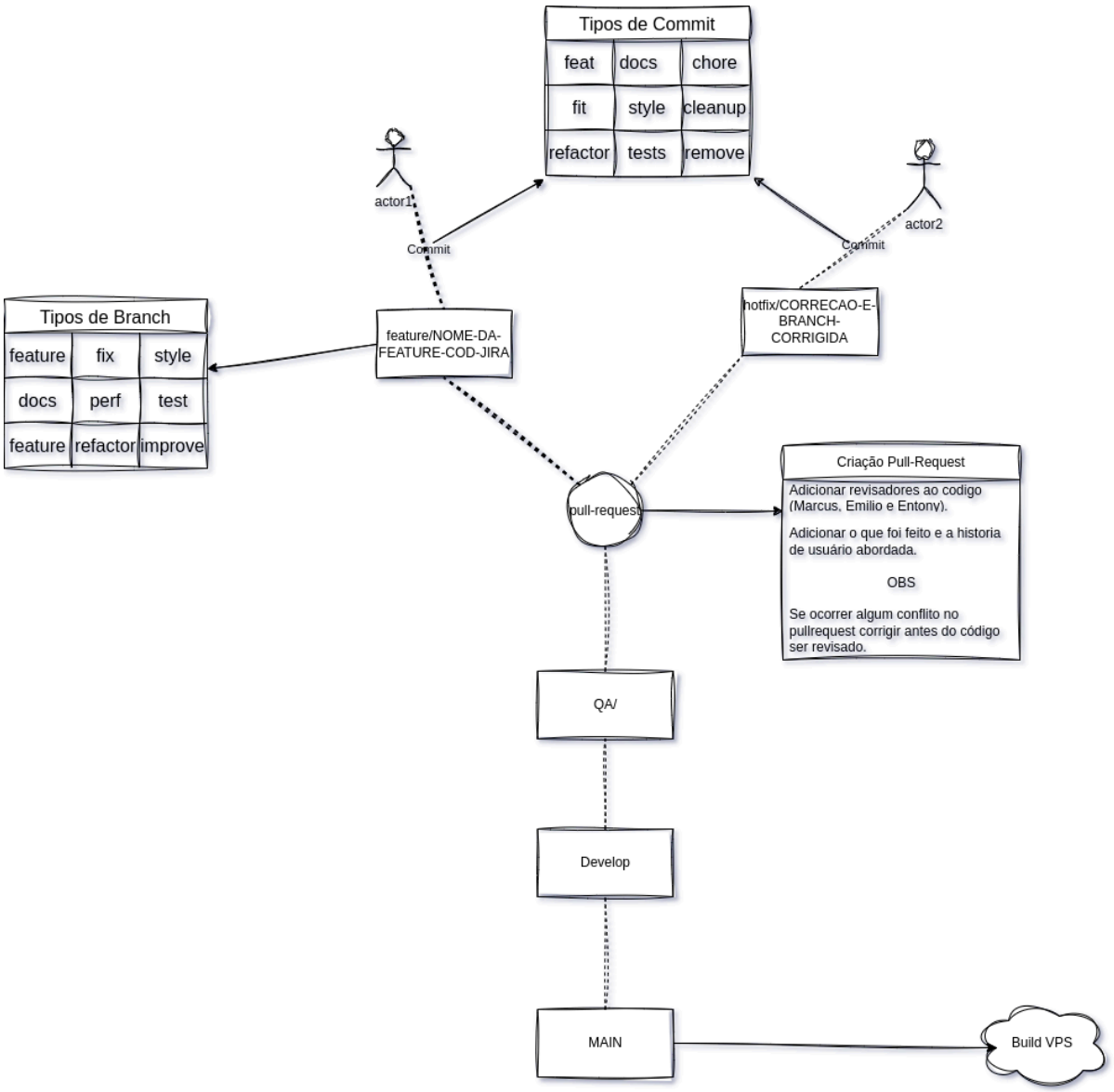
Guia de Git Flow do Projeto

Diagrama Flow



1. Introdução

Este guia tem como objetivo padronizar o uso do Git no projeto, estabelecendo um fluxo de trabalho eficiente, organizado e seguro. A adoção dessas boas práticas facilita a colaboração entre os membros da equipe, melhora o rastreamento de funcionalidades e bugs, e garante uma integração contínua mais estável.

---

2. Tipos de Branch

As branches devem ser criadas seguindo os padrões abaixo:

- `feature/NOME-DA-FEATURE-COD-JIRA`: Utilizada para desenvolvimento de novas funcionalidades.
- `fix/DESCRICAO-DO-FIX`: Para correção de bugs menores.
- `hotfix/CORRECAO-E-BRANCH-CORRIGIDA`: Para correções críticas em produção.
- `refactor/`: Refatoramento de código sem mudança de comportamento.
- `style/`: Mudanças de estilo, formatação.
- `test/`: Adição ou edição de testes.
- `docs/`: Atualização de documentação.
- `perf/`: Melhorias de performance.
- `improve/`: Outras melhorias não classificadas como features.

---

3. Padrão de Commits

Utilizaremos o formato de commits semânticos, baseado no Conventional Commits:

- `feat`: Adiciona nova funcionalidade.
- `fix`: Corrige um bug.
- `docs`: Mudanças na documentação apenas.
- `style`: Mudanças de estilo (espaço em branco, ponto e vírgula, etc).
- `refactor`: Refatoramento de código.
- `test`: Adiciona ou corrige testes.
- `perf`: Melhoria de performance.
- `chore`: Tarefas auxiliares sem impacto direto no código.
- `cleanup`: Limpeza de código morto ou não utilizado.
- `remove`: Exclusão de arquivos ou funcionalidades.

---

4. Processo de Pull Request

Ao finalizar uma branch, deve-se abrir um Pull Request (PR) para o merge:

- Adicionar os revisores: Marcus, Emílio e Entony.
- Na descrição do PR, incluir:
    - O que foi feito.

    - A história de usuário abordada.
- Se houver conflitos, eles devem ser resolvidos **antes** da revisão.

---

5. Ciclo de Integração

Fluxo de merge:

1. Branch criada a partir de `develop` ou `hotfix`, conforme o tipo.
2. PR é criado e revisado.
3. Merge feito na branch `QA` para testes.
4. Após validação, merge na `develop`.
5. Quando estiver pronto para produção, merge na `MAIN`.
6. Merge na `MAIN` dispara o build na VPS via CI/CD.

---

6. Responsabilidades do Time

- Cada dev é responsável por:

    - Criar branches seguindo a nomenclatura.
    - Escrever commits claros e informativos.
    - Testar antes de abrir PR.
    - Resolver conflitos.
    - Acompanhar feedbacks nas revisões.

---

7. Boas Práticas

- Nunca fazer commit direto em `MAIN` ou `develop`.
- Atualize sua branch com `rebase` ou `merge` antes de criar o PR.
- Revise o PR dos colegas.
- Utilize mensagens de commit que ajudem na rastreabilidade.
- Apague branches antigas após merge.

---

8. Links Úteis

Diagrama:
https://drive.google.com/file/d/1E_FZEvwXFI3C6VqCJra4HUVxEVq04u_f/view?usp=sharing

Padrão de Commits:
https://github.com/iuricode/padroes-de-commits?tab=readme-ov-file

Padrão de Criação de Branchs:
https://github.com/JuniorLima22/padroes-e-nomenclaturas-no-git