Boas Práticas de Programação (2025.2)

Projeto Integrado: MVP + Código Limpo + Refatoração

Prof. Fernando Figueira

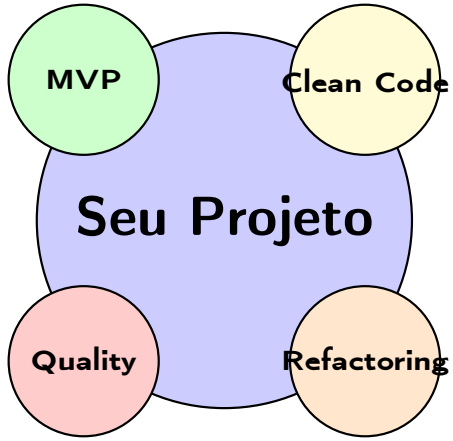
DIMAp - UFRN

Agosto de 2025

Projeto Integrado: MVP + Qualidade de Código

Objetivos do Projeto:

- **Entregar Valor**: MVP funcional
- **Código Limpo**: Nomenclatura e estrutura
- **Identificar Problemas**: Code smells
- **Melhorar Continuamente**: Refatoração



Nova Abordagem

Seu projeto será avaliado tanto pelo **valor entregue** quanto pela **qualidade do código**!

Visão do Produto: Template Prático

Template da Visão

Para [usuários-alvo]
Que [problema/necessidade]
O [nome do produto] **é um** [categoria]
Que [benefício principal]
Diferente de [alternativa existente]
Nosso produto [diferencial único]

Checklist da Visão

✓ Define usuário-alvo específico
✓ Identifica problema concreto
✓ Explicita valor único
✓ É inspiradora mas realista
✓ **Permite aplicar boas práticas de código**

Framework para MVP + Qualidade

1. Problema Core

Qual o **principal** problema que seu produto resolve?

2. Hipótese de Valor

"Acreditamos que [usuários] vão [comportamento] porque [benefício]"

3. Critérios de Qualidade

Como garantir que o código seja limpo e de fácil manutenção?

Exemplo - Sistema de Biblioteca

Problema: Estudantes perdem tempo procurando livros
MVP: Busca + Status + Empréstimo simples
Qualidade: Funções < 20 linhas, nomes descritivos

Princípios de Código Limpo no MVP

Nomenclatura:

- Nomes intencionais
- Evitar abreviações
- Usar termos do domínio
- Pronunciáveis

Funções:

- Pequenas (< 20 linhas)
- Uma responsabilidade
- Poucos parâmetros
- Nome descritivo

Formatação:

- Indentação consistente
- Espaçamento vertical
- Agrupamento lógico
- Linhas < 120 chars

Comentários:

- Apenas quando necessário
- Explicam "por quê"
- Mantidos atualizados
- Não redundantes

Code Smells: O que Procurar

Nível Método/Função:

- **Long Method**: $>$ 20-30 linhas
- **Long Parameter List**: $>$ 3-4 parâmetros
- **Duplicate Code**: Repetição
- **Dead Code**: Código não usado

Nível Classe:

- **Large Class**: Muitas responsabilidades
- **Data Class**: Só dados, sem comportamento
- **God Class**: Classe que faz tudo

Nível Estrutural:

- **Feature Envy**: Método interessado em outra classe
- **Inappropriate Intimacy**: Classes muito acopladas
- **Poor Naming**: Nomes ambíguos

Objetivo

Identificar pelo menos **3 code smells** diferentes no seu código!

Técnicas de Refatoração Essenciais

Extract Method:

- Transformar trecho em método
- Reduzir tamanho de funções
- Melhorar legibilidade

Rename Variable/Method:

- Nomes mais descritivos
- Eliminar ambiguidade
- Usar vocabulário do domínio

Introduce Parameter Object:

- Agrupar parâmetros relacionados
- Reduzir lista de parâmetros
- Criar classes de dados

Remove Duplicate Code:

- Extrair código comum
- Criar funções reutilizáveis
- Manter consistência

Meta

Aplicar pelo menos **3 refatorações** documentadas no seu projeto!

MoSCoW + Critérios de Qualidade

Must Have - MVP + Código Limpo

Should Have - Refatorações + SOLID

Could Have - Testes + Métricas

Won't Have - Features complexas desnecessárias

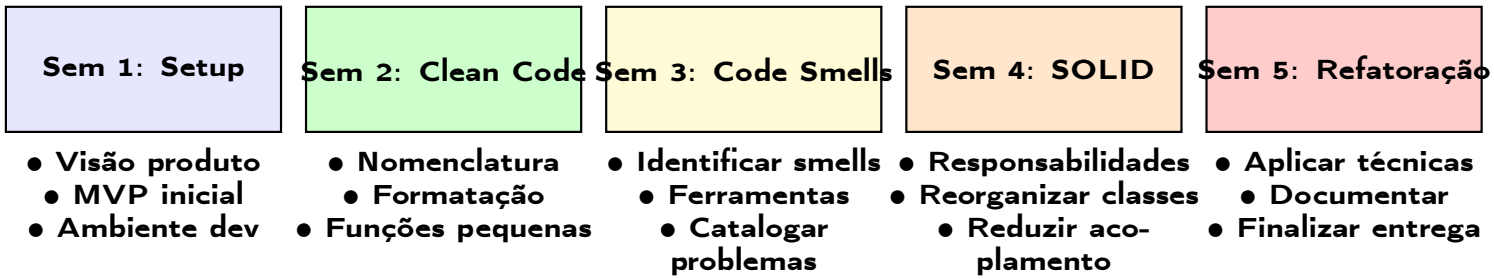
Lembre-se: Melhor um MVP simples com código de qualidade que features demais mal implementadas!

Backlog: Funcionalidade + Qualidade

Pri	User Story	Critérios de Qualidade	Est
P1	Como estudante, quero cadastrar tarefa	- Função cadastro $<$ 20 linhas - Nomenclatura descritiva - Validação de entrada	4h
P1	Como estudante, quero listar tarefas	- Separar lógica/apresentação - Função reutilizável - Tratamento de lista vazia	3h
P2	**Refatorar código duplicado**	- Identificar duplicação - Extrair função comum - Documentar refatoração	2h
P2	**Melhorar nomenclatura**	- Revisar nomes ambíguos - Aplicar convenções - Atualizar documentação	1h

Novidade: Itens de **qualidade** também entram no backlog!

Cronograma Integrado com Conteúdo



Cada semana: Desenvolvimento + aplicação dos conceitos da aula

Definition of Done com Qualidade

Sprint 1

- Funcionalidade OK
- Código compila
- **Nomes descritivos**
- **Formatação consistente**
- **Funções $<$ 20 linhas**

Sprint 2

- Tudo do Sprint 1 +
- **Code smells catalogados**
- **2+ smells corrigidos**
- Tratamento de erros
- Documentação inicial

Sprint 3

- Tudo do Sprint 2 +
- **SOLID aplicado**
- **3+ refatorações**
- **Relatório qualidade**
- Code review próprio

Evolução

Definition of Done evolui incorporando conceitos das aulas!

Ferramentas para Análise de Qualidade

Python:

- **pylint**: Análise estática completa
- **flake8**: Style guide enforcement
- **black**: Formatação automática
- **radon**: Métricas de complexidade

Java:

- **Checkstyle**: Convenções de código
- **PMD**: Code smells detector
- **SpotBugs**: Bug pattern detection

JavaScript:

- **ESLint**: Linting e qualidade
- **Prettier**: Formatação consistente
- **SonarJS**: Análise de qualidade

Multiplataforma:

- **SonarLint**: Plugin IDE
- **SonarCloud**: Análise online
- **CodeClimate**: Métricas contínuas

Recomendação

Use pelo menos **1 ferramenta** para identificar code smells automaticamente!

Métricas de Qualidade para Acompanhar

Métricas Básicas:

- **Linhas por Método**: $<$ 20-30
- **Parâmetros por Função**: $<$ 4
- **Complexidade Ciclomática**: $<$ 10
- **Duplicação**: $<$ 5

Indicadores de Qualidade:

- **Nomes descritivos**: 90
- **Comentários úteis**: Não redundantes
- **Code smells**: Identificados e catalogados
- **Refatorações**: Documentadas

Meta do Projeto

3+ code smells identificados e **3+ refatorações** aplicadas com documentação completa

Estrutura de Projeto + Qualidade

Estrutura Recomendada

```
projeto/
|- src/                    Código-fonte
|- tests/                  Testes
|- docs/                   Documentação
|- refactoring/            Análises e
refatorações
|- tools/                  Scripts de análise
|- README.md               Visão geral + qualidade
```

Pasta /refactoring/:

- code-smells-identified.md
- refactoring-log.md
- before-after-examples/
- quality-metrics.md

README.md deve incluir:

- Como executar análises
- Convenções de código
- Métricas atuais

Exemplo: Documentação de Refatoração

Template para refactoring-log.md:

Refatoração 1: Extract Method

Code Smell: Long Method em `process_data()` - 45 linhas
Técnica Aplicada: Extract Method
Justificativa: Método fazia validação + processamento + persistência
Resultado:

- `validate_input()`: 8 linhas
- `process_business_logic()`: 12 linhas
- `save_to_database()`: 6 linhas

Impacto: Melhor testabilidade e legibilidade

Documente: Cada refatoração com antes/depois e justificativa!

Critérios de Avaliação - Unidade 1

Qualidade do Código (30%)

- Aplicação de código limpo
- Nomenclatura e estrutura
- Formatação consistente

Code Smells (20%)

- Identificação correta
- Uso de ferramentas
- Catalogação detalhada

Refatorações (20%)

- Técnicas bem aplicadas
- Documentação completa
- Melhoria efetiva

MVP e Visão (15% cada)

- Funcionalidade e valor
- Planejamento coerente

Documentação Obrigatória Expandida

Documentos de Entrega:

- ✓ Visão do Produto (PDF, 2-3 páginas)
- ✓ Product Backlog (PDF)
- ✓ Relatório de Qualidade de Código (PDF, 3-4 páginas)
- ✓ Link para o repositório com o código-fonte completo
- ✓ Vídeo de apresentação (8-10 minutos)

Estrutura do Relatório de Qualidade:

1. Aplicação de Código Limpo (exemplos)
2. Code Smells Identificados (tabela + análise)
3. Refatorações Realizadas (antes/depois)
4. Ferramentas Utilizadas (prints + métricas)
5. Próximos Passos (melhorias planejadas)

Estrutura do Vídeo (8-10 min)

Roteiro Sugerido:

Min 1-2: Problema e Visão
- Apresentação do problema
- Solução proposta
- MVP definido

Min 3-4: Demo do MVP
- Funcionalidades principais
- Navegação pelo sistema
- Valor entregue

Min 5-6: Qualidade do Código
- Exemplos de código limpo
- Estrutura organizada
- Boas práticas aplicadas

Min 7-8: Code Smells e Refatoração
- Problemas identificados
- Técnicas aplicadas
- Melhorias alcançadas

Min 9-10: Conclusões
- Lições aprendidas
- Próximos passos

Sinais de um Bom Projeto

Sinais Positivos:

- ✓ MVP funcional e valioso
- ✓ **Código é legível**
- ✓ **Funções pequenas e focadas**
- ✓ **Nomes são descritivos**
- ✓ Code smells identificados honestamente
- ✓ Refatorações bem justificadas

Sinais de Atenção:

- ✗ Code smells ignorados
- ✗ Funções muito longas
- ✗ Nomes ambíguos (a, tmp, data)
- ✗ Código duplicado sem correção
- ✗ Refatorações superficiais
- ✗ Foco só nas features

Lembre-se

Código de qualidade hoje evita dor de cabeça amanhã!

Erros Comuns a Evitar

1. **Ignorar qualidade de código**
   - ✗ "Vou limpar depois"
   - ✓ "Código limpo desde o início"
2. **Code smells "falsos"**
   - ✗ Inventar problemas inexistentes
   - ✓ Usar ferramentas para detecção real
3. **Refatorações cosméticas**
   - ✗ Apenas renomear variáveis
   - ✓ Melhorias estruturais significativas
4. **Documentação insuficiente**
   - ✗ "O código se explica"
   - ✓ Justificar todas as decisões

Checklist Final Expandido

Código e Qualidade:

- ✓ Nomes são descritivos
- ✓ Funções < 20 linhas
- ✓ Código formatado consistente
- ✓ **3+ code smells identificados**
- ✓ **3+ refatorações aplicadas**
- ✓ Ferramentas de análise usadas

Documentação:

- ✓ Relatório de qualidade completo
- ✓ Refatorações bem documentadas
- ✓ Link para repositório do projeto
- ✓ Before/after examples
- ✓ Vídeo mostra código e qualidade
- ✓ README com padrões de qualidade

Autoavaliação Final

"Outro desenvolvedor conseguiria entender e manter meu código facilmente?"

Recursos de Apoio Disponíveis

Catálogos de Referência:

- **Code Smells**: `https://luzkan.github.io/smells/`
- **Refactoring**: Martin Fowler's catalog
- **Clean Code**: Robert C. Martin principles

Ferramentas Online:

- **SonarCloud**: Análise gratuita projetos públicos
- **CodeClimate**: Métricas de qualidade
- **Better Code Hub**: Compliance com guidelines

Apoio da Disciplina:

- **Atendimento**: Segundas 14h-16h (online)
- **Discord**: `https://discord.gg/bbMFJBQRT8`
- **GitHub**: `https://github.com/fmarquesfilho/bpp-2025-2`

Sucesso no Projeto!

Dúvidas? fernando@dimap.ufrn.br