Por que estudar Boas Práticas de Programação?

Introdução e Motivação

Prof. Fernando Figueira

DIMAp – UFRN

22 de Agosto de 2025

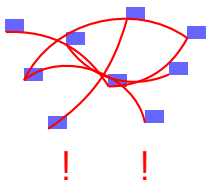
Sumário

1. Introdução
2. Impacto no Mundo Real
3. Benefícios das Boas Práticas
4. Princípios Fundamentais
5. CI/CD e Boas Práticas
6. Impacto na Carreira
7. Aplicação Prática
8. Conclusão

O Dilema do Desenvolvedor

Código que funciona ≠ Código bom

- Qualquer um pode escrever código que um computador entenda
- Bons programadores escrevem código que humanos entendam
- Manutenção consome 60-80% do custo total de software



Código Espaguete

Fonte: Ilustração própria - Representação de código espaguete

O que são Boas Práticas de Programação?

Definição

Conjunto de técnicas, princípios e padrões que melhoram a qualidade, legibilidade e manutenibilidade do código, facilitando a colaboração e reduzindo erros.

- Nomenclatura clara
- Organização lógica
- Comentários úteis
- Formatação consistente
- Tratamento de erros
- Princípios de design
- Testes automatizados
- Refatoração contínua

Custo do Código de Baixa Qualidade

Problema	**Custo Anual (Estimativa)**
Tempo perdido com manutenção	$85 bilhões
Bugs e erros evitáveis	$62 bilhões
Retrabalho por código ilegível	$45 bilhões
Complexidade desnecessária	$38 bilhões

Fonte: Estudos sobre produtividade em desenvolvimento de software (2023)

Caso Real: Knight Capital

Um exemplo extremo

Em 2012, a Knight Capital perdeu $460 milhões em 45 minutos devido a:

- Código não testado adequadamente
- Sistema de deploy falho
- Código legado não documentado
- Falta de práticas de segurança

Resultado

- Quase levou à falência da empresa
- Aquisição forçada por concorrente
- Multa de $12 milhões

Vantagens para Desenvolvedores

Produtividade

- Menor tempo de debug
- Entendimento mais rápido
- Menos retrabalho
- Integração facilitada

Qualidade de Vida

- Menor estresse
- Maior satisfação
- Menor carga cognitiva
- Orgulho do trabalho

Desenvolvedores felizes escrevem código melhor

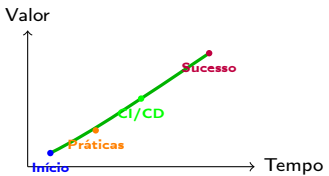
Vantagens para Empresas

Econômicas

- Redução de custos
- Menor rotatividade
- Retorno mais rápido
- Menor risco legal

Estratégicas

- Competitividade
- Escalabilidade
- Adaptabilidade
- Reputação



Crescimento com Boas Práticas

Fonte: Ilustração própria - Crescimento empresarial com boas práticas

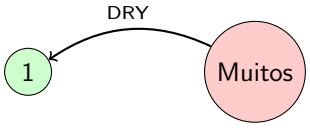
DRY: Don't Repeat Yourself

Princípio

"Cada pedaço de conhecimento deve ter uma representação única, não ambígua e autoritativa dentro de um sistema."

Exemplo Prático

- **Ruim**: Mesmo código em 10 lugares diferentes
- **Bom**: Uma função bem nomeada reutilizável
- **Resultado**: Mudanças em um único lugar



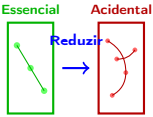
KISS: Keep It Simple, Stupid

Princípio

"A simplicidade deve ser um objetivo chave no design, e a complexidade desnecessária deve ser evitada."

Complexidade Acidental vs. Essencial

- **Essencial**: Complexidade inerente ao problema
- **Acidental**: Complexidade introduzida pela solução
- Boas práticas reduzem a acidental



Complexidade do Sistema

Fonte: Ilustração própria - Complexidade do sistema

YAGNI: You Ain't Gonna Need It

Princípio

"Implemente apenas funcionalidades que você realmente precisa, não as que você prevê que poderá precisar."

Armadilha Comum

- Desperdício de tempo com features não usadas
- Complexidade desnecessária
- Dificuldade de manutenção
- Custo aumentado sem benefício

Pergunte-se: Isso é necessário AGORA?

Benefícios do CI/CD

Integração Contínua

- Testes automatizados
- Builds frequentes
- Detecção precoce de bugs
- Merge diário no mínimo

Entrega Contínua

- Deploy automatizado
- Sempre pronto para release
- Menor risco de entrega
- Feedback rápido dos clientes

Valor de Negócio

- **Comercial**: Receita direta e redução de custos
- **Mercado**: Atração de novos clientes e vantagem competitiva
- **Eficiência**: Otimização de processos e time-to-market

Diferencial no Mercado de Trabalho

O que empregadores valorizam

- Capacidade de trabalhar em equipe
- Código de fácil manutenção e escalável
- Habilidade de documentar
- Entendimento de padrões e princípios

Pesquisa com Tech Recruiters (2024)

- 87% preferem desenvolvedores com boas práticas
- 76% pagam até 30% mais por esta habilidade
- 92% consideram em processos seletivos

Progressão na Carreira

Junior → Pleno

- Escreve código que funciona
- Foca em tarefas individuais
- Precisa de supervisão
- Resolve problemas imediatos

Pleno → Sênior

- Escreve código de fácil manutenibilidade
- Pensa no sistema completo
- Orienta outros devs
- Antecipa problemas futuros

O uso correto de boas práticas separa os níveis de experiência entre os devs.

Como Implementar no Dia a Dia

Passos Incrementais

1. **Revisar**: Analise seu próprio código criticamente
2. **Identificar**: Detecte code smells e problemas
3. **Refatorar**: Melhore gradualmente
4. **Automatizar**: Use ferramentas de análise
5. **Revisar**: Peça feedback constantemente

Ferramentas Úteis

- Linters (ESLint, Pylint, Checkstyle)
- Análise estática (SonarQube)
- Formatters (Prettier, Black)
- Testes automatizados

Métricas de Sucesso

Quantitativas

- ↓ Tempo de manutenção
- ↓ Bugs reportados
- ↓ Complexidade ciclomática
- ↑ Cobertura de testes
- ↑ Velocidade de desenvolvimento

Qualitativas

- ↑ Satisfação da equipe
- ↑ Confiança nas mudanças
- ↑ Clareza do código
- ↓ Estresse e sobrecarga
- ↑ Qualidade do produto

Por que Investir em Boas Práticas?

Resumo dos Benefícios

- **Econômico**: Reduz custos de desenvolvimento e manutenção
- **Técnico**: Aumenta qualidade e confiabilidade
- **Humano**: Melhora satisfação e colaboração
- **Profissional**: Diferencia no mercado de trabalho

Investimento que sempre retorna

Perguntas?

https://github.com/fmarquesfilho/bpp-2025-2