DESENVOLVIMENTO WEB III

Boas Práticas de Código

Prof. Esp. Érick Henrique Pereira Nicolau

Por que se preocupar com boas práticas?

- Código limpo e organizado é mais fácil de ler, entender e modificar.
- Facilita a colaboração entre desenvolvedores.
- Reduz a probabilidade de erros e bugs.
- Melhora a manutenibilidade do software a longo prazo.

Legibilidade

- Código legível é como uma boa história: fácil de acompanhar e entender.
- Use nomes descritivos para variáveis, funções e classes.
- Comente seu código para explicar o que ele faz e por quê.
- Use indentação consistente para mostrar a estrutura do código.
- Evite linhas de código muito longas.

```
function c(a,b){let d=0;for(let
i=0;i<a.length;i++){if(a[i]===b){d++}}return d}
```

```
function contarOcorrencias(array, elemento) {
  let contador = 0;
  for (let i = 0; i < array.length; i++) {
    if (array[i] === elemento) {
      contador++;
    }
  }
  return contador;
}
```

Organização

Organize seu código em módulos, classes e funções.

Cada módulo, classe ou função deve ter uma responsabilidade única e bem definida.

Separe a lógica de negócio da interface do usuário.

Use padrões de projeto para estruturar seu código de forma consistente.

```
// Função para cadastrar um novo usuário
function cadastrarUsuario(usuario) {
  // ... lógica de cadastro de usuário
}

// Representação de um usuário
class Usuario {
  // ... propriedades e métodos do usuário
}

// Operações de banco de dados
const database = require('./database');
```

```
// Banco de dados
const database = require('./database');

// Classe para representar um usuário
class Usuario {
  // ...
}

// Função para cadastrar um novo usuário
function cadastrarUsuario(usuario) {
  // ...
}
```

DRY (Don't Repeat Yourself)

- Evite repetir código.
- Se você está escrevendo o mesmo código várias vezes, provavelmente há uma maneira de abstraí-lo em uma função ou classe reutilizável.
- Isso torna o código mais fácil de manter e reduz a probabilidade de erros.

```
function calcularAreaRetangulo(largura, altura) {
  return largura * altura;
}

function calcularAreaQuadrado(lado) {
  return lado * lado;
}
```

```
function calcularArea(forma) {
  if (forma instanceof Retangulo) {
    return forma.largura * forma.altura;
  } else if (forma instanceof Quadrado) {
    return forma.lado * forma.lado;
  }
}
```

KISS (Keep It Simple, Stupid)

- Mantenha seu código simples e fácil de entender.
- Evite soluções complexas ou excessivamente engenhosas.
- Código simples é mais fácil de depurar, testar e manter.

```
function calcularPrecoTotal(produtos) {
  let precoTotal = 0;
  for (let i = 0; i < produtos.length; i++) {
    let produto = produtos[i];
    if (produto.desconto) {
      let desconto = produto.preco * (produto.desconto /
100);
      precoTotal += produto.preco - desconto;
    } else {
      precoTotal += produto.preco;
    }
    if (produto.frete) {
      precoTotal += produto.frete;
    } else {
      if (precoTotal < 100) {
        precoTotal += 10; // Frete padrão
      }
    }
  }
  return precoTotal;
}
```

```
function calcularPrecoTotal(produtos) {
  let precoTotal = 0;
  for (let produto of produtos) {
    precoTotal += calcularPrecoComDesconto(produto);
    precoTotal += calcularFrete(produto, precoTotal);
  }
  return precoTotal;
}

function calcularPrecoComDesconto(produto) {
  if (produto.desconto) {
    return produto.preco * (1 - produto.desconto / 100);
  } else {
    return produto.preco;
  }
}

function calcularFrete(produto, precoTotal) {
  if (produto.frete) {
    return produto.frete;
  } else if (precoTotal < 100) {
    return 10; // Frete padrão
  } else {
    return 0; // Frete grátis
  }
}
```