IA MAYÚSCULA

GUÍA PRÁCTICA · NOVIEMBRE DE 2026

ASÍ TRABAJA EL CREADOR DE CLAUDE CODE

12 técnicas para explorar proyectos, preparar cambios y comprobar resultados. Incluye cuatro instrucciones listas para copiar y una plantilla de CLAUDE.md.

UNA CLASE DE 28 MINUTOS, RESUMIDA

BORIS CHERNY

Una guía a partir de su conferencia «Mastering Claude Code in 30 minutes» (mayo de 2025), ampliada con documentación actual.

POR DÓNDE EMPEZAR

Del encargo a la comprobación.

Boris dedica una parte importante de la charla a enseñar cómo dar a Claude pruebas concretas para revisar su propio trabajo.

EL RECORRIDO DE UNA TAREA

EXPLORAR | PLANEAR | CONSTRUIR | PROBAR | AJUSTAR

Claude revisa el proyecto, propone un cambio, lo desarrolla y ejecuta una comprobación. Si aparece un fallo, vuelve al código y lo corrige.

EJEMPLO: REPRODUCIR UNA PANTALLA WEB

01

El encargo

Le enseñas una captura y le pides que implemente esa pantalla en tu proyecto. Claude examina los componentes y propone cómo hacerlo.

02

La comprobación

Una vez construida, abre la página en un navegador y hace una captura. Compara el resultado con la imagen que le pasaste.

03

La segunda pasada

Corrige márgenes, tipografías y elementos que hayan quedado fuera de sitio. Repite la captura y comprueba los cambios.

CONOCER EL PROYECTO

Antes de programar, investiga.

Localiza primero los archivos implicados y los cambios anteriores. Después decide qué modificar.

01 Explora el repositorio

Pídele que siga el recorrido de una función por el proyecto. Te señalará los archivos implicados, las dependencias y las piezas que podrían verse afectadas.

«Encuentra el código del inicio de sesión y explícame qué ocurre desde que envío el formulario hasta que se crea la sesión».

02 Consulta el historial de Git

Los commits cuentan la historia de una decisión. Claude puede revisar cambios anteriores, mensajes y referencias a incidencias para entender por qué una función acabó así.

«Investiga cuándo se añadió este parámetro, qué problema resolvía y qué commits están relacionados».

03 Pídele un plan para cambios grandes

Boris propone discutir primero las opciones. Pide dos o tres soluciones, con los archivos afectados y los riesgos de cada una, y elige después.

«Quiero añadir pagos recurrentes. Estudia el proyecto y preséntame tres opciones. Espera a que elija una para empezar».

04 Divide el trabajo en entregas pequeñas

Una tarea larga se revisa mejor por etapas: investigar, proponer, implementar y comprobar. Así puedes detectar una mala decisión antes de que afecte a más archivos.

«Empieza por localizar el origen del error. Enséñame el diagnóstico y una propuesta de solución».

COMPROBAR

Haz que pruebe lo que construye.

Después de cambiar el código, Claude puede ejecutar pruebas, abrir la aplicación y revisar qué ha pasado.

05 Dale acceso a los tests

Enséñale cómo ejecutar las pruebas, el linter y la comprobación de tipos. Puede usarlos después de cada cambio y reparar los errores que aparezcan.

«Ejecuta las pruebas de este módulo. Si falla alguna, averigua la causa, corrige el código y repite».

06 Comprueba las interfaces con capturas

Para una web, conecta un navegador. Claude puede abrir la pantalla, capturarla y comparar su aspecto con el diseño que le pasaste.

«Abre la página en el navegador, haz una captura y ajusta el diseño para que coincida con la referencia».

07 Escribe un CLAUDE.md útil

Guarda ahí los comandos del proyecto, las convenciones de código y las decisiones arquitectónicas que suele necesitar. Claude Code las leerá como contexto.

«Revisa este repositorio y propón un CLAUDE.md breve con los comandos reales y las reglas que seguimos».

08 Revisa los archivos modificados

Antes de integrar un cambio, mira el diff y comprueba qué tests se ejecutaron. Pide un resumen concreto de lo que ha cambiado y de lo que queda pendiente.

«Muéstrame los archivos modificados, los resultados de las pruebas y cualquier riesgo que hayas detectado».

USO DIARIO

Aprovecha el contexto y las herramientas.

Archivos concretos, herramientas conectadas y procedimientos que se pueden ejecutar una y otra vez.

09 Señala archivos con @

Escribe @ y selecciona el archivo que quieres revisar. Claude recibe así la referencia exacta y puede leerlo dentro del proyecto.

«Mira @src/auth/login.ts y las pruebas relacionadas. Explícame de dónde viene este error».

10 Conecta el navegador y otros servicios

Claude Code puede usar herramientas de terminal y servidores MCP. El navegador, GitHub o las utilidades internas de tu equipo amplían lo que puede hacer.

«Abre esta pantalla con el navegador disponible y guarda una captura para revisar el diseño».

11 Guarda tareas que repites

La charla incluye comandos personalizados. Hoy puedes crear skills para procedimientos como revisar una pull request o preparar una publicación.

«Prepara una skill que ejecute tests y lint, revise el diff y redacte el resumen de una pull request».

12 Separa las tareas paralelas

Puedes tener varias sesiones trabajando a la vez. Los worktrees de Git ayudan a que cada una trabaje en su propia rama y sus propios archivos.

«Propón cómo repartir estos tres cambios en worktrees independientes y qué comprobar antes de unirlos».

PROMPTS

Cuatro instrucciones para tu próxima sesión.

Adáptalas al proyecto y añade los requisitos concretos de tu tarea.

01 / ENTENDER EL PROYECTO

Revisa el repositorio y encuentra los componentes relacionados con [TAREA]. Explícame cómo se conectan, qué dependencias tienen y qué archivos habría que revisar. Presenta primero tus conclusiones.

02 / PREPARAR EL CAMBIO

Para implementar [TAREA], prepara dos o tres propuestas. Indica los archivos afectados, las ventajas, los riesgos y cómo comprobaríamos que funciona. Recomienda una y espera mi aprobación.

03 / IMPLEMENTAR Y PROBAR

Implementa el plan aprobado. Ejecuta los tests que correspondan y corrige los fallos. Si has cambiado una interfaz, ábrela en el navegador y compárala con la referencia. Repite la comprobación tras los ajustes.

04 / EXPLICAR EL RESULTADO

Entrega un resumen de los archivos que has cambiado, las pruebas que has ejecutado y sus resultados. Señala cualquier comprobación pendiente y cualquier riesgo que deba revisar antes de integrar el cambio.

Para que funcione mejor: sustituye [TAREA] por una acción concreta. Añade qué debería pasar al terminar y cómo vas a comprobarlo: un test, una captura o un comportamiento observable.

PLANTILLA

Prepara tu CLAUDE.md.

Guárdalo en la raíz del repositorio y rellénalo con datos reales. Puedes empezar con esta estructura y completarla a medida que trabajas.

```
# Proyecto
- Producto: [qué hace esta aplicación]
- Tecnologías: [stack real]

# Comandos habituales
- Arrancar: [comando real]
- Tests: [comando real]
- Tipos: [comando real]
- Lint: [comando real]

# Cómo trabajamos
- [convención específica del proyecto]
- [estructura que deben seguir los módulos]
- [decisión técnica importante]

# Antes de dar una tarea por cerrada
- [pruebas que debe ejecutar]
- [pantallas o flujos que debe verificar]

# Precauciones
- [archivos o acciones con revisión manual]
```

Qué merece la pena guardar: información propia del proyecto que Claude necesitará en distintas sesiones. Actualiza las reglas cuando cambie la forma de trabajar. Guarda las credenciales fuera del repositorio y reserva las skills para procedimientos largos.

PARA PRACTICAR

Pruébalo hoy en un proyecto.

Elige una tarea pequeña que tengas pendiente y sigue estos pasos.

1. Abre Claude Code en la carpeta del proyecto.
2. Pídele que localice y explique los archivos relacionados.
3. Solicita dos soluciones y revisa el plan antes de elegir.
4. Decide qué prueba o captura confirmará el resultado.
5. Deja que implemente y revisa después el diff y las pruebas.
6. Guarda en CLAUDE.md las reglas que volverás a utilizar.

FUENTES PARA SEGUIR APRENDIENDO

- **Charla de Boris Cherny · vídeo compartido**
- **Charla publicada por Anthropic en 2025**
- **Claude Code · buenas prácticas**
- **Claude Code · memoria y CLAUDE.md**
- **Claude Code · skills**

MÁS SOBRE INTELIGENCIA ARTIFICIAL

WEB / ARTÍCULOS	YOUTUBE / VÍDEOS	INSTAGRAM / REELS
iamayuscula.com	@iamayuscula	@iamayuscula

Guía de IA Mayúscula a partir de una charla de Boris Cherny. Publicación independiente.