MUESTRA GRATUITA



> method --version 1.5

NotFound

De Cero a Producción con Claude Code

Método universal: **9 fases, cualquier stack, cualquier tipo de web**. Ejemplo trabajado con Next.js + Supabase + Vercel. Dominio del caso histórico: 6,95 EUR; excluye CLI y otros costes (cap. 03).

Edición: Claude Code · v1.6 · septiembre 2026
Autor: NotFound
Idioma: español
Stack ejemplo: Next.js + Supabase + Vercel
Audiencia: desde CERO absoluto — el Nivel 0 te enseña hasta a abrir la terminal

> parte --nivel0 --fundamentos

NIVEL 0

Para cero absoluto

Nunca has abierto una terminal ni escrito una línea de código. Este es tu punto de entrada: qué es cada cosa (en cristiano), tu primera web online con un recorrido guiado, y cómo trabajar con la IA sin saber programar todavía.

⏱ Recorrido guiado: lectura y práctica a tu ritmo

N0 Fundamentos para cero absoluto FACIL

> Este capítulo no asume NADA. Ni que sabes qué es una terminal, ni que has visto código en tu vida. Si ya sabes abrir una terminal y ejecutar un comando, sáltatelo y ve directo a la Parte 1.

¿POR QUÉ TE IMPORTA ESTO? El resto del libro usa palabras como «terminal», «deploy» o «repositorio» sin pararse a explicarlas cada vez. Este capítulo te da ese vocabulario y esa base **una sola vez**, para que nada de lo que venga después te suene a chino. Son 30 minutos que desbloquean todo lo demás.

N0.1 Qué es la terminal (y por qué no da miedo)

La terminal es una ventana donde le escribes **órdenes de texto** a tu ordenador, una por línea. Escribes la orden, pulsas `Enter`, y el ordenador responde con texto. Eso es todo.

Analogía: es como el WhatsApp de tu ordenador. Tú escribes un mensaje (el *comando*), le das a enviar (`Enter`) y él te contesta. No hay botones ni menús: solo una conversación por escrito.

Cómo abrirla

- **Windows:** pulsa la tecla `Win`, escribe `powershell` y pulsa `Enter`. Se abre una ventana azul o negra que dice «Windows PowerShell». Ya está: eso es la terminal.
- **macOS:** pulsa `Cmd+Espacio`, escribe `terminal` y pulsa `Enter`.

Verás algo así (los nombres cambiarán en tu ordenador):

```
Windows PowerShell

Windows PowerShell
Copyright (C) Microsoft Corporation.
PS C:\Users\ana> node --version
v22.11.0
PS C:\Users\ana>
```

TERMINAL Así se ve una terminal. La línea que acaba en `>` es el «prompt»: ahí escribes tú. Lo de debajo es la respuesta del ordenador.

Anatomía de lo que ves:

1. **El prompt** (`PS C:\Users\ana>`): el ordenador te dice «estoy listo, dime». También te dice *en qué carpeta estás* (aquí, la carpeta de la usuaria «ana»).
2. **El comando** (`node --version`): la orden que tú escribes. Esta pregunta «¿qué versión de Node tienes?».

3. **La respuesta** (`v22.11.0`): lo que contesta el ordenador. Después vuelve a aparecer el prompt: listo para la siguiente orden.

Pruébalo ahora (2 minutos)

1. Abre la terminal como se explica arriba.
2. Escribe `node --version` y pulsa `Enter`.
3. Si responde algo como `v22.11.0` : ya tienes Node instalado. Si responde «no se reconoce» o «command not found»: **no pasa nada**, significa que aún no está instalado. Lo instalarás en el capítulo N1.

Los 3 únicos comandos de supervivencia

COMANDO	QUÉ HACE	ANALOGÍA
`cd nombre-carpeta`	Entra en esa carpeta («*c*hange *d*irectory»).	Hacer doble clic en una carpeta.
`dir` (Windows) / `ls` (Mac)	Lista lo que hay en la carpeta actual.	Abrir la carpeta y mirar dentro.
Flecha `↑`	Recupera el último comando escrito, para no reescribirlo.	Reenviar tu último mensaje.

> **LA TERMINAL NO MUERDE** Nada ocurre hasta que pulsas `Enter`, y cerrar la ventana no rompe nada. Dos reglas de higiene: (1) **no pegues comandos de internet que no entiendas** — a la IA siempre puedes preguntarle antes: «¿qué hace este comando, en lenguaje llano?»; (2) si un comando se queda «colgado», `Ctrl+C` lo corta. Eso es todo lo que necesitas para no tener miedo.

N0.2 Las piezas del puzle, de una en una

Para construir una web moderna intervienen unas pocas piezas. Una frase y una analogía por pieza; el vocabulario completo viene justo después.

PIEZA	QUÉ ES (1 FRASE)	ANALOGÍA
Editor de código (VS Code)	El programa donde se abren y editan los archivos de código.	El Word del código.
Node.js	El motor que permite ejecutar JavaScript (el lenguaje de la web) en tu ordenador.	El motor del coche: no lo ves, pero sin él nada arranca.
npm	El instalador que descarga piezas de código ya hechas por otros (viene incluido con Node).	La tienda de piezas: pides «quiero X» y te lo trae montado.
Framework (Next.js)	Un esqueleto de web con las decisiones difíciles ya tomadas; tú construyes encima.	Casa prefabricada: cimientos y tuberías hechos, tú eliges habitaciones y decoración.
Librería	Un paquete de código que resuelve UNA cosa concreta (enviar emails, validar formularios…).	Un mueble de IKEA: lo encajas en tu casa en vez de fabricarlo tú.
Deploy	Subir tu web a internet para que cualquiera pueda verla.	La mudanza + inauguración: de tu maqueta privada al local abierto al público.
Claude Code	Un programador experto en forma de programa: vive en tu terminal, lee tu proyecto y escribe el código que le pidas en tu idioma.	Un empleado técnico al que diriges: tú decides QUÉ, él resuelve el CÓMO.

CÓMO ENCAJAN Tú hablas con **Claude Code** en la terminal. Él escribe archivos de código en tu carpeta del proyecto (que puedes mirar con **VS Code**). Ese código usa un **framework** y **librerías** instaladas con **npm**, y corre gracias a **Node**. Cuando funciona en tu ordenador, haces **deploy** y pasa a estar en internet. Eso es TODO el proceso, de principio a fin. **Nota:** para el capítulo N1 no necesitas instalar VS Code — Claude Code crea y edita los archivos por ti. Lo instalarás en la Parte 1, cuando empieces a querer ver el código con tus propios ojos.

N0.3 Vocabulario mínimo: las 20 palabras

Estas 20 palabras cubren el 90% de la jerga del libro. No hace falta memorizarlas: léelas una vez y vuelve aquí cuando alguna te suene rara. (El glosario completo, con términos más técnicos, está en los Anexos A y B.)

PALABRA	QUÉ SIGNIFICA (EN CRISTIANO)
1. Terminal	Ventana donde escribes órdenes de texto a tu ordenador.
2. Comando	Una de esas órdenes. Se ejecuta al pulsar Enter.
3. Directorio	Sinónimo técnico de «carpeta». El «directorio del proyecto» es la carpeta donde vive TODO tu proyecto.
4. Editor de código	Programa para abrir y editar archivos de código. En este libro: VS Code.
5. Node.js	El motor que ejecuta JavaScript en tu ordenador.
6. npm	El instalador de librerías. Viene con Node.
7. Dependencia / librería	Código de otros que tu proyecto usa. Se instala con npm y se apunta en una lista (package.json).
8. Framework	Esqueleto de aplicación con decisiones ya tomadas. Aquí: Next.js.
9. Servidor de desarrollo	Tu web funcionando SOLO en tu ordenador mientras la construyes. Se arranca con `npm run dev`.
10. localhost	La dirección de tu ordenador para verse a sí mismo. `http://localhost:3000` = «mi web local, puerta 3000».
11. Build	Empaquetar tu web para publicarla. Si el build falla, hay un error que arreglar antes de subir nada.
12. Deploy	Subir la web empaquetada a internet.
13. Producción	La versión pública y real de tu web, la que ve la gente. Lo contrario de «local».
14. Git	El historial de versiones del proyecto: guarda «fotos» de tu código a las que puedes volver.
15. Repositorio (repo)	Tu proyecto tal y como lo guarda Git (en tu ordenador o en GitHub).
16. Commit	Una de esas «fotos», con un mensaje que dice qué cambió. Se hace cada vez que algo funciona.
17. GitHub	Web donde guardas una copia de tu repositorio en la nube (y desde donde se despliega).
18. Variable de entorno	Dato secreto o configurable (claves, contraseñas) que se guarda FUERA del código, en un archivo aparte (.env).
19. API	La «ventanilla» por la que dos programas se piden cosas: tu web le pide al servidor «guarda este mensaje» a través de una API.
20. Base de datos	Donde tu web guarda información que debe sobrevivir: usuarios, mensajes, pedidos. Aquí: Supabase.

N0.4 Cómo leer un mensaje de error sin entrar en pánico

Vas a ver errores. Muchos. Es **normal**: los ve también un senior con 15 años de experiencia, todos los días. La diferencia es que el senior sabe que un error no es un examen suspendido: es un **GPS**. Te está diciendo exactamente dónde está el problema.

Un error típico en la terminal se ve así:

Terminal

```
PS C:\Users\ana\mi-primera-web> npm run dev
Error: Cannot find module 'next'
Require stack:
- C:\Users\ana\mi-primera-web\server.js
    at Function.Module._resolveFilename (node:internal/modules/cjs/loader:1145:15)
    at Function.Module._load (node:internal/modules/cjs/loader:986:27)
    ...
```

ERROR Un error real. La primera línea es la importante; el resto es el «camino interno» que casi nunca necesitas leer.

Anatomía, de arriba abajo:

1. **La primera línea = QUÉ falló.** `Cannot find module 'next'` significa «no encuentro la pieza llamada next». Aunque no sepas arreglarlo, ya sabes de qué va.
2. **La ruta de archivo = DÓNDE.** `C:\...\server.js` te dice en qué archivo estalló.
3. **Las líneas «at ...» = el camino interno.** Son las tripas de Node. No las leas: no están escritas para ti.

Qué hacer con un error (siempre lo mismo)

1. **No lo interpretes: cópialo.** Selecciona TODO el bloque del error, desde la primera línea hasta la última.
2. **Pégaselo a la IA con este prompt** (literal, guárdatelo):

```
Me sale este error al ejecutar [EL COMANDO O LA ACCIÓN].
Te lo pego entero. Explícame en lenguaje llano (1) qué significa,
(2) cuál es la causa más probable, y (3) arréglalo con el cambio
más pequeño posible. No toques nada que no tenga que ver.

[PEGA AQUÍ EL ERROR COMPLETO]
```

ERROR VS. AVISO No todo lo rojo/amarillo es un incendio. Un **warning** (aviso) no detiene el programa: apúntalo y pregunta a la IA si afecta a seguridad, datos o publicación antes de dejarlo para luego («¿este aviso puede romper algo en producción o exponer datos?»). Un **error** sí detiene: algo NO va a funcionar hasta que se arregle. Ojo: que la web *se vea* no demuestra que funcione bien; eso lo comprueban las verificaciones de cada fase.

ANTES DE SEGUIR: MUESTRA LAS EXTENSIONES DE ARCHIVO El método usa archivos como `CLAUDE.md` o `.env.local`. Si tu sistema oculta las extensiones, puedes acabar con `CLAUDE.md.txt` sin darte cuenta y la IA no lo encontrará. **Windows:** Explorador → Vista → Mostrar → activa «Extensiones de nombre de archivo» y «Elementos ocultos». **macOS:** Finder → Ajustes → Avanzado → «Mostrar todas las extensiones», y `Cmd+Shift+.` para ver los archivos ocultos. Si dudas de dónde estás, en la terminal escribe `pwd`: te da la **ruta absoluta** (la dirección completa de la carpeta, desde la raíz del disco).

N0.5 Cómo trabajar con la IA sin entender el código (todavía)

No necesitas saber programar para dirigir a Claude Code, igual que no necesitas saber poner ladrillos para dirigir una reforma. Necesitas otra cosa: **saber pedir, saber verificar y saber cuándo desconfiar**. Lo primero y lo segundo va aquí; lo tercero tiene su propio capítulo (N2).

Las 6 reglas del principiante

1. **Pide en tu idioma, con contexto.** No hace falta jerga: «quiero que el botón de enviar se ponga gris mientras se envía el formulario» es un prompt perfecto.
2. **Una cosa cada vez.** Peticiones pequeñas = errores pequeños = arreglos fáciles. «Hazme la web entera» es la receta del desastre.
3. **Pide el plan antes de ejecutar.** Activa el plan mode (`Shift+Tab`): la IA te contará QUÉ va a hacer antes de tocar nada, y tú apruebas o corriges. Como pedir presupuesto antes de la obra.
4. **Pide la explicación después.** Tras cada cambio: «explícame en lenguaje llano qué acabas de hacer y qué archivos has tocado». Así aprendes con cada paso, gratis.
5. **Verifica TÚ en el navegador.** La prueba final no es que la IA diga «hecho»: es que tú lo veas funcionar con tus ojos.
6. **Guarda cuando funcione.** Cada vez que algo quede bien, haz un commit (la «foto» de Git). Si lo siguiente sale mal, siempre puedes volver a la última foto.

El ciclo completo, en un dibujo (este bucle es el corazón de TODO el libro):



Si algo falla — copia el error entero a la IA (N0.4) y pide el arreglo mínimo. Cap. N2 si se atasca.
Al cambiar de tarea grande — `/clear` (borra la conversación; el código NO se toca) y empieza limpio. Cerrar la terminal, en cambio, sí detiene lo que corre en ella (por ejemplo `npm run dev`).
Este mismo bucle — escalado a 9 fases, es el método completo del libro (Parte 2).

Prompts de bolsillo (copia y pega)

SITUACIÓN	PROMPT LITERAL
No entiendes qué ha hecho	`Explícame en lenguaje llano, como si tuviera 12 años, qué acabas de hacer y por qué.`
Te propone algo enorme	`Eso es demasiado de golpe. Divídelo en pasos pequeños y hacemos solo el primero.`
Quieres entender un archivo	`Ábreme [ARCHIVO] y explícame qué hace cada bloque, en lenguaje llano.`
Miedo a romper algo	`Antes de tocar nada: ¿qué riesgos tiene este cambio y cómo lo deshacemos si sale mal?`
Comando desconocido	`¿Qué hace exactamente este comando y qué puede salir mal si lo ejecuto? [COMANDO]`

N0.6 Checklist: ¿estás listo para seguir?

Autoevaluación honesta. Cuando puedas marcar las 8 casillas, estás listo para el método completo (Parte 1 y Fase 0). Las casillas 1—4 se consiguen leyendo este capítulo; la 5 y la 6, haciendo el capítulo N1; la 7, leyendo el N2; la 8, con el Anexo C.

- [ ] Sé abrir la terminal en mi ordenador y ejecutar un comando.
- [ ] `node --version` me responde con un número de versión.
- [ ] Sé qué son Node, npm, un framework y un deploy (sin mirar la tabla).
- [ ] Sé leer la primera línea de un error y copiárselo ENTERO a la IA.
- [ ] He construido la mini-web del capítulo N1 y la he publicado con URL real (si la dejaste solo en local, marca «completado en local»: puedes seguir, y publicarás de verdad en F8).
- [ ] He probado el bucle: pedir → plan → aprobar → verificar → guardar.
- [ ] Conozco las señales de alarma de que la IA se está equivocando (capítulo N2).
- [ ] Tengo Git configurado y he hecho mi primer push a GitHub (Anexo C).

> **TU RUTA DESDE AQUÍ** **N0 (esto)** → **N1** (tu primera web online, paso a paso) → **N2** (cómo detectar errores de la IA, 10 min) → **Anexo C** (Git paso a paso, ~30 min) → **Parte 1** (el método de verdad empieza ahí). No te saltes el N1: esa primera victoria es la que te da la confianza para todo lo demás.

N1 Tu primer proyecto: recorrido guiado de cero a publicación FACIL

Vas a construir y publicar una landing page real (1 página, con formulario de contacto que te envía un email) SIN saber programar. Cada comando y cada prompt están escritos literales: copia, pega, comprueba y corrige hasta obtener una URL pública. El tiempo depende de tu experiencia y de los problemas que encuentres.

¿POR QUÉ ESTO ANTES QUE EL MÉTODO? Porque la moral importa. El método completo (9 fases) es un maratón; esto es cruzar tu primera meta en una tarde. Cuando veas TU web en TU móvil con TU url, el resto del libro deja de ser teoría. Esta mini-web es de juguete (sin Git, sin base de datos, sin seguridad seria) — el método de verdad empieza en la Parte 1. Pero es TUYA y está ONLINE.

TERMINAL Y CHAT DEL CLI

TERMINAL: órdenes como cd, npm, node. **CHAT DEL CLI:** frases y prompts a la IA. Abre una segunda ventana de PowerShell (Windows) o Terminal (macOS), ejecuta `cd "RUTA ABSOLUTA DE TU PROYECTO"` con tu ruta y comprueba `pwd`. En VS Code: Terminal → Nueva terminal, y verifica pwd. Conserva la primera con la IA; usa la segunda para npm run dev y otros comandos. Ctrl+C para el servidor solo en su ventana.

Qué necesitas

- Un ordenador (Windows o Mac) con internet.
- Una dirección de email.
- **Una cuenta de Claude de pago que incluya Claude Code.** Claude Pro es la opción de entrada; el plan gratuito no la incluye. Qué cuenta exactamente, en el paso 2; precios en `6-CONSULTA/COSTES.md`.
- Tiempo para aprender, practicar y resolver incidencias; sin una duración garantizada.
- Haber leído el capítulo N0 (sobre todo N0.1: abrir la terminal).

PASO	QUÉ HACES	TIEMPO
1	Instalar Node.js	~10 min
2	Instalar Claude Code	~5 min
3	Crear cuenta en Formspree (recibirá los mensajes del formulario)	~5 min
4	Crear la carpeta del proyecto y abrir la IA	~2 min
5	Pedirle la landing (prompt literal)	~15-20 min
6	Verla en tu ordenador	~5 min
7	Subirla a internet con Vercel	~10 min
8	Probar el formulario online	~5 min

Tiempo real la primera vez: cuenta con una tarde (3–4 horas). Las altas, los correos de confirmación y algún error se comen más tiempo que los pasos. Puedes parar al final de cualquier paso: la tarjeta «Para seguir otro día», al final de este capítulo, te dice cómo volver.

Paso 1 — Instala Node.js (~10 min)

1. Abre el navegador y ve a **nodejs.org**.
2. Pulsa el botón de descarga que ponga **LTS** («Long Term Support»: la versión estable y recomendada — la otra es para probar novedades).
3. Abre el instalador descargado y dale a **Siguiente** en todo, sin cambiar nada.
4. **Cierra la terminal si la tenías abierta y ábrela de nuevo** (las terminales solo se enteran de programas nuevos al reiniciarse).
5. Comprueba que está vivo:

TERMINAL

```
node --version
npm --version
```

Si cada línea responde con un número (por ejemplo `v22.11.0` y `10.9.0`), sigue al paso 2. Si no, mira este recuadro:

SI EL PASO 1 FALLA

«node no se reconoce» o «npm no se reconoce». La terminal aún no sabe que Node existe. Cierra *todas* las ventanas de PowerShell y abre una nueva. Si sigue igual, reinicia el ordenador. Si aun así falla, vuelve a ejecutar el instalador de nodejs.org dejando las opciones que vienen marcadas.

«No se puede cargar el archivo ...\npm.ps1 porque la ejecución de scripts está deshabilitada» (solo Windows). No has roto nada: PowerShell viene configurado para no ejecutar ese tipo de archivo. Escribe lo mismo añadiendo `.cmd`:

```
npm.cmd --version
```

Si responde con un número, Node está bien. **A partir de aquí, en PowerShell escribe `npm.cmd` donde el libro ponga `npm`, y `npx.cmd` donde ponga `npx`** (por ejemplo `npm.cmd run dev` o `npx.cmd vercel`). No hace falta cambiar ningún permiso de Windows ni abrir nada como administrador. Si la IA se topa con el mismo error, dile: «en este ordenador usa npm.cmd y npx.cmd».

Opcional, solo si prefieres no escribir .cmd cada vez: `Set-ExecutionPolicy -Scope CurrentUser RemoteSigned` permite, solo para tu usuario, los scripts creados en tu ordenador y los descargados que vengan firmados. No necesita administrador; respóndele `S` si pregunta.

Paso 2 — Instala Claude Code (~10 min)

TERMINAL — instalación y apertura del CLI

Windows: pega esta línea en PowerShell (no hace falta abrirlo como administrador):

```
irm https://claude.ai/install.ps1 | iex
```

macOS:

```
curl -fsSL https://claude.ai/install.sh | bash
```

Salen mensajes en inglés: es normal. Cuando termine, **cierra la terminal y abre una nueva**, y comprueba:

```
claude --version
```

Debe responder con un número de versión, por ejemplo `2.1.211 (Claude Code)`. Todavía no abras la IA: lo harás en el paso 4, ya dentro de tu carpeta. Git for Windows es opcional para empezar (sin él, la IA trabaja con PowerShell); lo instalarás en el Anexo C. Node.js LTS sigue siendo necesario para la web. Instalación oficial.

SI «CLAUDE» NO SE RECONOCE

1) Cierra todas las terminales y abre una nueva. 2) Si sigue igual, comprueba si el programa está instalado:

```
Test-Path "$env:USERPROFILE\.local\bin\claude.exe"
```

Si responde `True`, está instalado pero Windows no sabe dónde buscarlo. Pulsa la tecla Windows, escribe **variables de entorno** y abre «Editar las variables de entorno de esta cuenta». Selecciona **Path** → Editar → Nuevo, pega `%USERPROFILE%\.local\bin` y acepta en todas las ventanas. Abre una terminal nueva y repite `claude --version`. Si responde `False`, repite la instalación. En Mac: abre una terminal nueva; si sigue igual, ejecuta `~/.local/bin/claude doctor` y sigue lo que te indique.

QUÉ CUENTA NECESITAS (DECÍDELO ANTES DEL PASO 4)

Esta edición usa **Claude Code**, que necesita una cuenta de Claude **de pago**. **Claude Pro** es la opción de entrada; Max da más uso. El plan gratuito de Claude no incluye Claude Code. Se contrata en claude.ai; antes de pagar, mira el precio final, la renovación y los límites de uso.

- **ChatGPT no sirve aquí**: es de otra empresa. Si ya pagas ChatGPT, la edición que te encaja es la de Codex CLI.
- Cuando la IA te pida iniciar sesión (paso 4), elige **tu cuenta de Claude con suscripción**. La opción **Console / API** no es la que usamos: se paga aparte, por consumo.
- Pro comparte el límite de uso entre Claude y Claude Code. Si lo alcanzas, no es un error de tu proyecto: la IA te dice cuándo se repone. Cierra por hoy con la tarjeta «Para seguir otro día».

No pegues mkdir ni cd en el chat de la IA: van en la terminal.

Paso 3 — Crea tu cuenta en Formspree (~5 min)

Formspree es un servicio que recibe lo que la gente escriba en tu formulario y te lo reenvía por email, sin que tengas que montar nada de backend (la parte servidor). Comprueba la cuota gratuita y el destinatario en tu panel; si la superas, espera su reposición o revisa el plan.

1. Ve a **formspree.io** → **Sign up** (gratis, con tu email).
2. Confirma tu email si te lo pide.
3. Pulsa **New form**, llámalo `contacto` y créalo.
4. Copia la URL que te da, algo como `https://formspree.io/f/abcdwxyz`. **Guárdala**: la pegarás en el prompt del paso 5.

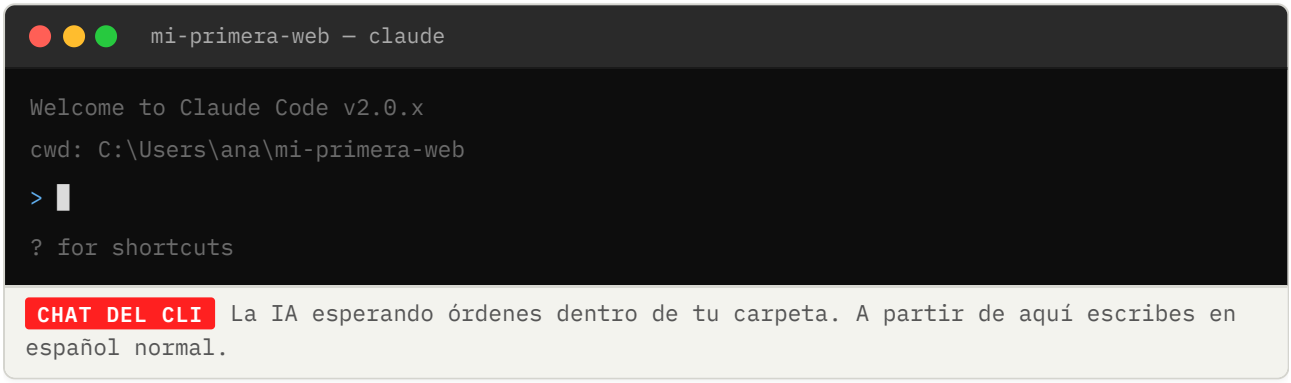
Paso 4 — Crea la carpeta y abre la IA (~2 min)

En la terminal:

TERMINAL

```
mkdir mi-primera-web
cd mi-primera-web
claude
```

Línea a línea: `mkdir` crea la carpeta («*make directory*»), `cd` entra en ella, y la tercera abre Claude Code AHÍ dentro (siempre se abre dentro de la carpeta del proyecto: así sabe qué archivos son los tuyos). Verás algo así:



CHAT DEL CLI La IA esperando órdenes dentro de tu carpeta. A partir de aquí escribes en español normal.

LA PRIMERA VEZ QUE ABRES LA IA

1. Te pide elegir los colores del texto: cualquiera vale.
2. Te abre el navegador para **iniciar sesión**: entra con tu cuenta de Claude con suscripción (no Console/API), autoriza y vuelve a la terminal.
3. Te pregunta si **confías en los archivos de esta carpeta**. Responde que sí: es `mi-primera-web`, tu carpeta nueva y vacía. Nunca lo aceptes en una carpeta cuyo contenido no conozcas.
4. Cuando veas la pantalla de arriba, ya puedes escribirle.

Dónde está tu carpeta: se ha creado dentro de tu carpeta de usuario. Para verla en el Explorador de Windows, pulsa `Windows+R`, escribe `%USERPROFILE%\mi-primera-web` y pulsa Enter. En Mac: Finder → Ir → Inicio.

Paso 5 — Pídele la landing (prompt literal) (~15-20 min)

Pega este prompt tal cual, rellenando los dos huecos `[ASÍ]`:

CHAT DEL CLI

```
Inicializa el proyecto en la MISMA carpeta raíz en la que estoy, sin crear
una subcarpeta. Si ya existe package.json, inspecciónalo y continúa sin
reinicializar ni sobrescribir trabajo.

Crea una landing page de UNA sola página con Next.js y Tailwind
para [DESCRIBE UNA IDEA DE PRÁCTICA EN UNA FRASE, p. ej. "un paseador
de perros ficticio en Valencia"]. Debe tener:

- Un titular grande con el nombre y una frase de qué ofrezco.
- Una sección con 3 ventajas o servicios (inventa textos razonables
  y márcalos con TODO para que yo los cambie luego).
- Un formulario de contacto (nombre, email, mensaje) que envíe los
  datos con method POST a esta URL de Formspree:
  [PEGA AQUÍ TU URL, p. ej. https://formspree.io/f/abcdwxyz]
- Tras enviar, mostrar un mensaje de "gracias, te contesto pronto".
- Diseño limpio y profesional, que se vea bien en móvil. Tema oscuro
  opcional: pregunta mi preferencia.

Soy principiante absoluto: ve creando el proyecto tú con los
comandos que hagan falta, y al final dime (1) el comando exacto
para ver la web en mi navegador y (2) una lista en lenguaje llano
de los archivos que has creado y para qué sirve cada uno.
```

Qué va a pasar (y qué haces tú):

1. La IA propondrá crear el proyecto y varios archivos. Te irá pidiendo **aprobación** para cada acción (crear archivo, ejecutar comando). Lee la línea de qué quiere hacer y aprueba (`Enter` o «yes») solo si entiendes qué es. Suele ofrecerte tres opciones: **sí**, **sí y no volver a preguntar** y **no**; mientras aprendes, elige la primera, así ves cada acción. Qué aprobar: crear archivos o instalar paquetes dentro de `mi-primera-web`, sí; borrar cosas fuera de esa carpeta, nunca. Si no entiendes una acción, pregúntale antes con el prompt «Comando desconocido» de N0.5.
2. Tardará unos minutos entre descargas y archivos. Normal.
3. Si en algún punto pregunta algo («¿prefieres X o Y?»), elige lo que te suene más simple o dile «elige tú lo más estándar».

Paso 6 — Mírala en tu ordenador (~5 min)

Necesitas una **segunda terminal abierta en la misma carpeta** (la primera sigue ocupada con la IA). La forma más fácil en Windows: abre la carpeta en el Explorador (`Windows+R` → `%USERPROFILE%\mi-primera-web` → Enter), haz clic en la **barra de dirección** de arriba, escribe `powershell` y pulsa Enter: se abre PowerShell ya dentro de esa carpeta. En Mac: en Finder, clic derecho sobre la carpeta → Servicios → «Nuevo terminal en la carpeta». Comprueba con `pwd` que la ruta termina en `mi-primera-web`; si un error dice que no encuentra `package.json`, estás en otra carpeta.

Cuando la IA termine, te dirá el comando para arrancar el servidor de desarrollo (tu web funcionando solo en tu ordenador). Será este:

TERMINAL

```
npm run dev
```

(Si en el paso 1 tuviste que usar `.cmd`: `npm.cmd run dev`.) Déjalo corriendo (esa terminal queda «ocupada»: es normal) y abre en el navegador: **http://localhost:3000** — `localhost` = tu propio ordenador; `3000` = la «puerta» donde escucha. Ahí está tu web. Pruébala y mírala en modo móvil: en Chrome pulsa `F12` y después `Ctrl+Shift+M` (F12 otra vez para salir).

¿Quieres cambiar algo? Se lo pides a la IA con frases normales: «pon el fondo más oscuro», «cambia el titular a X». Cada cambio se refresca solo en el navegador. ¿Sale un error? Al bucle de N0.4: copiar entero → pegar → arreglo mínimo.

Paso 7 — Súbela a internet (~10 min)

Para el servidor local con `Ctrl+C` (en la terminal donde corre). Ahora usaremos **Vercel**, un hosting compatible con Next.js. Como esta mini-web es una **práctica personal y ficticia**, encaja en el plan gratuito **Hobby**, que solo admite uso personal no comercial. Si la usas para un negocio real, necesitas Pro (desde 20 USD/mes más consumo) u otro hosting con contrato comercial. Si prefieres no publicar todavía, quédate en localhost: el ejercicio cuenta como **completado en local** y publicarás de verdad en F8.

TERMINAL

```
npx vercel
```

(O `npx.cmd vercel` si usas `.cmd`.) `npx` ejecuta una herramienta sin instalarla del todo. Te hará preguntas; respuestas para todas:

1. **Need to install the following packages… Ok to proceed?** → `y`: descarga la herramienta de Vercel.
2. **Log in:** sigue el flujo de autenticación que muestre tu versión y confirma en el navegador.
3. **Set up and deploy?** → `Y` (sí).
4. **Which scope?** → a qué cuenta de Vercel va el proyecto. Si solo aparece tu nombre, elige ese: es tu cuenta personal (Hobby), la correcta para esta práctica. **Project name / directory?** → pulsa Enter para aceptar lo que propone.
5. **Modify settings?** → `N` (no): detecta Next.js solo.

Al acabar te da una URL de prueba. Ahora publícala en producción (la versión pública «de verdad»):

TERMINAL

```
npx vercel --prod
```

La última línea te da tu URL definitiva, tipo **https://mi-primera-web.vercel.app**. Ábrela. Ábrela también en el móvil. Mándasela a alguien.

Paso 8 — Prueba el formulario online (~5 min)

1. En tu web pública, rellena el formulario y envíalo.
2. Revisa tu email (y el panel de formspree.io): debe llegar el mensaje.
3. Si no llega: mira spam, y revisa en formspree.io que el formulario esté «active». La primera vez, Formspree puede pedirte confirmar el email de destino.

> **APÚNTATELO: N1 HECHO**
>
> Abre `5-MI-PROGRESO.txt` (doble clic) y marca N1 cambiando `[ ]` por `[x]`; guarda con `Ctrl+S`. Vale tanto «publicado» como «completado en local».

Para seguir otro día (tarjeta de vuelta)

Puedes cerrar todas las ventanas y apagar el ordenador: tu trabajo está en la carpeta. Para retomarlo:

1. Abre la carpeta en el Explorador: `Windows+R` → `%USERPROFILE%\mi-primera-web` → Enter. En Mac: Finder → Ir → Inicio.
2. Haz clic en la barra de dirección, escribe `powershell` y pulsa Enter (en Mac: clic derecho en la carpeta → Servicios → Nuevo terminal en la carpeta). Comprueba con `pwd` que estás dentro.
3. Escribe `claude` para abrir la IA.
4. Abre `4-PROMPTS-PARA-COPIAR.txt`, busca con `Ctrl+F` **RETOMAR N1** y pégalo en el chat. Te dirá en qué paso estabas y qué sigue.
5. Para ver la web otra vez, abre una segunda terminal igual y escribe `npm run dev` (o `npm.cmd run dev`).

Con tu web de verdad la tarjeta es la misma, cambiando la carpeta por la de tu proyecto (en Documentos) y usando el **PROMPT RETOMAR** del método.

Qué acabas de hacer (traducción al método)

- Instalaste el **motor** (Node) y usaste **npm** — ese Node te sirve para todo el libro; el capítulo 01 solo lo comprueba.

- Un **framework** (Next.js) puso el esqueleto; la IA construyó encima — capítulos 02 y 04-07.
- Viste la web en el **servidor de desarrollo** (localhost) y luego hiciste **deploy** a **producción** — la Fase 8 hace esto con red de seguridad completa.
- Usaste el bucle **pedir → aprobar → verificar** — el método entero es este bucle con disciplina.

> **LO QUE ESTA MINI-WEB NO TIENE (A PROPÓSITO)** Sin Git (si borras la carpeta, no hay vuelta atrás), sin base de datos, sin seguridad revisada, sin SEO, sin tests. Para un juguete, perfecto. Para una web real (de cliente, con datos de gente), NO. Ese salto es exactamente lo que enseña el resto del libro. Siguiente parada: **capítulo N2** (10 min de lectura) y **Anexo C** (Git, para que tus proyectos tengan historial y marcha atrás).

Si algo se atasca

SÍNTOMA	QUÉ PASA	ARREGLO
«npm no se reconoce»	La terminal no se ha enterado de que Node existe.	Cierra y reabre la terminal. Si sigue: reinicia el ordenador.
«la ejecución de scripts está deshabilitada» o «running scripts is disabled» (Windows)	PowerShell viene de fábrica sin permitir ese tipo de archivo.	Escribe `npm.cmd` y `npx.cmd` en lugar de `npm` y `npx` (recuadro del paso 1). Sin administrador.
«claude no se reconoce»	La terminal no encuentra la IA recién instalada.	Recuadro del paso 2: terminal nueva y, si sigue, comprobar la instalación y el Path.
«Could not read package.json» / «ENOENT»	Estás en otra carpeta.	Abre la terminal desde la carpeta del proyecto (barra de dirección → `powershell`) y comprueba con `pwd`.
«Port 3000 is already in use»	Ya hay otra web tuya corriendo en esa «puerta».	Cierra las otras terminales, o pide a la IA que lo arranque en otro puerto.
El formulario no envía	La URL de Formspree está mal pegada en el código.	Dile a la IA: «verifica que el formulario apunta EXACTAMENTE a esta URL: [TU URL]».

Para cualquier otra cosa: el error entero a la IA (prompt de N0.4). Y si tras 3 intentos sigue igual, capítulo N2: ahí está el protocolo.

> muestra --fin

Continúa con el método completo.

Esta muestra incluye N0 y N1: fundamentos y primera web guiada. El aprendizaje y las incidencias requieren tiempo; no hay un plazo garantizado.

- **No está en esta muestra:** el capítulo N2 (detectar cuándo se equivoca la IA), la Parte 1, el método F0–F8, los anexos (Git paso a paso incluido) y los archivos que acompañan al libro. Las referencias a ellos que has visto apuntan a la guía completa.
- **39,99 € por edición con IVA incluido, pago único:** Claude Code, Codex CLI o Antigravity CLI. El **Pack 3 CLIs**, con las tres ediciones, cuesta **44,99 €**.
- **F0–F8:** cada sesión trabaja una sola fase; una fase puede necesitar varias sesiones.
- **25 plantillas, 10 archivos de fase, prompts, checklist, chuleta y tracker.** Clínica Luna es un ejemplo documental: documentación ficticia rellenada, no una aplicación ejecutable. Incluye una ruta corta para tu primer proyecto.
- **Caso real:** notfound404.es se construyó con Claude Code. No se garantizan resultados, ingresos, seguridad ni cumplimiento automático del RGPD.
- CLI, dominio y hosting aparte: consulta COSTES.md. Los 6,95 € del caso son gasto histórico del dominio, no coste total.
- Soporte del material y actualizaciones durante 12 meses; primera respuesta objetivo: 2 días laborables. Sin desarrollo ni auditoría individual. Contacto y actualizaciones sin publicidad: contacto@notfound404.es.
- Una persona o equipo de hasta 3 personas. Puedes entregar proyectos a clientes; no redistribuir el pack. Consulta QUE-INCLUYE.md y LICENCIA.md.

Producto y condiciones: **notfound404.es** · Payhip cobra y entrega.