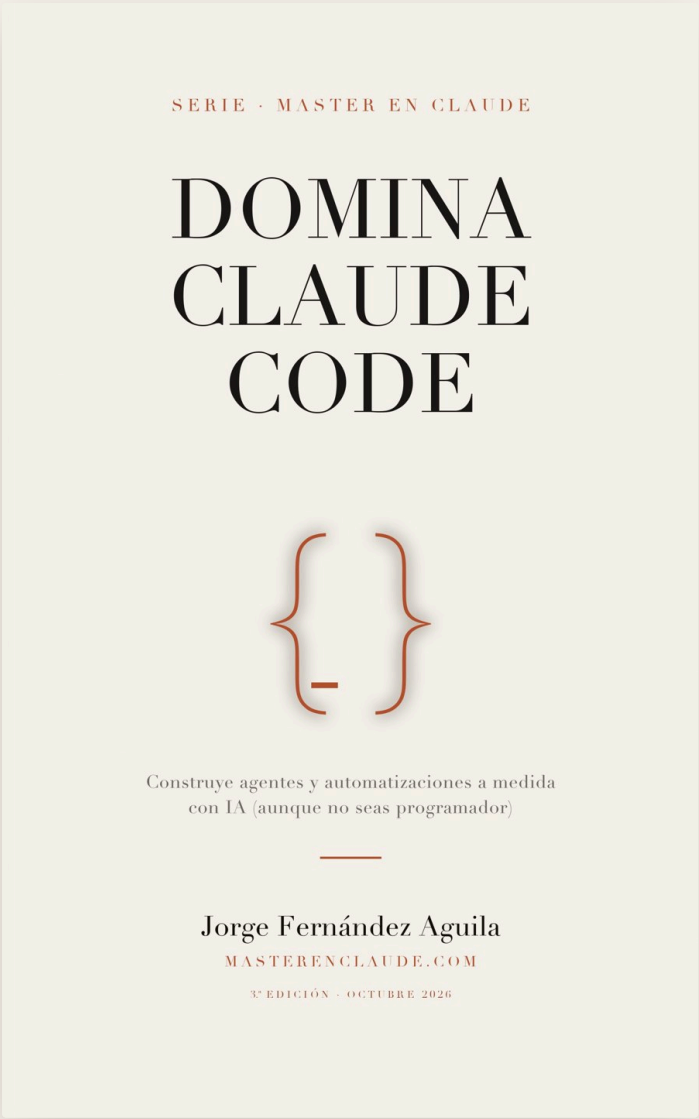
CAPÍTULO GRATIS



Domina Claude Code

Construye agentes y automatizaciones a medida con IA (aunque no seas programador)

Jorge Fernández Aguila

MASTERENCLAUDE.COM · 3.ª EDICIÓN

ADELANTO GRATUITO

Dos capítulos para empezar hoy

Aquí tienes completos, sin recortes, dos capítulos de *Domina Claude Code*:

Capítulo 1 · De dar órdenes a construir. Qué es exactamente Claude Code, por qué el código te da control total y por qué ya no hace falta ser programador para usarlo.

Capítulo 4 · Instala y configura Claude Code. La instalación exacta en Mac, Windows y Linux, tu primera conversación en el terminal y qué hacer si algo se atasca. Al terminarlo, lo tendrás funcionando.

Léelos con el ordenador delante. El primero te dice si este camino es para ti; el segundo te deja el taller montado.

EL LIBRO COMPLETO

Capítulo 1 · De dar órdenes a construir — ***incluido aquí***
Capítulo 2 · Qué es Claude Code y cómo piensa
Capítulo 3 · Cuándo dar el salto al código (y cuándo no)
Capítulo 4 · Instala y configura Claude Code — ***incluido aquí***
Capítulo 5 · Tu primer proyecto guiado
Capítulo 6 · Deja que Claude escriba el código por ti
Capítulo 7 · Dale herramientas y contexto (MCP y APIs)
Capítulo 8 · Construye un agente a medida
Capítulo 9 · Automatiza y programa tareas
Capítulo 10 · Supervisión, límites y seguridad en código
Capítulo 11 · Cuando algo se rompe (y se romperá)
Capítulo 12 · Vende soluciones a medida
Capítulo 13 · Construye tu propio producto

Más tres apéndices: glosario, recetario de peticiones y recursos con lista de comprobación.

CAPÍTULO 1

De dar órdenes a construir

> **3.ª edición, actualizada a octubre de 2026.** Claude Code cambia casi cada semana, y esta edición recoge lo que ha cambiado desde finales de julio de 2026 hasta la versión 2.1.311: las sesiones que ahora arrancan en **modo automático**, los modelos nuevos (Opus 5.5 pasa a ser el de serie también en el plan Pro), las **tareas programadas en la nube** con `/schedule`, la revisión de código con `/code-review`, el modo rápido `/fast`, los **artefactos** y varias funciones que la edición anterior no contaba: la vista de agentes, `/goal`, Remote Control y los worktrees. Si un comando no te responde exactamente como aquí, mira la documentación oficial: puede haber cambiado después de cerrar esta edición.

> **En este capítulo:** los cuatro niveles de trabajar con IA y en cuál estás · qué es exactamente Claude Code · por qué el código da control total, con tres casos reales · el mito de "hay que ser programador" y por qué ha caído · qué necesitas de verdad · qué vas a saber construir al terminar · y cómo leer este libro.

Si vienes de *Domina Cowork*, o simplemente ya usas Claude para trabajar, sabes montar agentes que hacen cosas por ti sin tocar código. Es potentísimo, y para el 90% de lo que querrás hacer, te sobra.

Este libro va del 10% restante: **cuando quieres control total y construir algo exactamente a tu medida.** Ahí es donde entra Claude Code.

Y déjame quitarte el miedo desde la primera página: **hoy no necesitas ser programador para esto.** El motivo lo verás en un momento, y no es un eslogan.

Los cuatro niveles

Antes de nada, sitúate. Trabajar con IA tiene cuatro escalones, y cada uno resuelve cosas distintas:

Nivel 1: conversación. Le preguntas, te contesta. Redactas, resumes, traduces, piensas en voz alta. Es el chat de toda la vida.

Nivel 2: contexto. Le das tus documentos, tus datos, tus instrucciones permanentes, y trabaja sobre lo tuyo. Ya no es un asistente genérico: sabe de tu negocio.

Nivel 3: agentes sin código. Ya no le pides cosas una a una: le das un objetivo y unas herramientas, y trabaja solo hasta conseguirlo. Es lo que viste en *Domina Cowork*.

Nivel 4: construir. Tú decides qué herramientas tiene, cómo funciona por dentro, con qué se conecta, cuándo se ejecuta y qué límites tiene. El resultado es tuyo, vive en tu ordenador y evoluciona contigo.

Este libro es el nivel 4. Y conviene decir algo que casi nadie dice: **subir de nivel no siempre es mejor.** El nivel correcto es el más bajo que resuelva tu problema. El capítulo 3 va entero de esa decisión.

De pedir a construir

En Cowork le das un objetivo a un agente y él trabaja. Es como tener un empleado buenísimo al que diriges con palabras.

Claude Code es otra cosa: es como tener un **taller propio** donde construyes la herramienta exacta que necesitas, pieza a pieza, con Claude haciendo de constructor.

La diferencia, en una frase: en Cowork **usas** agentes; con Claude Code **construyes** los tuyos, sin límites de lo que traen de serie. Si Cowork es comprar un mueble montado y genial, Claude Code es tener el taller para hacerte el mueble exacto que necesita tu hueco.

¿Y por qué querría alguien complicarse? Por una palabra: **control.**

Qué es exactamente Claude Code

Vamos a definirlo bien, porque la mitad de la confusión viene de aquí.

Claude Code es Claude con manos, trabajando dentro de una carpeta de tu ordenador.

Un chat de IA solo puede escribirte texto. Claude Code, además, puede leer los archivos de esa carpeta, crearlos, modificarlos, ejecutar comandos, ver el resultado y corregirse. Vive en el terminal, esa ventana de fondo oscuro que da respeto la primera vez, y hablas con él en español, no en comandos.

Eso es todo. No es un editor de código ni una web donde subes cosas. Es un programa que se instala en tu ordenador y trabaja sobre tus archivos. (Existen también sesiones de Claude Code que corren en los ordenadores de Anthropic, y las verás en el capítulo 9; pero la forma de trabajar de este libro, la que te da el control, es la de tu propio ordenador.)

Y de esa definición sale una consecuencia importante: **lo que construyas es tuyo.** No vive en la cuenta de nadie. Son archivos en tu disco que puedes copiar, mover, respaldar y llevarte.

Por qué el código da control total

Todo lo que has hecho hasta ahora ha ido por caminos que otros construyeron: las conexiones que la herramienta ofrecía, las opciones que traía. El código quita esos límites.

Tres casos que lo hacen concreto.

El fontanero con un sistema propio. Tiene sus presupuestos en un programa antiguo que solo funciona en su ordenador. Ninguna herramienta estándar se conecta a eso. Con código, sí: un agente lee sus archivos, saca los datos y le prepara el resumen mensual. Nadie iba a construir esa integración por él, porque él es el único cliente del mundo que la necesita.

La gestoría con una regla rarísima. Clasifican facturas siguiendo un criterio propio, mezcla de reglas fiscales y costumbres de la casa, que nadie de fuera entendería. Con código, esa lógica se escribe tal cual y funciona exactamente como la tienen en la cabeza.

El consultor que repite el mismo montaje. Instala el mismo agente en doce clientes. Sin código, son doce montajes a mano que hay que mantener uno a uno. Con código, es un proyecto que se despliega doce veces y se corrige en un sitio.

Fíjate en el patrón: **el código gana cuando eres el único que necesita eso.** Para lo que necesita mucha gente, ya existe una herramienta y es mejor usarla.

Y hay una cuarta ventaja que solo se aprecia con el tiempo: **queda guardado como código que evoluciona contigo.** Puedes mejorarlo, versionarlo, volver atrás si algo se rompe, y no dependes de que nadie te lo mantenga ni de que una empresa siga existiendo el año que viene.

El mito de "hay que ser programador"

Aquí está la parte que cambia el juego, y por la que este libro existe.

Durante décadas, construir software exigía años de aprender a programar. Ese muro **se ha caído**, y conviene entender por qué, para que no suene a promesa de vendedor.

Programar son en realidad dos habilidades distintas. Una es *saber qué debe hacer el programa*: entender el problema, decidir qué pasa en cada caso, saber cuándo el resultado es correcto. La otra es *saber escribirlo* en un lenguaje que la máquina entienda: la sintaxis, las librerías, los detalles.

La segunda es la que costaba años. Y es exactamente la que Claude hace ahora por ti.

Tu trabajo con Claude Code es la primera: **explicar bien lo que quieres, poner los límites y saber reconocer si el resultado es correcto.** Eso no te lo puede hacer nadie, porque nadie más conoce tu negocio. Y no requiere estudiar años; requiere pensar con claridad, que probablemente ya haces.

Así que sí, alguien tiene que saber programar en esta relación. Pero no eres tú.

Tres cosas que se dicen y no son verdad

Antes de seguir, conviene desactivar tres ideas que circulan mucho y que hacen daño en direcciones opuestas.

"Con esto ya no hace falta entender nada." Falso, y peligroso. Sigue haciendo falta entender **el problema**: qué debe pasar en cada caso, qué es un resultado correcto, dónde están los riesgos. Lo que ya no hace falta es entender la sintaxis. Quien confunde las dos cosas acaba con programas que funcionan perfectamente haciendo lo que no era.

"Esto es para gente técnica, yo no puedo." También falso, en la dirección contraria. La barrera de entrada real de este libro es una tarde de instalación y la paciencia de probar cosas. Nada más. Si has configurado alguna vez un router o has aprendido a usar una herramienta nueva de tu trabajo, tienes de sobra.

"La IA escribe el código, así que el código no importa." El código importa, pero de otra manera: importa como **activo tuyo**. No tienes que saber escribirlo, pero sí conviene que sepas que está ahí, que es tuyo, que se puede guardar, versionar y llevar a otro sitio. Esa es una diferencia enorme frente a las herramientas donde tu trabajo vive dentro de la cuenta de otro.

El cambio real: de escribir a decidir

Si tuviéramos que resumir en una frase qué cambia con Claude Code, sería esta: **el cuello de botella se ha movido de la ejecución a la decisión.**

Antes, si tenías una buena idea de automatización, el problema era construirla: había que saber, tardar y equivocarse mucho. La idea era barata y la ejecución cara.

Ahora la ejecución es barata. Puedes construir en una tarde algo que hace cinco años eran dos semanas. Y eso desplaza el problema a otro sitio: **qué construir, para quién, y cómo saber si está bien.**

Esto tiene una consecuencia práctica que va a recorrer todo el libro. Vas a ver mucho énfasis en cosas que no son técnicas: definir bien el problema, poner límites, decidir el criterio de "hecho", comprobar el resultado. No es relleno. Es que ahí es donde está ahora el trabajo de verdad.

Qué necesitas de verdad

Seamos honestos, sin asustar. Sí hace falta un poco más que en Cowork:

Abrir el terminal. Esa ventana negra que da respeto la primera vez. El capítulo 2 la abre contigo y verás que no muerde.

Paciencia para probar y corregir. Las cosas fallan, se arreglan y vuelven a fallar. Es el oficio, no un problema tuyo.

Criterio para revisar. No para leer código, sino para mirar un resultado y decidir si está bien. Eso ya lo tienes.

Un plan de pago de Claude. Es la única barrera de entrada real y conviene decirla en la página 5 y no en la 40: el plan gratuito no incluye Claude Code.

Lo que **no** hace falta: saber programar de antes, memorizar comandos, entender por dentro cómo funciona nada.

Si sabes explicar lo que quieres y tienes curiosidad, puedes. Este libro te lleva de la mano.

Qué vas a saber construir

No nos vamos a quedar en la teoría. Al cerrar la última página, sabrás:

- **Instalar y manejar Claude Code** sin miedo al terminal, y saber qué hacer cuando algo se atasca (capítulos 2 a 4).
- **Dictarle proyectos** y llevarlos de la idea a algo que funciona, probando y corrigiendo; y dirigirle bien, con plan previo, comandos propios y subagentes (capítulos 5 y 6).
- **Conectar tus agentes** a tus datos, tus apps y tus servicios, y darles la memoria del proyecto (capítulo 7).
- **Construir un agente completo** con objetivo, manos, memoria, bucle y límites; y dejarlo corriendo solo a su hora (capítulos 8 y 9).
- **Hacerlo seguro**: permisos, hooks, secretos, y las mismas reglas de oro de siempre aplicadas al código (capítulo 10).
- **Arreglar lo que se rompa**, con método y sin agobio, y desplegarlo donde trabaje de verdad (capítulo 11).
- **Ganar dinero**: vendiendo soluciones a medida, que se cobran mucho mejor que sin código, o montando tu propio producto (capítulos 12 y 13).

Es el tercer y último escalón de la serie: **usar Claude, crear agentes sin código, construir a medida con código.** Si has llegado hasta aquí, ya piensas en agentes; ahora te doy las llaves del taller.

Cómo leer este libro

Tres recomendaciones que cambian mucho el aprovechamiento.

Con el ordenador delante. Este libro no se aprende leyéndolo en el sofá. Cada capítulo termina con un ejercicio y todos están pensados para hacerse en el momento, no para "cuando tenga tiempo".

En orden hasta el capítulo 8. Los tres primeros son contexto y decisión; del 4 al 8 se construye acumulativamente y saltarse uno se nota. Del 9 en adelante puedes ir a lo que te interese.

Con un proyecto propio en la cabeza. Elige ya una cosa tuya, real, repetitiva y aburrida que te gustaría no volver a hacer nunca. Ese será tu hilo conductor. Los ejemplos del libro te servirán mucho más si los aplicas a algo que te importa.

Y una última cosa. Si en algún momento te atascas, la salida está siempre a una frase de distancia: se lo cuentas a Claude y él te desatasca. Ese es, literalmente, el superpoder de esta herramienta.

En 5 minutos

- Hay **cuatro niveles** de trabajar con IA: conversación, contexto, agentes sin código y construir. El nivel correcto es el más bajo que resuelva tu problema.
- **Claude Code es Claude con manos, trabajando dentro de una carpeta de tu ordenador.** Lee archivos, los edita, ejecuta comandos, ve el resultado y se corrige.
- El código gana **cuando eres el único que necesita eso**: una conexión rara, una lógica muy tuya, algo que hay que replicar muchas veces.
- **El mito ha caído** porque programar son dos habilidades y Claude hace la que costaba años. La otra, entender el problema y juzgar el resultado, es la tuya y no se puede delegar.
- Necesitas: **terminal, paciencia, criterio y un plan de pago.** No necesitas saber programar.
- Léelo **con el ordenador delante** y con un proyecto propio en la cabeza.

Ejercicio

1. Sitúate: ¿en qué nivel de los cuatro estás trabajando hoy? Sé honesto. Mucha gente cree estar en el 3 y está en el 1.
2. Piensa en algo que quisiste automatizar y no pudiste del todo: una conexión rara, una lógica muy tuya, algo que repites en muchos sitios. Apúntalo. Será tu hilo conductor durante el libro; para el primer proyecto del capítulo 5 elegirás algo más pequeño, como verás en el capítulo 3.
3. Escribe en una frase de andar por casa qué querrías que ese programa hiciera. Sin pensar en el cómo. Ese "qué" es el 80% del trabajo.
4. Comprueba hoy que tienes un plan de Claude que incluya Claude Code. Mejor descubrirlo ahora que en mitad de la instalación.

En el próximo capítulo conocemos a Claude Code por dentro, sin tecnicismos: el terminal sin miedo, su bucle de trabajo, qué es el contexto y por qué importa, y qué cosas **no** puede hacer.

MIENTRAS TANTO, EN EL LIBRO

Capítulos 2 y 3

Antes de instalar nada, el libro dedica dos capítulos a que entiendas lo que vas a usar:

Capítulo 2 · Qué es Claude Code y cómo piensa. El terminal sin miedo, el bucle con el que trabaja (lee, planifica, edita, ejecuta y corrige), qué es el contexto y qué cosas no puede hacer.

Capítulo 3 · Cuándo dar el salto al código (y cuándo no). Las señales de que una herramienta sin código se te queda corta, lo que solo el código te da y cómo elegir tu primer proyecto.

Este adelanto salta directamente al capítulo 4 para que puedas tener Claude Code funcionando hoy mismo.

Ver el libro completo en Amazon

CAPÍTULO 4

Instala y configura Claude Code

> **En este capítulo:** los requisitos reales, incluida la cuenta que hace falta · la instalación exacta en Mac, Windows y Linux · cómo comprobar que ha ido bien · iniciar sesión · tu primera conversación · dónde vive tu trabajo · los atajos que vas a usar todos los días · y qué hacer si la instalación se atasca.

Se acabó la teoría. Vamos a tener Claude Code funcionando en tu ordenador. No te saltes ningún paso y, si algo se atasca, respira: casi siempre es una tontería que se arregla en un minuto.

Este capítulo es tu taller montado. Hazlo con el ordenador delante.

Requisitos reales

No necesitas un ordenador potente. Necesitas esto:

Sistema operativo. macOS 13 o posterior, Windows 10 (versión 1809 o posterior) o Windows Server 2019 o posterior, Ubuntu 20.04 o posterior, Debian 10 o posterior, o Alpine Linux 3.19 o posterior.

Hardware. 4 GB de RAM o más, procesador x64 o ARM64. Cualquier portátil de los últimos años lo cumple de sobra.

Conexión a internet. Claude Code habla con los servidores de Anthropic; sin conexión no funciona.

Un terminal. Bash, Zsh, PowerShell o CMD. Los que ya trae tu sistema valen.

Y una cuenta de pago. Esto conviene decirlo claro para que no te lleves un chasco: **el plan gratuito de Claude no incluye Claude Code.** Necesitas un plan Pro, Max, Team o Enterprise, o bien una cuenta de Consola con facturación por uso.

Es la única barrera de entrada real del libro, y merece la pena ponerla en contexto: construir con Claude Code da muchos más pasos que una conversación de chat, y por tanto consume más. El plan de pago no es un peaje arbitrario; es lo que sostiene ese consumo.

Si prefieres no pasar por el terminal, existe también una aplicación de escritorio para Mac, Windows y Linux que hace lo mismo con una interfaz gráfica. Este libro va del terminal, porque es donde está todo el control, pero conviene que sepas que la otra puerta existe.

La instalación, exacta

Copia y pega. No memorices nada.

Mac, Linux y WSL

Abre la aplicación **Terminal** y pega esto:

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

Windows con PowerShell

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

Windows con CMD

```
curl -fsSL https://claude.ai/install.cmd -o install.cmd && install.cmd && del
install.cmd
```

¿Cómo sé si estoy en PowerShell o en CMD? Mira el principio de la línea. Si pone `PS C:\Usuarios\Tunombre>`, estás en PowerShell. Si pone `C:\Usuarios\Tunombre>` sin el `PS`, estás en CMD. Y hay dos mensajes de error que también te lo dicen: si ves `The token '&&' is not a valid statement separator`, has pegado el comando de CMD en PowerShell. Si ves `'irm' is not recognized`, has pegado el de PowerShell en CMD.

Otras vías

Si prefieres tu gestor de paquetes habitual:

```
brew install --cask claude-code        # Mac, con Homebrew
```

```
winget install Anthropic.ClaudeCode    # Windows, con WinGet
```

```
npm install -g @anthropic-ai/claude-code   # con npm, requiere Node.js 22 o posterior
```

Con una diferencia importante: **las instalaciones nativas (los tres primeros comandos de la sección anterior) se actualizan solas en segundo plano; las de Homebrew, WinGet y los gestores de Linux, no, salvo que lo actives tú.** Homebrew y WinGet ya permiten activar la actualización automática (la documentación oficial de instalación explica cómo); si no lo haces, tendrás que actualizar tú de vez en cuando con `brew upgrade claude-code` o `winget upgrade Anthropic.ClaudeCode`.

Recomendación para empezar: usa la instalación nativa. Es una preocupación menos.

Y no uses nunca `sudo npm install -g`. Genera problemas de permisos y es un riesgo de seguridad innecesario.

Un apunte para Windows

Puedes usar Claude Code directamente en Windows o dentro de WSL (el Linux integrado de Windows). Las dos opciones son válidas y la elección depende de dónde tengas tus proyectos.

Si vas por la vía nativa, conviene que instales **Git para Windows**. No es obligatorio, pero habilita la herramienta Bash, que es la que Claude Code usa para ejecutar comandos. Sin ella, usará PowerShell, que también funciona pero es un camino menos trillado.

Comprueba que ha ido bien

Este es el paso que la gente se salta. Dos comandos, diez segundos, y sabes que el taller está montado de verdad.

```
claude --version
```

Debe responder con un número de versión seguido de `(Claude Code)`, algo como `2.1.311 (Claude Code)`. Si eso funciona, estás dentro.

```
claude doctor
```

Este es el segundo, y merece la pena conocerlo desde el primer día. Hace un diagnóstico de solo lectura de tu instalación y tu configuración, sin abrir sesión, y te dice si algo está mal y cómo arreglarlo. Es tu comodín para cualquier cosa rara durante todo el libro.

Inicia sesión

Arranca Claude Code escribiendo:

```
claude
```

La primera vez te pedirá iniciar sesión. Se abre el navegador, autorizas con tu cuenta, vuelves al terminal, y ya estás.

Un detalle que te encontrarás si tienes la variable de entorno `ANTHROPIC_API_KEY` puesta en tu sistema: en vez de abrir el navegador, Claude Code te preguntará una vez si quieres usar esa clave. Si no sabes qué es eso, no la tienes, y no te preocupa.

Hay una segunda forma de autenticarse, pensada para agentes que se ejecutan solos sin nadie delante:

```
claude setup-token
```

Genera un token de larga vida para scripts y automatizaciones. No lo necesitas todavía; lo usaremos en el capítulo 9. Quédate solo con que existe y con que es la solución al pesadísimo bucle de "inicia sesión" cuando no hay nadie para iniciarla.

Tu primera conversación

Con `claude` arrancado, ya puedes hablarle. Prueba algo inofensivo, solo para verlo actuar:

> "Hola. Explícame en tres frases qué puedes hacer por mí y crea un archivo de texto llamado `hola.txt` que diga 'Mi primer día con Claude Code'."

Dale a Enter y observa dos cosas. La primera: te contesta. La segunda, y la que importa: **qué pasa con el permiso para crear el archivo.**

Si no has tocado nada, lo más probable es que estés en **modo automático**: desde agosto de 2026 en los planes Pro, Max y Team, y desde finales de septiembre en todos, es el modo con el que arranca Claude Code cuando no has configurado otro. En ese modo verás que crea el archivo sin preguntarte. No es que se haya saltado la seguridad: un segundo modelo ha revisado la acción, ha visto que es inofensiva y la ha dejado pasar.

Para ver el aviso de permiso clásico, pulsa una vez `Shift+Tab`: desde el modo automático, la primera pulsación te lleva al modo **Manual**, el que pregunta antes de cada cambio. Pídele ahora que cree otro archivo, `hola2.txt`, y ahí lo tienes: el aviso. Dile que sí. Los dos modos, y cuándo conviene cada uno, los vemos en el capítulo 10.

Cuando termine, busca el archivo en tu carpeta. Ahí está, creado de verdad.

Ese momento (pedirle algo y ver que *actúa*, no solo que habla) es cuando "Claude Code" deja de ser una palabra. Enhorabuena: acabas de dar tu primer paso construyendo.

Dónde vive tu trabajo

Todo lo que hagas con Claude Code vive en **carpetas de tu ordenador**. Esto importa por tres razones.

Es tuyo y está a la vista. No está atrapado en ninguna nube ajena; son archivos en tu disco, que puedes ver, copiar y respaldar.

Trabaja donde le dices. Claude Code actúa sobre la carpeta en la que lo abres. Esa carpeta es su mundo: dentro, lee sin pedirte permiso; para leer fuera de ella te avisa antes, también en modo automático. Es una barrera de seguridad, no una limitación caprichosa.

Puedes ampliarle el mundo si hace falta. Si necesitas que vea otra carpeta, se lo dices al arrancar con `claude --add-dir ../otra-carpeta`, o durante la sesión con `/add-dir`.

Un consejo desde ya: **una carpeta por proyecto, siempre.** Crea una carpeta limpia para tu primer proyecto de verdad (por ejemplo, `mi-primer-agente`) y arranca `claude` ahí dentro. Tener el taller ordenado desde el principio te ahorra líos después, y sobre todo evita que Claude ande revolviendo entre cosas que no tienen que ver con lo que estás haciendo.

Los atajos que vas a usar todos los días

No te aprendas todos. Aprende estos ocho y el resto los descubrirás solo.

ATAJO	QUÉ HACE
`Ctrl+C`	Cancela lo que está haciendo ahora mismo
`Ctrl+D`	Sale de Claude Code (dos veces seguidas para confirmar)
`Esc`	Interrumpe a Claude (para y conserva lo hecho) o cierra un menú
`Esc` `Esc`	Con la línea vacía, abre el menú de rebobinar
`Shift+Tab`	Cambia de modo de permiso (capítulo 10)
`Ctrl+R`	Busca en el historial de lo que has escrito
`Flecha arriba`	Recupera lo anterior que escribiste
`Ctrl+J`	Salto de línea sin enviar el mensaje

Los dos primeros son los importantes de verdad: **`Ctrl+C` para "para ahora mismo" y `Ctrl+D` para irte.** Saber cómo detener algo es lo primero que hay que aprender de cualquier herramienta.

Además, dos caracteres cambian lo que escribes:

- **`/` al principio** abre el menú de comandos. Pulsa `/` y verás la lista entera.
- **`@` en medio de una frase** hace referencia a un archivo, con autocompletado. "Revisa `@src/facturas.py` y dime qué hace" es mucho más preciso que describir el archivo con palabras.

Si quieres la lista completa de atajos y comandos en cualquier momento, escribe `/help`.

Sesiones: retomar donde lo dejaste

Un detalle que se agradece mucho el segundo día. Cada conversación con Claude Code queda guardada, y puedes volver a ella.

Para retomar la más reciente en esta carpeta:

```
claude --continue
```

Para elegir entre las anteriores:

```
claude --resume
```

Y desde dentro de una sesión, el comando equivalente:

```
/resume
```

Esto importa por una razón práctica: **el contexto se recupera con la sesión.** Si ayer estuviste dos horas explicándole tu proyecto y hoy arrancas de cero, has tirado esas dos horas. Si retomas, sigue sabiendo dónde estabais.

Un apunte: las sesiones se buscan en la carpeta desde la que arrancaste. Si retomas desde otro sitio, no la va a encontrar.

Y el movimiento contrario, igual de útil: cuando empieces algo que no tiene nada que ver con lo anterior, **límpialo** con `/clear`. Arrastrar una conversación vieja a un tema nuevo es una de las causas más comunes de respuestas raras.

Entrar en un proyecto que ya existe

Hasta ahora hemos hablado de carpetas vacías. Pero lo normal, a partir de la semana que viene, es abrir Claude Code en un proyecto que ya tiene cosas dentro.

La primera vez que lo hagas, dos movimientos que valen por una hora de exploración a mano:

> "Explórame este proyecto y hazme un resumen: qué hace, cómo está organizado, cómo se ejecuta y cómo se prueba. Sin tecnicismos, como a alguien que llega nuevo."

Y después:

```
/init
```

Que genera el `CLAUDE.md` con lo que ha descubierto, para no tener que volver a explorarlo cada día. Lo vemos a fondo en el capítulo 7.

Cuatro comandos para conocer hoy

De la lista larga, estos cuatro te van a servir desde el primer día:

COMANDO	PARA QUÉ
`/help`	La lista de todo lo disponible
`/status`	En qué estado está tu sesión
`/config`	Abrir los ajustes
`/doctor`	Revisar instalación y configuración

Deja el taller ordenado

Dos minutos de configuración inicial que te van a ahorrar disgustos durante todo el libro. Con Claude Code arrancado:

> "Créame un archivo de configuración personal en `~/.claude/settings.json` con reglas de permiso que denieguen leer archivos `.env` y denieguen ejecutar `rm -rf`. Después explícame qué has puesto y dónde."

Esa es tu primera línea de defensa, y la desarrollamos entera en el capítulo 10. Ponerla hoy es gratis.

Otras puertas: el editor y la aplicación de escritorio

Este libro va del terminal, porque es donde está el control completo y donde funciona todo lo que vamos a ver. Pero conviene que sepas que hay más puertas al mismo sitio.

Existe una **extensión para Visual Studio Code** y un **complemento para los entornos de JetBrains**, para quien ya trabaje en esos editores y prefiera no salir de ellos. Y existe la **aplicación de escritorio** para Mac, Windows y Linux, con interfaz gráfica; desde finales de septiembre de 2026, `claude --desktop` la abre directamente en la carpeta en la que estás.

Y una puerta más que la edición anterior no contaba: **Remote Control**. Dentro de una sesión escribes `/remote-control` (o arrancas con `claude --remote-control`), y puedes seguir esa misma sesión desde el navegador, en claude.ai/code, o desde la app de Claude en el móvil. Claude sigue

trabajando en tu ordenador, con tus archivos y tus conexiones; el móvil es solo otra pantalla para darle órdenes. Muy útil para lanzar algo largo en el despacho y vigilarlo desde el sofá. Eso sí: el ordenador tiene que seguir encendido y con la sesión abierta.

Todas comparten la misma configuración: leen y escriben en la misma carpeta `~/.claude`, así que tus permisos, tus subagentes y tus comandos propios valen en todas.

La recomendación para este libro es clara: **aprende en el terminal.** Si después prefieres otra puerta, la abres sabiendo qué hay detrás. Al revés no funciona igual de bien, porque la interfaz gráfica te esconde precisamente las cosas que conviene entender.

Preguntas del primer día

Las que le surgen a casi todo el mundo en las primeras dos horas.

¿Se está enviando mi código a algún sitio? Sí: para que Claude pueda trabajar con tus archivos, el contenido relevante viaja a los servidores de Anthropic, igual que en cualquier conversación. No es un programa que funcione sin conexión. Si trabajas con material sensible, léete las condiciones de tu plan y las políticas de tratamiento de datos, y ten en cuenta las reglas `deny` del capítulo 10 para que no lea lo que no debe.

¿Puede borrarme algo sin querer? Puede, si le das permiso. Por eso existe el sistema de permisos: en modo Manual te pregunta antes de ejecutar comandos, y en modo automático un revisor bloquea lo que va más allá de lo que pediste. Ninguno de los dos es infalible, y por eso el capítulo 10 existe. Con las reglas puestas y trabajando sobre copias, el riesgo real es muy bajo.

¿Puedo usarlo en un proyecto que no es de programación? Sí, y es de sus usos más infravalorados. Una carpeta de documentos, una hoja de datos, un archivo de notas: todo eso son archivos, y Claude Code trabaja con archivos. No hace falta que haya código de por medio.

¿Funciona en español? Perfectamente. Háblale en español y te responde en español. Los nombres de los comandos y las opciones están en inglés, y eso no cambia.

¿Y si me equivoco al darle permiso a algo? Se corrige. Abre `/permissions`, busca la regla que se guardó y quítala. Nada de lo que apruebes es para siempre.

¿Cuánto va a consumir esto? Depende mucho de lo que hagas. Mira `/usage` los primeros días para hacerte una idea con tus propios números, en lugar de con los de otro.

¿Por qué no me pregunta nada antes de hacer las cosas? Porque estás en modo automático, el de partida, como viste al principio del capítulo. Mira el modo activo en la parte de abajo de la pantalla y, si quieres que te pregunte, pulsa `Shift+Tab` para pasar a Manual. Para que arranque siempre en Manual, se configura en tu `settings.json` (capítulo 10).

¿Tengo que cerrar sesión al terminar? No. Sales con `Ctrl+D` y ya está. La sesión queda guardada y la puedes retomar mañana.

Si la instalación se atasca

Los cuatro tropiezos más comunes y su arreglo.

`command not found: claude` **justo después de instalar.** Cierra el terminal y ábrelo de nuevo. La instalación añade una ruta al sistema que solo se lee al arrancar una sesión nueva. Es, con diferencia, la causa más común.

El comando de instalación da un error de sintaxis o un 403. Casi siempre es haber pegado el comando de un sistema en otro (el de CMD en PowerShell, o al revés). Comprueba dónde estás con la pista del prompt que vimos antes.

En Windows, no encuentra Git Bash. Instala Git para Windows. Si aun así no lo encuentra, se le puede indicar la ruta en la configuración; pregúntale a Claude cómo, o ejecuta `claude doctor` y sigue lo que te diga.

Cualquier otra cosa rara. `claude doctor`. En serio. Antes de buscar en internet, antes de reinstalar, antes de darlo por imposible: `claude doctor`.

En 5 minutos

- Requisitos: **un ordenador normal, internet y una cuenta de pago** (Pro, Max, Team, Enterprise o Consola). El plan gratuito no incluye Claude Code.
- Instala con el comando nativo de tu sistema: `curl ... install.sh | bash` en Mac y Linux, `irm ... install.ps1 | iex` en PowerShell.
- **Comprueba siempre** con `claude --version` y `claude doctor`. Diez segundos que te ahorran una tarde.
- Inicia sesión arrancando `claude`. Para automatizaciones, existe `claude setup-token` (capítulo 9).
- Tu trabajo vive en **carpetas tuyas**. Una carpeta por proyecto, siempre.
- Atajos clave: `Ctrl+C` **para parar,** `Ctrl+D` **para salir**, `/` para comandos, `@` para archivos, `Shift+Tab` para cambiar de modo de permiso.
- Las sesiones arrancan en **modo automático**: no te preguntará casi nada. Para aprender, pasa a Manual con `Shift+Tab`.
- Ante cualquier cosa rara: `claude doctor`.

Ejercicio

1. Instala Claude Code siguiendo el comando de tu sistema. Tómate tu tiempo; es una vez en la vida.
2. Ejecuta `claude --version` y `claude doctor`. No sigas hasta que los dos respondan bien.

3. Arráncalo y lánzale el encargo de `hola.txt`. Fíjate en qué modo estás. Si es el automático, pasa a Manual con `Shift+Tab` y repite con `hola2.txt` para ver el aviso de permiso antes de decir que sí: eso es el sistema de seguridad presentándose.
4. Escribe `/help` y lee la lista entera una vez, sin intentar memorizarla. Solo para saber qué hay.
5. Crea una carpeta limpia para tu primer proyecto de verdad y arranca `claude` dentro. Mañana la llenamos.

Ya tienes el taller montado y a Claude Code respondiendo. En el próximo capítulo hacemos tu **primer proyecto guiado**: un pequeño programa que te automatiza algo tuyo, de la idea a funcionando.

SIGUE LEYENDO

Ya tienes el taller montado. Ahora, a construir.

En el capítulo 5 haces tu primer proyecto guiado, de la idea a funcionando. Y a partir de ahí: conectar tus agentes a tus datos y servicios con MCP, construir un agente completo, dejarlo corriendo solo a su hora, hacerlo seguro y, en la última parte, ganar dinero con lo que construyes.

Tapa blanda

Amazon.es

Kindle

Amazon.es

¿PREFIERES APRENDERLO CON VÍDEO Y GUÍA?

Máster en Claude es el curso en español que sale de los mismos materiales que este libro: 10 módulos y 88 lecciones con ejercicios, de usar Claude a construir agentes y automatizaciones con Claude Code, la API y MCP.

Conocer el curso en masterenclaude.com

Si este adelanto te ha servido, una reseña honesta en Amazon ayuda a que el libro llegue a más gente.

© 2026 Jorge Fernández Aguila · masterenclaude.com · Adelanto gratuito de «Domina Claude Code», 3.ª edición. Puedes compartir este PDF tal cual, sin modificarlo. Claude y Claude Code son marcas de Anthropic; este libro no está afiliado a Anthropic.