MANUAL PERSONAL · V5 · ESCALONES DEL DOMINIO

De chatbot a sistema autónomo

La evolución no es aprender más prompts: es dejar de usar la IA como asistente de chat y convertirla en un sistema de trabajo. Cuatro escalones — este manual es el roadmap completo, con todo verificado contra la documentación oficial y listo para copiar.

NIVEL 1 · PRINCIPIANTE
Hablar con un chatbot: prompts sueltos, respuestas genéricas. Salida: prompting efectivo.

NIVEL 2 · INTERMEDIO
Contexto real, Projects y fichas de trabajo. Claude ya sabe quién eres.

NIVEL 3 · AVANZADO
Sistemas: Skills, conectores MCP y Artifacts. Los procesos se empaquetan.

NIVEL 4 · EXPERTO
Claude Code, equipos de agentes y rutinas autónomas. Delegas, no pides.

EL MAPA
Todas las piezas del ecosistema, con activación y funcionamiento.

SEMANA 1
Los 4 escalones en 7 días, paso a paso con plantillas.

El ecosistema Claude

Claude no es un chatbot: es un sistema agéntico que razona, lee y edita archivos, ejecuta código, navega la web, controla tu pantalla y se conecta a tus herramientas. Vive en web, móvil, escritorio, terminal, y dentro de Excel, Word, PowerPoint y Chrome. Este capítulo es el mapa: qué pieza existe, para qué sirve y cuándo usarla.

Los modelos — la regla simple

Todos comparten las mismas capacidades base; cambia la profundidad de razonamiento, la velocidad y el costo. La decisión correcta casi siempre es la más barata que resuelva bien tu tarea.

Haiku 4.5

VELOCIDAD

El más rápido y económico.

- Clasificación y etiquetado masivo
- Automatizaciones de alto volumen
- Respuestas simples donde la latencia importa más que la profundidad

Sonnet 5

DÍA A DÍA

El caballo de batalla. Aquí vive la mayoría del trabajo profesional.

- Escritura, investigación y análisis
- Código cotidiano
- Buen equilibrio calidad/velocidad/costo

Opus 4.8

PROBLEMAS DUROS

Razonamiento profundo para lo que no perdona errores.

- Revisión de contratos
- Modelado financiero
- Arquitectura y estrategia

Fable 5

EL TOPE

El modelo más inteligente disponible, de la nueva clase Mythos (por encima de Opus).

- Cuando la primera respuesta debe ser la definitiva
- Análisis multivariable complejos

Regla memorable

Haiku para velocidad · Sonnet para el día a día · Opus/Fable para tus problemas más difíciles. Y si dudas: empieza con Sonnet y sube solo si el resultado se queda corto. *Cómo se cambia:* selector con el nombre del modelo junto al botón de enviar (en Claude Code, `/model`). Los nombres cambian rápido — verifica los vigentes en docs.claude.com antes de citar versiones.

¿Qué superficie uso para cada cosa?

Mismo cerebro, distintas puertas de entrada. Elegir la correcta ahorra la mitad del trabajo.

SI NECESITAS…	USA
Preguntar, redactar, analizar, pensar en voz alta	`Chat` (web, escritorio, móvil)
Trabajar sobre carpetas y archivos reales: PDFs por lote, reportes, organización, flujos completos sin programar	`Cowork` — dale acceso a una carpeta acotada
Escribir, revisar, refactorizar y desplegar código	`Claude Code` — terminal o su versión desktop
Entender un libro de Excel, fórmulas, dependencias, modelar datos	`Claude en Excel`
Que actúe dentro del navegador: leer páginas, llenar formularios web, resumir	`Claude en Chrome`
Operar software de escritorio que no tiene integración: clics, teclado, UI	`Computer Use`
Trabajar manos libres, dictar, iterar hablando	`Voice` (móvil y escritorio)

Las piezas del sistema

Cada tarjeta: qué es, cuándo usarla y un consejo práctico.

Projects

Espacios de trabajo persistentes. Subes documentos de referencia (briefs, manuales, datos) y defines instrucciones personalizadas de rol, tono y formato. Cada conversación dentro del Project las hereda automáticamente: nunca vuelves a explicar tu contexto.

Úsalo cuando: un mismo contexto se repite — un cliente, una campaña, un producto, un flujo. Crea uno por cada uno.

SE ACTIVA: barra lateral de claude.ai o la app → `Projects` → "New project" → escribe las instrucciones y sube archivos. **FUNCIONA:** toda conversación que abras dentro del Project hereda ese contexto automáticamente, sin que lo repitas.

Tip: las instrucciones del Project valen más que cualquier prompt individual: invierte 20 minutos en escribirlas bien y ese esfuerzo se amortiza en cada conversación futura.

Memory

Guarda tus preferencias y estilo de trabajo *entre* conversaciones: cómo te gusta el formato, tu industria, tus clientes recurrentes, tu terminología. Se actualiza sola y puedes ver, editar o borrar cualquier recuerdo desde ajustes.

Úsalo cuando: notes que repites las mismas aclaraciones ("sin viñetas", "en español neutro", "mi jefe se llama…"). Dícelo una vez explícitamente y quedará guardado.

SE ACTIVA: Ajustes → `Capabilities` → interruptor de Memory (con opciones de pausar o reiniciar). Para ver y editar lo guardado: "View and edit memory" en el mismo menú. **FUNCIONA:** Claude extrae preferencias de tus conversaciones y las aplica en las siguientes; también puedes dictarle recuerdos directamente.

Diferencia clave con Projects: Memory es transversal (te sigue a todas partes); un Project es contexto encapsulado de un tema.

Artifacts

La salida interactiva de Claude: en lugar de solo texto en el chat, produce resultados renderizados en un panel aparte — documentos, componentes React, páginas HTML, gráficos SVG, diagramas, archivos de código. Los previsualizas, refinas y exportas sin salir de la conversación.

Úsalo cuando: el resultado sea algo que vas a *usar* fuera del chat: una página, una herramienta, un documento. Este manual es un artifact.

SE ACTIVA: no requiere activación — Claude abre el panel solo cuando el resultado lo amerita; si no lo hace, pídelo: "hazlo como artifact". **FUNCIONA:** panel lateral con vista previa renderizada, versiones conforme iteras, y exportación/descarga directa.

Extended Thinking

El modo de razonamiento profundo: Claude trabaja el problema paso a paso antes de responder, explorando alternativas y verificándose. Tarda más, pero la primera respuesta suele ser la definitiva.

Úsalo cuando: análisis complejos, decisiones de arquitectura, problemas multivariable, o cualquier cosa donde un error cueste caro.

SE ACTIVA: clic en el nombre del modelo junto al botón de enviar → interruptor `Thinking`. En Claude Code, el equivalente es `/effort`. **FUNCIONA:** aparece una sección expandible "Thinking" sobre la respuesta donde puedes leer el razonamiento que siguió.

Tip: no lo dejes activado para todo — para preguntas simples solo añade latencia y costo.

Web Search y Research

Web Search trae información en tiempo real con citas en línea — imprescindible para cualquier dato que cambie con el tiempo. Research es el modo agéntico: combina búsqueda web, tus documentos e integraciones para producir reportes multi-fuente completos.

Úsalo cuando: precios, noticias, versiones, personas en cargos, competencia. Si el dato pudo cambiar desde el entrenamiento del modelo, búscalo.

SE ACTIVA: ícono de controles (deslizadores) en la caja de chat → interruptor `Web search`. Research tiene su propio botón en la misma caja. **FUNCIONA:** Claude decide cuándo buscar, cita fuentes en línea en cada respuesta; Research encadena decenas de búsquedas y entrega un reporte con bibliografía.

Tip probado: para investigaciones grandes, pide primero el *plan* de investigación, corrígelo, y luego suéltalo a investigar.

Skills

Paquetes de flujo de trabajo que le enseñan a Claude tus tareas recurrentes. Una Skill es un archivo `SKILL.md` con instrucciones y plantillas que Claude lee y aplica automáticamente cuando es relevante — o que invocas directo con `/nombre`.

Úsalo cuando: te descubras pegando las mismas instrucciones por tercera vez. Esa es la señal exacta.

SE ACTIVA: en la app: Ajustes → `Capabilities`, donde ves, instalas y creas Skills. En Claude Code viven en `.claude/skills/` y se invocan con `/nombre`. **FUNCIONA:** Claude carga la Skill solo cuando es relevante para la tarea — el material largo no gasta contexto hasta que se usa.

Hay Skills pre-hechas (Excel, Word, PowerPoint, PDF) y puedes crear las tuyas, componibles entre sí. Detalle completo en el capítulo de automatización.

Conectores MCP

MCP (Model Context Protocol) conecta a Claude con tus herramientas y datos externos: Gmail, Calendar, Drive, Slack, GitHub, Notion, Asana, bases de datos… Claude lee, actúa y responde directamente desde ellas en vez de que hagas copy-paste de ida y vuelta.

Úsalo cuando: la pregunta sea sobre *tus* datos ("¿cuántos registros tuvimos ayer?", "¿qué hay en mi agenda?"). Sin conector, Claude no puede saberlo.

Conectores especializados que valen oro: Fireflies (transcripciones de reuniones → decisiones, responsables y riesgos en dashboards), Metricool (métricas de redes, detectar qué contenido funcionó y programar publicaciones), Zoom, y decenas más en el directorio.

SE ACTIVA: Ajustes → `Connectors` → elige del directorio (Gmail, Drive, Slack, GitHub…) → autoriza con tu cuenta (OAuth). En Claude Code: `/mcp`. **FUNCIONA:** cada conector expone acciones concretas (leer, buscar, crear) que Claude llama cuando la tarea lo pide; tú controlas los permisos por conector.

Cowork

La aplicación de escritorio para trabajo no-dev. Le das acceso a una carpeta específica y Claude lee, escribe, edita y ejecuta flujos de trabajo completos de forma autónoma: procesar PDFs por lote, análisis de correos, tareas automatizadas, reportes recurrentes, modelado financiero.

Úsalo cuando: el trabajo involucre archivos reales de tu computadora y varios pasos seguidos.

SE ACTIVA: app de escritorio de Claude (Mac/Windows) → modo `Cowork` → selecciona la carpeta a la que le das acceso. **FUNCIONA:** Claude arma una lista de tareas, trabaja los archivos de esa carpeta de forma autónoma (leer, escribir, ejecutar) y te va mostrando avances; tú apruebas lo delicado.

Regla de seguridad: carpeta acotada, tarea bien definida, y respaldo antes de empezar.

Claude Code

La herramienta de línea de comandos para desarrolladores: delega tareas de código completas desde tu terminal. Control remoto, tareas programadas, integración MCP (leer Jira, implementar, revisar Sentry, redactar el update de Slack). Claude Code Desktop añade interfaz gráfica.

Úsalo cuando: cualquier cosa que viva en un repositorio. Tiene dos capítulos propios abajo: prácticas y comandos.

SE ACTIVA: descarga desde claude.ai/code o instala con `npm install -g @anthropic-ai/claude-code`; luego escribe `claude` en la terminal dentro de tu repo e inicia sesión.
FUNCIONA: conversas en la terminal y Claude usa herramientas de archivos, bash y git; qué aprueba y qué corre solo lo defines con `/permissions`.

Computer Use

Claude controla tu pantalla: lee el contenido, simula clics y teclado, y opera software directamente — sin necesidad de integración o API. Si un humano puede hacerlo con mouse y teclado, Claude puede intentarlo.

Úsalo cuando: llenar formularios, testing de UI, navegación de apps de escritorio, captura de datos de sistemas viejos sin API.

SE ACTIVA: desde la app de escritorio (Cowork): cuando la tarea lo requiere, Claude solicita acceso a aplicaciones específicas y tú apruebas cada una en un diálogo. **FUNCIONA:** ciclo de captura de pantalla → decidir siguiente clic o tecleo → ejecutar → volver a mirar. Por eso es preciso pero lento.

Tip: es la opción más lenta de todas — úsala solo cuando no exista conector ni versión web de la herramienta.

Claude en Excel / Word / PowerPoint / Chrome

Dentro de *Excel*: entiende libros completos, fórmulas y dependencias, y actualiza cálculos preservando estructura. En *Word*: borradores, ediciones, reestructura documentos. En *PowerPoint*: crea y reorganiza decks. En *Chrome*: agente dentro del navegador que asiste directamente en cualquier página.

Úsalo cuando: el trabajo ya vive en Office o el navegador — evitas exportar/importar.

SE ACTIVA: Excel/Word/PowerPoint: instala el complemento de Claude desde la tienda de complementos de Microsoft (AppSource) e inicia sesión. Chrome: extensión `Claude in Chrome` desde la Chrome Web Store. **FUNCIONA:** panel lateral dentro de la app con acceso al documento o página activa — lee y edita en el lugar.

Voice

Modo de voz en iOS, Android, web y escritorio. Contexto completo mantenido al alternar entre voz y texto; transcripciones guardadas automáticamente. Integración con Gmail, Calendar y Drive por voz.

Úsalo cuando: camines, manejes (con cuidado) o pienses mejor hablando que tecleando. Dictar un brief desordenado y pedir que lo estructure es un flujo sorprendentemente bueno.

SE ACTIVA: ícono de voz en la caja de chat de la app (móvil y escritorio); en Claude Code, `/voice`. **FUNCIONA:** conversación hablada con transcripción automática guardada; alternas a texto en cualquier momento sin perder el contexto.

¿Y qué es exactamente MCP? La analogía del menú

Una API es como el menú de un restaurante: te dice qué puedes pedir, qué no está disponible, y cómo pedirlo — tú ordenas y alguien te lo trae, sin que entres a la cocina. MCP es el protocolo que le entrega a Claude ese menú para cada una de tus herramientas: qué puede consultar, qué acciones puede ejecutar y con qué formato. Por eso un conector bien configurado se siente como un empleado con acceso a los sistemas, y la falta de conector como un empleado al que no le han dado usuario.

Los 4 pilares: glosario rápido

Toda la escalera de automatización en una tabla — vuelve a ella cuando dudes qué pieza usar.

ELEMENTO	QUÉ ES	CUÁNDO USARLO	AUTOMATIZACIÓN
Prompt	Instrucción puntual (ficha de trabajo)	Tareas rápidas y únicas	Baja (manual)
Skill	Proceso y memoria empaquetados	Tareas repetitivas con estándar	Media (asistida)
Artifact	Interfaz visual, editable y exportable	Prototipos, webs, dashboards, docs	Capa visual
Routine	Ejecución autónoma en la nube vía MCP	Flujos que corren solos	Alta (autónoma)

Configúralo como a un empleado nuevo

El error más común es tratar a Claude como buscador: pregunta suelta, respuesta suelta, nada persiste. La inversión es el setup; el retorno son horas cada semana. La secuencia, en orden:

1 **Elige el plan correcto**
Pro para individuos; Team/Enterprise para grupos. Define desde el inicio quién más lo va a usar, porque Projects y Skills se comparten distinto en cada plan.

2 **Crea tus Projects**
Uno por cliente, función o flujo. Sube los documentos de contexto (briefs, manuales de marca, datos históricos) y escribe instrucciones personalizadas: rol, tono, formato de salida, qué evitar. Es la parte que casi todos se saltan y la que más rinde.

3 **Alimenta Memory manualmente**
En las primeras sesiones, dile explícitamente tus preferencias, terminología y clientes: "recuerda que mis reportes van sin viñetas y con resumen ejecutivo al inicio". Verifica en ajustes qué guardó.

4 **Conecta tus herramientas por capas**
Primero el ecosistema principal (Google Workspace o Microsoft 365), verifica que funciona, y luego las del día a día: Slack, gestor de proyectos, GitHub. No conectes todo el primer día — no sabrás qué falló.

5 **Construye tu primera Skill**
Identifica la tarea repetitiva más frecuente de tu semana (un tipo de correo, un formato de reporte, un análisis). Escribe las instrucciones, prueba con casos reales, refina. Una Skill buena vale más que diez mediocres.

6 **Estrena Cowork con una tarea acotada**
Carpeta específica, objetivo claro, respaldo previo. Buenos primeros encargos: "organiza estos 200 PDFs por tema" o "extrae los datos de estas facturas a un Excel".

7 **Añade Claude Code si programas**
Conecta los servidores MCP del proyecto, corre /init para generar el CLAUDE.md, y dale una tarea real pequeña para verificar la calidad de salida antes de confiarle cosas grandes.

Manos a la obra: monta cada pieza en minutos

El capítulo anterior explica qué es cada cosa; este te lleva de la mano para dejarla funcionando. Cada guía tiene los pasos exactos y una plantilla lista para pegar. Hazlas en orden la primera semana: una por día basta.

Día 1 · Un Project que trabaje por ti

Pasos: 1) Barra lateral → Projects → "New project" y nómbralo por tema ("Cliente ACME", "Finanzas personales"). 2) Sube 3–5 documentos de referencia — los que siempre acabas explicando. 3) Pega esta plantilla en "Instructions" y rellénala. 4) Abre una conversación dentro y pruébalo con una tarea real.

PLANTILLA · INSTRUCCIONES DEL PROJECT

```
ROL: Eres mi asistente de [área: p. ej. operaciones / el cliente X].
CONTEXTO: Soy Mario, me dedico a [qué haces]. En este Project
trabajamos [tipo de tareas]. Los documentos adjuntos son [qué son
y para qué sirven].
FORMATO: Responde en español. Reportes de máximo 1 página, sin
viñetas anidadas. Los correos, de menos de 120 palabras.
REGLAS: Verifica cifras antes de afirmarlas. Si un dato no está en
los documentos, dilo — no lo inventes. Ante ambigüedad, pregunta
antes de asumir.
```

Sabes que quedó bien cuando: abres una conversación nueva y Claude ya sabe quién eres, qué tono usar y dónde buscar — sin que escribas nada de contexto.

Día 2 · Entrena tu Memory en 5 minutos

Pasos: 1) Ajustes → Capabilities → activa Memory. 2) Abre cualquier conversación y pega el mensaje de abajo, rellenado. 3) Verifica en "View and edit memory" qué guardó y corrige lo que haga falta. 4) Repite cuando notes que vuelves a aclarar lo mismo.

MENSAJE · DICTAR RECUERDOS

```
Guarda estas preferencias mías de forma permanente:
1. Respóndeme siempre en español.
2. Mis reportes: máximo 1 página, resumen ejecutivo al inicio.
3. Trabajo en [industria] y mis herramientas principales son [X, Y].
4. Mis proyectos/clientes recurrentes: [nombres y una línea de cada uno].
5. Cuando te pida un correo, fírmalo "Saludos, Mario".
6. Prefiero que me digas los riesgos sin suavizarlos.
Confírmame la lista exacta de lo que guardaste.
```

Diferencia con el Project: esto te sigue a TODAS las conversaciones; el Project solo aplica dentro de su carpeta.

Día 3 · Conecta tu primer MCP

Pasos: 1) Ajustes → Connectors → busca tu ecosistema principal (Google Workspace o Microsoft 365). 2) "Connect" → autoriza con tu cuenta (OAuth) → revisa qué permisos le das. 3) Pruébalo con las preguntas de abajo. 4) Solo cuando funcione, añade el siguiente conector (Slack, GitHub, Notion…). Uno por vez.

PROMPTS DE PRUEBA

```
¿Qué tengo en mi agenda mañana?
Resume mis correos sin leer de hoy y dime cuáles necesitan respuesta.
Busca en mi Drive el documento más reciente sobre [tema] y resúmelo.
```

Regla de oro: si Claude responde "no tengo acceso a…", el conector falta o no autorizaste ese permiso — no insistas con el prompt, arregla la conexión.

Día 4 · Tu primera sesión de Cowork

Pasos: 1) Abre la app de escritorio → modo Cowork. 2) Crea una carpeta de prueba con 10–20 archivos reales (copias, no originales). 3) Selecciónala como carpeta de trabajo. 4) Pega el encargo de abajo. 5) Revisa el plan que te proponga ANTES de aprobar.

PLANTILLA · ENCARGO BIEN ACOTADO

```
En esta carpeta hay [qué contiene]. Quiero que [objetivo concreto,
p. ej. "clasifiques los PDFs por tema en subcarpetas y me hagas un
índice en Excel"].
Reglas:
- No borres ni sobrescribas nada; los resultados van en una
  subcarpeta nueva llamada "salida".
- Muéstrame tu plan antes de ejecutar y espera mi OK.
- Al terminar, dame un resumen de qué hiciste y qué no pudiste hacer.
```

El patrón del encargo perfecto: qué hay + qué quieres + reglas de seguridad + pedir plan primero. Sirve para cualquier tarea de Cowork.

Día 5 · Tu primera Skill (sin escribirla tú)

Pasos: 1) Encuentra en tu historial un prompt que te funcionó muy bien dos o más veces. 2) Pega el mensaje de abajo con ese prompt dentro. 3) Revisa el SKILL.md que te proponga. 4) Instálalo: Ajustes → Capabilities (app) o `.claude/skills/nombre/SKILL.md` (Claude Code). 5) Pruébalo pidiendo la tarea con otras palabras — debe activarse solo.

MENSAJE · DESTILAR UNA SKILL

```
Estas instrucciones ya me funcionaron bien varias veces:
---
[pega aquí tu prompt ganador]
---
Conviértelas en un SKILL.md: nombre corto en kebab-case, descripción
con los disparadores claros (cuándo debe activarse), y pasos
numerados sin ambigüedad. Muéstramelo para revisarlo antes de
darlo por bueno, y dime exactamente dónde guardarlo.
```

Ya hay 6 listas: en la sección Biblioteca de este manual tienes correo, minuta, extracción, brief, revisión y deploy para copiar directo.

Día 6 · Primera sesión de Claude Code

Pasos: 1) Instala y entra a un repo pequeño (no tu monorepo gigante). 2) Corre la secuencia de abajo. 3) Dale una primera tarea real pero acotada. 4) Revisa el diff con `/diff` antes de aceptar nada.

TERMINAL · SECUENCIA COMPLETA

```
npm install -g @anthropic-ai/claude-code
cd mi-proyecto
claude          # inicia sesión la primera vez
/init           # genera el CLAUDE.md del repo
/permissions    # revisa qué puede tocar sin preguntar
/plan corrige el bug de [describe uno pequeño y real]
```

Sabes que quedó bien cuando: el plan que propone menciona los archivos correctos de tu repo. Si menciona archivos que no existen, tu CLAUDE.md necesita más contexto.

Día 7 · Cierra el circuito

Con todo montado, crea tu primera automatización: pídele a Claude el briefing diario del recetario (sección Recetario) como tarea programada. A partir de ahí, cada mañana empiezas con ventaja.

Nivel 1 · Principiante — hablar con un chatbot

Aquí empieza todo el mundo: entras, escribes una pregunta suelta, recibes la respuesta y cierras la pestaña. No está mal — pero es usar el 5% de la herramienta.

Cómo se ve este nivel

Prompts improvisados desde cero cada vez. Peticiones vagas, sin contexto ni condiciones ("hazme un reporte de ventas"). Cada conversación empieza de cero y nada se reutiliza.

El problema real

Las respuestas salen genéricas, amplias o poco útiles — y la reacción natural es culpar a la IA. Pero Claude no puede leer tu mente: una instrucción sin información produce una respuesta sin dirección.

La mentalidad equivocada

"Necesito una IA más lista." No: necesitas una instrucción más completa. La calidad de la salida es función directa de la calidad de la entrada — esa es la lección que te gradúa de este nivel.

Cómo salir

El capítulo siguiente — **Prompting efectivo** — es el currículum completo de este nivel: seis técnicas que convierten peticiones vagas en instrucciones que funcionan. Con eso, y con adjuntar tus archivos reales en vez de describirlos, ya estás en el Nivel 2.

Subiste al Nivel 2 cuando… (márcalo — se guarda solo)

- ☐ Dejaste de escribir prompts de una línea
- ☐ Toda petición lleva instrucción + contexto + condiciones
- ☐ Adjuntas el archivo real en lugar de describirlo
- ☐ Guardaste tu primer prompt para reutilizarlo

El currículum del ascenso: prompting efectivo

Las infografías dicen qué botón apretar; casi ninguna dice cómo pedir las cosas. Estas seis técnicas son la base — funcionan en chat, en Cowork y en Claude Code por igual.

1 · Sé claro y detallado

La causa #1 de malos resultados es el pedido vago. Incluye: el objetivo, el contexto (para qué es, para quién), las restricciones y el formato de salida esperado.

Vago: "hazme un reporte de ventas" → **Claro:** "reporte de 1 página para dirección: ventas de junio vs mayo, 3 hallazgos clave, tono ejecutivo, sin viñetas".

2 · Da ejemplos (positivos y negativos)

Un ejemplo de lo que quieres vale más que tres párrafos describiéndolo. Y un ejemplo de lo que *no* quieres cierra la otra mitad del espacio de error.

Patrón: "Así me gusta: [ejemplo bueno]. Así NO: [ejemplo malo]. Nota que el bueno hace X y evita Y."

3 · Pide razonamiento paso a paso

Para problemas con lógica encadenada, pide explícitamente que razone antes de responder: "piensa paso a paso antes de dar tu respuesta final". En la app, activa Extended Thinking para el mismo efecto con más profundidad.

4 · Estructura con etiquetas

Cuando el prompt mezcla instrucciones, datos y ejemplos, sepáralos con etiquetas tipo XML. Claude las respeta y deja de confundir qué parte es qué.

PATRÓN

```
<instrucciones>Resume en 5 líneas, tono neutro.</instrucciones>
<documento>[texto a resumir]</documento>
<formato>Párrafo único, sin viñetas.</formato>
```

5 · Especifica longitud y formato

Claude no puede adivinar si esperas 3 líneas o 3 páginas. Dilo siempre: "máximo 200 palabras", "tabla de 2 columnas", "solo el código, sin explicación". Es la instrucción más barata con el mayor impacto perceptible.

6 · Itera en vez de reintentar

Si el resultado no te gustó, no borres y empieces de cero: di exactamente *qué* cambiar ("el punto 2 está bien, el tono del 3 es muy formal, corta el cierre"). El feedback específico convierte un resultado del 70% en uno del 95% en una vuelta.

Y si algo funcionó muy bien: guárdalo. Ese prompt es candidato a Skill.

El truco de la ambigüedad

Si no estás seguro de que Claude tenga todo lo necesario, cierra tu prompt con esta coletilla — lo obliga a pensar antes de ejecutar.

COLETILLA UNIVERSAL

```
Antes de responder, revisa si falta contexto o si hay algo ambiguo. Si es así, hazme preguntas breves para afinar mejor la tarea.
```

Referencia completa

La guía oficial de prompt engineering está en docs.claude.com — vale la pena leerla una vez completa: *anthropic prompt engineering overview*.

Nivel 2 · Intermedio — contexto, Projects y fichas de trabajo

El salto clave: dejas de trabajar en abstracto. Claude analiza tus datos reales, vive dentro de espacios de trabajo con contexto fijo, y tus prompts se vuelven fichas estructuradas que produces en segundos.

La ficha de trabajo

Toda petición seria tiene tres partes: **instrucción** clara (qué hacer), **contexto** detallado (para qué, para quién, con qué datos) y **condiciones** (formato, longitud, restricciones). Memoriza la plantilla:

PLANTILLA · FICHA DE TRABAJO

```
INSTRUCCIÓN: [qué quieres, en un verbo concreto]
CONTEXTO: [situación, audiencia, datos adjuntos y qué son]
CONDICIONES: [formato de salida, longitud, tono, qué evitar,
y si quieres revisión antes de la versión final]
```

Vago: "Dame ideas para un post de LinkedIn sobre IA" → **Profesional:** "Actúa como estratega de contenido senior. Revisa el documento adjunto sobre mi herramienta y genera 3 ideas de posts que ataquen los miedos del sector creativo. Tono provocador pero educativo. Evita 'revolucionario' y 'disrupción'."

Contexto real, no descrito

Usa el botón **+** para subir PDFs, hojas de cálculo, imágenes y capturas. "Analiza este CSV" con el archivo adjunto vale diez veces más que describir sus columnas de memoria. Y los **Projects** eliminan la re-explicación: cada frente de trabajo (un cliente, un canal, un área) con sus archivos y sus instrucciones fijas — montado en la guía del Día 1.

El optimizador de prompts

Tu instrucción puede nacer caótica (dictada por voz, escrita a las prisas) — el truco es no ejecutarla así. Pide primero que la profesionalice:

PROMPT · OPTIMIZADOR

```
Antes de ejecutar nada: toma esta petición desordenada y
reescríbela como ficha de trabajo profesional (instrucción,
contexto, condiciones). Señala qué información me falta darte.
Muéstrame la versión mejorada y espera mi OK.
Petición: [pégala tal cual salió]
```

Tu arsenal de este nivel

El **Recetario** que sigue son 12 fichas de trabajo ya redactadas para los pedidos más comunes. Y las guías del **Día 1 y Día 2** (Projects y Memory) son las dos configuraciones que hacen permanente este nivel.

Subiste al Nivel 3 cuando…

- ☐ Tienes 2+ Projects con instrucciones propias
- ☐ Memory entrenada con tus preferencias
- ☐ Nunca explicas el mismo contexto dos veces
- ☐ Notaste que repites las mismas fichas… y ya te molesta (buena señal)

Recetario: 12 fichas de trabajo para el día a día

Los pedidos que más se repiten en la vida real, ya redactados con las técnicas del capítulo anterior. Copia, rellena los corchetes y envía. Los dos últimos son para programar y olvidarte.

1 · Domar un PDF largo

Cuándo: te llega un documento de 40 páginas y necesitas decidir algo hoy.

PROMPT

```
Lee este documento completo. Entrégame:
1. Resumen de 10 líneas.
2. Las 5 decisiones o datos que me afectan directamente.
3. Lo que tú harías en mi lugar, y por qué.
4. Las preguntas que debería hacer antes de decidir o firmar.
Contexto: soy [tu situación frente al documento].
```

2 · Radiografiar un Excel

Cuándo: te comparten datos y no sabes ni qué hay dentro.

PROMPT

```
Analiza este archivo. Primero dime qué contiene: hojas, columnas,
rango de fechas, huecos. Después: 1) tendencias principales,
2) valores atípicos que merecen explicación, 3) las 3 gráficas que
más valdría la pena hacer, y hazlas. No modifiques el original —
todo lo nuevo en un archivo aparte.
```

3 · De notas sueltas a presentación

Cuándo: tienes ideas desordenadas y una fecha de entrega.

PROMPT

```
Convierte estas notas en una presentación de [N] diapositivas para
[audiencia: dirección / cliente / equipo]. Primero propónme solo el
esqueleto (título y mensaje de cada lámina) y espera mi OK. Luego
genera el .pptx: poco texto por lámina, estilo [sobrio/visual].
Notas: [pégalas]
```

4 · Responder el correo difícil

Cuándo: llevas 20 minutos mirando el cursor sin saber cómo decir que no.

PROMPT

```
Ayúdame a responder este correo: [pégalo].
Contexto: [qué pasó antes]. Quiero [decir que no / negociar plazo /
poner un límite] sin dañar la relación.
Dame 2 versiones de menos de 120 palabras: una diplomática y una
más directa. Señala en cada una la frase que hace el trabajo duro.
```

5 · Decidir una compra o proveedor

Cuándo: tres opciones, precios que cambian, y opiniones contradictorias.

PROMPT

```
Compara estas opciones para [uso]: [A, B, C]. Busca información
actual en la web (precios, reseñas, problemas conocidos).
Entrégame: 1) tabla comparativa con los criterios que me importan
—[costo, durabilidad, soporte…]— ponderados, 2) tu recomendación,
3) el riesgo principal de esa recomendación.
```

6 · Llegar preparado a una reunión

Cuándo: reunión importante mañana y cero tiempo de prepararla.

PROMPT

```
Mañana tengo reunión con [quién] sobre [tema]. Con base en este
contexto: [pega correos/notas, o dile que revise tu correo si tienes
el conector], prepárame: 1) el objetivo realista de la reunión,
2) 5 puntos a tratar en orden estratégico, 3) las preguntas
incómodas que me pueden hacer, con respuestas sugeridas, 4) qué NO
me conviene tocar.
```

7 · Aprender un tema en 7 días

Cuándo: necesitas dominar algo nuevo sin perderte en tutoriales.

PROMPT

```
Quiero dominar [tema] a nivel [conversación de trabajo / uso
práctico]. Diseña un plan de 7 días, 30 minutos diarios: cada día
un concepto, por qué importa, un ejercicio aplicado a MI contexto
([cuál es]) y una pregunta de autoevaluación. Empieza ahora con el
día 1 completo.
```

8 · Revisar un contrato o documento

Cuándo: antes de firmar cualquier cosa.

PROMPT

```
Revisa este documento como asesor cauto que está de MI lado.
Lista: 1) cláusulas o frases que me exponen, 2) ambigüedades que
podrían interpretarse en mi contra, 3) qué falta y debería estar,
4) qué renegociaría y con qué redacción alternativa.
Al final dime si esto amerita pasar por un abogado de verdad.
```

9 · Pulir un texto sin perder tu voz

Cuándo: el texto ya existe, solo necesita quedar impecable.

PROMPT

```
Pule este texto: corrige gramática, claridad y ritmo, pero NO
cambies mi estilo ni añadas florituras. Mantén mis palabras donde
funcionen. Entrégame el texto pulido y, aparte, la lista de qué
cambiaste y por qué.
Texto: [pégalo]
```

10 · Poner orden en una carpeta

COWORK

Cuándo: esa carpeta que da miedo abrir.

PROMPT

```
Organiza esta carpeta: 1) clasifica los archivos en subcarpetas por
tipo y tema, 2) renombra los de nombre críptico con el patrón
AAAA-MM-DD_descripcion-corta, 3) identifica duplicados pero NO los
borres — lístalos en un reporte, 4) déjame un resumen de qué
moviste y a dónde. Muéstrame el plan antes de ejecutar.
```

11 · Briefing diario

PROGRAMADO

Cuándo: una vez. Después corre solo cada mañana.

PROMPT

```
Crea una tarea programada: cada día laboral a las 7:00 am, revisa
mi correo y mi calendario y entrégame: 1) lo nuevo que importa y
qué necesita respuesta mía, 2) mis 3 prioridades del día según mis
compromisos, 3) conflictos de agenda o plazos que se acercan.
Máximo 15 líneas, sin viñetas anidadas.
```

12 · Cierre semanal

PROGRAMADO

Cuándo: una vez. Cada viernes te espera el resumen.

PROMPT

```
Crea una tarea programada: cada viernes a las 5:00 pm, con base en
mi correo y calendario de la semana, resume: 1) qué se avanzó y qué
se cerró, 2) qué quedó pendiente y de quién depende, 3) las 3
prioridades sugeridas para el lunes. Formato: media página, tono
directo.
```

El multiplicador

Cada receta que uses más de dos veces, conviértela en Skill (guía del Día 5) y desaparece el copy-paste. Ese es el camino: prompt → receta → skill → automatización.

Nivel 3 · Avanzado — sistemas: Skills, Conectores y Artifacts

El punto de inflexión: dejas de pensar en prompts y empiezas a pensar en **sistemas**. Los procesos se empaquetan, las herramientas se conectan, y la salida deja de ser texto plano.

De prompts a Skills

Una Skill empaqueta un proceso completo — descripción, instrucciones, reglas y plantillas — para tareas repetitivas: humanizar textos, armar presentaciones, revisar errores. Se activa sola cuando toca. El capítulo **Anatomía de una Skill** que sigue es la ingeniería completa; la **Biblioteca** trae 6 listas para instalar.

Skills que aprenden

La diferencia entre nivel 3 y nivel 2 no es crear la skill: es **iterarla**. Cada vez que Claude cometa un error usando una skill, añade la regla que lo previene:

PROMPT · ITERAR UNA SKILL

```
En la última ejecución de la skill [nombre] pasó esto que no
quiero: [describe el error]. Propón la regla exacta que lo
previene, añádela al SKILL.md, y muéstrame el diff del cambio.
```

Conectores: Claude sale del chat

Con MCPs conectados (Drive, Notion, GitHub, correo, calendario…), Claude lee y actúa sobre tus sistemas reales en vez de opinar en el vacío. Montaje en la guía del Día 3; el detalle de cada conector, en el Mapa. Regla: un conector nuevo por vez, y pruébalo antes de añadir el siguiente.

Artifacts: pide interfaces, no párrafos

La salida deja de ser texto: paneles interactivos, diagramas, calculadoras, webs funcionales que editas visualmente en el chat. El hábito de este nivel: ante cualquier entregable, pregúntate "¿esto debería ser un artifact?" — si alguien lo va a *usar*, la respuesta es sí. Este manual es la prueba.

Y con **Claude Design**: editas señalando el elemento (un botón, un logo) sin reescribir prompts, subes tu identidad de marca (logos, colores, fuentes) para que diseñe con TU estilo y no algo genérico, y exportas directo a Canva, PowerPoint o Vercel.

Subiste al Nivel 4 cuando…

☐ Tu primera skill propia funciona y ya la iteraste tras un error

☐ Tienes 2+ conectores en uso diario

☐ Pides artifacts por defecto para entregables

☐ Empiezas a querer que las cosas corran *sin ti*

Anatomía de una Skill: de la tarea repetitiva al sistema inteligente

La productividad real no nace de redactar mejores prompts aislados, sino de pasar de la improvisación a la sistematización: convertir a Claude en un repositorio de tu conocimiento procedimental. Este capítulo es la ingeniería detrás del Nivel 3.

La filosofía, según los ingenieros de Anthropic

"No se trata de construir agentes cada vez más complejos, sino de desarrollar habilidades concretas (skills) que le den a la IA experiencia específica para hacer mejor ciertas tareas."

Chat vs. Skills: el cambio de paradigma

Un chat es un interlocutor efímero; una Skill es un activo digital que hereda tu metodología.

ASPECTO	ENFOQUE DE CHAT	ENFOQUE BASADO EN SKILLS
Naturaleza	Improvisación lineal y aislada	Sistematización de procesos
Persistencia	*Context decay:* las reglas se diluyen al cerrar la sesión	*Knowledge persistence:* el método queda blindado como activo permanente
Preparación	Explicar el contexto a mano en cada interacción	Cero: el conocimiento ya reside en el sistema
Evolución	El aprendizaje muere tras la respuesta	Incremental: cada ajuste mejora el sistema para siempre

La triple capa

Una Skill no es un párrafo: es un sistema modular de tres capas.

Capa 1 · Descripción (el disparador)

La identidad de la habilidad. Su función vital: que Claude reconozca *cuándo* activarse solo. Una descripción precisa es un sensor — con disparadores vagos, la skill nunca se autoinvoca; con disparadores claros, aparece justo cuando la necesitas.

Capa 2 · Instrucciones (el proceso)

El manual de operaciones: no solo "qué hacer" sino los pasos lógicos, los criterios de calidad, el tono y — lo que separa un juguete de un sistema profesional — la gestión de **casos límite**: qué hacer cuando los datos faltan, se contradicen o son ilegibles.

Capa 3 · Recursos y herramientas (el soporte)

El arsenal de ejecución: documentos de referencia, ejemplos de éxito (few-shot) y **scripts ejecutables**. El hallazgo de los ingenieros de Anthropic: en vez de dejar que la IA "reinvente la lógica" escribiendo el mismo código cada vez, guarda el script dentro de la Skill y que lo ejecute como herramienta de precisión.

EJEMPLO TÉCNICO · SKILL DE ANÁLISIS FINANCIERO

```
SKILL: FIN_ANALYZER_PRO

CAPA 1 (DESCRIPCIÓN):
Se activa ante la carga de balances (CSV/XLSX) o consultas de
rentabilidad trimestral. Objetivo: auditar desviaciones
presupuestarias y proyectar flujo de caja.

CAPA 2 (INSTRUCCIONES):
1. Ejecutar limpieza de datos con el script adjunto.
2. Comparar contra el benchmark histórico (Ref: Q4_Master.pdf).
3. Identificar 3 anomalías y proponer medidas de mitigación.
Criterio: aplicar las normativas del manual adjunto.
Caso límite: si falta un mes de datos, NO extrapolar — reportarlo.

CAPA 3 (RECURSOS):
- Documento: Standard_Operating_Procedures_V2.pdf
- Herramienta: Calculadora_EBITDA_Audit.py
- Caso de éxito: Ejemplo_Reporte_Aprobado_Board.md
```

Metodología de construcción: de la idea al proceso guardado

No adivines el proceso: extráelo de tu cabeza de forma estructurada.

1 **Identifica con la ecuación Frecuencia × Fricción**
Lo que haces a diario Y te genera resistencia es el candidato ideal. *Tip de experto:* evita las mega-skills — si una tarea tiene más de 7 pasos críticos, divídela; la especificidad es la madre de la precisión.

2 **Entrevista y planifica**
Usa el "entrevistador de procesos" (prompt abajo) para que Claude te interrogue sobre objetivos, inputs y criterios de éxito — captura el contexto que sueles omitir. *Tip de experto:* define siempre los casos límite: qué hacer cuando los datos faltan. Si el proceso es grande, pide que la entrevista termine en un documento de requisitos (PRD) antes de construir nada.

3 **Construye con skill-creator**
Empaqueta la lógica extraída con la skill oficial `skill-creator` — es "guardar trabajo para tu futuro yo". *Tip de experto:* si notas que Claude repite un cálculo o proceso lógico, oblígalo a guardarlo como script en la Capa 3.

4 **Depura iterativamente**
Ejecución de prueba, analiza los puntos de falla, ajusta. No busques perfección inicial: busca estabilidad del flujo. *Tip de experto:* usa el modo Plan para que explique su lógica antes de ejecutar — es más barato corregir un plan que un archivo generado.

PROMPT · ENTREVISTADOR DE PROCESOS

```
Vas a ayudarme a convertir una tarea mía en una Skill. Actúa como
entrevistador de procesos: hazme preguntas, UNA a la vez, hasta
tener claro: 1) el objetivo y cómo se ve un resultado excelente,
2) los inputs exactos que recibo y en qué formato, 3) los pasos que
sigo (incluye los que hago "sin pensar"), 4) los criterios de
calidad y el tono, 5) los casos límite: qué hago cuando falta algo
o algo se contradice. Cuando tengas todo, propón el SKILL.md con
sus tres capas y espera mi revisión.
La tarea es: [descríbela en una línea]
```

El ciclo de evolución: nunca corrijas el output, corrige el sistema

El error más grave: corregir el resultado dentro del chat y dejar que ese aprendizaje muera al cerrar la pestaña.

¿Faltó conocimiento?

Añade un manual, FAQ o documento de referencia a la **Capa 3**. La IA no puede aplicar un criterio que nunca le diste por escrito.

¿Faltó criterio?

Añade una restricción o instrucción negativa a la **Capa 2**: "nunca hagas X", "si pasa Y, pregunta antes". Las reglas negativas son tan valiosas como las positivas.

¿Faltó un patrón?

Incluye el error y su versión corregida como ejemplo de "qué no hacer" en la **Capa 3**. Un contraejemplo real enseña más que tres párrafos de teoría.

PROMPT · AUTODIAGNÓSTICO TRAS UN FALLO

```
Revisa nuestra interacción y dime qué regla o recurso debemos
inyectar en la Skill [nombre] para que este fallo no vuelva a
ocurrir jamás. Clasifícalo: ¿faltó conocimiento (Capa 3), criterio
(Capa 2) o un patrón de ejemplo? Propón el cambio exacto al
SKILL.md y muéstrame el diff.
```

Modularidad y orquestación: el ecosistema de Skills

La automatización avanzada no es una habilidad monolítica: son piezas pequeñas y concretas que colaboran. Ejemplo de flujo: la skill de *Investigación* extrae datos crudos → alimentan a la de *Redacción estratégica* → que pasa el testigo a la de *Humanización* para el tono. Si el tono falla, ajustas una pieza — no todo el motor.

Mantenibilidad

Actualizar tu estándar de marca solo toca la skill de "Diseño", sin comprometer la lógica de "Análisis".

Reutilización

Una skill de "Verificación de hechos" sirve igual a un reporte financiero, un email o un artículo técnico.

Precisión quirúrgica

Al limitar el foco de cada pieza reduces el ruido en el contexto y maximizas el razonamiento en esa tarea específica.

La Regla de los 30 Días

La meta no es ahorrar minutos hoy: es que tras un mes tu Claude sea 10 veces más capaz que el primer día, porque heredó sistemáticamente tu experiencia, tus criterios y tu rigor. Documentar procesos y blindar metodologías es dejar de ser operario de prompts para volverte
arquitecto de sistemas
— tus Skills son el activo digital más valioso de tu carrera.

Biblioteca de Skills: tus primeros sistemas

Ocho plantillas para las tareas personales más comunes. Las primeras que conviene instalar: el optimizador de prompts (está en el banner del Nivel 2), el humanizador de textos y el verificador de datos — juntos forman tu control de calidad de salida. Instálalas creando la carpeta `.claude/skills/<nombre>/SKILL.md` (Claude Code) o subiéndolas en Ajustes → Capabilities (app). Edita las reglas a tu gusto — son tuyas.

Correo profesional

Redacta correos en tu voz, con tus reglas, sin que las repitas cada vez.

SKILL.MD

```
---
name: correo-profesional
description: Redacta correos en mi estilo
---
Cuando me ayudes a redactar un correo:
1. Pregunta destinatario y objetivo si no los di
2. Saludo breve, sin "Espero que estés bien"
3. Máximo 120 palabras; el pedido concreto en el primer párrafo
4. Cierre: "Saludos, Mario"
5. Tono: directo pero cordial; nada de jerga corporativa
6. Muéstrame el borrador y espera mi OK antes de darlo por final
```

Minuta de reunión

Convierte notas desordenadas (o un dictado por voz) en minuta con acuerdos y responsables.

SKILL.MD

```
---
name: minuta-reunion
description: Convierte notas de reunión en minuta estructurada
---
Cuando te pase notas o audio de una reunión:
1. Estructura: Asistentes / Temas tratados / Acuerdos / Pendientes
2. Cada acuerdo con responsable y fecha; si faltan, márcalo [FALTA]
3. Pendientes ordenados por urgencia
4. Máximo 1 página; prosa breve, no transcripción
5. Al final, lista de dudas que la reunión dejó abiertas
```

Extracción de datos

Facturas, recibos o PDFs → tabla limpia. Ideal con Cowork sobre una carpeta.

SKILL.MD

```
---
name: extraccion-datos
description: Extrae datos de facturas y PDFs a tabla
---
Cuando te pida extraer datos de documentos:
1. Campos por defecto: fecha, emisor, concepto, subtotal, impuestos, total
2. Salida: archivo .xlsx con una fila por documento
3. Los montos como números, no texto; fechas en formato AAAA-MM-DD
4. Columna final "confianza": alta/media/baja según legibilidad
5. Lista aparte de documentos que no pudiste procesar y por qué
6. NUNCA inventes un valor ilegible — déjalo vacío y márcalo
```

Brief de investigación

Investigaciones con formato fijo y fuentes verificables, siempre igual.

SKILL.MD

```
---
name: brief-investigacion
description: Investiga un tema y entrega un brief estándar
---
Cuando te pida investigar un tema:
1. Antes de empezar: propón el plan (preguntas, fuentes) y espera OK
2. Usa búsqueda web; prioriza fuentes primarias y datos con fecha
3. Formato: Resumen (5 líneas) / Hallazgos / Implicaciones para mí / Fuentes
4. Cada hallazgo con su cita; separa hechos de interpretaciones
5. Si dos fuentes se contradicen, dilo — no promedies
6. Máximo 2 páginas
```

Revisión de código

Checklist fijo para que toda revisión cubra lo mismo (complementa /code-review).

SKILL.MD

```
---
name: revision-codigo
description: Mi checklist de revisión de código
---
Cuando revises código mío o de un PR:
1. Orden: correctitud → seguridad → legibilidad → rendimiento
2. Seguridad siempre: inputs sin validar, secretos hardcodeados, inyección
3. Cada hallazgo: severidad (alta/media/baja) + ubicación + fix sugerido
4. Señala también lo que está BIEN hecho (una línea)
5. Si el cambio necesita test y no lo trae, es hallazgo de severidad alta
6. Veredicto final: aprobar / aprobar con cambios / rechazar
```

Checklist de deploy

La secuencia previa a producción, para no confiar en la memoria un viernes.

SKILL.MD

```
---
name: checklist-deploy
description: Verificación previa a deploy a producción
---
Cuando te pida preparar un deploy:
1. Corre la suite completa de tests y repórtame el resultado
2. /security-review sobre el diff de la rama
3. Verifica migraciones pendientes y su plan de rollback
4. Revisa que no haya secretos ni URLs de dev en el diff
5. Genera el changelog desde los commits
6. Pregunta: ¿hay feature flags que activar/desactivar?
7. NO ejecutes el deploy — prepara todo y espera mi orden explícita
```

Humanizador de textos

Le quita el tono robótico a cualquier borrador. Junto con el optimizador, la primera que deberías instalar.

SKILL.MD

```
---
name: humanizador
description: Reescribe textos con tono humano y natural
---
Cuando te pida humanizar un texto:
1. Elimina muletillas de IA: "En resumen", "Es importante
   destacar", "En el mundo actual", "cabe mencionar".
2. Prohibido: "revolucionario", "disrupción", "sinergia",
   "potenciar", "game-changer".
3. Varía la longitud de las frases; una corta de vez en cuando.
4. Usa giros de conversación real en mi español.
5. Mantén datos y mensaje intactos — cambia la piel, no el
   esqueleto.
6. Entrégame el texto y una lista breve de qué "olía a IA".
```

Verificador de datos

Caza alucinaciones antes de que publiques. La pieza reutilizable por excelencia: sirve igual a un reporte, un correo o un guion.

SKILL.MD

```
---
name: verificador-datos
description: Valida la veracidad de afirmaciones antes de publicar
---
Cuando te pida verificar un texto:
1. Extrae TODAS las afirmaciones factuales (cifras, fechas,
   nombres, causalidades) en una lista numerada.
2. Clasifica cada una: verificada (con fuente), plausible sin
   fuente, o sospechosa.
3. Para las sospechosas: busca en la web y confirma o corrige.
4. Señala afirmaciones absolutas ("siempre", "el único", "el
   mejor") que convenga matizar.
5. Entrégame el texto corregido y la tabla de verificación.
6. Si algo no se pudo verificar, se marca — NUNCA se deja pasar.
```

Cómo nacen las buenas Skills

No se escriben desde cero: se destilan. Cuando un prompt te funcione muy bien dos veces, pídele a Claude: "convierte estas instrucciones en un SKILL.md" — y existe `skill-creator`, una skill oficial que crea, prueba y optimiza otras skills. También hay bibliotecas externas de skills hechas por la comunidad (como skillsmp.com) — revisa antes de construir de cero, y audita lo que instales.

Nivel 4 · Experto — delegar como a un equipo

Sales del navegador. Claude trabaja directamente sobre tus carpetas y repos, los agentes se especializan y coordinan entre sí, y las rutinas corren en la nube aunque tu computadora esté apagada.

Claude Code: el agente con manos

Terminal, app de escritorio o IDE: Claude planifica proyectos, crea y edita archivos, ejecuta y prueba código, y corrige sus propios errores sobre la marcha. Todo lo que sigue en este nivel — prácticas, flujos y los ~85 comandos — es su manual de operación. Arranque en la guía del Día 6.

Equipos de agentes especializados

En vez de un chat que lo hace todo, roles coordinados: un estratega define el ángulo, un copywriter redacta sobre esa estrategia, un revisor audita el resultado. Cada subagente con contexto propio y limpio.

PROMPT · MONTAR UN EQUIPO

```
Crea dos subagentes para este proyecto:
1. "estratega": analiza [mercado/audiencia] y define el ángulo de
   venta, el mensaje central y 3 objeciones a resolver.
2. "copywriter": con la estrategia del primero, redacta [los
   emails / la landing], sin desviarse del mensaje central.
Coordínalos en ese orden y entrégame: estrategia, textos, y una
nota de qué decisiones tomó cada uno.
```

Rutinas: automatización sin computadora encendida

Con `/schedule` las tareas complejas corren solas en la nube de Anthropic — con activadores por horario (cron), webhook o API: revisar correos, resumirlos, buscar noticias, dejar borradores. Sin plataformas de terceros ni tu máquina prendida. Las dos recetas programadas del Recetario (briefing diario, cierre semanal) son el punto de partida.

El orquestador es el **lenguaje natural** — sustituye la lógica de nodos de herramientas como N8N. El flujo: *Trigger* (lunes 9:00) → *lógica en lenguaje natural* (Claude razona y decide) → *conector MCP* (actúa en Gmail/Notion). Ejemplo real: se despierta solo, lee tus correos por el MCP de Gmail, los clasifica con criterio de estratega, y te deja en Notion el resumen y los borradores de respuesta — mientras duermes.

Claude Design: prototipar tocando

Interfaz visual para diseñar y refinar interfaces con detalle estético: seleccionas y modificas elementos directamente en vez de reescribir prompts. Y con `/design-sync` le subes tu design system real para que diseñe con tus componentes.

Eres Nivel 4 pleno cuando…

☐ Tu primera sesión de Claude Code pasó por /plan y /diff

☐ Montaste un equipo de 2+ agentes coordinados

☐ Al menos una rutina corre sola cada semana

☐ Tu pregunta ya no es "¿qué le pido?" sino "¿qué le delego?"

Claude Code: las 6 prácticas

Las prácticas que separan resultados mediocres de resultados excelentes, ordenadas por impacto. Cada una incluye por qué funciona, cómo aplicarla y el error que la arruina.

1 · Planifica antes de tocar código

No saltes directo a programar. Pide primero el enfoque, revísalo, cuestiónalo, y solo entonces autoriza la ejecución. Claude Code tiene un modo dedicado (`/plan`) que fuerza exactamente esta disciplina: propone el plan completo y espera tu aprobación antes de tocar un archivo.

Por qué funciona: corregir un plan cuesta segundos; corregir 400 líneas de código construidas sobre el enfoque equivocado cuesta horas. El plan además te revela qué entendió Claude de tu pedido — los malentendidos salen a la luz antes de ser caros.

PROMPT

```
Explícame cómo resolverías X. Aún no escribas código. Incluye: archivos que
tocarías, riesgos, y qué alternativas descartaste y por qué.
```

Revisa el plan, refínalo, y solo entonces ejecuta

Dejar que programe sin pensar la solución

2 · Define reglas con CLAUDE.md

Un archivo central de reglas en la raíz del repo que Claude lee antes de cada tarea. Es tu contrato permanente: estándares de código, estructura de carpetas, convenciones de nombres, comandos del proyecto, qué no tocar. Genéralo con `/init` y edítalo sin salir de la sesión con `/memory`.

Por qué funciona: todo lo que pongas ahí deja de ocupar espacio en tus prompts. Es la diferencia entre repetir las reglas 50 veces y escribirlas una.

CLAUDE.MD EJEMPLO

```
# Reglas del proyecto
- Python 3.12, tipado estricto, Ruff para lint/format
- Tests en /tests con Pytest; toda función pública lleva test
- Nombres: snake_case funciones, PascalCase clases
- Commits: convencional (feat:, fix:, chore:)
- NUNCA tocar /migrations sin pedir confirmación
- Correr `make check` antes de dar una tarea por terminada
```

Corto, específico y accionable; añade linting desde el inicio

Documentos largos y vagos que Claude no puede obedecer

3 · Desarrollo guiado por tests (TDD)

Empieza con un test que falla. Dale el test a Claude y deja que implemente hasta que pase. Los tests son la especificación ejecutable: definen el comportamiento esperado sin ambigüedad, y le dan a Claude un criterio de éxito objetivo contra el cual iterar solo.

Por qué funciona: sin tests, "ya quedó" significa "compila". Con tests, significa "cumple los 12 casos que definimos, incluidos los límite". Claude itera contra los tests sin que tengas que revisar cada intento intermedio.

PROMPT

```
Para esta funcionalidad: [descríbela]. 1) Escribe una suite de tests en Pytest con 10+ casos, cubriendo casos límite y entradas inválidas. 2) Muéstramelos y espera mi OK. 3) Escribe la implementación y corre los tests hasta que todos pasen.
```

Los tests definen el comportamiento esperado con claridad

Saltarte los tests y corregir bugs después, uno por uno

4 · Tareas paralelas con subagentes

Divide el trabajo en unidades independientes y lanza varios agentes a la vez con `/agents`: uno genera código, otro revisa, otro refactoriza. Cada subagente trabaja con su propio contexto limpio y te reporta solo la conclusión.

Por qué funciona: además del tiempo, ganas calidad — un agente revisor con contexto fresco encuentra errores que el agente que escribió el código ya no ve (el mismo sesgo que tienen los humanos revisando su propio trabajo).

PROMPT

```
Divide esta tarea en unidades independientes y lánzalas en paralelo: implementación de [X], revisión de seguridad del módulo [Y], y actualización de los tests de [Z]. Repórtame un resumen consolidado al final.
```

Combina implementación + review simultáneos

Hacer todo paso a paso, en serie, manualmente

5 · Aísla el trabajo con git worktrees

Un worktree es una copia de trabajo del repo en otra carpeta, apuntando a otra rama — sin clonar de nuevo. Una feature = un worktree = una sesión de Claude. Puedes tener tres sesiones construyendo tres features a la vez, sin pisarse los archivos.

Por qué funciona: el enemigo del trabajo paralelo con agentes es el estado compartido. Worktrees eliminan el problema de raíz: cada sesión ve su propio sistema de archivos.

COMANDOS

```
git worktree add ../proyecto-feature-a feature-a
git worktree add ../proyecto-feature-b feature-b
# una sesión de Claude Code en cada carpeta
git worktree list   # ver los activos
git worktree remove ../proyecto-feature-a   # al terminar
```

Desarrollo paralelo sin conflictos

Construirlo todo sobre una sola rama

6 · Mantén el contexto limpio

Los chats largos degradan la calidad: el contexto se llena de intentos fallidos, archivos irrelevantes y decisiones viejas que confunden al modelo. Usa `/context` para ver qué está llenando la ventana y `/compact` para resumir el historial y liberar espacio. Para comentarios al margen que no deben ensuciar el historial, usa `/btw`.

Por qué funciona: el rendimiento del modelo no es constante a lo largo de la ventana de contexto — señal clara y reciente gana sobre volumen. Gestionar contexto es gestionar calidad.

Sesión fresca por feature; contexto pequeño al empezar un proyecto

Una megasesión para todo; pegar directorios enteros sin necesidad

Flujos de trabajo probados

Recetas completas, listas para copiar. Cada una resuelve un escenario que vas a encontrar tarde o temprano.

El loop TDD completo

CORE

Test → implementación → correr tests → repetir hasta verde. El flujo con mejor relación esfuerzo/calidad que existe para trabajar con agentes de código. La clave está en el paso 2: revisar los tests *antes* de la implementación te da control sobre la especificación sin leer una línea del código final.

PROMPT

```
Para esta funcionalidad: [descríbela]. 1) Escribe una suite de tests en Pytest
con 10+ casos cubriendo casos límite y entradas inválidas. 2) Muéstramelos y
espera mi OK antes de continuar. 3) Escribe la implementación que pase todos los
tests. 4) Corre los tests y corrige hasta que todo pase. 5) Termina con un
resumen de decisiones tomadas.
```

CI/CD automático

DEPLOY

El siguiente paso natural del TDD: que los tests corran solos en cada commit. Ahorra tiempo de revisión manual y se integra directo a tu pipeline. Después pídele también el badge de cobertura y las reglas de protección de rama.

PROMPT

```
Crea un archivo YAML de GitHub Actions que corra mi suite de Pytest en cada
commit y publique un reporte de cobertura como comentario en el PR. Añade caché
de dependencias para que corra rápido.
```

Código legacy

REFACTOR

Para archivos viejos y enredados, pide oportunidades concretas y acotadas — nunca una reescritura total (esa es la receta del desastre). Y no pegues directorios enteros: dale el contexto mínimo necesario. Si el repo es muy grande (cientos de archivos), deja que un subagente explore y traiga solo lo relevante.

PROMPT

```
Analiza este archivo JavaScript antiguo: [contenido]. Encuentra 5 oportunidades de refactor que reduzcan complejidad sin romper la lógica. Para cada una: qué cambiar, por qué, y el riesgo. Usa ejemplos con sintaxis actual. No reescribas todo el archivo.
```

Debugging sistemático

DEBUG

El anti-patrón es pegar el error y decir "arréglalo": Claude adivina, parcha el síntoma, y el bug vuelve. El patrón correcto es forzar diagnóstico antes de corrección — o directamente usar `/debug`, que ya trae esta disciplina.

PROMPT

```
Este es el error: [error + stack trace]. Antes de proponer una corrección: 1) Formula 3 hipótesis de causa raíz. 2) Dime qué evidencia confirmaría o descartaría cada una. 3) Verifica las hipótesis leyendo el código. 4) Solo entonces propón la corrección para la causa confirmada.
```

Investigación profunda

RESEARCH

Para reportes multi-fuente, el error típico es soltar la pregunta y recibir un reporte con el enfoque equivocado 20 minutos después. Pide el plan primero: corregir el plan cuesta un minuto.

PROMPT

```
Quiero investigar [tema] para [decisión que voy a tomar]. Antes de investigar: propónme un plan — qué preguntas responderás, qué fuentes usarás y qué estructura tendrá el reporte final. Espera mi OK para empezar.
```

Empezar bien un proyecto

SETUP

Si tu proyecto no está estructurado para IA, usa una estructura base simple: reglas separadas de tareas, tareas separadas de herramientas. Y conecta pocos repos clave — nunca un monorepo gigante y desordenado al primer intento: primero demuestra que el flujo funciona en pequeño.

ESTRUCTURA

```
ai-project/
├── system/   ← reglas e instrucciones (CLAUDE.md vive aquí conceptualmente)
├── tasks/    ← specs y tareas pendientes, una por archivo
└── tools/    ← scripts y utilidades del proyecto
```

Comandos de Claude Code — referencia completa

Extraída de la referencia oficial (julio 2026): ~85 comandos organizados por lo que quieres lograr, más atajos de teclado y comandos de terminal. **Haz clic en cualquier comando para copiarlo.** No todos aparecen para todos los usuarios — dependen de plataforma y plan; escribe / para ver los tuyos.

El mapa por momento del flujo

Primera sesión en un repo

`/init` genera el CLAUDE.md → `/memory` lo refina → `/mcp` conecta servidores → `/permissions` define reglas de aprobación.

Durante la tarea

`/plan` antes de cambios grandes; `/model` y `/effort` ajustan potencia; `/context` y `/compact` cuando crece; `/btw` para apartes.

En paralelo

`/agents` para subagentes, `/batch` para cambios masivos, `/background` suelta la sesión, `/tasks` vigila todo.

Antes de enviar

`/diff` muestra qué cambió, `/code-review` caza bugs (`--fix` los aplica), `/simplify` limpia, `/security-review` revisa seguridad.

SESIONES Y CONTEXTO

`/clear [nombre]` Conversación nueva con contexto vacío; la anterior queda disponible en /resume. Pasa un nombre para etiquetarla. **Alias: /new, /reset**

`/resume [sesión]` Retoma una conversación por nombre o ID, o abre el selector — incluye las sesiones en segundo plano marcadas "bg". **Alias: /continue**

`/rename [nombre]` Nombra la sesión y muéstralo en la barra; sin argumento genera un nombre desde el historial.

`/recap` Resumen de una línea de la sesión, bajo demanda. El automático aparece solo cuando vuelves tras ausentarte.

`/branch [nombre]` Ramifica la conversación en este punto para probar otra dirección sin perder la original (vuelves con /resume).

`/fork <directiva>` Subagente en segundo plano que **hereda toda la conversación** y trabaja la directiva mientras tú sigues; su resultado regresa solo.

`/rewind` Regresa código y/o conversación a un punto anterior (checkpoints), o resume un tramo. **Alias: /undo, /checkpoint**

`/compact [enfoque]` Libera contexto resumiendo la conversación; acepta instrucciones de qué priorizar en el resumen.

`/context [all]` Cuadrícula visual de qué está llenando tu ventana de contexto, con sugerencias de optimización.

`/btw <pregunta>` Pregunta al margen: ve toda la conversación pero no entra al historial. Funciona incluso mientras Claude trabaja; pulsa "f" para convertirla en sesión propia.

`/cd <ruta>` Mueve la sesión a otro directorio de trabajo conservando la caché de la conversación.

`/add-dir <ruta>` Da acceso a un directorio adicional sin mover la sesión.

`/export [archivo]` Exporta la conversación como texto plano — al portapapeles o directo a un archivo.

`/copy [N]` Copia la última respuesta (o la N-ésima). Con bloques de código, abre un selector; "w" los escribe a archivo (útil por SSH).

`/desktop` Continúa esta sesión en la app Claude Code Desktop. **Alias: /app**

`/teleport` Trae una sesión de Claude Code web a esta terminal: rama y conversación incluidas. **Alias: /tp**

`/remote-control` Haz esta sesión local controlable desde claude.ai en otro dispositivo. **Alias: /rc**

PLANEAR Y PENSAR

`/plan [tarea]` Modo plan: Claude propone el enfoque completo y espera tu aprobación antes de tocar archivos. Ej: **/plan arregla el bug de auth**.

`/model [modelo]` Cambia de modelo y lo guarda como default; en el selector, "s" cambia solo para esta sesión.

`/effort [nivel]` Profundidad de razonamiento: low, medium, high, xhigh, max, o **ultracode** (xhigh + orquestación automática de workflows). Sin argumento abre un slider.

`/advisor [modelo]` Un segundo modelo (opus, sonnet, fable) aconseja en momentos clave de la tarea. "off" lo desactiva.

`/goal [condición]` Fija una meta: Claude sigue trabajando turno tras turno hasta cumplirla. "clear" la cancela.

`/fast [on|off]` Alterna el modo rápido.

`/ultraplan <prompt>` Redacta un plan en una sesión en la nube, revísalo en el navegador, y ejecútalo remoto o de vuelta en tu terminal.

`/deep-research <pregunta>` **Workflow:** abanico de búsquedas web, verificación cruzada de fuentes y reporte final con citas.

PARALELO Y SEGUNDO PLANO

`/agents` Crea y gestiona subagentes (viven en **.claude/agents/**) — o simplemente pídele a Claude que los cree.

`/batch <instrucción>` Descompone un cambio grande en 5–30 unidades independientes; cada una corre en su propio worktree y abre su PR. Ej: **/batch migra src/ de Solid a React**.

`/background` Suelta la sesión completa a segundo plano y libera tu terminal; monitoréala con **claude agents**. **Alias: /bg**

`/tasks` Ve y gestiona todo lo que corre en segundo plano de la sesión: shells y subagentes. **Alias: /bashes**

`/stop` Detiene la sesión en segundo plano a la que estás conectado (transcript y worktree se conservan).

`/loop [intervalo] [prompt]` Repite un prompt mientras la sesión siga abierta. Ej: **/loop 5m revisa si terminó el deploy**. **Alias: /proactive**

`/schedule [descripción]` Rutinas programadas que corren en la nube de Anthropic; Claude te guía conversacionalmente. **Alias: /routines**

`/autofix-pr [prompt]` Sesión en la nube que vigila el PR de tu rama y empuja arreglos cuando CI falla o los reviewers comentan.

`/workflows` Ve, pausa, reanuda o guarda los workflows en curso.

REVISAR Y ENVIAR

`/diff` Visor interactivo de cambios: diff sin commitear y diffs por turno de Claude, navegable por archivo.

`/code-review [nivel] [--fix]` Revisa el diff buscando bugs de corrección y limpiezas. **--fix** aplica los hallazgos; **--comment** los publica en el PR; **ultra** corre revisión profunda multi-agente en la nube.

`/review [PR]` La misma revisión, sobre un pull request de GitHub por número; sin argumento lista los PRs abiertos.

`/simplify [target]` 4 agentes en paralelo limpian el código (reuso, simplificación, eficiencia, nivel de abstracción) y aplican los fixes. No busca bugs — para eso, /code-review.

`/security-review` Pasa de seguridad sobre el diff de la rama: inyección, problemas de auth, exposición de datos.

`/run` Lanza y opera tu app para **ver** el cambio funcionando — no solo tests en verde.

`/verify` Confirma que un cambio hace lo que debe: construye la app, la corre y observa el resultado.

`/run-skill-generator` Le enseña a /run y /verify cómo construir y lanzar **tu** app, escribiendo una skill por proyecto.

`/install-github-app` Instala la GitHub App de Claude en un repo, con configuración opcional de Actions y secrets.

SKILLS, PLUGINS Y MCP

`/skills` Lista tus skills; escribe para filtrar, "t" ordena por tokens, Espacio cambia la visibilidad de cada una.

`/reload-skills` Re-escanea los directorios de skills para cargar las añadidas en disco sin reiniciar.

`/plugin [subcomando]` Menú de plugins, o directo: **list, install, enable, disable**.

`/reload-plugins` Recarga los plugins activos para aplicar cambios pendientes sin reiniciar.

`/mcp` Gestiona servidores MCP y su autenticación OAuth: **reconnect, enable, disable** por servidor o "all".

`/claude-api [migrate]` Carga la referencia del API de Claude para tu lenguaje; **migrate** actualiza tu código a modelos nuevos (IDs, thinking, parámetros).

`/dataviz [petición]` Guía de diseño para gráficas y dashboards: elige la forma correcta, valida paleta por daltonismo y contraste.

`/design-sync [nombre]` Convierte tu design system React y súbelo a Claude Design para que los diseños usen tus componentes reales.

CONFIGURA Y PERSONALIZA

`/init` Genera el CLAUDE.md inicial analizando tu repo. Primer comando en todo proyecto nuevo.

`/memory` Edita los archivos CLAUDE.md y gestiona auto-memory sin salir de la sesión.

`/permissions` Reglas allow/ask/deny por herramienta, por alcance; revisa denegaciones recientes del modo auto. **Alias: /allowed-tools**

`/fewer-permission-prompts` Escanea tus transcripts y arma una allowlist priorizada para reducir preguntas de permiso.

`/config [clave=valor]` Ajustes completos; directo sin menú: **/config theme=dark**, **/config model=sonnet**, **/config thinking=false**. **Alias: /settings**

`/theme` Temas de color: auto (según tu terminal), claros, oscuros, accesibles para daltonismo, y personalizados.

`/color [color]` Color de la barra de prompt de esta sesión: red, blue, green, yellow, purple, orange, pink, cyan.

`/statusline` Configura la línea de estado — descríbele lo que quieres mostrar.

`/keybindings` Abre tu archivo de atajos de teclado para re-mapearlos.

`/terminal-setup` Instala el atajo Shift+Enter en terminales que lo necesitan (VS Code, Cursor, Zed…).

`/sandbox` Alterna el modo sandbox (aislamiento de ejecución), en plataformas soportadas.

`/hooks` Ve la configuración de hooks: scripts tuyos que corren en eventos de herramientas.

`/ide` Gestiona integraciones con tu IDE y muestra su estado.

`/tui [fullscreen]` Cambia el renderer de la terminal; **fullscreen** activa el modo sin parpadeo con transcript navegable.

CUENTA, USO Y DIAGNÓSTICO

`/usage` Costo de sesión, límites del plan y desglose por skill, subagente, plugin y servidor MCP. **Alias: /cost, /stats**

`/status` Versión, modelo, cuenta y conectividad — funciona incluso mientras Claude responde.

`/doctor` Diagnostica instalación y configuración; pulsa "f" para que Claude repare lo reportado.

`/debug [descripción]` Activa logging de debug desde este punto y analiza el log de la sesión para resolver problemas.

`/insights` Reporte de tus patrones de uso: áreas de proyecto, interacción y puntos de fricción.

`/team-onboarding` Genera una guía de onboarding para un compañero desde tu historial de 30 días, con link compartible.

`/release-notes` Changelog interactivo, versión por versión.

`/feedback` Reporta un bug con el contexto de la sesión adjunto. **Alias: /bug**

`/login · /logout` Inicia o cierra sesión de tu cuenta Anthropic.

`/privacy-settings` Ve y ajusta tu configuración de privacidad (Pro/Max).

`/upgrade` Abre la página para subir de plan.

`/usage-credits` Configura créditos extra para seguir trabajando al llegar a un límite.

`/help` Ayuda y lista de comandos disponibles.

`/powerup` Aprende features con lecciones interactivas y demos animadas.

`/voice [hold|tap]` Dictado por voz: mantén o toca Espacio para hablar.

`/mobile · /radio · /stickers` QR de la app móvil · radio lo-fi de Claude FM · pide stickers. Los extras existen.

Atajos de teclado esenciales

Los que usarás a diario dentro de una sesión. En macOS, los de Option requieren configurar "Option como Meta" en tu terminal.

`Esc` Interrumpe a Claude a mitad de turno para redirigirlo — conserva el trabajo hecho hasta ahí.

`Esc + Esc` Con texto en el input: lo limpia (recuperable con ↑). Con input vacío: abre el menú de **rewind** para restaurar código y conversación.

`Shift + Tab` Cicla los modos de permiso: manual → aceptar ediciones → plan → auto.

`Ctrl + O` Visor de transcript: detalle completo de herramientas y llamadas MCP expandidas.

`Ctrl + R` Búsqueda inversa en tu historial de prompts; Ctrl+S cambia el alcance (sesión / proyecto / todos).

`Ctrl + B` Manda el comando o agente en curso a segundo plano (en tmux, dos veces).

`Ctrl + T` Muestra/oculta la checklist de tareas de Claude.

`Ctrl + G` Edita tu prompt en tu editor de texto por defecto — para prompts largos.

`Ctrl + V` Pega una imagen del portapapeles como [Image #N] referenciable en el prompt.

`! al inicio` Modo shell: corre el comando directo (ej. **! npm test**), la salida entra al contexto y Claude la comenta.

`@` Autocompletado de rutas de archivo dentro del prompt.

`\ + Enter` Salto de línea sin enviar (Shift+Enter funciona nativo en iTerm2, Warp, Windows Terminal…).

`Option/Alt + P` Cambia de modelo sin borrar lo que llevas escrito.

`Option/Alt + T` Alterna extended thinking (sin efecto en Fable 5, que siempre lo usa).

`Tab` Acepta la sugerencia de prompt que aparece en gris.

Desde la terminal (antes de entrar a la sesión)

Los comandos CLI que lanzan y administran sesiones desde fuera.

Arrancar

BASH

```
claude                    # sesión interactiva
claude "explica este proyecto"
claude -p "¿qué hace esta función?"   # respuesta única, sin sesión
claude -c                 # continúa la conversación más reciente
claude -r "auth-refactor" "termina este PR"
```

Agentes en segundo plano

```bash
claude agents             # lista las sesiones en segundo plano
claude attach 7c5dcf5d    # conéctate a una
claude logs 7c5dcf5d      # ve su salida
claude stop 7c5dcf5d      # deténla
claude rm 7c5dcf5d        # elimínala
```

Mantenimiento

```bash
claude update             # actualiza Claude Code
claude auth status        # verifica tu sesión
claude mcp                # gestiona servidores MCP
claude plugin install code-review@claude-plugins-official
```

Tips para recordar

`/effort` alto en problemas complejos · `/agents` o `/batch` para ahorrar horas · `/usage` a menudo para controlar costo · y ante la duda, escribe `/` y filtra.

Automatiza lo aburrido

Todo lo que haces más de dos veces es candidato a automatizarse. La escalera tiene tres peldaños: hacerlo con Claude → convertirlo en Skill → programarlo para que corra solo.

Scheduled Tasks

Tareas programadas que corren solas, con o sin ti presente: un briefing de noticias cada mañana a las 6, el resumen semanal de correos sin responder, un chequeo diario de "¿algo se rompió?".

La señal: si tu pedido incluye "cada mañana", "todos los lunes" o "recuérdame en una hora" — eso es una scheduled task, no un pedido normal.

SE ACTIVA: pidiéndolo tal cual en la conversación ("cada viernes a las 3 prepara X") — Claude crea la tarea programada; las existentes se listan y editan desde el mismo chat o ajustes.
FUNCIONA: a la hora fijada, Claude ejecuta el prompt guardado con tus conectores y te deja el resultado listo.

Ejemplos que funcionan: "cada día a las 7am, resume mis correos nuevos y mi agenda del día" · "cada viernes, borrador del status report de la semana" · "el día 1 de cada mes, organiza mi carpeta de descargas".

Skills: anatomía

Una Skill vive en `.claude/skills/nombre/SKILL.md` (o se instala en la app). Claude la carga solo cuando es relevante, así que el material largo no cuesta contexto hasta usarse. Estructura mínima:

SKILL.MD

```
---
name: reporte-semanal
description: Genera mi status report de los viernes
---
Cuando me pidan el reporte semanal:
1. Revisa los commits y PRs de la semana
2. Estructura: logros / bloqueos / siguiente semana
3. Tono directo, máximo 1 página, sin viñetas anidadas
4. Termina preguntándome si falta algo antes de dar por cerrado
```

Skills como estándar de equipo

El problema real de los equipos no es que la IA responda mal: es que cada persona promptea distinto y obtiene resultados inconsistentes. Skills compartidas (checklist de code review, pasos de deploy, formato de specs) son la solución: todos invocan el mismo flujo con la misma calidad.

Empieza por: la tarea donde más se nota la inconsistencia entre personas.

Plugins

Paquetes instalables que agrupan MCPs + Skills + herramientas para un dominio (operaciones, ventas, legal...). Extienden capacidades de golpe y se comparten en marketplaces. Antes de construir algo desde cero, revisa si ya existe el plugin.

SE ACTIVA: en la app: Ajustes → `Capabilities` → sección de plugins/marketplaces. En Claude Code: comando `/plugin` para añadir marketplaces e instalar. **FUNCIONA:** al instalarse, sus Skills, conectores y comandos aparecen en tu sesión como si fueran nativos.

El patrón general: la escalera de automatización

Peldaño 1 — Hazlo una vez a mano con Claude. Si el resultado sirvió, ya validaste que la tarea es automatizable. **Peldaño 2 — Conviértelo en Skill.** Captura las instrucciones exactas que funcionaron; ahora es repetible con un comando. **Peldaño 3 — Prográmalo.** Si además es recurrente en el tiempo, scheduled task y te olvidas.

La inversión es el setup. El retorno son horas cada semana. Y el error clásico es saltar al peldaño 3 sin pasar por el 1: automatizar algo que nunca verificaste a mano.

Lo que Claude no hace (y qué hacer al respecto)

Un manual honesto incluye los límites. Cada uno viene con su mitigación — porque casi todos tienen una.

Puede equivocarse en cálculos y salidas complejas

Los modelos de lenguaje no son calculadoras: en aritmética larga, conteos y agregaciones "de cabeza" pueden fallar con total confianza.

MITIGACIÓN: pídele que calcule ejecutando código ("hazlo con Python y muéstrame el script") — ahí la precisión es la de la máquina, no la del modelo. Y verifica las cifras que van a decisiones importantes.

No accede a herramientas ni datos que no conectaste

Si le preguntas por tus ventas de ayer sin conector a tu base de datos, no puede saberlo — y el riesgo es que algunos modelos "rellenan" con algo plausible.

MITIGACIÓN: conecta el MCP correspondiente o pásale el archivo. Y si sospechas que inventó un dato, pregunta directamente: "¿esto lo leíste de mis datos o lo estás asumiendo?"

No es quien decide — la responsabilidad queda contigo

Claude propone, redacta y ejecuta, pero el juicio final sobre enviar, publicar, desplegar o firmar es tuyo. Esto no es letra chica: es el modelo operativo correcto.

MITIGACIÓN: establece puntos de control en tus flujos: "muéstrame antes de enviar", "espera mi OK antes de hacer commit". Los flujos de este manual ya los incluyen.

Los chats largos degradan la calidad

No es falta de voluntad sino física del contexto: la señal se diluye entre intentos fallidos y material viejo.

MITIGACIÓN: sesiones frescas por tema, /compact cuando crezca, /context para diagnosticar, y Projects para que el contexto que sí importa persista sin ocupar la conversación.

El conocimiento del modelo tiene fecha de corte

Precios, versiones, personas en cargos y noticias posteriores al corte de entrenamiento no están en su memoria.

MITIGACIÓN: para cualquier dato del mundo actual, pide que busque en la web. Si te da un dato presente sin citar fuente, desconfía por sistema.

Desconfía de infografías sobre Claude (sí, incluidas las que originaron este manual)

De las 6 que analizamos: 4 tenían nombres de modelos inventados o typos de generación, y una era relleno genérico. La ironía: contenido sobre IA generado con IA sin verificación humana.

MITIGACIÓN: verificar contra docs.claude.com y code.claude.com/docs — como hicimos con los 21 comandos de este manual antes de incluirlos.

Los 4 principios del manual

1. No te saltes el setup — es donde vive el retorno. 2. Los tests no son negociables. 3. Gestiona el contexto como CEO. 4. Automatiza las partes aburridas, en ese orden: a mano → Skill → programada.

Manual personal de Mario · v5 · Escalones del dominio · Construido con Claude a partir de 6 infografías analizadas y depuradas · Comandos verificados contra la referencia oficial de Claude Code · Datos de modelos y features: verificar periódicamente en docs.claude.com · Rutas de activación verificadas en julio 2026 contra el Centro de ayuda — los menús pueden moverse entre versiones