Regras e Padrões de Uso do Git

1. Objetivo

Este documento define um conjunto de regras, padrões e boas práticas para o uso do Git pela equipe, com o objetivo de facilitar o trabalho colaborativo, evitar conflitos desnecessários e aprimorar a organização da documentação e do código-fonte. O modelo adotado é baseado no Git Flow, uma estratégia amplamente utilizada para projetos que exigem controle de versões claro, estágios bem definidos de desenvolvimento e padronização no fluxo de trabalho.

---

2. Estrutura de branches

Branch Principal

- main: Reúne o código em desenvolvimento, integrando novas funcionalidades antes de ir para a produção.

Branches Auxiliares

- feature/nome-da-feature. Criadas para desenvolvimento de novas funcionalidades. Exemplo: feature/cadastro-cliente

Regras Gerais

- Toda branch deve ser removida após ser mergeada.

---

3. Padrões de Commit

Formato recomendado

<tipo>: descrição curta e objetiva

Tipos sugeridos

- feat: nova funcionalidade
- fix: correção de bug
- docs: alteração ou inclusão de documentação
- refactor: melhoria de código sem mudança de comportamento
- test: inclusão/alteração de testes
- style: mudanças de formatação (indentação, aspas, etc.)
- chore: mudanças menores (configurações, ajustes internos)

---

4. Fluxo de Trabalho

1. Criar uma branch para implementar uma nova funcionalidade;
2. Desenvolver e fazer commits seguindo os padrões.
3. Enviar as alterações para o repositório remoto:
4. Abrir um Merge Request (MR) solicitando revisão.
5. Após aprovação:
    a. Merge na main
    b. Deletar branch
6. Para entregas:
    a. Criar branch release
    b. Testar
7. Merge em main
8. Criar tag correspondente à versão
9. Para correções urgentes:
    a. Criar branch hotfix a partir de main

---

5. Padronização da Documentação

Todo documento deve conter:

- Título
- Pequena descrição do objetivo do documento
- Pequena conclusão do documento
- Seguir padrão de assuntos em tópicos
- Usar formato PDF sempre que possível.
- Diagramas devem incluir exportação em imagem (png/jpg/svg)

---

6. Arquivo .gitignore

- Arquivos temporários, caches e pastas geradas automaticamente devem ser ignorados.
- Documentos de build, objetos binários e dependências locais devem permanecer fora do controle de versão.