Regras de Boas Práticas de Programação

Objetivo

Este documento apresenta um conjunto de regras e boas práticas que devem ser seguidas pelo grupo durante o desenvolvimento do projeto. O objetivo é garantir padronização, legibilidade, qualidade de código e facilidade de manutenção. As práticas foram selecionadas com base em princípios de Clean Code, SOLID e diretrizes de codificação abordadas em sala de aula.

Cada item abaixo constitui uma regra obrigatória para todos os membros da equipe.

---

1. Padrão de Notação de Código

Toda a equipe deve utilizar um padrão consistente de nomeação, seguindo normas adequadas ao paradigma e à linguagem utilizada.

Padrões adotados

- Variáveis e funções: camelCase.
- Classes: PascalCase.
- Constantes: UPPER_CASE_SNAKE_CASE.
- Arquivos: nomes descritivos, sem abreviações desnecessárias.

Regras gerais

- Evitar nomes genéricos como x ou temp.
- Nomes devem expressar claramente a intenção.
- Evitar abreviações que prejudiquem entendimento.

---

2. Documentação e Comentários no Código

Comentários

- Comentar por que algo foi feito, e não o que foi feito.
- O código deve ser autoexplicativo.
- Comentários devem ser atualizados quando o código mudar.
- Comentários redundantes ou óbvios devem ser evitados.

Documentação

Toda função deve possuir documentação mínima contendo:

- Descrição do propósito
- Parâmetros
- Valor retornado
- Exceções/erros relevantes

---

3. Aplicar o Princípio da Responsabilidade Única (SRP – SOLID)

Cada classe, função ou módulo deve ter apenas uma responsabilidade bem definida. Isso significa:

- Evitar funções grandes e multifuncionais.
- Dividir lógicas complexas em componentes menores.
- Uma alteração no sistema deve exigir mudança em apenas um local coerente.

---

4. Código Limpo (Clean Code): Funções Pequenas e Claras

Com base nos princípios de Clean Code:

- Funções devem ser curtas (idealmente entre 5–15 linhas).
- Cada função deve fazer **uma única coisa**, mas fazê-la bem.
- Funções devem ter nomes descritivos, começando com verbos.

Outras boas práticas incluídas nesta regra:

- Evitar repetições de código (DRY – *Don’t Repeat Yourself*).
- Manter indentação uniforme.
- Evitar uso excessivo de condicionais aninhadas (*nested ifs*).

---

5. Evitar Código Morto, Complexidade Desnecessária e Abordagens Anti-Padrão

Para melhorar a manutenibilidade do sistema:

- Remover variáveis, funções ou classes não utilizadas.
- Não deixar trechos comentados que nunca serão reativados.
- Evitar algoritmos complexos sem necessidade.
- Utilizar estruturas de dados adequadas a cada situação.
- Evitar "code smells", como:
    - funções enormes
    - parâmetros demais
    - dependências cíclicas
    - nomes vagos

Essa regra ajuda a manter o código limpo, claro e sustentável ao longo do tempo.

---

6. Aplicar o Princípio da Interface Segregation (ISP – SOLID)

- Interfaces grandes devem ser divididas em interfaces menores e específicas.
- Consumidores não devem depender de métodos que não utilizam.

---

7. Conclusão

As regras definidas neste documento têm o objetivo de promover organização, clareza e padronização no código produzido pela equipe. O uso combinado de padrões de notação, documentação adequada, princípios do SOLID e recomendações de Clean Code contribuirá para um software mais sustentável, de fácil manutenção e alinhado às práticas profissionais da Engenharia de Software.