Actividad 8: Informe Final Integrado

Tarea

Elabora un informe final que integre y resuma todo el proceso realizado. El documento debe incluir:

- Resumen ejecutivo.
- Justificación de las decisiones técnicas y arquitectónicas.
- Documentación de cada actividad (incluyendo enlaces al repositorio y al video).
- Conclusiones y aprendizajes obtenidos.

Entregable: Documento en PDF (entre 8 y 12 páginas).

Puntuación: 2 puntos

1. Resumen Ejecutivo

Este informe documenta el proceso de diseño, desarrollo y evaluación de una plataforma inteligente de gestión de proyectos basada en inteligencia artificial. Se presenta un resumen de las decisiones arquitectónicas, la implementación de principios y patrones de diseño, estrategias de pruebas y comparación con soluciones industriales. El objetivo es garantizar una arquitectura modular, extensible y mantenible, cumpliendo con los principios de diseño modernos y asegurando la calidad del software a través de pruebas rigurosas.

2. Justificación de las Decisiones Técnicas y Arquitectónicas

2.1 Principios de Diseño

Para garantizar la calidad del software, se aplicaron los siguientes principios fundamentales:
-**S.O.L.I.D**: Facilita la modularidad y el desacoplamiento, reduciendo la deuda técnica y permitiendo modificaciones sin afectar módulos independientes.
-**SRP (Single Responsibility Principle)**: Cada módulo tiene una sola razón para cambiar.
-**OCP (Open/Closed Principle)**: Los componentes son extensibles sin modificar su estructura básica.
-**LSP (Liskov Substitution Principle)**: Se garantiza la intercambiabilidad entre clases derivadas y sus bases.
-**ISP (Interface Segregation Principle)**: Evita interfaces infladas que los clientes no necesitan.
-**DIP (Dependency Inversion Principle)**: Se depende de abstracciones en lugar de implementaciones concretas.
-**DRY (Don't Repeat Yourself)**: Minimiza la redundancia mediante la reutilización de código, optimizando el mantenimiento.
-**KISS (Keep It Simple, Stupid)**: Se adopta un diseño claro y minimalista para facilitar la comprensión.
-**YAGNI (You Ain't Gonna Need It)**: Evita el desarrollo de funcionalidades innecesarias, permitiendo una evolución ágil del sistema.

2.2 Patrones de Diseño

Para mejorar la arquitectura del software, se han seleccionado patrones de diseño que aportan escalabilidad y flexibilidad:
-**Factory Method (Creacional)**: Proporciona un mecanismo flexible y escalable para la creación de objetos sin especificar su clase concreta.
-**Adapter (Estructural)**: Facilita la interoperabilidad con servicios externos sin alterar la estructura interna de la plataforma.
-**Observer (Comportamiento)**: Permite la implementación de notificaciones en tiempo real, mejorando la interactividad del sistema.

3. Documentación de Cada Actividad

Toda la documentación se encuentra en el repositorio creado para esta práctica:

https://github.com/SMayomboS/Feedback_Tema5.git

3.1 Actividad 1: Análisis y Selección de Principios de Diseño

Se investigaron los principios de diseño fundamentales y se documentó su aplicación en la plataforma. Se diseñaron módulos desacoplados y extensibles, asegurando la mantenibilidad del software.

3.2 Actividad 2: Identificación y Justificación de Patrones de Diseño

Se definieron y justificaron los patrones Factory Method, Adapter y Observer. Estos patrones permitieron desarrollar un sistema flexible y extensible que puede adaptarse a nuevas funcionalidades sin afectar su base.

3.3 Actividad 3: Esquema de la Arquitectura del Software

Se elaboró un diagrama UML representando la estructura del sistema, detallando la interacción entre los componentes principales. Este diagrama facilita la comprensión del diseño y su posible evolución.

3.4 Actividad 5: Estrategia de Control de Calidad y Pruebas Unitarias

Para garantizar la fiabilidad del software, se implementaron pruebas unitarias y de integración con:
-**JUnit 5**: Verifica la funcionalidad de componentes individuales.
-**Mockito**: Simula dependencias externas en las pruebas.
-**Checkstyle**: Asegura buenas prácticas de codificación.
-**SonarQube**: Realiza análisis estáticos de calidad y seguridad.
-**JaCoCo**: Mide la cobertura de pruebas, asegurando que el código crítico esté probado.

3.5 Actividad 7: Análisis Comparativo con Arquitecturas Industriales

Se comparó la arquitectura diseñada con Jira, Trello y Asana. Se identificaron fortalezas y oportunidades de mejora:
- **Fortalezas**:
  - Arquitectura modular y desacoplada basada en SOLID.
  - Implementación de patrones de diseño para extensibilidad.
  - Notificaciones en tiempo real mediante el patrón Observer.
- **Oportunidades de Mejora**:
  - Implementar una arquitectura basada en microservicios.
  - Diseñar una API RESTful para integración con terceros.
  - Optimizar el rendimiento con caching y balanceo de carga.

4. Conclusiones y Aprendizajes Obtenidos

Este proyecto ha permitido consolidar conocimientos en diseño arquitectónico, aplicación de patrones y estrategias de calidad de software. La comparación con soluciones industriales muestra que el sistema es modular y extensible, pero puede beneficiarse de:
-**Escalabilidad**: Implementar una arquitectura basada en microservicios.
-**Integración**: Diseñar APIs RESTful para interoperabilidad con otras plataformas.
-**Optimización**: Mejorar el rendimiento con estrategias avanzadas de caching.

La aplicación de principios de diseño y pruebas automatizadas garantiza la calidad del software, reduciendo la deuda técnica y mejorando la mantenibilidad. En futuras iteraciones, se recomienda explorar herramientas de contenedorización como Docker y Kubernetes para optimizar la escalabilidad y distribución del sistema.