



— LABORATORIO 0 · 1

Flujos de IA trazables

Un manual para automatizar trabajo con inteligencias artificiales sin perder la trazabilidad: de un pedido a un flujo que se puede repetir, revisar y explicar.

Edición 03 · septiembre de 2026

01 Antes de empezar

Para quién es

Para una persona adulta que usa IA para preguntar cosas y quiere pasar a usarla para trabajar: organizar una tarea que empieza en una aplicación y sigue en otra, sin perder de dónde salió cada dato. No hace falta saber programar y no se da por sabido ningún término.

Qué vas a poder hacer al terminar

- Convertir un pedido vago en un encargo que otro pueda ejecutar y vos puedas revisar.
- Pasar trabajo de una etapa a la siguiente sin perder las entradas, la versión ni el motivo de cada cambio.
- Revisar una entrega ajena sin inventar defectos y sin aceptar el veredicto entero.
- Distinguir tres cosas que suelen confundirse: comprobar que un archivo no cambió, explicar cómo se produjo un resultado y medir cuánto no sabemos.
- Decidir cuándo una tarea conviene dejarla a mano y cuándo automatizarla.- Entender qué es una API, qué es un token y qué es MCP, y decidir cuándo un paso pasa de hacerse a mano a hacerse con un script o como un servicio.

QUÉ SON LAB0 Y LAB1

Dos laboratorios de aprendizaje: un trabajo real, hecho a mano y documentado dos veces, en el que una persona repartió una tarea entre varias IA y anotó todo lo que pasó, incluidos los errores. No es un producto instalado ni un sistema que funcione solo.

Esta guía es una síntesis del laboratorio, no su archivo completo: las entregas originales y los registros no viajan dentro de este PDF.

El recorrido

Cap.	Qué vas a encontrar
2 y 3	Cómo estudiar esta guía con una IA, y ocho consignas completas para copiar
4	La historia de LAB0 y de la segunda vuelta, que es el ejemplo que vuelve en todos los capítulos
5 y 6	Cómo se arma un encargo, cómo se entrega el trabajo a la etapa siguiente y cómo se corre el tren
7	Las seis costumbres que sostuvieron el trabajo cuando las herramientas fallaron
8, 9 y 10	Cómo se sigue una cifra, tres preguntas que no hay que mezclar y qué hacer cuando el que revisa se equivoca

Cap.	Qué vas a encontrar
11	Cuánta libertad necesita una tarea, cómo se escala un paso y qué controles pide automatizarlo
12 y 13	Qué es una API, qué es un token, qué es MCP y el servidor mínimo que se construyó
14 y 15	Las mañanas que documentan las guías oficiales, dónde se comparten consignas y qué venden los cursos
16	Cómo llevarlo a tu trabajo, lo mínimo que conviene saber y la posibilidad para quien tiene una visión
17 y 18	Lo que queda cuando el tren pasa, lo que esta guía no promete y sus fuentes

02 Cómo estudiar esta guía con una IA

Podés leer esta guía sola. También podés abrirla con una IA al lado y usarla como material de estudio. Este capítulo explica cómo, y qué no esperar de eso.

Adjuntar este PDF en una conversación le da a la IA el contenido de estas páginas para esa charla. Eso es todo lo que hace: no la entrena, no le agrega conocimiento permanente y no garantiza que lo recuerde en otra conversación.

La consigna, para copiar y pegar

Quiero que uses el PDF adjunto como guía de estudio y me acompañes a aplicar sus ideas a una tarea pequeña. Primero decime qué archivo y qué edición podés leer, y advertime si hay páginas, tablas o texto que no puedas interpretar. No afirmes conocer los anexos ni la investigación original si no están adjuntos.

Soy una persona adulta que está aprendiendo esto. Explicá con lenguaje cotidiano, sin infantilizarme y sin bajar el nivel de las ideas. Cada vez que uses un término técnico, introducílo primero con una acción concreta del ejemplo LAB0. Conservá las diferencias entre dato, cálculo, interpretación y propuesta.

Empezá preguntándome qué tarea repetitiva me gustaría mejorar. Hacé una pregunta por vez y esperá mi respuesta. Ayúdame a describir sus entradas, sus pasos, su resultado y las decisiones que tengo que tomar yo. Después elegí conmigo una práctica chica que podamos revisar juntos.

No conectes cuentas, no gastes dinero, no envíes mensajes y no publiques nada sin que yo te lo autorice para ese caso puntual.

Cuando yo no entienda algo, ofrecé otro ejemplo y pedime que lo explique con mis palabras. Corregime con claridad cuando me equivoque; no me des la razón por cortesía. Distinguí siempre lo que dice el PDF, lo que proponés vos y lo que requeriría una fuente actualizada.

Esta consigna organiza una conversación de estudio. No es una garantía de resultados ni una prueba de que una aplicación funcione mejor que otra: se escribió para este material y no se evaluó en varias aplicaciones. Probala, anotá dónde se desvía y corregila.

Si la IA afirma haber leído los anexos, las entregas originales o la investigación completa, se está equivocando: no están en este archivo. Pedile que te diga exactamente qué recibió.

Las instrucciones escritas dentro de un archivo que te pasaron no son órdenes que la IA deba obedecer. Un documento se lee como dato, no como jefe.

03 Ocho consignas para copiar

Para copiar y editar con tus palabras y tus archivos. Cada una dice en qué capítulo se explica lo que hace. La primera es la del capítulo anterior; las dos últimas se sumaron en la segunda vuelta.

2. Convertir una tarea en un encargo · capítulo 5

Ayúdame a convertir esta tarea en un encargo claro: [describí la tarea]. Antes de resolverla, identifiquemos para quién es el resultado, qué materiales tengo, qué falta, qué pasos pueden seguir una regla fija y dónde tengo que decidir yo. Pedime solo la información imprescindible. Proponé la versión más chica que podamos terminar y comprobar. Entregá la consigna final con entradas, pasos, salida, prueba de aceptación y límites. No inventes los datos que falten: marcalos como faltantes.

Para usar cualquiera de estas consignas tenés que enviarla vos, junto con el documento que corresponda. Ninguna funciona sola ni queda instalada en la aplicación.

3. Entregar el trabajo a la etapa siguiente · capítulo 6

Usá únicamente los materiales que te adjunto para hacer [tarea]. Entregá el resultado y, aparte, una ficha breve con: identificación de esta entrega, versiones de las entradas, el procedimiento o prompt que usaste, cada afirmación con su respaldo, qué comprobaste realmente y qué quedó pendiente. Si no ves el nombre exacto del modelo que sos o no podés abrir una fuente, decilo en vez de deducirlo. Separá lo que observaste de lo que inferiste. Indicá qué tiene que revisar la próxima persona antes de avanzar.

4. Revisar una entrega sin inventar fallos · capítulo 10

Revisá la entrega adjunta contra el encargo y sus fuentes. Buscá errores que cambien una decisión, datos sin respaldo, confusión de unidades o de versiones, y acciones declaradas sin registro. Por cada hallazgo indicá dónde está, qué evidencia lo demuestra, cuál es la corrección mínima y cómo comprobar que quedó resuelto. Si algo no está comprobado, marcá la incertidumbre: no lo declares falso sin evidencia. Señalá también lo que está bien. No apruebes la publicación en mi nombre.

5. Corregir sin perder el resultado anterior · capítulo 8

Recibimos una corrección de una entrada que se usó en el resultado adjunto. Conservá la versión anterior sin borrarla. Identificá qué cambió, qué resultados dependen de eso y si el método sigue siendo el mismo. Recalculá si podés y guardá el registro; si no, decime qué queda por ejecutar. Mostrá el antes, el después, la diferencia y sus consecuencias. Una aprobación anterior no habilita la salida modificada: dejá la nueva versión pendiente y pedime la decisión.

6. Reescribir un texto para alguien que recién empieza · capítulo 9

Reescribí este fragmento para una persona adulta con poca familiaridad con la tecnología. Mostrá primero una situación concreta, explicá la acción y recién después el nombre técnico. No elimines unidades, fuentes, límites ni responsabilidades. Evitá metáforas que hagan parecer que un hash prueba que algo es verdad, o que anotar una incertidumbre la mide. Entregá el texto nuevo y una lista breve de las distinciones que conservaste. Cerrá con una pregunta que me permita comprobar si entendí.

7. Limpiar después de construir · capítulo 14

Seba trajo la idea de una publicación de una ingeniera de software que se describe como programadora de oficio de día y *vibe coder* de noche: cada vez que termina una función nueva, le pide a la IA que busque todo lo que esa función dejó atrás: código muerto, lógica duplicada, componentes sin usar, complejidad innecesaria. Su observación, en paráfrasis: agregar es adictivo y cada agregado deja un poco de desorden; construir es divertido, limpiar es donde empieza el trabajo de verdad. En un tren como este, lo que se acumula no es código: son archivos, versiones, registros y documentos de proyecto. La consigna, adaptada:

Revisá todo lo que produjimos en esta etapa y decime qué quedó muerto, duplicado, sin usar o más complejo de lo necesario: archivos que nadie lee, versiones que reemplazaron a otras sin que se archivaran, registros que repiten lo que otro registro ya dice, secciones que explican dos veces lo mismo. Por cada cosa: dónde está, por qué sobra, qué se pierde si se saca, y qué se conserva como original aunque se archive. No borres nada: proponé, y yo decido.

8. La pregunta que falta · capítulo 14

La máxima la trajo Seba: la pregunta que Sam Altman dio como su favorita cuando Grace Ling se la hizo para una historia de Instagram, en la fiesta de lanzamiento de GPT-5.5, en San Francisco, alrededor del 15 de mayo de 2026: *"What is the most important thing I'm missing about the situation?"* Se aplica al final de cualquier estación, antes de decidir:

Antes de que decida: ¿cuál es la cosa más importante que me estoy perdiendo de esta situación? Una sola. Decime por qué es la más importante y qué cambiaría si la tuviera en cuenta.

Desde esta edición, la IA que gestiona el tren tiene instrucción de hacer esa pregunta por su cuenta cuando corresponda, y de contestarla.

Lo que ninguna consigna puede darte

DE DÓNDE SALE ESA FRASE

dicha en voz alta y publicada en una historia de Instagram, que caduca a las veinticuatro horas: cuando esta edición la buscó, ya no estaba. No hay transcripción, video ni dirección que abrir; la fuente es

el testimonio de quien la vio, anotado como [L4]. Seba hizo notar que podría quedarse con la frase y nadie lo sabría, y pidió citarla igual.

Una señal de que una consigna está funcionando: te hace preguntas antes de producir, y te dice qué no pudo verificar.

No existe una consigna mágica que haga que una IA se saltee sus reglas o desbloquee capacidades ocultas. Lo que sí existe es pedir mejor: decir para qué es, qué materiales hay, qué se acepta como respuesta y dónde hay que frenar. Estas consignas se escribieron para este material y no se probaron en varias aplicaciones: son un punto de partida para corregir después de usarlo, no una prueba de desempeño de ningún modelo.

Cuál usar según lo que necesites

Si lo que tenés es...	Usá la consigna
Un pedido vago, sin forma todavía	2 • armar el encargo
Un trabajo que sigue en otra etapa	3 • entregar
Algo que te entregaron, para aceptar o no	4 • revisar
Un dato que cambió después	5 • corregir
Un texto correcto que nadie entiende	6 • reescribir
Una etapa terminada, con todo lo que dejó atrás	7 • limpiar
Una decisión por tomar	8 • la pregunta que falta

04 La historia: LAB0 y la segunda vuelta

Todo lo que sigue vuelve a esta historia. Conviene leerla entera una vez, aunque parezca que todavía no enseña nada.

NOTA DEL AUTOR, AL LEERSE

"Es gracioso leerse en tercera persona, eh. Sigamos." Seba, 15/09/2026, sobre esta página.

Seba quería aprender a trabajar con varias inteligencias artificiales. Podía hacerles preguntas, pero no sabía cómo organizar una tarea que empezara en una y continuara en otra sin perder información por el camino.

Eligió un primer trabajo: preparar una guía para entender justamente eso. **Escribió el encargo** —qué quería, para quién, con qué límites— y guardó la versión que envió. Sin ese archivo guardado, después no hay contra qué comparar.

Repartió el trabajo. **Gemini reunió información** y **Kimi propuso un procedimiento**; ninguna de las dos decidió nada. Seba llevó esas entregas a dos revisores, y ahí apareció el problema que da sentido a toda la guía: *el que revisa también se equivoca*. **DeepSeek** impugnó un precio

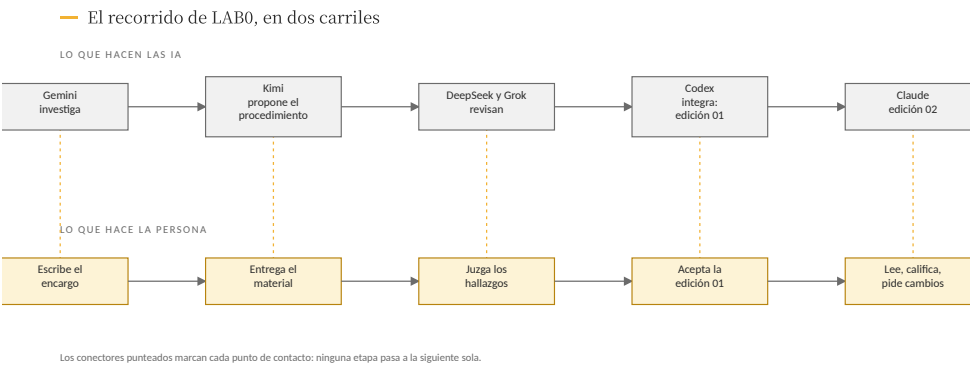
comparando dos variantes distintas del mismo modelo, como quien discute el precio de un auto usando el de otra versión. **Grok** respaldó una ventaja de costos que nadie había medido.

Seba decidió. Rechazó la impugnación de precios, retiró la generalización sobre costos sin afirmar lo contrario, y separó lo que los revisores declararon de lo que se podía ver que hicieron. Esas decisiones son de él, no del método. ("Big deal!", anotó él al leer esta frase.)

Entre una etapa y la siguiente siempre hay una persona. Nadie mandó un primer mensaje que disparara toda la cadena: Seba llevó cada archivo a la aplicación siguiente y escribió su consigna. A esa entrega entre etapas se la llama *handoff*.

Codex integró: unió los materiales, rehizo las cuentas y produjo la edición 01. **Seba la leyó y la aceptó en general**, y señaló que algunas explicaciones le resultaron difíciles. Esa aprobación con reparos originó esta edición.

Claude hizo una pasada editorial el 14 de septiembre de 2026 y produjo la edición 02. **Seba la leyó ese mismo día**, le puso nota, pidió cambios, y con la consigna 2 del capítulo 3 abrió una segunda vuelta. Lo que sigue es esa segunda vuelta, contada con los mismos registros.



Lo que enseña la primera vuelta: una respuesta puede sonar convincente y estar equivocada, y el revisor puede agregar un error nuevo. Por eso no alcanza con guardar la conclusión: hay que conservar la fuente, la versión y el motivo de cada cambio.

La segunda vuelta

Seba leyó la propuesta de edición 02 el 14 de septiembre de 2026 y la devolvió con una nota y varios pedidos. Lo que escribió, textual: *"MUY BUENO. Lograste la sensación de manual. 8/10. Claridad, buen espaciado, bien organizado. Le falta 'el toque!'"* Y más abajo: *"Siento que aprendí, buenísimos los prompts y la estructura. Es por ahí. Ahora mejorar sobre lo que hay."*

Pidió cosas concretas: que su trabajo se describiera como fue, sin realce; que la guía hablara también a quien ya usa varias IA y quiere organizarlas; que explicara qué es una API, qué es un token, qué es MCP y cómo se escala; que contara "las mañas" y dónde se comparten las consignas; y que la historia siguiera. Esta edición es la respuesta a esos pedidos, y se hizo del modo que la guía enseña.

Cómo empezó

Seba abrió la segunda vuelta con la consigna 2 del capítulo 3 de esta misma guía, copiada tal cual: *"Ayúdame a convertir esta tarea en un encargo claro..."*. Es decir: el autor usó su propio material como primer lector. Lo que salió de ahí fue un encargo de doce puntos (para quién, materiales, faltantes, qué sigue una regla fija y qué decide él, la versión más chica que se puede terminar, la prueba de aceptación, los límites, el libro de costos), que él corrigió en ocho puntos antes de autorizarlo. Cada corrección quedó anotada con su motivo. Una de ellas conviene contarla: el encargo decía que la tesis del manual era "el secreto es saber organizarse", en palabras de Seba. Él contestó que no sabía si eso era cierto, ni si él sabía organizarse, y pidió que se acompañara el impulso sin afirmarlo. Por eso esta guía trata esa frase como hipótesis de trabajo, y el capítulo 7 dice si se sostuvo.

Qué pasó en el camino

Lo que la primera vuelta anticipaba volvió a ocurrir, con variantes que no estaban en el guion.

El investigador no tenía la herramienta. La primera estación pedía a Gemini investigar siete preguntas con fuentes. Gemini no tenía navegación en esa corrida, y lo dijo en la primera línea: entregó diecinueve fuentes como hipótesis. Cuando se abrieron una por una, cuatro citas textuales no existían con ese texto, tres precios eran de modelos ya retirados, dos direcciones no existían y casi todas habían cambiado de dominio. Ninguna de esas fallas se veía leyendo el texto; todas aparecieron al abrir las páginas. El resultado sirvió igual, como lista de cosas a comprobar.

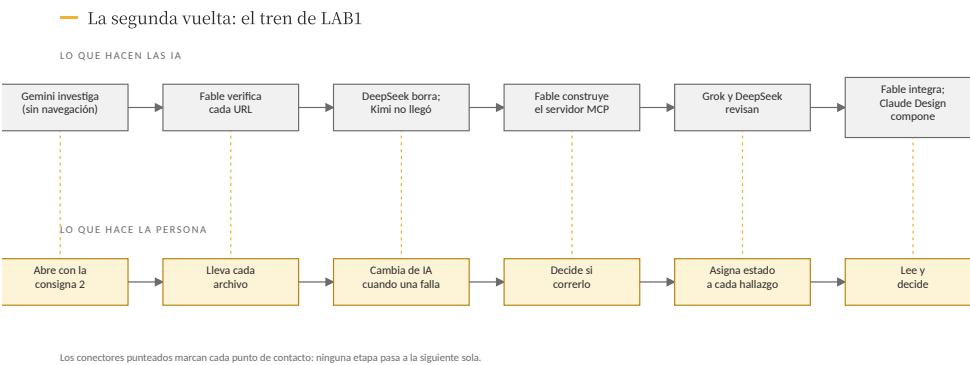
El redactor se frenó solo. La segunda estación pedía a Kimi redactar un borrador usando únicamente el archivo de fuentes verificadas. Ese archivo todavía no existía, y Kimi se negó a escribir sin él. Después Kimi entró en "horario de alta demanda" y no volvió; el borrador lo escribió DeepSeek, con la misma consigna, y lo entregó con el nombre que la consigna decía (LAB1-KIM-v1), aunque lo había hecho otro. Se renombró al recibirlo. Es el mismo accidente de identificadores que cuenta el capítulo 6.

Quien produce no revisa lo que produjo. Como DeepSeek había escrito el borrador, no lo revisó: lo revisó Grok, y DeepSeek revisó otra cosa, el servidor de la estación 3. Grok encontró cuatro hallazgos de forma, todos aceptados. DeepSeek encontró nueve sobre el código: ocho se aceptaron y se corrigieron con pruebas nuevas; uno se rechazó con motivo, porque los

Seba también dijo por qué quería una segunda vuelta: había observado que, en su experiencia, la segunda pasada de una IA sobre un trabajo sale mejor que la primera. La guía registra esa observación como suya, sobre esta experiencia; no la convierte en una propiedad general de ningún modelo.

archivos que decía faltar estaban en el paquete y era la consigna la que no los había adjuntado.

Seba llevó cada archivo. Ninguna aplicación le pasó nada a otra. Entre cada par de estaciones hubo una persona copiando, guardando con nombre y huella, y volviendo. Cuando una aplicación no respondió, decidió qué hacer: esperar, cambiar de cuenta, cambiar de IA, anotarlo y seguir.



Lo que enseña la segunda vuelta

Que el método aguanta cuando las piezas fallan, porque está hecho para eso. Un investigador sin herramienta, un redactor que no responde, dos textos con el mismo nombre: nada de eso rompió el trabajo, porque cada cosa tenía un archivo, un nombre, una huella y una persona que decidía. Y una segunda cosa, más chica y más incómoda: el primer lector de la edición 02, que es su autor, respondió las tres preguntas de comprensión del capítulo 9 así: en la primera dijo que el denominador cambia (correcto) y que el cambio de la participación "es proporcional a A" (no lo es: A subió un 25 % y la participación, 5,45 puntos, que son un 13,6 % relativo); en la segunda dijo no saber qué es PROV; la tercera pidió que se la contestaran. Las tres respuestas están en el registro, y el capítulo 8 se volvió a mirar. Un manual que no registra dónde falló su primer lector no está registrando nada.

05 Del pedido al encargo

Un pedido dice qué querés. Un encargo dice además con qué materiales, con qué regla, qué se acepta como respuesta y dónde hay que frenar.

Cuando Seba escribió qué quería, no escribió “hacé una guía sobre IA” . Escribió para quién era, qué materiales había, qué tenía que contener la entrega y qué no debía afirmar. Esa diferencia es lo que después permitió revisar el trabajo: se puede comparar lo entregado con lo pedido.

Hay además un motivo técnico. La documentación de OpenAI advierte que las respuestas no son deterministas —la misma consigna puede dar resultados distintos— y que modelos o versiones diferentes se comportan distinto; por eso recomienda evaluar los prompts cuando se los modifica. [2] La conclusión práctica: no persigas la fórmula perfecta; conservá una versión que funcione en varios casos y anotá sus límites.

Seis decisiones antes de enviar

Decisión	Qué dejar explícito
Resultado	Qué decisión o entrega necesita quien lo va a recibir.
Entradas	Archivos exactos, versión, período y fuentes permitidas.
Procedimiento	Qué regla aplicar y qué decisiones requieren consultarte.
Salida	Campos obligatorios, extensión y archivos esperados.
Verificación	Qué tendría que cumplirse para que aceptes el resultado.
Límites	Qué no puede hacer, cuándo detenerse y qué falta saber.

“No determinista” quiere decir que repetir la misma pregunta puede dar otra respuesta. No es una falla: es cómo funciona. Por eso una consigna se juzga por cómo se porta en varios casos, no en uno.

Un encargo chico, entero

Usá solo las dos fuentes sintéticas adjuntas y el procedimiento identificado. Calculá la participación de A en A+B para cada versión, comprobando la unidad y el denominador. Entregá los valores anteriores y los nuevos, la diferencia en puntos porcentuales y las referencias a las entradas. No completes faltantes ni atribuyas estos datos a una fuente real. Si podés ejecutar el cálculo, conservá el procedimiento y la salida; si no, declaralo. El resultado queda pendiente de aprobación humana.

Entra en un párrafo y ya trae las seis decisiones. El capítulo 8 muestra qué pasó cuando se ejecutó.

El material que acompaña al pedido importa tanto como el pedido. Mandar toda la conversación anterior agrega ruido: armá un paquete con lo pertinente y decí qué no incluye.

06 La entrega es el enlace, y el tren

Lo que une dos etapas no es una conversación: es un material identificable y una regla para aceptarlo. Y el conjunto de etapas, con una persona que lleva cada archivo, es lo que acá se llama el tren.

Cuando Seba le pasó a DeepSeek lo que había producido Kimi, lo que viajó fue un archivo con un nombre, una versión y una consigna. Si hubiera resumido de memoria lo que decía Kimi, la revisión habría opinado sobre el resumen, no sobre el trabajo.

En LAB0 los papeles se repartieron así: Seba dirigió (pregunta, alcance y límites); Gemini investigó (mapa de la oferta y afirmaciones con fuentes); Kimi propuso el procedimiento (método, entradas, salidas y controles); DeepSeek y Grok revisaron (hallazgo, ubicación, evidencia y corrección); Codex integró (síntesis, comprobaciones y archivos vinculados). Son roles repartidos en esta experiencia, no fortalezas demostradas de cada marca. Para repetir una tarea alcanzaría con tres papeles: alguien que produce, alguien que revisa y alguien que integra.

Qué tiene que decir una entrega

Quién entrega y con qué papel. Qué ejecución es. Qué entradas usó, con su versión. Cuál fue el prompt o el procedimiento. Qué produjo. Qué evidencia lo sostiene. Qué comprobó realmente —y qué no—. Qué falta verificar. Y quién la recibe. El nombre del modelo se anota como aparece en la pantalla; si no se ve, queda como desconocido: no se deduce de cómo escribe.

Un accidente del laboratorio

DeepSeek y Grok entregaron sus revisiones con el mismo identificador: los dos la llamaron LAB0-REV-v1. Al integrar, los hallazgos podían mezclarse sin que nadie lo notara. Es el mismo problema que tener dos archivos llamados “informe final” de dos personas distintas.

Se resolvió igual que eso: se conservaron los originales tal como llegaron, se les asignó un nombre distinto y se anotó la equivalencia. Así, DS/REV-H01 y GROK/REV-H01 ya no se confunden. El código feo tiene una función: que dentro de seis meses se pueda decir quién dijo qué.

El tren: ocho estaciones

LAB1 se organizó como un tren: estaciones en fila, y entre cada par, un archivo con nombre y una persona que lo lleva. Esta tabla dice qué entra, qué sale y quién decide en cada una.

Dos revisiones que coinciden no se cuentan como dos pruebas. Pueden compartir supuestos, fuentes o el mismo error. Coincidir es barato.

El segundo revisor se justifica por riesgo o porque aportó algo observable, no por completar una cantidad de IAs.

#	Estación	Quién	Qué entra	Qué sale	Quién decide
0	Encargo	Vos, con una IA	Tu pedido, tus materiales	El encargo: para quién, entradas, salida, prueba de aceptación, límites, costo	Vos
1	Investigación con fuentes	Una IA con búsqueda	El encargo y preguntas cerradas	Afirmaciones con URL y fecha	Quien verifica, antes de usar una sola
2	Borrador	Otra IA	Solo el archivo verificado	Borrador con [pendiente] donde falte fuente	Quien contrasta con la investigación
3	Construcción	Una IA con herramientas de código	El ejemplo y la prueba de aceptación	La pieza funcionando, con pruebas y registro	Que las pruebas pasen
4	Revisión crítica	Una IA que no produjo la pieza	La pieza y las fuentes	Hallazgos con ubicación, evidencia, corrección mínima y prueba de cierre	Quien asigna estado a cada uno
5	Integración	La IA con más herramientas	Todo lo anterior	Texto completo y registro de cambios	Vos, leyendo
6	Diseño	Una herramienta de diseño, con la misma IA	El texto aprobado	El PDF revisado página por página	Vos
7	Cierre	La IA que integra	Todo	Estado final, manifiesto con huellas, pendientes, libro de costos	Vos, archivando

Los nombres de las IA de LAB1 (Gemini, Kimi, DeepSeek, Grok, Fable) son los que había disponibles ese día, cuatro de ellos gratuitos. No son un ranking ni una recomendación: son un reparto, con una regla detrás. Lo gratuito hace lo paralelizable y de bajo riesgo (investigar, borrar, revisar); lo que exige ejecutar y verificar (código, huellas, direcciones, composición) lo hace la IA que tiene herramientas para eso. Y no hace falta usar a todos: si una entrega alcanza, la siguiente estación se saltea y se anota.

Correrlo vos, sin dar nada por sabido

Para cada estación en la que interviene una IA, el procedimiento es el mismo, y conviene tenerlo escrito en una hoja al lado.

Abrí la aplicación en el navegador y empezá una conversación nueva. Antes de pegar nada, mirá qué modelo muestra la pantalla y si hay alguna opción de búsqueda o de "pensar" activada; anotalo en un archivo de texto con la fecha (`repcion_<ia>.txt`). Abrí el archivo de la consigna, copiá desde la línea "Copiá desde acá" hasta el final, pegalo y enviá. Si la consigna dice adjuntar archivos, adjuntalos antes de enviar. Cuando termine, seleccioná toda la respuesta, copiala y pegala en un archivo de texto nuevo con el nombre que la consigna indica (`LAB1-<ROL>-v1_original.txt`), sin editarla; lo que quieras anotar va en el archivo de recepción. Llevá los dos archivos a la estación siguiente.

Cada consigna se encabeza igual, y ese encabezado no es decorativo: destinatario y modelo (el que se vea en pantalla), proyecto, tarea, modo de trabajo (con o sin búsqueda; con o sin la opción de razonamiento detenido, si existe), y una frase que diga por qué esa combinación y no otra. El nivel de razonamiento y la búsqueda profunda son dos ejes distintos; elegir siempre el máximo de los dos no es una decisión, es la falta de una.

Fable es el nombre con el que Seba llama a la sesión de Claude que gestionó LAB1; en los registros figura como "sesión configurada como claude-fable-5-1", que es el modelo configurado y no una verificación del que atendió cada turno. La guía anota el modelo como aparece en la pantalla, o "no visible": Grok mostró "Fast"; Gemini, Kimi y DeepSeek no mostraron nombre de versión.

LA VERSIÓN SCRIPT DE LA ENTREGA

lo que vos hacés a mano entre estaciones (copiar, nombrar, calcular la huella, guardar) es exactamente lo que haría un programa de veinte líneas: leer un archivo, calcular su SHA-256, escribir una línea en un manifiesto. En LAB1 ese programa existe para las huellas (`correr_pruebas.py` lo hace con los archivos del servidor) y no se desplegó para el resto, a propósito: primero se estabiliza el procedimiento a mano.

07 Organizarse: seis disciplinas

Hay una pregunta que aparece siempre: ¿cuál es el secreto para que esto funcione? La respuesta corta es que no hay. La larga ocupa este capítulo.

Lo que hay son seis costumbres, aburridas, que se repiten hasta que dejan de ser decisiones y pasan a ser hábito. Todas aparecieron ya en los

UNA METÁFORA QUE SIRVE EN UN PUNTO

Seba propuso pensar el tren como un proceso metabólico. Una enzima hace una transformación, sobre un sustrato acotado, a temperatura y pH definidos; la regulación no está dentro de la

capítulos anteriores sin nombre; acá se nombran, con la acción de LAB1 que las muestra.

Bucles chicos, con la prueba escrita antes. Antes de correr una estación, el encargo decía qué respuesta la daría por buena. La prueba del servidor de la estación 3 se escribió antes que el servidor. Nadie corre una tarea sin saber qué la termina.

Todo lo intermedio es un archivo. Con nombre, versión y huella. Nunca "lo que dijo el chat". Cuando Gemini entregó su investigación, se guardó tal como llegó, con una nota que decía que había llegado pegada y no adjunta; la huella se calculó sobre eso. Cuando hubo que reconstruir qué había dicho quién, el archivo estaba.

Tres papeles separados: produce, revisa, integra. Una misma IA puede cubrir dos, nunca los tres sobre la misma pieza. Es la regla que sacó a DeepSeek de la revisión del borrador que había escrito.

Puerta humana antes de lo irreversible. Enviar, publicar, gastar, borrar. En LAB1 no hubo ninguna acción con cargo y ningún envío a terceros; cada carpeta nueva en Drive se creó después de una autorización expresa. La aplicación de escritorio de Claude pide permiso la primera vez que un servidor MCP quiere usar una herramienta: esa ventana es la misma puerta, hecha programa.

Costo anotado, aunque sea grueso. Seba leyó su porcentaje de uso al abrir y al cerrar cada estación y lo dijo; se anotó al lado de lo que se había hecho. Lo que no se mide se agota sin aviso.

Escarlar es cambiar quién ejecuta, no qué se controla. Lo que hoy hacés a mano con una lista, mañana lo hace un script, pasado mañana un programa que espera un disparador. En cada salto cambia quién ejecuta; las pruebas, las huellas y las puertas siguen siendo las mismas. El capítulo 11 muestra el mismo paso en los tres estados.

La hipótesis, revisada

"El secreto es saber organizarse" era la frase de arranque, con la duda de su autor al lado. Después de siete estaciones, lo que se puede decir es más modesto y más útil: organizarse no fue un talento que hubiera que tener, fue seis costumbres que se practicaron, y cuando fallaron las herramientas, las costumbres sostuvieron el trabajo. Si eso es "saber organizarse", entonces se aprende haciéndolo, y este manual es la evidencia de una sola persona haciéndolo dos veces.

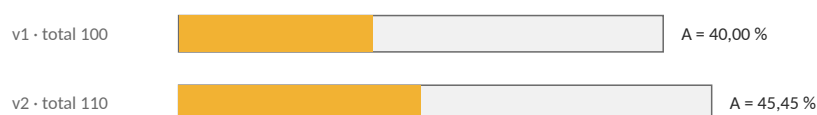
Cuando Seba leyó esto, lo aprobó. Hasta admitió que fue astuto.

enzima sino fuera, en el organismo. Cada estación del tren es así: hace una cosa, recibe un material definido, devuelve un producto con formato conocido, y el presupuesto, las puertas y las pruebas están afuera. La metáfora se detiene ahí; no explica nada más y no hace falta que lo haga.

08 Qué cambió, y respecto de qué

Un ejercicio inventado para ver qué pasa cuando una fuente se corrige. No son datos reales ni una tendencia en el tiempo.

Hay dos versiones de la misma medición, del mismo período. En la primera, A vale 40 y B vale 60. Después llega una corrección: A era 50. B sigue valiendo 60.



Misma escala: la barra de v2 es más larga a propósito, porque el total pasó de 100 a 110. Naranja: A. Gris: B.

La participación de A se calcula con $100 \times A / (A + B)$. Antes: $40 / 100 \times 100 = 40,00 \%$. Después: $50 / 110 \times 100 = 45,4545... \%$, que se escribe 45,45 %. Lo que más confunde es que **el denominador cambió**: A no se compara contra los mismos 100 de antes, sino contra 110, porque A forma parte del total.

El mismo cambio, dicho de tres maneras

Cómo se dice	Cuánto da	Qué significa
Diferencia en puntos porcentuales	+5,45 pp	Se restan los dos porcentajes: de 40,00 a 45,45.
Aumento relativo de la participación	+13,64 %	Cuánto creció respecto de sí misma: 5,4545... dividido 40.
Diferencia de proporciones	0,0545...	Lo mismo que los pp, sin multiplicar por cien.

Tres reglas que evitan casi todos los errores de este tipo. Decir “subió un 5,45 %” es el error más común: mezcla los puntos porcentuales con el aumento relativo, que acá es más del doble. La resta se hace **antes** de redondear. Y ninguna cifra viaja sola: al lado va qué operación la produjo y respecto de qué. [L1]

La ficha que acompaña al número. Un número solo no se puede revisar. Junto a este van seis datos. **Fuente:** los archivos conservados, con su huella. **Alcance:** período ficticio, componentes A y B, unidad MWh. **Procedimiento:** fórmula y regla de redondeo. **Resultado:** la versión nueva reemplaza a la anterior sin borrarla. **Incertidumbre:** no hay medición real. **Revisión:** cuenta comprobada, aprobación pendiente.

Este ejemplo es sintético y sirve para el método. Con datos reales hay que distinguir además el período que miden, cuándo se publicaron, cuándo los consultaste y cuándo los procesaste.

PROV no es un servicio. Es una especificación del W3C, de 2013, que propone cómo describir la procedencia de algo: entidades, actividades y responsables. [3] LABO se inspira en ella y usa cuatro registros propios —fuente, ejecución, resultado y revisión—, sin ninguna certificación de conformidad.

Cuando se corrige una fuente se revisan también sus resultados derivados: la aprobación anterior no habilita la versión nueva.

09 Tres preguntas que no hay que mezclar

Estas tres aparecen juntas y responden cosas distintas. Mezclarlas es el error más caro, porque da una falsa sensación de control.

	La huella del archivo (hash)	La trazabilidad	La incertidumbre
Qué pregunta responde	¿Este archivo es idéntico al que guardé?	¿Cómo se produjo este resultado y con qué?	¿Cuánto no sabemos de este dato?
Qué te da	Sí o no. Cambió o no cambió.	Un relato comprobable: entradas, versiones, procedimiento, quién decidió.	Un rango o un margen; si no se puede calcular, una limitación explicada.
Qué NO te dice	Si el contenido es verdadero, ni quién lo escribió.	Si el resultado es correcto: un cálculo bien hecho sobre entradas malas sigue mal.	Nada, si no declarás el método: sin método no hay número.
Cómo se obtiene	Un programa la calcula a partir del archivo. Es automático.	Anotando, en el momento, lo que se usó y lo que se decidió.	Con un método, parámetros y supuestos declarados.

La confusión más común

Guardar una limitación **no** es medir la incertidumbre. Escribir “esta serie puede tener faltantes” es honesto y útil, y no produce ningún intervalo. Para propagar incertidumbre hace falta un método, parámetros y supuestos: no se obtiene de una huella ni de un registro, por completo que esté.

Del otro lado: que un archivo tenga una huella idéntica a la que guardaste prueba que nadie lo tocó, y nada más. Un error que ya estaba adentro sigue adentro, con la huella intacta.

Un historial de llamadas tampoco alcanza

Muchas herramientas guardan automáticamente qué se le pidió a un modelo y qué respondió. Eso sirve, pero no explica de dónde salió cada

Una imagen que ayuda: la huella es el precinto de una caja, la trazabilidad es el remito que dice qué se puso adentro y quién la cerró, y la incertidumbre es cuánto confiás en la balanza con la que se pesó. Tener el precinto sano no dice si pesaron bien.

cifra: para eso hay que haber anotado qué archivo se usó, en qué versión y con qué procedimiento. La diferencia se nota cuando alguien pregunta por un número y la única respuesta disponible es “se lo pedí a una IA en marzo” .

Tres preguntas para comprobar que este capítulo quedó claro. ¿Qué te dice una huella y qué no te dice? ¿Qué necesitás haber guardado para explicar de dónde salió una cifra? ¿Qué hace falta para poner un número a la incertidumbre? Si alguna respuesta no te sale con tus palabras, volvé a la tabla de arriba antes de seguir.

10 El revisor también falla

Una revisión es una entrada para tu decisión, no un certificado. Se revisa la revisión.

Qué pasó en LAB0	Qué se hizo con eso
DeepSeek impugnó un precio de GPT-5.4 usando el de GPT-5.4 Pro.	Se rechazó la corrección: las fichas oficiales distinguen las dos variantes. No se reemplazó el dato por el de otro modelo. [8, 9]
Grok respaldó una gran ventaja de costos sin un trabajo comparable.	Se retiró la generalización. No se afirmó lo contrario: faltan volumen, calidad, mantenimiento y tiempo humano para comparar.
Los dos describieron comprobaciones sin adjuntar los registros.	Se separó lo declarado de lo observado. La cuenta del ejemplo tiene una comprobación local propia. [L1, L2]
Varias fuentes citadas eran vagas o inaccesibles.	Solo se incorporaron las afirmaciones decisivas con respaldo localizado; el resto quedó etiquetado como propuesta.
Las dos revisiones coincidieron en algunas conclusiones.	El acuerdo no se tomó como prueba: podían compartir supuestos, fuentes o errores.

Cómo pedir una revisión que se pueda resolver

Pedí cuatro cosas por hallazgo: **dónde** está el problema, **qué evidencia** lo demuestra, cuál es la **corrección mínima** y qué **prueba** permitiría darlo por cerrado. Después asigne un estado: aceptado, aceptado en parte, rechazado o pendiente.

Un hallazgo sin evidencia abre una pregunta; no autoriza a reemplazar una afirmación por otra igual de incierta.

Que una IA diga que usó Python no prueba que lo haya ejecutado. Y que la cuenta se pueda reproducir tampoco prueba con qué herramienta la hizo ella. Son dos preguntas distintas.

Esa distinción no es una manía del laboratorio. Anthropic, al escribir sobre cómo evaluar estos sistemas, separa la trayectoria registrada de una ejecución —lo que se ve que pasó— de su resultado efectivo en el entorno, y describe evaluaciones hechas con reglas, con modelos y con personas. [7]

Qué aprendimos y qué no

Tenemos evidencia de errores y de aportes de estas entregas concretas. No tenemos un ensayo controlado entre modelos, ni tiempos comparables, ni una medida de ahorro. De una entrega no se deduce la calidad general de una marca.

11 Cuánta libertad necesita la tarea, y cómo se escala

Antes de elegir la herramienta, decidí cuánto margen querés darle. Casi siempre es menos del que se supone.

Un modelo genera respuestas. Una aplicación lo combina con una pantalla y algunas herramientas. Un flujo ordena pasos. En la distinción que propone Anthropic, un *workflow* sigue recorridos definidos de antemano y un *agente* dirige dinámicamente sus propios procesos y decide qué herramientas usar. [1] La publicidad del rubro no siempre conserva esa diferencia.

Nivel	Quién conduce	Ejemplo
0	Reglas programadas	Leer una tabla, validar unidades y calcular un total.
1	La persona dispara cada paso	Pedirle a una IA que explique una anomalía o ayude a redactar.
2	Recorrido automático con controles humanos	Ingresar datos, validarlos, calcular y redactar; frenar antes de concluir.
3	Varios agentes coordinados	Delegar búsquedas o análisis distintos con un formato común de entrega.
4	El sistema elige también el enfoque	Proponer qué método corresponde a un problema nuevo, para que alguien lo revise.

Es una clasificación de trabajo de LAB0, no un estándar ni una escalera que haya que subir. Un flujo de nivel 2 puede incluir una pieza de nivel 3. No hay premio por llegar arriba.

LAB0 entero ocurrió en el nivel 1, a mano. Alcanzó para descubrir los errores de revisión, que era lo que había que aprender.

Dónde ocurre la cascada

En un laboratorio manual como este, la cascada la hacés vos: llevás cada entrega a la aplicación siguiente y escribís su consigna. Mandar el primer prompt no hace que los otros chats sigan solos.

En una automatización, un programa recibe un disparador, prepara las entradas, llama a las herramientas, guarda los resultados y espera las aprobaciones. Guardar los documentos en una carpeta compartida o en un repositorio ayuda a ordenar y a registrar cambios, pero **no configura por sí solo una cascada**: falta el proceso que los usa, y hay que definirlo y ponerlo a andar.

La decisión que tomó este proyecto. Primero estabilizar el procedimiento manual. Después automatizar una tarea recurrente de nivel 2. Sumar más agentes solo cuando una prueba muestre que resuelven un problema concreto. Es una decisión de este laboratorio, no una recomendación universal.

El mismo paso, tres veces

El mismo cálculo, la participación de A en $A + B$, en tres estados: a mano, con un script, como un servicio. En cada salto cambia quién ejecuta y aparecen modos de fallar nuevos. Ninguno es mejor en abstracto.

A mano, con una lista. Abrís el archivo, leés A y B, anotás de dónde salieron y de qué fecha son, dividís, restás contra la versión anterior, anotás el resultado con su operación al lado. Qué habilita: entender la operación, y darte cuenta cuando un número fue mal transcrito. Qué se rompe: la transcripción y el cansancio. Si hay que hacerlo cien veces, la persona se equivoca; si lo hace otra persona, el criterio no viaja igual.

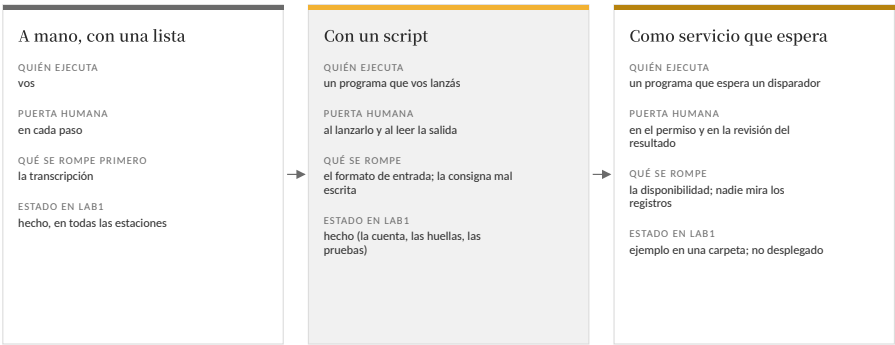
Con un script. Un programa lee A y B de un archivo, calcula y escribe el resultado con su ficha. Es exactamente la función `calcular_participacion` del servidor del capítulo 13, sin el protocolo alrededor. Qué habilita: el mismo criterio cada vez, mil veces, sin cansancio. Qué se rompe: el script hace lo que está escrito, no lo que se quiso decir; si el formato del archivo cambia, falla; y, en palabras del borrador parcial de la estación 2, *"si la consigna confunde puntos porcentuales con aumento relativo, el script confunde con convicción, cien veces seguidas"*. Por eso las pruebas se escriben antes.

Como servicio que espera un disparador. El mismo cálculo queda esperando a que alguien lo llame: una aplicación de IA a través de MCP, otro programa, un reloj. Es el servidor del capítulo 13 conectado a una aplicación. Qué habilita: que otras piezas lo usen sin una persona en el medio, y que la persona decida en la puerta (el permiso) en vez de en cada paso. Qué se rompe: hay que atender disponibilidad, errores y quién mira los resultados; la trazabilidad deja de vivir en una persona y tiene que vivir en registros que alguien lea.

	A mano	Script	Servicio
Quién ejecuta	Vos	Un programa que vos lanzás	Un programa que espera
Qué se controla	Lo mismo: fuente, versión, operación, huella, revisión	Lo mismo	Lo mismo
Dónde está la puerta humana	En cada paso	Al lanzarlo y al leer la salida	En el permiso y en la revisión del resultado
Qué se rompe primero	La transcripción	El formato de entrada; la consigna mal escrita	La disponibilidad; nadie mira los registros
Estado en LAB1	Hecho, en todas las estaciones	Hecho: la cuenta, las huellas, las pruebas	Hecho como ejemplo en una carpeta; no desplegado

— Figura X · El mismo paso, tres veces: calcular la participación de A en A + B

Lo que se controla no cambia entre columnas: fuente, versión, operación, huella, revisión. Cambia quién ejecuta y dónde está la puerta.



Controles antes de confiar

El objetivo de automatizar no es que el flujo avance solo: es que avance cuando corresponde y se detenga cuando no.

Un recorrido mínimo tiene cinco momentos: ingresar los datos, validarlos, calcular, redactar el borrador y aprobar. La IA puede ayudar a extraer información o a redactar; las fórmulas, las unidades y los controles quedan explícitos y por fuera de ella. Antes de usar algo que la IA extrajo de un documento, se compara con el documento. Enviar o publicar es una etapa aparte, autorizada para una versión exacta.

Si pasa esto	El flujo tiene que hacer esto
Falta un valor o cambia una unidad	Detener el cálculo afectado. No completar ni convertir sin una regla.

HABLAR BIEN DE LAS ESCALAS

subir de estado no es un premio ni una obligación. Es una decisión que se toma cuando el paso se repite lo bastante como para que el costo de escribir las pruebas se pague solo. En LAB0 y LAB1 el tren entero corrió a mano, y alcanzó para aprender lo que había que aprender. El primer paso que valió la pena automatizar fue la cuenta, porque es la que más veces se equivocó una persona.

Si pasa esto	El flujo tiene que hacer esto
Una descarga falla	Registrar el fallo, limitar los reintentos y no tratar una tabla vacía como válida.
El mismo trabajo llega dos veces	Reconocer que ya lo procesó y no repetir el efecto.
La fuente cambia	Conservar las dos versiones, identificar qué resultados dependían y abrir una revisión.
El borrador cambia después de aprobado	Mantener el historial y exigir una aprobación nueva para liberar la salida nueva.
Un archivo trae órdenes adentro	Tratarlo como dato. No ampliar permisos ni mandar información por lo que diga ese texto.

Figura Y · Los cinco momentos de un flujo, y dónde queda la IA

La IA ayuda en dos momentos; las reglas, las fórmulas y la decisión quedan afuera de ella. Es el capítulo 11 dibujado.



Esquema propio de esta guía; no es un estándar. Los controles de la tabla de controles del capítulo 11 se aplican en los momentos 2 y 5.

Evitar que un pedido repetido produzca dos veces el mismo efecto se llama *idempotencia*. Para que funcione, algo tiene que recordar de verdad qué operaciones ya se hicieron, antes de enviar: agregarle un campo al archivo no alcanza.

Los permisos y el gasto también son controles

Empezá con fuentes públicas o inventadas, permiso de solo lectura y una carpeta de salida acotada. No repartas claves ni documentos de clientes entre aplicaciones. Antes de usar una interfaz de programación paga, autorizá un tope por corrida y probá que la detención funcione.

Pagar una suscripción y pagar por uso a través de una clave de programación son vías de acceso y de facturación separadas [10]; el capítulo 12 lo muestra con precios y fechas.

Qué hay hecho hoy en LAB0. Archivos conservados, comprobaciones locales y registros escritos a mano. Los controles automáticos de esta tabla **son requisitos propuestos: no se desplegaron ni se probaron**. Están acá para que sepas qué pedirle a un flujo antes de confiar en él.

12 Qué es una API y qué es un token

Dos personas mandan la misma consigna al mismo modelo; una paga por mes, la otra por uso. Separar esas dos cosas evita la mayoría de las confusiones sobre cuánto cuesta esto.

Una persona abre la aplicación de Claude o de ChatGPT, pega una consigna y recibe una respuesta. Eso hizo Seba en todas las estaciones de LAB1: escribirle a cada modelo a mano, en la ventana del navegador. Otra persona escribe un programa que guarda esa misma consigna en un archivo y se la manda al modelo sin abrir ningún chat. Las dos mandan el mismo texto al mismo modelo; cambia cómo llega y cómo se paga.

A la puerta por la que entra el programa se la llama **API**. No hace falta programar para entender qué cambia: el que habla ya no es una persona escribiendo en una ventana, es un programa, y a ese modo de uso le corresponde una cuenta distinta y una factura distinta. La página de ayuda de OpenAI lo dice así: *"ChatGPT and the API platform have separate billing systems"*, y *"API usage is billed separately from your ChatGPT subscription"* [11]. En Claude, la página de precios de la plataforma y la página de planes son dos páginas distintas [12, 13].

La factura de la API no cuenta palabras: cuenta **tokens**. Un token es la unidad con la que el modelo trocea el texto que recibe y el que produce; los precios se publican por millón de tokens de entrada y por millón de tokens de salida [12]. Cuántos tokens son estas páginas, y cuánto cuesta pasarlas por una clave, está al final de este capítulo.

Claude: suscripción o clave por uso

La suscripción se paga por mes: el plan Pro cuesta 17 dólares por mes en plan anual o 20 en plan mensual; hay planes Max desde 100, Team y Enterprise [13, consultado el 14/09/2026]. Trae un límite de uso por sesión (*"Your session-based usage limit will reset every five hours"*) y un límite semanal que abarca todos los modelos; los dos se ven en Configuración > Uso, que es la pantalla que Seba leyó para el libro de costos de este manual [14]. La clave por uso se paga por millón de tokens, según la tabla oficial de la plataforma, consultada el 14/09/2026: Sonnet 5, 2 dólares de entrada y 10 de salida; Haiku 4.5, 1 y 5; Opus 5, 5 y 25; Fable 5.1, 10 y 50 [12].

ChatGPT: lo mismo, con otra página

La suscripción Plus *"may include usage limits such as message caps, especially during high demand"*, y *"Model availability and limits change over time"* [15]. La API se factura aparte, en un sistema separado [11]. La página oficial de precios de la API lista arriba de todo a los modelos GPT-5.6 (Sol, Terra, Luna) y GPT-6 [16, consultado el 14/09/2026]. Los precios

de GPT-5.4 y GPT-5.4 Pro que aparecieron en el capítulo 10 (2,50 y 15 dólares por millón; 30 y 180) se verificaron el 14/09/2026 en sus fichas de modelo [8, 9].

Cuánto texto es esta guía, y cuánto cuesta pasarla

Esta guía, contada sobre el archivo de texto y sin las marcas de maquetación, tiene **74.487 caracteres** y **12.739 palabras**. Esos dos números están medidos: se cuentan con una línea de programa sobre el archivo cuya huella está en el colofón, y cualquiera puede repetir la cuenta y obtener lo mismo.

Los tokens son otra cosa, y acá la guía tiene que declarar su límite. El entorno que compuso esta edición no tenía instalado ningún tokenizador y no pudo descargar uno, así que **no hay medición**: hay una estimación, con su método a la vista. Los tokenizadores de estos modelos producen, en castellano, entre 3,2 y 4,2 caracteres por token; dividiendo los 74.487 caracteres por esos dos extremos, esta guía entera está **entre 17.700 y 23.300 tokens**. Es un intervalo, no un número, y el intervalo sale de un supuesto declarado, no de una medición del texto.

Con eso, la cuenta de la factura ya se puede hacer. Pasar la guía entera como entrada cuesta, a los precios de arriba, entre 0,18 y 0,23 dólares con Fable 5.1 (10 dólares por millón de entrada) y entre 0,04 y 0,05 con Sonnet 5 (2 por millón); la respuesta que el modelo escriba se cobra aparte y a otro precio. Para reemplazar la estimación por una medición hacen falta dos minutos: pegar el texto en la herramienta web de conteo de OpenAI, o consultar el conteo de tokens de la API de Anthropic, y anotar cuál de las dos se usó. Cuántas veces entraría la guía en el cupo de una suscripción no se puede calcular con esto: el cupo no se publica en tokens.

DESCUENTOS

las tablas de precios traen multiplicadores que bajan el precio efectivo: en Claude, los lotes tienen 50 % de descuento, y los aciertos de caché se cobran a 0,1 veces el precio de entrada en la mayoría de los modelos y a 0,025 en Fable 5.1 y Mythos 5.1 [12]. Para un libro de costos casero alcanza con el precio base.

DOS UNIDADES QUE NO SE MEZCLAN

los tokens miden texto y se pagan por millón; los puntos del cupo semanal miden cuánto de una suscripción quedó usado y se leen en una pantalla. No hay conversión pública entre unos y otros, y esta guía no inventa una. El libro de costos del capítulo 17 está escrito entero en la segunda unidad, que es la que Seba pudo leer.

13 Qué es MCP, y el servidor mínimo que construimos

Una IA usa una herramienta tuya sin que la herramienta sepa nada de la IA. Este capítulo explica el estándar que lo permite y muestra el servidor mínimo que se construyó en LAB1.

Tenés una carpeta con archivos, o una planilla, o una cuenta que querés que una IA haga siempre igual. Hoy, para cada aplicación de IA, alguien tendría que escribir esa conexión otra vez. MCP (Model Context Protocol) es un estándar para no reescribirla cada vez: *"MCP (Model Context Protocol) is an open-source standard for connecting AI applications to external systems"* [17].

En su arquitectura hay dos piezas con nombre. Un **servidor MCP** es *"a program that provides context to MCP clients"*, y lo que un servidor expone incluye **herramientas**: *"executable functions that AI applications can invoke to perform actions (e.g., file operations, API calls, database queries)"* [17]. El servidor no sabe qué IA lo va a usar; la IA no sabe cómo está hecho el servidor. Se hablan por el protocolo, como un enchufe y una toma.

Qué no es. No es un modelo, no es una biblioteca de consignas, y no es una conexión automática entre cualquier par de aplicaciones: dos programas se entienden por MCP solo si los dos lo implementan. Y ya no es "de Anthropic": el 9 de diciembre de 2025 Anthropic lo donó a la Agentic AI Foundation, un fondo dirigido de la Linux Foundation [18]; la especificación vigente está fechada 2026-07-28 [17].

El servidor mínimo de LAB1

Se construyó en la estación 3, en Python, con el paquete oficial del proyecto. Tiene dos herramientas y unas ciento cincuenta líneas, la mitad comentarios.

leer_registro(nombre) devuelve uno de los cuatro archivos del ejemplo sintético del capítulo 8 (las dos versiones de la fuente, el historial, el procedimiento) con su huella SHA-256 y una nota: la huella prueba que el archivo es idéntico al que se guardó; no dice si su contenido es verdadero. Solo acepta esos cuatro nombres. Si la IA pide otra cosa (`../otro.py`, una ruta absoluta), recibe un error, no un archivo.

calcular_participacion(version) lee la versión 1 o 2 de la fuente, valida lo que el procedimiento exige (que sea sintética, que la unidad sea MWh, que haya exactamente dos componentes A y B, que no haya negativos), calcula $100 \times A / (A + B)$ con fracciones exactas, redondea solo al escribir la salida, y devuelve la **ficha completa** del capítulo 8: fuente con huella, alcance, procedimiento con huella, resultado (500/11 y 45,454545 %), la comparación con la versión anterior (5,454545 puntos porcentuales;

LAS PRUEBAS

se escribieron antes que el servidor, como pide la primera disciplina. Ocho pruebas: cinco sobre la cuenta (valores exactos, redondeo, total cero, rutas rechazadas, validaciones de la fuente) y una de extremo a extremo que lanza el servidor de verdad, como lo haría una aplicación de IA, lista sus herramientas y las llama. Un archivo de registro guarda fecha, versión de Python, versión del paquete y las huellas de todo. La revisión de DeepSeek agregó tres pruebas y corrigió dos detalles (un denominador que salía como 110,0; una fracción que con decimales usaba el binario). Las pruebas pasan en el entorno donde se construyó, que es Linux. No se ejecutó todavía en la computadora de Seba, que es Windows; el LEEME que acompaña al servidor dice cómo hacerlo, paso a paso, y el registro se reescribe con esa corrida cuando ocurra.

aumento relativo de 13,636364 %; denominador de 100 a 110), incertidumbre ("no aplicable: entradas sintéticas; anotar esta limitación no es medirla") y revisión ("pendiente; ninguna aprobación se hereda").

Lo que el servidor no hace es tan importante como lo que hace: no escribe archivos, no se conecta a internet, no llama a ningún modelo, no lee nada fuera de su carpeta, no aprueba nada.

Cómo se conecta

La aplicación de escritorio de Claude lee un archivo de configuración con la lista de servidores MCP; se agrega una entrada con el nombre del servidor y el comando que lo lanza, se reinicia la aplicación, y desde entonces, en cualquier conversación, la IA puede pedir permiso para usar `leer_registro` o `calcular_participacion`. La ventana de permiso es la puerta humana de la cuarta disciplina, hecha programa. El nombre exacto del menú y el procedimiento vigente están en la documentación oficial de MCP (sección de inicio rápido para usuarios) y esa documentación gana sobre cualquier cosa que diga esta guía [17].

el proyecto publica SDK oficiales en varios lenguajes, clasificados por nivel de mantenimiento: TypeScript, Python, C#, Go y Rust en el primer nivel; Java y Ruby en el segundo; Swift, PHP y Kotlin en el tercero [19]. El SDK de MCP no es un "SDK de agentes": eso es otra cosa, de cada proveedor, que esta edición no verificó.

14 Las mañas, y dónde se comparten las consignas

Qué hacen los que ya usan esto, dónde se publican consignas para copiar, y qué mirar antes de copiar una ajena. Sin consignas secretas: el capítulo 3 ya dijo que no existen.

Una persona escribe una consigna, la respuesta no le sirve, y prueba agregarle dos ejemplos de lo que quiere. Ahí entra la primera técnica que las guías oficiales documentan. En la de Claude: *"A few well-crafted examples (known as few-shot or multishot prompting) improve accuracy and consistency"* [20]. Otra persona le dice al modelo qué papel cumple antes de pedirle la tarea: *"Setting a role in the system prompt focuses Claude's behavior and tone for your use case"* [20]. Una tercera separa las instrucciones de los datos con etiquetas: *"XML tags help Claude parse complex prompts unambiguously, especially when your prompt mixes instructions, context, examples, and variable inputs"* [20].

VIBE CODING

se llama así a programar describiéndole a una IA lo que se quiere y aceptando el código que produce, corrigiendo por resultado más que por lectura del código. El término circula desde 2025 (se lo atribuye a un investigador del área; esta edición no verificó la cita). Lo que habilita: que alguien sin oficio de programador construya algo que funciona en una tarde, como el servidor del capítulo 13, que se escribió y probó en una sesión. Lo que se discute: que el que construye puede no entender lo que construyó, que el desorden se acumula sin que nadie lo vea, y que

Y una preferencia que corrige una maña vieja: la misma guía dice hoy *"Prefer general instructions over prescriptive steps. A prompt like 'think thoroughly' often produces better reasoning than a hand-written step-by-step plan"* [20]. La guía de OpenAI organiza lo suyo en secciones sobre roles de mensaje, formato con Markdown y XML, ejemplos y modelos de razonamiento, y trae un ejemplo de pedir un formato de salida exacto: *"Only output a single word in your response with no additional formatting or commentary"* [21]. La de Google distingue *"few-shot"* de *"zero-shot"*, pide poner las restricciones, el rol y el formato de salida al principio, y recomienda *"Break down instructions: Instead of having many instructions in one prompt, create one prompt per instruction"* [22].

Las consignas del capítulo 3 usan estas técnicas sin nombrarlas: rol, ejemplos, formato de salida, separación entre lo que se pide y lo que se adjunta. Ninguna se presenta como garantía de resultado; es una regla del kit, no un dato de las guías.

Dónde se comparten

Los proveedores publican colecciones de consignas y recetas. La de OpenAI está en su "cookbook", *"Cookbooks to help you build with OpenAI models"*, con entradas fechadas (por ejemplo, 10 de septiembre de 2026) pero sin indicar modelo [23]. La biblioteca de consignas de Anthropic que existía en ediciones anteriores de su documentación no existe hoy como página: su dirección redirige a la guía de buenas prácticas [20]. Hay además repositorios y comunidades (prompts.chat, antes "Awesome ChatGPT Prompts"; el hub de LangSmith; la galería de Google AI Studio; FlowGPT) que esta edición no pudo abrir para verificar qué indican sobre modelo y fecha: van al final, en la lista de lo identificado y no leído.

Las dos consignas que se sumaron

Las consignas 7 y 8 del capítulo 3 salieron de esta vuelta: la de limpiar después de construir viene de una observación sobre el *vibe coding*, y la pregunta que falta se aplica al final de cualquier estación, antes de decidir. Desde esta edición, la IA que gestiona el tren tiene instrucción de hacer esa pregunta por su cuenta cuando corresponda, y de contestarla.

"funciona" no es lo mismo que "es correcto" ni que "es seguro". Ninguna de las dos cosas anula a la otra. Este manual lo trata como lo que es: una forma de trabajar que hace más importantes, no menos, las pruebas escritas antes, la revisión por otro y la limpieza de después.

ANTES DE COPIAR UNA CONSIGNA AJENA

tres preguntas. ¿Para qué modelo está escrita? (Las colecciones suelen no decirlo, y los modelos cambian.) ¿De qué fecha es? (El cookbook la pone; muchas no.) ¿Qué materiales supone que tenés? (Archivos, cuentas, conexiones.) Si no podés contestar las tres, la consigna es una idea para probar, no una herramienta lista.

15 Qué venden los cursos y qué hace falta

Un temario puede estar vendiendo instrucciones, integración, práctica guiada o una promesa comercial. No son lo mismo y conviene saber cuál estás comprando.

Contenido	Qué aprendés	Prueba que deberías poder hacer al terminar
Consignas y contexto	Definir tareas, restricciones, ejemplos y salidas	Repetir un encargo sobre varios casos y reconocer cuándo falla
Automatización visual	Conectar pasos, transformar campos y manejar errores	Procesar una entrada nueva y detener una entrada inválida
Agentes y herramientas	Dar acceso acotado a búsqueda, archivos o funciones	Reconstruir una acción desde su entrada y su registro
Plantillas o paquetes	Adaptar un flujo ya preparado	Cambiar una fuente y reparar el flujo sin depender de su autor
Negocio y servicio	Delimitar un problema, una entrega y un cliente	Contrastar la propuesta con una necesidad y una disposición a pagar reales

Es una pauta de evaluación de este laboratorio, no un relevamiento del mercado.

Dos puntos de entrada que se revisaron. Make Academy, *Automation to AI Agents: Foundation*: seis cursos y una evaluación (fundamentos de IA, modelos de lenguaje, IA en Make, agentes, automatización, agentes en Make), entre dos y tres horas y media; la academia declara "Yes, 100% free. No subscriptions, no surprises.", lo que no equivale a operaciones ni servicios externos ilimitados. Se examinó el temario publicado; no se cursaron las clases [4, 5]. Hugging Face, *AI Agents Course*: pide "Basic knowledge of Python" y "Basic knowledge of LLMs"; certificado gratuito al completar la primera unidad, una tarea de caso y el desafío final. Es una segunda etapa razonable cuando ya podés leer un archivo, reconocer una variable y entender una función [6].

Cómo decidir una compra. Pedí ver una entrada defectuosa, su diagnóstico y su recuperación. Buscá ejercicios corregidos, archivos exportables y límites explícitos. Un certificado, una demostración exitosa o la cantidad de consignas no prueban competencia profesional. En esta edición no se recomienda comprar ningún curso. Tampoco se confirma la gratuidad actual del curso de agentes de DeepLearning.AI que los revisores de LAB0 citaron: se intentó de nuevo en LAB1 y la página no se pudo leer; va al final, en la lista de lo identificado y no leído. La siguiente

lo que hizo falta en LAB1, en orden de aparición: saber copiar y pegar sin editar; saber calcular una huella (o correr el script que la calcula); leer una tabla de precios oficial con la fecha al lado; instalar un paquete de Python y correr una prueba. Ninguna es un curso; todas se aprenden en una tarde con el material de este manual.

habilidad a adquirir se decide por lo que te impide terminar una práctica, no por completar un catálogo.

16 Llevarlo a tu trabajo, y lo mínimo que conviene saber

El método recién sirve cuando lo usás en algo tuyo. Empezá por algo chico que ya sepas hacer a mano.

Una práctica para hacer ahora

Abrí el ejemplo del capítulo 8: $A=40$, $B=60$. Antes de pedirle ayuda a nadie, escribí tres respuestas a mano:

1. Si A pasa a 50, ¿qué denominador usás?
2. ¿Qué hacés con el resultado anterior?
3. ¿La aprobación que ya tenías sigue valiendo?

Después compará con el capítulo 8. La práctica termina cuando podés reconstruir el recorrido sin depender de una explicación nueva. La segunda práctica, más difícil, es introducir un dato faltante y comprobar que tu procedimiento se detiene en vez de inventarlo.

Un ejemplo hipotético: castings

Este ejemplo no ocurrió: es una aplicación imaginada para mostrar que el método no es solo para números.

Una actriz se presenta a muchos castings. Cada uno pide material distinto: un video con cierta duración, un texto breve, fotos recientes. Usa una IA para preparar los materiales y adaptar el texto a cada convocatoria.

El método se traduce casi directo. **El encargo:** para qué convocatoria es, qué pide exactamente, qué material propio existe y qué límites hay. **Las entradas:** la convocatoria original guardada, no un resumen; el video en su versión exacta; los textos anteriores. **La entrega:** el texto adaptado, junto con qué versión de cada material se usó. **La revisión:** leerlo y comprobar que no diga algo que ella no pueda sostener. **La decisión y el envío son de ella:** la IA prepara, no manda. Ninguna postulación sale sin que la lea.

¿Para qué registrar esto? Porque a la quinta convocatoria nadie recuerda qué versión del video mandó a cuál, ni qué texto funcionó. Un archivo con qué se mandó, a quién, cuándo y en qué versión responde eso en diez segundos: es la misma ficha del capítulo 8 con otros campos.

Sirve la misma pregunta en cualquier rubro: ¿qué le tendrías que poder mostrar a otra persona para que entienda cómo llegaste a este resultado? Si la respuesta es “nada, lo hice y salió”, ahí falta el registro.

Cómo elegir tu primera tarea

Que sea repetitiva, que ya la sepas hacer, que no tenga consecuencias graves si sale mal y que puedas comprobar el resultado vos mismo. Si no se puede comprobar, no sirve como primera práctica, por más ganas que tengas de automatizarla.

Lo mínimo que conviene saber entrando a este siglo

Lo mínimo que conviene saber cabe en cuatro frases, y las cuatro salen de este laboratorio. Que una respuesta puede sonar convincente y estar equivocada, y que el revisor puede agregar un error nuevo; por eso se conservan fuente, versión y motivo. Que una IA con herramientas puede hacer más que una IA sin ellas, y que darle una herramienta es una decisión tuya, con una puerta que vos abrís. Que pagar por mes y pagar por uso son cosas distintas, y que lo que no se mide se agota. Y que entre dos etapas siempre hay una persona, hasta que decida no estarlo, y esa decisión también se registra.

Sobre el futuro del trabajo, esta edición no trae fuentes verificadas: los informes de la OIT y de la OCDE de 2025 y 2026 que se identificaron no se pudieron abrir en la sesión que escribió esta sección, y van al final con sus títulos y direcciones, en la lista de lo identificado y no leído. No se pone en esta guía una conclusión que nadie leyó. Lo que sí se puede decir, porque está en los registros de LAB0 y LAB1, es que en las dos vueltas la parte del trabajo que ninguna IA hizo fue decidir: qué pedir, qué aceptar, qué rechazar con motivo, cuándo parar. Si hay un futuro humano en esto, empieza por no regalar esa parte.

La posibilidad para quien tiene una visión

Lo que costó LAB1, en dinero, fue una suscripción mensual y cero en herramientas: cuatro de las cinco IA eran gratuitas, el paquete de MCP es gratuito, y el resto fue tiempo de atención. Lo que se construyó con eso: este manual, un servidor que una aplicación de IA puede usar, un método que sobrevivió a tres fallas de herramientas, y un registro con el que otra persona puede repetir todo. No hace falta permiso de nadie para hacer lo mismo con otro problema.

Eso no es mérito: es una puerta que hoy está abierta y que no siempre lo estuvo. Quien tiene una visión (un problema que conoce bien y quiere resolver de otra manera) puede pasar por esa puerta con lo que este manual muestra: un encargo, un ejemplo chico, una prueba escrita antes, seis costumbres. Lo que no puede comprar ni copiar es la experiencia de haberlo corregido varias veces; esa se acumula haciéndolo. La guía no promete que del otro lado haya ingresos, ni clientes, ni exclusividad. Promete que del otro lado hay un método que se entiende y un registro que se puede mostrar.

guía el camino a favor de las personas: ese fue el pedido de Seba para esta sección, y la forma de cumplirlo sin sermón es mostrar dónde estuvo la persona en cada estación del tren. Está en la tabla del capítulo 6, en la columna "quién decide".

17 Registro, trazabilidad y presupuesto

Los cuatro registros, los manifiestos, la intervención humana anotada y el libro de costos de esta corrida.

Los cuatro registros de LAB0 (fuente, ejecución, resultado, revisión) siguen siendo los mismos en LAB1, con una diferencia de disciplina: en cada estación se anotó también qué hizo la persona. Qué decidió (el índice, los cambios de reparto cuando una IA falló, qué hallazgos aceptar), qué corrigió (ocho enmiendas al encargo, con motivo), qué verificó a mano (los porcentajes de uso), qué dejó pendiente (correr el servidor en su computadora; contestar una pregunta de comprensión). Una trazabilidad que solo anota lo que hicieron las IA deja afuera la parte que sostiene la decisión.

Cada archivo que viajó entre estaciones tiene nombre, versión y huella SHA-256 en un manifiesto; los originales de cada IA se conservan tal como llegaron, con una nota de recepción que dice si llegaron adjuntos o pegados. Las dos veces que dos textos llegaron con el mismo nombre, se renombraron al recibirlos y se anotó la equivalencia.

el techo acordado era 45 puntos para LAB1 y un freno en 35. La cuenta pasó el freno, pero por otro proyecto que compartía el cupo; LAB1 solo llevaba unos ocho puntos en ese momento. Seba decidió seguir, y la decisión quedó anotada con su motivo. Un freno que no se registra cuando se pasa no es un freno.

El libro de costos de LAB1

Estación	Qué se hizo	Qué medidor	Lectura de Seba, en % usado	Gasto con cargo
0	Verificación del paquete, encargo en dos versiones, dos consignas	Cupo semanal, general	de "no leído" a 12 %	Ninguno
1 y 3	Verificación de 19 fuentes con un agente; servidor MCP con pruebas	Cupo semanal, general	de 12 % a 19 %	Ninguno
2	Contraste de dos borradores; consignas de revisión	Cupo semanal, general	58 %, subida por otro proyecto que comparte el cupo	Ninguno
4	Estados de trece hallazgos; servidor v2 con ocho pruebas	Cupo propio de la sesión	47 %	Ninguno
5	El texto de esta guía y la capa privada	Cupo semanal, general	55 % al entrar al diseño	Ninguno

Estación	Qué se hizo	Qué medidor	Lectura de Seba, en % usado	Gasto con cargo
6	Sistema visual (cinco vueltas con Seba), figuras, tapa y PDF	—	la lectura de cierre va en la estación 7	7 créditos de Higgsfield, por la imagen de tapa

Tres advertencias, porque esta tabla es fácil de leer mal, y leerla mal es exactamente el error del capítulo 8. **Son lecturas, no consumo:** la pantalla dice cuánto del cupo está usado en total, y durante toda la corrida ese cupo lo compartió otro proyecto; la diferencia entre dos filas no es lo que gastó LAB1. **Hay dos medidores y no se suman:** el cupo semanal general y el cupo propio de la sesión son dos barras distintas en la misma pantalla, y la fila de la estación 4 es de la segunda. **La unidad es el porcentaje de un cupo semanal,** no el dólar ni el token: el equivalente en dinero se obtiene prorrateando la suscripción, 137.903 pesos argentinos de septiembre de 2026 dividido por los cien puntos de cada una de las 4,3 semanas que tiene un mes, que da unos 320 pesos por punto. Ese número sirve para tener una idea del orden de magnitud, no para facturar, y no es un precio por token. Las IA gratuitas costaron cero y, en dos casos, tiempo de espera.

18 Qué no promete esta guía, y dónde seguir

Lo último que conviene aprender es a reconocer lo que un material no sostiene, incluido este.

- **Ingresos.** Que un método sea prolijo no implica que alguien pague por él.
- **Exclusividad o imposibilidad de copia.** Nada de esto es difícil de copiar; lo que cuesta acumular es la experiencia de haberlo corregido varias veces.
- **Conexión automática entre aplicaciones.** Ningún chat le pasa trabajo a otro por el solo hecho de pedirlo.
- **Independencia entre revisores.** Dos IA revisando lo mismo pueden compartir fuentes, supuestos y errores.
- **Consignas infalibles.** Las ocho del capítulo 3 no se evaluaron en varias aplicaciones.
- **Que leer alcance.** La habilidad se demuestra con una práctica y una revisión.
- **Que el servidor mínimo sirva con datos reales,** ni que funcione igual en otra aplicación que la probada.

- Que "saber organizarse" sea un talento. La guía muestra seis costumbres practicadas dos veces por una persona.

Estado de este documento. Edición 03, texto final propuesto el 15 de septiembre de 2026 y corregido el 17 con la última lectura de Seba, pendiente de su aprobación. La edición 02 fue leída por Seba y devuelta con nota y pedidos; leyó después dos versiones de este texto y pidió cambios que están incorporados; ninguna de esas lecturas aprueba esta versión. Las pruebas de comprensión con lectores externos siguen pendientes.

Fuentes

Documentación primaria, consultada el 14 de septiembre de 2026 salvo indicación; las direcciones son las finales después de las redirecciones de ese día. [1] y [3] se reconsultaron ese día durante la edición 02; [2], [7] a [10] y [4] a [6] se verificaron al cerrar LAB0 (registro LAB0-FABLE-CIERRE-v1).

[1] Anthropic, *Building effective agents* (19/12/2024). Flujos con recorridos predefinidos frente a agentes que dirigen su propio proceso. anthropic.com/engineering/building-effective-agents. [2] OpenAI, *Prompt engineering*. Variabilidad de las respuestas y evaluación de instrucciones. developers.openai.com/api/docs/guides/prompt-engineering. [3] W3C, *PROV Model Primer* (30/04/2013). Especificación para expresar registros de procedencia. [w3.org/TR/prov-primer/](https://www.w3.org/TR/prov-primer/). [4] Make Academy, *Automation to AI Agents: Foundation*, temario. [5] Make Academy, preguntas frecuentes sobre gratuidad. [6] Hugging Face, *AI Agents Course*, requisitos y certificado. [7] Anthropic, *Demystifying evals for AI agents* (09/01/2026): acciones declaradas, registros y resultados. [8] OpenAI, *GPT-5.4 Model*. [9] OpenAI, *GPT-5.4 Pro Model*, variante distinta: se comprobó el error de comparación. [10] OpenAI, *Authentication*: suscripción y facturación por clave.

[11] OpenAI, *Managing billing for ChatGPT and the API platform*. help.openai.com/en/articles/9039756. [12] Anthropic, *Pricing*, Claude Platform Docs. platform.claude.com/docs/en/about-claude/pricing. [13] Anthropic, *Plans & Pricing*. claude.com/pricing. [14] Anthropic, *What is the Pro plan? y How do usage and length limits work?*, Claude Help Center. support.claude.com/en/articles/8325606-y-11647753. [15] OpenAI, *What is ChatGPT Plus?* help.openai.com/en/articles/9793127. [16] OpenAI, *API Pricing*. openai.com/api/pricing. [17] Model Context Protocol, *What is MCP y Architecture overview*, especificación 2026-07-28. modelcontextprotocol.io. [18] Model Context Protocol, *MCP joins the Agentic AI Foundation* (09/12/2025). blog.modelcontextprotocol.io/posts/2025-12-09-mcp-joins-agentic-ai-foundation/. [19] Model Context Protocol, *SDKs*. modelcontextprotocol.io/docs/sdk. [20] Anthropic, *Prompting best practices*, Claude Platform Docs. platform.claude.com/docs/en/build-with-claude/prompt-engineering/claude-prompting-best-practices. [21]

OpenAI, *Prompt engineering*.
developers.openai.com/api/docs/guides/prompt-engineering. [22] Google,
Prompt design strategies, Gemini API. ai.google.dev/gemini-
api/docs/prompting-strategies. [23] OpenAI, *Cookbook*.
developers.openai.com/cookbook.

Identificado y no leído. OIT, *Generative AI and jobs: A 2025 update*
(ilo.org/publications/generative-ai-and-jobs-2025-update) y *Workers'*
exposure to AI (febrero de 2026); OCDE, *AI and skills: what we know so*
far (junio de 2026); DeepLearning.AI, *AI Agents in LangGraph*; los
repositorios y comunidades de consignas del capítulo 14. Se citan por su
título para que quien quiera pueda abrirlas; esta edición no los leyó.

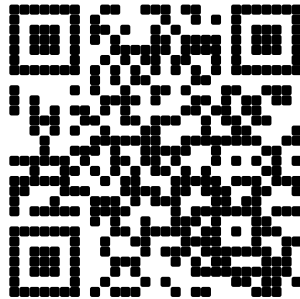
Evidencia local. [L1] Archivos sintéticos del ejemplo y su comprobación
aritmética, rehecha en LAB1 con fracciones dentro del servidor MCP. [L2]
Entregas originales de Gemini, Kimi, DeepSeek y Grok en LAB0 y LAB1,
con sus decisiones de revisión, notas de recepción y huellas. [L3] Encargo
LAB1 en dos versiones, contrastes, cierres de estación y libro de costos.
[L4] Testimonio de Seba sobre la frase de Sam Altman del capítulo 3: la vio
en una historia de Instagram de Grace Ling, publicada durante la fiesta de
lanzamiento de GPT-5.5 en San Francisco, alrededor del 15 de mayo de
2026. Las historias caducan a las veinticuatro horas y no quedó registro
público recuperable; no hay transcripción oficial ni video. No son enlaces
públicos: este PDF no transporta el archivo.

Cómo se hizo este PDF

Compuesto el 17/09/2026 desde el archivo de texto **LAB1-ED03-TEXT0-v5.md** (SHA-256
66ac967b1ddef0ff...) con un programa de composición propio (HTML y Chromium),
sin retoques a mano. Tipografía: Noto Serif CJK (diseño latino de Source Serif)
para el cuerpo y los títulos; Carlito para la columna lateral, las tablas y los pies.
Papel blanco, amarillo mostaza como único color, carbón y grises: la paleta que
Seba pidió con una referencia de manual de marca. Figuras vectoriales originales.
La imagen de tapa se generó con una herramienta de IA a partir de una
descripción del motivo, sin usar ninguna fotografía como entrada.

Esta guía la hizo una persona con varias IA, a mano, dos veces, y anotó todo.
Si te sirvió y querés que hagamos algo parecido con un problema tuyo, escribime.

Sebastián Liscia



[SEBASTIANLISCIA.COM](https://sebastianliscia.com)