Fundamento da Metodologia de Desenvolvimento - Gerenciador de Contatos

Bem-vindo à equipa de desenvolvimento do **Gerenciador de Contatos**. Este documento descreve a cultura de trabalho, os processos e a metodologia ágil que utilizaremos para transformar código em solução.

O nosso objetivo é simular um ambiente real de desenvolvimento de software, utilizando práticas de mercado para garantir qualidade, organização e colaboração.

1. O Modelo de Trabalho (SCRUM Adaptado)

Utilizaremos uma adaptação da metodologia **Scrum**, focada na entrega contínua de valor e na inspeção constante do código.

1.1 Papéis

- **Product Owner & Tech Lead (Instrutor):** Define as prioridades, cria as demandas (Issues), avalia a qualidade técnica (Code Review) e aprova a integração do código.
- **Development Team (Alunos):** Responsáveis por escolher as tarefas, desenvolver a solução, garantir que o código funciona e solicitar a revisão.

1.2 Artefatos

- **Product Backlog (GitHub Issues):** A lista de todo o trabalho que precisa ser feito. Nada é desenvolvido sem estar registado numa Issue.
- **Sprint Backlog (GitHub Projects):** O quadro Kanban (To Do, In Progress, Done) que mostra visualmente o que cada aluno está a fazer no momento.
- **Incremento (Branch Main):** A versão funcional do software. O código na `main` é sagrado e deve estar sempre funcional.

---

2. Fluxo de Trabalho (Workflow)

Adotamos o **GitHub Flow**, um fluxo de trabalho leve e baseado em branches, ideal para equipas que implementam funcionalidades regularmente.

O Ciclo de Vida de uma Tarefa

O nosso panorama geral segue este caminho linear:

Issues → Branch → Commits → Pull Request → Revisão (Avaliação) → Merge

1. **Definição (Issue):** O problema a ser resolvido é documentado.
2. **Isolamento (Branch):** O trabalho é feito num ambiente isolado para não partir o código principal.
3. **Histórico (Commits):** O progresso é salvo em pequenos passos lógicos.

4. **Solicitação de Integração (Pull Request):** O aluno avisa que terminou e pede para juntar o seu código.
5. **Controlo de Qualidade (Code Review):** O instrutor valida o código.
6. **Integração (Merge):** O código é oficializado no projeto.

---

3. A Cultura de Code Review (Avaliação)

A "Revisão de Código" não é apenas uma correção de prova; é o momento de **mentoria técnica**.

- **O que é:** Um processo onde o instrutor analisa o código submetido no Pull Request linha por linha.
- **Objetivo:** Encontrar bugs, garantir padrões de arquitetura, sugerir otimizações e elogiar boas soluções.
- **Processo:**
    - Se o código estiver bom: O PR é aprovado e o *Merge* é feito. A Issue fecha-se automaticamente.
    - Se precisar de ajustes: O instrutor solicitará mudanças ("Request Changes"). O aluno deve corrigir na mesma branch e enviar novos commits.

---

4. Boas Práticas Gerais

- **Comunicação:** Use os comentários das Issues e PRs para tirar dúvidas.
- **Compromisso:** Nunca faça *commit* de código que não compila.
- **Atualização:** Mantenha o seu repositório local sempre sincronizado com o remoto.

> "Software funcional é a medida primária de progresso." - Manifesto Ágil