RESUMO DA AULA

Desenvolvimento Gradual e Refatoração e DRY

Desenvolvimento Gradual e Refatoração

O desenvolvimento gradual e a refatoração são duas práticas essenciais para manter o código limpo ao longo do tempo. O desenvolvimento gradual sugere que você não precisa tentar resolver tudo de uma vez. Em vez disso, você deve criar soluções simples que podem ser melhoradas ao longo do tempo.

A refatoração é o processo de reescrever o código sem modificar sua funcionalidade, mas melhorando sua estrutura e legibilidade. Refatorar periodicamente é importante para evitar que o código se torne confuso ou ineficiente.

Dicas para refatoração eficiente:

Pequenas mudanças: Não tente refatorar um grande bloco de código de uma só vez. Refatore gradualmente, garantindo que cada mudança seja testada.

Priorize simplicidade: Sempre que possível, busque simplificar o código. Remova duplicações, reduza funções longas e elimine variáveis desnecessárias.

Teste sempre: Após refatorar, certifique-se de que o comportamento do código permaneça o mesmo, verificando com testes unitários ou funcionais.

Exemplo de refatoração:

Antes:

```
function calcularDesconto(preco, desconto) {
  if (desconto > 0) {
    return preco - (preco * desconto);
  } else {
```

```
    return preco;
  }
}
```

Após refatoração:

```
function calcularDesconto(preco, desconto) {
  return desconto > 0 ? preco - (preco * desconto) : preco;
}
```

DRY - Don't Repeat Yourself

O princípio DRY (Don't Repeat Yourself) é essencial para evitar repetição desnecessária de código. Sempre que você perceber que está duplicando lógica em diferentes partes do seu projeto, deve tentar isolar essa lógica em uma função ou componente reutilizável.

A duplicação de código torna o software mais difícil de manter e aumenta as chances de erros. Alterações que precisam ser feitas em vários lugares são mais propensas a passar despercebidas em algum ponto, causando inconsistências no sistema.

Exemplo de como aplicar DRY:

Antes (código repetido):

```
function calcularImposto(salario) {
  return salario * 0.2;
}
function calcularImpostoBonus(bonus) {
  return bonus * 0.2;
}
```

Após aplicar DRY:

```
function calcularImposto(valor) {
  return valor * 0.2;
}
```

Com esse simples ajuste, removemos a duplicação e agora, qualquer mudança na taxa de imposto só precisa ser feita em um lugar.

O Código Limpo não é apenas uma prática recomendada - ele é essencial para garantir a longevidade e a qualidade do software. Seguindo os princípios de nomes claros, refatoração constante, desenvolvimento gradual e evitando repetição, você estará no caminho certo para se tornar um desenvolvedor que escreve código de fácil manutenção e altamente escalável.
