Actividad 1: Análisis y Selección de Principios de Diseño

Tarea:

Investiga y analiza los principios S.O.L.I.D., DRY, KISS y YAGNI. Redacta un documento en el que expliques cómo aplicarás cada uno de estos principios en el diseño de la arquitectura de la plataforma.

Entregable: Documento en PDF (máximo 2 páginas).

Puntuación: 1.5 puntos

1. Principios de Diseño Aplicados

1.1 Principios S.O.L.I.D.

S.O.L.I.D. es un acrónimo de cinco principios que favorecen la creación de código modular y flexible:

- Single Responsibility Principle (SRP): Cada clase o módulo debe tener una única razón para cambiar. En la arquitectura propuesta, esto se aplica dividiendo las responsabilidades en servicios independientes.

-Open/Closed Principle (OCP): Los componentes deben estar abiertos a la extensión pero cerrados a la modificación. Esto se logrará mediante el uso de interfaces y la inyección de dependencias, permitiendo agregar nuevas funcionalidades sin modificar el código ya generado.

-Liskov Substitution Principle (LSP): Una clase derivada debe poder sustituir a su clase base sin afectar el correcto funcionamiento del programa. Aplicando este principio en la plataforma, se diseñarán clases de IA y predicción que sean intercambiables sin modificar el flujo de trabajo.

-Interface Segregation Principle (ISP): Es preferible tener interfaces específicas en lugar de una única y extensa. Se diseñarán interfaces enfocadas en tareas específicas para evitar que los componentes dependen de métodos que no utilizan.

-Dependency Inversion Principle (DIP): Se debe depender de abstracciones en lugar de implementaciones concretas. Se utilizará la inyección de dependencias para desacoplar los módulos de la plataforma.

1.2 Principio DRY (Don’t Repeat Yourself)

El principio DRY fomenta la eliminación de la redundancia en el código. Se aplicará en la plataforma mediante la reutilización de componentes comunes, como validadores de datos y servicios de autenticación, asegurando que la lógica compartida esté centralizada en módulos reutilizables.

1.3 Principio KISS (Keep It Simple, Stupid)

La simplicidad es clave para mejorar la comprensión y mantenibilidad del código. Se adoptarán diseños minimalistas y se evitará la complejidad innecesaria en la implementación de funciones, dividiendo los módulos en componentes bien definidos y fácilmente comprensibles.

1.4 Principio YAGNI (You Ain’t Gonna Need It)

Este principio sugiere que no se deben implementar funcionalidades que no sean estrictamente necesarias en el momento. Se priorizará el desarrollo iterativo, agregando funcionalidades según las necesidades del usuario y evitando una sobrecarga de código innecesario.

La aplicación de estos principios permitirá diseñar una arquitectura modular, extensible y mantenible para la plataforma de gestión de proyectos. Al garantizar la separación de responsabilidades, minimizar la redundancia y enfocarse en la simplicidad, se logrará una solución escalable que pueda evolucionar sin generar una alta deuda técnica.