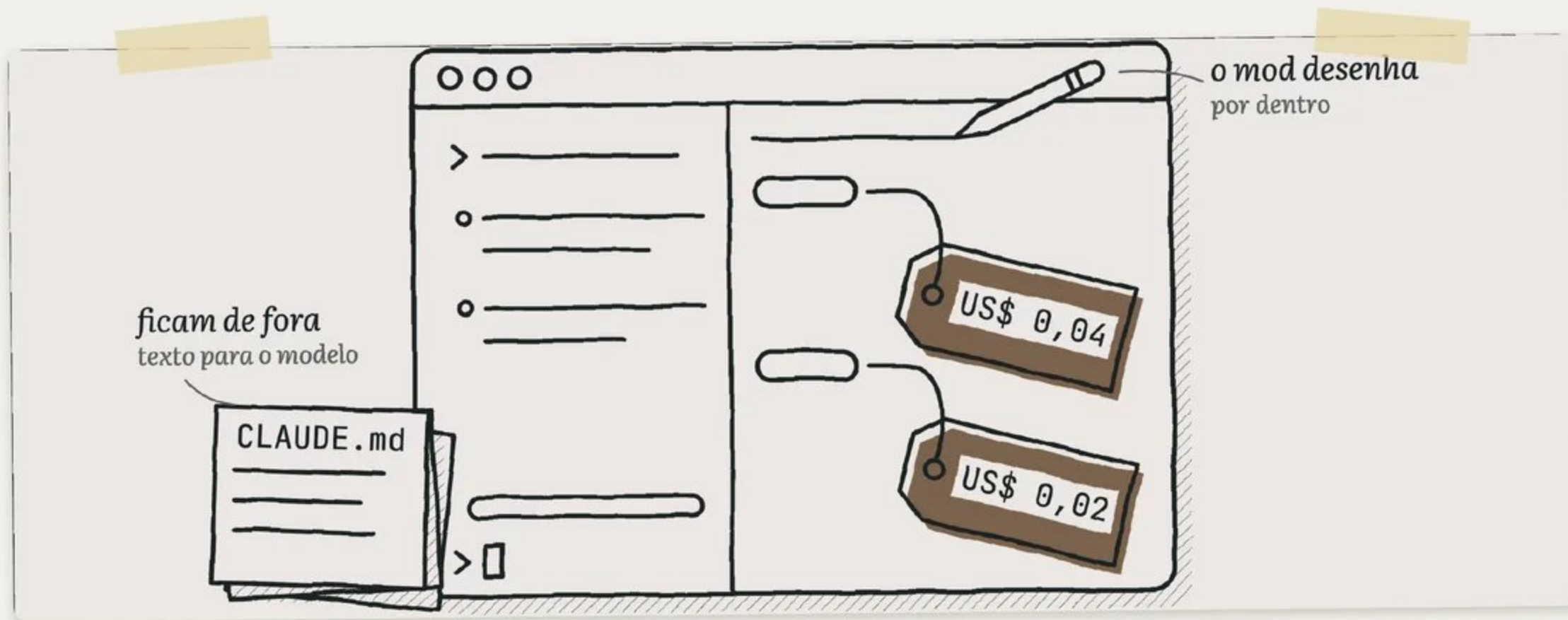


Um mod do Claude Code na prática

Instalação, testes e limites · parte 2 de 2



Um painel ao lado da conversa

Agente

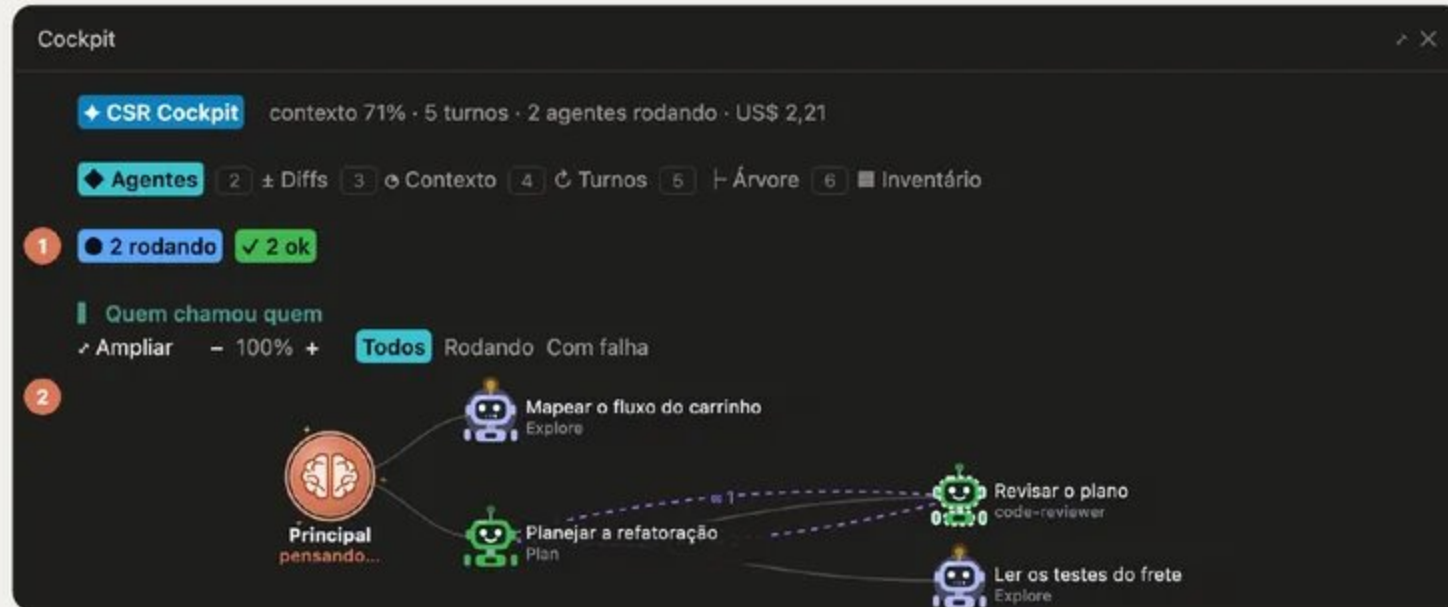
O subagente: o ajudante que o Claude Code chama para cuidar de uma parte da tarefa.

Turno

Cada pedido feito na sessão, com tudo o que ele desencadeia.

Sem um mod, o /usage mostra quanto a sessão inteira custou e, nos planos pagos, que fatia do uso foi para subagentes, em percentual.

Quanto custou cada agente, ou cada turno, ele não diz.



A aba Agentes: (1) as contagens e (2) o grafo de quem chamou quem. A tela sai do próprio mod, numa sessão de exemplo; as bolinhas numeradas apontam o que os próximos slides explicam.

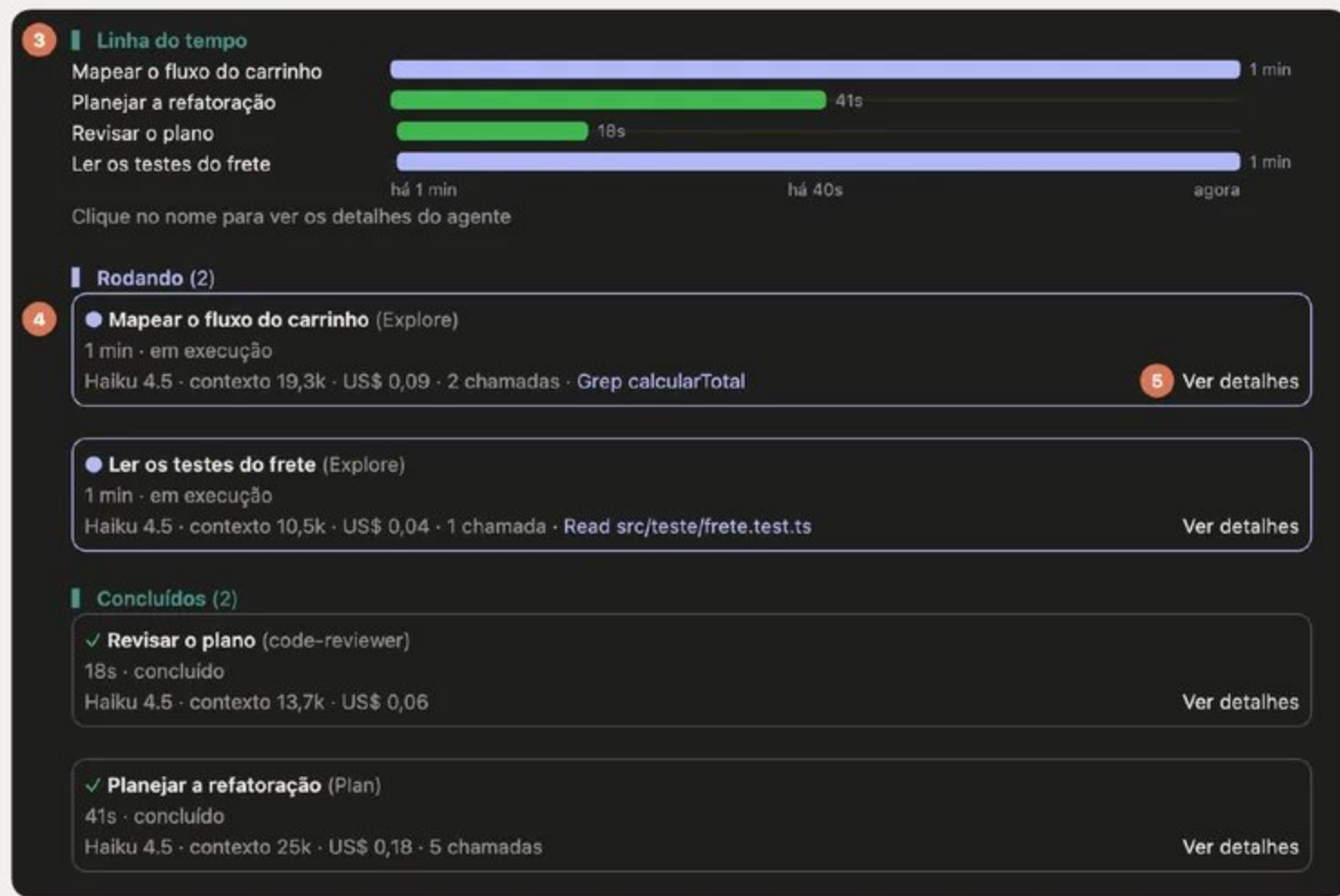
Seis abas, seis perguntas

O painel abre e fecha com `/cockpit`.

Aba	A pergunta	O que mostra
1: Agentes	Quem está trabalhando agora, e quem chamou quem?	os subagentes com o custo atribuído, o grafo de quem criou quem e a linha do tempo
2: Diffs	O que mudou, arquivo por arquivo?	os arquivos alterados e o diff de cada um: da sessão, de um turno ou do git
3: Contexto	Quanto do contexto sobra, e quanto já custou?	o percentual usado, o custo, o limite de uso e o contexto por categoria
4: Turnos	O que aconteceu, na ordem?	um registro por pedido, com a duração, o custo e os comandos Bash
5: Árvore	Onde o Claude está mexendo agora?	os arquivos lidos e editados, acesos enquanto a ferramenta roda
6: Inventário	O que está instalado, e o que está ligado?	plugins, skills, comandos, agentes, hooks e MCP

Clicar no nome de um agente, de um comando Bash ou de um turno abre o detalhe: o pedido, as chamadas e a resposta.

Quem chamou quem, e quem está rodando

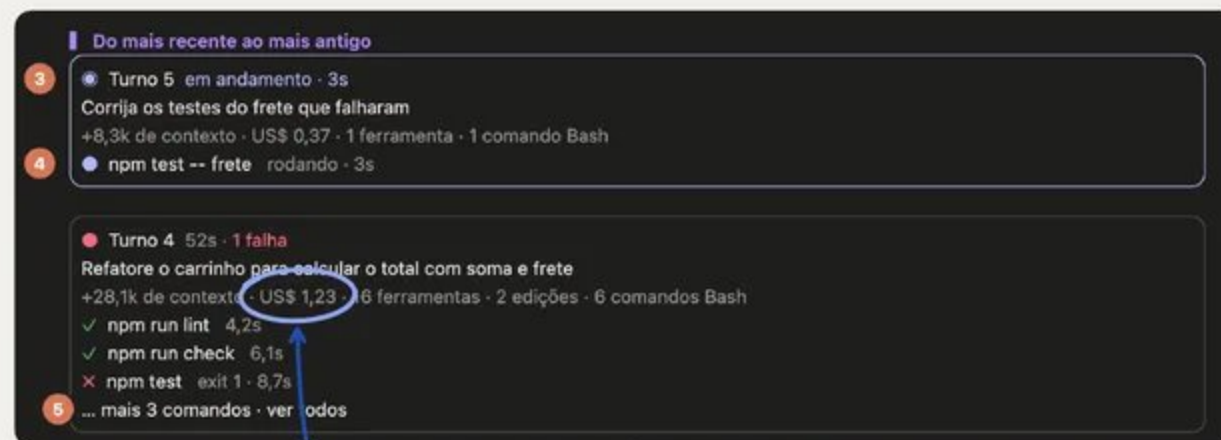
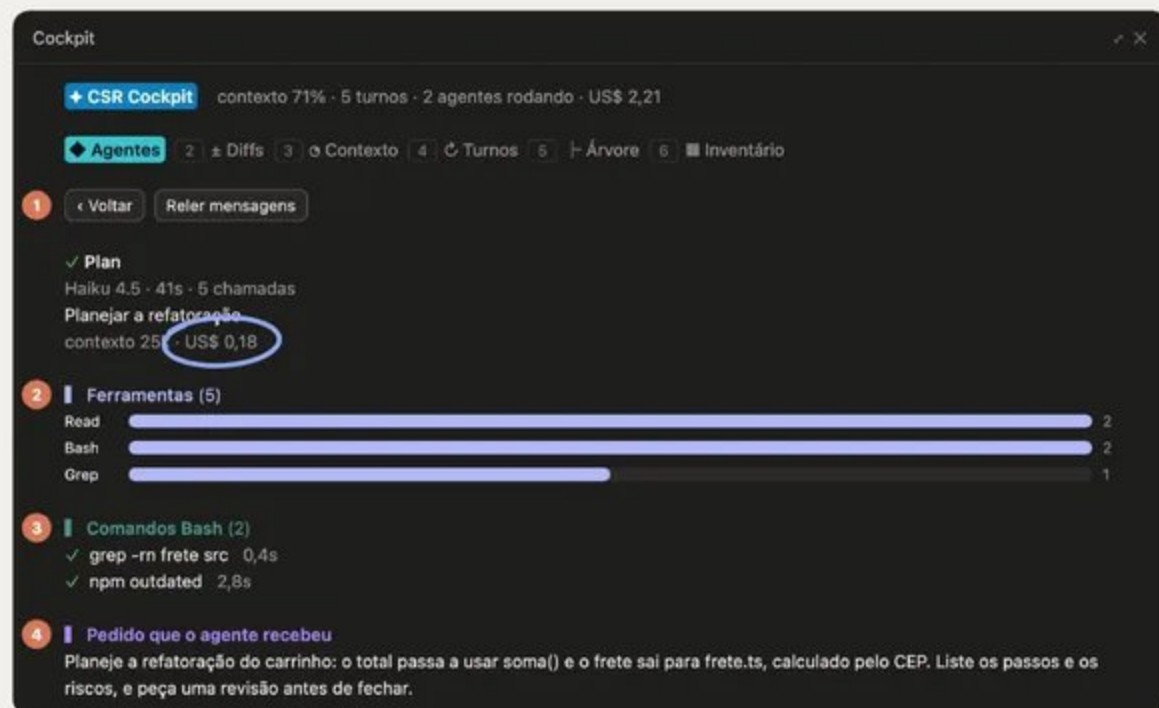


No grafo do slide anterior, a cor é o estado do agente, e a borda diz de onde vem a definição dele: tracejada para os globais, de `~/ .claude/agents`. As mensagens do `SendMessage` aparecem com a contagem.

- 3 A linha do tempo:** uma barra por agente, na mesma régua; mostra quem rodou em paralelo e quem demorou.
- 4 Os cartões:** a tarefa, o modelo, o contexto do agente, o custo atribuído e a ferramenta de agora.
- 5 Ver detalhes** abre o detalhe. Clicar no agente do grafo ou no nome na linha do tempo faz o mesmo.

A linha do tempo e os cartões da aba Agentes (tela do README do `csr-cockpit`).

Quanto custou cada agente, e cada turno



o custo do turno: o fim menos o começo

US\$ 0,18

o Plan: 41 s, 5 chamadas, 25 mil tokens de contexto

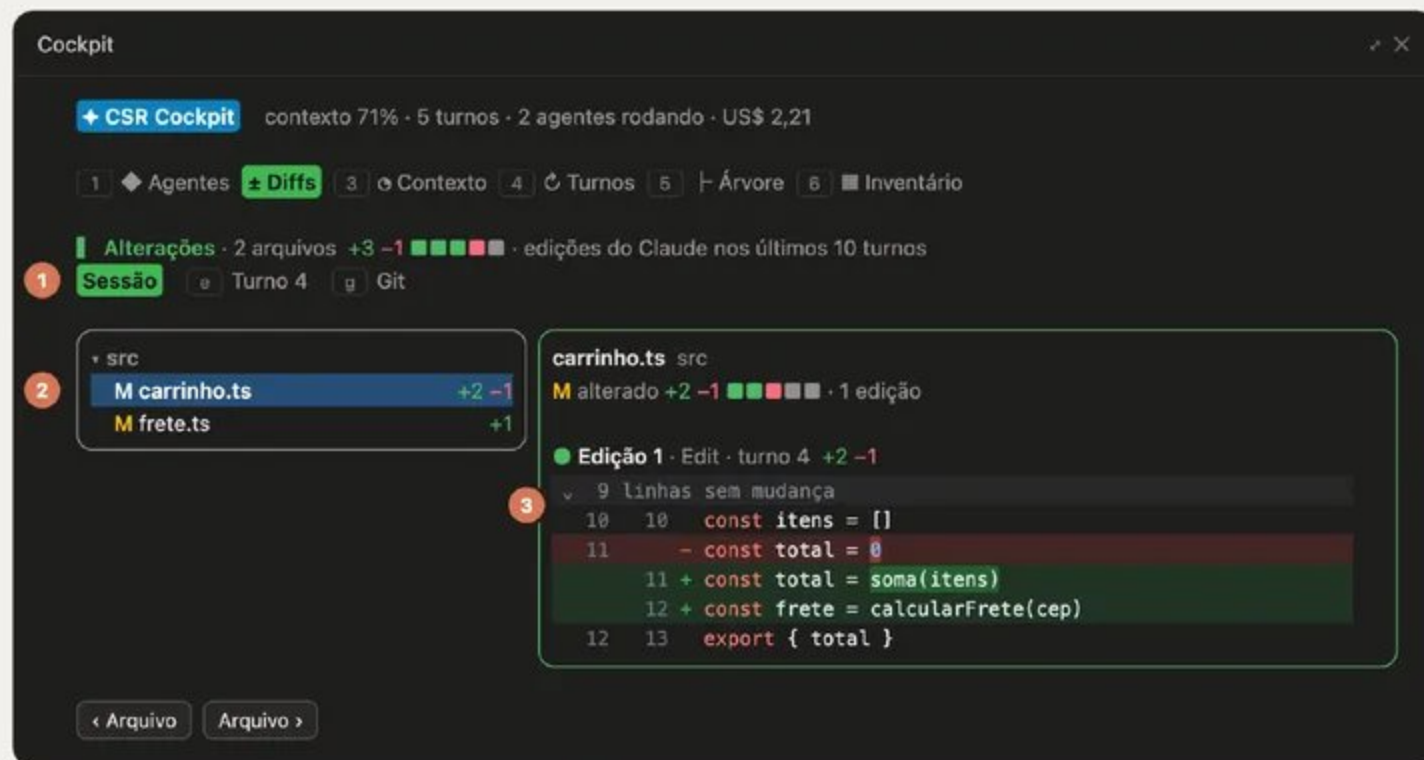
US\$ 0,37

os quatro agentes juntos

US\$ 2,21

a sessão inteira; US\$ 1,84 ficaram com a conversa principal

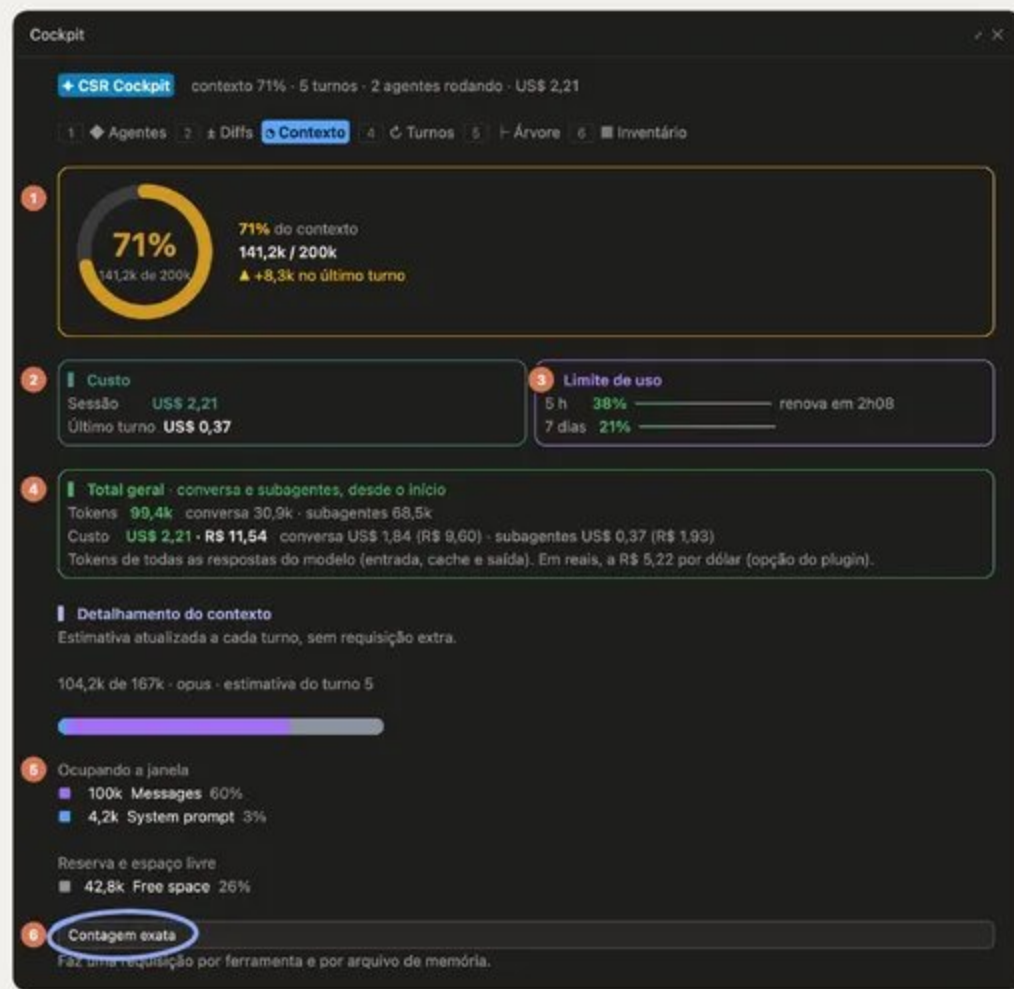
O que mudou, arquivo por arquivo



A aba Diffs, como a visão de alterações do VS Code (tela do README do csr-cockpit).

- 1 **A fonte:** as edições do Claude na sessão, as de um turno ou a árvore de trabalho contra o último commit.
- 2 **Os arquivos** numa árvore, com a letra do status do git e as linhas somadas.
- 3 **O diff**, desenhado como o do próprio Claude Code: números de linha, verde e vermelho e a palavra que mudou em destaque.

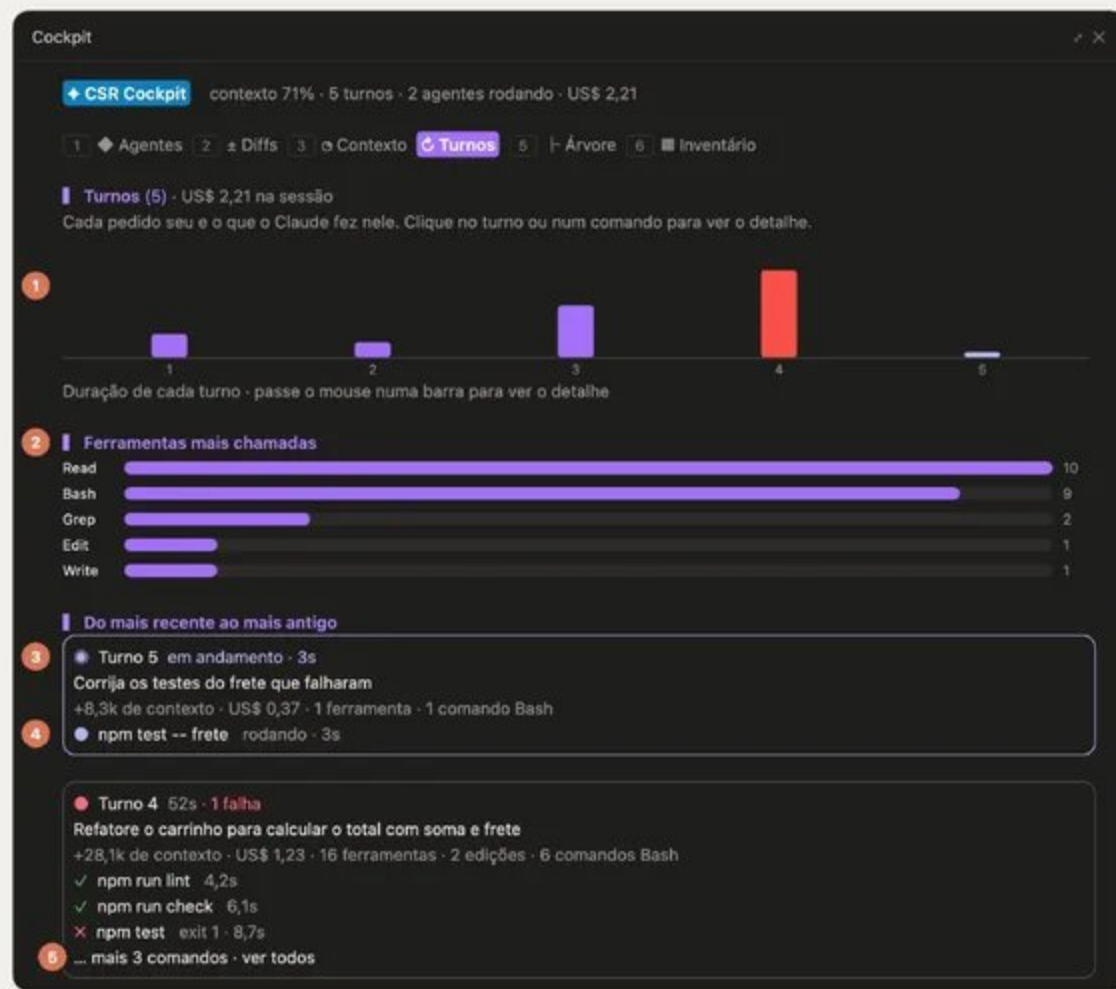
Quanto do contexto sobra, e quanto custou



A aba Contexto (tela do README do csr-cockpit).

- 1 **O uso do contexto:** o percentual, os tokens e quanto o último turno somou.
- 2 **O custo** da sessão e do último turno.
- 3 **O limite de uso** do plano, cada janela com o percentual e quando renova.
- 4 **O total geral:** a conversa e os subagentes, em tokens e em dólar e reais, desde o início.
- 5 **O contexto por categoria:** uma estimativa local, atualizada a cada turno.
- 6 **Contagem exata** troca a estimativa pela contagem de verdade, com requisições a mais.

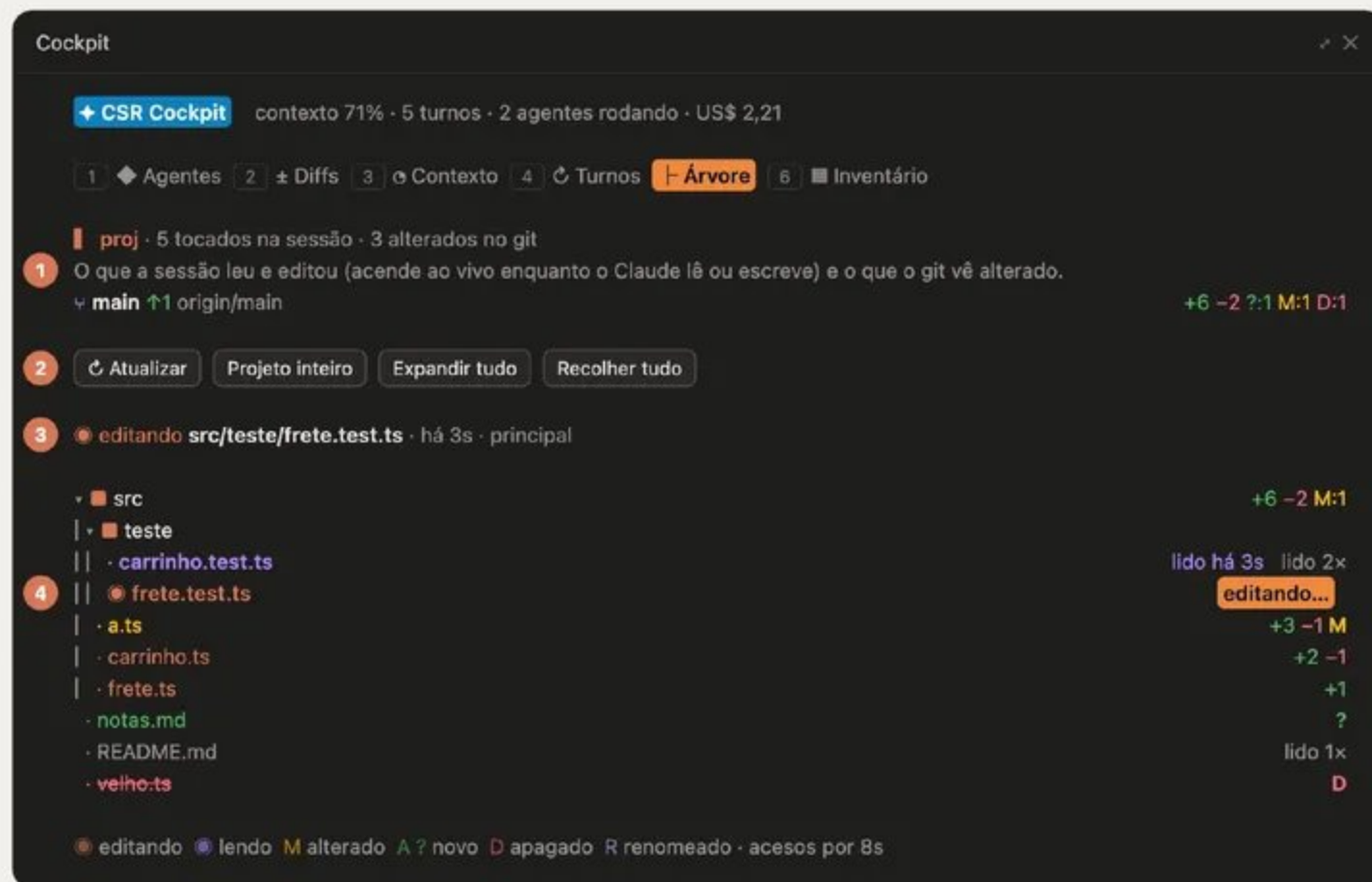
O que aconteceu, na ordem



A aba Turnos (tela do README do csr-cockpit).

- 1 A duração de cada turno:** roxo concluído, vermelho com falha, azul em andamento.
- 2 As ferramentas** mais chamadas na sessão.
- 3 O turno em andamento**, num cartão, com o que ele já usou.
- 4 Os comandos Bash** ficam no turno em que rodaram, com o estado e o exit code.
- 5 Ver todos** abre o detalhe do turno: as ferramentas, os comandos, o pedido e a resposta.

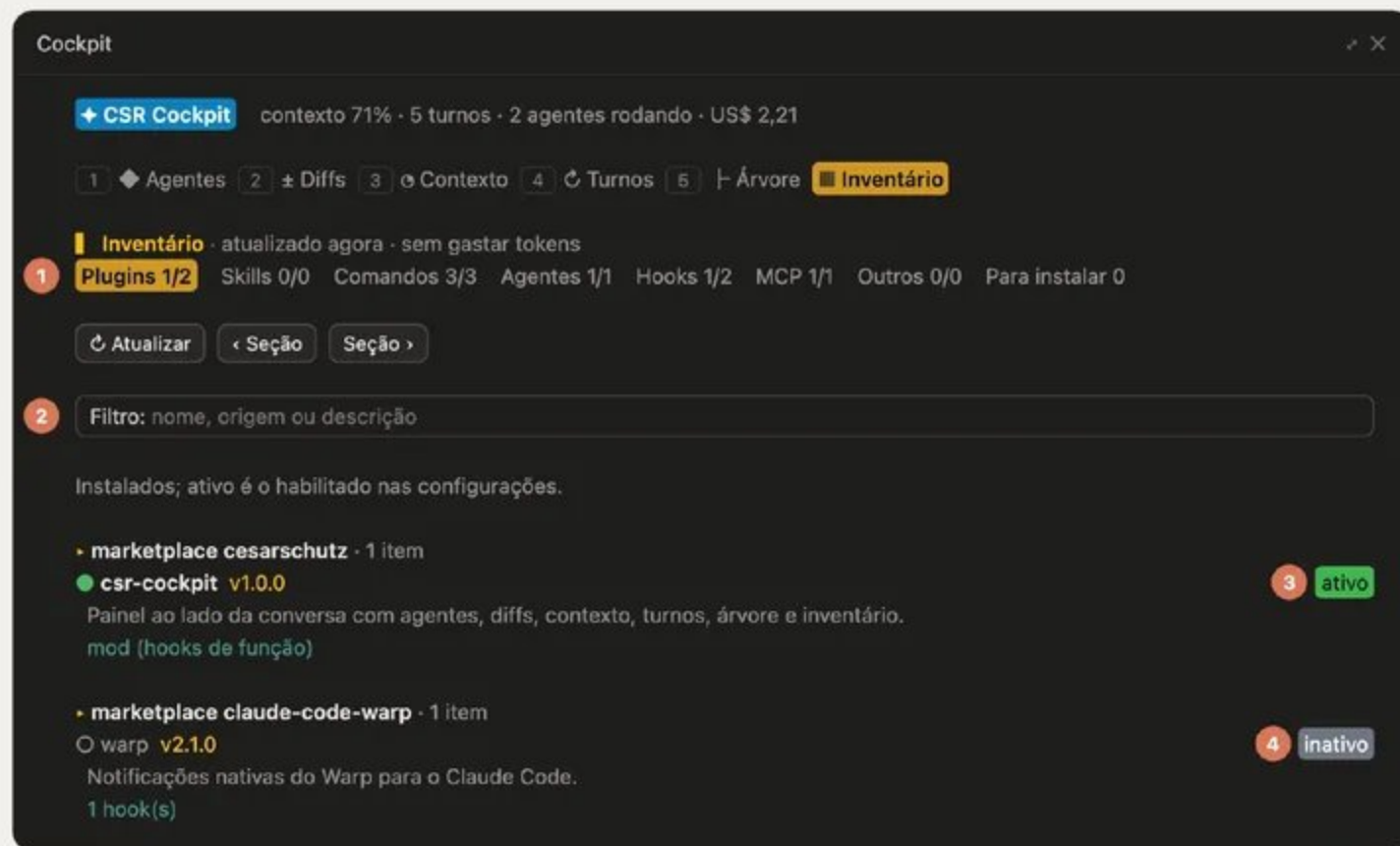
Onde o Claude está mexendo agora



A aba Árvore, com o arquivo que está sendo escrito aceso (tela do README do csr-cockpit).

- 1 **O que a aba mostra:** os arquivos lidos ou editados na sessão e os que o git vê alterados.
- 2 **Os botões:** ler o git de novo, ver o projeto inteiro, expandir e recolher.
- 3 **Agora mesmo:** o arquivo que uma ferramenta está lendo ou escrevendo, e quem.
- 4 **Na árvore,** o mesmo arquivo acende com o selo editando... e fica destacado por 8 segundos.

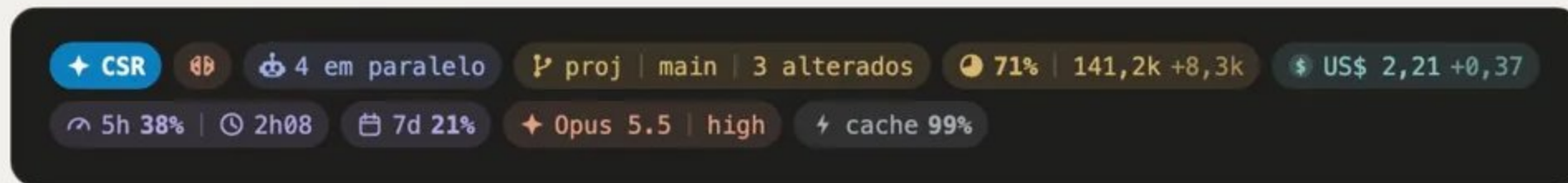
O que está instalado, e o que está ligado



- 1 **As seções:** plugins, skills, comandos, agentes, hooks, MCP, outros e o que há para instalar.
- 2 **O filtro,** por nome, origem ou descrição.
- 3 **Ativo** ou (4) **inativo:** cada peça com a origem, a descrição e o que ela traz.

A aba Inventário (tela do README do csr-cockpit).

Uma faixa acima do prompt



A linha de resumo no aplicativo de desktop (tela do README do csr-cockpit).

Fica ali com o painel aberto ou fechado. Não é a linha de status do rodapé, que é outra peça do Claude Code.

O que está vivo vem primeiro: a conversa principal trabalhando e os agentes em paralelo. Depois, o repositório e as medidas.

No aplicativo de desktop, clicar no repositório ou no contexto abre a aba que fala dele.

A regra do cockpit é só observar.

Nenhum hook do cockpit bloqueia ou altera uma chamada de ferramenta ou uma resposta do modelo: o evento passa adiante como chegou, e o resultado volta como veio.

0 chamadas de rede

Na máquina, só comandos de leitura do git e o arquivo antigo antes de uma escrita.

2 arquivos gravados

Um registro de erros e o diagnóstico de cliques, quando ligado; o resto some com a sessão.

0 tokens gastos

O mod não chama o modelo. A exceção é o botão “Contagem exata”. apertado, o Claude Code faz requisições a mais.

Um repositório com um marketplace.json

O arquivo fica na pasta `.claude-plugin/` e tem três campos obrigatórios: `name`, `owner` e `plugins`.

Cada entrada de plugins é um item instalável, com um `name` e um `source` a pasta dele dentro do repositório.

plugin completo

csr-cockpit

A pasta `plugins/csr-cockpit` tem o seu `.claude-plugin/plugin.json`, o manifesto que descreve o plugin.

peça avulsa

explicar-erro

Não há `plugin.json`: a própria entrada faz o papel de manifesto e diz o que a pasta traz, no campo `skills`.

`.claude-plugin/marketplace.json`

```
{
  "name": "cesarschutz",
  "owner": { "name": "Cesar Schutz" },
  "plugins": [
    {
      "name": "csr-cockpit",
      "source": "./plugins/csr-cockpit",
      "description": "Mod: painel ao lado da conversa..."
    },
    {
      "name": "explicar-erro",
      "source": "./skills/explicar-erro",
      "description": "Exemplo de skill: explica um erro...",
      "skills": ["/"]
    }
  ]
}
```

é este campo que dispensa o plugin.json

Dois nomes, dois comandos

O nome do repositório e o nome do marketplace são coisas diferentes. O repositório se chama `claude-code-kit`, e o marketplace, `cesarschutz`.

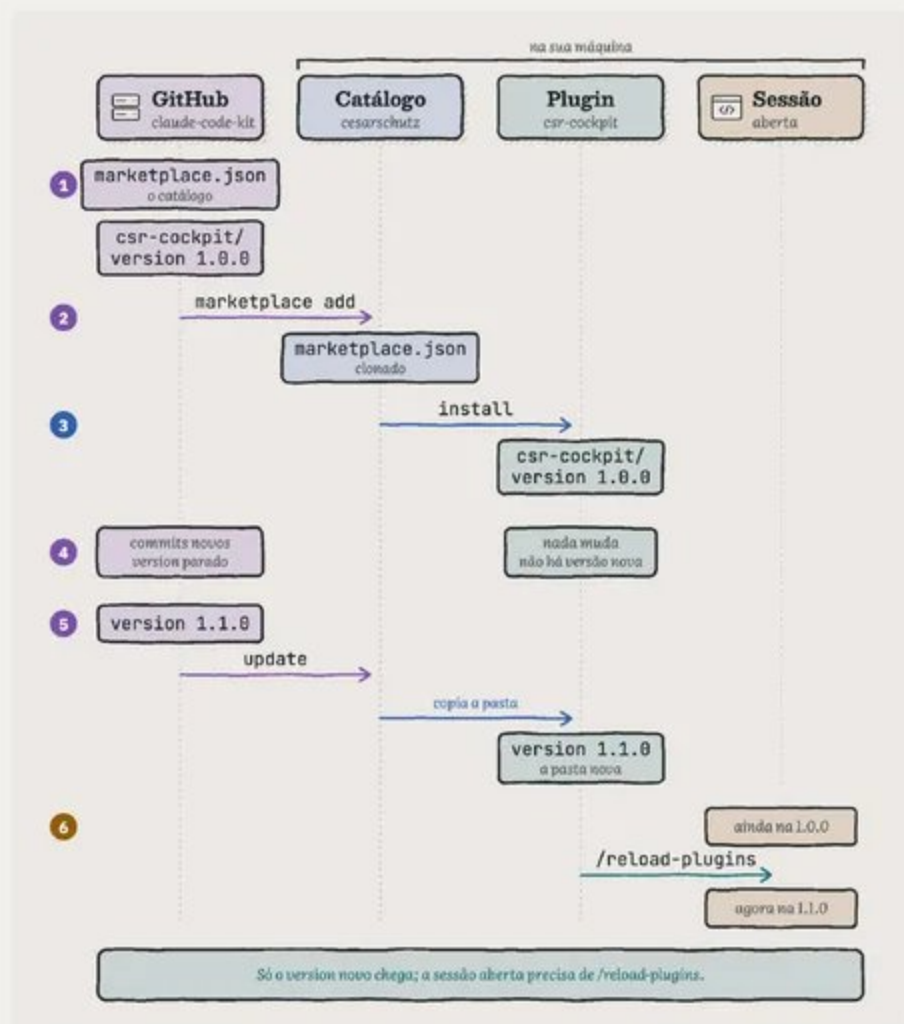
O primeiro aparece na hora de cadastrar; o segundo, na hora de instalar:

Terminal

```
claude plugin marketplace add cesarschutz/claude-code-kit o repositório  
claude plugin install csr-cockpit@cesarschutz o marketplace
```

Se o `/cockpit` não aparecer depois de instalar, confira a versão: o mod pede o Claude Code 2.1.287 ou mais novo.

Instalar e atualizar, em seis passos



- 1 O repositório no GitHub guarda o catálogo e a pasta do plugin, na version 1.0.0.
 - 2 O marketplace add clona o catálogo para a sua máquina, com o nome cesarschutz.
 - 3 O install copia a pasta do plugin na versão do catálogo: 1.0.0.
 - 4 Commits novos chegam ao repositório, mas o version não sobe: na máquina, nada muda.
 - 5 O version sobe para 1.1.0, e o update renova o catálogo e copia a pasta nova.
 - 6 A sessão aberta segue na 1.0.0 até o /reload-plugins carregar a 1.1.0.
- ! O Claude Code só vê atualização quando a versão muda. Com o campo `version` no `plugin.json`, vale esse número; sem ele, que é o caso das peças avulsas, a versão é o commit.

Instalar, conferir, atualizar e testar

No terminal

<code>claude plugin marketplace add <dono>/<repo></code>	cadastra um marketplace de um repositório do GitHub
<code>claude plugin install <item>@<marketplace></code>	instala um item do catálogo
<code>claude plugin list</code>	lista o que está instalado, com a versão e o estado
<code>claude plugin details <item></code>	mostra as peças e o custo delas em tokens
<code>claude plugin marketplace update <marketplace></code>	renova o catálogo, sem mexer no que está instalado
<code>claude plugin update <item>@<marketplace></code>	atualiza para a versão mais nova do catálogo
<code>claude plugin uninstall <item>@<marketplace></code>	remove o plugin
<code>claude plugin validate <pasta></code>	confere o manifesto e as peças; com <code>--strict</code> , aviso vira erro
<code>claude plugin test <pasta></code>	roda os testes de um mod
<code>claude --plugin-dir <pasta></code>	abre uma sessão com o plugin de uma pasta local, sem instalar

Dentro da sessão

<code>/plugin</code>	abre a tela de plugins e marketplaces
<code>/reload-plugins</code>	aplica na sessão aberta o que mudou
<code>/cockpit</code>	abre e fecha o painel do csr-cockpit

*Num marketplace de terceiros, a atualização automática vem desligada: o **update** fica por conta de quem instalou.*

Três camadas, da mais barata à mais cara

camada 1

sem executar nada

`claude plugin validate`

Confere o manifesto e lista os eventos que o mod escuta e as chamadas que faz à API dos mods.

Serve para auditar o mod de outra pessoa antes de instalar.

No cockpit, a lista não tem rede; os processos são só o git, de leitura, e as gravações, só os dois registros.

camada 2

sem login e sem rede

`claude plugin test`

Roda os testes contra um Claude Code de mentira: dispara os eventos, monta o painel, aperta os botões e confere o que foi desenhado.

O cockpit tem 52 testes; um deles confere a regra de só observar.

rodado em 04/10/2026: 52 de 52

camada 3

a mais cara

uma sessão de verdade

O Claude Code num projeto de exemplo, com o mod carregado por `claude --plugin-dir`.

A pasta fica vigiada: o mod recarrega a cada arquivo salvo.

É a única camada que mostra o painel desenhado no terminal.

Os limites da versão 1.0.0

O custo por agente não é um número oficial.

O Claude Code informa o custo da sessão inteira. Ao fim de cada resposta do modelo, o cockpit vê quanto ele subiu e põe a diferença na conta de quem respondeu, um agente ou a conversa principal. Nada é contado duas vezes.

Se duas respostas terminam no mesmo instante, uma fração pode cair no agente vizinho.

O valor em dólar é uma estimativa a preço de tabela. Para quem paga assinatura, ele não é cobrança: serve para comparar um agente com outro.

Os outros limites

Não há custo por comando Bash.

O Bash aparece com o status e a duração.

O raciocínio interno não aparece.

O detalhe mostra só o texto que o agente escreveu.

O código de saída do Bash não é um campo.

Na falha, o cockpit o tira do texto de erro.

A API de mods ainda pode mudar.

De uma versão para outra.

No desktop, depende da versão.

Em 04/10/2026, a versão 2.19675.0 do app já roda o cockpit.

O claude-code-kit: público, com licença MIT

github.com/cesarschutz/claude-code-kit

Em 04/10/2026, ele tem:

- o `csr-cockpit`, na versão 1.0.0;
- o `exemplos`, um plugin com uma skill, um agente, um hook e um estilo de saída;
- sete peças avulsas: duas skills, dois agentes, um estilo de saída, um tema e um hook.

*cada linha instala
sozinha*

O `README.md` explica o catálogo, e o `CONTRIBUTING.md`, como acrescentar um item. O repositório vai continuar recebendo peças.

A parte 1 e as fontes

- ✓ A **parte 1** explica o que é um mod e as peças que vieram antes dele: CLAUDE.md, hooks, skills e plugins.

blog.cesarschutz.com.br/posts/claude-code-do-claude-md-ao-mod

github.com/cesarschutz/claude-code-kit

As fontes do post

Claude: Claude Code mods (o anúncio, 01/10/2026)

Claude Code: Mods overview, Create a mod e Test a mod

Claude Code: Create a marketplace e Marketplace reference

Claude Code: Install and manage plugins, Plugin loading reference e Plugin commands reference

Claude Code: Manage costs effectively

Cesar Schutz: o claude-code-kit e o README do csr-cockpit

Do livro



Volume 07 • 2 artigos

IA

[Ver o livro →](#)

blog.cesarschutz.com.br

cs Cesar Schutz *blog*