Padrões do Projeto:

Código:

1. Organização

3 tags: header, main e footer (colocar classes cabecalho, principal e rodape)

```
<!-- Cabeçalho-->
<header class="cabecalho">···
</header>

<!-- Parte principal -->
<main class="principal">···
</main>

<!-- Rodapé -->
<footer class="rodape">···
</footer>
```

Se caso quiser colocar algo dentro dessas tags, respeite a hierarquia

```
        <div class="principal-mensagem">
            <h1>Olá, esse é o meu site</h1>
            <h2>Bom dia</h2>
            <button onclick="bomdia()">Clique aqui</button>
        </div>
```

Exemplo: Se caso quiser colocar uma navegação do cabeçalho, crie respeitando a hierarquia

Nome de classe: class=”cabecalho-navegacao”

2. Separação de responsabilidades

- **CSS** (visual) está separado em arquivos próprios (`global.css`).
- **JavaScript** (funcionalidade) também separado (`global.js`).

Arquivos global devem ser usados em todas as páginas, geralmente são colocadas estilizações ou funções que são usadas em todo o site (cabeçalho, rodapé, tema escuro ao click etc.)

3. Padrão de nomeação de classes

- Crie nomes em português, que SEJAM coerentes, seguindo a hierarquia
- Exemplo: principal, principal-mensagem, principal-poster, principal-imagem-texto, rodape-navegacao

4. Comentários

- TODOS os blocos de código devem ter um comentário explicando para que ele serve e o que faz (divs, functions, declaração de variáveis, loops, etc).
- HTML: <!-- --!>
- CSS: /* */
- JS: // Comentário de Linha ou /* */ comentário de bloco

Comentários:

<!-- Este é um comentário em **HTML** -->

/* Este é um comentário em **CSS** */

// Este é um comentário de uma linha em JavaScript

/*

Este é um comentário

de várias linhas em **JavaScript** */

Commits:

fix

Você **consertou um erro** (bug) no código.
Exemplo: O botão não funcionava, agora funciona.

feat

Você **criou algo novo** no projeto.
Exemplo: Adicionou uma nova página, um novo botão, um novo recurso.

docs

Você **mexeu só na documentação**.
Exemplo: Atualizou o README, escreveu instruções, mas não mexeu no código que roda.

style

Você **arrumou a aparência do código**, mas sem mudar o que ele faz.
Exemplo: Tirou espaço sobrando, arrumou indentação, colocou ponto e vírgula.

refactor

Você **reestruturou o código** para deixar melhor, mas **sem mudar o que ele faz**.
Exemplo: Deixou o código mais limpo, mais rápido, mais bonito.

build

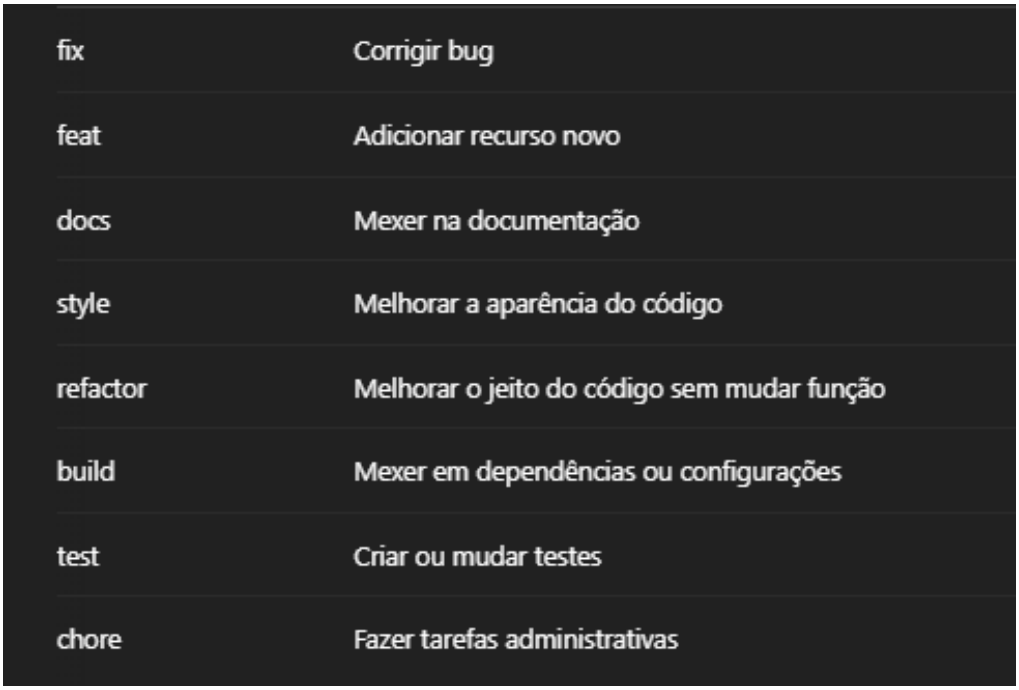
Você **mexeu em coisas de configuração ou de dependências**.
Exemplo: Instalou uma biblioteca nova, arrumou o arquivo de build.

test

Você **criou ou mudou testes**.
Exemplo: Fez um teste para ver se uma função está funcionando, mas **não mudou** o funcionamento da função em si.

chore

Você **fez tarefas de "manutenção"** que não afetam o código principal.
Exemplo: Atualizou o `.gitignore`, mexeu nas configurações do projeto.



fix	Corrigir bug
feat	Adicionar recurso novo
docs	Mexer na documentação
style	Melhorar a aparência do código
refactor	Melhorar o jeito do código sem mudar função
build	Mexer em dependências ou configurações
test	Criar ou mudar testes
chore	Fazer tarefas administrativas