Guia de Boas Práticas Git para o Projeto NativeNews

Este guia visa auxiliar os desenvolvedores a utilizar o Git de forma eficiente e consistente, garantindo um histórico de commits limpo e informativo, facilitando o acompanhamento do desenvolvimento e a colaboração entre a equipe.

1. Fluxo de Trabalho:

- **Fork:** Cada desenvolvedor possui um fork pessoal do repositório principal no GitHub.
- **Branching:** Trabalhe em branches separadas para cada nova funcionalidade ou correção de bug, a partir da branch `development`.
- **Commits:** Faça commits frequentes e descritivos para registrar as alterações. Preferencialmente ao criar qualquer mudança no código, se você criou uma página nova > commit, se editou algo > commit.
- **Pull Requests:** Crie Pull Requests para integrar as suas branches na `development`, as mudanças serão integradas na `main` após a revisão do código.
- **Merging:** Após a aprovação, o líder da equipe irá realizar o merge da sua Pull Request para a `development`

2. Convenções de Commits:

A descrição dos commits deve ser concisa, clara e informativa, seguindo um padrão definido. Utilize a sintaxe padrão em inglês para facilitar a compreensão por toda a equipe.

Formato:

```
<tipo>(<escopo>): <descrição>

<opcional: corpo da mensagem com mais detalhes>

<opcional: links relacionados>
```

Tipos de Commits:

- **feat:** Nova funcionalidade.
- **fix:** Correção de bug.
- **docs:** Documentação.
- **refactor:** Refatoração de código.
- **test:** Adicionando ou atualizando testes.
- **style:** Alterações de estilo de código (ex: formatação).
- **chore:** Tarefas de manutenção (ex: scripts de build, configuração).
- **revert:** Revertendo um commit anterior.

Escopo:

- **Nome do componente ou área do código afetada.**
- **Exemplos:** `auth`, `database`, `frontend`, `api`.

Descrição:

- **Uma frase concisa que resume a mudança.**
- **Comece com letra maiúscula e termine com ponto final.**
- **Evite frases muito longas.**

Exemplo:

```
feat(auth): Implementa autenticação de usuário

Adiciona um novo endpoint para autenticação de usuário usando JWT.
O usuário fornece email e senha, e recebe um token JWT caso a autenticação seja bem-sucedida

Relacionado a #123
```

3. Boas Práticas Adicionais:

- **Commits Pequenos:** Faça commits com alterações focadas e pequenas.
- **Mensagens Relevantes:** Escreva mensagens que expliquem claramente o que foi modificado e porquê.
- **Revisão do Código:** Antes de enviar um Pull Request, revise o seu código para garantir que ele atenda aos padrões do projeto.
- **Atualização da Branch:** Mantenha sua branch local sincronizada com a branch `develop` para evitar conflitos.
- **Resolvendo Conflitos:** Se ocorrer algum conflito, resolva-o com cuidado e dê commit nas alterações.

4. Ferramentas Úteis:

- **GitHub:** Utilize as funcionalidades do GitHub para facilitar o gerenciamento de Pull Requests e revisão de código.
- **GitKraken:** Considere utilizar o GitKraken para gerenciar o fluxo de trabalho do projeto.
- **Commitizen:** Use uma ferramenta para automatizar o processo de criação de commits seguindo as convenções definidas.
- **Gitlens:** É um plugin do vscode para facilitar a visualização da arvore de commits.

Lembre-se: A comunicação clara e eficiente através do Git é fundamental para o sucesso do projeto. Seguir estas boas práticas ajudará a manter o histórico do código organizado, a facilitar a colaboração e a garantir a qualidade do software.