

Las 7 formas de controlar Claude Code

CLAUDE.md, reglas, skills, subagentes, hooks, estilos de salida y prompt del sistema: qué hace cada uno, cuándo se carga y cuándo usarlo.

Hay siete formas de controlar cómo se comporta Claude Code. Cada una decide cuándo aparece una instrucción y cuánto peso tiene. Cuando sabes cuál usar, dejas de sobrecargar a Claude y empiezas a dirigirlo.

HERRAMIENTA	CUÁNDO SE CARGA	ÚSALA PARA
1. CLAUDE.md	Siempre, al iniciar	El mapa del proyecto: qué es, comandos, convenciones
2. Reglas	Cuando toca ciertos archivos	Qué hacer y qué no en una parte del proyecto
3. Skills	Cuando la activas o cuando aplica	Procesos paso a paso que repites
4. Subagentes	Cuando Claude delega	Tareas secundarias sin ensuciar tu chat
5. Hooks	En eventos, de forma automática	Cosas que deben pasar siempre, sin pedirlos
6. Estilos de salida	Toda la sesión	Cambiar el rol completo de Claude
7. Prompt del sistema	Solo esa sesión	Ajustes rápidos de tono o formato

1. CLAUDE.md

Un archivo en la raíz de tu proyecto que Claude lee siempre. Es la tabla de contenidos: qué es el proyecto, cómo se corre, qué convenciones seguir. Mantenlo corto, porque todo lo que pongas ahí ocupa espacio en cada conversación. Créalo con `/init`.

2. Reglas

Archivos en `.claude/rules/` que solo se cargan cuando Claude trabaja con ciertos archivos. Así no cargas las reglas de la API cuando está tocando el diseño.

`.claude/rules/api.md`

```
---
paths:
  - "src/api/**/*.ts"
---

# Reglas de la API
- Valida siempre la entrada antes de usarla.
- Nunca devuelvas errores con datos internos al cliente.
- Cada endpoint nuevo lleva su prueba.
```

3. Skills

Guías paso a paso en `.claude/skills/nombre/SKILL.md`. Claude solo lee su descripción hasta que la necesita, así que puedes tener muchas sin que pesen. Las activas escribiendo `/nombre`, o Claude las usa sola cuando la tarea coincide con la descripción.

4. Subagentes

Pequeños ayudantes con su propia ventana de contexto. Van, hacen una tarea secundaria (buscar en el código, revisar, investigar) y regresan solo con la respuesta. Tu conversación principal queda limpia. Se definen en `.claude/agents/` o con `/agents`.

5. Hooks

Comandos que se ejecutan solos en momentos específicos: antes o después de usar una herramienta, o cuando Claude termina. A diferencia de una instrucción, un hook siempre pasa. Este te avisa en Slack cuando Claude termina:

`.claude/settings.json`

```
{
  "hooks": {
    "Stop": [
      {
        "hooks": [
          {
            "type": "command",
            "command": "curl -s -X POST -H 'Content-type: application/json' --data '{\"text\": \"Claude terminó la tarea\"}' $SLACK_WEBHOOK_URL"
          }
        ]
      }
    ]
  }
}
```

```
]
}
]
}
}
```

6. Estilos de salida

Cambian el rol completo de Claude, por ejemplo de programador a profesor o a asistente general. Se eligen con `/output-style` y puedes crear los tuyos en `.claude/output-styles/`.

7. Agregar al prompt del sistema

Instrucciones rápidas solo para esa sesión, sin tocar ningún archivo:

Terminal

```
claude --append-system-prompt "Responde siempre en español y explica cada cambio en una línea."
```

Regla práctica: si lo necesita siempre, va en CLAUDE.md. Si solo aplica a unos archivos, es una regla. Si es un proceso, es una skill. Si tiene que pasar sí o sí, es un hook.

¿Quieres ir más lejos?

Ascendent Academy · 20 USD al mes, 7 días gratis: skool.com/ascendent-academy-6509/about

Que lo implementemos por ti · agenda una llamada: calendly.com/vanessa-ascendentllc/asesoria-gratuita

Más recursos gratis: boveda-ascendent.netlify.app