Principios SOLID de Diseño de Software

Principio de Responsabilidad Única

- Una clase solo debe tener una razón para cambiar.
- Cada clase debe ser responsable de una única parte de la funcionalidad.
- Promueve la cohesión y evita la dispersión de responsabilidades.

Principio de Abierto/Cerrado

- Las entidades de software deben estar abiertas para su extensión pero cerradas para su modificación.
- Se busca evitar cambiar el código existente cada vez que se agrega nuevas funcionalidades.
- Permite el uso de la herencia y la composición para lograr extensibilidad.

Principio de Inversión de Dependencias

- Los módulos de alto nivel no deben depender de módulos de bajo nivel.
- Ambos deben depender de abstracciones.
- Promueve el uso de interfaces y la inyección de dependencias para facilitar la flexibilidad y extensibilidad del sistema.

Principio de Segregación de Interfaces

- Los clientes no deben depender de interfaces que no utilizan.
- Se busca minimizar las dependencias y acoplamiento entre componentes.
- Promueve la creación de interfaces específicas para cada cliente o módulo.

Principio de Sustitución de Liskov

- Las clases derivadas deben poder usarse en lugar de sus clases bases.
- Se busca garantizar la compatibilidad y el correcto funcionamiento de la jerarquía de clases.
- Evita situaciones donde el comportamiento de una clase derivada rompe el contrato definido por su clase base.