Actividad 1: Análisis y Selección de Principios de Diseño

El diseño de la arquitectura de una plataforma inteligente de gestión de proyectos requiere la aplicación de principios modernos de desarrollo de software que aseguren un código limpio, reutilizable y eficiente. A continuación, se presentarán los principios S.O.L.I.D., DRY(Don't Repeat Yourself), KISS(Keep It Simple, Stupid) y YAGNI.

1. **S.O.L.I.D**
   Los principios S.O.L.I.D son un conjunto de cinco principios de diseño en programación orientada a objetos que promueven la mantenibilidad y escalabilidad del código.

   - **S**: Principio de responsabilidad.
     (Single Responsibility Principle, SRP)
     Cada clase o módulo debe tener una única razón para cambiar. Esto mejora la organización y evita dependencias innecesarias.
   - **O**: Principio de Abierto/Cerrado
     (Open/Closed Principle, OCP)
     Los módulos deben estar abiertos para la extensión pero cerrados para la modificación, permitiendo mejorar funcionalidades sin alterar el código base
   - **L**: Principio de Sustitución de Liskov
     (Liskov Substitution Principle, LSP)
     Las subclases deben ser sustituibles por sus clases base sin afectar el comportamiento del sistema.
   - **I**: Principio de Segregación de Interfaces
     (Interface Segregation Principle, ISP)
     Las interfaces deben ser específicas y pequeñas, evitando que una clase deba implementar métodos que no utiliza.
   - **D**: Principio de Inversión de Dependencias
     (Dependency Inversion Principle, DIP)
     Los módulos de alto nivel no deben depender de módulos de bajo nivel, sino de abstracciones para evitar un acoplamiento fuerte.

2. **DRY**
   (Don't Repeat Yourself)
   Este principio establece que cada pieza de conocimiento dentro del sistema debe tener una única representación en el código. Evita la duplicación de lógica mediante la reutilización de funciones, clases o módulos. Esto reduce la probabilidad de errores y facilita el mantenimiento.

3. **KISS**
   (Keep It Simple, Stupid)
   YAGNI sugiere que no se debe agregar funcionalidad al software a menos que sea estrictamente necesaria en el momento. Esto evita el desperdicio de recursos en funcionalidades que pueden no ser requeridas en el futuro.

4. **YAGNI**
   (You Aren't Gonna Need It)
   YAGNI sugiere que no se debe agregar funcionalidad al software a menos que sea estrictamente necesaria en el momento. Esto evita el desperdicio de recursos en funcionalidades que pueden no ser requeridas en el futuro.

CONCLUSIÓN

La aplicación de estos principios en la plataforma inteligente de gestión de proyectos garantizará un código modular, mantenible y eficiente. Los principios S.O.L.I.D. guiarán el diseño orientado a objetos, mientras que DRY, KISS y YAGNI ayudarán a mantener el sistema limpio, simple y libre de funcionalidades innecesarias.