Aluno: Antônio de Oliveira Agnolin R.A.: 22012580-2 Turma: ADSIS5S

No vídeo, Rodrigo Branas fala sobre conceitos de extrema importância para criar projetos de software, com código modular, desacoplamento, inversão de dependências e abertura para extensões, conforme proposto pelos princípios SOLID.

Os princípios de SOLID são:

1. Single Responsibility Principle (SRP - Princípio da Responsabilidade Única): Uma classe deve ter apenas uma razão para mudar, ou seja, deve ter apenas uma responsabilidade bem definida. Ao seguir esse princípio, as classes se tornam mais coesas e fáceis de entender, facilitando a manutenção do código.

2. Open/Closed Principle (OCP - Princípio Aberto/Fechado): As entidades de software devem estar abertas para extensão, mas fechadas para modificação. Isso significa que o comportamento de uma classe deve poder ser estendido sem que seja necessário modificar seu código-fonte original.

3. Liskov Substitution Principle (LSP - Princípio da Substituição de Liskov): Subtipos devem ser substituíveis por seus tipos base sem afetar a corretude do programa. Isso garante que a herança seja usada de forma correta e que as classes derivadas possam ser usadas no lugar de suas classes base sem problemas.

4. Interface Segregation Principle (ISP - Princípio da Segregação de Interface): Uma classe não deve ser forçada a depender de métodos que não utiliza. Interfaces devem ser específicas e conter apenas os métodos necessários para as classes que as implementam.

5. Dependency Inversion Principle (DIP - Princípio da Inversão de Dependência): Módulos de alto nível não devem depender de módulos de baixo nível. Ambos devem depender de abstrações. Isso promove um baixo acoplamento entre as classes e facilita a substituição de implementações.

Rodrigo Branas mostra como aplicar esses princípios na prática, criando um projeto de software com uma arquitetura sólida e desacoplada. Ele remodela classes, desenvolve estruturas e refatora o código para torná-lo mais modular e manutenível. Além dos princípios SOLID, Rodrigo Branas aborda outros conceitos importantes durante sua apresentação:

1. Portas e Adaptadores (Ports and Adapters): Também conhecido como arquitetura hexagonal ou arquitetura de porta/ adaptador, esse conceito foca na separação entre o domínio do negócio e as tecnologias externas (como banco de dados, frameworks, etc.). O objetivo é garantir que o domínio do negócio (conhecido como "lado do domínio" ou "driver side") não dependa diretamente das implementações técnicas (conhecidas como "lado técnico" ou "driven side"). Isso permite que o código do domínio permaneça independente das tecnologias específicas, facilitando testes e evolução.

2. Arquitetura Limpa (Clean Architecture): Esse é um padrão arquitetural proposto por Robert C. Martin, que define uma separação clara entre as diferentes camadas de um sistema. As camadas são organizadas em círculos concêntricos, onde o círculo interno representa o núcleo do sistema (que contém as regras de negócio) e os círculos externos representam as camadas externas (como interfaces com o usuário, infraestrutura, etc.). A arquitetura limpa promove um design modular e desacoplado, facilitando a manutenção e evolução do sistema ao longo do tempo.