Boas Práticas de Programação (2025.2)

Planejamento Estratégico de Projeto com Princípios Ágeis

Prof. Fernando Figueira

DIMAp - UFRN

Setembro de 2025

Por que Planejamento Estratégico?

- **Foco no Valor**: Resolver problemas reais
- **Gestão de Escopo**: Evitar feature creep
- **Entrega Incremental**: Feedback constante
- **Aprendizado Ágil**: Adaptar baseado em evidências

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
✓ Cabe no escopo acadêmico

Exemplo: Visão do Produto

Sistema de Controle de Gastos Universitário

Para estudantes universitários
Que têm dificuldade em controlar gastos mensais
O UniBudget **é uma** aplicação de controle financeiro
Que permite registro rápido e visualização de padrões
Diferente de apps complexos como Mobills
Nosso produto foca na simplicidade e contexto estudantil

- **Usuário**: Específico e bem definido
- **Problema**: Concreto e relevante
- **Diferencial**: Simplicidade + contexto

Framework para Definir MVP

1. Problema Core

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

2. Hipótese de Valor

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

3. Métricas de Sucesso

Como você saberá se funcionou?

Exemplo - Sistema de Biblioteca

Problema: Estudantes perdem tempo procurando livros
Hipótese: Vão consultar antes de ir à biblioteca
Métrica: Redução de idas "em vão"à biblioteca

Técnica MoSCoW para Definir Escopo

Must Have - Essencial para o MVP

Should Have - Importante, versão 2.0

Could Have - Desejável, backlog futuro

Won't Have - Explicitamente excluído

Dica: Se você tem dúvidas se algo é Must/Should, provavelmente é Should!

Exemplo: MoSCoW - Sistema de Biblioteca

Must Have (MVP)

- Cadastrar livro
- Listar livros
- Busca simples
- Marcar emprestado/disponível

Should Have (v2.0)

- Filtros avançados
- Histórico de empréstimos
- Dados do usuário

Could Have

- Sistema de reservas
- Notificações
- Estatísticas
- API externa

Won't Have

- Pagamentos de multa
- Integração com biblioteca física
- App mobile

Product Backlog: Organização Prática

Estrutura de cada item:

$$[\text{Prioridade}] + [\text{User Story}] + [\text{Critérios}] + [\text{Estimativa}]$$

User Story Template:

Como [tipo de usuário], quero [funcionalidade] para [benefício]
Exemplo: "Como estudante, quero buscar livros por autor para encontrar obras específicas rapidamente"

Critérios de Aceitação:

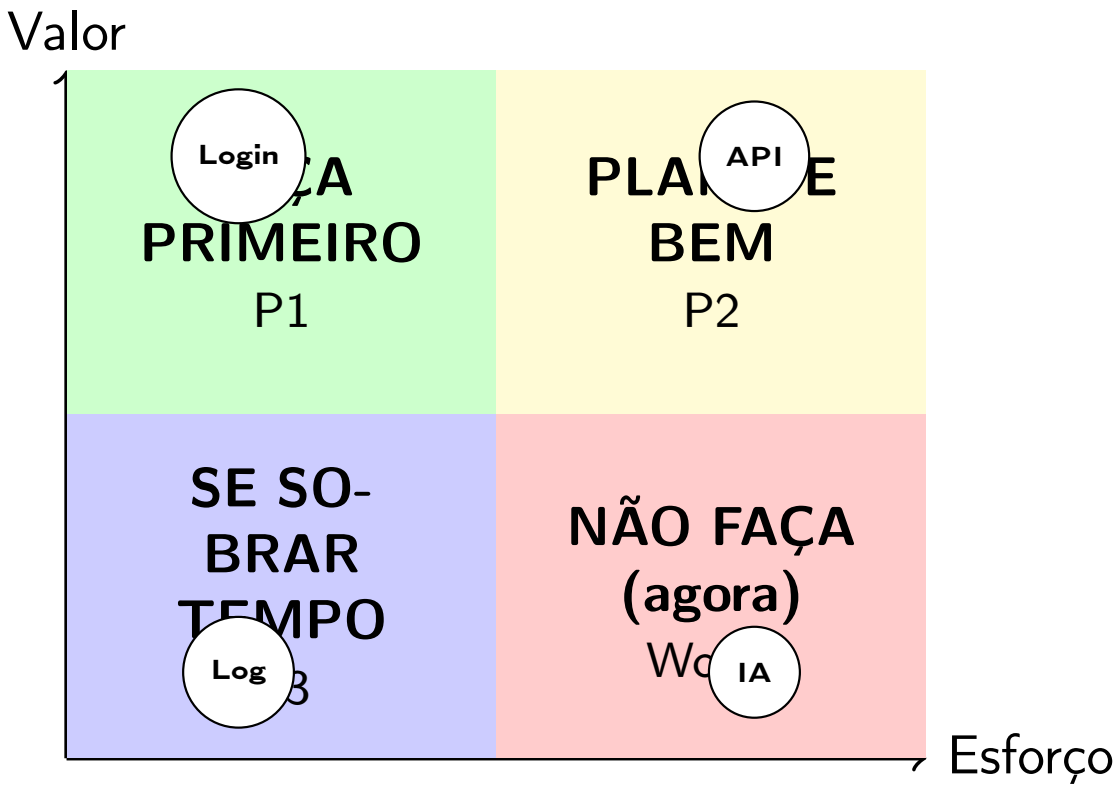
- Condições que a funcionalidade deve atender
- Cenários de teste implícitos
- Definição clara de "pronto"

Exemplo: Backlog Completo

Pri	User Story	Critérios de Aceitação	Est
P1	Como estudante, quero cadastrar uma nova tarefa para não esquecer	- Campos: título, descrição, data - Validação obrigatórios - Mensagem de confirmação	4h
P1	Como estudante, quero ver todas as tarefas para ter visão geral	- Lista ordenada por data - Indicador de urgência - Máximo 50 por tela	3h
P2	Como estudante, quero filtrar por status para focar no importante	- Filtros: todas, pendentes, feitas - Filtro persiste na sessão	2h

Observe: User stories focam no valor, não na implementação!

Matriz Valor x Esforço para Priorização



Cronograma: Setembro 2025 (Unidade 1)

Sem 1: Setup	Sem 2: Sprint 1	Sem 3: Sprint 2	Sem 4: Sprint 3	Sem 5: Entrega
• Visão do produto • Backlog inicial • Setup ambiente	• Funcionalidades P1 • Primeira versão • Testes básicos	• Completar P1 • Tratamento erros • Melhorar UX	• Funcionalidades P2 • Refatoração • Documentação	• Vídeo • Revisão docs • Upload SIGAA

Entrega: 02/10/2025 até 23:59

Definition of Done (DoD) Progressivo

Sprint 1

- Funcionalidade implementada
- Código compila
- Teste manual OK
- Commit descritivo

Sprint 2

- Tudo do Sprint 1 +
- Tratamento de erros
- Código comentado
- Refatoração inicial

Sprint 3

- Tudo do Sprint 2 +
- Testes automatizados
- Documentação completa
- Code review próprio

Dica

DoD evolui ao longo do projeto. Comece simples e melhore gradualmente!

Ferramentas para Gestão do Projeto

Gestão de Backlog:

- **Trello**: Visual e simples
- **GitHub Projects**: Integrado com código
- **Notion**: Robusto para docs
- **Google Sheets**: Rápido de usar

Controle de Versão:

- Commits frequentes
- Mensagens descritivas
- Branches por feature

Estrutura de Commits:

Exemplo de commit

```
git commit -m "[TIPO] Descrição"
```

Tipos:

- `FEAT`: nova funcionalidade
- `FIX`: correção de bug
- `DOCS`: documentação
- `REFACTOR`: refatoração
- `TEST`: testes

Estrutura de Pastas Recomendada

Estrutura Básica

```
projeto/
|- src/              Código fonte
|- tests/            Testes
|- docs/             Documentação
|- README.md         Visão geral
|- CHANGELOG.md      Mudanças
```

README.md essencial:

- Título do projeto
- Descrição breve
- Como instalar/executar
- Como usar
- Estrutura do código
- Autor e contato

Dica

README é o cartão de visitas do seu projeto. Invista tempo nele!

Sinais de um Bom Projeto

Sinais Positivos:

- ✓ Explica o valor em 30s
- ✓ MVP pode ser usado por alguém real
- ✓ Você está animado para desenvolvê-lo
- ✓ Escopo é realista para o tempo

Sinais de atenção:

- ✗ MVP muito complexo
- ✗ Foco na tecnologia, não no usuário
- ✗ Backlog com 50+ itens
- ✗ Problema vago ou inexistente

Lembre-se

Melhor um projeto simples bem executado que um complexo pela metade!

Erros Comuns a Evitar

1. **MVP muito ambicioso**
   - ✗ "Sistema completo de e-commerce"
   - ✓ "Catálogo simples com carrinho básico"
2. **Foco na tecnologia**
   - ✗ "Quero usar React, Node, MongoDB..."
   - ✓ "Como resolver o problema do usuário?"
3. **Backlog estático**
   - ✗ "Planejei tudo, não vou mudar"
   - ✓ "Vou adaptar conforme aprendo"

Mantra: Ação e feedback são mais valiosos que planejamento excessivo

Recursos de Apoio Disponíveis

Durante o Desenvolvimento:

- **Atendimentos**: Segundas 14h-16h (online) é bom?
- **Fórum da Disciplina**: Dúvidas técnicas e conceituais
- **GitHub do Curso**: Templates e exemplos
- **Material Complementar**: Artigos sobre MVP e backlog

Templates Disponíveis:

- Documento de Visão do Produto
- Planilha de Product Backlog
- Estrutura de README.md
- Checklist de Definition of Done

Importante

Não hesite em buscar ajuda! Melhor esclarecer dúvidas cedo que descobrir problemas na entrega.

Preparação da Entrega Final

Documentos Obrigatórios:

- Visão do Produto (PDF, 2-3 páginas)
- Product Backlog (PDF ou planilha)
- Vídeo de apresentação (5-8 minutos)

Estrutura do Vídeo:

- **Min 1-2**: Problema e visão do produto
- **Min 3-4**: Demonstração do MVP planejado
- **Min 5-6**: Backlog e priorização
- **Min 7-8**: Stack tecnológica e próximos passos

Formato dos Arquivos: `[TipoDoc]_[NomeProjeto]_[NomeAluno].pdf`

Checklist Final da Entrega

Antes de Enviar:

- Todos os arquivos no formato correto
- Links de vídeo acessíveis
- Documentos autoexplicativos
- Nomes seguem o padrão
- ZIP dentro do limite de tamanho

Autovaliação:

- Visão é clara e inspiradora
- MVP é realista e valioso
- Backlog está bem priorizado
- Vídeo comunica bem a ideia
- Estou orgulhoso do resultado

Entrega

SIGAA - Tarefa "Entrega U1"
Deadline: 02/10/2025 até 23:59
Formato: ZIP único com todos os arquivos

Próximos Passos

Esta Semana (02-08/09):

- Definir o problema que quer resolver
- Aplicar o template de visão do produto
- Criar backlog inicial com 10-15 itens
- Configurar ambiente de desenvolvimento

Próxima Semana (09-15/09):

- Iniciar Sprint 1 com funcionalidades P1
- Setup do repositório Git
- Primeira versão funcional (mesmo simples)
- Buscar feedback de colegas

Lembre-se: O objetivo é aprender aplicando conceitos ágeis, não criar o projeto perfeito!

Valeu e Bom Trabalho!

O projeto é uma oportunidade de:

Aplicar conceitos do curso na prática
Aprender com feedback e iteração
Desenvolver algo que você(s) tem orgulho

Dúvidas? fernando@dimap.ufrn.br