# Padrão de Commit e Branches

Adotamos o padrão de commits descrito na documentação oficial:
https://github.com/iuricode/padroes-de-commits

## GitFlow

- **Branches principais**:
  - **main**: Contém apenas as versões oficiais do projeto (releases).
  - **develop**: Contém o código em estado de desenvolvimento.

- **Criação de nova funcionalidade**:
  Para desenvolver uma nova funcionalidade, crie uma nova branch de feature, seguindo a nomenclatura: `feat/nome-da-funcionalidade`.
  Ao concluir o desenvolvimento, crie um PR para a branch `develop`.

- **Preparação de Release**:
  Quando a branch `develop` acumular funcionalidades suficientes para um novo release (ou a data de release estiver próxima), crie uma nova branch de `release` a partir da `develop`.
  A partir desse ponto, somente as seguintes ações são permitidas na branch `release`:
  - Correções de bugs.
  - Geração de documentação.
    Outras tasks relacionadas ao release.
    **Não devem ser adicionadas novas funcionalidades na branch de release**.
    Após a conclusão do release, faça o merge da branch `release` tanto na `main` quanto na `develop`, e exclua a branch `release`.

- **Correção de bugs (Hotfixes)**:
  Se um bug for encontrado na branch `main` após o merge, crie uma branch `hotfix/nome-do-bug` a partir da `main`.
  Realize as correções e, em seguida, faça o merge tanto na `main` quanto na `develop`.
  Caso o bug seja encontrado na branch `release`, crie uma branch `hotfix/nome-do-bug` a partir da `release`.
  Após a correção, faça o merge apenas na branch `release`.

# Padrão de Pull Requests (PR)

Título da issue/task relacionada | (link Issue) |

O que foi feito:
Mudanças

Passos para testar:
1.

Checklist
- ☐ Atualizado com a develop
- ☐ Dentro dos critérios de aceitação
- ☐ Adicionado novas dependências
- ☐ Código limpo e comentado
- ☐ Dentro dos padrões de Projeto

# Padrões de Projeto

**Frontend**

- Para cada componente, incluindo páginas, crie uma pasta dedicada que contenha tanto o componente em si quanto seu arquivo de estilo (module.css). Exemplo de estrutura: `/Home -> Home.tsx e Home.module.css`.

- **Responsividade**:
  Todas as unidades devem ser responsivas, utilizando unidades `REM`. Sempre que possível, prefira usar `flex` e `grid` para o posicionamento de componentes, evitando o uso de `margin` e `position: absolute`.

- **Cores**:
  Utilize as variáveis de cores definidas no arquivo `/styles/colors.css`. Exemplo de uso:
  `var(--yellow)`.

- **Composição de Componentes**:
  Busque sempre criar componentes pequenos, independentes e reutilizáveis. Evite componentes extensos e complexos.
  Antes de criar novos componentes, verifique se o framework ShadCN já oferece alguma solução. Se criar novos componentes, considere a possibilidade de variações para reutilização futura.