Vibe Coding: Cómo construir software sin saber programar

Contenido

Nicolás Dubois

- Licenciado en Sistemas
- +18 años en la industria del software — empresas nacionales e internacionales
- Fundador de Pymbu — la primera agencia de inteligencia artificial de Uruguay
- Fundador de Agentes IA en Acción, academia IA. Mentor técnico

¿Alguna vez tuviste una idea de app... y no supiste cómo hacerla realidad?

Vibe Coding: programar describiendo, no escribiendo

1

Idea

Tenés algo en mente que querés construir

2

Descripción en texto

Lo describís en lenguaje natural, como si se lo explicaras a alguien

3

App funcionando

La IA genera el código y vos iterás hasta llegar al resultado

El término lo acuñó Andrej Karpathy en febrero 2025

> "Fully give in to the vibes, embrace exponentials, and forget that the code even exists"

Ex-empleado de OpenAI · Ex director de IA en Tesla · Uno de los investigadores más influyentes del mundo.

Es como tener un desarrollador disponible 24/7 que habla tu idioma

Antes

- Necesitabas saber programar
- O pagar a alguien
- O esperar meses

Ahora

- Describís lo que querés en tu idioma
- La IA lo construye por vos
- Iterás hasta llegar al resultado exacto

¿Cómo hace la IA para "entenderte"?

1 **Entrenada en todo internet**

Leyó libros, código, conversaciones y documentación con miles de millones de ejemplos.

2 **Aprendió patrones**

Reconoce patrones de lenguaje, lógica y código de forma profunda y sofisticada.

3 **Predice, no piensa**

No "piensa" como humano: predice muy bien qué respuesta tiene sentido dado lo que le dijiste.

Como el autocomplete del celular… pero mil veces más sofisticado.

Lo que la IA no hace perfecto (y es importante saber)

⚠️ Alucinaciones

A veces inventa cosas que suenan correctas pero no lo son. Siempre revisá el resultado.

⚠️ Contexto limitado

Si el proyecto crece mucho, puede "olvidar" partes anteriores de la conversación.

⚠️ Iteración es normal

Raramente sale perfecto al primer intento. Eso no es un error, es el proceso.

Esto es completamente normal y manejable — no es un problema bloqueante.

Tu nuevo trabajo: dirigir, no ejecutar

Antes

Ejecutabas tareas técnicas que no sabías hacer, bloqueado por el conocimiento que te faltaba.

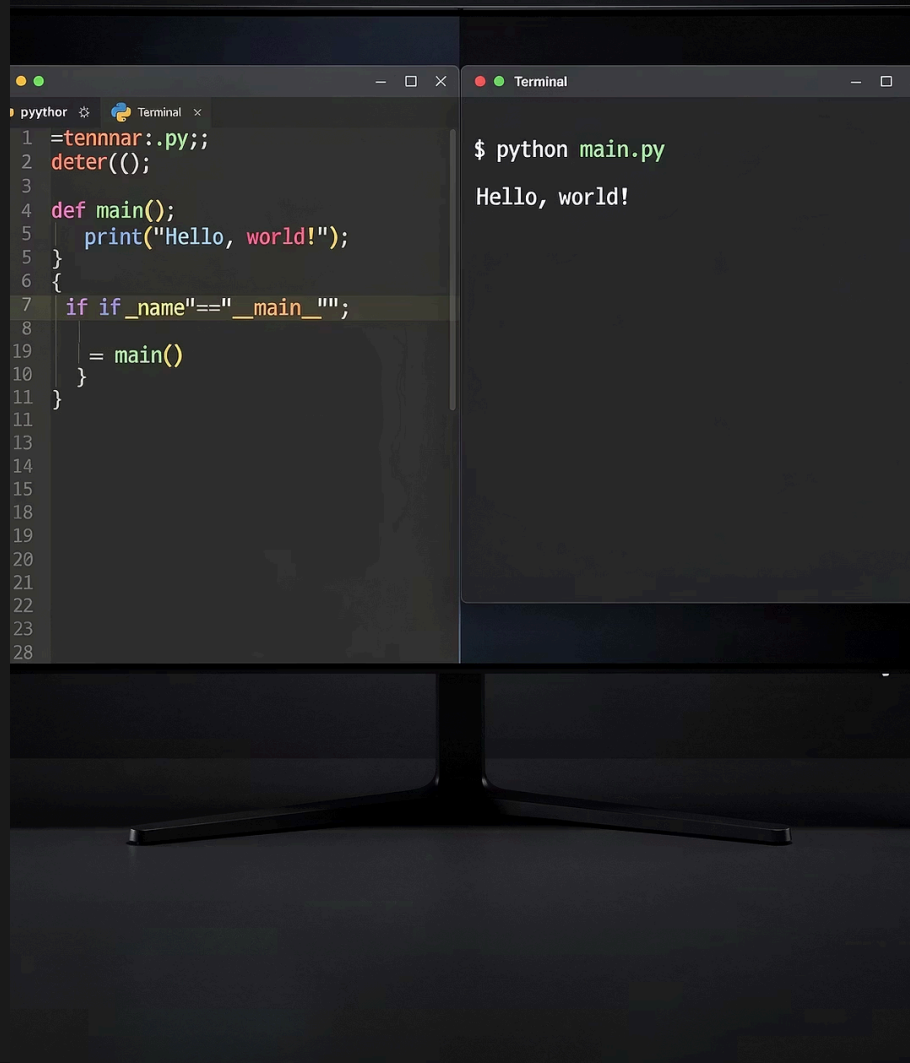
Ahora

Sos el director de un desarrollador muy veloz que necesita dirección clara de tu parte.

Antes de hablar de apps: ¿qué ve realmente una máquina cuando abrís código?

- Un archivo de código no es más que texto plano escrito con reglas muy estrictas
- La máquina no "entiende ideas" como un humano: lee caracteres, palabras y símbolos
- Para ejecutar ese archivo, necesita un programa que lo interprete o lo traduzca
- En Python, ese programa se llama intérprete

> "Primero es texto. Después se convierte en instrucciones."



Ejemplo real: un archivo de Python muy simple

```
nombre = "Nicolás"
print("Hola " + nombre)
```

- La primera línea guarda un dato en memoria
- La segunda línea le dice a Python que muestre un texto en pantalla
- Si ejecutás ese archivo, el resultado sería: Hola Nicolás

```
python hola.py
Hola Nicolás
```

¿Qué hace la máquina paso a paso?

Abre el archivo hola.py

Lee la primera línea

Entiende: "guardá el texto Nicolás en una variable llamada nombre"

Lee la segunda línea

Entiende: "mostrá en pantalla el texto Hola junto con lo que tenga nombre"

Ejecuta esa instrucción

Devuelve el resultado en pantalla

> "La máquina no improvisa: sigue instrucciones exactas, una por una."

Y si el código está mal, la máquina se frena

```
print("Hola"
```

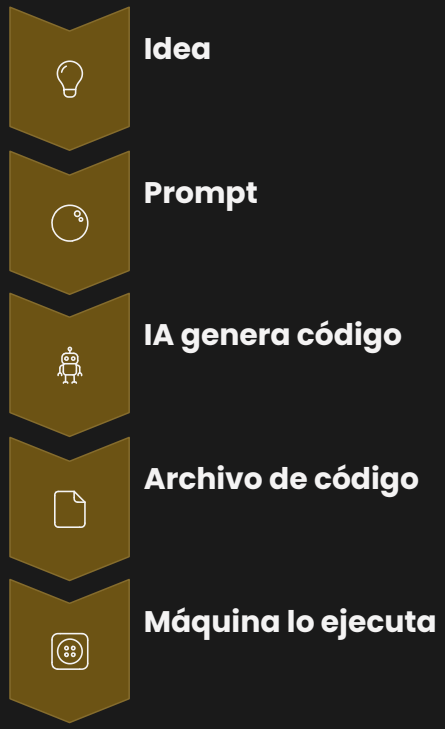
- A vos te puede parecer "casi correcto"
- Pero a la máquina le falta una pieza: cerrar el paréntesis
- Entonces no ejecuta nada y devuelve un error
- El error no es un fracaso: es la forma en que la máquina te marca exactamente dónde no entendió

```
SyntaxError: '(' was never closed
```

> "La computadora no adivina. O lo entiende, o se detiene."

Entonces, ¿qué cambia con vibe coding?

- Antes, vos tenías que escribir estas instrucciones manualmente
- Ahora, la IA las escribe por vos a partir de una descripción en lenguaje natural
- Pero al final del camino, siempre termina existiendo un archivo de código que una máquina tiene que interpretar
- Por eso entender esta base te da muchísimo poder, aunque no seas programador



> "La IA no reemplaza cómo funciona el software. Te simplifica cómo lo creás."

No necesitás saber programar. Sí necesitás hablar el idioma.

Estos conceptos no son para que los domines. Son para que puedas pedirle las cosas correctas a la IA y entender lo que te responde.

Frontend

La interfaz visual que el usuario ve y usa

Backend

La lógica del servidor que procesa datos y reglas

Base de datos

Almacenamiento permanente de información

Framework

Herramientas y estructuras de código prearmadas

GitHub

Repositorio para control de versiones y colaboración

API

El puente que conecta apps y servicios entre sí

Frontend: todo lo que el usuario ve

¿Qué es?

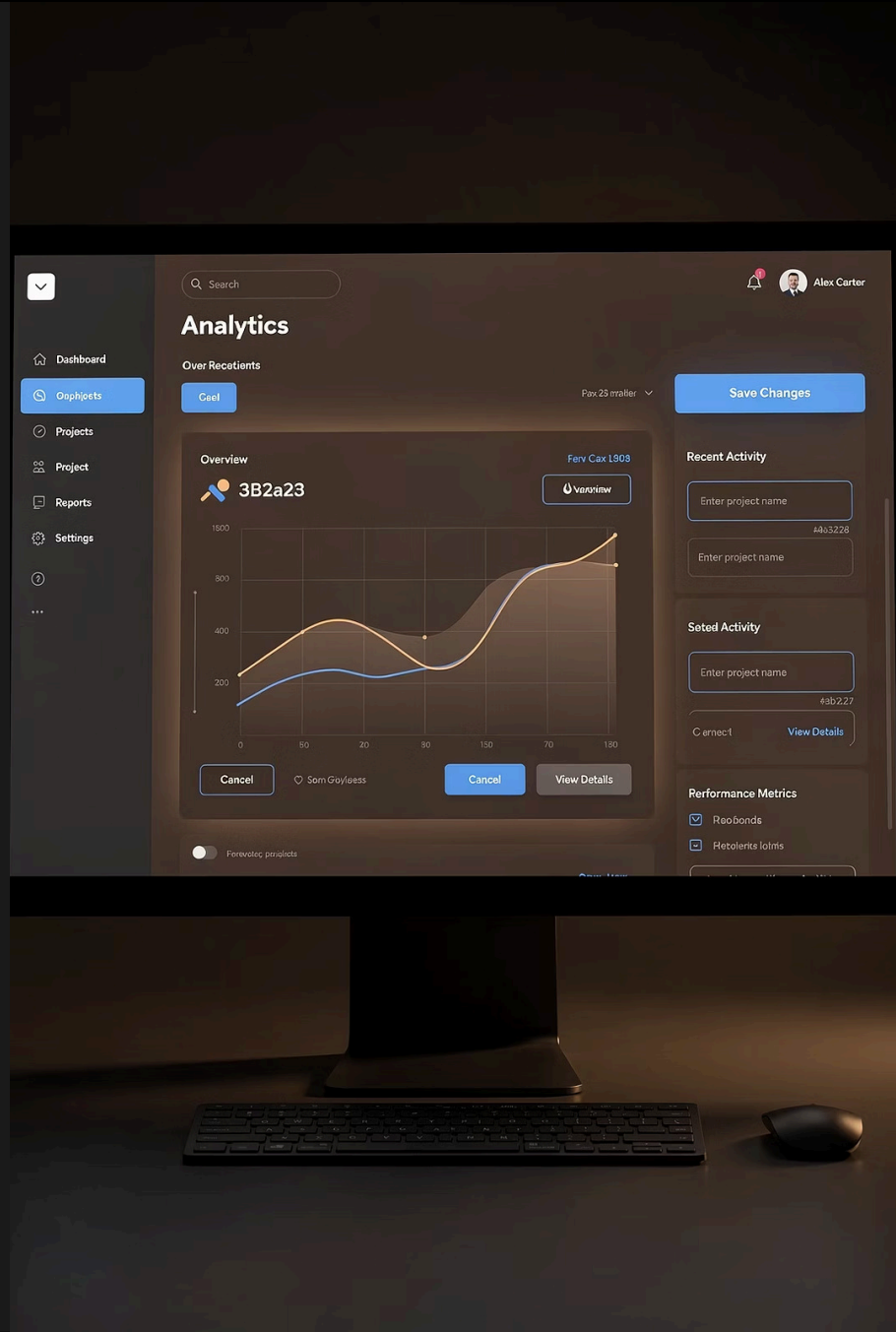
La interfaz: botones, colores, formularios, pantallas. Si lo podés ver y tocar, es frontend.

Analogía

El salón de un restaurante: lo que el cliente experimenta directamente.

Ejemplo

"Hacé que el botón sea azul y esté centrado en la pantalla" → pedido de frontend.



Backend: todo lo que pasa por detrás

¿Qué es?

La lógica y las reglas que el usuario no ve. Procesa información y conecta con la base de datos.

Analogía

La cocina del restaurante: donde se prepara todo lo que el cliente recibe.

Ejemplo

"Cuando el usuario se registra, mandá un mail de bienvenida automático" → eso es backend.

Base de datos: donde vive la información

¿Qué es?

El lugar donde se guardan todos los datos de tu app de forma permanente. Sin base de datos, todo desaparece cuando cerrás la app.

Analogía: la libreta de pedidos del restaurante.

Ejemplos de datos guardados

- Usuarios registrados
- Productos del catálogo
- Historial de compras
- Mensajes enviados

ID	Nombre	Email	Fecha de registro
001	Ana García	ana@mail.com	2025-01-15
002	Luis Pérez	luis@mail.com	2025-01-18

API: el puente entre apps

¿Qué es?

Una API es una puerta de entrada que permite que dos programas se comuniquen entre sí. No ves la API, pero la usás todo el tiempo.

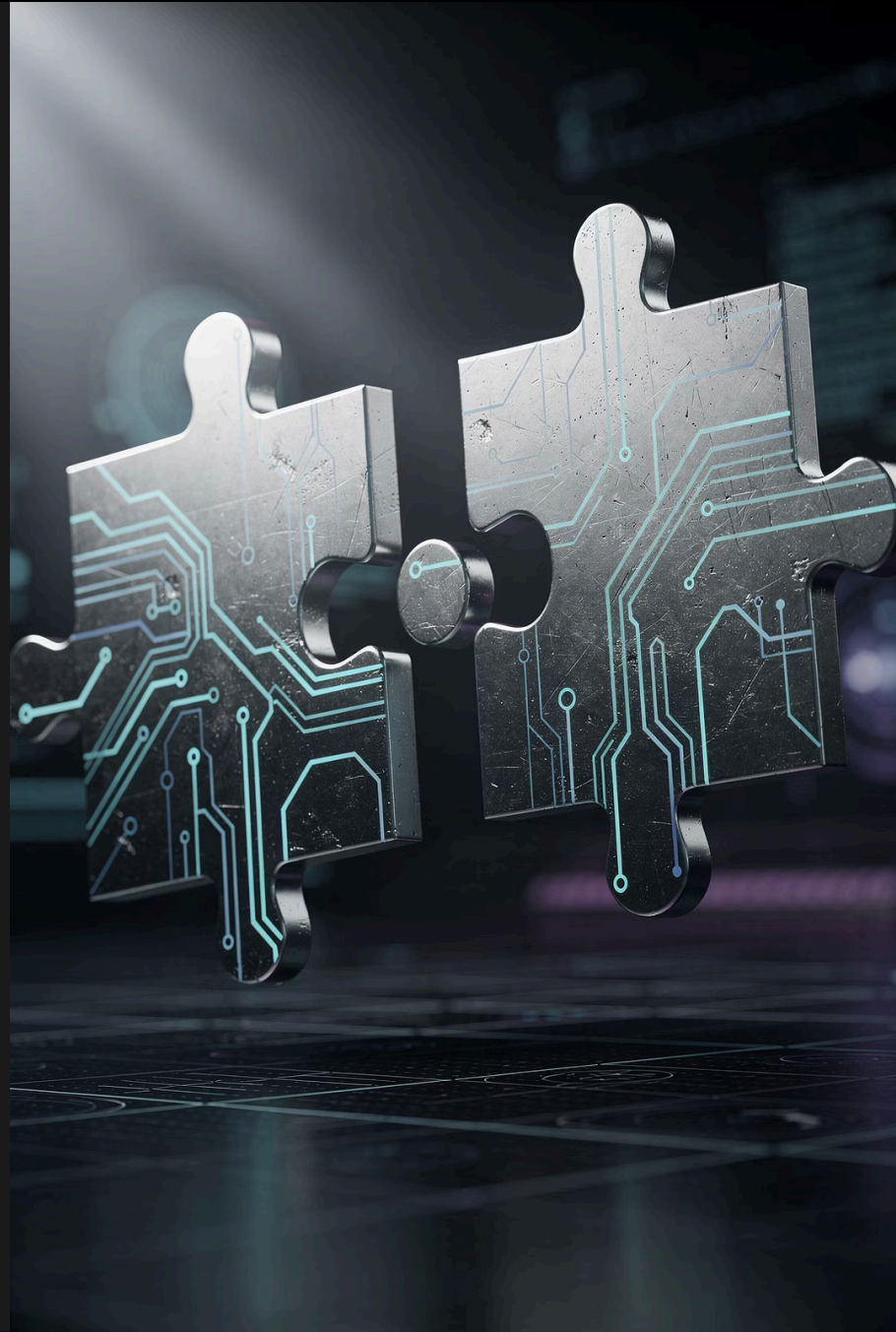
Analogía

El mozo de un restaurante: vos pedís (frontend), él lleva el pedido a la cocina (backend) y trae el resultado. La API es ese mozo.

Ejemplo real

Cuando tu app muestra el clima, no calcula nada: le pregunta a la API de OpenWeather y muestra lo que recibe.

> "Conectar tu app con la IA de Claude, con pagos, con mapas o con cualquier servicio externo: todo pasa por una API."



Frameworks: la cocina pre-equipada

¿Qué son?

Conjuntos de herramientas y estructuras ya armadas para no empezar de cero. Alguien ya resolvió los problemas comunes.

Ejemplos clave

React (interfaces) · **Next.js** (apps web completas) · **Node.js** (backend)

Por qué importa

La IA siempre va a elegir uno. Si sabés de qué habla, podés guiarla mejor.

GitHub: Google Drive para tu código

💾 Guardá versiones

Cada cambio queda registrado con fecha. Podés volver atrás en cualquier momento.

🚀 Publicá tu app

Desplegá tu proyecto para que el mundo lo use con un solo comando.

🤝 Colaborá

Trabajá con otros desarrolladores o IAs sobre el mismo proyecto.

> Pensalo como Google Drive, pero diseñado específicamente para código.

El mapa completo

Usuario

Interacción con la interfaz

Frontend

Presentación y lógica cliente

Backend

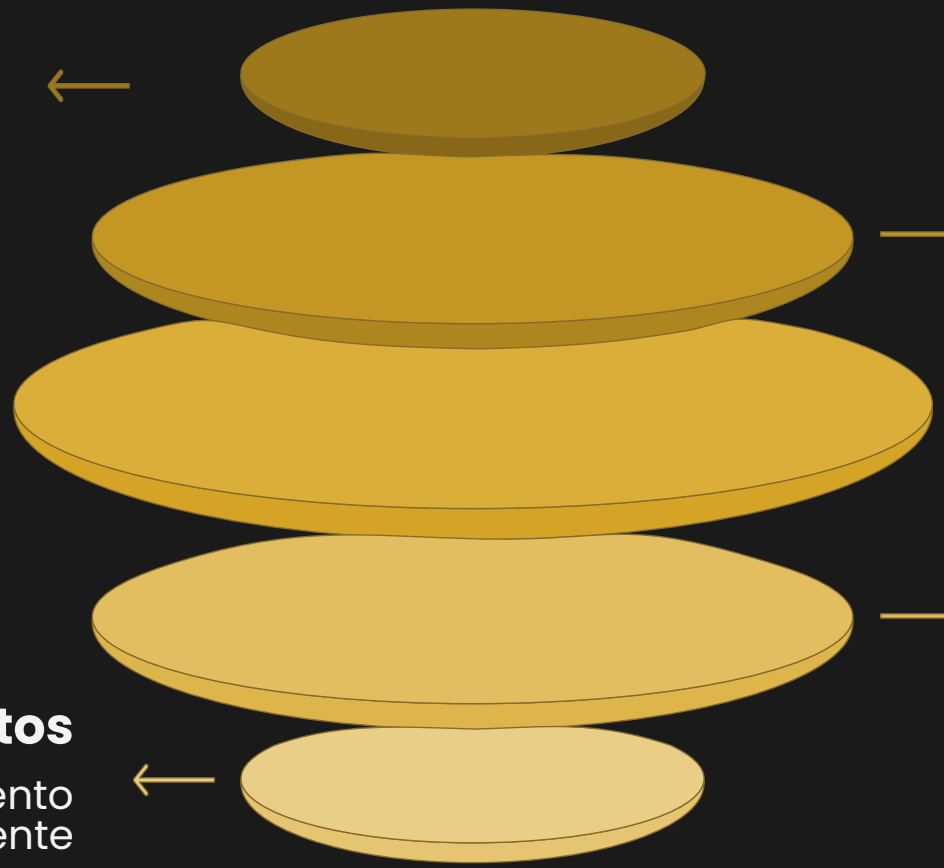
Procesos y APIs bidireccional

API

Puente entre apps y servicios externos

Base de datos

Almacenamiento persistente



Este es el flujo completo de cualquier aplicación web. Cada capa tiene su rol, y GitHub envuelve todo el sistema como capa de control de versiones.

No todas las herramientas de vibe coding son iguales

🏗️ App Builders

Describís y listo. Rápido. Sin instalar nada. Ideales para prototipos y validar ideas rápidamente.

- Bolt · Lovable · Replit · v0

⚙️ Agentic IDEs

Vos dirigís, la IA ejecuta. Más control y potencia para proyectos reales y complejos.

- Claude Code · Cursor · Google Antigravity

App Builders: llave en mano

Sin instalar nada

Entrás desde el navegador y empezás a construir de inmediato.

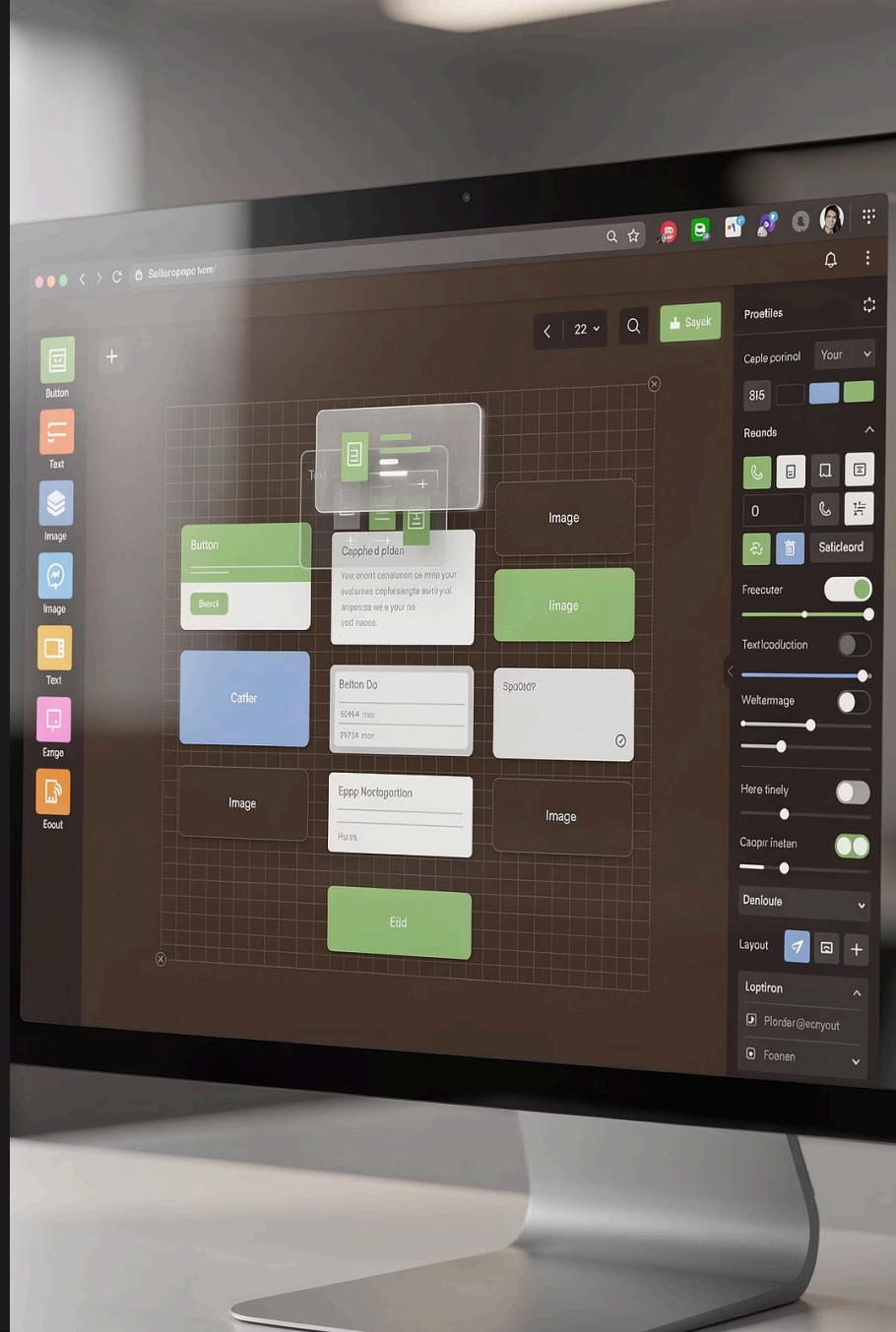
Generación automática

Describís la app y generan todo automáticamente . Ideales para prototipos rápidos.

Límites claros

Tienen restricciones cuando el proyecto crece o se complejiza demasiado.

Analogía: contratar una constructora llave en mano. Rápido y fácil, pero no elegís cada material.



Agentic IDEs: vos dirigís, la IA ejecuta

Entornos completos

Trabajan sobre tu proyecto real, archivo por archivo, con IA totalmente integrada.

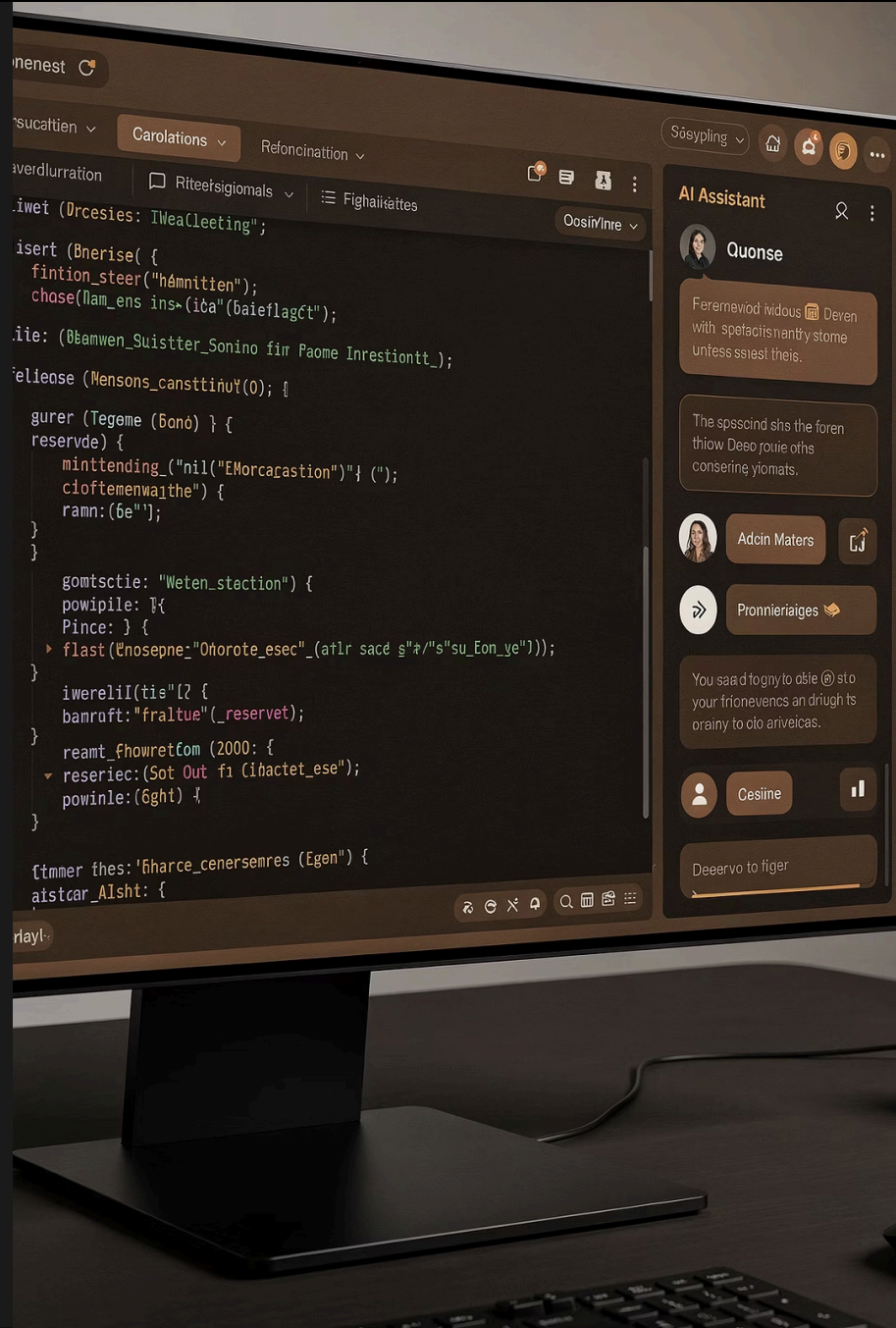
Agentes autónomos

Planifican, escriben, detectan errores y corrigen solos sin intervención constante.

Máximo control

Más configuración inicial, pero mucho más potencia para proyectos serios.

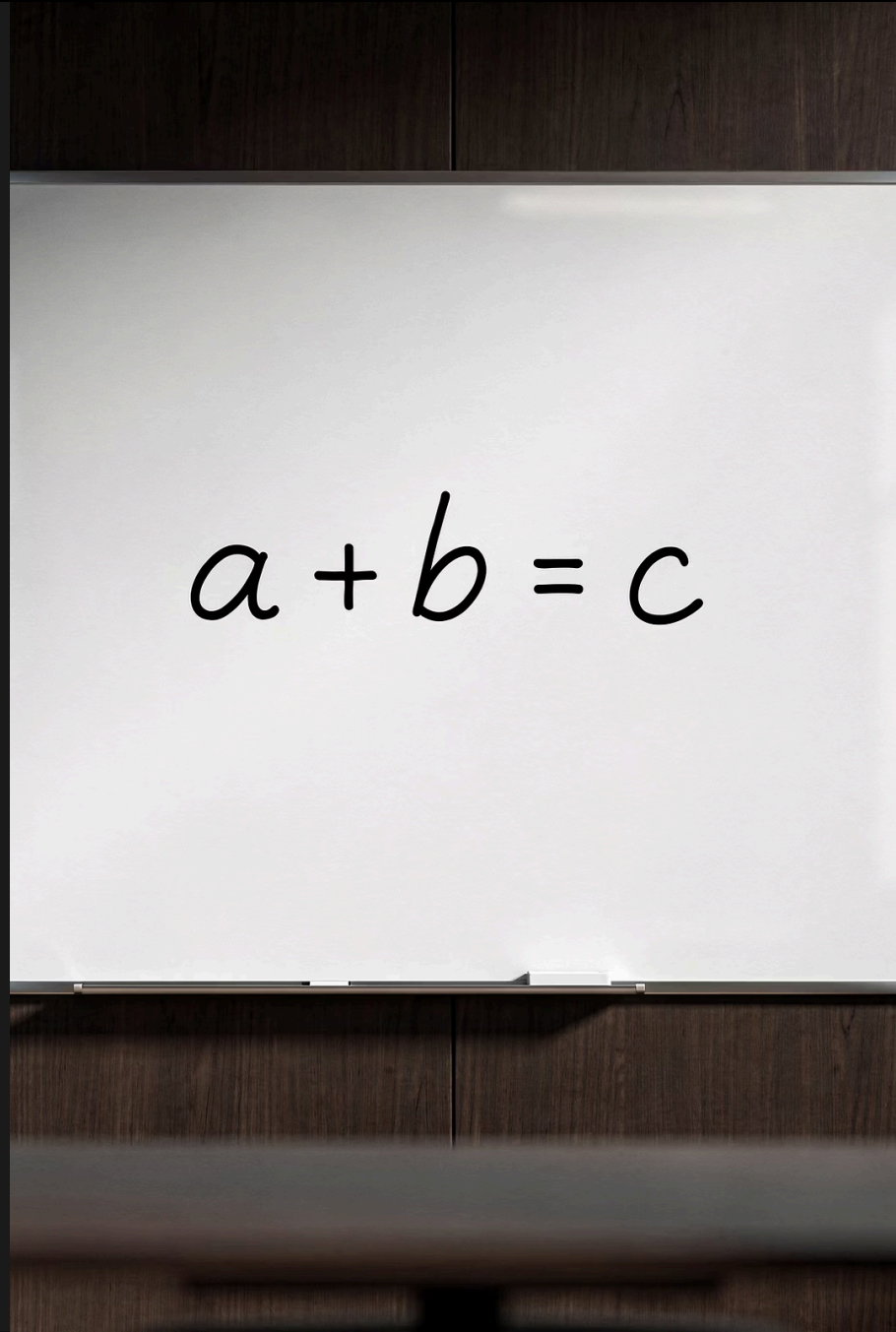
Hoy vamos a trabajar con **Claude Code** — la herramienta más potente de esta categoría.



La habilidad más importante: saber pedir

La IA es tan buena como las instrucciones que recibe. Un prompt vago da resultados vagos. Un prompt preciso da resultados precisos.

Prompt claro Contexto + objetivo + restricciones + resultado esperado	=
Resultado útil Exactamente lo que necesitabas, en menos iteraciones	



Ejemplo 1: crear un formulario

❌ PROMPT VAGO

> "Haceme un formulario"

Resultado: un formulario genérico que probablemente no sirva para nada concreto.

✅ PROMPT PRECISO

> "Creá un formulario de contacto con tres campos: nombre, email y mensaje. Cuando el usuario lo envíe, mostrá un cartel de confirmación en verde que diga 'Gracias, te respondemos en 24hs' y limpiá todos los campos automáticamente."

Ejemplo 2: pedir un cambio

❌ PROMPT VAGO

> "No me gusta cómo se ve"

La IA no sabe qué cambiar ni en qué dirección ir.

✅ PROMPT PRECISO

"El botón de enviar está demasiado pegado al campo de mensaje. Agregá 24px de espacio entre ambos, cambiá el color del botón a azul oscuro (#1a3c6e) y hacé el texto del botón más grande, de 14px a 16px."

No existe el prompt perfecto. Existe la iteración.

1. Escribís un prompt

2. Revisás el resultado

4. Escribís el siguiente prompt

3. Identificás qué ajustar

> "Cada vuelta del ciclo te acerca más a lo que querés"

Cómo construir un buen prompt

🗂️ Contexto

Qué tenés hasta ahora, qué es el proyecto, cuál es el estado actual.

🎯 Objetivo

Qué querés lograr con este prompt específico, en esta iteración.

🚧 Restricciones

Qué no debe cambiar, qué no debe tocar, qué debe preservar.

✅ Resultado esperado

Cómo sabés que está bien hecho. El criterio de éxito concreto.

El error más común: pedir todo junto

❌ UN SOLO PROMPT ENORME

Diseño + lógica + base de datos + validaciones + emails automáticos... todo en un solo mensaje.

Resultado: confusión, errores en cascada, difícil de corregir.

✅ CINCO PROMPTS PEQUEÑOS

Uno por cosa, en orden. Cada paso construye sobre el anterior de forma controlada.

Resultado: progreso claro, errores aislados, fácil de iterar.

> "Una idea. Un prompt. Iterá de a pasos."

Antes de escribir el primer prompt: definí qué vas a construir

1 **Describí tu app en lenguaje natural**

2 o 3 párrafos explicando qué hace tu app, sin tecnicismos.

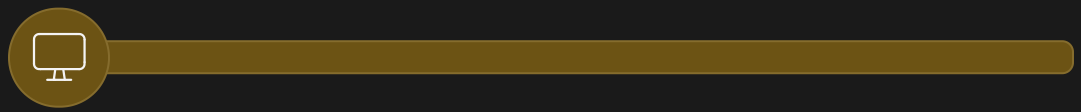
2 **Listá las pantallas que necesita**

Cada sección o vista que el usuario va a ver y usar.

3 **Definí quién la usa y para qué**

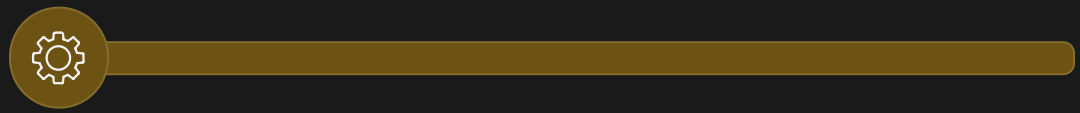
Esto es tu especificación (spec). Claude Code lo usará como contexto.

Siempre de afuera hacia adentro



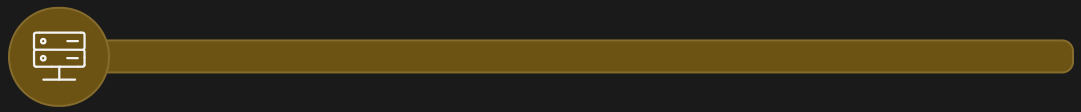
1. El diseño y la interfaz (frontend)

Primero lo que el usuario ve. Validá la experiencia antes de construir la lógica.



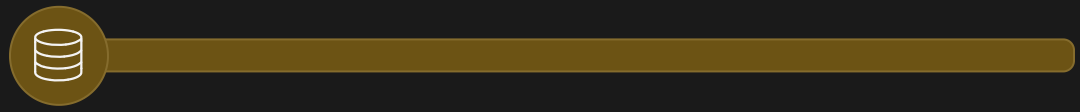
2. La lógica de negocio (backend)

Las reglas, procesos y conexiones que hacen funcionar la app.



3. Los datos (base de datos)

Definir la estructura de almacenamiento y persistencia de información.



4. Conectar todo y probar

Integración final y verificación end-to-end del sistema completo.

Tres reglas para no perder el trabajo

💾 Guardá antes de cambiar

Antes de un cambio grande, siempre guardá el estado actual en GitHub.

🚧 Describí qué NO cambiar

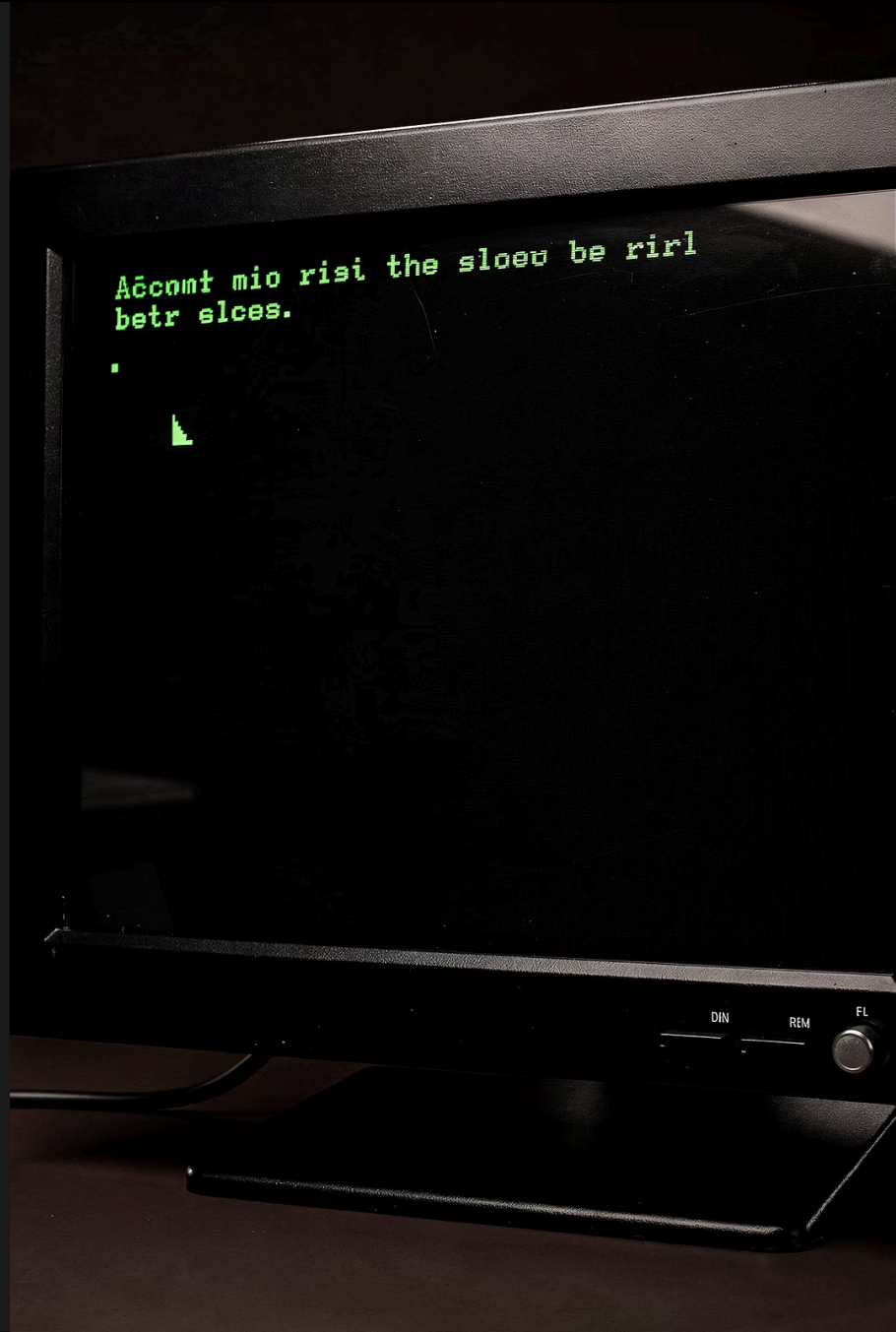
Especificá qué no debe tocar, no solo qué sí. Esto protege lo que ya funciona.

↩️ Revertí si algo se rompe

Si algo falla, pedile a Claude Code que revierta el último cambio inmediatamente.

Veámoslo en acción

Vamos a construir una app en vivo con Claude Code desde cero.
Prestá atención al proceso, no solo al resultado.



Qué mirar durante la demo

01

El primer prompt en Claude Code

Cómo se estructura y qué información incluye para arrancar bien.

02

Cómo planifica antes de ejecutar

Claude Code piensa el plan antes de escribir una sola línea de código.

03

Cómo se itera

Qué pasa cuando el resultado no es exacto y cómo se corrige.

04

Cambios puntuales sin romper nada

Cómo pedir modificaciones preservando lo que ya funciona.

05

Cómo se guarda el progreso

El flujo de guardado en GitHub durante el desarrollo.

De acá para adelante: tu roadmap con Claude Code

Cada nivel es una sesión. No necesitás dominar todo hoy — el objetivo es tener el mapa completo.

HOY

Nivel 1 — Primeras apps ✓

→ Instalar y configurar Claude Code

Entender cómo funciona la terminal y el flujo básico de trabajo.

→ Primeros prompts

Crear archivos, componentes y páginas desde cero con instrucciones en lenguaje natural.

→ Primera app end-to-end

Una aplicación completa funcionando, construida por vos desde el primer prompt.

Nivel 2 — Darle memoria y contexto a tu proyecto

CLAUDE.md

Archivo donde le explicás a Claude quién sos, qué estás construyendo y cómo debe trabajar. Es la **memoria persistente** del proyecto — Claude lo lee cada vez que abre el proyecto.

Reglas

Qué tecnologías usar, cómo nombrar archivos, qué no tocar nunca, cómo estructurar el código.

SDD — Spec Driven Development

Escribir el plano completo del proyecto *antes* de escribir una sola línea de código. La IA trabaja mejor con un mapa claro.

Nivel 3 — Potenciar y conectar Claude Code

Skills

Archivos de instrucciones para tareas recurrentes en tu proyecto. Ejemplo: "cada vez que creés un componente nuevo, seguí esta estructura específica". Claude las aprende y las reutiliza automáticamente.

MCP — Model Context Protocol

Conectar Claude Code con el mundo real. Claude deja de vivir solo en tu editor y empieza a actuar dentro de tus herramientas.

- GitHub · Notion · Base de datos
- APIs externas · Google Drive · Slack

Nivel 4 — Claude Code en modo autónomo

Modo agente

Claude Code ejecuta tareas largas y complejas sin que estés mirando cada paso.

Multi-agent

Varios agentes trabajando en paralelo en distintas partes del proyecto al mismo tiempo.

Hooks

Automatizaciones que se disparan antes o después de cada acción del agente.

Equipos mixtos

Trabajar junto a desarrolladores técnicos usando Claude Code como puente de comunicación.

Ya no sos solo un usuario de tecnología

Antes

Usabas herramientas que otros construyeron. Eras espectador de lo que otros creaban.

Ahora

Podés construir las tuyas. Sos creador, no solo consumidor de tecnología.

Las mejores aplicaciones de IA para 2026

🤖 Asistentes Generales

Claude · ChatGPT · Perplexity · Gemini

💻 Desarrollo / Programación

Cursor · Lovable · Replit · Base44

🎬 Creación de Contenido

Manus AI · HeyGen · Synthesia · Descript · Opus Clip · Beehiiv

⚡ Productividad

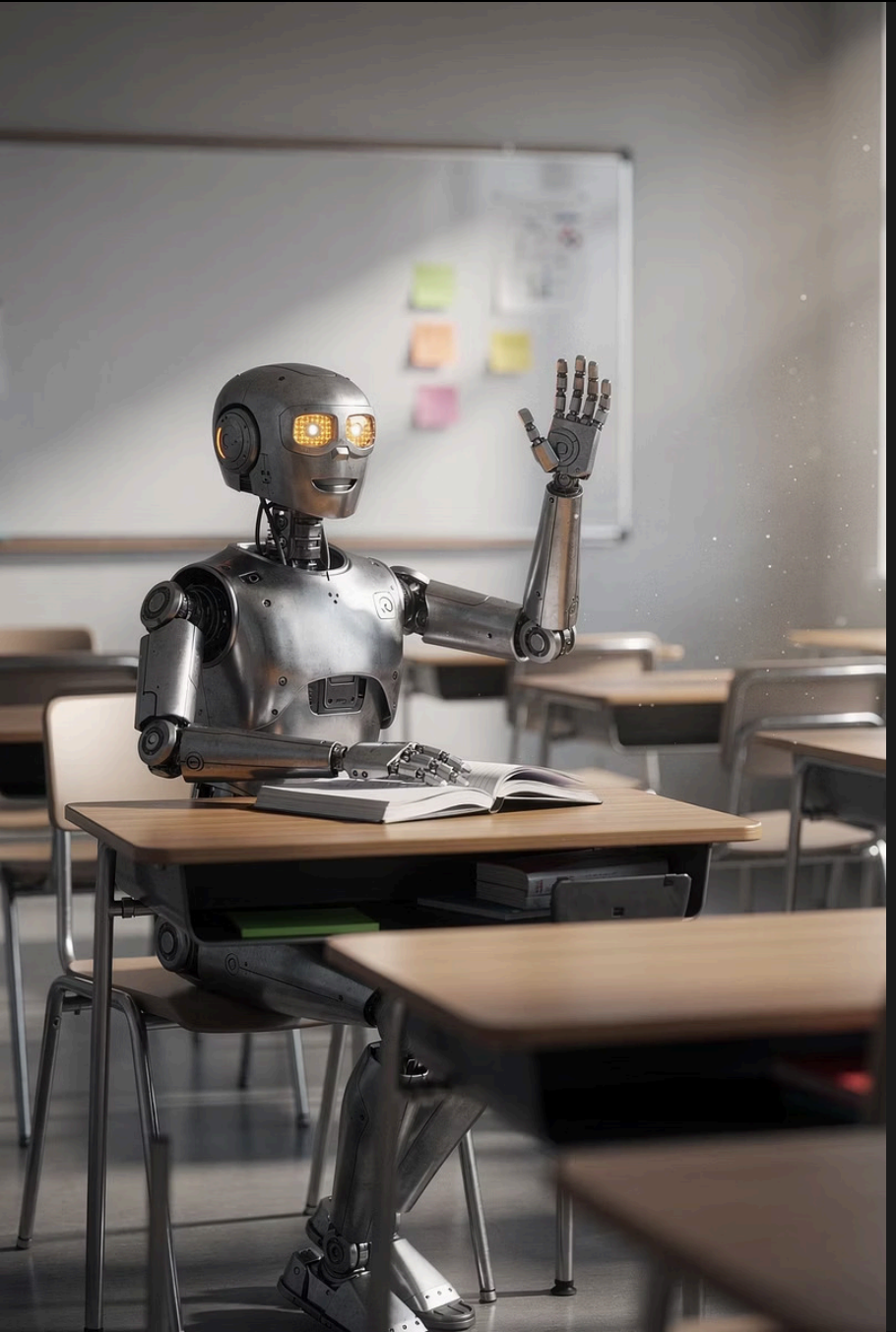
Grammarly · NotebookLM · Otter AI · Gamma · Granola · SuperHuman · Wispr Flow

🎨 Creatividad

ElevenLabs · Suno · Midjourney · Runway · Kling · Pika Labs · · Figma · Canva · Google Veo

🔗 Automatización e Integración

Softr · n8n · Zapier · Lindy AI · Claude Code · Chatbase · Gemini · Notion AI · Apify · Clay



¿Preguntas?

Este es el momento para resolver dudas, compartir ideas y explorar casos de uso concretos.

Agentes IA en Acción

¿Listo para crear?

Nos vemos en la próxima sesión.

@Lic.Nicolasdubois

🌐 pymbu.com