Principios Solid

1. Principio de Responsabilidad Única: Cada clase o módulo debe tener una única razón para cambiar. Esto significa que una clase debe tener una sola responsabilidad y función específica en el sistema.

2. Principio Abierto/Cerrado: Las entidades de software deben estar abiertas para la extensión, pero cerradas para la modificación. Esto implica que se pueden agregar nuevas funcionalidades sin cambiar el código existente.

3. Principio de Sustitución de Liskov: Las clases derivadas deben ser sustituibles por sus clases base sin afectar la funcionalidad correcta del programa. En otras palabras, una clase hija debe poder usarse en lugar de su clase padre sin problemas.

4. Principio de Segregación de Interfaces: Los clientes no deben verse obligados a depender de interfaces que no utilizan. Las interfaces deben estar diseñadas específicamente para los requerimientos de sus clientes.

5. Principio de Inversión de Dependencias: Los módulos de alto nivel no deben depender de módulos de bajo nivel; ambos deben depender de abstracciones. Además, las abstracciones no deben depender de los detalles, sino que los detalles deben depender de las abstracciones.