

Codex avanzado para perfiles técnicos · Cuaderno de prácticas · 2 sesiones de 3 horas

Cuaderno de prácticas · 2 sesiones de 3 horas

Cibernàrium · Barcelona Activa · Formador: Daniel M. Casas · Octubre de 2026

Este cuaderno recoge las nueve prácticas del curso con sus explicaciones, los prompts y comandos de referencia, las comprobaciones con las cifras exactas y los enlaces a la documentación oficial de OpenAI, revisada en octubre de 2026. Todo se hace sobre el repositorio de Cafès Roca (codex-avancat-6h-repositori-cafes-roca.zip). Si te quedas atrás, cada práctica tiene una etiqueta de Git con su resultado en la rama referencia.

Sesión 1 • Crear con Codex

De una idea a algo que funciona: una landing desde un brief, un análisis de datos reproducible y un dashboard, todo con método y guardado en Git.

Cómo se organiza la sesión 1

Método de trabajo, caso y repositorio.

Cada práctica sigue el mismo ciclo: explorar, planificar, implementar, verificar y guardar con un commit. Codex escribe el código; tú decides qué se construye, revisas el plan, compruebas el resultado y aceptas o rechazas cada cambio.

Paso	Qué haces tú	Qué hace Codex
Explorar	Le pides que lea el proyecto y te lo explique	Abre ficheros, ejecuta comandos de lectura y resume
Planificar	Activas /plan, respondes sus preguntas y corriges el plan	Propone pasos, ficheros afectados y cómo comprobarlo
Implementar	Apruebas lo que pide permiso	Edita ficheros y ejecuta comandos dentro del sandbox
Verificar	Miras el resultado en el navegador y contrastas cifras	Ejecuta pruebas, abre la página y corrige
Guardar	Revisas el diff y haces el commit	Prepara el mensaje del commit y, si se lo pides, la rama y la PR

El caso: Cafès Roca

Una cadena ficticia de tres cafeterías de barrio en Barcelona: Gràcia, Poblenou y Sants. Es el mismo caso del curso básico, ahora con un repositorio de código: datos de ventas del primer semestre, opiniones de clientes, el brief de la web y una aplicación interna de gestión de estoc.

- [Descargar el repositorio codex-avancat-6h-repositori-cafes-roca.zip](#)

Carpeta	Contenido
dades/	Ventas de enero a junio de 2026 (CSV, 2.487 líneas), opiniones de clientes, brief web y acta de reunión
estoc/	Aplicación de estoc en Node.js sin dependencias, con pruebas e incidencias abiertas en ISSUES.md
web/	Carpeta de la landing. Dos artículos en Markdown esperando a convertirse en páginas

Etiquetas de rescate. El repositorio lleva una etiqueta de Git con el estado final de cada práctica (p1-posada-a-punt ... p9-automatitzar) en la rama referencia. Si te quedas atrás o se te acaba el límite de uso, sigues desde el punto del grupo con `git switch -c mi-p4 p4-dashboard`.

Bloque 1 • Codex: el agente para proyectos de software

Dónde vive, cómo trabaja, modelos y planes.

Codex es el agente de programación de OpenAI. Lee un proyecto, planifica los cambios, edita ficheros, ejecuta comandos y pruebas en un entorno aislado y trabaja con Git y GitHub. Desde el 9 de julio de 2026 es uno de los tres modos de la aplicación de escritorio de ChatGPT: Chat, Work y Codex.

Dónde vive

Superficie	Para qué	Dónde se ejecuta
App de escritorio de ChatGPT, modo Codex (Alt+3 en Windows y Linux; ⌘3 en Mac)	La forma principal de trabajar en el curso: proyectos, diffs, navegador integrado, worktrees, commits y tareas programadas	En tu ordenador, o en la nube si eliges «Cloud»
Codex CLI (codex en la terminal)	Scripts, servidores, automatización con codex exec, trabajo remoto por SSH	En tu ordenador
Extensión para VS Code (también Cursor y Windsurf)	Trabajar al lado del código, con los ficheros abiertos como contexto	En tu ordenador o delegando en la nube
Codex cloud	Tareas largas o en paralelo sobre un repositorio de GitHub, con varios intentos y pull requests	En una máquina virtual de OpenAI
Móvil (iOS)	Lanzar tareas en la nube y seguir desde la pestaña Remote los chats del escritorio	Nube o tu ordenador

Cómo funciona por dentro. Codex trabaja en un bucle. Recibe tu petición junto con las instrucciones de AGENTS.md, decide qué herramienta usar (leer un fichero, ejecutar un comando, aplicar un parche, buscar en la web), mira el resultado y vuelve a decidir, hasta que considera la tarea terminada. Cada comando se ejecuta dentro de un **sandbox**: un perímetro que limita qué carpetas puede escribir y si puede salir a internet. Cuando necesita salir de ese perímetro, te pide permiso.

Modelos y razonamiento (octubre de 2026)

Modelo	Cuándo usarlo	Planes
GPT-6.1 Sol	El recomendado para programar y para tareas agénticas. Rendimiento cercano a Astra con menos coste	Plus, Pro, Business, Enterprise, Edu
GPT-6 Astra	El más capaz, para problemas difíciles. Gasta el límite mucho más rápido	Limitado en Plus; amplio en Pro
GPT-6 Sol	Alternativa equilibrada	Plus y superiores
GPT-6 Luna	Rápido y eficiente para tareas acotadas y de volumen	Todos, también Free y Go

El **nivel de razonamiento** decide cuánto piensa el modelo antes de actuar: Light, Medium, High, Extra High, Max y Ultra. **Medium** va bien para casi todo. Sube a High o Extra High para depurar un error difícil o planificar una arquitectura; Ultra reparte el trabajo entre subagentes en paralelo. Más razonamiento significa más tiempo y más consumo. GPT-5.5 se retira de Codex el 14 de octubre de 2026.

Planes y límites

Plan	Codex	Precio
Free y Go	App de escritorio con GPT-6 Luna, límites pensados para tareas cortas	0 \$ y 8 \$/mes
Plus (recomendado para el curso)	App, web, CLI, IDE, iOS y nube; revisión automática de código en GitHub; GPT-6.1 Sol, GPT-6 Sol, Luna y Astra limitado	20 \$/mes
Pro	Todo lo de Plus con mucho más uso; actualmente sin límite de cinco horas	Desde 100 \$/mes
Business, Enterprise y Edu	Administración, SSO y no entrenan	Desde 20 \$/usuario/mes

Plan	Codex	Precio
	con tus datos por defecto. El administrador puede limitar funciones	

El uso se mide en mensajes por **ventanas de cinco horas**, con un máximo semanal. Lo local y lo que se ejecuta en la nube salen de la misma bolsa, que también comparten Work y ChatGPT for Excel. Como referencia, en Plus OpenAI estima entre 15 y 160 mensajes locales cada cinco horas con GPT-6.1 Sol y entre 350 y 3.000 con Luna: depende del tamaño del proyecto y de la duración de cada tarea.

Antes de cada sesión. No gastes el límite los días previos. Si se te agota en clase, cambia a GPT-6 Luna para seguir, o continúa desde la etiqueta de la práctica.

☐ Sé en qué superficie trabajo, qué modelo uso y dónde miro el límite

- **Codex · Modelos** · <https://learn.chatgpt.com/codex/models>
- **Codex · Elegir modelo** · <https://learn.chatgpt.com/codex/model-selection>
- **Codex · Precios y límites** · <https://learn.chatgpt.com/codex/pricing>
- **Help Center · Codex con tu plan de ChatGPT** · <https://help.openai.com/en/articles/11369540>

Bloque 2 · Puesta a punto y primera sesión

Práctica 1 · instalar, abrir el proyecto, AGENTS.md y permisos.

Dejamos Codex instalado en la app de escritorio y en la terminal, abrimos el repositorio de Cafès Roca, generamos AGENTS.md con /init y elegimos el modo de permisos.

1. Instalar

Pieza	Windows (PowerShell)	Mac (Terminal)
App de ChatGPT	Microsoft Store, o winget install --id 9PLM9XGG6VKS -s msstore	chatgpt.com/download
Codex CLI	powershell - ExecutionPolicy Bypass -c "irm https://chatgpt.com/codex/install.ps1 iex"	curl -fsSL https://chatgpt.com/codex/install.sh sh
Git	winget install --id Git.Git -e	xcode-select --install
Node.js 20 o posterior	winget install OpenJS.NodeJS.LTS	nodejs.org (instalador LTS)
GitHub CLI	winget install --id GitHub.cli	brew install gh

Con npm también funciona: `npm install -g @openai/codex`. Para actualizar la CLI, `codex update`.

Comprueba que todo responde

```
codex --version
git --version
node --version
gh --version
```

2. Iniciar sesión y abrir el proyecto

1. Descomprime **codex-avancat-6h-repositori-cafes-roca.zip** en tu carpeta de documentos. Dentro está **cafes - roca**, que ya es un repositorio de Git.
2. Abre la app de ChatGPT, pasa al modo **Codex** e inicia sesión con tu cuenta.
3. Añade la carpeta **cafes - roca** como proyecto (**Add new project** o **Ctrl+O**). Si te pregunta si confías en la carpeta, di que sí: es tuya.
4. En el selector de abajo, elige **GPT-6.1 Sol** y razonamiento **Medium**. Comprueba que trabajas en **Local**.
5. En la terminal, entra en la carpeta y escribe **codex**. La primera vez elige **Sign in with ChatGPT**.

Primer encargo · explorar sin tocar

Explícame este repositorio: qué hay en cada carpeta, cómo se arranca la aplicación de **estoc**, cómo se ejecutan las pruebas y qué incidencias hay abiertas. Ejecuta las pruebas para ver si pasan. No modifiques ningún fichero.

Comprobación. Codex describe **dades/**, **estoc/** y **web/**, encuentra **npm test** y **npm start**, informa de que pasan **5 pruebas** y lista las incidencias **#12**, **#13** y **#14** de **estoc/ISSUES.md**.

3. AGENTS.md con /init

Qué es AGENTS.md. Un fichero de texto con las instrucciones del proyecto que Codex lee **antes de cada tarea**: cómo se ejecutan las pruebas, qué no se toca, convenciones de estilo. Codex combina el global (**~/ .codex/AGENTS.md**) con los de cada carpeta, desde la raíz del repositorio hasta donde trabajas; el más cercano manda. Límite: 32 KiB.

1. Escribe **/init** en el chat. Codex lee el proyecto y propone un **AGENTS.md**.
2. Revisa el diff antes de aceptarlo.
3. Pídele los ajustes de abajo.

Ajustar AGENTS.md

Revisa el **AGENTS.md** y añade estas normas:

- Responde en castellano y escribe los textos para el usuario en catalán, que es el idioma de **Cafès Roca**.
- No modifiques nunca los ficheros de **dades/**: son el origen. Si hay que transformarlos, guarda el resultado en otra carpeta.
- Antes y después de cambiar código de **estoc/**, ejecuta las pruebas.
- No instales paquetes **npm** sin preguntarme.
- Formato de números en catalán: coma decimal y punto de millares.

Incluye una sección «**Dades**» que describa **vendes_cafes_roca_2026.csv**: separador punto y coma, coma decimal, UTF-8 y saltos de línea CRLF.

4. Permisos: qué puede hacer sin preguntarte

Modo en la app	Qué permite	Cuándo
Pedir aprobación (Ask for approval)	Lee y edita dentro del proyecto; pregunta para salir a internet o tocar ficheros externos	Por defecto y en todo el curso
Aprobar por mí (Approve for me)	Lo mismo, pero un agente revisor aprueba o deniega en tu lugar las peticiones	Tareas largas cuando ya confías en el proyecto
Acceso completo (Full access)	Sin sandbox: cualquier fichero del ordenador e internet	Casi nunca. Solo en entornos aislados
Personalizado	Lo que digas en config.toml	Equipos con normas propias

En la terminal es lo mismo con dos opciones: `--sandbox` (read-only, workspace-write, danger-full-access) y `--ask-for-approval` (on-request, never). En una sesión abierta, `/permissions`. La opción antigua `--full-auto` está obsoleta.

5. Local, Worktree y deshacer

- **Local:** Codex trabaja en tu carpeta. Ves los cambios al momento.
- **Worktree:** trabaja en una copia aislada del repositorio (un *git worktree*) para no pisar lo que tienes a medias. La usaremos en la sesión 2.
- Cada cambio aparece como **diff**. Puedes aceptarlo, pedir que lo rehaga o revertirlo. Con Git, nada es irreversible: `git restore` . deshace lo que no has guardado.

Si algo falla. En Windows, si PowerShell no deja ejecutar el instalador, abre PowerShell como usuario normal y usa exactamente el comando con `-ExecutionPolicy Bypass`. Si la app no ve Git, cierra y vuelve a abrir la app después de instalarlo. Si la CLI no te deja iniciar sesión en un ordenador compartido, usa `codex login --device-auth`.

□ Codex instalado, proyecto abierto, AGENTS.md revisado y modo «Pedir aprobación»

- **Codex · Inicio rápido** · <https://learn.chatgpt.com/codex/quickstart>
- **Codex · App** · <https://learn.chatgpt.com/codex/app>
- **Codex · CLI** · <https://learn.chatgpt.com/codex/cli>
- **Codex · AGENTS.md** · <https://learn.chatgpt.com/codex/agent-configuration/agents-md>
- **Codex · Modos de permisos** · <https://learn.chatgpt.com/codex/permission-modes>
- **Codex · Windows** · <https://learn.chatgpt.com/codex/windows/windows-app>

Bloque 3 · Una landing page a partir de un brief

Práctica 2 · plan, construcción, navegador integrado y Markdown.

El brief de Cafès Roca pide una página con los tres locales, la carta destacada y un formulario de càterring. Primero el plan; después la página; después la revisamos en el navegador integrado de la app, como lo haría un cliente.

Por qué empezar con `/plan`. En modo Plan, Codex lee, pregunta y propone, pero no edita. Las preguntas que hace son las decisiones que de otro modo tomaría solo: tipografías, qué pasa al enviar el formulario, si usa librerías. Responderlas cuesta un minuto y evita rehacer media página.

1. Chat nuevo en el proyecto. Escribe `/plan` para activar el modo Plan.
2. Pega el prompt de planificación y responde a sus preguntas.
3. Cuando el plan te convenza, sal del modo Plan y pide que lo implemente.
4. Abre `web/index.html` en el **navegador integrado** de la app y revísala en escritorio y en móvil.

Plan de la landing

Lee `dades/brief_web_cafes_roca.txt`. Quiero la landing que describe dentro de `web/index.html` y una hoja de estilos aparte, sin frameworks ni librerías externas, que funcione bien en el móvil. Antes de escribir nada, hazme las preguntas que necesites y propón un plan con las secciones, los ficheros y cómo comprobaremos que está bien.

Implementar

Adelante con el plan. Textos en catalán, tomados del brief sin inventar datos. El formulario de càterring valida los campos obligatorios y, al enviarlo, abre el correo del usuario dirigido a `hola@cafesroca.cat` con los datos rellenados. Cuando acabes, ábrela en el navegador integrado y revisa que no haya desplazamiento horizontal a 390 px de ancho.

Iterar con capturas y anotaciones

En el navegador integrado, activa el modo de anotación, marca lo que no te guste (un botón que no se ve, un espacio excesivo) y envía la captura con un comentario. Codex recibe la imagen con tus marcas.

Corrección con captura

En la captura que te adjunto, el botón de càtering de la cabecera queda demasiado pegado al texto y las tarjetas de los locales tienen alturas distintas en el móvil. Corrígelo sin cambiar los colores.

Artículos en Markdown convertidos en páginas

En `web/articles/` hay dos artículos en Markdown. Pedimos un script que los convierta en páginas con el mismo estilo; así, quien escriba artículos no tendrá que tocar HTML.

De Markdown a páginas

Crea `web/build.mjs`: un script de Node sin dependencias que convierta cada `.md` de `web/articles/` en un `.html` con la cabecera, el pie y los estilos de la landing (títulos, párrafos, listas, negritas y enlaces). Añade a la landing una sección «Del nostre torrador» con enlaces a los dos artículos. Ejecuta el script y comprueba que las páginas se abren.

Ejecutar el script

```
node web/build.mjs
```

Comprobación. La página tiene cabecera con botón al formulario, los tres locales con dirección y horario exactos del brief, los cinco productos con su precio (flat white 3,00 €, cold brew 3,40 €, croissant 1,90 €, carrot cake 3,80 €, tostada 5,90 €), el formulario y el pie con `hola@cafesroca.cat`. Los colores son `#3B2A1F`, `#F4EBDD` y `#C8553D`. Los dos artículos existen en HTML y se enlazan desde la landing.

1. Revisa el diff completo en la app.
2. Haz el commit desde la app con un mensaje en catalán, por ejemplo: «Afegeix la landing de Cafès Roca i els articles».

Si algo falla. Si inventa datos (un cuarto local, otro horario), dile «usa solo lo que pone el brief» y que lo revise línea a línea. Si la página se ve sin estilos en el navegador integrado, comprueba la ruta de la hoja de estilos: Codex a veces la enlaza desde la raíz del proyecto en lugar de desde `web/`.

☐ Landing revisada en escritorio y móvil, artículos generados y commit hecho

- **Codex · Modos (Local, Worktree, Cloud)** · <https://learn.chatgpt.com/codex/environments/modes>
- **Codex · Navegador integrado** · <https://learn.chatgpt.com/codex/browser>
- **Codex · Comandos de barra** · <https://learn.chatgpt.com/codex/reference/slash-commands>

Bloque 4 · Análisis técnico de datos

Práctica 3 · un script reproducible, gráficos e informe.

En el curso básico ChatGPT calculaba las cifras dentro del chat. Aquí pedimos un script que se queda en el repositorio: cualquiera puede volver a ejecutarlo el mes que viene con datos nuevos y obtener el mismo informe.

Reproducible quiere decir. Que las cifras salen de código que puedes leer y volver a ejecutar, así que sabes exactamente cómo se han calculado. Cada número del informe se puede rastrear hasta la línea que lo calcula, y el script comprueba que las partes suman el total.

El script de análisis

Crea `analisi/analisi.mjs`, un script de Node sin dependencias que:

1. Lea `dades/vendes_cafes_roca_2026.csv` sin modificarlo (punto y coma, coma decimal, CRLF) y avise si alguna fila tiene un formato incorrecto.
 2. Calcule la facturación total, por local, por categoría con su peso en %, por mes y local, las bebidas frías por mes y los cinco productos que más facturan.
 3. Compruebe que las sumas por local, categoría, mes y producto coinciden con el total, y se detenga con un error si no cuadran.
 4. Guarde los resultados en `analisi/resultats.json`.
- Ejecútalo y enséñame las cifras.

Gráficos

Añade al script tres gráficos en SVG, sin librerías, en `analisi/grafics/`: ventas mensuales por local (líneas), ventas por categoría (barras horizontales con importe y %) y bebidas frías por mes (columnas, con mayo y junio destacados). Colores de la marca: Gràcia #3B2A1F, Poblenou #C8553D, Sants #B08B5B. Para cada gráfico, explícame por qué ese tipo responde mejor a la pregunta.

Informe

Genera `analisi/informe.md` en catalán a partir de `resultats.json`: cifras principales, una tabla por local, una por categoría, la evolución de las bebidas frías y el top 5 de productos, con los gráficos insertados. Cada cifra tiene que salir del JSON; no escribas ningún número a mano.

Comprobación. Facturación del semestre: **104.827,00 €** en 2.487 líneas y 35.593 unidades. Poblenou **44.468,90 €** (42,4 %), Gràcia 33.548,00 € y Sants 26.810,10 €. Café 31.565,50 € (30,1 %), desayunos 24.722,30 € (23,6 %). Mejor mes: junio, 18.991,60 €. Las bebidas frías pasan de 1.226,90 € en enero a 5.198,00 € en junio. Producto que más factura: flat white, 10.293,00 €.

Qué gráfico para qué pregunta. **Líneas** para ver la evolución en el tiempo. **Barras horizontales** para comparar categorías con nombres largos. **Columnas** para pocos periodos con un cambio que se quiere señalar. Los gráficos circulares cuestan de leer cuando hay más de tres partes parecidas.

Si algo falla. Si el total sale diez veces más grande o más pequeño, el script ha leído mal la coma decimal: pídele que muestre las tres primeras filas parseadas. Si propone instalar una librería de gráficos, recuérdale la norma de AGENTS.md.

☐ Script, gráficos e informe generados y cifras comprobadas

- **Codex · Prompting** · <https://learn.chatgpt.com/docs/prompting>
- **Codex · Inicio rápido** · <https://learn.chatgpt.com/codex/quickstart>

Bloque 5 · Un dashboard interactivo

Práctica 4 · especificar antes de construir.

Un dashboard tiene muchas decisiones pequeñas: qué indicadores, qué filtros, de dónde salen los datos. Las escribimos en un `SPEC.md` antes de construir, con la ayuda de Codex en modo Plan, y después le pedimos que lo construya contra esa especificación.

Por qué una especificación. El `SPEC.md` se queda en el repositorio. Sirve de contrato con el agente (si se desvía, le señalas el punto), de documentación para quien venga después y de lista de comprobación: los criterios de aceptación dicen exactamente qué cifras tiene que mostrar.

1. Chat nuevo. Activa el modo Plan con `/plan` y pega el prompt.

Especificación en modo Plan

Quiero un dashboard de ventas de Cafès Roca que la dirección y los encargados puedan abrir desde el navegador sin servidor. Hazme las preguntas que necesites sobre usuarios, indicadores, filtros, gráficos y diseño. Con mis respuestas, escribe

dashboard/SPEC.md con objetivo, datos, contenido, diseño y criterios de aceptación con cifras que se puedan comprobar con analisi/resultats.json.

Respuestas de referencia, por si quieres ir rápido: filtros por local y por intervalo de meses; cuatro indicadores (facturación, unidades, precio medio por unidad y peso de las bebidas frías); facturación por mes en columnas apiladas por local; facturación por categoría; tabla de productos ordenable con el top 5 destacado; colores de la marca; un solo HTML sin librerías.

Construir contra la especificación

Implementa dashboard/SPEC.md. Los datos agregados los genera un script dashboard/genera-dades.mjs que lee el CSV y escribe dashboard/dades.js; el dashboard no lee el CSV directamente. Al acabar, abre el dashboard en el navegador integrado, prueba los filtros y comprueba uno por uno los criterios de aceptación. Dime cuáles se cumplen y cuáles no.

Generar los datos del dashboard

```
node dashboard/genera-dades.mjs
```

Comprobación. Sin filtros: **104.827,00 €** y 35.593 unidades, precio medio 2,95 € y bebidas frías 16,8 %. Solo Poblenou: **44.468,90 €**. Solo junio: **18.991,60 €**, con las bebidas frías al 27,4 %. Ningún error en la consola.

1. Contrasta las tres cifras con analisi/resultats.json.
2. En el panel de cambios de la app, revisa el diff, prepara los ficheros (**stage**) y haz el commit: «Afegeix l'especificació i el dashboard de vendes».

Si algo falla. Si los importes no cuadran por céntimos, pídele que sume en céntimos enteros y divida al final. Si el dashboard sale en blanco, dile que mire la consola del navegador integrado y corrija el error.

☐ SPEC.md escrito, dashboard verificado contra los criterios y commit hecho

- **Codex · App: revisar y hacer commit** · <https://learn.chatgpt.com/codex/app>
- **Codex · Comandos de barra** · <https://learn.chatgpt.com/codex/reference/slash-commands>

Bloque 6 · Contexto, uso y buenas prácticas

Cómo trabajar muchas horas sin perder el hilo ni el límite.

La ventana de contexto es todo lo que Codex tiene presente en un hilo: tus mensajes, los ficheros que ha leído, la salida de los comandos. Cuando se llena, responde peor y gasta más. Hay comandos para controlarlo.

Comando	Qué hace	Cuándo
/status	Modelo, uso del contexto y límites restantes	Antes de una tarea larga
/usage	Uso de la cuenta y, si tienes, reinicios de límite guardados	Cuando vas justo
/compact	Resume la conversación para liberar contexto	Hilos largos que quieres continuar
/new	Chat nuevo en el mismo proyecto	Cada tarea nueva
codex resume · /resume	Retoma un chat guardado (en la CLI)	Al día siguiente
/fork	Copia el chat en otro hilo o worktree para probar otra vía	Comparar dos soluciones
/model · /reasoning	Cambia el modelo o el nivel de razonamiento	Bajar a Luna para tareas mecánicas

Comando	Qué hace	Cuándo
/fast	Respuestas más rápidas a cambio de consumir 2,5 veces más	Raramente

Hábitos que funcionan

- **Una tarea por hilo.** Al acabar, /new.
- **Peticiones verificables:** di cómo sabrás que está bien («las pruebas pasan», «el total es 104.827,00 €»).
- **Para y corrige** en cuanto veas que va por mal camino: es más barato que dejar que acabe.
- **Lo que repites, a AGENTS.md;** lo que es un procedimiento, a una skill (sesión 2).
- **Commit después de cada paso bueno:** es tu punto de retorno.

Repaso con /status

Antes de seguir: ¿cuánto contexto llevamos en este hilo y cuánto límite me queda? Si conviene, resume lo hecho en cinco líneas para que pueda abrir un chat nuevo y continuar.

☐ Sé consultar el contexto y el límite, y cuándo abrir un chat nuevo

- **Codex · Comandos de barra** · <https://learn.chatgpt.com/codex/reference/slash-commands>
- **Codex · Precios y límites** · <https://learn.chatgpt.com/codex/pricing>
- **Panel de uso de Codex** · <https://chatgpt.com/codex/settings/usage>

Repaso de la sesión 1

Lo que tienes que tener hecho antes de la sesión 2.

Práctica	Resultado	Etiqueta de rescate
P1 · Puesta a punto	AGENTS.md revisado	p1-posada-a-punt
P2 · Landing	web/index.html y artículos	p2-landing
P3 · Análisis	analisi/ con script, gráficos e informe	p3-analisi
P4 · Dashboard	dashboard/ con SPEC.md	p4-dashboard

Para la sesión 2. Crea una cuenta gratuita en GitHub si no la tienes, y ejecuta `gh auth login` en la terminal para conectar GitHub CLI con tu cuenta. Trae el repositorio con todos tus commits: lo subiremos a GitHub al empezar.

Sesión 2 · Repositorios, calidad y automatización

Trabajamos como un equipo de desarrollo: GitHub, pruebas, pull requests y revisión; adaptamos Codex al proyecto, lo conectamos con otras herramientas y automatizamos tareas.

Cómo se organiza la sesión 2

Del repositorio local a GitHub.

Partimos del repositorio con tus commits de la sesión 1 (o de la etiqueta **p4-dashboard**). Primero lo subimos a GitHub; a partir de ahí, cada cambio va por rama, pull request y revisión.

Concepto	En palabras llanas
Repositorio	La carpeta del proyecto con todo su historial de cambios
Commit	Una foto guardada de los cambios, con un mensaje que explica qué y por qué
Rama (branch)	Una línea de trabajo paralela para no tocar la versión buena mientras pruebas
Pull request (PR)	La propuesta de juntar una rama con la principal, con su diff, comentarios y revisión
Worktree	Una segunda carpeta del mismo repositorio, con otra rama, para trabajar dos cosas a la vez
Merge	Juntar una rama con otra. Si dos ramas cambian las mismas líneas, hay un conflicto que alguien decide

Bloque 7 · Git y GitHub con Codex

Práctica 5 · subir el repositorio, entender la app de estoc y corregir un error empezando por una prueba.

Subimos el repositorio a GitHub, exploramos la aplicación de estoc antes de tocarla y resolvemos la incidencia #12 con el método de los equipos profesionales: primero una prueba que falla, después la corrección.

1. Conectar con GitHub y subir el repositorio

Iniciar sesión en GitHub CLI (una sola vez)

```
gh auth login
```

Crear el repositorio en tu cuenta y subirlo

```
gh repo create cafes-roca --public --source . --remote origin --push
```

Lo creamos público porque en la práctica 9 publicaremos la web con GitHub Pages, que en cuentas gratuitas necesita un repositorio público. Los datos son ficticios.

2. Entender antes de tocar

Explorar la aplicación de estoc

```
Explícame cómo funciona la aplicación de estoc/: de dónde salen los datos, cómo se calcula el estoc a partir de los movimientos, cuándo un producto entra en alertas y
```

cómo se calcula la comanda sugerida. Hazme un esquema del flujo desde que llega una petición a /api/comanda hasta que se pinta la tabla. No cambies nada.

Arrancar la aplicación

```
cd estoc
npm start
```

Ábrela en **http://localhost:3000** desde el navegador integrado. Hay 4 productos en alerta y una comanda de 770,70 €.

3. Incidencia #12: primero la prueba

La incidencia. Laia, de Gràcia, contó 6 litros de leche de avena. El mínimo es 6: debería salir en alertas y en la comanda, y no sale. Un cliente se quejó (opinión número 7).

Reproducir el error con una prueba

Lee la incidencia #12 de estoc/ISSUES.md. Crea una rama codex/incidencia-12. Escribe en estoc/test/estoc.test.js una prueba que reproduzca exactamente el caso (leche de avena en Gràcia, 6 l, mínimo 6) y que compruebe que sale en alertas y en la comanda. Ejecuta las pruebas y enséñame que la nueva falla. Todavía no corrijas nada.

Corregir

Ahora corrige el error con el cambio mínimo. No toques los datos ni las otras pruebas. Ejecuta todas las pruebas, explícame la causa en una frase y haz un commit por cada paso: uno con la prueba y otro con la corrección, mensajes en catalán con «(#12)».

Comprobación. La prueba nueva falla antes de la corrección y pasa después: **6 pruebas en verde**. La causa: estaSotaMinim comparaba con < en lugar de <=. En la app, ahora hay **8 alertas** y la comanda sube a **872,20 €**.

4. Push y pull request

Desde el panel de cambios de la app: **push** de la rama y **Create pull request**. O desde la terminal:

Con GitHub CLI

```
git push -u origin codex/incidencia-12
gh pr create --fill
gh pr view --web
```

Revisa el diff en GitHub y fusiona la PR (**Merge pull request**). Después, en local: `git switch main` y `git pull`.

Si algo falla. Si `gh repo create` dice que el nombre existe, usa otro (cafes-roca-2). Si el push pide usuario y contraseña, vuelve a ejecutar `gh auth login` y elige HTTPS. Si Codex corrige el error cambiando el mínimo en los datos, recházalo: es justo lo que prohíbe AGENTS.md.

☐ Repositorio en GitHub, prueba que reproduce #12, corrección y PR fusionada

- **Codex · App (Git, commit y PR)** · <https://learn.chatgpt.com/codex/app>
- **Codex · GitHub** · <https://learn.chatgpt.com/codex/third-party/github>
- **GitHub CLI · Manual** · <https://cli.github.com/manual/>

Bloque 8 · Trabajar en paralelo y revisar

Práctica 6 · dos mejoras a la vez en worktrees, conflicto y revisión.

Las incidencias #13 (filtro por local) y #14 (exportar la comanda en CSV) son independientes. Las lanzamos a la vez, cada una en su **worktree**, revisamos el código y las juntamos.

Cómo funcionan los worktrees en Codex. Al elegir **Worktree** en lugar de Local, la app crea una copia del repositorio en `$CODEX_HOME/worktrees` con su propia rama. Los dos chats trabajan sin pisarse, y tu carpeta principal queda intacta. Cuando el trabajo está bien, lo pasas a una rama (**Create branch here**, con prefijo `codex/`) o lo traes a Local con **Handoff**.

1. Chat nuevo en modo **Worktree**: prompt de la #14.
2. Sin esperar, otro chat nuevo en modo **Worktree**: prompt de la #13.
3. Mientras trabajan, mira la lista de chats: cada uno muestra su estado.

Worktree A · #14 exportar CSV

Resuelve la incidencia #14 de `estoc/ISSUES.md`: función `comandaCsv` en `src/estoc.js` con su prueba, endpoint `/api/comanda.csv` con BOM para que Excel lea bien los acentos, y botón «Descarga la comanda (CSV)». Ejecuta las pruebas.

Worktree B · #13 filtro por local

Resuelve la incidencia #13 de `estoc/ISSUES.md`: selector de local que filtre el resumen, las alertas, la tabla y la comanda, y que recuerde la última opción en `localStorage`. Solo cambios en `public/`. Comprueba en el navegador integrado que funciona con cada local.

Revisar antes de juntar

En cada chat, escribe `/review`. Codex revisa los cambios sin aplicarlos, como lo haría otra persona del equipo, y marca problemas por prioridad. Lee el diff tú también: la revisión del agente no sustituye la tuya.

Revisión desde la terminal

```
codex review --base main
```

Juntar y resolver el conflicto

Crea una rama en cada **worktree** (**Create branch here**), súbelas y abre dos PR. Al fusionar la segunda, es probable que GitHub avise de un conflicto en `style.css`: las dos ramas han añadido reglas al final del mismo fichero.

Resolver el conflicto

Estoy fusionando `codex/filtre-local` en `main` y hay un conflicto en `estoc/public/style.css`. Enséñame las dos versiones, explícame qué quería cada rama y resuélvelo conservando las dos. Ejecuta las pruebas y abre la aplicación para comprobar que el botón y el selector se ven bien.

Revisión automática en GitHub

- Conecta Codex con GitHub desde la configuración de Codex en `chatgpt.com` y activa la **revisión de código** para tu repositorio.
- En cualquier PR, escribe un comentario `@codex review`, o activa la revisión automática para que revise todas las PR.
- Codex solo señala problemas importantes (P0 y P1). Las reglas de revisión del equipo van en `AGENTS.md`, bajo `## Code Review Rules` (bloque 9).

Demostración: varios intentos en la nube

Con un entorno cloud configurado, Codex puede probar la misma tarea varias veces en paralelo y dejarte elegir la mejor solución. Desde la terminal: `codex cloud exec --env ID --attempts 3 "...",` y después `codex cloud diff y codex apply`.

Comprobación. Las dos mejoras están en main, **7 pruebas en verde**, el CSV se descarga con la cabecera `local;codi;producte;lots;quantitat;unitat;import_eur` y el filtro recuerda el local al recargar.

Si algo falla. Si un worktree no ve los cambios de la sesión 1, es que se creó desde un commit antiguo: crea uno nuevo desde main actualizado. Si no puedes conectar GitHub con Codex (cuentas de empresa con restricciones), haz la revisión con `/review` en local.

□ #13 y #14 hechas en paralelo, revisadas, conflicto resuelto y fusionadas

- **Codex · Worktrees** · <https://learn.chatgpt.com/codex/environments/git-worktrees>
- **Codex · Revisión de código** · <https://learn.chatgpt.com/codex/code-review>
- **Codex · Codex cloud** · <https://learn.chatgpt.com/codex/cloud>
- **Codex · Entornos cloud** · <https://learn.chatgpt.com/codex/environments/cloud-environments>

Bloque 9 · Personalizar el agente

Práctica 7 · AGENTS.md de equipo, skills, config.toml, perfiles y hooks.

Cuatro piezas para que Codex trabaje como quiere tu equipo sin repetírselo cada vez. Cada una resuelve un problema distinto.

Pieza	Problema que resuelve	Dónde vive
AGENTS.md	Normas que valen para todas las tareas	Raíz del repo y carpetas; global en <code>~/.codex/</code>
Skill	Un procedimiento que se repite (con scripts si hace falta)	<code>.agents/skills/<nombre>/SKILL.md</code> ; personales en <code>~/.agents/skills</code>
config.toml y perfiles	Modelo, razonamiento, permisos y servidores MCP	<code>~/.codex/config.toml</code> ; perfiles en <code>~/.codex/<nombre>.config.toml</code> ; del proyecto en <code>.codex/config.toml</code>
Hook	Algo que tiene que pasar siempre, sin depender de que el agente se acuerde	<code>.codex/hooks.json</code> o <code>~/.codex/hooks.json</code>

1. AGENTS.md para todo el equipo y por carpetas

AGENTS.md de equipo

Amplía el AGENTS.md de la raíz con dos secciones:

- «Flux de treball»: una rama por tarea con prefijo `codex/`, commits en catalán en imperativo con el número de incidencia, y antes de una PR `pruebas en verde` y `/review` sin problemas graves.
- «Code Review Rules»: grave `modificar dades/`, grave cualquier clave o token escrito en el código, toda función nueva de `estoc/src/` con su prueba, importes en euros redondeados al final.

Y crea `estoc/AGENTS.md` con las normas propias de la aplicación: Node 20 con módulos ES, sin dependencias, lógica en funciones puras en `src/estoc.js`, pruebas con `node:test` y, para un error, primero la prueba que lo reproduce.

Comprobar qué instrucciones lee Codex

```
codex --ask-for-approval never "Resume las instrucciones que tienes para la carpeta estoc/"
```

2. Una skill para el informe semanal de estoc

Cómo funciona una skill. Una carpeta con un SKILL.md: arriba, un nombre y una descripción; debajo, los pasos. Codex solo carga la descripción de cada skill; si tu petición encaja, lee el resto. Por eso la descripción tiene que decir **cuándo** usarla. Puede llevar scripts que hacen la parte mecánica. Se invoca con \$nombre, desde /skills o, en la app, con @.

Crear la skill con \$skill-creator

\$skill-creator Crea una skill de proyecto llamada informe-estoc en .agents/skills/. Tiene que generar el informe semanal de estoc de Cafès Roca en informes/estoc_AAAA-MM-DD.md: primero ejecuta las pruebas (si fallan, se detiene), después un script scripts/informe.mjs que calcula alertas y comanda con las funciones de estoc/src/estoc.js, y al final añade tres líneas de observaciones. No modifica estoc/data/ ni envía nada. La descripción tiene que activarla cuando pidan el informe de estoc o qué hay que pedir al proveedor.

Usar la skill

```
$informe-estoc
```

Comprobación. Se crea informes/estoc_2026-...md con **8 alertas** y comanda de **872,20 €**. Añade informes/ al .gitignore.

3. config.toml y perfiles

~/codex/config.toml

```
model = "gpt-6.1-sol"
model_reasoning_effort = "medium"
approval_policy = "on-request"
sandbox_mode = "workspace-write"
web_search = "cached"

[sandbox_workspace_write]
network_access = false
```

Un **perfil** es otro fichero de configuración que se activa cuando lo pides. Desde la versión 0.134 los perfiles van en ficheros aparte. En el repositorio tienes config/revisio.config.toml: cópialo a ~/.codex/.

~/codex/revisio.config.toml

```
model = "gpt-6.1-sol"
model_reasoning_effort = "high"
sandbox_mode = "read-only"
approval_policy = "on-request"
```

Usar el perfil

```
codex --profile revisio "Revisa estoc/src/estoc.js y dime qué casos límite no cubren las pruebas"
```

4. Un hook que ejecuta las pruebas después de cada cambio

Cómo funciona un hook. Un comando que Codex ejecuta en un momento concreto del ciclo: antes de usar una herramienta (PreToolUse), después (PostToolUse), al enviar un prompt, al terminar el turno

(Stop)... Si el comando sale con código 2, lo que escribe en la salida de error vuelve a Codex como respuesta, y tiene que corregirlo. Por seguridad, un hook nuevo o modificado no se ejecuta hasta que lo revisas y confías en él con `/hooks`.

`.codex/hooks.json`

```
{
  "hooks": {
    "PostToolUse": [
      {
        "matcher": "apply_patch",
        "hooks": [
          {
            "type": "command",
            "command": "node .codex/hooks/proves-estoc.mjs",
            "timeout": 120,
            "statusMessage": "Executant les proves de l'estoc"
          }
        ]
      }
    ]
  }
}
```

Crear el hook

Crea un hook de proyecto en `.codex/hooks.json` que, después de cada edición de ficheros (PostToolUse con matcher `apply_patch`), ejecute `node .codex/hooks/proves-estoc.mjs`. El script lee de la entrada estándar el evento; si el cambio afecta a `estoc/`, ejecuta `node --test` dentro de `estoc/` y, si alguna prueba falla, sale con código 2 y escribe el resultado en `stderr`. Que funcione en Windows y en Mac: nada de `bash`.

1. En la terminal, abre `codex` en el proyecto y escribe `/hooks`: revisa el hook y márcalo como de confianza.
2. Pide a Codex un cambio que rompa una prueba a propósito («cambia el redondeo de la comanda a `Math.round`») y mira cómo el hook lo detiene y Codex lo deshace.

Más piezas, para cuando las necesites. **Rules** (experimental): reglas en `~/ .codex/rules/` que permiten, piden confirmación o prohíben comandos concretos, como `git push --force`. **Subagentes**: agentes con instrucciones propias en `.codex/agents/` que Codex lanza en paralelo; se gestionan con `/agents`.

☐ AGENTS.md por niveles, skill que funciona, perfil de revisión y hook de confianza

- **Codex · AGENTS.md** · <https://learn.chatgpt.com/codex/agent-configuration/agents-md>
- **Codex · Skills** · <https://learn.chatgpt.com/codex/build-skills>
- **Codex · Configuración básica** · <https://learn.chatgpt.com/codex/config-file/config-basic>
- **Codex · Referencia de config.toml** · <https://learn.chatgpt.com/codex/config-file/config-reference>
- **Codex · Hooks** · <https://learn.chatgpt.com/codex/hooks>
- **Codex · Rules** · <https://learn.chatgpt.com/codex/agent-configuration/rules>
- **Codex · Subagentes** · <https://learn.chatgpt.com/codex/agent-configuration/subagents>

Bloque 10 · Conectar herramientas: MCP y plugins

Práctica 8 · un navegador real para probar la web.

MCP (Model Context Protocol) es el estándar con el que un agente usa herramientas externas: un navegador, la documentación de una librería, Figma, Sentry, GitHub. Conectamos Playwright para que Codex pruebe la landing en un navegador real.

Codex avanzado para perfiles técnicos · Cuaderno de prácticas · barcelonactiva.cat/formacio

Cómo funciona MCP. Un **servidor MCP** es un pequeño programa que expone herramientas con nombre y descripción («abrir página», «hacer clic», «rellenar campo»). Codex lo arranca al empezar la sesión, ve sus herramientas y decide cuándo usarlas, como hace con sus herramientas propias. Puede ser local (un comando, normalmente con npx) o remoto (una URL con inicio de sesión).

Añadir Playwright

```
codex mcp add playwright -- npx @playwright/mcp@latest
```

Comprobar

```
codex mcp list
```

En la app: **Configuración > Servidores MCP**. En una sesión, /mcp lista los servidores y sus herramientas. La configuración queda en config.toml:

config.toml

```
[mcp_servers.playwright]
command = "npx"
args = ["@playwright/mcp@latest"]
startup_timeout_sec = 30
```

Probar la landing en un navegador real

Con el servidor MCP de Playwright, abre web/index.html y prueba: (1) a 390 × 844 no hay desplazamiento horizontal; (2) enviar el formulario vacío muestra un aviso; (3) con 3 personas (el mínimo es 5) el mensaje dice qué campo está mal; (4) los enlaces a los dos artículos funcionan; (5) no hay errores en la consola. Guarda capturas en web/proves/ y un informe PROVES_NAVEGADOR.md. Si algo falla, corrígelo y vuelve a probar.

Comprobación. La prueba 3 suele fallar: el formulario da un mensaje genérico. Después de la corrección dice «Revisa el nombre de personas (entre 5 i 300)». Hay tres capturas y el informe en web/proves/.

Otros servidores útiles

Servidor	Para qué	Cómo
Context7	Documentación actualizada de librerías y frameworks	codex mcp add context7 -- npx -y @upstash/context7-mcp
OpenAI Docs	La documentación oficial de OpenAI, para preguntar por Codex con fuentes	Desde Servidores MCP de la app
Figma	Leer diseños y convertirlos en código	Servidor remoto: https://mcp.figma.com/mcp
Chrome DevTools, GitHub, Sentry	Depurar en el navegador, gestionar incidencias, leer errores de producción	Desde Servidores MCP de la app

Plugins

Un plugin empaqueta skills, servidores MCP, hooks y conexiones con aplicaciones para instalarlos de una vez. En la terminal, /plugins abre el catálogo (el oficial es openai-curated); en la app, desde el menú de plugins. Los equipos pueden publicar su propio catálogo en un repositorio con codex plugin marketplace add.

Seguridad. Un servidor MCP es código que se ejecuta en tu ordenador con tus permisos. Instala solo los de origen conocido, limita sus herramientas con enabled_tools si no las necesitas todas, y recuerda que lo que lee de fuera (una web, una incidencia) puede contener instrucciones maliciosas.

❑ Playwright conectado, landing probada y corrección hecha

- **Codex · MCP** · <https://learn.chatgpt.com/codex/extend/mcp>
- **Codex · Plugins** · <https://learn.chatgpt.com/codex/plugins>
- **Playwright MCP** · <https://github.com/microsoft/playwright-mcp>
- **Model Context Protocol** · <https://modelcontextprotocol.io>

Bloque 11 · Automatizar y publicar

Práctica 9 · codex exec, tareas programadas, GitHub Action y GitHub Pages.

Codex también trabaja sin que estés delante: dentro de un script, a una hora fija o cada vez que alguien abre una pull request. Y la landing se publica sola cada vez que cambia.

1. codex exec: Codex dentro de un script

Cómo funciona codex exec. Ejecuta una tarea sin interfaz y termina. Por defecto en **solo lectura**. El progreso sale por la salida de error y la respuesta final por la salida estándar, así que se puede encadenar con otros comandos. Con `--output-schema` le das un esquema JSON y la respuesta lo cumple; con `--json` recibes cada evento en una línea, para procesarlo con otro programa.

Comanda semanal en JSON (Mac y Linux)

```
bash automatitzacio/comanda-setmanal.sh
```

Comanda semanal en JSON (Windows)

```
powershell -ExecutionPolicy Bypass -File automatitzacio\comanda-setmanal.ps1
```

Lo que ejecuta el script

```
codex exec \  
  --sandbox read-only \  
  --output-schema automatitzacio/comanda.schema.json \  
  -o informes/comanda_AAAA-MM-DD.json \  
  "Calcula la comanda setmanal de Cafès Roca amb les funcions de estoc/src/estoc.js  
  ..."
```

Crear el esquema y los scripts

Crea en `automatitzacio/`: un esquema JSON `comanda.schema.json` (fecha, alertas con local, producto, cantidad y mínimo; líneas de comanda con local, código, lotes, cantidad e importe; total en euros y observaciones; todos los campos obligatorios y sin propiedades adicionales), y dos scripts, `comanda-setmanal.sh` y `comanda-setmanal.ps1`, que llamen a `codex exec` en solo lectura con ese esquema y guarden el resultado en `informes/comanda_FECHA.json`. Pruébalo.

Comprobación. El JSON valida contra el esquema y dice **8 alertas** y **total_eur: 872.2**.

2. Tareas programadas en la app

1. En la app, modo Codex: **Programadas > Nueva tarea**.
2. Proyecto `cafes-roca`, cada lunes a las 8:00, en el proyecto local.
3. Petición: «Fes servir la skill \$informe-estoc per generar l'informe d'estoc d'aquesta setmana. No modifiquis cap altre fitxer.»
4. Los resultados aparecen en **Programadas**. La app y el ordenador tienen que estar encendidos a esa hora.

En los planes de pago, una tarea también puede dispararse por un evento de Gmail, Slack o GitHub, por ejemplo cuando llega un correo del proveedor.

3. La GitHub Action de Codex (demostración)

openai/codex-action@v1 ejecuta Codex en GitHub Actions. En el repositorio, `.github/workflows/codex-revisio.yml` revisa cada PR con las instrucciones de `.github/codex/revisio.md` y deja la revisión como comentario. Funciona con una **clave de API** de OpenAI guardada como secreto `OPENAI_API_KEY`, que se factura aparte de la suscripción: por eso la hace el formador.

4. Publicar la landing en GitHub Pages

Workflow de GitHub Pages

Crea `.github/workflows/pages.yml` para publicar la carpeta `web/` en GitHub Pages en cada push a `main` que cambie `web/`: checkout, Node 20, `node web/build.mjs` para generar los artículos, y despliegue con las acciones oficiales `configure-pages`, `upload-pages-artifact` y `deploy-pages`. Explícame qué hace cada paso y qué permisos necesita.

1. En GitHub: **Settings** > **Pages** > **Source: GitHub Actions**.
2. Commit y push del workflow. En la pestaña **Actions** verás la ejecución.
3. La web queda en `https://TU-USUARIO.github.io/cafes-roca/`. Ábrela en el móvil.

Si algo falla. Si el despliegue da error de permisos, revisa que el workflow tenga `pages: write` e `id-token: write` y que Pages esté en «GitHub Actions». Si `codex exec` se queja de que no es un repositorio, ejecútalo desde la raíz del proyecto.

☐ Comanda en JSON con `codex exec`, tarea programada creada y web publicada

- **Codex · Modo no interactivo (`codex exec`)** · <https://learn.chatgpt.com/codex/non-interactive-mode>
- **Codex · Tareas programadas** · <https://learn.chatgpt.com/codex/automations>
- **Codex · GitHub Action** · <https://learn.chatgpt.com/codex/github-action>
- **GitHub Pages con Actions** · <https://docs.github.com/en/pages/getting-started-with-github-pages/configuring-a-publishing-source-for-your-github-pages-site>

Bloque 12 · Criterio

Seguridad, datos, costes y cuándo hace falta una revisión experta.

Seguridad

Riesgo	Qué hacer
El agente ejecuta algo destructivo	<code>Sandbox workspace-write</code> y «Pedir aprobación». Acceso completo solo en máquinas aisladas. Commit antes de tareas grandes
Salida a internet	Desactivada por defecto en el sandbox. Actívala solo cuando la tarea lo necesite, y en la nube limita los dominios
Claves y contraseñas	Nunca en el código ni en el chat. En un fichero <code>.env</code> excluido con <code>.gitignore</code> , o como secretos de GitHub y de los entornos cloud
Inyección de prompts	Una web, una incidencia o un PDF pueden llevar instrucciones escondidas. Revisa lo que hace el agente después de leer contenido externo
Servidores MCP y plugins	Código de terceros con tus permisos. Solo de origen conocido

Datos y privacidad

- En Plus y Pro, decides en **Configuración > Controles de datos** si tus conversaciones sirven para mejorar los modelos.
- Business y Enterprise no entrenan con tus datos por defecto, y el administrador controla qué funciones están disponibles.
- No subas al curso repositorios ni datos reales de la empresa.

Costes

- Plus basta para trabajar a diario con GPT-6.1 Sol y razonamiento Medium.
- Astra, Extra High, Ultra, /fast y las tareas en la nube gastan el límite mucho más rápido.
- La GitHub Action y codex exec con clave de API se pagan por uso, aparte de la suscripción.

Cuándo basta el vibe coding y cuándo no

Basta con el agente y tu revisión	Hace falta revisión experta, pruebas y mantenimiento
Prototipos, landings, herramientas internas sin datos sensibles	Pagos, datos personales, autenticación, salud
Scripts de análisis que se comprueban contra cifras conocidas	Sistemas con muchos usuarios o que no pueden caer
Automatizaciones que solo leen	Automatizaciones que escriben, envían o borran
Código que entiendes cuando lo lees	Código que nadie del equipo sabría mantener

Lista de comprobación antes de dar algo por bueno. Pruebas en verde · cifras contrastadas con una fuente independiente · diff leído entero · sin claves en el código · /review sin problemas graves · alguien del equipo sabría explicarlo.

Reto final: tu proyecto

Arrancar un proyecto propio

Quiero empezar un proyecto nuevo con Codex: [describe tu idea en dos líneas]. Antes de escribir código: hazme las preguntas necesarias, propón el stack más sencillo que funcione, escribe un SPEC.md con criterios de aceptación comprobables y un AGENTS.md con las normas del proyecto, y dime cómo lo probaremos. Trabajaremos por pasos pequeños, con un commit por paso.

☐ Tengo mi lista de comprobación y un proyecto propio para empezar

- [Codex · Sandbox](https://learn.chatgpt.com/codex/sandboxing) · <https://learn.chatgpt.com/codex/sandboxing>
- [Codex · Seguridad y aprobaciones](https://learn.chatgpt.com/codex/permission-modes) · <https://learn.chatgpt.com/codex/permission-modes>
- [Codex · Autenticación](https://learn.chatgpt.com/codex/auth) · <https://learn.chatgpt.com/codex/auth>
- [Help Center · Controles de datos](https://help.openai.com/en/articles/7730893-data-controls-faq) · <https://help.openai.com/en/articles/7730893-data-controls-faq>

Repaso de la sesión 2

Estado final del repositorio.

Práctica	Resultado	Etiqueta de rescate
P5 · Git y GitHub	Repositorio en GitHub, #12 corregida con prueba	p5-incidencia-12
P6 · Paralelo y revisión	#13 y #14 fusionadas, conflicto resuelto	p6-paral-lel
P7 · Personalizar	AGENTS.md por niveles, skill, perfil	p7-agent

Práctica	Resultado	Etiqueta de rescate
	y hook	
P8 · MCP	Pruebas de la landing con Playwright	p8-mcp
P9 · Automatizar	codex exec, tarea programada, Action y Pages	p9-automatitzar

Referencia

Cómo funciona Codex: referencia rápida

El ciclo de una tarea

- 1. Codex reúne el contexto: tu petición, AGENTS.md (global y del proyecto), las skills disponibles y la configuración.
- 2. El modelo decide la siguiente acción: leer, buscar, ejecutar un comando, aplicar un parche, usar una herramienta MCP o preguntarte.
- 3. Si la acción sale del sandbox (escribir fuera del proyecto, internet), pide aprobación según el modo de permisos.
- 4. Lee el resultado (salida del comando, error, prueba) y vuelve al paso 2.
- 5. Cuando cree que ha terminado, te lo resume y deja los cambios como diff para que los revises.

Ficheros que controlan a Codex

Fichero	Para qué	Ámbito
AGENTS.md	Instrucciones en lenguaje natural	Global (~/.codex/), repositorio y carpetas
config.toml	Modelo, razonamiento, permisos, MCP, hooks, funciones	~/.codex/ y .codex/ del proyecto de confianza
<nombre>.config.toml	Perfil que se activa con --profile	~/.codex/
SKILL.md	Procedimientos reutilizables	.agents/skills/ y ~/.agents/skills/
hooks.json	Comandos en eventos del ciclo	.codex/ y ~/.codex/
*.rules	Permitir, preguntar o prohibir comandos (experimental)	~/.codex/rules/ y .codex/rules/

Sandbox y aprobaciones

Sandbox	Lee	Escribe	Internet
read-only	Sí	No	No
workspace-write	Sí	En el proyecto (y /tmp)	No, salvo que lo actives
danger-full-access	Sí	Todo	Sí

La política de aprobación decide si te pregunta al salir del sandbox: on-request (pregunta) o never (no pregunta: el sandbox es el único límite). En macOS el sandbox usa Seatbelt; en Linux y WSL2, bubblewrap; en Windows, un sandbox nativo.

Comandos de barra más usados

Comando	Qué hace
/init	Genera un AGENTS.md para el proyecto
/plan	Activa o desactiva el modo Plan: pregunta y propone sin editar
/review	Revisa los cambios sin commit o respecto a una rama
/model · /reasoning	Modelo y nivel de razonamiento
/permissions	Modo de permisos (CLI)

Comando	Qué hace
/status · /usage	Contexto, límites y uso de la cuenta
/compact · /new · /fork	Gestionar el hilo
/worktree · /local · /cloud	Dónde se ejecuta el chat (app)
/mcp · /plugins · /skills · /hooks	Extensiones
/goal	Objetivo persistente para una tarea larga
/side	Conversación lateral temporal sin ensuciar el hilo
/diff	Diff de Git, también de ficheros nuevos (CLI)

Codex CLI: chuleta

Comandos de terminal

```

codex                                # sesión interactiva en la carpeta actual
codex "explica este repo"           # empieza con un prompt
codex -m gpt-6-luna                  # elegir modelo
codex --profile revisio              # usar un perfil
codex -s read-only -a on-request     # sandbox y aprobaciones
codex resume --last                  # retomar el último chat
codex exec "...                     # sin interacción (scripts)
codex exec --json "...              # eventos en JSONL
codex review --base main             # revisar cambios respecto a main
codex mcp add NOMBRE -- COMANDO
codex mcp list
codex cloud exec --env ID --attempts 3 "...
codex apply ID_TAREA                 # traer a local el diff de una tarea cloud
codex login status
codex update

```

- [Codex · Comandos de la CLI](https://learn.chatgpt.com/codex/reference/commands) · <https://learn.chatgpt.com/codex/reference/commands>
- [Codex · Comandos de barra](https://learn.chatgpt.com/codex/reference/slash-commands) · <https://learn.chatgpt.com/codex/reference/slash-commands>
- [Codex · Configuración avanzada](https://learn.chatgpt.com/codex/config-file/config-advanced) · <https://learn.chatgpt.com/codex/config-file/config-advanced>
- [Codex · Ejemplo de config.toml](https://learn.chatgpt.com/codex/config-file/config-sample) · <https://learn.chatgpt.com/codex/config-file/config-sample>
- [Codex · Sandbox](https://learn.chatgpt.com/codex/sandboxing) · <https://learn.chatgpt.com/codex/sandboxing>

Casos de uso

Día a día

Entender un proyecto heredado. Te pasan un repositorio y nadie lo documentó.

Explícame este repositorio como si fuera mi primer día: para qué sirve, cómo se arranca, dónde está la lógica principal y qué partes parecen frágiles. Hazme un mapa de carpetas. No cambies nada.

Limpiar y unir ficheros. Exportaciones de distintos sistemas que hay que juntar cada mes.

En la carpeta exports/ hay tres CSV de ventas con columnas distintas. Escribe un script que los unifique en un solo CSV con fecha, cliente, producto e importe, que avise de las filas que no puede interpretar y que no toque los originales.

Renombrar y ordenar en lote. Cientos de fotos o facturas con nombres sin sentido.

Propón un plan para renombrar los PDF de facturas/ con el formato AAAA-MM-DD_proveedor_importe.pdf leyendo los datos de cada factura. Enséñame la lista de cambios antes de aplicarlos y guarda un registro para poder deshacerlo.

Comandos que no recuerdas. Git, PowerShell, ffmpeg...

Necesito reducir todos los vídeos MP4 de esta carpeta a 720p y menos de 20 MB para enviarlos por correo. Dame el comando, explícame cada parte y pruébalo con un solo fichero primero.

Técnico

Pruebas para código sin pruebas. Antes de tocar algo importante.

Escribe pruebas para src/facturacion.js que cubran el comportamiento actual, incluidos los casos límite (importes 0, negativos, redondeos). No cambies el código: si encuentras un error, anótalo en un comentario de la prueba.

Migración de versión. Actualizar una dependencia o un framework. En modo Plan.

Quiero pasar este proyecto de Node 18 a Node 22. Revisa qué dependencias y APIs cambian, propón el plan por pasos pequeños con las pruebas que validan cada uno y empieza solo cuando lo apruebe.

Depurar con el error delante. Un fallo en producción con su traza.

Te pego la traza del error y el log. Encuentra la causa, escribe una prueba que lo reproduzca y propón la corrección mínima. Explícame por qué no lo detectaban las pruebas actuales.

Revisión antes de la PR. Que otra mirada pase por el código.

Revisa mis cambios respecto a main con atención especial a seguridad, gestión de errores y rendimiento. Prioriza los problemas y dime cuáles bloquearían la PR.

Proyectos personales

Tu web o portafolio. Publicado gratis en GitHub Pages.

Crea mi web personal de una sola página con mi presentación, tres proyectos con imagen y enlace, y un formulario de contacto que abra el correo. HTML y CSS sin librerías, adaptable al móvil. Después prepara el workflow para publicarla en GitHub Pages.

Control de gastos. Con tus extractos del banco en CSV.

Escribe un script que lea los CSV de extractos de la carpeta banco/, clasifique cada movimiento en categorías que yo pueda editar en un fichero JSON y genere un informe mensual con un gráfico por categoría. Los datos no salen de mi ordenador.

Una app pequeña para casa. Lista de la compra, turnos, recetas.

Quiero una aplicación web para planificar los menús de la semana y generar la lista de la compra. Escribe primero un SPEC.md con lo mínimo que tiene que hacer y pregúntame lo que necesites. Guarda los datos en el navegador.

Aprender un lenguaje nuevo. Con un proyecto real y explicaciones.

Quiero aprender Python haciendo un proyecto: un script que descargue el tiempo de mi ciudad cada mañana y me lo resuma. Ve paso a paso, explícame cada concepto nuevo cuando aparezca y déjame escribir yo algunas partes.

Empresa

Herramienta interna. Sustituir una hoja de cálculo que ya no da más de sí. En modo Plan.

Hoy gestionamos las reservas de salas con una hoja de cálculo. Quiero una aplicación interna sencilla con usuarios, reservas y avisos de solapamiento. Hazme las preguntas necesarias y escribe el SPEC.md antes de proponer tecnología.

Informe semanal automático. Datos que alguien copia a mano cada lunes.

Crea una skill informe-semanal que lea los CSV de ventas/, calcule los indicadores que te detallo y genere un informe en Markdown. Después prepara una tarea programada para cada lunes a las 8:00.

Documentación del código. Para que alguien nuevo pueda empezar.

Genera la documentación para desarrolladores de este repositorio: README con instalación y arranque, arquitectura con un diagrama Mermaid, y un AGENTS.md con las normas que deduces del código. Marca lo que no puedas confirmar.

Revisión de cada pull request. Calidad homogénea en el equipo.

Escribe en AGENTS.md una sección «Code Review Rules» con nuestras normas: [tus normas]. Prepara un workflow con la GitHub Action de Codex que revise cada PR con esas reglas y deje un comentario.

Preguntas frecuentes

¿Necesito saber programar?

Basta con moverte con soltura por una terminal y leer código sencillo para revisar lo que hace el agente. Si nunca has abierto una terminal, empieza por «Trabaja con ChatGPT y Codex».

¿Qué cuenta necesito?

ChatGPT Plus o superior para hacer todas las prácticas con tu cuenta. Con el plan gratuito, Codex funciona en la app de escritorio con GPT-6 Luna, pero el límite se acaba pronto; puedes seguir cada práctica desde su etiqueta en el repositorio y rehacerla después.

¿Puedo usar los ordenadores del Cibernàrium?

Sí, siempre que estén habilitados para instalar la app de ChatGPT, Codex CLI, Git, Node.js y GitHub CLI. Recomendamos traer ordenador propio: la configuración queda hecha para seguir trabajando después.

¿App de escritorio o terminal?

Las dos. La app es más cómoda para revisar diffs, usar el navegador integrado, worktrees y tareas programadas. La terminal es imprescindible para codex exec, perfiles, hooks y trabajo en servidores. Comparten configuración.

¿Qué modelo uso?

GPT-6.1 Sol con razonamiento Medium para casi todo. GPT-6 Luna para tareas mecánicas o cuando vas justo de límite. Astra o niveles altos de razonamiento solo para problemas difíciles.

¿Codex puede romper algo en mi ordenador?

Con el sandbox por defecto solo escribe dentro del proyecto y no sale a internet sin preguntarte. Trabaja con Git y haz commits a menudo: cualquier cambio se puede deshacer. No uses Acceso completo fuera de entornos aislados.

¿OpenAI entrena con mi código?

En cuentas personales depende de tu ajuste en Controles de datos. En Business, Enterprise y Edu, no por defecto. En el curso trabajamos con datos ficticios.

¿Qué diferencia hay entre AGENTS.md y una skill?

AGENTS.md son normas que se aplican siempre. Una skill es un procedimiento que se carga solo cuando hace falta, y puede incluir scripts.

¿Funciona en Windows?

Sí. La app, la CLI y la extensión de VS Code funcionan de forma nativa en Windows con PowerShell y un sandbox propio. WSL2 es opcional.

¿Qué pasa si se me acaba el límite en clase?

Cambia a GPT-6 Luna, que gasta mucho menos, o continúa desde la etiqueta de la práctica (`git switch -c mi-pX pX-...`). El formador hace cada práctica en directo.

¿Tengo que pagar la GitHub Action?

La Action usa una clave de API que se factura por uso, aparte de la suscripción. En el curso la muestra el formador. La revisión con @codex review en GitHub sí entra en el plan Plus.

¿Cuál es la diferencia con Claude Code?

Es el agente equivalente de Anthropic. El Cibernàrium ofrece «Claude Code para perfiles técnicos» con la misma estructura y el mismo caso, para comparar las dos herramientas con criterio.

Glosario

Término	Definición
Agente	Un modelo que no solo responde: decide acciones (leer, ejecutar, editar), las hace y revisa el resultado hasta terminar la tarea.
Codex	El agente de programación de OpenAI. Modo de la app de escritorio de ChatGPT, CLI, extensión de IDE y servicio en la nube.
Codex CLI	Codex en la terminal. Se instala con el script oficial, npm o Homebrew y se abre con codex.
Codex cloud	Codex ejecutándose en una máquina virtual de OpenAI sobre un repositorio de GitHub, con tareas en paralelo y varios intentos.
GPT-6.1 Sol	Modelo recomendado para Codex desde el 29 de septiembre de 2026.
GPT-6 Luna	Modelo rápido y eficiente; el disponible en los planes Free y Go.
GPT-6 Astra	El modelo más capaz, con uso limitado en Plus.
Nivel de razonamiento	Cuánto piensa el modelo antes de actuar: Light, Medium, High, Extra High, Max y Ultra.
Ventana de contexto	Todo lo que el modelo tiene presente en un hilo: mensajes, ficheros leídos, salidas de comandos.
Límite de uso	Mensajes disponibles por ventana de cinco horas y por semana, según plan y modelo.
Sandbox	Perímetro técnico que limita qué ficheros puede escribir Codex y si puede salir a internet.
Política de aprobación	Cuándo te pregunta Codex antes de actuar fuera del sandbox: on - request o never.
Modo de permisos	En la app: Pedir aprobación, Aprobar por mí, Acceso completo o Personalizado.
Local	Codex trabaja directamente en tu carpeta del proyecto.
Worktree	Copia de trabajo del repositorio con su propia rama, para tareas en paralelo sin pisarse.

Término	Definición
Handoff	Pasar un chat de Worktree a Local, o al revés, con sus cambios.
Modo Plan	Codex pregunta y propone un plan sin editar ficheros. Se activa con <code>/plan</code> .
AGENTS.md	Fichero de instrucciones del proyecto que Codex lee antes de cada tarea.
Code Review Rules	Sección de AGENTS.md con las reglas que Codex aplica al revisar pull requests en GitHub.
config.toml	Fichero de configuración de Codex: modelo, razonamiento, permisos, servidores MCP, hooks.
Perfil	Fichero <code>~/.codex/<nombre>.config.toml</code> que se activa con <code>--profile nombre</code> .
Skill	Carpeta con un SKILL.md (y scripts opcionales) que enseña a Codex un procedimiento. Se invoca con <code>\$nombre</code> .
Hook	Comando que se ejecuta automáticamente en un evento del ciclo del agente, como después de editar un fichero.
Rules	Reglas experimentales que permiten, preguntan o prohíben comandos concretos.
Subagente	Agente con instrucciones propias que Codex lanza para una parte de la tarea, a menudo en paralelo.
MCP	Model Context Protocol: estándar para conectar un agente con herramientas externas mediante servidores.
Servidor MCP	Programa que expone herramientas (abrir una web, consultar documentación) que Codex puede usar.
Plugin	Paquete instalable de skills, servidores MCP, hooks y conexiones con aplicaciones.
codex exec	Ejecución de Codex sin interacción, para scripts y procesos automáticos.
Esquema JSON	Descripción de la forma que tiene que tener un JSON. Con <code>--output-schema</code> , la respuesta de Codex la cumple.
Tarea programada	Tarea que la app ejecuta a una hora o con una frecuencia fija, o cuando ocurre un evento.
GitHub Action de Codex	<code>openai/codex-action</code> : ejecuta Codex en GitHub Actions con una clave de API.
Repositorio	Carpeta de un proyecto con todo el historial de cambios gestionado por Git.
Commit	Conjunto de cambios guardado en el historial con un mensaje.
Rama	Línea de trabajo paralela dentro de un repositorio.
Pull request	Propuesta para juntar una rama con otra, con diff, comentarios y revisión.
Merge y conflicto	Juntar ramas. Hay conflicto cuando dos ramas cambian las mismas líneas y alguien tiene que decidir.
Diff	Comparación línea a línea entre dos versiones de un fichero.
Prueba automática	Código que comprueba que otro código hace lo que debe. Aquí, con <code>node:test</code> .

Término	Definición
GitHub Pages	Servicio de GitHub que publica una web estática desde un repositorio.
Vibe coding	Construir software describiendo lo que quieres y dejando que un agente escriba el código. Con criterio: especificar, verificar y revisar.
Inyección de prompts	Instrucciones maliciosas escondidas en contenido que el agente lee (una web, un fichero) para que haga algo que no le has pedido.

Documentación oficial

Empezar

- **Inicio rápido** · <https://learn.chatgpt.com/codex/quickstart>
- **App de escritorio** · <https://learn.chatgpt.com/codex/app>
- **Codex CLI** · <https://learn.chatgpt.com/codex/cli>
- **Extensión IDE** · <https://learn.chatgpt.com/codex/ide>
- **Codex cloud** · <https://learn.chatgpt.com/codex/cloud>
- **Windows** · <https://learn.chatgpt.com/codex/windows/windows-app>
- **Novedades** · <https://learn.chatgpt.com/codex/whats-new>
- **Changelog** · <https://learn.chatgpt.com/codex/changelog>

Modelos, planes y permisos

- **Modelos** · <https://learn.chatgpt.com/codex/models>
- **Elegir modelo** · <https://learn.chatgpt.com/codex/model-selection>
- **Precios y límites** · <https://learn.chatgpt.com/codex/pricing>
- **Modos de permisos** · <https://learn.chatgpt.com/codex/permission-modes>
- **Sandbox** · <https://learn.chatgpt.com/codex/sandboxing>
- **Help Center · Codex con tu plan** · <https://help.openai.com/en/articles/11369540>
- **Help Center · ChatGPT Work y Codex** · <https://help.openai.com/en/articles/20001275>
- **Precios de ChatGPT** · <https://chatgpt.com/pricing>

Configurar y ampliar

- **AGENTS.md** · <https://learn.chatgpt.com/codex/agent-configuration/agents-md>
- **config.toml básico** · <https://learn.chatgpt.com/codex/config-file/config-basic>
- **Referencia de configuración** · <https://learn.chatgpt.com/codex/config-file/config-reference>
- **Skills** · <https://learn.chatgpt.com/codex/build-skills>
- **Hooks** · <https://learn.chatgpt.com/codex/hooks>
- **MCP** · <https://learn.chatgpt.com/codex/extend/mcp>
- **Plugins** · <https://learn.chatgpt.com/codex/plugins>
- **Subagentes** · <https://learn.chatgpt.com/codex/agent-configuration/subagents>
- **Rules** · <https://learn.chatgpt.com/codex/agent-configuration/rules>

Git, nube y automatización

- **Modos Local, Worktree y Cloud** · <https://learn.chatgpt.com/codex/environments/modes>
- **Worktrees** · <https://learn.chatgpt.com/codex/environments/git-worktrees>
- **Entornos cloud** · <https://learn.chatgpt.com/codex/environments/cloud-environments>
- **Revisión de código** · <https://learn.chatgpt.com/codex/code-review>

- **Integración con GitHub** · <https://learn.chatgpt.com/codex/third-party/github>
- **codex exec** · <https://learn.chatgpt.com/codex/non-interactive-mode>
- **GitHub Action** · <https://learn.chatgpt.com/codex/github-action>
- **Tareas programadas** · <https://learn.chatgpt.com/codex/automations>

Referencia

- **Comandos de barra** · <https://learn.chatgpt.com/codex/reference/slash-commands>
- **Comandos de la CLI** · <https://learn.chatgpt.com/codex/reference/commands>
- **Glosario oficial** · <https://learn.chatgpt.com/codex/glossary>
- **Repositorio de Codex en GitHub** · <https://github.com/openai/codex>
- **Panel de uso** · <https://chatgpt.com/codex/settings/usage>