GUIA DO GIT - PADRÃO DE MENSAGENS NO COMMIT

API 2025-1

A padronização de mensagens de commits são importantes pois, além de ajudar na compreensão do histórico de commits, facilita a criação de ferramentas automatizadas baseadas na especificação. Existe um padrão convencional de mensagens (https://www.conventionalcommits.org/en/v1.0.0/). Porém, visando simplificar:

<tipo> (<id_demanda1>, <id_demanda2>, ..., <id_demandaN>): <descrição da entrega feita no commit>

<tipo>	***Descrição***	***Exemplo***
<feat>	Quando da adição de um recurso, uma "feature" (funcionalidade)	feat (AB-1243, AB-56): Implementação dos repositories usados nas operações com as tabelas de variações climáticas
<fix>	Correção de um bug	fix (#45): Correção do componente de seleção de município
<docs>	Atualização de documentação	docs (#45): inclusão de diagrama de modelo de BD para a aplicação
<style>	Mudança de formatação, sem afetar o código	style (AB-1243, AB-56): ajuste de nomes de variáveis para o padrão camelCase
<refactor>	Refatoração do código, sem alterar funcionalidade	Seguir exemplos anteriores, alterando o tipo, IDs e descrições correspondentes
<test>	Adiciona ou modifica testes	Seguir exemplos anteriores, alterando o tipo, IDs e descrições correspondentes
<chore>	Atualizações menores que não impactam diretamente a funcionalidade do código	Seguir exemplos anteriores, alterando o tipo, IDs e descrições correspondentes

<id_demandaN> - Identificador da demanda criada na ferramenta de gestão de Stories/Tasks que o Time estiver usando (Github Issues, Jira Software, GitLab Issues, ClickUp, etc, podendo estar entre 1 e N.

<descrição da entrega feita no commit> - Descrição clara sobre o que está sendo entregue no commit criado e enviado para o Git

Boas práticas para Uso do Git

https://pt.linkedin.com/pulse/boas-pr%C3%A1ticas-para-git-leticia-coelho:

- Antes de iniciar os commits no projeto GIT, crie um arquivo .gitignore que irá ignorar arquivos de logs, binários e arquivos com senhas. Você pode encontrar vários modelos na internet.
- Nunca realize commit direto na Main. Essa prática poderá facilmente ocasionar em perda de código quando você atuar em conjunto com outros desenvolvedores. Utilize a criação de branchs.
- Combine com o time o padrão que deverá ser utilizado para nomes de branchs e commits.
- Crie um arquivo README com versões, regras combinadas, modo de executar e testar o projeto.
- Crie uma branch para cada tarefa. Separar as implementações em branchs, facilita a localização de bugs.
- Especifique a referência única (número da tarefa, nome da tarefa) no nome da branch ou em cada commit.
- Realize pequenos commits, sempre mantendo a última versão mais estável para o commit atual.
- Seja direto nos commits, pequenas modificações com pequenos comentários. Utilize o imperativo, por exemplo: “Adicionado box-shadow no botão da home”.
- Ao criar um PR, resumidamente descreva o que foi implementado. Se não houver outras ferramentas que descrevam os testes, esse também é um ótimo espaço para descrever todos os cenários de teste.
- Sempre utilize o ciclo de Kanban, ou seja, aguarde *code review* e análise do código. Preferencialmente não aprove o seu próprio PR, sempre que possível realize o processo completo.