UNIVERSIDADE FEDERAL DO CARIRI

Análise e Desenvolvimento de Sistemas (ADS)
Centro de Educação a Distância (CEAD)

DOCUMENTAÇÃO

PROPRIEDADE DE BONS PROJETOS

&

PRINCÍPIOS DE BONS PROJETOS

CONTROLE DE PROJETOS

05 / FEV

Discente: Pedro Hilbert Silva Guerra

- **INTRODUÇÃO**

Este documento tem como objetivo apresentar uma documentação didática das **propriedades de bons projetos** e dos **princípios de bons projetos**, aplicados à funcionalidade de Controle de Projetos para uma empresa de consultoria em engenharia civil.

● Propriedades de Bom Projetos

- Integridade Conceitual

Integridade conceitual desempenha um papel crucial para garantir que o sistema seja bem compreensível e eficaz para os gerentes de projeto. Dentro o novo sistema, será implementado as funcionalidades:

1. Cadastro de novos projetos: Onde será informado o nome do novo projeto, localização, informações do cliente, data de início prevista para conclusão.
2. Remover projetos: Aqui o gerente de projeto pode excluir os projetos pelo qual o contrato foi cancelado ou em outras opções.
3. Consultar Projetos: Aqui o gerente vai poder consultar informações dos projetos que já estão em andamento ou que já foram concluido.
4. Alterar Projetos: Aqui o gerente de projetos vai poder alterar informações dos projetos, essencial nos casos de falhas no cadastro inicial, ele poderá alterar informações de acordo com suas necessidades.

- Ocultamento de Informação

Nesse projeto, será importante ocultar informações que não possam ser acessadas por qualquer pessoas, então será vital o ocultamento de informações, além disso, facilitará na implementação e manutenção por outros usuários nas seguintes funcionalidades:

1. Cadastro de novos projetos: Ao cadastrar novos projetos, é crucial esconder os detalhes internos da implementação, será encapsulada as informações que o gerente cadastrar.
2. Remover projetos: Ao remover projetos, é importante esconder os processos internos para garantir que mudanças na forma como projetos são removidos não vá trazer problemas para as outras partes do sistema.

3. Consultar Projetos: Durante a consulta de projetos, devemos oferecer uma interface clara, expondo apenas as informações necessárias e escondendo a complexidade interna.
4. Alterar Projetos: Ao gerente realizar alguma alteração, devemos criar classes para os campos em que é permitido alterar informações, para que não traga graves problemas de informações erradas em que foram alteradas.

- Coesão

Cada parte do Controle de Projetos deve ser como um especialista em uma única tarefa. Isso não apenas simplifica o desenvolvimento e a manutenção, mas também torna o sistema mais robusto e fácil de entender.

- Acoplamento

Entre as funções apresentadas, o sistema contará com acoplamento entre as funcionalidades de Cadastro e Alteração. Sem aplicar o acoplamento, nosso sistema pode ficar um caos, para isso aplicaremos o acoplamento que será o equilíbrio em nosso sistema.

● Princípios de Bons Projetos

- SOLID

- **(S) Princípio de Responsabilidade Única**

Garantir que cada classe tenha uma única responsabilidade, promovendo a coesão e facilitando a manutenção. Iremos evitar sobrecarregar a classe com funcionalidades não relacionadas.

- **(O) Aberto Fechado/Princípio**

O “O” incentiva o design de software que permite a introdução de novas funcionalidades sem alterar o código existente. Isso é alcançado através das interfaces.

- **(L) Princípio da Substituição de Liskov**

O “L” destaca que os objetos de uma classe derivada, devem poder substituir objetos de sua classe base sem mudar a performance do programa. Isso promove uma hierarquia de classes mais sólida e flexível.

- **(I) Princípio de segregação de interface**

Uma classe não deve ser forçada a implementar interfaces que ela não usa. uma classe deve implementar apenas as interfaces que são relevantes para sua funcionalidade. Evita a necessidade de implementar métodos que não são necessários para a classe.

- **(D) Princípio de Inversão de Dependência**

Em vez de depender diretamente dos detalhes internos de uma classe específica, a ideia aqui é depender de uma "ideia geral" ou abstração. Isso vai ajuda a manter as coisas flexíveis. Se precisarmos mudar como fazemos algo no futuro.

- Prefira Interfaces Classes

Esse princípio fala para utilizarmos interfaces em vez de classes concretas sempre que possível, promovendo flexibilidade e extensibilidade. Aqui devemos preferir interfaces concretas que permitam diferentes implementações possam ser trocadas sem afetar o resto do sistema. Isso promove uma maior flexibilidade e facilita a adição de novas funcionalidades.

- Demeter

Aqui iremos trabalhar com as interações dos objetos, controlando a comunicação entre os objetos. Restringindo as chamadas de métodos entre classes diferentes para promover uma comunicação mais direta e eficiente.