Totem.

CONSTRUIR · GUÍA 04

Cómo armar un equipo de agentes en Codex (que trabajan en paralelo y se revisan entre sí)

Un agente principal reparte el trabajo entre subagentes especializados, cada uno con su archivo y su límite, y devuelve un solo resultado revisado. Con los seis archivos de configuración listos para copiar a tu proyecto.

JULIO 2026 · V1.0 · MEGATOTEM.COM

CONSTRUIR

Cómo armar un equipo de agentes en Codex (que trabajan en paralelo y se revisan entre sí)

Un agente principal reparte el trabajo entre subagentes especializados, cada uno con su archivo y su límite, y devuelve un solo resultado revisado. Con los seis archivos de configuración listos para copiar a tu proyecto.

ACTUALIZADA JUL 2026 · V1.0 · 30 MIN · TOTEM

QUÉ TE LLEVÁS

- El equipo de cuatro agentes ya escrito: cuatro archivos `.toml` para copiar a tu proyecto
- La plantilla del brief maestro, que es donde se gana o se pierde el trabajo
- Los cinco prompts de la ejecución: delegar, pedir estado, desbloquear, sintetizar y revisar
- Las seis preguntas para saber si conviene delegar, y las señales de cuándo no
- Lo que cuesta de verdad: qué modelo va en cada rol, y por qué el explorador sale veinticinco veces menos que el revisor

ANTES DE EMPEZAR

Necesitás: una cuenta de ChatGPT con acceso a Codex y un proyecto propio donde probar. Con el plan Plus alcanza.
No necesitás: ser programador. El equipo se arma con archivos de texto, y el caso que documentamos no toca una línea de código.

Las capturas salen de **ChatGPT Work**, que es donde corrimos el caso: ahí la delegación se pide escribiéndola y se ve en un panel. Los archivos de configuración de los pasos 04 y 05 son los mismos en el CLI y en la extensión de IDE, porque comparten el `config.toml`. La tabla oficial de disponibilidad lista subagentes para Plus, Pro, Business, Enterprise y clave de API; Free y Go no figuran ahí.

Un equipo de agentes en Codex es un agente principal que reparte trabajo acotado entre subagentes especializados, los deja correr en paralelo y junta lo que devuelven en una sola respuesta. El nombre oficial del patrón es **flujo de subagentes**. Sirve cuando el trabajo se puede partir en ramas que no se esperan entre sí y cada rama produce algo verificable. Un pedido confuso no mejora por delegarlo: se multiplica por cuatro.

- **Para qué existen los subagentes.** Sobre todo, para sacar el ruido del hilo principal. Las exploraciones, los logs, las trazas y la salida de comandos entierran lo que importa —los requisitos, las restricciones, las decisiones— y la sesión se vuelve menos confiable a medida que se llena. La documentación oficial lo llama *contaminación* y *degradación* de contexto. Un subagente se lleva el ruido a su propio hilo y devuelve un resumen. Que además termine antes es una consecuencia, no el motivo.

OCHO PIEZAS QUE SE CONFUNDEN ENTRE SÍ

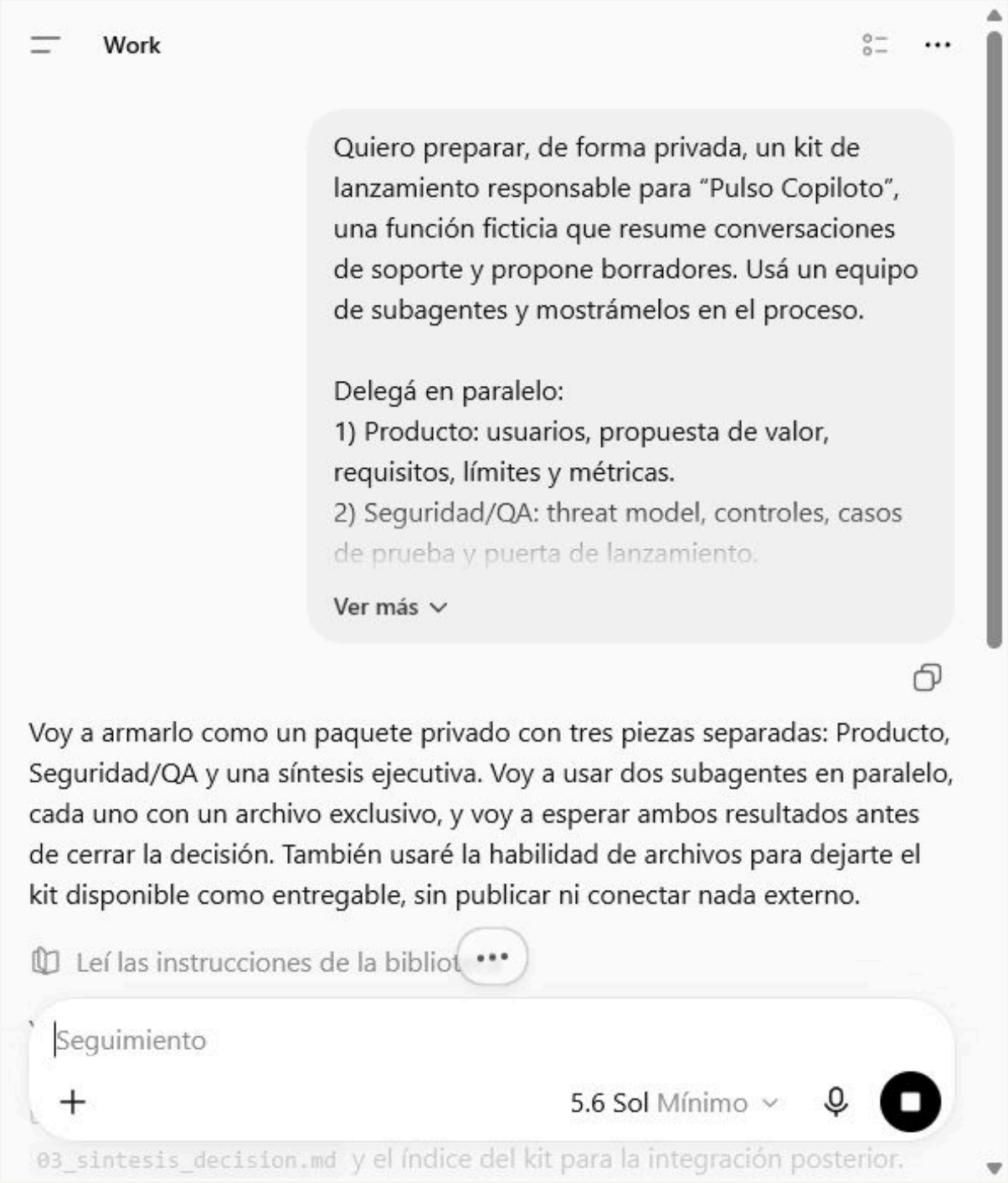
Casi todos los malentendidos con esto vienen de mezclar estas ocho cosas. La tabla se lee despacio una vez y después no hace falta volver.

PIEZA	QUÉ ES	QUÉ NO RESUELVE
Chat aparte	Una conversación independiente, para explorar sin ensuciar la principal.	No coordina ni sintetiza: al final juntás vos.
Subagente	Un agente delegado dentro del mismo flujo, con su propio hilo.	No reemplaza el encargo ni la revisión.
Trabajo en paralelo	Varias ramas avanzando a la vez.	Si dos escriben lo mismo, se pisan.
Agente personalizado	Un archivo `.toml` con nombre, instrucciones, modelo y sandbox propios.	No se activa solo: alguien tiene que pedir la delegación.
Tarea programada	Trabajo que corre después, a una hora o ante un disparador.	No es coordinación en vivo entre agentes.
Skill	Un procedimiento reutilizable con instrucciones y recursos.	No da acceso a datos externos por sí sola.
Plugin	Un paquete instalable de skills, herramientas y configuración.	Hay que instalarlo, habilitarlo y respetar las políticas del espacio.
Conector o MCP	La interfaz hacia una herramienta o un dato externo autorizado.	Nunca supone autorización automática, y suma contexto en cada mensaje.

> - **La palabra «equipo» es nuestra, no del producto.** El término oficial es *subagente*, y describe una unidad delegada dentro de una ejecución. No hay una consola de organización permanente ni un producto aparte que se llame así. Hablamos de equipo porque enseña mejor el reparto de responsabilidades.

El patrón mínimo de un equipo confiable

Cuatro roles alcanzan para casi todo. Lo que decide el resultado es que cada uno tenga un objetivo, una salida propia y un límite.



El agente principal declarando el reparto antes de arrancar. Los archivos van numerados — `01_`, `02_`, `03_` — y el tercero se lo reserva para la síntesis. Ese anuncio es el contrato: si no aparece, el reparto no quedó fijado.

ROL	RESPONDE POR	ENTREGA	NO HACE
Principal	Objetivo, decisiones, dependencias, integración	El brief y el paquete final	Delegar la decisión de fondo
Explorador	Mapear el terreno y juntar evidencia	Un resumen al hilo principal	Proponer arreglos ni editar
Producto	Usuarios, alcance, exclusiones, métricas	`producto.md`	Evaluar seguridad ni aprobar
Seguridad y QA	Amenazas, controles, pruebas, puerta de salida	`riesgos.md`	Publicar ni ampliar permisos
Revisor	Contradicciones, cobertura, trazabilidad	Hallazgos priorizados P0 a P3	Reescribir los entregables
Vos	Autoridad, permisos, aceptación	Aprobaciones explícitas	Ceder credenciales

Lo que hace funcionar un equipo es el **contrato de delegación**: objetivo, entradas, salida, límites, evidencia, criterio de terminado y ruta de escalamiento. Siete piezas. Cuando un subagente se sale del carril, casi siempre falta una, y la que más falta es la última.

La ejecución que documentamos delegó sólo dos de esos roles, producto y seguridad, que eran los únicos que podían avanzar sin esperarse. La exploración y la revisión las hizo el principal. Un equipo se arma con los roles que el trabajo pide, no con los seis de la tabla.

Cuándo conviene, y cuándo empeora las

cosas

Delegar tiene un precio fijo en coordinación que se paga antes de ver ningún beneficio. Vale la pena cuando hay al menos dos ramas independientes, cada una produce un artefacto verificable y la forma de la síntesis se puede definir de antemano.

BUEN ENCAJE

Explorar un código que no conocés. Revisar una rama por dimensiones separadas. Triangular fuentes. Analizar logs. Probar en paralelo. Cualquier trabajo donde el ruido intermedio arruinaría el hilo principal.

El común denominador: son tareas de **lectura**.

MAL ENCAJE

Una pregunta corta. Una edición chica sobre un archivo que ya conocés. Una secuencia donde B necesita el resultado de A. «Que tres agentes voten» para decidir un hecho.

Tres agentes pueden repetir el mismo error con la misma seguridad. La convergencia no es evidencia.

Lo que cuesta de verdad

La documentación oficial lo dice sin vueltas: cada subagente hace su propio trabajo de modelo y de herramientas, así que un flujo de subagentes consume más que una ejecución comparable con un solo agente. La latencia baja; el consumo sube. Vale saber en qué proporción antes de armar un equipo de seis.

MODELO	CRÉDITOS POR MILLÓN DE TOKENS DE ENTRADA	DE SALIDA	PARA QUÉ ROL
`gpt-5.6-sol`	125	750	Revisor, seguridad
`gpt-5.6-terra`	50	300	Producto, trabajo de todos los días
`gpt-5.6-luna`	5	30	Explorador, barridos de lectura

Entre el más chico y el más grande hay veinticinco veces de diferencia por token de entrada. Un explorador que lee cuarenta archivos y devuelve diez líneas es exactamente el trabajo que no justifica el modelo caro.

- **Dónde ver lo que te queda.** Dentro de una sesión del CLI, `/status`. Los límites publicados para Plus van de 10 a 100 mensajes locales por ventana de cinco horas con `gpt-5.6-sol`, y de 250 a 2.000 con `gpt-5.6-luna`. El rango es tan ancho porque un mensaje con mucho contexto y muchas herramientas cuenta muy distinto que uno corto.

01

Decidí si el trabajo se puede partir

Seis preguntas, antes de escribir nada. Están enteras en el kit, pero las dos que más deciden son estas: *¿hay dos ramas que pueden avanzar sin esperarse?* y *¿cada agente puede ser dueño de su propio archivo?*

Si el paso B necesita el resultado de A, no hay paralelo posible: hay una secuencia. Y una secuencia la hace mejor un solo agente que ya tiene el contexto encima, sin pagar el traspaso.

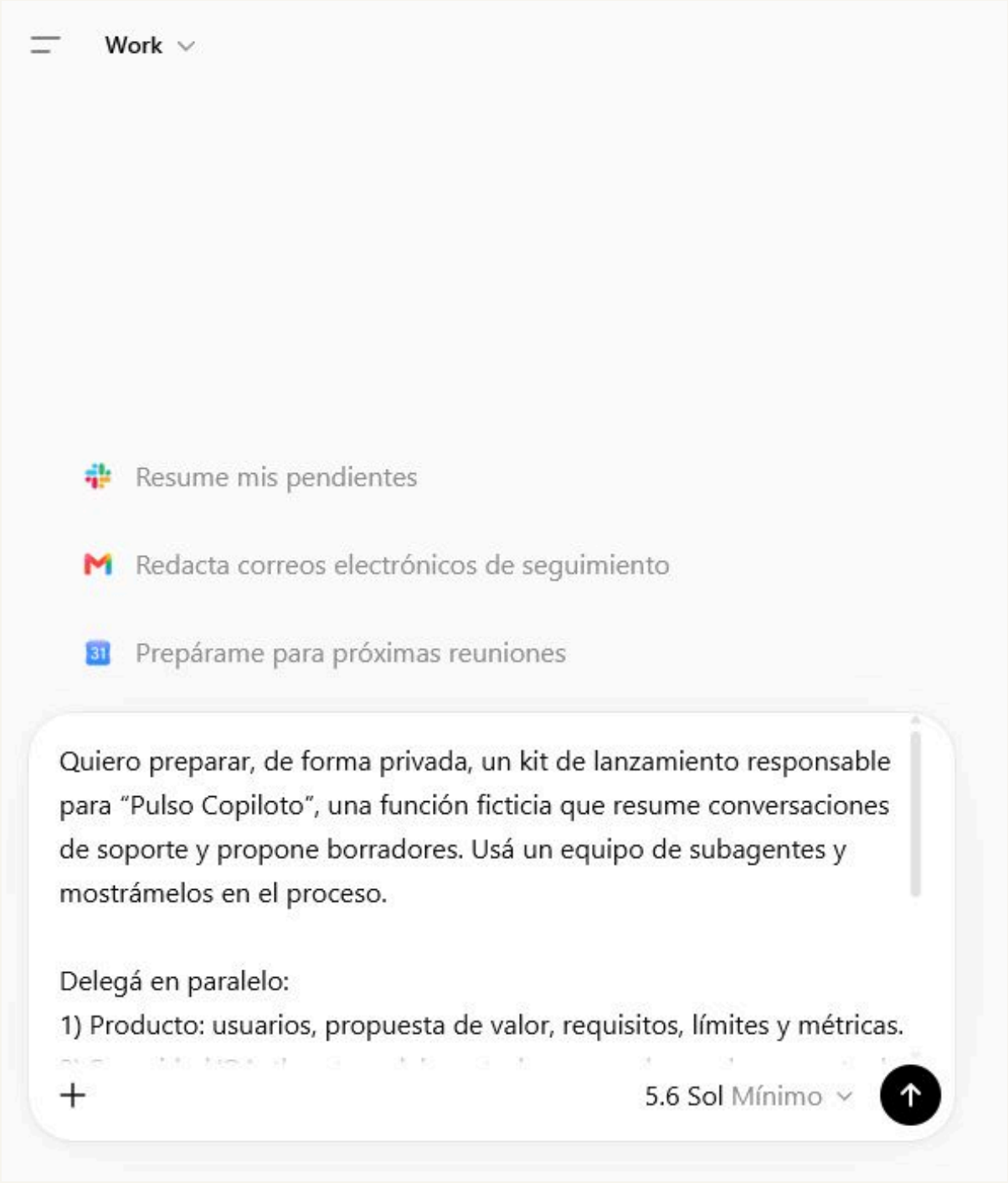
Con cinco o seis respuestas afirmativas, armá el equipo. Con tres o cuatro, delegá una sola rama —la más ruidosa— y hacé el resto vos en el hilo principal. Con menos de tres, un agente solo. Vas a llegar antes y más barato.

02

Escribí el brief maestro

El brief es el mensaje que abre el turno, antes de que se delegue nada. Tiene cinco partes y ninguna es decorativa: objetivo, contexto, entregables, restricciones y criterio de terminado.

La parte que más se saltea es la de restricciones, y es la que evita los tres accidentes caros: que dos agentes escriban el mismo archivo, que alguien afirme algo sin fuente y que un subagente se quede esperando un permiso que vos nunca viste.



El encargo real, escrito y todavía sin enviar. Abajo a la derecha, junto al botón de envío, está el selector de modelo y esfuerzo: decía **5.6 Sol · Mínimo**. Coordinar dos subagentes no pidió el esfuerzo alto.

BRIEF MAESTRO · PLANTILLA

```
Objetivo: [el resultado concreto, en una oración]

Contexto: [qué es este proyecto, en qué estado está, qué decidimos antes]

Entregables:
- [artefacto 1: nombre de archivo y qué tiene adentro]
- [artefacto 2]
- una síntesis con decisiones, evidencia, contradicciones e incertidumbres

Restricciones:
- Cada agente edita únicamente su archivo asignado.
- Toda afirmación factual lleva fuente, archivo y línea, o se marca como
supuesto.
- Ninguna herramienta externa se considera autorizada por defecto.
- Escalá a mí si hace falta una credencial, un dato personal, un permiso
nuevo
  o un cambio fuera de este directorio.

Criterio de terminado:
- Los artefactos están completos y sin contradicciones entre sí.
- Los riesgos están priorizados y cada uno tiene dueño.
- Cada afirmación importante es rastreable hasta su fuente.
- Existe una recomendación explícita: avanzar, iterar o parar.
```

- **Las instrucciones de estilo no van acá.** Van en el `AGENTS.md` del proyecto, que Codex lee una vez al arrancar la sesión. Repetirlas en cada encargo las hace pagar dos veces.

03

Delegá con fronteras: un archivo por agente

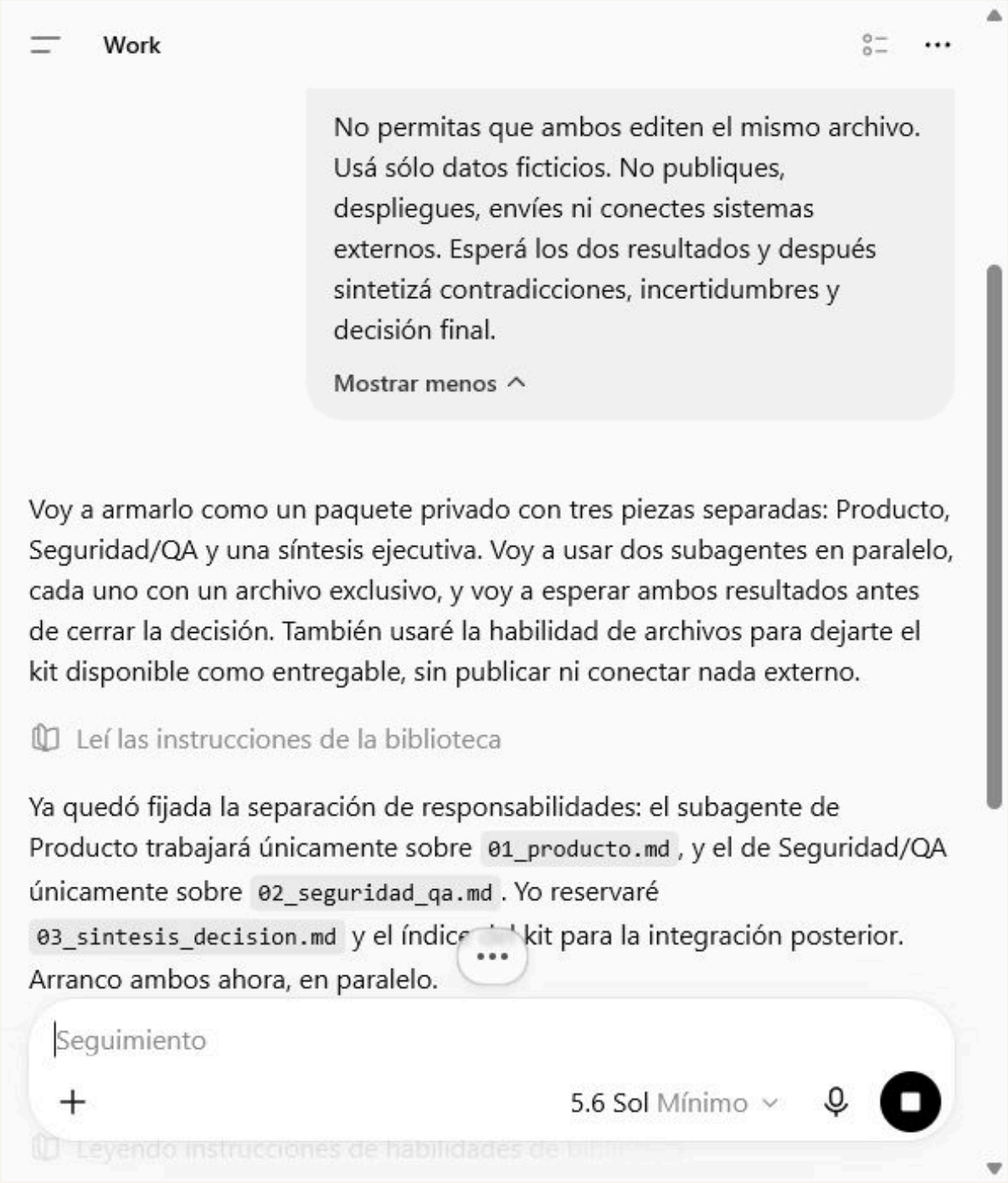
Un buen pedido de delegación dice tres cosas: cómo se parte el trabajo, si hay que esperar a todos antes de seguir, y qué formato tiene el resumen que volvés a recibir. Si falta alguna, Codex la completa por su cuenta y no siempre como esperabas.

EL PEDIDO DE DELEGACIÓN

```
Usá subagentes para esto. Delegá en paralelo:

1) producto: usuarios, problema, alcance, exclusiones y métricas.
   Entregá producto.md.
2) seguridad_qa: amenazas, controles, casos de prueba y puerta de salida.
   Entregá riesgos.md.

Ninguno de los dos toca el archivo del otro. Esperá los dos resultados.
Después
compará supuestos, resolvé las contradicciones que encuentres, marcá lo que
quede incierto y armá una síntesis. No conectes sistemas externos.
```



«Arranco ambos ahora, en paralelo». Arriba, la parte del encargo que fija los límites: nadie edita el archivo del otro, nada se publica y hay que esperar los dos resultados antes de cerrar.

En el CLI también se puede pedir un agente por punto, que es el formato que mejor rinde para revisar una rama:

UN AGENTE POR PUNTO

```
Revisá esta rama contra main con subagentes en paralelo. Un agente por
punto:

1. Riesgos de seguridad
2. Huecos de test
3. Mantenibilidad
4. Condiciones de carrera

Esperá a los cuatro y resumí los hallazgos por categoría, con archivo y
línea.
```

- **La escritura en paralelo es el terreno resbaladizo.** La recomendación oficial es empezar por trabajos de lectura —exploración, tests, triage, resúmenes— y ser cuidadoso con los flujos donde varios agentes editan a la vez. Si tenés que escribir con varios, dale a cada uno su archivo, su rama o su worktree, y que integre uno solo.

04

Armá el equipo fijo con archivos `.toml`

Todo lo que viste hasta acá se armó escribiéndolo en el pedido, que es como corrimos el caso: los subagentes se llamaron `Producto` y `Seguridad qa` porque así los nombró el encargo. Sirve para probar. Para no volver a explicarlo nunca más, el equipo se deja escrito: un archivo `.toml` por agente.

Van en `.codex/agents/` dentro del proyecto, o en `~/.codex/agents/` si querés el mismo equipo en todos lados. Codex ya trae tres de fábrica — `default`, `worker` y `explorer` — y si tu agente se llama igual que uno de ellos, gana el tuyo.

DÓNDE VA CADA ARCHIVO

```
tu-proyecto/
├── AGENTS.md
└── .codex/
    ├── config.toml
    └── agents/
        ├── explorador.toml
        ├── producto.toml
        ├── seguridad-qa.toml
        └── revisor.toml
```

`AGENTS.md` y `config.toml` tienen nombre fijo: Codex los busca ahí y si los renombrás no los lee. Los de `agents/` no, porque cada agente se identifica por el campo `name` de adentro. Que el nombre del archivo coincida es una comodidad para vos cuando abras la carpeta dentro de seis meses.

Cada archivo necesita tres campos y admite varios más. Los obligatorios son `name`, `description` y `developer_instructions`. Los útiles son `model`, `model_reasoning_effort` y `sandbox_mode`.

.CODEX/AGENTS/SEGURIDAD-QA.TOML

```
name = "seguridad_qa"
description = "Seguridad y calidad. Amenazas, controles, casos de prueba y
puerta de salida. Nunca publica ni amplía permisos."
model = "gpt-5.6-sol"
model_reasoning_effort = "high"
sandbox_mode = "read-only"
developer_instructions = """
Escribís un solo archivo: el que te asigne el agente principal. No toques
ningún otro.

Cada amenaza va con probabilidad e impacto de 1 a 5, el control que la
mitiga y
la evidencia concreta que haría falta para darla por aceptada.

Nunca publiques, despliegues, envíes ni amplíes un permiso. Si algo lo
necesita, pará y escalá al agente principal con la razón.
"""
```

Y una línea de configuración del proyecto, en `.codex/config.toml`, para poner un techo:

.CODEX/CONFIG.TOML

```
[agents]
max_concurrent_threads_per_session = 4
default_subagent_model = "gpt-5.6-terra"
default_subagent_reasoning_effort = "medium"
```

Nosotros lo dejamos en cuatro. Subirlo no acelera: multiplica el consumo y te devuelve más informes de los que vas a leer con atención. Si querés apagar el multiagente en un proyecto, `enabled = false` en la misma sección.

- **El formato puede cambiar.** La propia documentación admite que cargar cada agente como una capa de configuración «se siente más pesado que un manifiesto dedicado» y que el formato puede

evolucionar a medida que madure la forma de escribir y compartir agentes. Guardá los archivos en el repositorio: cuando cambie, vas a querer ver el diff.

05

Elegí modelo y esfuerzo por rol

Acá se decide la mitad del costo y buena parte de la calidad. La regla corta: el agente que lee mucho y decide poco va en el modelo más chico; el que tiene que encontrar lo que nadie vio va en el más grande.

ROL	MODELO	ESFUERZO	SANDBOX	POR QUÉ
Explorador	`gpt-5.6-luna`	medium	read-only	Lee mucho, decide poco, devuelve poco
Producto	`gpt-5.6-terra`	medium	workspace-write	Redacta con criterio, sin ambigüedad extrema
Seguridad y QA	`gpt-5.6-sol`	high	read-only	Tiene que anticipar lo que no está escrito
Revisor	`gpt-5.6-sol`	high	read-only	Buscar contradicciones exige seguir hilos largos

Los niveles de esfuerzo van de `low` a `max`, con `ultra` arriba de todo en los modelos que lo admiten. Usá el más bajo que te dé el resultado que necesitás y subí sólo cuando el trabajo pida planificar, contrastar o revisar supuestos.

Quién gana cuando hay tres configuraciones

Es la duda que aparece la primera vez que un agente se comporta distinto de lo esperado. El orden está documentado: si el archivo del agente define `model` o `model_reasoning_effort`, ese valor manda. Si no lo define, Codex resuelve cada ajuste por separado: primero el valor explícito de la

delegación, después el de `[agents]`, y por último el del agente padre. Lo demás —`sandbox_mode`, servidores MCP, skills— se hereda del padre cuando el archivo no lo dice.

- **La excepción que sorprende.** Los ajustes que hacés en vivo durante la sesión —un `/permissions`, un `--yolo`— se vuelven a aplicar al hijo *aunque* el archivo del agente declare otra cosa. Un agente marcado como sólo lectura no te protege de una sesión que abriste en modo permisivo.

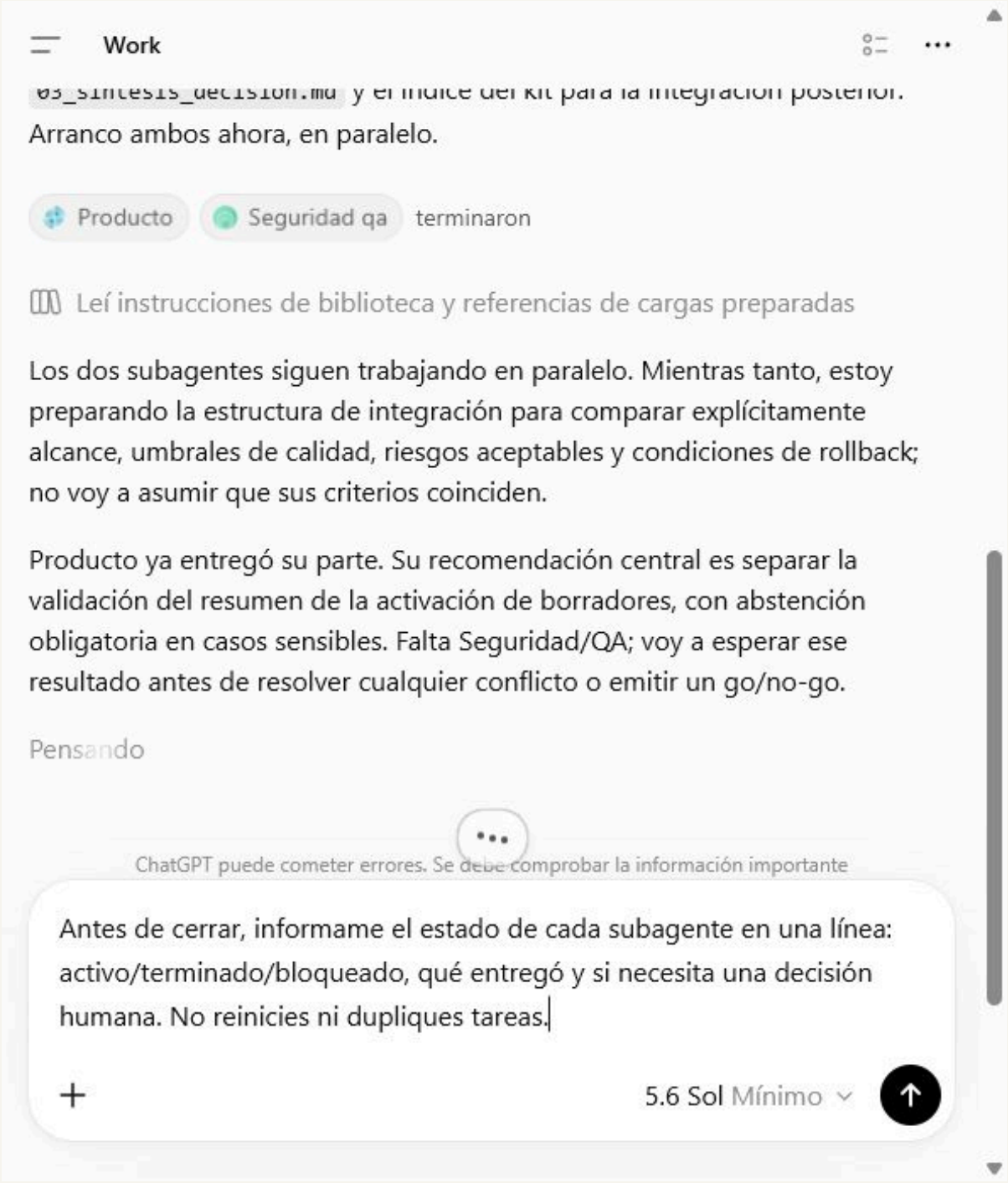
06

Seguí el trabajo sin ahogarlo

Preguntar «¿cómo va?» cada dos minutos no acelera nada y le come contexto al principal, que es justamente lo que estabas tratando de cuidar. Pedí eventos.

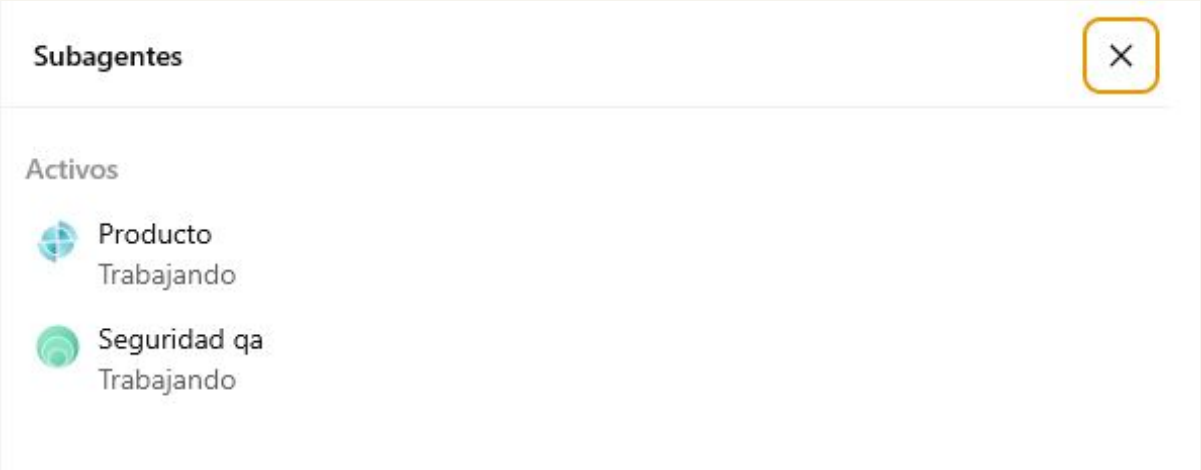
SEGUIMIENTO POR EVENTOS

```
Informá sólo cuando cambie el estado: iniciado, bloqueado, necesita decisión
o
terminado. Si estás bloqueado, devolvé cuatro cosas: causa, evidencia,
impacto
y la decisión mínima que necesitás de mí. No amplíes permisos por tu cuenta.
```



El seguimiento se escribe sin frenar nada: los agentes siguen mientras vos tipeás. En la conversación de arriba se ve el reporte que devuelve el principal, incluido el aviso de que todavía falta uno.

Mientras trabajan, el estado vive en un panel aparte:



El panel **Subagentes**, con los dos en **Trabajando**. Es de sólo lectura: no hay botón para frenar uno. Si querés detener a uno solo, se lo pedís al principal por chat.

Cada superficie muestra el trabajo de manera distinta, y conviene saber cuál te deja intervenir:

DÓNDE	QUÉ VES	QUÉ PODÉS HACER
CLI	`/agent` recorre los hilos activos	Abrir un hilo, cambiar de hilo, pedirle a Codex que frene o redirija uno
App de escritorio	Listas de Activos y Terminados, de sólo lectura	Inspeccionar el resultado; para frenar uno hay que pedírselo a Codex
IDE	Los subagentes activos, arriba del compositor	Ver estado, frenar todos, abrir un hilo

- **El detalle que nadie te cuenta hasta que te pasa.** En el CLI, un pedido de aprobación puede aparecer desde un hilo que no estás mirando. El aviso trae la etiqueta del hilo de origen y con la tecla `o` lo abrís antes de decidir. Aprobar sin mirar de dónde viene es cómo se autoriza algo que no querías autorizar.

En flujos no interactivos pasa lo contrario y es sano: una acción que necesita una aprobación nueva falla, y el error sube al flujo padre en vez de quedarse esperando para siempre.

07

Consolidá sin concatenar

Este es el paso donde se pierde el trabajo que costó cuatro agentes. Pegar los informes uno abajo del otro produce un documento largo con dos afirmaciones incompatibles a doce páginas de distancia, y nadie las va a cruzar.

El aviso de fin son dos chips con la palabra «terminaron». Debajo, la frase que conviene exigir siempre: el principal preparando la comparación *antes* de tener los dos resultados, y avisando que no va a asumir que los criterios coinciden.

LA SÍNTESIS

Consolidá los entregables. No los concatenes.

1. Listá las contradicciones entre los documentos, con la cita de cada lado.
2. Resolvé cada una y decí con qué criterio la resolviste.
3. Marcá lo que quedó incierto, sin taparlo con una redacción prolija.
4. Devolvé: decisiones tomadas, evidencia de cada una, riesgos abiertos con dueño, y una recomendación explícita: avanzar, iterar o parar.

Cuando dos agentes se contradicen, no promedies. Andá a la evidencia de cada uno y al criterio de aceptación. Si la diferencia es material y no se resuelve mirando fuentes, sube a vos: eso es una decisión, no un empate.

La contradicción real de esta ejecución. No estaba en el objetivo: los dos querían lo mismo. Estaba en los umbrales, en la tolerancia por segmento y en la retención. Y el criterio con el que la resolvió: el más estricto por dimensión, con las incertidumbres dejadas como bloqueantes explícitos.

Esa última parte es la que separa una síntesis buena de una prolija. Una incertidumbre que sobrevive redactada como si fuera una conclusión es peor que una contradicción visible, porque ya nadie la va a revisar.

08

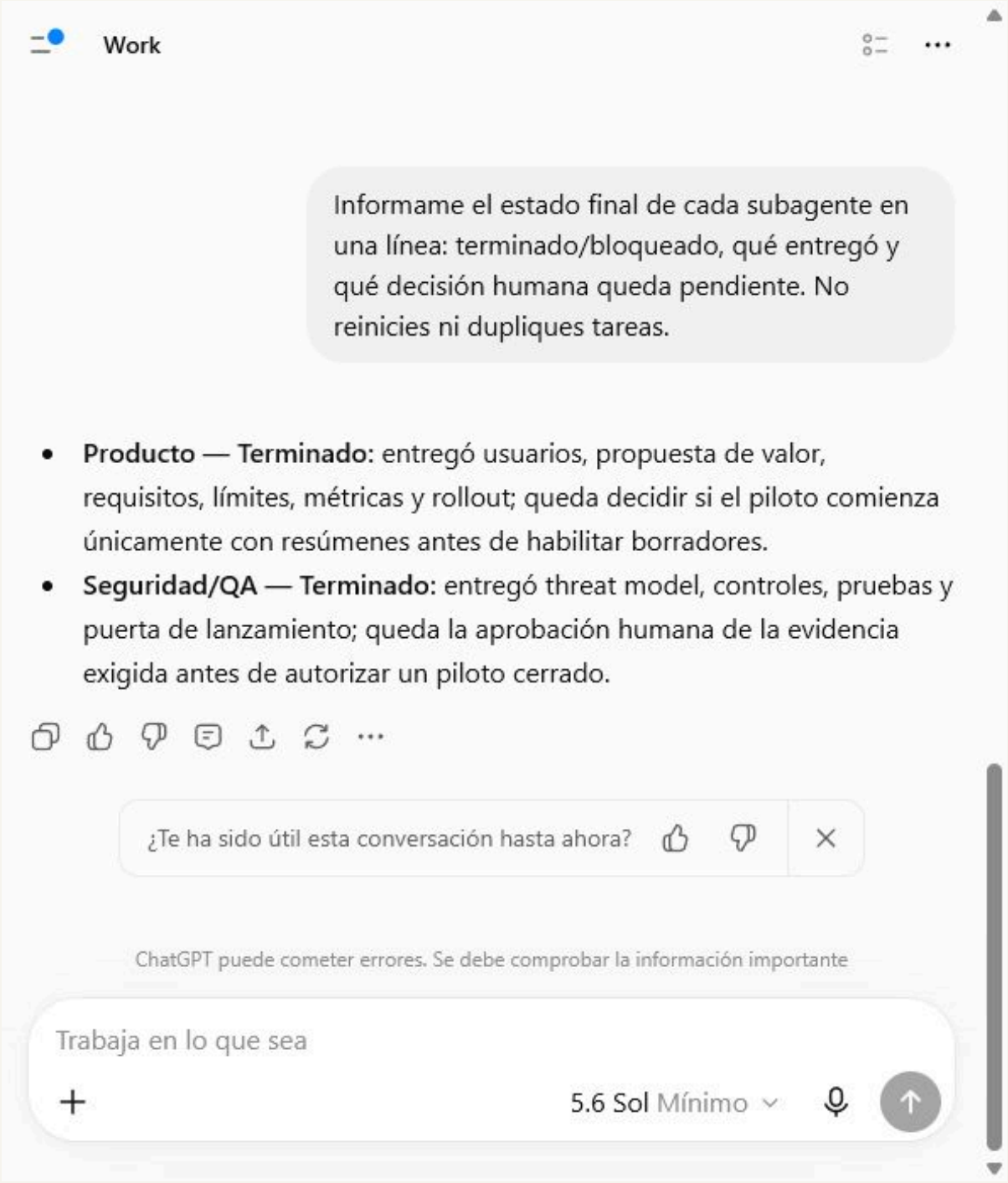
Cruzá la puerta de calidad

«Los agentes terminaron» y «el trabajo está bien» son dos afirmaciones distintas. La segunda necesita un revisor que no haya escrito nada de lo que revisa.

REVISOR INDEPENDIENTE

```
Revisá estos entregables sin reescribirlos. Buscá: contradicciones entre
requisitos y controles; afirmaciones sin evidencia; riesgos sin dueño;
métricas
que incentiven el comportamiento equivocado; permisos asumidos; pruebas sin
resultado esperado; y decisiones irreversibles.

Priorizá P0 a P3, citá archivo y sección, y proponé la corrección mínima. Si
no
hay hallazgos, explicá qué verificaste.
```



El cierre que sirve. Cada agente terminado, qué entregó, y —lo que importa— qué decisión humana queda pendiente en cada caso. Sin esa última parte, «terminado» se lee como «aprobado».

La última línea del prompt es la que más rinde. Un revisor que dice «está todo bien» no informó nada; uno que enumera qué verificó te muestra, sin querer, qué quedó sin revisar.

Que un agente se active solo, sin que se lo pidas

Hasta acá la delegación se pide en cada turno. Hay dos formas de que ocurra sin pedirla, y conviene conocer las dos porque se comportan distinto.

La primera es el `AGENTS.md` del proyecto. Codex lo lee antes de trabajar y, si el archivo pide delegación para cierto tipo de tarea, delega. Es la forma que preferimos: queda escrita, se versiona con el repositorio y la puede leer cualquiera del equipo.

EL BLOQUE QUE VA EN AGENTS.MD

```
## Cuándo delegar en subagentes

Delegá en paralelo cuando el trabajo tenga dos o más ramas que puedan
avanzar
sin esperarse y cada una produzca un archivo distinto. Casos típicos acá:

- Revisión de una rama: un agente por dimensión.
- Investigación de un bug: uno reproduce, otro mapea el código, otro
corrige.
- Auditoría de una carpeta grande: un agente por subsistema.

No delegues para una edición chica, para una pregunta con una sola fuente de
verdad, ni cuando el paso B necesita el resultado del paso A.

Reglas del equipo:
- Cada subagente es dueño de un archivo. Nadie escribe en el archivo de
otro.
- Los subagentes de exploración y revisión van en modo lectura.
- Todo hallazgo se devuelve con archivo y línea.
- Si un subagente necesita un permiso nuevo, para y escala. No lo amplía
solo.
```

La segunda es **Ultra**, el nivel de razonamiento más alto, y el único donde Codex decide por su cuenta que conviene repartir. En el resto de los niveles la delegación se pide de forma explícita, que mientras estás aprendiendo es preferible: ves qué se reparte y por qué. Ultra depende de

la cuenta y del modelo, y en la app de escritorio hay que habilitarlo primero en **Ajustes → Configuración**, en la opción del selector de modelos.

- **Cuidado con el largo del `AGENTS.md`.** Todo lo que escribas ahí entra en el contexto de cada mensaje y de cada subagente. El tope por defecto entre todos los archivos de instrucciones es de 32 KiB; pasado eso, Codex deja de sumar y no te avisa con un cartel. Las reglas de una subcarpeta van en esa subcarpeta, no en la raíz.

Permisos: lo que se hereda y lo que no

Los subagentes heredan la política de sandbox y el modo de permisos del turno principal. De ahí salen dos consecuencias prácticas que conviene tener presentes antes de delegar, no después.

La primera: **el modo se elige antes**. Si vas a repartir trabajo, decidí el modo del turno padre y recién después pedí la delegación. Los cuatro modos son Pedir aprobación (el razonable para empezar), Aprobar por mí, Acceso total y Personalizado desde `config.toml`. Los dos del medio hay que habilitarlos en los ajustes antes de que aparezcan en el menú.

La segunda: **heredar el sandbox no es heredar autorizaciones**. Que la sesión esté autenticada no habilita Gmail, GitHub, Drive, un repositorio privado ni producción. Cada herramienta, conector y sitio mantiene sus propios requisitos, y cambiar quién revisa un pedido no agranda el sandbox.

- **Las credenciales no van en el prompt.** Ni en el brief, ni en el `AGENTS.md`, ni en el archivo de un agente. Empezá en lectura, habilitá escritura sólo para el archivo o el repositorio que hace falta, y dejá publicar y desplegar como una aprobación humana aparte.

LOS DIEZ PATRONES DE FALLO

Todos aparecen antes de la primera hora. Los primeros cinco son de diseño del encargo; los últimos cinco, de ejecución.

1. Brief ambiguo. Cada agente resuelve un problema distinto y ninguno el tuyo.
2. Fronteras superpuestas. Dos agentes editan el mismo archivo y el que tarda más pierde.
3. Contexto insuficiente. El delegado no conoce una restricción que para vos era obvia.
4. Falsa independencia. Una rama necesitaba un dato que la otra todavía no produjo.
5. Final sin dueño. Nadie responde por la coherencia del conjunto.

6. Síntesis por pegado. El principal concatena y las contradicciones sobreviven.
7. Falta de evidencia. Conclusiones sin fuente, sin prueba y sin archivo.
8. Permisos excesivos. Se habilitó red o escritura global «por comodidad».
9. Aprobación huérfana. Un subagente quedó detenido esperando una autorización que nadie vio.
10. Convergencia aparente. Los cuatro coinciden porque los cuatro partieron del mismo supuesto equivocado.

Si algo sale mal

Un agente quedó esperando una aprobación que nunca viste.
En el CLI el aviso puede venir de un hilo que no estás mirando. Trae la etiqueta del hilo de origen y con `o` lo abrís antes de decidir.

Un agente no puede escribir donde debería.

Heredó el modo del turno padre. Se elige antes de delegar; cambiarlo a mitad de camino no reconfigura a los hijos que ya arrancaron.

Dos agentes editaron el mismo archivo.
Faltó dueño único. Repartí de nuevo y, si es código, dale a cada uno su rama o su worktree.

Volvieron dos informes que dicen lo mismo.
El brief no separaba bien los objetivos. Es más barato reescribir el brief que fusionar los informes.

El equipo se comió el presupuesto de la tarde.
Bajá `max_concurrent_threads_per_session`, pasá los agentes de lectura a `gpt-5.6-luna` y acortá el `AGENTS.md`. Cada servidor MCP que tengas prendido también suma contexto en cada mensaje: apagá los que no uses.

Codex ignora las instrucciones que acabás de escribir.
La cadena de instrucciones se arma al arrancar y no hay caché que limpiar: reiniciá la sesión en ese directorio. Si sigue igual, buscá un `AGENTS.override.md` más arriba en el árbol, que pisa al `AGENTS.md` de su carpeta.

Los cuatro coinciden y el resultado igual está mal.
Buscá el supuesto compartido. Si todos leyeron el mismo documento equivocado, la coincidencia es un eco.

Preguntas frecuentes

¿Un equipo de agentes es lo mismo que abrir varios chats?

No. Dos chats separados no comparten un encargo ni producen una síntesis: al final juntás vos. En un flujo de subagentes, el agente principal reparte, espera y consolida dentro de una misma ejecución.

¿Más agentes dan mejores respuestas?

Sólo cuando el trabajo es divisible y verificable. En tareas chicas o muy secuenciales, agregar agentes suma costo y coordinación. Tres agentes tampoco sirven para votar un hecho: pueden repetir el mismo error con la misma seguridad.

¿Pueden trabajar sobre el mismo repositorio?

Sí, pero la escritura concurrente aumenta los conflictos. La recomendación oficial es empezar por tareas de lectura y ser cuidadoso con los flujos de escritura en paralelo. Si varios tienen que escribir, dale a cada uno su archivo, su rama o su worktree, y que integre uno solo.

¿Los subagentes heredan mis permisos?

Heredan la política de sandbox y el modo de permisos del turno principal, así que el modo se elige antes de delegar. Estar autenticado no equivale a estar autorizado: cada herramienta, conector o sitio mantiene sus propios requisitos.

¿Cuánto más caro sale un equipo que un solo agente?

Más, y la documentación oficial lo dice sin vueltas: cada subagente hace su propio trabajo de modelo y de herramientas. La latencia baja con el paralelo; el consumo sube. Se controla eligiendo el modelo por rol, limitando los hilos concurrentes y acortando el `AGENTS.md`, que se paga en cada turno y en cada subagente.

¿Necesito el CLI o sirve la app de escritorio?

Los dos funcionan y comparten el mismo `config.toml`. El CLI da el control más fino: `/agent` recorre los hilos activos. En la app de escritorio la lista de subagentes es de sólo lectura y para frenar uno hay que pedírselo a Codex. En el IDE aparecen arriba del compositor cuando ese panel está disponible.

¿Qué es Ultra y me hace falta?

Es el nivel de razonamiento más alto y el único que delega solo, sin que se lo pidas. En el resto de los niveles la delegación se pide de forma explícita, que para aprender es mejor: ves qué se reparte y por qué. Depende de la cuenta y del modelo, y en la app de escritorio hay que habilitarlo en Ajustes.

¿Puedo usar un equipo de agentes con datos de la empresa?

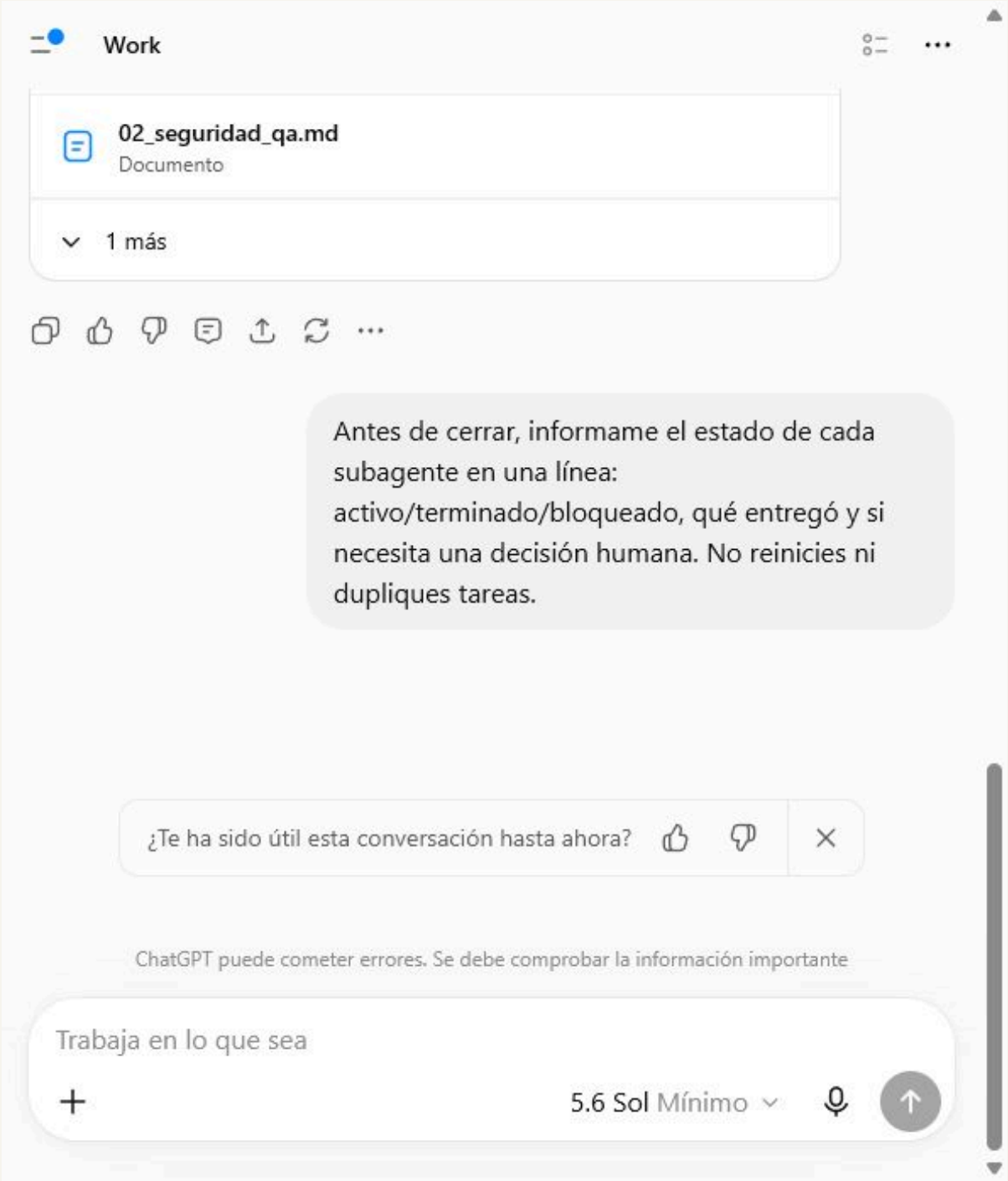
Sólo si la organización autoriza la superficie, la herramienta y el tipo de dato. Los subagentes no cambian esa ecuación: heredan el sandbox, pero no otorgan permisos. Empezá en modo lectura, habilitá escritura sólo donde haga falta y dejá la publicación como una aprobación humana aparte.

¿Cómo sé si el equipo funcionó?

Cuando cada requisito tiene dueño y prueba, cada riesgo alto tiene control o aceptación con nombre, ninguna acción externa ocurrió sin autorización previa, y el paquete se puede regenerar desde los prompts guardados. Que los agentes hayan terminado no dice nada sobre si el trabajo está bien.

De dónde salen los datos

El caso de esta guía se ejecutó el 31 de julio de 2026 en ChatGPT Work, en un espacio privado, con un agente principal y dos subagentes en paralelo sobre archivos separados, y datos enteramente ficticios. No se publicó nada, no se desplegó nada y no se conectó ningún sistema externo. Todas las capturas son de esa ejecución, sin retoques.



Los entregables, como archivos. Que el trabajo termine en documentos con nombre —y no en una respuesta larga de chat— es lo que después permite revisarlos, versionarlos y discutirlos por separado.

Al verificar los entregables después, dos comprobaciones no se pudieron correr y quedaron declaradas como pendientes en vez de darse por buenas, que es la parte más útil de contar. El documento final no se pudo revisar página por página porque el entorno no tenía LibreOffice instalado: quedó validado en estructura y accesibilidad, con cero hallazgos, pero sin inspección visual. Y la revisión interactiva del HTML quedó fuera de

alcance porque el navegador de esa sesión no abre URLs `file://`. Un equipo que devuelve «todo bien» cuando dos comprobaciones no se pudieron correr todavía no tiene puerta de calidad.

1. Subagents — documentación oficial de Codex (https://learn.chatgpt.com/docs/agent-configuration/subagents)
2. Custom instructions with AGENTS.md (https://learn.chatgpt.com/docs/agent-configuration/agents-md)
3. Permissions (https://learn.chatgpt.com/docs/permission-modes)
4. Models (https://learn.chatgpt.com/docs/models)
5. Pricing y límites de uso (https://learn.chatgpt.com/docs/pricing)
6. Codex CLI (https://learn.chatgpt.com/docs/codex/cli)

- **Cómo envejece esta guía.** Los nombres de los modelos, los límites por plan y la forma exacta de los paneles van a cambiar; ya cambiaron dos veces este año. El reparto por archivo, el contrato de delegación y la puerta de calidad no dependen de ninguna de esas cosas. Antes de repetirle una cifra a un cliente, mirá la fecha de la documentación oficial.

El kit

`README.md` — dónde va cada archivo y en qué orden usarlo

`checklist-antes-de-delegar.md` — las seis preguntas, y cómo se lee el resultado

`brief-maestro.md` — la plantilla del encargo, con un ejemplo completo

`contrato-de-delegacion.md` — las siete piezas y los cuatro contratos listos

`agents-md.md` — la plantilla de `AGENTS.md` y cómo comprobar que se lee

`config.toml` — la configuración del equipo, comentada

`agente-explorador.toml` — lectura barata, mapea y no toca nada

`agente-producto.toml` — alcance, exclusiones y métricas con umbral

`agente-seguridad-qa.toml` — amenazas, controles y puerta de salida

`agente-revisor.toml` — contradicciones y afirmaciones sin respaldo

`prompts-de-delegacion.md` — los cinco prompts de la ejecución

`protocolo-de-revision.md` — las diez pasadas y los umbrales

`runbook-si-algo-falla.md` — clasificar el fallo y los siete pasos

`glosario-agentes.md` — los trece términos, en una página

Totem.

MEGATOTEM.COM