Os padrões de commit no Git ajudam a manter um histórico limpo e organizado. Um dos padrões mais usados é o **Conventional Commits**, que define um formato padronizado para as mensagens de commit.

**🔹 Formato do Conventional Commits**

<tipo>(<escopo opcional>): <mensagem breve>

<descrição opcional>

**🔹 Tipos Comuns de Commit**

| **Tipo** | **Uso** |
| --- | --- |
| **feat** | Adiciona uma nova funcionalidade |
| **fix** | Corrige um bug |
| **chore** | Alterações na configuração do projeto (sem impacto no código-fonte) |
| **docs** | Alterações na documentação |
| **style** | Ajustes de formatação (espaços, indentação, etc.), sem mudar o código |
| **refactor** | Refatoração do código sem mudar comportamento |
| **test** | Adição ou modificação de testes |
| **perf** | Melhorias de desempenho |
| **build** | Alterações no processo de build ou dependências |
| **ci** | Alterações em arquivos de integração contínua (CI/CD) |
| **revert** | Reverte um commit anterior |

**🔹 Exemplos de Commit**

git commit -m "feat: adicionar botão de busca na página inicial"

git commit -m "fix(login): corrigir erro de validação de senha"

git commit -m "docs(readme): adicionar exemplo de uso da API"

git commit -m "refactor: remover código duplicado na função de cálculo"

git commit -m "style: corrigir indentação do arquivo app.js"

**🔹 Recomendações**

✅ Seja claro e objetivo.  
✅ Use o tempo verbal no imperativo (exemplo: "adicionar" ao invés de "adicionado").  
✅ Utilize **escopo** quando necessário para indicar a área afetada.  
✅ Separe commits por tipo de alteração (não misture refatoração e feature no mesmo commit).

Você quer que eu te ajude a configurar um lint para forçar esse padrão no seu projeto? 🚀