Instruções para o Projeto - BPP 2025.2

Visão Geral

O projeto da disciplina de Boas Práticas de Programação (BPP) será desenvolvido ao longo do semestre utilizando **princípios ágeis**. A primeira entrega (Unidade 1) consiste no **planejamento completo** do projeto, aplicando conceitos de visão de produto, definição de MVP e Backlog Priorizado e também uma versão preliminar (ainda inicial) do produto desenvolvido no projeto.

Fase 1 - Planejamento Estratégico (Unidade 1)

1. Visão do Produto

A **visão do produto** é a declaração estratégica que guia todo o desenvolvimento. Ela deve responder: "Por que este produto existe?" e "Qual o impacto desejado?"

Template da Visão do Produto:

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

Exemplo Prático - Sistema de Gestão Financeira Pessoal:

```
Para jovens universitários e profissionais iniciantes
Que têm dificuldade em controlar gastos e planejar orçamento
O FinanceTracker é uma aplicação de controle financeiro
Que permite registro rápido de gastos e visualização de padrões
Diferente de aplicativos complexos como o Mobills
Nosso produto foca na simplicidade e gamificação do controle financeiro
```

Checklist da Visão do Produto:

- [ ] Define claramente o usuário-alvo
- [ ] Identifica o problema específico a ser resolvido
- [ ] Explicita o valor único oferecido
- [ ] É inspiradora, mas realista para o escopo acadêmico
- [ ] Pode ser desenvolvida em 3-4 meses por 1-3 pessoas

---

2. Minimum Viable Product (MVP)

O MVP não é apenas "versão pequena", mas sim **o produto mais simples que valida sua hipótese principal de valor**. Ele deve ter três características essenciais:

2.1 Características de um MVP Eficaz:

1. **Viável**: Pode ser desenvolvido no tempo disponível
2. **Valioso**: Resolve o problema core do usuário
3. **Validável**: Permite testar a hipótese principal

2.2 Framework de Definição de MVP:

Problema Core: Qual o principal problema que seu produto resolve?

Hipótese de Valor: "Acreditamos que [usuários] vão [comportamento esperado] porque [benefício percebido]"

Métricas de Sucesso: Como você saberá se o MVP funcionou?

Exemplo Detalhado - Sistema de Biblioteca Digital:

Problema Core: Estudantes perdem tempo procurando livros disponíveis na biblioteca

Hipótese de Valor: "Acreditamos que estudantes vão consultar nosso sistema antes de ir à biblioteca porque saberão quais livros estão disponíveis"

MVP Funcionalidades:

- Cadastro simples de livros (título, autor, status)
- Busca por título
- Visualização de disponibilidade
- Sistema simples de empréstimo/devolução

Fora do MVP (para versões futuras):

- Sistema de reservas
- Notificações automáticas
- Histórico detalhado
- Integração com sistema acadêmico

2.3 Técnica MoSCoW para Definir MVP:

- **Must have**: Funcionalidades essenciais (seu MVP)
- **Should have**: Importantes, mas não críticas (versão 2.0)
- **Could have**: Desejáveis (backlog futuro)
- **Won't have**: Explicitamente excluídas desta versão

---

3. Product Backlog - Organização Prática

O backlog é sua ferramenta de gestão de escopo e prioridades. Ele deve ser **dinâmico** e **orientado a valor**.

3.1 Estrutura do Backlog:

Cada item deve seguir o formato:

```
[Prioridade] [Funcionalidade] – [User Story] – [Critérios de Aceitação] – [Estimativa]
```

3.2 Exemplo Completo de Backlog - Sistema de Tarefas Acadêmicas:

Pri	User Story	Critérios de Aceitação	Est	Sprint
P1	Como estudante, quero cadastrar uma nova tarefa para não esquecer de fazê-la	- Campos: título, descrição, data limite - Validação de campos obrigatórios - Confirmação de cadastro	4h	1
P1	Como estudante, quero ver todas as minhas tarefas para ter visão geral	- Lista ordenada por data limite - Indicador visual de urgência - Máximo 50 tarefas por tela	3h	1
P1	Como estudante, quero marcar tarefa como concluída para acompanhar progresso	- Checkbox/botão de conclusão - Mudança visual da tarefa - Confirmação da ação	2h	1
P2	Como estudante, quero editar uma tarefa para corrigir informações	- Formulário de edição - Manter dados anteriores - Validação de mudanças	3h	2
P2	Como estudante, quero filtrar tarefas por status para focar no que importa	- Filtros: todas, pendentes, concluídas - Filtro persiste na sessão	2h	2
P3	Como estudante, quero ver estatísticas das minhas tarefas para medir produtividade	- Total de tarefas - Taxa de conclusão - Gráfico simples	4h	3

3.3 Técnicas de Priorização:

Matriz Valor x Esforço:

```
Alto Valor + Baixo Esforço = Prioridade 1 (Faça primeiro!)
Alto Valor + Alto Esforço = Prioridade 2 (Planeje bem)
Baixo Valor + Baixo Esforço = Prioridade 3 (Se sobrar tempo)
Baixo Valor + Alto Esforço = Não faça (pelo menos agora)
```

---

4. Planejamento de Sprints

4.1 Estrutura:

Duração: 1-2 semanas (alinhado com conteúdo das aulas) **Objetivo**: Entrega incremental funcional **Cerimônias adaptadas**:

- Planning: 30 min (início da sprint)
- Review: Autoavaliação + demo para colegas
- Retrospective: Reflexão escrita sobre aprendizados

4.2 Cronograma (Unidade 1):

Semana 1 (02-08/09): Setup e Planejamento

- [ ] Definir visão do produto
- [ ] Criar backlog inicial
- [ ] Configurar ambiente de desenvolvimento
- [ ] Criar repositório Git
- [ ] Escolher stack tecnológica

Semana 2 (09-15/09): Sprint 1 - Core MVP

- [ ] Implementar funcionalidades P1 básicas
- [ ] Setup de testes básicos
- [ ] Primeira versão funcional (mesmo que simples)
- [ ] Documentação inicial

Semana 3 (16-22/09): Sprint 2 - Refinamento

- [ ] Completar funcionalidades P1
- [ ] Adicionar tratamento de erros
- [ ] Melhorar interface/UX
- [ ] Testes mais robustos

Semana 4 (23-29/09): Sprint 3 - Polimento

- [ ] Funcionalidades P2 (se tempo permitir)
- [ ] Refatoração e limpeza de código

- [ ] Documentação final
- [ ] Preparação da apresentação

Semana 5 (30/09-02/10): Entrega

- [ ] Vídeo de apresentação
- [ ] Revisão final dos documentos
- [ ] Upload no SIGAA

4.3 Definition of Done (DoD) por Sprint:

Sprint 1:

- [ ] Funcionalidade implementada
- [ ] Código compila sem erros
- [ ] Teste manual realizado
- [ ] Commit com mensagem descritiva

Sprint 2:

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

Sprint 3:

- [ ] Tudo do Sprint 2 +
- [ ] Testes automatizados (se aplicável)
- [ ] Documentação atualizada
- [ ] Code review próprio

---

5. Ferramentas e Técnicas Recomendadas para Gestão de Backlog

- **Trello**: Boards visuais (To Do, Doing, Done)
- **GitHub Projects**: Integrado com código

---

6. Templates para Entrega

6.1 Template de Visão do Produto:

```
# Visão do Produto: [Nome do Projeto]

## Declaração da Visão
[Use o template fornecido anteriormente]

## Problema a Ser Resolvido
- Contexto atual:
- Principais dores:
```

```
- Impacto do problema:

## Solução Proposta
- Abordagem escolhida:
- Diferencial principal:
- Benefícios esperados:

## Usuários-Alvo
- Perfil primário:
- Perfil secundário:
- Necessidades principais:

## Sucesso do Projeto
- Métricas de sucesso:
- Critérios de validação:
```

6.2 Template de Backlog:

```
# Product Backlog: [Nome do Projeto]

## MVP (Must Have)
| ID | User Story | Critérios de Aceitação | Estimativa |
|----|------------|------------------------|------------|
| 1  |            |                        |            |

## Versão 2.0 (Should Have)
| ID | User Story | Critérios de Aceitação | Estimativa |
|----|------------|------------------------|------------|
| 2  |            |                        |            |

## Futuro (Could Have)
| ID | User Story | Critérios de Aceitação | Estimativa |
|----|------------|------------------------|------------|
| 3  |            |                        |            |

## Explicitamente Excluído (Won't Have)
- Funcionalidade X: Motivo
- Funcionalidade Y: Motivo
```

---

7. Preparação dos Arquivos para Entrega

7.1 Checklist de Entrega:

Documentos Obrigatórios:

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

Documentos Complementares (Opcionais, mas recomendados):

- ☐ Documento de arquitetura (1 página)
- ☐ Cronograma de desenvolvimento
- ☐ Análise de riscos do projeto

7.2 Estrutura do Vídeo de Apresentação:

Minuto 1-2: Apresentação do problema e visão **Minuto 3-4**: Demonstração do MVP planejado **Minuto 5-6**: Explicação do backlog e priorização **Minuto 7-8**: Stack tecnológica e próximos passos

7.3 Critérios de Avaliação:

Critério	Peso	Detalhamento
Clareza da Visão	25%	Problema bem definido, solução coerente
Qualidade do MVP	25%	Escopo realista, foco no valor
Organização do Backlog	25%	Priorização justificada, user stories bem escritas
Apresentação	25%	Comunicação clara, demonstração eficaz

---

8. Dicas Práticas Finais

8.1 Erros Comuns a Evitar:

- **MVP muito complexo**: Comece menor do que imagina
- **Backlog estático**: Ele deve evoluir conforme você aprende
- **Foco na tecnologia**: O valor para o usuário vem primeiro
- **Planejamento excessivo**: Ação e feedback são mais valiosos

8.2 Sinais de um Bom Projeto:

- Você consegue explicar o valor em 30 segundos
- O MVP pode ser usado por alguém real
- As funcionalidades P1 resolvem o problema core
- Você está animado para desenvolvê-lo

8.3 Recursos de Apoio:

- **Office hours**: Segundas 14h-16h (online)
- **Fórum da disciplina**: Para dúvidas técnicas
- **GitHub do curso**: Exemplos e templates
- **Material complementar**: Artigos sobre MVP e backlog

---

9. Instruções de Entrega Final

9.1 Formato dos Arquivos:

- **Documentos**: PDF (máximo 10MB cada)
- **Vídeo**: MP4 ou link para YouTube/Drive
- **Nomenclatura**: `[TipoDoc]_[NomeProjeto]_[NomeAluno].pdf`

9.2 Envio:

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

9.3 Checklist Final:

- [ ] Todos os arquivos estão no formato correto
- [ ] Links de vídeo estão acessíveis
- [ ] Documentos são autoexplicativos
- [ ] Nome dos arquivos segue o padrão
- [ ] Arquivo ZIP está dentro do limite de tamanho