Guia de Contribuição para o Repositório

Modelo de Branches

Nossa organização de branches segue o modelo Git Flow, que utiliza as seguintes branches principais:

1. **main**: Esta é a branch de produção, onde o código está sempre em estado pronto para ser lançado. Nenhum desenvolvimento direto é feito nesta branch.
2. **develop**: Esta é a branch de integração principal, onde todo o desenvolvimento ocorre antes de ser preparado para a produção.

Branches de Feature

Para desenvolver novas funcionalidades, utilizamos branches de feature que são derivadas da branch **develop**. O nome das branches de feature segue o formato:

`feature/nome-da-issue`

Processo de Merge

1. **Criação da Branch de Feature**: Crie uma nova branch a partir da **develop**.
2. **Desenvolvimento**: Realize o desenvolvimento na branch de feature. Certifique-se de fazer commits frequentes.
3. **Pull Request (PR)**: Quando a feature estiver pronta, crie um Pull Request (PR) da branch **feature/nome-da-issue** para a branch **develop**.
4. **Revisão de Código**: Um ou mais colaboradores devem revisar e aprovar o PR antes do merge. Isso garante a qualidade e a integridade do código.
5. **Merge**: Após a aprovação, a branch **feature/nome-da-issue** pode ser mesclada na **develop**.

Padrão de Commits

Utilizamos um padrão de commits específico para manter um histórico de commits claro e organizado. Aqui estão os tipos de commits que usamos:

- **feat**: Adição de uma nova funcionalidade.
- **fix**: Correção de um bug.
- **chore**: Tarefas de manutenção que não afetam o código de produção diretamente (ex.: atualização de dependências, configuração de ferramentas).
- **refactor**: Mudanças no código que não corrigem bugs nem adicionam funcionalidades, mas melhoram a estrutura ou a legibilidade.
- **style**: Mudanças que não afetam o significado do código (ex.: formatação, pontuação, espaços em branco).

Estrutura da Mensagem de Commit

A mensagem de commit deve seguir o formato:

```
<tipo>: descrição breve
```

Exemplos de Mensagens de Commit

feat:

```
feat: adiciona autenticação de usuários com JWT
```

fix:

```
fix: corrige erro de autenticação na página de login
```

chore:

```
chore: atualiza dependências do projeto
```

refactor:

```
refactor: move lógica de autenticação para um serviço separado
```

style:

```
style: corrige indentação e remove espaços em branco desnecessários
```

Processo de Revisão e Aprovação

1. **Criação do PR**: Quando a feature estiver pronta, crie um Pull Request da branch de feature para a branch **develop**.
2. **Revisão de Código**: Um ou mais colaboradores devem revisar o código. Utilize comentários para sugerir melhorias ou apontar problemas.
3. **Aprovação**: Após as correções e melhorias sugeridas, aprove o PR.
4. **Merge**: Após a aprovação, realize o merge do PR na branch **develop**. Delete a branch de feature após o merge para manter o repositório limpo.

Conclusão

Seguir este guia ajudará a manter nosso repositório organizado, nosso histórico de commits claro e nosso código de alta qualidade. Obrigado pela colaboração!