RODRIGO ORTEGO · CARA SUR · PATAGONIA · 2026

El Manual

De qué es una IA hasta construir con agentes: la teoría completa, en criollo, sin humo. Se lee solo — los 80 videos del roadmap quedan como profundización opcional.

12 capítulos + anexo | ~45 min de lectura | Gratis · sin registro | v1 · julio 2026

ÍNDICE

PARTE I

Entender

CAPÍTULO 01

Qué es la inteligencia artificial (y qué es esta ola)

"Inteligencia artificial" no es una cosa nueva ni una sola cosa. Es un paraguas que existe hace décadas: programas que resuelven tareas que antes pedían inteligencia humana. Adentro conviven cosas muy distintas — desde reglas escritas a mano ("si el cliente debe más de 30 días, mandá recordatorio") hasta sistemas que **aprenden patrones de los datos** en vez de seguir reglas fijas. Eso segundo se llama *machine learning* y es lo que recomienda películas en Netflix o detecta fraudes en tu tarjeta desde hace años.

Lo que explotó desde fines de 2022 es una rama específica: la **IA generativa**. La diferencia es qué produce. La IA "clásica" clasificaba o predecía (¿este mail es spam? ¿cuánto voy a vender en marzo?). La generativa **crea contenido**: texto, imágenes, código, audio. ChatGPT, Claude y Gemini son la cara visible de esto.

Y acá viene la distinción más importante de todo el manual, la que separa a los que la usan bien de los que se frustran:

El software tradicional es determinístico. La IA generativa es probabilística.

En Excel, la misma fórmula con los mismos datos da siempre el mismo resultado. En un modelo de IA, la misma pregunta puede dar respuestas distintas — mejores, peores, con otro enfoque. No es un defecto: es la naturaleza de la herramienta. Y cambia por completo cómo se opera: a un Excel se lo programa; a una IA **se le habla, se itera y se verifica**. Todo lo que sigue en este manual son consecuencias de esa frase.

> EN UNA FRASE
>
> No es un Excel que habla; es otra categoría de herramienta, y se opera distinto.

> PROBALO
>
> Hacele exactamente la misma pregunta dos veces a ChatGPT o Gemini, en dos chats nuevos. Compará las respuestas. Eso que ves — dos outputs distintos del mismo input — es la propiedad central que tenés que internalizar.

Para profundizar (videos): You're Not Behind: Become AI-Native (Pilar 1) · AI and human evolution — Harari (Pilar 1, para el marco grande).

CAPÍTULO 02

Qué es un LLM y cómo funciona por dentro

LLM significa *Large Language Model* — modelo grande de lenguaje. Detrás del nombre hay una idea sorprendentemente simple: es un sistema entrenado para hacer una sola cosa, **predecir la próxima palabra**. Le mostrás "el gato se subió al..." y calcula qué sigue con mayor probabilidad ("techo", "árbol", "sillón").

Lo contraintuitivo es que de esa tarea boba, hecha a escala gigantesca, **emergen** capacidades que nadie programó explícitamente: redactar un contrato, traducir, resumir un balance, escribir código. Para predecir bien la próxima palabra de cualquier texto humano, el modelo terminó teniendo que "entender" gramática, hechos, lógica, estilos.

El entrenamiento tiene dos fases que conviene conocer:

1. **Pre-entrenamiento:** el modelo "lee" una porción enorme de internet, libros y código. De ahí saca sus patrones. Es carísimo y se hace una vez.
2. **Afinado:** humanos le enseñan a comportarse — a ser útil, a seguir instrucciones, a rechazar pedidos problemáticos. Acá el "predictor de texto" se convierte en asistente.

Después de eso, el modelo queda **congelado**. Dos consecuencias prácticas enormes:

- **No aprende de tus chats.** Lo que le contás hoy no lo "sabe" mañana en otro chat (salvo funciones de memoria explícitas, que son otra cosa y se configuran). Tus datos no entrenan al modelo por defecto en los productos pagos — pero verificá la configuración de privacidad igual.
- **No tiene una base de datos de hechos.** "Sabe" cosas como vos sabés tu idioma: patrones incorporados, no registros consultables. Por eso responde con la misma fluidez cuando sabe y cuando no — tema del capítulo 4.

Los tokens. Un detalle técnico que vale plata: el modelo no lee palabras ni letras, lee **tokens** — fragmentos de texto (una palabra corta puede ser un token; una larga, dos o tres). Es la unidad en la que lee, escribe, "piensa" y **cobra**: los precios de las APIs se expresan en millones de tokens. Para el uso por chat no lo ves, pero cuando quieras automatizar algo (Parte III), entender tokens es entender el costo y el límite físico de cada operación.

> EN UNA FRASE
>
> Le hablás a un motor de lenguaje entrenado y congelado, no a un buscador ni a una base de datos.

Para profundizar (videos): AI prompt engineering: A deep dive (Pilar 1 — la teoría detrás) + los videos de fundamentos del roadmap.

CAPÍTULO 03

La memoria de la IA: ventana de contexto

Si el capítulo 2 explica qué sabe el modelo, este explica qué **ve**. Y la respuesta es: solamente lo que entra en su **ventana de contexto**.

La ventana es la mesa de trabajo del modelo. Cuando responde, lo único que tiene delante es lo que está sobre esa mesa: tu conversación hasta ahí, los documentos que adjuntaste, las instrucciones que le diste. Nada más. No "recuerda" el chat de la semana pasada, no ve tus archivos, no sabe qué le dijiste en otra pestaña.

La mesa es grande — los modelos actuales manejan cientos de miles de tokens, o sea cientos de páginas — pero es **finita**. Y de ahí sale el fenómeno que todo usuario sufre sin saber por qué: en un chat muy largo, la IA "se olvida" de lo que acordaron al principio. No se olvidó: lo del principio **se cayó de la mesa** (o quedó resumido y perdió detalle).

Tres prácticas que se desprenden directo de esto:

1. **Un chat nuevo por tema.** Arrastrar un chat eterno degrada la calidad. Cerrá y abrí.
2. **Repetí el contexto clave.** Si algo es importante, que esté cerca del final de la conversación, no enterrado 40 mensajes atrás.
3. **Tené un documento maestro.** Tu contexto de negocio escrito una vez (qué hacés, tus números clave, tu tono), listo para pegar al arrancar cualquier conversación seria. Esto se convierte en método en el capítulo 6.

> EN UNA FRASE
>
> La IA no tiene memoria; tiene una mesa de trabajo de tamaño fijo — y vos decidís qué ponés arriba.

Para profundizar (videos): 5 maneras de dominar el contexto (Pilar 1) — el puente perfecto entre este capítulo y el de prompting.

CAPÍTULO 04

Alucinaciones: por qué inventa y qué hacer

Este es el capítulo que evita que te quemes. Los modelos de lenguaje **inventan cosas con total seguridad**: citas que no existen, números plausibles pero falsos, leyes con artículos inventados, links rotos. Se llama *alucinación* y no es un bug que van a arreglar el mes que viene — es el modo de falla natural de la herramienta.

¿Por qué pasa? Volvé al capítulo 2: el modelo es un predictor de texto. Su trabajo es **siempre completar**. Cuando sabe, completa bien. Cuando no sabe, completa igual — con la misma fluidez y el mismo tono seguro. No tiene un mecanismo interno que distinga "esto lo sé" de "esto lo estoy inventando". La seguridad del tono no es evidencia de nada.

Dónde pega más fuerte: **datos puntuales**. Números específicos, fechas, citas textuales, referencias legales o académicas, precios, links. Donde menos pega: tareas de forma — redactar, resumir, reestructurar, traducir — porque ahí el material se lo diste vos.

La mitigación es un proceso, no una esperanza:

1. **Dale vos las fuentes.** Si la respuesta tiene que salir de tus documentos, adjuntalos. Se llama *grounding* y es el tema del capítulo 7 — reduce la alucinación de forma drástica.
2. **Activá la búsqueda web** cuando preguntes por hechos actuales, y pedí que cite fuentes.
3. **Verificá lo crítico, siempre.** Todo número, cita o dato que vaya a una decisión o a un documento con tu firma, se chequea contra la fuente. Sin excepción.
4. **Usala donde el error es barato.** Borradores, ideas, estructura: el costo de una alucinación es cero porque vos revisás. Asesoramiento legal sin verificar: el costo puede ser tu empresa.

EN UNA FRASE

La pregunta no es "¿alucina?" — alucina. La pregunta es qué proceso ponés alrededor para que no te importe.

PROBALO — EL TEST DE LA ALUCINACIÓN

Preguntale a un chatbot (sin búsqueda web) por datos de tu industria o tu ciudad que conocés de primera mano. Pedile fuentes. Verificalas. Ver el límite con tus propios ojos vale más que cualquier advertencia.

Para profundizar (videos): los de NotebookLM del Pilar 2 muestran el antídoto en acción (respuestas ancladas a tus fuentes, con citas).

CAPÍTULO 05

El mapa de herramientas

El ecosistema cambia cada mes, pero el mapa de fondo es estable. Lo que necesitás no es conocer 40 herramientas: es saber **qué categoría resuelve qué problema** y tener una opción probada en cada una.

HERRAMIENTA	QUÉ ES	CUÁNDO USARLA
ChatGPT (OpenAI)	El generalista de mayor ecosistema	Uso diario amplio, GPTs personalizados, imágenes
Claude (Anthropic)	Fuerte en texto largo, análisis fino y código	Documentos extensos, redacción cuidada, programar
Gemini (Google)	Integrado al mundo Google, multimodal, capa gratis potente	Si vivís en Gmail/Drive/Docs; research con video y audio
NotebookLM (Google)	Responde SOLO desde los documentos que le cargás	Research sobre TUS fuentes: manuales, contratos, normativa
n8n (y similares)	Automatización visual de flujos	Procesos que corren solos: la IA como pieza, no como chat
Claude Code / Antigravity	Agentes de programación	Vibe coding: construir apps y herramientas (Parte III)

Tres reglas para no marearse:

1. **Elegir mal cuesta tiempo y plata.** No todas sirven para todo: para leer 200 páginas tuyas, NotebookLM le gana a pegar texto en un chat; para programar, Claude; para lo cotidiano, la que ya uses.
2. **Empezá con una y en serio**, no con seis a la vez. La curva de una herramienta bien usada rinde más que el tour por todas.

3. **Ignorá el lanzamiento de la semana.** Si tu problema está resuelto, el último modelo no te cambia nada. Volvé al mapa cuando tengas un problema nuevo, no cuando haya una novedad nueva.

> EN UNA FRASE
>
> Conocé las categorías, elegí un caballo por categoría, y cambialo solo cuando tu problema lo pida.

Para profundizar (videos): ¿Gemini o ChatGPT? Elige mal y perderás tiempo y dinero (Pilar 2 — exactamente este capítulo) · Master 85% of Google Gemini in 12 Minutes (Pilar 2).

PARTE II

Operar

CAPÍTULO 06

Prompting: cómo hablarle

Primera verdad incómoda: **el 80% de los malos resultados no son culpa del modelo — son falta de contexto**. La IA no lee mentes. Si le pedís "escribime una propuesta" sin decirle para quién, de qué, con qué tono y con qué datos, te devuelve la propuesta genérica que le devolvería a cualquiera.

Anatomía de un buen pedido. No hay fórmulas mágicas, hay ingredientes:

- **Rol:** desde dónde quiero que responda. *"Actuá como un analista financiero con experiencia en PyMEs."*
- **Contexto:** lo que necesita saber. Tu negocio, tu cliente, tus números, el trasfondo. Acá se gana o se pierde el partido.
- **Tarea:** qué quiero, concreto. No "ayudame con esto" sino "compará estas dos opciones y recomendame una".
- **Formato:** cómo lo quiero. Tabla, mail de tres párrafos, lista priorizada, tono formal o criollo.
- **Ejemplos:** si tenés un modelo de cómo te gusta, mostráselo. Un ejemplo vale más que diez adjetivos.

La iteración es el método. El primer output es un borrador, no la entrega. Los que le sacan jugo a la IA no escriben el prompt perfecto de entrada: conversan. "Está bien, pero más corto." "Ahora criticá tu propia respuesta: ¿qué le falta?" "Rehacelo para un cliente que no sabe de finanzas." Cada vuelta mejora el resultado a costo casi cero.

De artesanía a sistema. Cuando notás que repetís el mismo contexto en cada chat, es hora de sistematizar:

- **Instrucciones permanentes** (custom instructions, Projects, Gems): configurás una vez quién sos, qué hacés y cómo querés las respuestas — y aplica a todos los chats.
- **El documento maestro** (*master prompt*): un archivo con el contexto completo de tu negocio — qué vendés, a quién, tus números clave, tu tono, tus reglas. Lo pegás al inicio de cualquier conversación importante y el modelo arranca sabiendo lo que un empleado tardaría meses en absorber. Es la pieza de prompting con mejor relación esfuerzo/retorno que existe: se escribe una tarde, se usa para siempre.

> EN UNA FRASE
>
> Contexto + iteración + sistema. El prompt perfecto no existe; el proceso que lo vuelve innecesario, sí.

> PROBALO
>
> Escribí tu documento maestro (una página: negocio, clientes, números, tono). Usalo en el próximo pedido serio y compará contra lo que te daba sin él.

Para profundizar (videos): 5 maneras de dominar el contexto · The Master Prompt Method · El mejor mensaje de ChatGPT (la iteración) · 12 ChatGPT Tricks (custom instructions) — todos Pilar 1.

CAPÍTULO 07

Research con tus propios documentos

El problema real de un profesional o un dueño de PyME casi nunca es "quiero saber algo de internet". Es: **tengo 200 páginas propias — contratos, manuales, normativa, balances, informes — y necesito respuestas que salgan de ahí**. Ni la ventana de contexto alcanza siempre, ni el modelo conoce tus documentos, y el capítulo 4 ya te mostró qué pasa cuando responde "de memoria".

La solución tiene nombre: **grounding** — anclar las respuestas a tus fuentes. En vez de preguntarle al modelo qué sabe, le das los documentos y le exigís que responda **solo desde ahí**, citando de dónde sacó cada cosa. El efecto es doble: la alucinación cae en picada, y cada respuesta trae su comprobante.

NotebookLM es esto empaquetado. Le cargás tus fuentes (PDFs, docs, links, videos) y todo lo que responde sale de ellas, con citas que podés clickear y verificar. Para research sobre material propio es hoy la herramienta más segura que existe para un no-técnico: si no está en tus fuentes, te dice que no está.

RAG, en criollo. La técnica detrás merece dos párrafos porque explica los límites. RAG (*Retrieval-Augmented Generation*) funciona así: tus documentos se trocean en fragmentos; cuando preguntás, un buscador encuentra los fragmentos más relevantes a tu pregunta; y el modelo responde **usando solo esos fragmentos** en su ventana. Es como un asistente que no leyó los 12 tomos, pero sabe exactamente qué páginas abrir antes de contestarte.

Entenderlo te dice cuándo confiar: RAG brilla respondiendo preguntas puntuales sobre corpus grandes, y flaquea cuando la respuesta exige leer *todo* a la vez (por ejemplo, "resumime la evolución del tono a lo largo de los 5 años de actas"). Para eso, mejor documentos completos en la ventana de un modelo con contexto grande.

Caso PyME típico: cargar en NotebookLM el convenio colectivo de tu actividad, tus contratos con proveedores y tu manual de procedimientos. De ahí en más: "¿qué dice el convenio sobre las horas extra en sábado?" con cita exacta, en segundos, sin depender de memoria — ni tuya ni del modelo.

> EN UNA FRASE
>
> No le preguntes qué sabe — dale tus fuentes y exigile que responda desde ahí, con citas.

Para profundizar (videos): los tutoriales de NotebookLM del Pilar 2 (12 casos de uso épicos · Curso NotebookLM + Gemini) · Google cambió el RAG para siempre + File Search API (Pilar 2) para la técnica.

PARTE III

Producir

CAPÍTULO 08

Agentes: la IA que hace, no solo responde

Hasta acá, la IA fue una conversación: preguntás, responde, y todo lo que pase después lo hacés vos. Un **agente** rompe ese molde. Le das un **objetivo** — no una pregunta — y el sistema planifica los pasos, usa herramientas (navegar, leer archivos, escribir, ejecutar código), evalúa el resultado de cada paso y sigue hasta terminar o trabarse.

La diferencia práctica: a un chatbot le preguntás "¿cómo armo un informe de ventas?"; a un agente le decís "armá el informe de ventas de junio con los CSV de esta carpeta y dejámelo en PDF". El chatbot te explica; el agente **lo hace**.

¿Cómo se conecta a tus cosas? Para que un agente opere tu mundo real necesita enchufes. Eso son los *connectors* y el protocolo **MCP**: puentes estandarizados entre la IA y tus aplicaciones — mail, calendario, planillas, CRM, tu navegador. La IA deja de ser una pestaña aislada que solo ve lo que le pegás, y pasa a leer y operar tus herramientas de verdad.

> 🔧 GEEK DETAIL — TU PRIMER AGENTE PROPIO, SIN UNA LÍNEA DE CÓDIGO
>
> Los tres grandes te dejan armar un "especialista" tuyo en diez minutos: **Gems** en Gemini, **GPTs** en ChatGPT, **Projects** en Claude. La receta es siempre la misma: instrucciones permanentes (cap. 6) + tus documentos de referencia + un nombre. El resultado es un agente reutilizable — y compartible con tu equipo — que arranca cada conversación sabiendo tu contexto. Es el documento maestro convertido en empleado fijo: el que traduce informes técnicos a lenguaje de cliente, el que formatea minutas, el que responde como tu manual de procedimientos.

¿Y la automatización? Es el paso siguiente: sacar al humano del loop de inicio. Con herramientas como **n8n** armás flujos visuales (cajitas conectadas) que corren solos: llega un mail de cliente → la IA lo clasifica → responde lo simple → te deriva lo complejo. La IA como **pieza dentro de un proceso**, no como chat que atendés vos.

La advertencia sin hype. Más autonomía = más superficie de error. Un chatbot que alucina te da un texto malo; un agente equivocado ejecuta acciones equivocadas. Las reglas de oro:

1. **Empezá con tareas de bajo riesgo** y resultado verificable (borradores, clasificación, reportes internos).
2. **Humano en el loop** para todo lo que toque plata, clientes o datos sensibles: el agente propone, vos aprobás.
3. **Ampliá la autonomía con confianza ganada**, no con entusiasmo inicial.

> EN UNA FRASE
>
> El chatbot responde; el agente ejecuta. Y la autonomía se otorga como a un empleado nuevo: de a poco y mirando.

Para profundizar (videos): Agent mode in ChatGPT · ChatGPT Connectors · Build Your Entire AI Workforce · Chatbot con IA para una cafetería en 30 min (n8n) — todos Pilar 4.

CAPÍTULO 09

Vibe coding: programar describiendo

Acá llega la frontera que hace dos años parecía ciencia ficción: **construir software sin saber programar**. Vibe coding es programar describiendo lo que querés en tu idioma — "una página donde mis clientes cargan su pedido y me llega por mail" — mientras la IA escribe, prueba y corrige el código. Vos dirigís; ella teclea.

No es magia: es una **escalera**, y conviene subirla en orden.

Escalón 1 — Prototipos en el chat. Claude y otros generan mini-aplicaciones funcionando dentro de la propia conversación (artifacts): una calculadora para tu negocio, un simulador de precios, un dashboard. Sin instalar nada, sin servidores. El costo de probar una idea de software bajó a cero — y esa es quizás la noticia más importante del capítulo.

> 🔧 GEEK DETAIL — EL CHAT TAMBIÉN DISEÑA
>
> Los artifacts no son solo apps: son un estudio de diseño adentro de Claude. Pedile un dashboard con tus números y lo ves renderizado en la misma conversación. Pegale la captura de un diseño que te guste y decile "clonalo con mis datos". Y el truco pro: pasale al inicio tus **tokens de marca** (paleta, tipografías, reglas) y todo lo que genere sale uniformado con tu identidad. El loop es visual: capturás lo que devolvió → "el gráfico de barras no se lee, probá horizontal" → siguiente versión.

Escalón 2 — Apps reales con servicios armados. Para que una app viva fuera del chat necesita dónde correr, login, base de datos. Antes: un equipo de sistemas. Hoy: servicios como **Firebase** los resuelven configurados por la propia IA en minutos. El no-técnico arma una app completa combinando piezas que otros mantienen.

Escalón 3 — Agentes de programación. Herramientas como **Claude Code** o **Antigravity** son los agentes del capítulo 8 aplicados a programar: trabajan sobre un proyecto entero, crean archivos, ejecutan, testean, corrigen. Acá se construyen herramientas internas serias: el capítulo 8 te dio el concepto; este escalón es su versión más productiva hoy.

> 🔧 GEEK DETAIL — STITCH: DE BOCETO A INTERFAZ
>
> **Stitch** (Google Labs) genera diseños de pantallas — app o web — desde una descripción en texto o incluso la foto de un boceto a mano alzada. Lo exportás a Figma o directo a código. Y el gag mayor: conectado por MCP (cap. 8) a Antigravity, un agente diseña las pantallas en Stitch y otro las construye — diseñador y desarrollador, los dos agentes, coordinados por vos. Antes de encargarle una app a nadie (humano o IA), discutir sobre pantallas visibles ahorra la mitad de las vueltas.

Lo que la IA no pone — lo ponés vos:

- **Claridad de producto.** "Hacé una app para mi negocio" da basura; "una pantalla donde el vendedor carga pedido y stock se descuenta solo" da resultados. Saber qué querés sigue siendo el trabajo difícil.
- **La prueba.** El código que no probaste con datos reales no funciona — todavía no lo sabés.
- **Realismo de alcance.** Tu primera app: una herramienta interna chica que te ahorre algo concreto esta semana. No un producto para vender. El escalón 1 primero; producción (capítulo 10) es otro deporte.

> EN UNA FRASE
>
> El código dejó de ser la barrera; pensar claro qué querés, nunca va a dejar de serlo.

> PROBALO
>
> Pedile a Claude una calculadora interactiva de algo tuyo (margen por producto, presupuesto de un servicio). En 10 minutos tenés software hecho por vos. Ese click mental — "puedo producir" — es el objetivo de todo el manual.

Para profundizar (videos): Claude Artifacts · Firebase con Gemini · Claude 4 × Gemini: workflow híbrido (Pilar 3) · Antigravity tutorial completo (Pilar 4) · Android app en 8 hs hasta la Play Store (Pilar 3).

CAPÍTULO 10

De la demo al sistema: producción real

La parte que nadie te cuenta. Todo lo anterior produce **demos**: cosas que funcionan una vez, con vos mirando, con datos limpios. Un **sistema** es otra cosa: corre todos los días, sin vos, con datos sucios, y alguien depende de que ande. La distancia entre demo y sistema es donde vive el trabajo real — y donde muere el 90% del entusiasmo con la IA.

Qué aparece en el camino que la demo no mostró:

- **Datos reales.** El CSV de prueba era prolijo; el de tu administración tiene celdas vacías, fechas en tres formatos y un "VER NOTA" en la columna de importes. Manejar el caso feo es la mitad del trabajo.
- **Errores.** ¿Qué pasa cuando la API no responde, el archivo no está, el resultado es absurdo? Un sistema necesita respuesta para cada "¿y si...?". La demo, ninguna.
- **Acceso y seguridad.** Quién puede usarlo, qué datos toca, dónde quedan. Con datos de clientes, esto no es opcional.
- **Mantenimiento.** Las herramientas cambian, las APIs se actualizan, el proceso de negocio muta. Sistema sin dueño = sistema muerto en seis meses.

Dónde está el retorno de verdad. La trampa del entusiasmo es construir cosas nuevas y brillantes. El mayor retorno casi siempre está en lo aburrido: **meter IA en el trabajo que ya hacés todas las semanas.** La conciliación bancaria, el reporte mensual, la respuesta a consultas repetidas, el Excel que armás a mano cada lunes. Ahí hay horas concretas, medibles, todas las semanas — sin cambiar ningún hábito de nadie.

Medí, o es un juguete. El KPI de la IA en tu negocio no es la fascinación: son **horas por semana ahorradas** (o errores evitados, o días de demora menos). Anotá cuánto tardaba el proceso antes, medí cuánto tarda ahora, restá el tiempo de supervisión. Si el número no es claramente positivo después del período de aprendizaje, cortá sin culpa: fue un experimento barato, no un fracaso.

> EN UNA FRASE
>
> Demo es lo que corre una vez con vos mirando; sistema es lo que corre solo cada día. Construí demos para aprender — pero cobrale resultados solo a los sistemas.

Para profundizar (videos): Business is hard until you build systems like this · Claude en Excel: modelo financiero de 3 estados · Conciliación bancaria sola · Así entreno a Claude para ser mi analista financiero · 7 maneras en que la IA me ahorra 10 horas a la semana — todos Pilar 5.

CAPÍTULO 11

Harness engineering: el andamiaje alrededor del modelo

Acá subimos un escalón técnico. Si venís aplicando los capítulos anteriores, ya notaste algo: **el mismo modelo, en manos distintas, da resultados de calidad completamente distinta**. La diferencia casi nunca es el modelo — es todo lo que lo rodea. Eso que lo rodea tiene nombre en la jerga: el ***harness*** (literalmente, "arnés").

Cuando usás ChatGPT o Claude no hablás con el modelo pelado: hablás con un producto que lo envuelve — instrucciones de sistema que no ves, herramientas que puede usar, memoria, límites. Cuando construís tus propios sistemas (caps. 8-10), ese envoltorio **lo diseñás vos**. Y ahí aparece la disciplina:

- **Prompt engineering** (cap. 6): qué le pedís.
- **Context engineering** (caps. 3 y 7): qué le mostrás.
- **Harness engineering**: diseñar el **entorno completo de trabajo** del agente — reglas, herramientas, permisos y verificación. Es la capa que convierte un agente impresionante en un agente confiable.

Las cinco piezas de un buen harness:

1. **Las reglas del juego.** Un archivo de instrucciones permanentes del proyecto: qué hace el agente, con qué convenciones, qué no debe tocar jamás. Se escribe una vez, el agente lo lee siempre. Es el documento maestro del cap. 6 elevado a infraestructura: ya no lo pegás vos — vive en el sistema.
2. **Las herramientas.** Qué puede *hacer*: leer archivos, buscar en la web, ejecutar código, mandar mails. Principio de mínimo poder: dale exactamente lo que la tarea necesita, no todo lo que existe. Cada herramienta de más es superficie de error de más.
3. **El contexto.** Qué *ve*: qué carpetas, qué documentos, qué memoria persiste entre sesiones. Volvé al cap. 3: vos decidís qué hay sobre la mesa.
4. **Los permisos.** Qué hace solo y qué requiere tu aprobación. Regla simple: todo lo **irreversible o externo** (borrar, enviar, pagar, publicar) pasa por un humano. Todo lo reversible e interno puede correr solo.
5. **La verificación.** Cómo se comprueba que el resultado está bien **sin confiar en la palabra del agente**: tests que corren, checklists automáticos, recálculo por otra vía. La pieza más olvidada y la más importante.

> DE MI COCINA
>
> Cuando genero un modelo en Excel para un cliente, el harness incluye un paso que recalcula el archivo completo con una herramienta independiente y lo compara contra un espejo del cálculo hecho en otro lenguaje. El agente que armó el Excel nunca se auto-aprueba: la aprobación sale de un cálculo que él no tocó.

Tips para armarlo consciente:

- **Las reglas van a un archivo, no a tu cabeza.** Si repetís una instrucción dos veces, va al archivo de reglas del proyecto.

- **Definí "terminado" de forma verificable.** "Que quede bien" no es un criterio; "que los totales cuadren con la fuente y el checklist pase" sí.
- **Cuando el agente falla dos veces igual, el problema es el harness.** No lo retes en el chat: agregá la regla, la herramienta o el paso de verificación que faltaba. Iterá el andamiaje, no solo el pedido.
- **Auditá el rastro.** Un buen harness deja registro de qué hizo el agente y por qué — sin eso, no hay confianza que escale.

> EN UNA FRASE
>
> El modelo es el motor; el harness es el auto. Nadie gana carreras con un motor suelto arriba de una mesa.

> PROBALO
>
> Elegí un proceso tuyo recurrente y escribile su archivo de reglas (media página: objetivo, convenciones, qué no tocar, cómo se verifica que salió bien). Usalo en cada sesión durante una semana y sumale una regla cada vez que algo falle. Eso es harness engineering, versión casera.

CAPÍTULO 12

Multiagente: dividir para verificar

El paso siguiente al agente único bien harnessado es la pregunta de moda: ¿y si pongo **varios**? La respuesta corta: multiagente vale la pena por exactamente **dos razones**, y si tu caso no es ninguna de las dos, no lo compliques.

Razón 1 — Paralelismo. Un agente tiene una sola ventana de contexto (cap. 3); si la tarea es grande, se satura. Diez agentes con contextos limpios pueden cubrir diez partes a la vez — revisar veinte archivos, investigar cinco fuentes, analizar cada sucursal por separado.

Razón 2 — Verificación independiente. Nadie es buen juez de su propio trabajo, y un modelo tampoco: si le pedís al mismo chat que revise lo que acaba de hacer, revisa con los mismos sesgos con los que lo hizo. Un **segundo agente con contexto limpio** — que no vio el razonamiento del primero, solo el resultado — encuentra lo que el primero no puede ver.

Los patrones que valen la pena conocer:

PATRÓN	CÓMO FUNCIONA	CUÁNDO
Abanico (fan-out)	N agentes en paralelo, cada uno una parte; al final se juntan	Tarea grande y divisible
Generador → Verificador	Uno produce; otro, en limpio, controla contra la fuente	Todo entregable que importe
Adversarial	Un agente cuya única misión es refutar el resultado: buscarle el error, el caso borde, la trampa	Cuando un error es caro; lo que sobrevive al ataque, vale
Panel / votación	3+ agentes opinan independientes; decide la mayoría o un juez	Decisiones ambiguas, evaluaciones
Pipeline	Etapas encadenadas (encontrar → clasificar → verificar), cada ítem avanza solo	Procesos por lotes

¿Y "ultracode"? Es el nombre que recibe el modo exhaustivo de esta idea, que las herramientas de agentes serias ya traen incorporado: para una tarea sustantiva, no un agente sino una **flota orquestada** — abanico para cubrir todo, verificación adversarial para cada hallazgo, síntesis al final. En criollo: máxima confianza a cambio de máximo gasto de cómputo. La cuenta que lo justifica es una sola: **¿el costo de un error supera con claridad el costo de los tokens?** Un modelo financiero que va a un cliente, código que va a producción, una auditoría — sí. Un borrador interno — no.

> DE MI COCINA
>
> Antes de entregar un script o un modelo importante, corro un review con tres agentes en abanico: uno audita los cambios línea por línea, otro valida el resultado contra los datos fuente, y un tercero juega de adversario — su consigna es demostrar que está mal. Solo entrego lo que sobrevive a los tres.

> 🔧 GEEK DETAIL — MULTIAGENTE SIN ESCRIBIR CÓDIGO
>
> No hace falta programar una orquestación para verlo funcionar. **Antigravity** trae un gestor de agentes donde lanzás varios en paralelo y les repartís tareas (uno maqueta, otro programa, otro testea), y **Claude Code** despliega subagentes y flotas de review desde una consigna. Empezá mirando: ver trabajar a una flota bien orquestada enseña más que cualquier diagrama de arquitectura.

Tips para el armado consciente de una base multiagente:

1. **Escalá de a uno.** Primero un agente con buen harness (cap. 11). Multiagente se gana, no se arranca: sumá el segundo cuando el único se sature o necesites verificación independiente.
2. **Contextos limpios, en serio.** El valor del verificador es que NO vio cómo pensó el generador. Si le pasás el razonamiento, hereda los sesgos y el control cruzado se vuelve teatro.
3. **Entre agentes, datos estructurados.** Que se pasen tablas, listas, campos definidos — no prosa. La prosa se degrada en cada mano; la estructura se puede validar.
4. **La orquestación es determinística.** Quién corre, en qué orden, qué pasa si uno falla: eso lo decide un flujo fijo que escribiste vos, no la improvisación del modelo en el momento. Los agentes son creativos *adentro* de cada caja; las flechas

entre cajas son tuyas.

5. **Un agente, una tarea, un criterio de éxito.** "Agente que ayuda con todo" no es un diseño, es un deseo.
6. **Presupuesto consciente.** Multiagente multiplica tokens (cap. 2). Reservalo para donde el error es caro; para el resto, el agente único ya era la respuesta.
7. **Trazabilidad total.** Guardá qué devolvió cada agente. Cuando el resultado final sorprenda — para bien o para mal — vas a querer saber en qué caja nació.
8. **El humano al final del embudo.** La flota filtra, verifica y sintetiza para que tu revisión final sea corta y de alto nivel — no para que no exista. La firma sigue siendo tuya.

> EN UNA FRASE
>
> Multiagente no es más magia — es división del trabajo más control cruzado: las mismas dos ideas con las que funciona cualquier organización seria, aplicadas a agentes.

> PROBALO (SIN INSTALAR NADA)
>
> Patrón generador→verificador a mano. Chat 1: producí un análisis con tus datos. Chat 2, limpio: pegá solo el resultado + la fuente y la consigna "encontrá todos los errores, inconsistencias y afirmaciones sin respaldo". La diferencia entre lo que el chat 1 defendía y lo que el chat 2 encuentra te muestra, en cinco minutos, por qué existe todo este capítulo.

Para profundizar (videos): I Built an Entire App with Codex and 8 AI Agent Employees (Pilar 3) · Build Your Entire AI Workforce (Pilar 4) · How to Use Claude Skills 2.0 (Pilar 3) · AGENTES IA en n8n: curso completo (Pilar 5).

CIERRE

Límites y criterio

Si llegaste hasta acá, tenés el mapa completo: qué es esta tecnología (Parte I), cómo operarla bien (Parte II) y cómo construir con ella (Parte III). Cierro con lo que sostiene todo lo demás.

Los límites, en limpio:

- **Corte de conocimiento.** El modelo se entrenó hasta una fecha; lo de ayer no lo sabe salvo que busque en la web. Preguntale siempre desde cuándo es su información si el dato es sensible al tiempo.
- **Sesgos.** Aprendió de texto humano, con todos sus vicios. En decisiones sobre personas (contratar, evaluar, prestar), el output es una opinión a auditar, jamás un veredicto.
- **Privacidad.** Lo que subís viaja a servidores de terceros. Regla simple: lo que no mandarías por mail a un desconocido, no lo pegues en un chat sin revisar qué herramienta es, qué plan tenés y qué dice su política de datos.
- **Dependencia.** Si un proceso crítico de tu negocio depende de una herramienta, tené claro qué hacés el día que cambie de precio, de dueño o de reglas.

Y el principio que ordena todo:

La IA reduce tiempo. No reemplaza criterio.

Todo output de IA es el borrador de un junior brillante, incansable y ocasionalmente mentiroso. Trabaja gratis, no se ofende cuando lo corregís, y mejora cuanto mejor le explicás las cosas. Pero no firma. **La firma es tuya.** El día que dejás pasar un número sin verificar porque "la IA lo dijo", dejaste de usar la herramienta y empezaste a ser usado por ella.

El criterio — saber qué preguntar, qué verificar, qué aceptar y qué tirar — no se terceriza. Y la buena noticia es que ese criterio ya lo tenés: es el de tu oficio. Este manual solo le agregó una herramienta nueva.

Cómo seguir: elegí UN proceso tuyo que te duela cada semana. Aplicale la escalera: entendé qué herramienta corresponde (cap. 5), armale contexto (cap. 6), dale tus fuentes (cap. 7), y si se repite, automatizalo o construile una herramienta (caps. 8-10). Cuando funcione, medilo (cap. 10), dale andamiaje (cap. 11) y — solo si el error es caro — ponele control cruzado (cap. 12). Después, el siguiente proceso. Los videos del roadmap están para profundizar cada etapa — pero el camino se hace produciendo, no mirando.

ANEXO

El taller geek

Colección de trucos chicos que en el día a día valen más que un capítulo entero. Sin orden de importancia. Probalos de a uno — cada uno paga su lectura la primera vez que lo usás.

01

Captura → clon

¿Viste un dashboard, una web o un informe con un diseño que te gustó? Captura de pantalla, pegala en el chat: "replicá este diseño con mis datos". Los modelos leen imágenes — el diseño de referencia es el mejor prompt de diseño que existe.

02

Tus tokens de marca, en un archivo

Colores, tipografías, reglas de logo, tono — todo en un archivo de texto que le pasás al agente antes de generar cualquier pieza. Cada dashboard, informe o presentación sale uniforme sin repetir instrucciones. Es el harness (cap. 11) aplicado al diseño.

03

Un solo archivo HTML = un entregable

Pedí "un único archivo HTML autocontenido, sin dependencias externas". Se abre en cualquier navegador, viaja por mail o WhatsApp, no necesita servidor, licencia ni instalación. Dashboards, reportes, hasta presentaciones enteras: todo puede ser un archivo.

04

HTML → PDF sin instalar nada

Ese HTML abierto en Chrome + Imprimir + "Guardar como PDF" (jugá con la escala si algo se corta) = documento con diseño profesional, sin tocar Word ni PowerPoint. El circuito completo: datos → HTML con tu marca → PDF entregable.

05

Excel con fórmulas vivas, no valores pegados

Cuando pidas planillas, exigilo: "con fórmulas, no valores". Un Excel de valores muertos no se puede auditar ni actualizar; uno con fórmulas es un modelo. La diferencia entre un informe y una herramienta.

06

La iteración visual

El feedback de diseño más efectivo es una captura: sacale foto a lo que generó, marcá lo que no funciona ("esta tabla no se lee en el celular") y devolvésela. Ojos → captura → ajuste, en loop. Tres vueltas de esto superan a cualquier prompt inicial perfecto.

07

Dark mode y celular salen gratis

"Que se vea bien en el celular y agregale modo oscuro" es una línea en el pedido — y horas de trabajo en el mundo tradicional. Nadie mira dashboards solo en la notebook: pedilo siempre.

08

Diagramas desde texto (Mermaid)

Organigramas, flujos de proceso, líneas de tiempo: pedilos "en formato Mermaid" y obtenés diagramas que se editan como texto — cambiar una caja es cambiar una palabra, no pelearte con las flechas de PowerPoint.

09

Opal: la mini-app que se arma sola

Opal (Google Labs) te deja armar mini-aplicaciones de IA encadenando pasos descritos en tu idioma — "tomá este texto, resumilo, traducilo, formatealo así" — y compartirlas como una app con link. Para flujos chicos y repetitivos del equipo, es el punto medio perfecto entre "un prompt que repito a mano" y "una automatización seria" (cap. 8).

10

Pomelli: tu marca escaneada

Pomelli (Google Labs + DeepMind) escanea tu sitio web y arma el "ADN de tu negocio" — logo, paleta, tipografía, tono — y desde ahí genera campañas y piezas de marketing editables, ya alineadas a tu identidad. Para una PyME sin diseñador es un atajo brutal; para una con diseñador, un generador de borradores. Mismo principio que el truco 2, versión automática.

11

Nano Banana: el editor de imágenes que entiende frases

El modelo de imágenes de Gemini edita fotos con instrucciones en tu idioma: "sacá el fondo", "ponele el producto sobre una mesa de madera", "mismo personaje, ahora de perfil". Piezas para redes, fotos de producto, variaciones de una misma creatividad — sin Photoshop ni curva de aprendizaje.

12

El gag final: tu research hecho podcast

NotebookLM convierte tus fuentes en una conversación de audio entre dos voces que discuten el material (Audio Overview). ¿Rigor académico? No. ¿Repasar un informe denso manejando camino a la reunión? Impagable.

Cara Sur

CONSULTORÍA · 45.86° S

El Manual es un proyecto de **Rodrigo Ortego**, founder de Cara Sur — consultoría financiero-estratégica para PyMEs, Patagonia Argentina. Gratis y sin registro: aprendí gratis de internet, devuelvo por la misma vía. Si encontrás un error o un capítulo flojo, avisá — la próxima versión se hace con eso.

carasur.org · hola@carasur.org · IG @carasurconsultoria · LinkedIn