplicar las mismas políticas de seguridad y configuración, lo que permite realizar pruebas de aceptación y detectar incidentes sin afectar al usuario final de manera importante y validar el funcionamiento real de la aplicación.

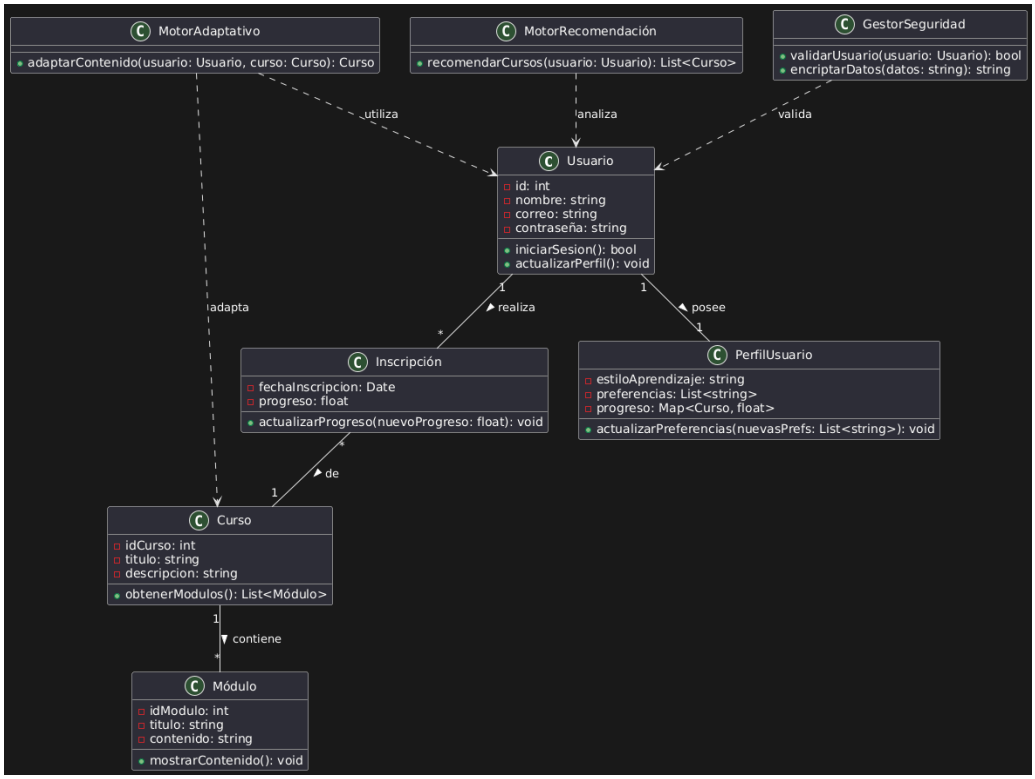
Entorno Laboral:
Se utiliza una estructura escalable y flexible que hace uso de contenedores coordinados y equilibradores de carga para garantizar una disponibilidad óptima del sistema informático. Para mantener la estabilidad del entorno se implementan ajustes de “privilegio mínimo” y se integran herramientas de supervisión y registro para monitorear el buen funcionamiento del sistema, identificar desviaciones en etapas tempranas y llevar a cabo un mantenimiento preventivo proactivo.

Automatización en Integración Continua y Entrega Continua:
La integración de Jenkins y Docker en el flujo CI / CD permite realizar despliegues de manera uniformemente repetible y consistente, reduciendo la necesidad de intervención manual y disminuyendo posibles errores en el proceso automatizado que garantiza una implementación fiable y coherente de cada cambio en todos los entornos.

Supervisión y Control:
Se configuran alarmas y tableros de control en tiempo real para vigilar la eficiencia y estabilidad de cada entorno específico. Las opciones de supervisión y registro ayudan a identificar prontamente situaciones inusuales y facilitar la adopción de medidas correctivas rápidas para garantizar un mantenimiento proactivo.

Actualizaciones y Mejoras:
Se llevan a cabo actualizaciones regulares de software y revisiones de seguridad para mantener la protección del sistema al día según las prácticas recomendadas en la industria tecnológica actualizada. Además de esto se utilizan métodos de seguimiento para ajustar los recursos tanto horizontal como verticalmente según la demanda que se presente en cada momento específico; todo ello garantizando no solo la escalabilidad del sistema sino también una protección integral de la información contenida en él.

3.Diagramas de Arquitectura (MVP y MVC)



Diagramas de Arquitectura (Modelo Vista Presentador y Modelo Vista Controlador)
El esquema del Producto Mínimo Viable (MVP), que se encuentra en la imagen adjunta, ilustrando la disposición general de la plataforma y resaltando la integración de los elementos que posibilitan personalizar cursos y contenidos de acuerdo al perfil y requerimientos de cada usuario. Además se incluye un esquema de la arquitectura Modelo-Vista-Controlador que muestra claramente la separación entre el Modelo ,la Vista y el Controlador facilitando así la comprensión de la estructura del sistema.

Gráficos que representan el flujo de trabajo de integración y entrega continua (CI / CD) y la administración de entornos.
La guía “Organización del Proceso de Integración Continua y Entrega Continua (CI / CD)” ilustrada mediante diagramas detalla todo el proceso del pipeline; desde la confirmación en GitHub hasta la implementación en el entorno de pruebas (staging), y finalmente en producción. Además del anteriormente mencionado documento “Plan de Manejo y Mejora de Entornos", se incluyen representaciones visuales que muestran cómo se configuran y monitorean los entornos de desarrollo en etapas iniciales y producción para destacar la automatización y estandarización implementadas para garantizar la estabilidad del sistema.

Se ha creado un video demostrativo que simula cómo se implementa el Producto Viable Minimo en un entorno de pruebas; en él se describen paso a paso todas las fases del pipeline CI / CD y se verifica el correcto funcionamiento del sistema en detalle.

Link del Vídeo