

Workflow com Claude Code + Cursor

Fluxos de IA no desenvolvimento em 2026

Alan Rocha · Abril / 2026

Cursor · Claude Code · RTK · Caveman · Settings scopes

Agenda

1. Por que isto importa em 2026
2. Cursor — o que é, como funciona, começando, exemplo
3. Claude Code — o que é, como funciona, começando, exemplo
4. Tabela comparativa Cursor × Claude Code
5. Workflow combinado
6. Economia de contexto: **RTK** e **Caveman**
7. Settings scopes — e o poder do **managed**
8. Anti-padrões, recursos e Q&A

Por que isto importa em 2026

- IA deixou de ser autocomplete: hoje é **par de programação agêntico**.
- Duas superfícies dominam o dia a dia:
 - **Cursor** → editor com IA embutida (edição cirúrgica, feedback visual).
 - **Claude Code** → CLI agêntico (tarefas longas, multi-arquivo, automação).
- Usar **a ferramenta certa para cada momento** (e controlar contexto/custo) é o que separa produtividade real de "brincar com IA".

A questão não é "qual é melhor?", é **quando usar cada uma**.

Cursor — o que é

- **IDE fork do VS Code** com IA nativa, feita pela Anysphere.
- Quatro modos de interação:
 - **Tab** — autocomplete preditivo multi-linha.
 - **Chat (Ask)** — perguntas com contexto do arquivo/projeto.
 - **Composer** — edição multi-arquivo via prompt.
 - **Agent** — execução autônoma com acesso a ferramentas.
- Compatível com extensões do VS Code.

Cursor — como funciona

- **Contexto:** `@file`, `@folder`, `@web`, `@docs`, `@git`, imagens.
- **Rules** (`.cursor/rules/*.mdc`) — instruções persistentes por projeto.
- **Modelos:** Claude, GPT, Gemini, modelos próprios (`cursor-small`).
- **Privacy mode:** código não é usado para treino.
- **Indexação de repositório:** embeddings locais para busca semântica.

Onde fica o privacy mode do cursor:

The image shows the settings interface of the Cursor IDE. On the left is a sidebar with categories like General, Plan & Usage, Agents, Tab, Models, Cloud Agents, Plugins, Rules, Skills, Subagents, Tools & MCPs, Hooks, Indexing & Docs, Network, Beta, Marketplace, and Docs. The main area is titled 'Reset "Don't Ask Again" Dialogs' and contains several sections: 'Layout' with options for Window Layout, Title Bar, Status Bar, Review Control Location, Auto-hide editor when empty, and Open chat as editor tabs; 'Notifications' with options for System Notifications, System Tray Icon, and Completion Sound; and 'Privacy' with the 'Privacy Mode' toggle. Two large white arrows point from the 'Completion Sound' and 'System Tray Icon' settings down to the 'Privacy Mode' setting. The 'Privacy Mode' toggle is currently turned on, and its dropdown menu is open, showing the text 'Privacy Mode'.

alanmrochadeveloper@gm...
Pro Plan

Search settings Ctrl+F

General

Plan & Usage

Agents

Tab

Models

Cloud Agents

Plugins

Rules, Skills, Subagents

Tools & MCPs

Hooks

Indexing & Docs

Network

Beta

Marketplace

Docs

Reset "Don't Ask Again" Dialogs
See warnings and tips that you've hidden

Show

Layout

Window Layout
Switch between Agent and Editor default layouts

Agent Editor

Title Bar
Show title bar in agent layout

Status Bar
Show status bar at the bottom of the window

Review Control Location
Show inline diff review controls in top level breadcrumbs or floating island

Breadcrumb

Auto-hide editor when empty
When all editors are closed, hide the editor area and maximize chat

Open chat as editor tabs NEW
Show chats as editor tabs inside the chat area instead of the legacy stacked view

Notifications

System Notifications
Show system notifications when Agent completes or needs attention

System Tray Icon
Show Cursor in system tray

Completion Sound
Play a sound when agent finishes responding

Privacy

Privacy Mode
Your code data will not be trained on or used to improve the product. Code may be stored to provide features such as Background Agent.

Privacy Mode

```
.cursor/rules/  
10-language.mdc # "responda em pt-BR, código em inglês"  
20-style.mdc # "sem comentários triviais; named exports"  
30-testing.mdc # "cada nova função precisa de teste Vitest"
```

Cursor — começando

1. Baixar em `cursor.com` → instalar → entrar com conta.
2. Abrir um projeto. `Ctrl/Cmd + K` → edit inline. `Ctrl/Cmd + L` → chat.
3. `Ctrl/Cmd + I` → Composer (multi-arquivo).
4. Criar regras do projeto:

```
.cursor/rules/  
10-language.mdc # "responda em pt-BR, código em inglês"  
20-style.mdc # "sem comentários triviais; named exports"  
30-testing.mdc # "cada nova função precisa de teste Vitest"
```

5. Ativar **Privacy mode** em Settings → Beta.

Exemplo da configuração direto no cursor:

The screenshot displays the settings interface for a user named alanmrochadeveloper@gm... on a Pro Plan. The left sidebar contains navigation options: General, Plan & Usage, Agents, Tab, Models, Cloud Agents, Plugins, Rules, Skills, Subagents (selected), Tools & MCPs, Hooks, Indexing & Docs, Network, Beta, Marketplace, and Docs. The main content area is titled 'Rules, Skills, Subagents' and features a search bar (Ctrl+F) and a toggle for 'Include third-party Plugins, Skills, and other configs' which is currently turned on. Below this, there are sections for 'Rules' and 'Skills'. The 'Rules' section includes a '+ New' button and two rule entries: 'You are a frontend specialist in nextjs 16, css and html, FP, OOP, DSA, with the best frontend practices, specially for landing pages highly customizable, with advanced animations using or not ...' and 'you are a frontend specialist with specialization in html, css, javascript, typescript, react 19, ant design 6 or greater, react router v7, data structure and algorithm, fp, oop, with best practices.' The 'Skills' section also has a '+ New' button and lists several skills: 'caveman' (Ultra-compressed communication mode), 'caveman-commit' (Ultra-compressed commit message generator), 'caveman-help' (Quick-reference card), 'caveman-review' (Ultra-compressed code review comments), and 'stripe-best-practices' (Guides Stripe Integration decisions). A 'Show all (46 more)' link is visible at the bottom of the skills list. At the very bottom, the 'Subagents' section is shown with a '+ New' button and a message: 'No Subagents Yet. Create specialized agents to handle focused tasks.' A 'New Subagent' button is located at the bottom center.

Cursor — exemplo prático

Cenário: refatorar `UserCard.tsx` para receber `variant`.

Ctrl/Cmd + I (Composer)

- > @UserCard.tsx @UserList.tsx
- > Adicione a prop `variant: 'compact' | 'detailed'` em UserCard,
- > mantendo 'detailed' como default. Atualize UserList para passar
- > 'compact'. Não altere testes sem necessidade.

Resultado: diff multi-arquivo com apply/revert por hunk.

Cursor é **imbatível** quando você já sabe **qual arquivo** quer tocar.

Um exemplo real:

src > pages > PDV > Caixa > Fechamento > Index.tsx > ...

```
130 const submitCaixaFechamento = (id: string, utilizadorId: string) => async (values: DataTypeSubmi
153   data: payload
154   });
155   } catch (error) {
156     if (error instanceof AxiosError) {
157       if (error.response?.status === 404) {
158         throw new Error(caixaJaFechadoMsg);
159       }
160     }
161     throw error;
162   }
163 };
164
165 const confirmImpressao = () => {
166   return new Promise<boolean>((resolve) => {
167     Modal.confirm({
168       title: "Imprimir Fechamento?",
169       content: "Deseja imprimir o Fechamento do Caixa?",
170       okText: "Sim",
171       cancelText: "Não",
172       icon: (
173         <div style={{ fontSize: 20, width: 20, height: 20, marginRight: 10 }}>
174           <LuPrinter />
175         </div>
176       ),
177       maskClosable: false,
178       cancelButtonProps: { danger: true },
179       onOk: () => {
180         resolve(true);
181       },
182       onCancel: () => {
183         resolve(false);
184       }
185     });
186   });
187 }
```

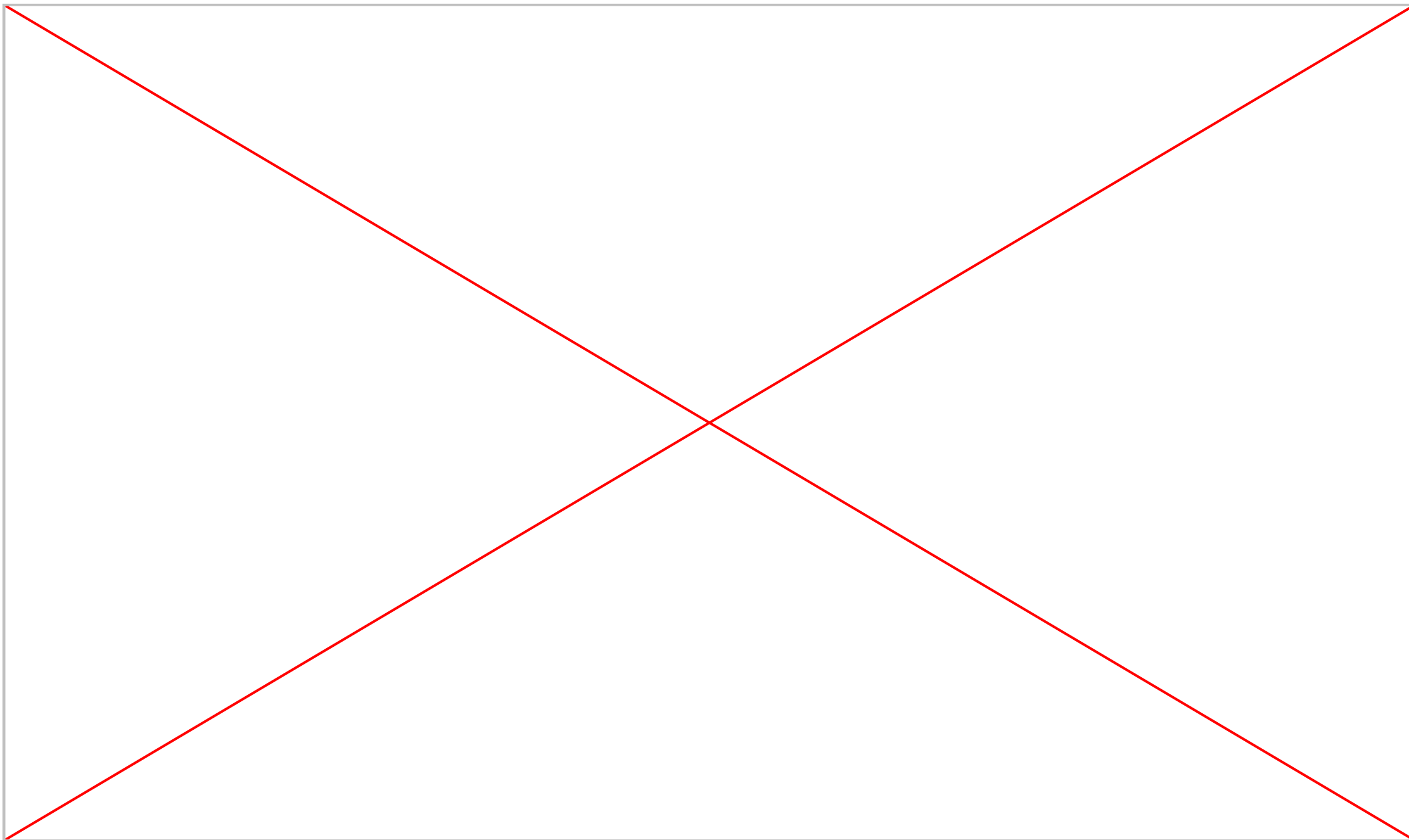
index.tsx (165-187) altere os textos, para uma melhor compreensão do usuário

```
return new Promise<boolean>((resolve) => {
  Modal.confirm({
    title: "Imprimir comprovante de fechamento?",
    content:
      "O caixa foi fechado com sucesso. Deseja imprimir agora o comprovante com o resumo do fechamento?",
    okText: "Sim, imprimir",
    cancelText: "Não imprimir",
    icon: (
      <div style={{ fontSize: 20, width: 20, height: 20, marginRight: 10 }}>
        <LuPrinter />
      </div>
    ),
    maskClosable: false,
    cancelButtonProps: { danger: true },
    onOk: () => {
      resolve(true);
    },
    onCancel: () => {
      resolve(false);
    }
  });
});
```

Mudanças feitas:

- **Título:** "Imprimir Fechamento?" → "Imprimir comprovante de fechamento?" (deixa claro que é um comprovante, não a ação de fechar).
- **Conteúdo:** agora informa que o caixa já foi fechado com sucesso e descreve o que será impresso (resumo do fechamento), evitando ambiguidade sobre o que o "Sim" realmente fará.
- **Botões:** "Sim"/"Não" → "Sim, imprimir"/"Não imprimir" — ações explícitas, seguindo boas práticas de UX (botões autoexplicativos, sem depender apenas do título para entender o efeito).

Um exemplo de edição inline com o cursor:



src > pages > PDV > hooks > usePDVCancelReasonModal > usePDVProductCancelReasonModal.tsx > ...

```
You, 16 hours ago | 1 author (You)
1 import { usePDVModalCancelReason } from "."; FP-4006 adiciona implementação inicial do
2
3 export const usePDVProductCancelReasonModal = () => {
4   return usePDVModalCancelReason({
5     cancelMoment: "NaPassagemDoItem",
6     modalProps: {
7       title: "Deseja remover esse produto",
8       okText: "Remover Produto",
9       cancelText: "Manter Produto"
10    }
11  });
12 };
13
```

src > pages > PDV > FixedFooter > index.tsx > ...

```
You, 18 hours ago | 5 authors (You and others)
1 import { TipoStatusPedidoEnum, TipoTerminalEnum } from "@/backend/api";
2 import { FButton } from "@/components/FButton";
3 import { FModalMethods } from "@/components/FModal";
4 import { FResponseError } from "@/futura-utils";
5 import { FixedFooterModal } from "@/pages/PDV/FixedFooter/Modals";
6 import { FixedFooterButton } from "@/pages/PDV/FixedFooter/mock";
7 import { DiscountPreviewProps, modalBodyContent } from "@/pages/PDV/PedidosHistorico/utills";
8 import { useOrderContext } from "@/pages/PDV/contexts/order-context";
9 import { getShortcutQueryList, useVerticalize } from "@/pages/PDV/hooks";
10 import { DataType as PedidoComprasDataType } from "@/services/pedidoDeCompras";
11 import { newPath, pdvNewPath } from "@/utils/defaults";
12 import { DataTypeDefault } from "@/utils/types/common";
13 import { useFModal } from "@/utils/useFModal";
14 import { useStyle } from "@/utils/useStyle";
15 import { FormInstance, InputProps, message, notification, Space } from "antd";
16 import { Rule } from "antd/es/form";
17 import useFormInstance from "antd/es/form/hooks/useFormInstance";
18 import { BasicProps } from "antd/es/layout/layout";
19 import { Dispatch, ReactNode, SetStateAction, useCallback, useEffect, useRef, useState } from "react";
20 import { AiOutlinePercentage } from "react-icons/ai";
21 import { BiSearch } from "react-icons/bi";
22 import { CiEdit } from "react-icons/ci";
23 import { ImCancelCircle } from "react-icons/im";
24 import { IoCheckmarkOutline } from "react-icons/io5";
25 import { MdOutlineSave } from "react-icons/md";
26 import { SlTrash } from "react-icons/sl";
27 import {
28   NavigateFunction,
29   Params,
30   RelativeRoutingType,
31   useLocation,
32   useNavigate,
33   useNavigation,
34   useParams
35 } from "react-router-dom";
36 import { usePaymentContext } from "../contexts/payment-context";
37 import { useShortcutsContext } from "../contexts/shortcuts-context";
38 import { OrderItemsContextProps, UpdateCurrentItemProps } from "../contexts/types";
39 import { pdvFormFieldNames } from "../formFieldNames";
40 import { ExcludedProperties, PDVStorageTypeDefault } from "../hooks/types";
41 import { usePDVModalCancelReason } from "../hooks/usePDVCancelReasonModal";
```



Claude Code — o que é

- **CLI agêntico oficial da Anthropic**, rodando no terminal ou dentro do IDE.
- Opera por **sessões**: lê arquivos, edita, roda comandos, testa.
- Extensível por:
 - **Hooks** (eventos do ciclo de vida).
 - **MCP** (servidores que expõem ferramentas externas).
 - **Sub-agents** e **skills** (especialistas invocáveis).
 - **Plugins** oficiais e comunitários.

Um exemplo de como é CLI no terminal do Claude Code:

```
Claude Code v2.1.114

Welcome back Alan!



Opus 4.7 · Claude Pro ·
alanmrochadeveloper@gmail.com's Organization
~/Documents/personal/playbooks

Tips for getting started
Run /init to create a CLAUDE.md file w...

Recent activity
5d ago create a subagent or a skill f...
6d ago /context
/resume for more

> /clear
└ (no content)

> █

0 tokens
current: 2.1.114 · latest: 2.1.117
1 MCP server needs auth · /mcp
>> accept edits on (shift+tab to cycle)
```

Claude Code — como funciona

- **CLAUDE.md** — arquivo de memória lido no início da sessão (user/project).
- **Tools nativas:** **Read**, **Edit**, **Write**, **Bash**, **Grep**, **WebFetch**...
- **Plan mode** (**/plan**) — planeja antes de executar; nada é escrito sem aprovação.
- **Sub-agents** — delegam pesquisa/planejamento sem poluir contexto principal.
- **Hooks** em **settings.json** disparam comandos em eventos (**PreToolUse**, **Stop**...).

Claude Code — CLAUDE.md: hierarquia

Lido no início de toda sessão. Mais específico tem precedência.

Escopo	Caminho	Compartilhado?
Managed (org)	<code>/Library/Application Support/ClaudeCode/CLAUDE.md</code> · <code>/etc/claude-code/CLAUDE.md</code> · <code>C:\ProgramData\ClaudeCode\CLAUDE.md</code>	todos na máquina
User global	<code>~/.claude/CLAUDE.md</code>	só você
Project	<code>./CLAUDE.md</code> ou <code>./.claude/CLAUDE.md</code>	sim (git)
Local	<code>./CLAUDE.local.md</code>	não (gitignore)

Imports: dentro de qualquer CLAUDE.md, use `@caminho/arquivo.md` para incluir outro — modulariza instruções sem inflar um só arquivo.

Claude Code — começando

```
curl -fsSL https://claude.ai/install.sh | bash
$claude # login na primeira execução
```

Comandos essenciais:

Comando	Efeito
<code>/init</code>	gera <code>CLAUDE.md</code> inicial do projeto
<code>/plan</code>	entra em plan mode (read-only até aprovar)
<code>/model opus</code>	troca de modelo (opus / sonnet / haiku)
<code>/clear</code>	limpa contexto da sessão
<code>/hooks</code>	gerencia hooks

Claude Code — exemplo prático

Cenário: remover acoplamento de `OrderService` com `EmailClient`.

```
$ claude
> /plan
> Quero extrair o envio de email de OrderService para um
> NotificationPort (hexagonal). Manter testes verdes.
> Liste arquivos afetados antes de editar.
```

Claude Code:

1. Explora o repo (`grep`, `read`).
2. Propõe um plano (arquivos + diffs previstos).
3. Após **ExitPlanMode**, aplica, roda testes e reporta.

Ideal para **refactors amplos, tarefas longas e automação em CI.**

Cursor × Claude Code

Dimensão	Cursor	Claude Code
Tipo	IDE com IA embutida	CLI agêntico (+ extensão IDE)
Superfície	Editor visual, diffs inline	Terminal, sessões longas
Interação	Tab, Chat, Composer, Agent	Prompts + tools + hooks + sub-agents
Modelos	Claude, GPT, Gemini, próprios	Claude-first (Opus / Sonnet / Haiku)
Autonomia	Assistência inline + Agent	Execução agêntica multi-step nativa
Contexto	@file, @docs, Rules, index do repo	CLAUDE.md, MCP, memória, sub-agents
Extensibilidade	Rules, MCP, extensões VS Code	Hooks, skills, plugins, sub-agents, MCP
Billing	Assinatura Cursor	Plano Claude / API Anthropic
Melhor para	Edição cirúrgica, UI, debug visual	Refactors amplos, tarefas longas, CI/headless

Workflow combinado

Use Cursor quando:

- Já sabe o arquivo a alterar.
- Quer feedback visual imediato, diff hunk a hunk.
- Está depurando UI / estilos / pequenas correções.

Use Claude Code quando:

- A tarefa toca muitos arquivos e não sabe todos.
- Precisa de **plano antes de executar** (refactor grande, migração).
- Quer rodar em CI/headless, com hooks e política de permissões.

Combo clássico: Claude Code planeja e aplica mudanças amplas → Cursor faz fine-tuning visual no diff.

Economia de contexto — por que importa

- Janela de contexto é **finita** — mais tokens = mais custo + mais latência.
- Sessões longas **degradam qualidade** (atenção dispersa).
- Output verboso ("Certo! Vou agora...") consome limite sem entregar valor.
- Input inchado (CLAUDE.md gigante, arquivos lidos à toa) é o maior vilão.

Dois aliados no dia a dia: **RTK** (lado CLI) e **Caveman** (lado prompt).

RTK — Rust Token Killer

Proxy CLI que filtra saídas de `git`, `npm`, `ls`, etc. **antes** de chegarem no contexto.

```
rtk --version      # checagem
rtk gain           # analytics de economia (60-90% típico)
rtk gain --history # histórico por comando
rtk discover       # varre histórico e sugere otimizações
rtk proxy          # executa sem filtro (debug)
```

- **Hook transparente** reescreve `git status` → `rtk git status`.
- Zero mudança de hábito: você continua digitando comandos normais.
- ⚠ Colisão de nome: `reachingforthejack/rtk` **não é o mesmo** (Rust Type Kit).

<https://github.com/rtk-ai> <- verdadeiro

Caveman plugin (comunitário)

Skill/plugin de **Julius Brussee** — `github.com/JuliusBrussee/caveman`.

Não é oficial da Anthropic. Reduz verbosidade do modelo.

Modo	Estilo
<code>/caveman lite</code>	remove filler ("Claro!", "Vou agora...")
<code>/caveman full</code>	respostas em fragmentos
<code>/caveman ultra</code>	máxima concisão, estilo telegráfico

Caveman Compress reescreve seu `CLAUDE.md` em estilo comprimido — ataca o lado **input** (maior fatia real de tokens).

Ganho médio em output: ~**65%** em tarefas reais de dev.

Settings scopes no Claude Code

Quatro escopos, precedência `managed > project > local > user`:

Escopo	Caminho	Compartilhado?	Para quê
User	<code>~/.claude/settings.json</code>	não	preferências pessoais (modelo, tema)
Project	<code>.claude/settings.json</code>	sim (git)	hooks, permissões do time, plugins
Local	<code>.claude/settings.local.json</code>	não (gitignore)	overrides por máquina
Managed	diretório do sistema (ver adiante)	imposto por TI	política organizacional, não sobrescrevível

Um `deny` em `managed` torna a tool indisponível **silenciosamente**.

Managed settings — como criar e blindar

Caminhos oficiais (managed scope):

- macOS: `/Library/Application Support/ClaudeCode/managed-settings.json`
- Linux/WSL: `/etc/claude-code/managed-settings.json`
- Windows: `C:\ProgramData\ClaudeCode\managed-settings.json`

Anti-padrões e dicas finais

- ❌ CLAUDE.md com 500 linhas → use **Caveman Compress**.
- ❌ Rodar `git log` sem `--oneline -10` em cada sessão.
- ❌ Aprovar plano sem ler arquivos afetados.
- ✅ `/plan` sempre antes de tarefa multi-arquivo.
- ✅ `rtk gain` semanal para ver o que está economizando.
- ✅ Cursor para hunk-a-hunk, Claude Code para varredura.
- ✅ Hooks para automatizar lint/test após edição.

Recursos + Q&A

Docs & repos

- docs.anthropic.com/claude-code
- cursor.com/docs
- github.com/JuliusBrussee/caveman
- github.com/anthropics/claude-plugins-official

Obrigado! 🧠 Dúvidas?