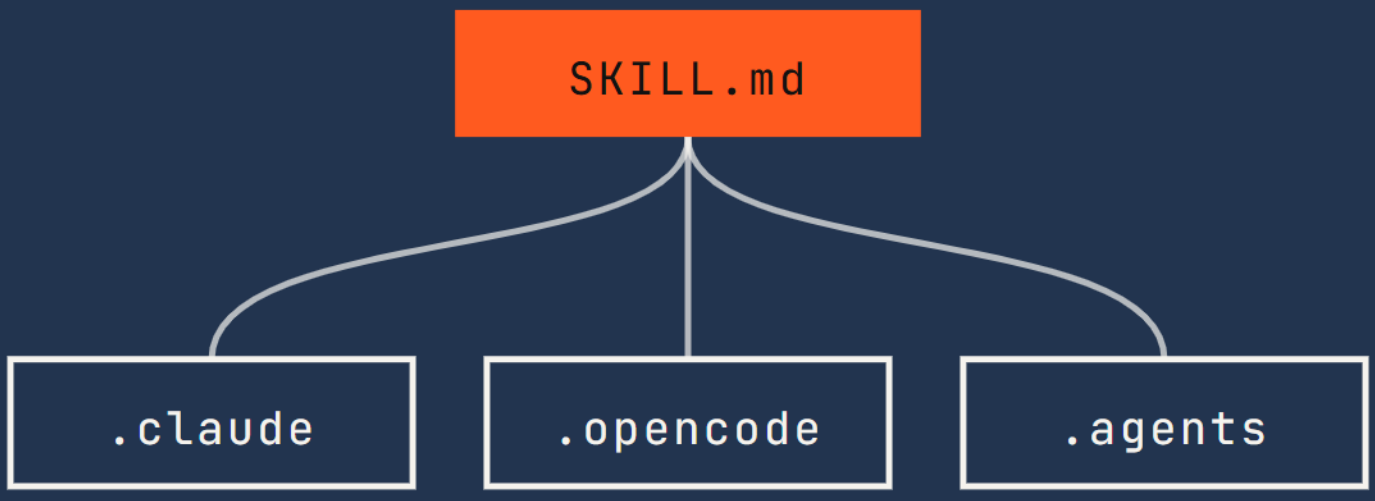
Tu stack de IA, sin suscripciones innecesarias



UNA SKILL · TRES HERRAMIENTAS

Tu stack de IA, sin suscripciones innecesarias

Skills, herramientas y automatizaciones elegidas mediante pruebas, no modas

Para quién: trabajas por tu cuenta o en un equipo pequeño, ya usas varias aplicaciones de IA y quieres un sistema pequeño que puedas mantener y deshacer. Tu stack es el conjunto de herramientas que usas para trabajar. **Qué vas a conseguir:** saber qué te aporta de verdad cada herramienta, quitar lo que se repite y poder cambiar de una a otra sin rehacer el trabajo. **Cómo se probó:** lo que cuenta el libro lo probamos en un PC con Windows 11 y una gráfica NVIDIA de 16 GB, con Claude Code, OpenCode, Codex CLI, Kimi CLI y Ollama. En multimedia, solo la lectura de facturas escaneadas tiene prueba propia (capítulo 06). La receta de correo a borrador (capítulo 07) es un ejercicio para que la midas tú. Las condiciones y precios de cada herramienta, compruébalos en su web oficial el día que decidas.

Qué vas a aprender

Harás un inventario de tu trabajo, distinguirás las piezas de tu stack y escribirás skills que funcionen en varias herramientas. Compararás alternativas con las mismas pruebas, cerrarás permisos y ensayarás cómo dejar una herramienta. No se trata de quitar herramientas por principio, sino de quitar las que se repiten sin aportar un resultado comprobado.

Reglas del libro

- No damos importes ni recomendamos planes: cambian, y se consultan en la web oficial el día de decidir. Sí decimos qué tipo de acceso usamos en las pruebas (suscripción, cuenta gratuita, API o local).
- Los ejemplos usan personas, empresas y datos ficticios.
- Todas las cifras salen de nuestras pruebas. En tus registros, lo que no hayas medido se apunta «sin medir».
- MCP es una forma de conectar herramientas a una IA, no un sistema de permisos.
- Toda integración se prueba primero sin efectos hacia fuera: sin enviar, borrar ni pagar nada.

Contenido

01 · Mapa del trabajo antes del stack **6**

Paso 1 · Apunta tareas, no marcas . . . 6
Paso 2 · Apunta qué tienes de verdad . . . 6
Paso 3 · Dibuja el mapa de dependencias . . . 7
Paso 4 · Clasifica cada pieza . . . 7
Paso 5 · Escribe las duplicidades como sospecha . . . 7
Paso 6 · Elige la primera prueba . . . 8
Un ejemplo real: la combinación que usamos en las pruebas . . . 8
Regla de simplificación . . . 8
Resumen . . . 8

02 · Prompt, instrucciones, skill, asistente, herramienta, agente y MCP **10**

Las siete piezas . . . 10
Paso 1 · Etiqueta tu stack . . . 11
Paso 2 · Conecta un servidor MCP mínimo . . . 11
Paso 3 · Apunta el nombre exacto de cada herramienta en cada cliente . . . 12
Paso 4 · Comprueba la conexión con una llamada real . . . 12
Resumen . . . 13

03 · Diseño y versionado de skills **14**

Paso 1 · Pon siempre la cabecera YAML . . . 14
Paso 2 · Copia la skill a tres carpetas . . . 14
Paso 3 · Verifica que cada herramienta la carga como skill . . . 15
Paso 4 · Escribe el contrato . . . 16
Paso 5 · Versiona cada cambio . . . 16
Paso 6 · Prueba contra respuestas guardadas . . . 16
Señales de que debes dividirla . . . 17
Resumen . . . 17

04 · Evaluación de alternativas **18**

Paso 1 · Rellena la ficha del encargo . . . 18
Paso 2 · Elige un corrector objetivo . . . 18
Paso 3 · Mismo caso, mismas reglas, varias repeticiones . . . 18
Paso 4 · Reúne cuatro tipos de evidencia . . . 19
Paso 5 · Rellena la matriz con estados, no con notas . . . 19
Paso 6 · Haz una integración mínima . . . 19
Paso 7 · Decide con fecha de revisión . . . 19
Lo probamos: una comparación con 72 ejecuciones . . . 20
Resumen . . . 20

05 · Investigación y documentos **21**

Dos flujos, dos controles . . . 21
Paso 1 · Pide respuesta, URL y cita literal . . . 21
Paso 2 · Comprueba cada cita . . . 21
Paso 3 · Usa la tabla de la skill de investigación . . . 21
Paso 4 · Confirma en el registro qué acceso tuvo el modelo . . . 22
Lo probamos: ¿existen las citas? . . . 22
Documentos internos . . . 23
Qué herramienta usar para cada cosa . . . 23
Resumen . . . 24

06 · Datos y multimedia **25**

Paso 1 · Divide el trabajo entre código y modelo . . . 25
Paso 2 · Prepara una muestra en tres calidades . . . 25
Paso 3 · Pide los campos con un valor para lo ilegible . . . 25
Paso 4 · Valida por código . . . 25
Paso 5 · Clasifica los fallos . . . 26
Lo probamos: facturas en imagen . . . 26
Contrato de una skill de datos . . . 26
Gráficos y cifras . . . 26
Imagen y diseño . . . 27
Audio y vídeo . . . 27
Antes de publicar . . . 27
Resumen . . . 27

07 · Tres recetas agnósticas **28**

Receta 1 · Una skill, cuatro herramientas . . . 28
Receta 2 · Script con cualquier modelo: seguimiento.py . . . 28
Receta 3 · Correo a borrador (ejercicio) . . . 30
Registro común de las tres recetas . . . 30
Cómo sustituir una pieza sin romper nada . . . 30
Resumen . . . 30

08 · Permisos y secretos **32**

Paso 1 · Escribe qué necesita cada pieza . . . 32
Paso 2 · Cierra por configuración las acciones hacia fuera . . . 32
Paso 3 · Prefiere una lista de permitidos . . . 32
Paso 4 · Comprueba el bloqueo con una llamada . . . 33
Paso 5 · Guarda las claves en variables de entorno . . . 33
Paso 6 · Separa pruebas y trabajo real . . . 33
Lo probamos: ¿obedece un agente una orden escondida en un documento? . . . 34
Resumen . . . 35

09 · Registros, reintentos y aprobación **36**

Paso 1 · Provoca fallos y mira los reintentos . . . 36
Paso 2 · Lanza lo mismo otra vez . . . 36
Paso 3 · Prueba el validador con una salida manipulada . . . 36
Paso 4 · Aprueba un borrador con nombre . . . 36
Paso 5 · Clasifica los pasos de tus flujos . . . 36
Lo probamos: el registro de una ejecución completa . . . 37
Por qué funciona: las cinco piezas . . . 37
Si algo falla . . . 39
Resumen . . . 39

10 · Auditoría de suscripciones y salida **40**

Paso 1 · Reabre el inventario . . . 40
Paso 2 · Comprueba la disponibilidad con una llamada . . . 40
Paso 3 · Mira si hay señales de revisión . . . 40
Paso 4 · Clasifica con lo que has medido . . . 41
Paso 5 · Prueba la sustitución . . . 41
Paso 6 · Exporta y reabre fuera . . . 41
Paso 7 · Retira la herramienta . . . 42
Paso 8 · Escribe el changelog y la próxima revisión . . . 42
El stack mínimo viable . . . 42
Resumen . . . 42

01 · Mapa del trabajo antes del stack

Si haces la lista de tus aplicaciones de IA antes que la de tus tareas, acabarás justificando lo que ya pagas. Un stack útil empieza por el trabajo: qué entra, qué sale y cómo lo revisas.

Al acabar este capítulo tendrás un inventario de tareas, un mapa de qué depende de qué y una primera sospecha de qué herramientas se repiten. Las decisiones de mantener o retirar llegan en los capítulos 04 y 10, cuando tengas datos.

Paso 1 · Apunta tareas, no marcas

Elige una semana de trabajo normal y apunta cada vez que una IA interviene. Registra qué entra y qué sale, no el nombre del programa.

Tarea	Entrada	Salida	Veces por semana	Cómo se revisa	Herramienta actual
Seguimiento de reunión	Acta con turnos	Acciones, desacuerdos, excepciones	Sin medir	Contra el acta	Sin registrar
Responder solicitud	Correo + ficha de servicio	Borrador de respuesta	Sin medir	Hechos y compromisos	Sin registrar
Comprobar un dato actual	Pregunta	Respuesta con fuente y cita	Sin medir	Abrir la fuente	Sin registrar
Leer facturas escaneadas	Imagen	Campos de la factura	Sin medir	Sumas y registro de proveedor	Sin registrar

Cuatro reglas:

1. La frecuencia se cuenta, no se calcula a ojo. Si no la has contado, escribe «sin medir».
2. Cada fila tiene una persona que usa la salida. Si nadie la usa, sobra la tarea antes que la herramienta.
3. Apunta cómo se revisa hoy. Si la respuesta es «no se revisa», ya has encontrado un riesgo.
4. Usa la plantilla `plantillas/inventario-stack.csv` del paquete.

Cómo sabes que ha ido bien: tienes al menos cinco filas, cada una con entrada, salida y persona que usa el resultado.

Paso 2 · Apunta qué tienes de verdad

Para cada herramienta, anota cuatro datos que puedas comprobar hoy: versión instalada, cuenta con la que la usas, tipo de acceso (suscripción, cuenta gratuita, API o local) y qué tareas del paso 1 cubre. Una API es la puerta por la que un programa usa un modelo pagando por uso, sin la aplicación de chat. Las condiciones comerciales, consúltalas en la web oficial el día de decidir, no de memoria.

Para sacar las versiones, abre PowerShell (la consola de Windows) y pide la versión a cada herramienta de línea de comandos que tengas:

```
claude --version
opencode --version
codex --version
kimi --version
ollama --version
```

Cómo sabes que ha ido bien: cada herramienta tiene su versión apuntada, o un «no responde». Si alguna no responde, no está en el `PATH` (la lista de carpetas donde Windows busca programas) o no la tienes: anótalo

igual. Un inventario con huecos visibles es mejor que uno completo de memoria.

Paso 3 · Dibuja el mapa de dependencias

Escribe cada flujo en una línea, de la entrada a la decisión de una persona:

```
Acta → Claude Code + skill extraer-acciones-acta → tabla → revisión → correo manual
Acta → seguimiento.py + modelo por API → bandeja → aprobación → envío manual
Pregunta → Kimi CLI con web → respuesta con URL y cita → comprobación de la cita
Factura escaneada → modelo con visión → campos → comprobación de sumas por código
```

Marca en cada línea:

- **Dónde se bloquea si una pieza desaparece.** Si el flujo solo funciona con una herramienta, es una dependencia.
- **Dónde copias la misma información a mano.** Cada copia es un sitio donde se pierde una versión.
- **Dónde hay una comprobación que no depende del modelo** (código, persona, fuente abierta). Si no hay ninguna, el flujo descansa entero en la confianza.

Cómo sabes que ha ido bien: cada línea tiene al menos una marca, y sabes cuál se para si mañana pierdes una herramienta.

Paso 4 · Clasifica cada pieza

Asigna a cada herramienta una etiqueta:

1. **Prescindible:** puedes quitarla sin efecto.
2. **Sustituible:** existe otro procedimiento y lo has probado.
3. **Dependencia:** no tienes salida ensayada; necesita plan.
4. **Sin uso:** nadie la usó en la semana registrada. Confírmalo con quien la tenga antes de tocarla.

«Sustituible» exige haberlo probado. Que otra aplicación «también tenga IA» no cuenta. En el capítulo 03 verás por qué: la misma skill, en el mismo proyecto, no cargaba en OpenCode ni en Codex por un detalle de formato.

Cómo sabes que ha ido bien: ninguna herramienta está marcada «sustituible» sin una prueba detrás.

Paso 5 · Escribe las duplicidades como sospecha

Dos herramientas se repiten para una tarea cuando comparten entrada, salida y nivel de revisión, y las pruebas del capítulo 04 lo confirman. Hasta entonces, escríbelo como hipótesis:

```
Hipótesis: para «seguimiento de reunión», la herramienta A y la herramienta B
podrían ser intercambiables.
Prueba que lo decidiría: misma acta, mismo encargo, mismos criterios, tres repeticiones.
Qué haría que no lo fueran: una no carga la skill, una no exporta la tabla,
una necesita permisos que la otra no.
```

Antes de sospechar de la herramienta, sospecha del encargo. En la tarea del acta, la mayor mejora vino de reescribir el encargo (pasar de una línea ingenua a uno con reglas y esquema), no de cambiar de modelo: con el encargo bueno, los cinco modelos de nube pasaron todos los criterios.

Cómo sabes que ha ido bien: cada sospecha tiene escrita la prueba que la decidiría.

Paso 6 · Elige la primera prueba

Escoge una sola tarea para la primera prueba de sustitución. Que cumpla esto:

- Tiene entrada y salida claras (un acta y una tabla, no «ayudarme con el proyecto»).
- La puedes corregir con reglas objetivas, mejor si es con código.
- No tiene efectos hacia fuera: no envía, no borra, no paga.
- La haces a menudo, para que el resultado importe.

El seguimiento de reunión cumple las cuatro. Por eso recorre todo el libro, y por eso el paquete trae la skill `extraer-acciones-acta`, el acta ficticia y la receta `seguimiento.py`.

Cómo sabes que ha ido bien: tienes una tarea elegida y guardas el inventario con la fecha en el nombre del archivo. Lo volverás a abrir en el capítulo 10.

Un ejemplo real: la combinación que usamos en las pruebas

Te la enseñamos como ejemplo de inventario, no como recomendación de compra. No damos importes ni planes; sí el tipo de acceso de cada herramienta. Lo probamos en un PC con Windows 11 y una gráfica NVIDIA de 16 GB. Los tiempos no se trasladan a un portátil sin gráfica dedicada.

Herramienta	Tipo de acceso en las pruebas
Claude Code	Suscripción (modelos Claude Haiku 5.5 y Claude Sonnet 5.5)
Codex CLI	Cuenta gratuita de ChatGPT, modelo por defecto de esa cuenta
OpenCode	Modelos por API compatible con OpenAI, sin coste cuando lo probamos
Kimi CLI	Su modelo por defecto
Ollama	Local: qwen3.5:4b, qwen3.5:9b, gemma4:12b, qwen3.6:35b, embeddinggemma:300m

Por API usamos GPT-6 luna, DeepSeek V4.1 Flash y DeepSeek V4.1 Flash 1017, sin coste cuando lo probamos (compruébalo en la web oficial de cada proveedor). En la comparación del capítulo 04 participaron además Claude Opus y GPT-6 sol; este último, por una API de pago con saldo, la del caso del capítulo 10.

Lo útil no son las marcas, sino la mezcla: una suscripción, una cuenta gratuita, modelos gratuitos por API y modelos locales. Cada tipo de acceso tiene su riesgo. Lo gratuito puede cambiar de condiciones. La API depende de un saldo (el capítulo 10 tiene un caso de saldo que engañaba). Lo local depende de tu hardware.

Regla de simplificación

Quita pasos antes que herramientas. Si un documento pasa por tres aplicaciones para acabar en la misma tabla, quizá no necesitas sustituir ninguna: basta con quitar dos copias. Un stack pequeño es la consecuencia de un proceso claro.

Resumen

- [ ] Cada tarea tiene entrada, salida y persona usuaria.
- [ ] Las frecuencias no contadas dicen «sin medir».
- [ ] Cada herramienta tiene versión, cuenta y tipo de acceso apuntados, sin precios de memoria.
- [ ] Al menos una dependencia sin salida ensayada está señalada.
- [ ] Las duplicidades están escritas como hipótesis con su prueba.

- [ ] Hay una tarea elegida para la primera prueba, sin efectos hacia fuera.
- [ ] Antes de culpar a una herramienta, has revisado el encargo.