El potencial de Claude Code y las terminales

De cero a dejar que la IA construya, investigue y enseñe por ti

Un agente que lee, escribe, ejecuta y verifica. Versión definitiva, con demos en vivo.

Claude

Anthropic · Claude Code

Eduardo C. Garrido-Merchán

Profesor Propio Adjunto, Métodos Cuantitativos, ICADE

ecgarrido@comillas.edu · *Madrid, 2026*

COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI ICADE CIHS

El plan de hoy.

1. Fundamentos
2. Construir
3. Investigar
4. Orquestar
5. Conectar
6. Docencia
7. Criterio
8. Soberanía

Una hora, con demos en directo, para enseñaros hasta dónde

En la próxima hora me veréis hacer esto, en directo.

🚀 **Construir.** Una figura de paper reproducible y un método numérico verificado contra su solución cerrada, desde una frase.

📖 **Investigar.** Un *survey* automático de una carpeta de papers, un artículo difícil vuelto dossier y un análisis científico reproducible.

🕸️ **Orquestar.** Tres *referees* adversariales revisando tu manuscrito y un *pipeline* que escribe un paper de principio a fin.

🔌 **Conectar.** Enchufar Claude a tu Slack, tu Gmail y tus herramientas con MCP, con un estándar abierto.

🎓 **Docencia.** `/teaching` para materiales y Kahoots, y slides interactivas en HTML para tus alumnos.

⭐ **Tu mundo.** `CLAUDE.md` y 47 agentes que trabajan a tu manera, versionados en archivos.

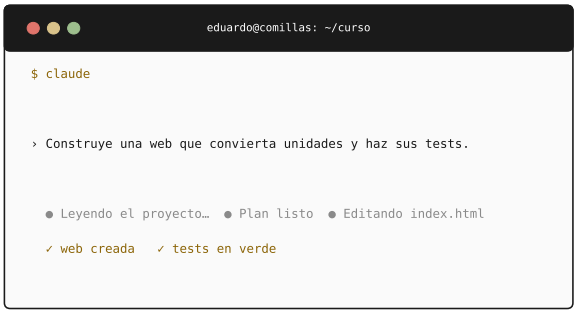
Idea clave. No es autocompletar código. Es delegar tareas completas de investigación y docencia, y revisarlas.

Parte 1 / 8

Fundamentos

Claude

🔧 La terminal ha vuelto, y ahora habla tu idioma.

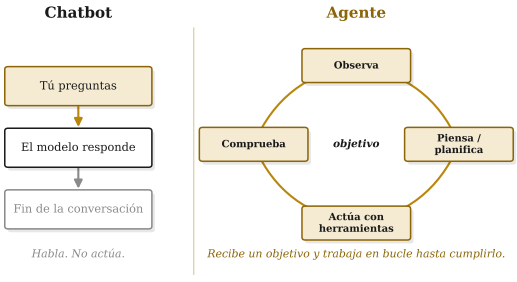


Qué es. La terminal es esa ventana de texto donde le das órdenes directas al ordenador. Daba respeto: comandos crípticos, nada de ratón.

Qué ha cambiado. Ahora dentro de la terminal vive un agente al que le hablas en español llano. Él traduce tu intención en acciones.

Idea clave. La barrera de entrada ya no es la sintaxis: es saber qué pedir y cómo revisarlo.

🤖 De chatbot a agente: ese es el salto.



El chatbot responde y calla. La conversación es el producto.

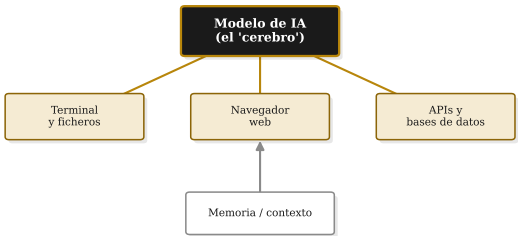
El agente recibe un objetivo y trabaja: observa, planifica, usa herramientas, comprueba y repite hasta lograrlo.

IDEA CLAVE. La diferencia no es un modelo más listo: es darle manos y un bucle para usarlas.

⚙️ Tres ingredientes convierten un modelo en un agente.

Tres ingredientes que un chatbot no tiene:

uso de herramientas · planificación · autonomía



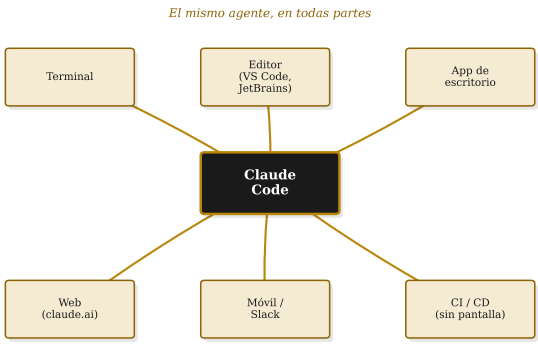
Herramientas. Lee y escribe ficheros, ejecuta comandos, navega, consulta bases de datos.

Planificación. Descompone la tarea y decide el siguiente paso según lo que ve.

Autonomía. Encadena los pasos sin que le lleves de la mano.

Idea clave. Claude Code es exactamente esto: un modelo Claude con manos dentro de tu ordenador.

Claude es el cerebro; Claude Code es el agente.



Claude. La familia de modelos de IA de la empresa Anthropic.

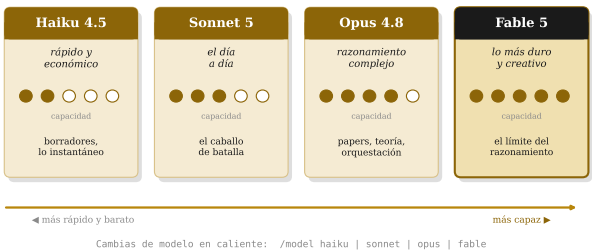
Claude Code. El agente que pone a Claude a trabajar sobre tus archivos: el mismo agente en el terminal, el editor, la app de escritorio, la web, tu Slack e incluso sin pantalla, dentro de tu CI.

IDEA CLAVE. Aprendes una herramienta y la usas en todas partes.

NUEVO

Los nuevos Claude: una familia, un modelo por trabajo.

La familia Claude 5 (2026): un modelo para cada trabajo



Elige según la tarea. `Haiku` para lo rápido y barato; `Sonnet` para el día a día; `Opus` para razonar fuerte; `Fable` para lo más difícil y creativo.

Sin reiniciar. Cambias de modelo en mitad de la sesión con `/model`.

Idea clave. Mismo agente, distinta potencia bajo el capó según lo que cueste el problema.

🚀 Arrancar es una palabra: `claude`.

1. Lanzarlo. En la carpeta de tu proyecto, escribes `claude` y se abre la conversación. Le hablas en lenguaje natural.

2. Que conozca el proyecto. El comando `/init` hace que Claude lea tu código y escriba un `CLAUDE.md`: un resumen del proyecto y tus reglas, que recordará en cada sesión.

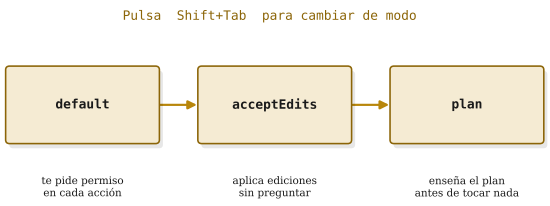
3. Pedir. A partir de ahí, describes objetivos: "añade un botón de exportar a PDF y sus tests".

```
> claude
```

```
> /init
```

Idea clave. Dos comandos y ya estás dictando trabajo. El resto es saber pedir y revisar.

🛡️ Seguridad primero: tú mandas, él propone.



Empieza siempre en plan o default: tú mandas, él propone.

Por defecto pide permiso. No toca nada sin tu visto bueno. Ves cada cambio antes de aceptarlo.

Modo plan. Con `Shift+Tab` le pides que te enseñe la estrategia completa *antes* de editar.

IDEA CLAVE. Probarlo es seguro: trabaja siempre sobre una copia y revisa antes de aplicar.

Parte 2 / 8

Construir

DEMO EN VIVO

De una frase a una figura de paper reproducible.

Lo que tecleo:

```
> Crea un script en Python que dibuje un proceso gaussiano: media posterior, banda de incertidumbre y puntos observados. Paleta sobria, ejes con unidades. Y un test que compruebe que se genera el PDF y que la media interpola los datos.
```

Y luego:

```
> Genérala y enséñame el PDF.
```

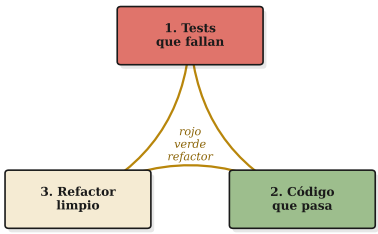
Qué veréis. Claude escribe el script, lo ejecuta, produce la figura y deja un test en verde que la valida.

El momento. De una frase en español a una figura lista para el paper, en un par de minutos.

Idea clave. No es "una gráfica bonita": es una figura *reproducible* y comprobada.

DEMO EN VIVO

Un método numérico, verificado contra la solución cerrada.



Claude escribe los tests primero, luego el código, y los pone en verde.

Lo que tecleo:

```
> Implementa value iteration para un MDP tabular. Primero un test que compare el valor óptimo con la solución cerrada de un MDP de dos estados; luego la implementación; y ejecútalo hasta que pase.
```

Qué veréis. El test sale en rojo, Claude implementa, se corrige solo y termina en verde: el valor calculado coincide con el óptimo analítico.

Idea clave. Lo asombroso no es el código: es que se autocorrige contra una verdad conocida.

✅ Por qué los tests lo cambian todo en investigación.

El test es el contrato. Si le pides "que pasen los tests", le has dado un criterio objetivo de éxito. El agente itera contra ese criterio sin que tú supervises cada paso.

Confianza verificable. No te crees que funciona: lo ves en verde, contrastado con una solución cerrada, un límite analítico o una implementación de referencia.

Esto es reproducibilidad. Semillas fijas, asertos de forma y de finitud, comparación con la verdad conocida: el mismo rigor que exiges a un experimento de tu paper.

Idea clave. Pedir el criterio de éxito junto con la tarea es la habilidad número uno.

Parte 3 / 8

Investigar

📖 El caso que de verdad os interesa: la literatura.

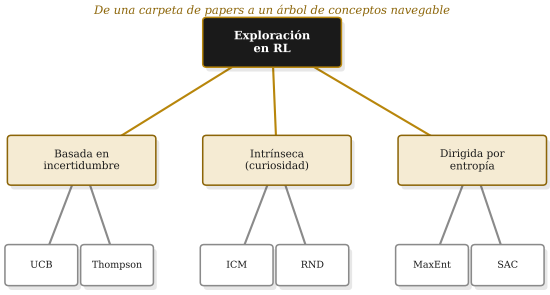
El problema de siempre. Tienes una carpeta con veinte PDFs de un tema y dos horas para entender el panorama.

Lo que puede hacer Claude. Leer toda la carpeta, identificar las familias de métodos, situar cada trabajo y devolverte una *taxonomía* con sus relaciones, no un resumen plano.

Resumir vs. sintetizar. Resumir es acortar cada texto; sintetizar es cruzarlos y encontrar el esqueleto común: qué enfoques existen, en qué se oponen y qué hueco queda.

Idea clave. El agente no resume: estructura. Y la estructura es lo que cuesta.

DEMO EN VIVO Un *survey* automático: de una carpeta de papers al estado del arte.



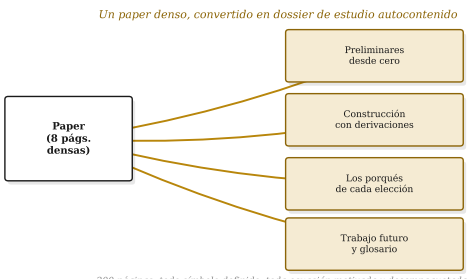
Lo que tecleo:

```
>Lee la carpeta papers/, sintetiza el tema y construýeme una taxonomía de métodos como árbol en Markdown y como JSON, con el hueco que deja cada familia.
```

Qué veréis. Un árbol de familias y subfamilias con cada trabajo en su rama y un fichero reutilizable: el esqueleto de la sección de estado del arte.

Idea clave. De veinte abstracts a un mapa del campo en lo que tarda un café en enfriarse.

DEMO EN VIVO `paper_explained`: un paper denso, vuelto un dossier de estudio.



Lo que tecleo:

```
> Usa la skill paper_explained sobre este artículo: explícamelo desde cero, con todas las derivaciones y un glosario.
```

Qué veréis. Un documento extenso y autocontenido: preliminares, construcción paso a paso, los *porqués* de cada decisión y el trabajo futuro.

Idea clave. Sirve para entender tú un paper difícil, o para que tus alumnos lo entiendan.

NUEVO

Ciencia: de la hipótesis a la figura reproducible.

Ciencia: de la hipótesis a la figura reproducible y verificada

Claude no inventa resultados: los deriva, los computa y los verifica contra una verdad conocida.

El agente como colaborador. Formula el análisis, lo implementa, lo ejecuta y verifica el resultado contra una verdad conocida antes de dibujarlo.

Rigor, no adornos. Semillas fijas, tests en verde, intervalos de confianza y repeticiones: nada de números inventados.

Idea clave. Claude no "recuerda" resultados: los deriva, los computa y los comprueba.

DEMO EN VIVO

Bibliografía sin alucinaciones: descargar, validar, repetir.

Cada campo de cada referencia, descargado y verificado contra la fuente

```
referencias.bib
@article{garrido2024,
  author  = {Garrido-Merchán, E.},
  title   = {{B}ayesian optimization...},
  journal = {JMLR},
  volume  = {25}, number = {3},
  pages   = {1--34}, year = {2024},
  doi     = {10.5555/xxxx}}
```

campo	vs. fuente
author	✓ coincide
title	✓ coincide
journal	✓ coincide
volume / number	✓ coincide
pages	✓ coincide
year	✓ coincide
doi	✓ coincide

4 agentes verificadores independientes, en rotación · 0 campos alucinados · fuente = Google Scholar / DOI

Lo que tecleo:

```
> Valida referencias.bib: descarga cada fuente, compara campo a campo con un script y bloquea si algo no cuadra.
```

Qué veréis. Un verificador mecánico contrasta autor, año, volumen, páginas y DOI contra la fuente; cuatro agentes independientes repiten la auditoría.

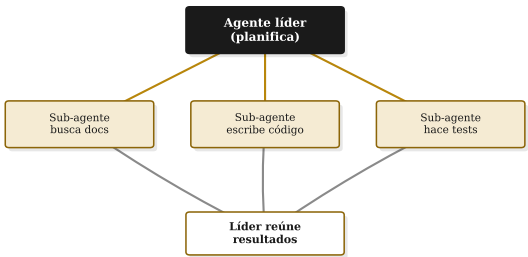
Idea clave. Ni un campo inventado: la regla de oro para citar sin miedo.

Parte 4 / 8

Orquestar

Subagentes: un agente que dirige a un equipo.

Varios agentes en paralelo, como un equipo que reparte el trabajo



La idea. Un agente líder reparte el trabajo entre varios subagentes, cada uno con su contexto y su especialidad.

Aislamiento. Cada uno trabaja en su hilo y devuelve solo un resumen: el contexto del líder no se contamina con el detalle.

Idea clave. Como un equipo humano que se divide la tarea: más rápido, cada uno a lo suyo.

📁 Un subagente es un archivo `.md` con reglas.

`.claude/agents/exam-author.md`

```
---
name: exam-author
description: Redacta un examen completo a
partir del blueprint: ítems con puntos,
rúbrica, versión B paralela y solucionario.
tools: Read, Edit, Write
---

Redactas un examen sumativo a partir de un
blueprint. Cada ítem lleva puntos, tiempo,
rúbrica de corrección y solución completa.
La versión B examina lo mismo con otros
números. Solo materia de la lista impartida.
```

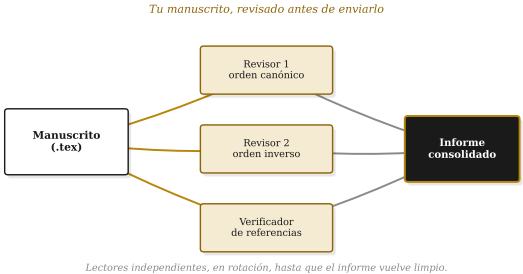
Frontmatter. Nombre, descripción de *cuándo* invocarlo y qué herramientas puede tocar.

Cuerpo. Sus instrucciones permanentes y su contrato de salida.

Idea clave. Un especialista se define escribiéndolo, no programándolo.

DEMO EN VIVO

Tres revisores adversariales leen tu manuscrito a la vez.



Lo que tecleo:

```
>Revisa borrador.tex con tres revisores en paralelo: uno en orden canónico, uno empezando por los experimentos y un verificador de referencias. Informe único por severidad.
```

Qué veréis. Tres *referees* adversariales, cada uno con su lente, encuentran cosas distintas; el líder las consolida.

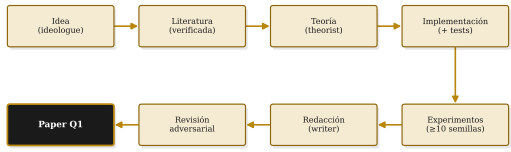
IDEA CLAVE. Te revisan el paper *antes* de enviarlo, como un panel de área chairs.

DEMO EN VIVO

De una orden a un paper de principio a fin.

```
> /bo: propon una alternativa a EI, verifica su novedad, formaliza la teoría, impleméntala con tests, corre experimentos y redáctame el manuscrito.
```

Una orden → un paper: el pipeline de agentes de extremo a extremo

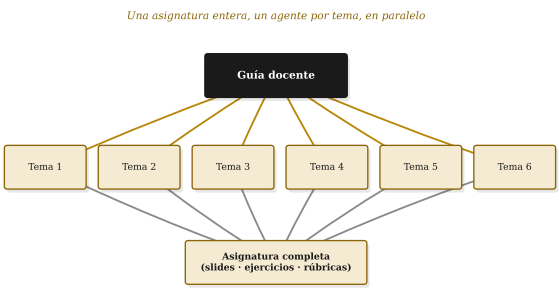


Cada etapa es un subagente especializado; la revisión y la verificación bloquean hasta quedar limpias.

IDEA CLAVE. Una *skill* (`/bo`, `/rl`, `/philosophy`) encadena una decena de subagentes especializados; cada etapa bloquea hasta quedar limpia.

DEMO EN VIVO

Paralelizar una asignatura entera: un agente por tema.



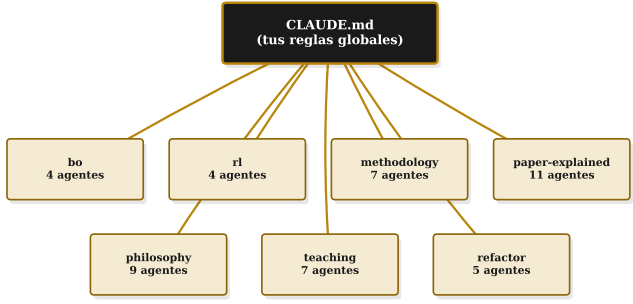
Lo que tecleo:

```
>ultracode: construye el material de los seis temas de la asignatura en paralelo, un agente por tema, cada uno con sus slides y tres ejercicios. Consolida un índice.
```

Qué veréis. Varios agentes construyen temas distintos a la vez y un líder reúne todo en un índice.

Idea clave. Escala que una sola conversación no alcanza: una asignatura entera en paralelo.

La misma idea a escala: 47 agentes en 7 *pipelines*.



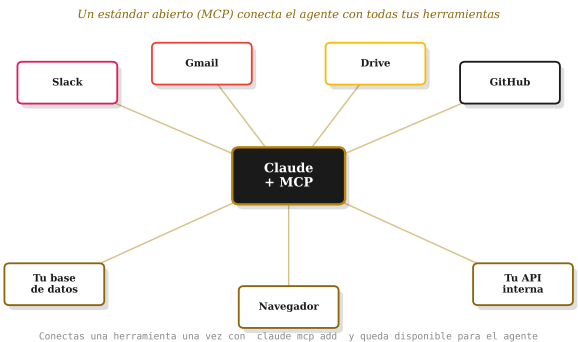
7 pipelines · 47 subagentes especializados · todo versionado en archivos

Idea clave. Cada *pipeline* (`bo`, `rl`, `methodology`, `philosophy`, `paper-explained`, `teaching`, `refactor`) es una cadena de subagentes especializados; todos heredan el mismo `CLAUDE.md` y viven versionados en archivos.

Parte 5 / 8

Conectar

🔌 MCP: el “USB-C” que conecta Claude a tus herramientas.



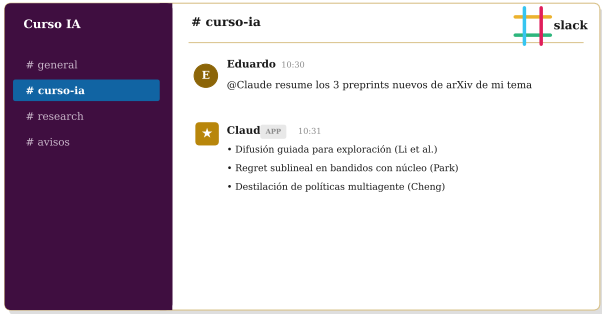
El problema. Cada herramienta (Drive, Slack, Gmail, tu base de datos) habla un idioma distinto.

La solución. *MCP*, un estándar abierto creado por Anthropic, hoy adoptado por todo el sector y bajo la Linux Foundation.

Cómo se añade. Un comando: `claude mcp add`.

Idea clave. Conecta una herramienta una vez y queda disponible para el agente.

NUEVO Slack: el agente vive donde ya trabaja tu equipo.



El mismo agente, respondiendo desde el móvil o el navegador, en tu Slack

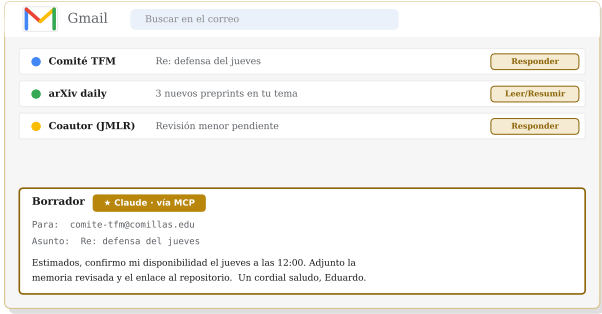
Lo mencionas y responde. Etiquetas a `@Claude` en un canal y trabaja ahí mismo: resume, busca, redacta, sin abrir el portátil.

Mismo agente, otra superficie. Es Claude Code, no un bot aparte: hereda tus reglas y tus herramientas conectadas.

Idea clave. Delegar deja de exigir que estés en la terminal: le hablas desde el móvil.

NUEVO

Gmail vía MCP: triaje de bandeja y borradores listos.



Claude lee, etiqueta y prepara borradores; tú solo revisas y envías

Lo que tecleo:

```
>Revisa mi bandeja, etiqueta lo urgente y prepárame borradores de respuesta; no envíes nada.
```

Qué veréis. Claude clasifica los hilos y deja borradores redactados; tú solo revisas y pulsas enviar.

Idea clave. La regla de oro: prepara, no envía. El humano aprueba lo que sale.

⚙️ Hooks: automatiza lo que se repite.

Un gancho dispara una orden tuya tras cada acción del agente

Qué es un hook. Una orden tuya que se dispara automáticamente cuando el agente hace algo: antes o después de editar, al terminar, al empezar.

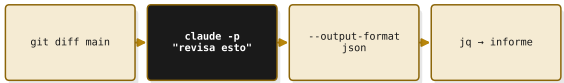
Ejemplo típico. "Cada vez que edites un archivo, pásale el formateador." Se configura en `.claude/settings.json`.

Idea clave. Tus estándares dejan de depender de que el agente se acuerde: se aplican solos.

DEMO EN VIVO

Claude sin pantalla: dentro de un script y de tu CI.

Claude como un comando más de tu 'pipeline': sin pantalla, en CI



El mismo contenido, vestido para otra revista

Plantilla, longitudes de sección, estilo de cita y bibliografía: todo adaptado de una pasada.

Modo *headless*. `claude -p "tarea"` ejecuta una orden sin abrir conversación y devuelve el resultado, encadenable con otros comandos.

Un caso real. `claude -p` reformatea un paper del estilo de una revista al de otra, o revisa un *diff* y bloquea el *merge* si hay un bug crítico.

Idea clave. Claude deja de ser una app y pasa a ser una pieza de tus automatizaciones.

Parte 6 / 8

Docencia


📁 Todo lo que personaliza a Claude vive en archivos.

Todo lo que personaliza a Claude vive en archivos del repo

```
▸ proyecto/
    • CLAUDE.md                          tus reglas siempre presentes
    ▸ .claude/
        ▸ agents/                        tus subagentes
            • revisor-seguridad.md
        ▸ skills/                        flujos reutilizables
            • nuevo-ejercicio/SKILL.md
        • settings.json                  hooks y permisos
        • .mcp.json                      servidores MCP
```

`CLAUDE.md`. Tus reglas, presentes en cada sesión.

`.claude/agents/`. Tus subagentes especializados.

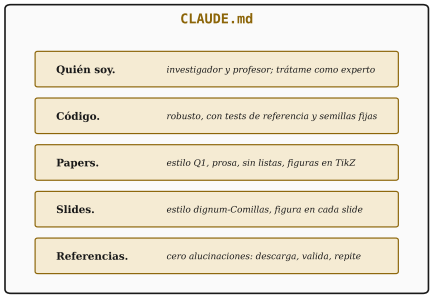
`.claude/skills/`. Flujos reutilizables que empaquetas una vez.

`settings.json` y `.mcp.json`. Hooks, permisos y herramientas conectadas.

Idea clave. Versionas estos archivos con tu código: tu configuración viaja con el proyecto y con tu equipo.

📖 CLAUDE.md: mi constitución, escrita una sola vez.

Una constitución que el agente lee al empezar cada sesión

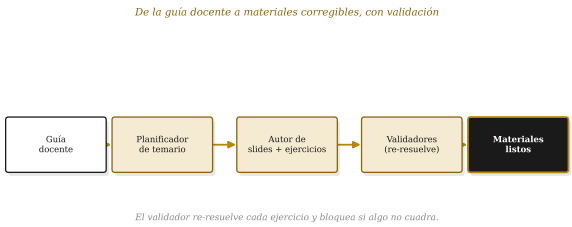


Qué es. Un archivo donde escribes cómo quieres que trabaje, y que Claude lee al empezar cada sesión. No repites instrucciones.

El mío, por bloques. Reglas de código (tests de referencia, semillas fijas), de *papers* Q1 (prosa, sin listas, figuras en TikZ), de *slides* (este mismo estilo) y una política de *cero alucinaciones* en bibliografía.

Idea clave. Cinco minutos escribiéndolo ahorran cien correcciones después.

DEMO EN VIVO /teaching: de la guía docente a slides, ejercicios y rúbricas.



Lo que tecleo:

```
>Usa /teaching: del tema de árboles de decisión, genera slides, tres ejercicios con solución y rúbrica, y un Kahoot de 8 preguntas.
```

Qué veréis. El planificador, el autor y los validadores en cadena; salen materiales en *tu* formato y un Kahoot listo para importar.

Idea clave. Tu metodología, reproducible, validada y delegable.

DEMO EN VIVO

Ejercicios y exámenes con rúbrica, solución y versión B.

Un examen completo: ítems con puntos, rúbrica, solución y versión B

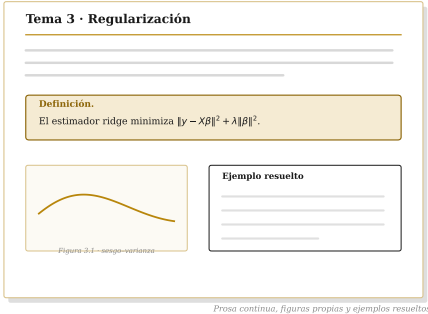
Lo que tecleo:

```
>Usa /examenes: un examen de 4 ítems con puntos y tiempo, rúbrica de corrección, solucionario y una versión B de igual dificultad.
```

Qué veréis. El *blueprint* se convierte en un examen completo, con su rúbrica y su solucionario, y una convocatoria paralela lista.

Idea clave. Generas y *varias* evaluación sin perder consistencia ni criterio.

DEMO EN VIVO /apuntes: de un temario y sus fuentes a apuntes con figuras.



Prosa continua, figuras propias y ejemplos resueltos: apuntes, no un resumen.

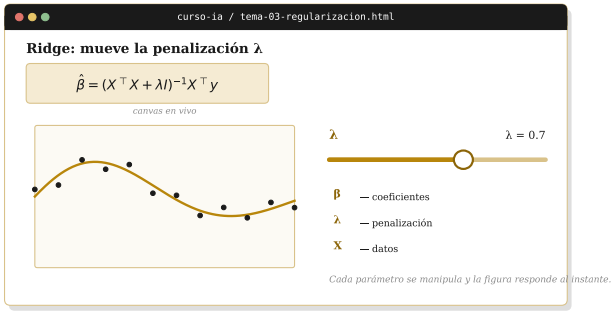
Lo que tecleo:

```
> Usa /apuntes: teje el temario y estas fuentes en apuntes de la asignatura, en prosa continua, con definiciones, ejemplos resueltos y figuras propias.
```

Qué veréis. Un capítulo de apuntes coherente: prosa, no viñetas; definiciones enmarcadas, ejemplos y figuras generadas.

IDEA CLAVE. Apuntes de verdad, no un resumen: material que el alumno estudia.

DEMO EN VIVO Slides interactivas en HTML para que el alumno explore.



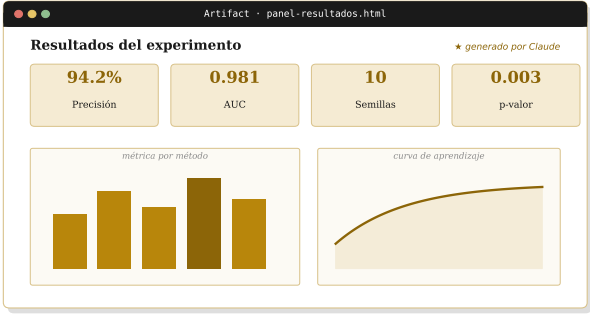
Lo que tecleo:

```
> Usa interactive-slides sobre el tema de regularización: cada parámetro manipulable en vivo, con canvas.
```

Qué veréis. Un *deck* HTML autocontenido donde el alumno mueve λ y la figura responde al instante.

Idea clave. Material que el alumno *explora*, no solo lee.

NUEVO Design: Claude también diseña (artifacts y *dataviz*).



Artifacts. Le pides un panel y devuelve una página interactiva autocontenida, lista para compartir, con *tu* paleta.

Buen gusto por defecto. Skills de `dataviz` y `artifact-design` fijan tipografía, color y accesibilidad: gráficas que se leen como un sistema.

Idea clave. No solo calcula el resultado: lo presenta con criterio de diseño.

Parte 7 / 8

Criterio

👁 Cómo no perder el control: plan, revisión y vuelta atrás.

Planifica antes. En tareas grandes, pídele el plan primero (`Shift+Tab`) y apruébalo antes de que toque nada.

Revisa los cambios. Cada edición es visible y reversible; trabaja en una rama o una copia y mira el *diff*.

Punto de control. Puedes deshacer y volver a un estado anterior si una dirección no te convence.

Idea clave. Autonomía no es ceguera: tú marcas el rumbo y revisas las entregas.

⚠️ Lo que NO es verdad (para que no os engañen).

“Es instantáneo.” No: lee, razona y pide permiso; tarda segundos por paso, y está bien que así sea.

“Los subagentes corren a la vez que tu sesión.” El líder los lanza y espera su resumen; tú sigues hablando con el líder.

“MCP es de Claude.” Es un estándar abierto que usan muchas herramientas.

“CLAUDE.md se relee en cada mensaje.” Se carga al empezar la sesión; si lo cambias a media sesión, hay que recargarlo.

Idea clave. Entender los límites es lo que separa usarlo bien de frustrarse con él.

Poder con cabeza: cuatro cosas que vigilar.

Se equivoca. Actúa en cadena; revisa antes de aplicar.

Cuesta. Las tareas largas consumen tokens: usa el modelo justo y pon límites.

Inyección. Texto malicioso puede "secuestrarlo"; azíslalo de datos sensibles.

Autonomía. Permisos mínimos y humano en el bucle.

Idea clave. La regla de oro: tú apruebas lo que importa.

🚀 Cómo empezar mañana por la mañana.

1. Instala y arranca. Abre una terminal en una carpeta de prueba y escribe `claude`.

2. Una tarea acotada y con criterio. "Creáme un script que lea este CSV, haga esta gráfica y tenga un test que compruebe el resultado."

3. Escribe tu `CLAUDE.md`. Diez líneas con tus reglas. A partir de ahí, todo es más tuyo.

4. Sube el listón. Cuando te fíes, prueba subagentes, una skill, un servidor MCP y tu Slack.

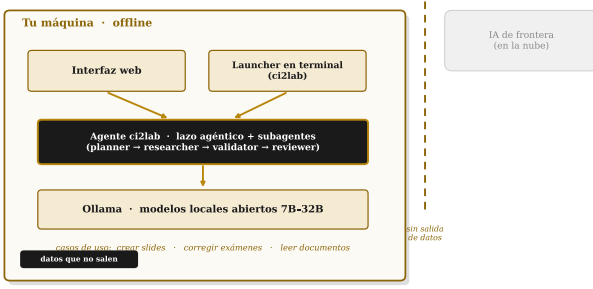
Idea clave. La competencia de 2026 no es programar: es saber encargar, dar criterio y revisar.

Parte 8 / 8

Soberanía

CI2LAB INVESTIGACIÓN
COMILLAS ICAI

🔒 Y si mañana cambia el modelo: soberanía sobre tu IA.

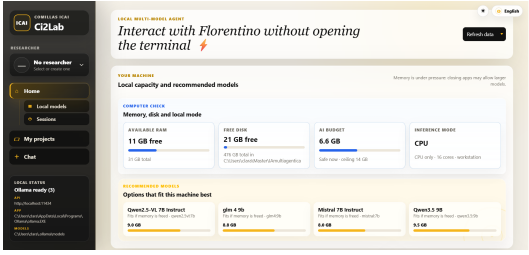


El escenario. Si la IA de frontera pasa de tarifa plana a *cobro por uso*, o si el dato no puede salir de la organización, necesitas una alternativa que gobiernes tú.

Florentino. La misma idea agéntica, pero *en local*: un agente multi-modelo sobre `Ollama` con modelos abiertos de 7B–32B que elige según tu RAM y tu GPU.

Nuestra arquitectura. Del CI2LAB (Cátedra de Industria Inteligente, Comillas ICAI): mismo lazo agéntico y mismos subagentes, sin que un byte salga de tu máquina.

👁 Con o sin terminal: dos formas de usar Florentino.



Interfaz web: elige modelo según tu máquina

Launcher `ci2lab` en la terminal

Idea clave. Casos de uso: crear slides desde notas, corregir exámenes con rúbrica y feedback, y resumir documentos largos. Todo en local.

⚖️ El trade-off: qué cedes y qué ganas al llevarla a local.

Usar IA agéntica en local implica un trade-off

Lo que se cede	Lo que se gana
Potencia bruta un modelo local (7B–32B) razona peor que uno de frontera en la nube	**Privacidad total** tus prompts, ficheros y datos nunca salen de la máquina
Techo de hardware lo que ejecutas depende de tu RAM y tu GPU	**Independencia** sin API key, sin suscripción, sin límites, sin lock-in; offline
Mantenimiento hay que preparar el entorno y mantenerlo tú	**Confianza verificable** código abierto y comprobaciones: fiable incluso con un modelo modesto

Idea clave. *"Un asistente que trabaja para ti, sin que tus datos salgan de casa."* Florentino, por el CI2LAB (J. Alonso, T. Contreras, C. Lucena, A. Rodríguez, P. Sánchez), Comillas ICAI.

Gracias. ¿Lo probáis?

El potencial de Claude Code y las terminales

Eduardo C. Garrido-Merchán

ecgarrido@comillas.edu