Os princípios SOLID são um conjunto de boas práticas que ajudam a tornar o design de software orientado a objetos mais flexível, compreensível e fácil de manter.

SRP - Princípio da Responsabilidade Única

Cada classe deve ter apenas uma razão para mudar, ou seja, deve ser responsável por uma única funcionalidade do sistema. Ao misturar responsabilidades, mudanças em uma funcionalidade exigem alterações desnecessárias em outras partes.

OCP - Princípio Aberto / fechado

O software deve ser aberto para extensão, mas fechado para modificação. Ou seja, é possível adicionar novos comportamentos sem alterar o código existente. Isso é feito, por exemplo, utilizando herança ou interfaces, tornando o sistema mais modular.

LSP - Princípio da Substituição de Liskov

Classes derivadas devem poder substituir suas classes base sem afetar o comportamento do programa. Um problema comum ocorre quando uma subclasse altera o comportamento esperado da superclasse, como no exemplo de quadrado herdando de retângulo (que pode levar a inconsistências).

ISP - Princípio da Segregação de Interface

É melhor ter várias interfaces específicas do que uma única interface genérica. Assim, classes não são forçadas a implementar métodos que não utilizam. Isso leva a uma composição de comportamentos, permitindo maior flexibilidade.

DIP - Princípio da inversão de dependência

As classes devem depender de abstrações e não de implementações concretas. Isso se faz usando técnicas como injeção de dependência (via construtor, métodos, etc), permitindo maior desacoplamento e facilitando a substituição ou extensão de componentes

Os princípios SOLID não são regras rígidas, mas diretrizes que auxiliam no desenvolvimento de software de qualidade, favorecendo a manutenção, a reutilização e a evolução do sistema. Eles também incentivam o uso de composição ~~ao mais~~ invés de herança, promovendo projetos mais ~~flexi~~ flexíveis e robustos.