16 — Template de `CLAUDE.md`

16 — Template de `CLAUDE.md`

> Arquivo pronto para colar na raiz do projeto novo. Substitua tudo entre `{{ }}` e apague o que não se aplicar. Inclui a seção de harness — ver 18 e 19.

Como manter

O `CLAUDE.md` é carregado automaticamente em toda sessão. Ele é **índice**, não enciclopédia: aponta para os guias, não os duplica. Duplicar cria duas versões da verdade — é assim que um `CLAUDE.md` diverge do código com o tempo (ver anti-padrões em 13).

Regra: se um fato pode mudar sem que ninguém lembre de atualizar o `CLAUDE.md`, ele não deveria estar aqui — deveria estar num guia, referenciado daqui.

Template

```
# CLAUDE.md
Guia para o Claude Code (claude.ai/code) trabalhar neste repositório.
Projeto: **{{NOME_DO_PROJETO}}** — SPA React que fala com backend Curio via `@curio/client`.
Stack, arquitetura e convenções detalhadas vivem em `harness/guides/` — ver tabela abaixo.
---
```

Comandos

```bash
npm run dev              # ambiente dev + Vite (porta {{PORTA}})npm run start:homolog    # ambiente de homologação
npm run build            # lint + tsc + build de produçãonpm run lint             # ESLint, zero warnings
npm run format           # Prettier
npx tsc --noEmit         # type-check completo
```

Node >= {{VERSAO_NODE}} (fixado em `.node-version`). Ative antes de qualquer comando — o `pre-commit`
valida e barra o commit se estiver errado.

Não há suíte de testes. A verificação é `lint` + `tsc` + `build` + teste manual da tela contra o
backend real. Não afirme que algo "passou nos testes".

Guide Files

Toda a documentação de harness vive em `harness/`. `CLAUDE.md` e `init.sh` ficam na raiz — são os pontos de entrada.

```
harness/
├── guides/                  — referência técnica — versionado
├── _examples/│   ├── state/                — feature_list.example.json, progress.example.md, session-handoff.example.md
│   ├── handoffs/              — handoff.example.md
│   └── user_preferences.example.md├── <branch>/                — tudo que a branch produziu — VERSIONADO│   ├── state/                 — feature_list.json, progress.md, session-handoff.md
│   ├── plans/                  — planos de implementação│   ├── specs/                   — specs de design
│   └── handoffs/                 — handoffs de sessão└── user_preferences.md       — preferências do dev — ÚNICO ARQUIVO LOCAL (gitignored)
```

`<branch>` é o nome literal da branch atual (`git rev-parse --abbrev-ref HEAD`). **Só `harness/user_preferences.md` é local.** Tudo dentro de `harness/<branch>/` é versionado — se `harness/<branch>/state/` não existir ainda, **crie a pasta a partir de `_examples/`** (mecânico); não
invente conteúdo de `feature_list.json`/`progress.md`.

```
### Guias técnicos
- `harness/guides/commands.md` — build, run, ambientes, variáveis- `harness/guides/architecture.md` — camadas do front, fluxo até o backend, autenticação- `harness/guides/curio-framework-guide.md` — sessão, caso de uso, `useCurioMutation`, React Query
- `harness/guides/dev-environment.md` — Node/fnm, `.env`, acesso aos servidores- `harness/guides/husky-git-hooks.md` — o que cada hook faz e como configurar- `harness/guides/gitlab-access.md` — autenticação com a API do GitLab {{REMOVER_SE_NAO_USA}}
### Estado de features
- `harness/<branch>/plans/*.md` — planos task-by-task por feature- `harness/<branch>/specs/*.md` — specs de design
- `harness/<branch>/handoffs/*.md` — handoffs por feature (versionado)
Quando existir plano/spec da feature ativa, ele tem precedência sobre o `feature_list.json` genéricopara o detalhe fino — mas atualize os dois ao final da sessão. Um plano/spec fica na branch onde
nasceu — não realoque se o trabalho continuar em outra branch depois.
### Sem histórico compartilhado entre branches
Não existe mais `global_feature_list.json`. Cada branch é autocontida — decisão deliberada em troca desimplicidade (ver [18](18-HARNESS.md#reestruturação-2026-08-25-por-que-branch-primeiro)). Não recrie
esse arquivo.
---
## Harness — Estado, Verificação e Módulos
### Startup Workflow
Antes de escrever código:
1. Confirme o diretório com `pwd`.2. Resolva a branch atual (`git rev-parse --abbrev-ref HEAD`). Se `harness/<branch>/state/` não   existir, crie a partir de `harness/_examples/` (scaffolding — não preencha conteúdo).
3. Leia `harness/user_preferences.md` (se existir).4. Leia `harness/<branch>/state/feature_list.json` e `harness/<branch>/state/progress.md` (se existirem).
5. Revise commits recentes: `git log --oneline -5`.6. Rode `./init.sh` (FAST por padrão; `FULL=1 ./init.sh` para a suíte completa).
Se o baseline já estiver quebrado, corrija antes de implementar feature nova.
### Interação com o usuário
Ao levantar decisões ou pontos em aberto, **faça uma pergunta por vez** e espere a resposta antes da
próxima. Não empacote várias perguntas numa lista só.
```

Convenções de código

Comentários em código seguem estilo ultra-comprimido (telegráfico), não prosa. Comente **por quê**,

não **o quê**. Ver `harness/guides/` e a seção de convenções da documentação do projeto.

Módulos (fonte de verdade)

Módulo	Path	Doc	Verificação
Frontend (React)	`{{PATH}}`	`harness/guides/curio-framework-guide.md`	`npm run lint` (rápido) / `npm run build` (completo)
Configuração	`config/`	`harness/guides/commands.md`	`npm run env:dev` gera `public/config.json`

Antes de editar um módulo, leia o guia correspondente.

Nunca editar à mão: `public/config.json` — é gerado por `setEnvironment.js` a partir de `config/{env}.json`. Toda mudança de configuração vai no arquivo de origem.

Definition of Done

Uma tarefa só está concluída quando:

- o comportamento alvo foi implementado;- `./init.sh` passa (ao menos FAST; FULL antes de considerar pronto para revisão);- a tela foi **aberta no browser** e o fluxo verificado contra o backend real — sem suíte de testes,
  esta é a única evidência funcional que existe;- `harness/<branch>/state/feature_list.json` foi atualizado (`status` + `evidence`);- `harness/<branch>/state/progress.md` foi atualizado ao fim da sessão.

Não marque como `passing` só porque o código foi escrito.

Fim de sessão

1. Atualizar `harness/<branch>/state/feature_list.json` (status + evidência); se algo chegou a `passing`, mover de `features` para `completed` no mesmo arquivo.2. Atualizar `harness/<branch>/state/progress.md`.3. Atualizar `harness/<branch>/state/session-handoff.md` com o resultado da verificação.4. Commit descritivo com o repo em estado seguro — inclui `state/`, `plans/`, `specs/` e `handoffs/`
   da branch, tudo versionado.5. A próxima sessão deve conseguir rodar `./init.sh` imediatamente.

Arquitetura — resumo

```Componente → hook de caso de uso → useCurioMutation → UseCaseManager → @curio/client → Backend
```

- **Não é REST.** Casos de uso por id numérico, requests nomeados (`RM_SALVA_OBJETO`).- **Erro é

```
global.** `queryClient` captura e notifica; páginas não fazem `try/catch` para exibir erro.-
**Config em runtime.** `public/config.json` é lido por `fetch`, não embutido no bundle.-
**Propriedades do backend:** `_Prefixo` para primitivos, sem prefixo para objetos; nomes em
português.
Detalhe completo em `harness/guides/curio-framework-guide.md`.
---
## Aliases
| Alias         | Path              |
| ------------- | ----------------- |
| `@`           | `src/`            |
| `@components` | `src/components/` |
| `@pages`      | `src/pages/`      |
| `@hooks`      | `src/hooks/`      |
| `@utils`      | `src/utils/`      |
| `@types`      | `src/types/`      |
| `@api`        | `src/api/`        |
| `@context`    | `src/context/`    |
| `@theme`      | `src/theme/`      |
Todo alias novo entra em `vite.config.ts` **e** `tsconfig.json`.
```

Se o projeto já tem um `CLAUDE.md`

Insira a seção de harness nele, não sobrescreva. Leia o arquivo inteiro antes, para não duplicar uma seção "Harness" criada numa tentativa anterior.

Revision #2
Created Thu, Aug 20, 2026 5:07 PM by Geraldo Barbosa
Updated Tue, Aug 25, 2026 4:55 PM by Geraldo Barbosa