GUÍA PRÁCTICA EN ESPAÑOL

Claude Opus 5

Cómo sacarle provecho al nuevo modelo de Anthropic

Qué trae este lanzamiento, cómo escribir mejores prompts para este modelo en particular y cómo aplicarlo a soporte, moderación, research legal y tu propio flujo de trabajo como freelancer o developer.

BASADA EN
Documentación oficial de Anthropic

LANZAMIENTO
Julio de 2026

NIVEL
Freelancers & developers

ÍNDICE

Qué vas a encontrar acá

Esta guía traduce y organiza la documentación pública de Anthropic sobre Claude Opus 5 en un recorrido pensado para gente que ya usa Claude a diario: qué cambia respecto a los modelos anteriores, cómo ajustar tus prompts para este modelo puntual, y cómo se aplica en casos de uso reales.

¿Qué es Claude Opus 5?

El 24 de julio de 2026, Anthropic presentó **Claude Opus 5**, la nueva generación del nivel Opus. La idea central detrás del lanzamiento es simple de entender: un modelo que se acerca a la inteligencia de punta de **Claude Fable 5** —el modelo más avanzado disponible públicamente— pero a la **mitad del precio**.

Anthropic lo posiciona como el modelo pensado para el uso diario: no es el más inteligente de todo el catálogo (ese lugar lo sigue ocupando Fable 5), pero es el que mejor equilibra inteligencia, velocidad y costo para el tipo de tareas que ocupan la mayor parte del trabajo real: programación, análisis, uso de herramientas y tareas de oficina de duración larga.

LANZAMIENTO
24 de julio de 2026 Disponible el mismo día en todas las plataformas de Claude

POSICIONAMIENTO
Rendimiento cercano a Fable 5 A mitad de precio

ROL POR DEFECTO
Modelo por defecto en Claude Max Y el más potente disponible en Claude Pro

STRING DEL MODELO
claude-opus-5 Para usar vía API

Para quién está pensado

Si laburás con código, herramientas de agentes, documentos de oficina o cualquier flujo donde antes tenías que elegir entre "el modelo caro que rinde mejor" y "el modelo barato que rinde peor", Opus 5 apunta directamente a borrar esa disyuntiva. Anthropic lo describe como un modelo **meditado y proactivo**: piensa antes de actuar, pero no se pasa de rosca verificando cosas que no hace falta verificar (más sobre esto en el capítulo 3).

> **DATO PARA TENER A MANO**
>
> Opus 5 mantiene el mismo precio que su predecesor, Opus 4.8: **US$5 por millón de tokens de entrada** y **US$25 por millón de tokens de salida**. La mejora de rendimiento llega sin que el costo por token suba un centavo.

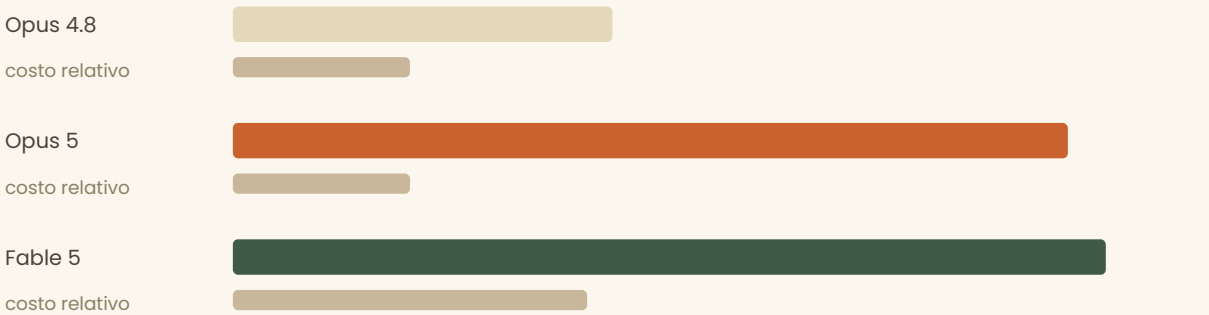
Qué NO es

Opus 5 no reemplaza a Fable 5 como el modelo más inteligente del catálogo, ni a Mythos 5 en tareas de investigación biológica o ciberseguridad de alto riesgo. Tampoco es un modelo entrenado específicamente en tareas ofensivas de ciberseguridad: su mejora en ese terreno es un efecto secundario de ser un modelo más capaz en general, no el resultado de entrenamiento dedicado.

Rendimiento y eficiencia

La gracia de Opus 5 no es solo "es más inteligente que Opus 4.8": es que logra ese salto **sin encarecerse**. En programación agéntica —tareas de varios archivos, refactors grandes, features de punta a punta— Anthropic reporta que Opus 5 más que duplica el desempeño de Opus 4.8 en Frontier-Bench v0.1, y a un costo por tarea más bajo. En CursorBench, con el nivel de esfuerzo máximo, queda a menos de medio punto porcentual del máximo de Fable 5, pero a la mitad del costo por tarea.

RENDIMIENTO PERCIBIDO (ILUSTRATIVO) — COSTO POR TAREA



Barra superior = capacidad percibida en tareas de código/razonamiento · Barra inferior = costo por token, a escala

Ilustración conceptual basada en las comparaciones publicadas por Anthropic (Frontier-Bench v0.1, CursorBench). No reproduce cifras exactas de benchmark.

Otros resultados que vale la pena conocer

- **ARC-AGI 3** (resolución de problemas novedosos): el puntaje de Opus 5 triplica al del siguiente mejor modelo.
- **Zapier AutomationBench** (tareas de negocio de punta a punta): tasa de éxito ~1,5× la del siguiente mejor modelo al mismo costo por tarea. Incluso en el nivel de esfuerzo más bajo, supera a cualquier otro modelo.
- **OSWorld 2.0** (uso de computadora): supera a todos los modelos a cualquier nivel de costo, y logra el mejor resultado de Fable 5 con poco más de un tercio de su costo.
- **Investigación científica**: mejora sobre Opus 4.8 en todas las evaluaciones de ciencias de la vida. El salto más marcado está en química orgánica (inferir estructuras moleculares a partir de espectroscopía) y en tareas de proteínas (predecir cómo afectan las variaciones de secuencia a su función).

> **PARA DEVS Y DISEÑADORES**
>
> Anthropic destaca una mejora notable en **salidas visuales**: Opus 5 arma mejores interfaces, animaciones y trabajo 3D que las generaciones anteriores de Opus. Si laburás con frontend, es un salto que vas a notar directamente en la calidad de los primeros intentos.

Cómo trabaja, y qué tan seguro es

Un modelo que se autoverifica

La característica de comportamiento más importante de Opus 5 —y la que más cambia cómo tenés que escribirle prompts, como vas a ver en el capítulo 6— es que **verifica su propio trabajo sin que se lo pidas**. En las pruebas de Anthropic, esto se tradujo en casos concretos: frente a un plano de una pieza mecánica y sin forma de "ver" el dibujo directamente, el modelo escribió su propio pipeline de visión por computadora para extraer la geometría de los píxeles y reconstruir la pieza en 3D. Ningún modelo competidor, con la misma consigna, lo logró en cinco intentos.

En otro caso, frente a un bug real en un gestor de paquetes open source, encontró la causa de fondo y corrigió un caso límite que el parche de la comunidad había pasado por alto —mientras que un modelo competidor sólo tapó el síntoma superficial y reportó el bug como resuelto.

Alineación

En las pruebas previas al lanzamiento, la auditoría de comportamiento automatizada de Anthropic ubicó a Opus 5 como su modelo más alineado hasta la fecha: sigue mejor la Constitución de Claude que Opus 4.8, Sonnet 5 o Fable 5, muestra las tasas más bajas de comportamiento engañoso, y es el menos propenso a dejarse manipular hacia usos indebidos.

Seguridad frente a riesgos de doble uso

Opus 5 no corre la frontera en capacidades riesgosas de doble uso. En evaluaciones conjuntas con socios del sector privado y gobiernos, se mantiene por detrás de Mythos 5 tanto en investigación biológica como en ciberseguridad ofensiva. Anthropic evitó deliberadamente entrenarlo en tareas cibernéticas —igual que con Opus 4.8—, aunque mejoró bastante en ese terreno como efecto de ser un modelo más capaz en general. Encuentra vulnerabilidades casi tan bien como Mythos 5, pero queda muy por detrás a la hora de convertir esas vulnerabilidades en exploits utilizables.

SALVAGUARDAS DE CIBERSEGURIDAD

Los clasificadores de Opus 5 son proporcionalmente menos restrictivos que los de Fable 5: permiten buscar vulnerabilidades en código fuente, pero bloquean el escaneo binario, las pruebas de penetración y la generación de exploits. Anthropic estima ~85% menos intervenciones de estos clasificadores que en Fable 5.

SALVAGUARDAS BIOLÓGICAS

Mantiene salvaguardas similares a Opus 4.8. Es el modelo de acceso general más capaz para investigación científica, aunque todavía muestra límites en tareas autónomas de investigación muy prolongadas (ahí Mythos 5 sigue siendo superior).

Precios y primeros pasos

Buena noticia para el bolsillo: Opus 5 **no cuesta más caro** que Opus 4.8. Los precios por token quedaron exactamente iguales, así que la mejora de rendimiento es, en los hechos, una suba de valor gratuita para cualquiera que ya estuviera pagando por Opus.

CONCEPTO	VALOR
Entrada (input)	**US$ 5 por millón de tokens**
Salida (output)	**US$ 25 por millón de tokens**
Modo rápido (Fast mode)	**2,5× la velocidad, al doble del precio base**
Ventana de contexto	**1.000.000 de tokens (por defecto y como máximo)**
String del modelo (API)	**claude-opus-5**

Retención de datos

Como en los Opus anteriores, Opus 5 no tiene requisitos de retención de datos para el acceso general.

Dos novedades beta que llegan junto con el modelo

CAMBIO DE HERRAMIENTAS A MITAD DE CONVERSACIÓN

Ahora podés cambiar qué herramientas tiene disponibles Claude en medio de una conversación sin invalidar el caché del prompt. Útil si tu agente pasa por distintas etapas con distintas necesidades de herramientas.

FALLBACKS AUTOMÁTICOS EN LA API

Podés activar que las solicitudes marcadas por los clasificadores de seguridad se redirijan automáticamente a otro modelo en vez de bloquearse. Con esto activado, la API siempre intenta responder con el mejor modelo disponible.

¿Cuándo conviene elegir Opus 5 y cuándo otro modelo?

- **Elegí Opus 5** para programación agéntica compleja, trabajo de oficina (planillas, presentaciones), tareas de visión (gráficos, diagramas, UI) y cualquier flujo de razonamiento largo y de varios pasos.
- **Elegí Haiku 4.5** cuando el cuello de botella es la latencia o el volumen: clasificación de tickets, moderación de contenido a gran escala, primeras pasadas de un pipeline.
- **Elegí Fable 5** cuando necesitás la inteligencia máxima disponible y el costo no es la variable que más te importa.

PARA ARRANCAR EN LA API

Alcanza con apuntar tus llamadas al string **claude-opus-5**. Si tu aplicación necesita identificarse a sí misma o declarar qué modelo está usando, podés indicárselo directamente en el system prompt: "El asistente es Claude, creado por Anthropic. El modelo actual es Claude Opus 5."

Fundamentos de prompt engineering

Estas técnicas aplican a cualquier modelo actual de Claude, Opus 5 incluido. Si ya las conocés, el capítulo 6 es el que trae lo específico de este modelo.

Sé claro y directo

Pensá a Claude como a una persona nueva en el equipo, brillante pero sin contexto sobre tus normas de trabajo: cuanto más precisamente expliques lo que querés, mejor el resultado. La regla de oro: mostrale tu prompt a un colega con poco contexto sobre la tarea y pedile que lo siga. Si esa persona se confundiría, Claude también.

Dale contexto, no solo instrucciones

Explicar el "por qué" detrás de una instrucción ayuda a que el modelo generalice mejor tu objetivo, en vez de seguir la letra de la instrucción de forma rígida.

Usá ejemplos (few-shot)

Los ejemplos son una de las formas más confiables de guiar formato, tono y estructura. Para que funcionen bien, tienen que ser **relevantes** (parecidos a tu caso real), **diversos** (que cubran casos límite sin que el modelo copie un patrón no deseado) y estar **estructurados** dentro de etiquetas `<example>`. Entre 3 y 5 ejemplos suele ser el punto óptimo.

Estructurá con etiquetas XML

Cuando tu prompt mezcla instrucciones, contexto, ejemplos y datos variables, envolver cada tipo de contenido en su propia etiqueta (`<instructions>`, `<context>`, `<document>`) reduce la ambigüedad y le hace más fácil al modelo separar cada parte.

Dale un rol

Definir un rol en el system prompt enfoca el tono y el comportamiento del modelo para tu caso de uso. Alcanza con una frase.

```
system = "Sos un asistente de programación especializado en Python."
```

Documentos largos: el orden importa

Cuando trabajás con documentos extensos (más de 20.000 tokens), poné el material largo **arriba** de tu instrucción y tu consulta, no al final. En las pruebas de Anthropic esto mejoró la calidad de la respuesta hasta un 30% en casos con varios documentos. Para tareas sobre documentos largos, pedile al modelo que primero extraiga citas textuales relevantes antes de responder: eso ancla la respuesta en el texto real y reduce el margen de error.

Fundamentos de prompt engineering – parte 2

Formato de salida

Funciona mejor decirle al modelo **qué hacer** en vez de qué no hacer. Cambiá "no uses markdown" por "tu respuesta debe estar compuesta por párrafos de prosa fluida". También ayuda usar etiquetas XML como indicadores de formato, y hacer que el estilo de tu prompt se parezca al estilo de salida que buscás: si sacás el markdown de tu prompt, el modelo tiende a devolver menos markdown también.

LaTeX en matemática

Por defecto, los modelos actuales usan LaTeX para expresiones matemáticas. Si preferís texto plano, hay que pedirlo explícitamente en el prompt.

Uso de herramientas (tools)

Si le decís "podés sugerir cambios", a veces el modelo va a sugerir en vez de implementar, aunque tu intención real fuera que actuara. Para que actúe por defecto, sumá una instrucción explícita de proactividad; para lo contrario (que no toque nada sin que se lo pidas), sumá la instrucción inversa. Los modelos actuales también corren llamadas a herramientas independientes en paralelo por su cuenta — podés reforzar o frenar esa conducta pidiéndolo explícitamente.

Trabajo de largo aliento y estado

Para tareas que se extienden por varias ventanas de contexto (proyectos grandes de código, por ejemplo), conviene: escribir tests en un formato estructurado antes de empezar a trabajar, crear scripts de arranque para no repetir trabajo entre sesiones, usar git como bitácora de estado, y guardar notas de avance en texto libre. Cuando el contexto se reinicia, el modelo es muy bueno reconstruyendo dónde quedó a partir del sistema de archivos.

Autonomía con criterio

Sin instrucciones de por medio, el modelo puede tomar acciones difíciles de revertir: borrar archivos, hacer force-push, publicar en sistemas compartidos. Si querés que confirme antes de acciones riesgosas, pedíselo explícitamente y dale ejemplos concretos de qué considerás "riesgoso" (operaciones destructivas, difíciles de revertir, o visibles para terceros).

> **TIP PARA FRONTEND / DISEÑO**
>
> Sin dirección, los modelos tienden a converger en patrones genéricos —la estética "AI slop": tipografías de manual (Inter, Arial), gradientes violetas sobre blanco, layouts predecibles. Si pedís explícitamente tipografías distintivas, una paleta de color con un dominante fuerte y acentos filosos, animaciones de entrada bien orquestadas y fondos con atmósfera (en vez de color sólido liso), la diferencia en el resultado es notable.

Evitá el sobre-diseño

Pedile explícitamente que no agregue abstracciones, configurabilidad o manejo de errores para escenarios que no van a pasar. Un fix de bug no necesita una limpieza del código de alrededor; una feature simple no necesita flexibilidad extra "por las dudas".

Cómo promptear específicamente a Opus 5

Este es el capítulo con más impacto práctico si ya venís usando Claude en producción: son los ajustes puntuales que cambian frente a Opus 4.8, según la guía oficial de prompting de Anthropic para este modelo.

Por defecto, responde más largo

Las respuestas de cara al usuario de Opus 5 son, por defecto, más extensas que las de los Opus anteriores. Ojo con un detalle importante: el parámetro de **esfuerzo** (effort) controla cuánto "piensa" el modelo puertas adentro, no cuánto escribe puertas afuera. Bajar el esfuerzo no acorta la respuesta de forma confiable. Si querés respuestas más breves, hay que pedirlo explícitamente.

```
Mantené las respuestas enfocadas, breves y concisas. Los descargos y
aclaraciones, cortos. Dedicale la mayor parte de la respuesta a la
respuesta principal. Si te piden explicar algo, dá un resumen de alto
nivel salvo que se pida explícitamente profundizar.
```

Narra lo que va haciendo

En sesiones agénticas, Opus 5 tiende a anunciar lo que está por hacer antes de cada llamada a herramientas, y sus mensajes intermedios suelen ser más largos que los de modelos anteriores. Si esto no encaja con tu producto, describile explícitamente el ritmo y la forma que querés para esas actualizaciones.

```
Antes de tu primera llamada a una herramienta, decí en una oración qué
vas a hacer. Mientras trabajás, dá una actualización breve solo cuando
encuentres algo importante o cambies de rumbo. Al terminar, empezá por
el resultado: la primera oración debe responder "qué pasó" o "qué
encontraste", con el detalle de apoyo después.
```

Si en cambio querés *más* narración o un estilo distinto, la misma palanca funciona al revés: describí con ejemplos positivos cómo querés que se vean esas actualizaciones. Los ejemplos de lo que sí querés funcionan mejor que las instrucciones de lo que no.

Los documentos que escribe también son más largos

Separado de la verborragia conversacional: los archivos que Opus 5 escribe a disco (informes, documentos, resúmenes) también tienden a ser más extensos por defecto. Si tu producto incluye documentos redactados por Claude, sumá una calibración explícita de extensión: que cubra la sustancia, sin rellenar con secciones redundantes o texto de relleno.

Cómo promptear a Opus 5 — parte 2

Ya se autoverifica: sacá las instrucciones de verificación

Este es, probablemente, el cambio de hábito más importante al migrar prompts viejos. Si tu prompt tiene instrucciones del tipo "incluí un paso final de verificación para cualquier tarea no trivial" o "usá un subagente para verificar", **sacalas**: en Opus 5 generan sobre-verificación, gastan tokens de más y no mejoran la calidad, porque el modelo ya hace ese chequeo por su cuenta. Lo mismo vale para instrucciones de "doble chequeo" o "reverificá antes de responder".

Puede ampliar el alcance de la tarea por su cuenta

Opus 5 tiende a agregar pasos que no pediste, o a aplicar su propio criterio sobre qué debería incluir la tarea. Para tareas acotadas, hay que ponerle límites explícitos.

```
Entregá lo que se pidió, en el alcance que se pidió. Tomá vos las
decisiones de rutina, y consultá solo cuando interpretaciones distintas
del pedido lleven a un trabajo materialmente distinto. Si el pedido
parece equivocado o hay un mejor enfoque, decilo en una oración y seguí
con la tarea tal como fue pedida. Terminá la tarea completa, y no
avances hacia acciones que claramente exceden lo pedido.
```

Delega en subagentes con más facilidad

Delegar tiene sentido en tramos de trabajo grandes, genuinamente independientes y paralelizables — pero multiplica costo y tiempo si se aplica a tareas chicas. Si tu implementación soporta subagentes, dale una guía explícita de cuándo delegar o ponele un techo determinístico a la cantidad de agentes.

```
Delegá en un subagente solo para tareas grandes y genuinamente
independientes, como una investigación amplia en varios archivos. No
delegues trabajo que podés terminar vos con pocas llamadas a
herramientas, y no uses subagentes para verificar tu propio trabajo. Si
un subagente alcanza, usá uno solo en vez de varios.
```

Se autocorrige, y a veces lo anuncia de más

Opus 5 detecta y corrige sus propios errores sin que se lo pidas —así que instrucciones como "revisá dos veces tu respuesta" son redundantes y suman costo sin mejorar el resultado. Además tiende a narrar más sus propias correcciones, lo cual puede no encajar en un producto de cara al usuario.

```
Corregí una afirmación anterior solo cuando el error cambiaría el
código, las conclusiones o las decisiones del usuario. Indicá las
correcciones de forma simple y breve, y seguí con la tarea. Para
errores menores que no cambian nada para el usuario, corregí y seguí
sin marcarlo.
```

Cómo promptear a Opus 5 — parte 3

El "pensamiento" (thinking) viene activado por defecto

A diferencia de Opus 4.8, en Opus 5 el pensamiento extendido viene **activado por defecto**, y solo se puede desactivar en niveles de esfuerzo *high* o menores. Con el pensamiento desactivado pueden aparecer dos artefactos ocasionales en la salida visible: llamadas a herramientas que terminan escritas como texto plano en vez de ejecutarse, y etiquetas XML internas (como `‹thinking›`) filtrándose en la respuesta.

La mitigación más simple para ambos casos: **dejá el pensamiento activado** y controlá el costo bajando el nivel de esfuerzo, en vez de desactivarlo del todo. Para la mayoría de las tareas, pensamiento activado con esfuerzo bajo rinde mejor que pensamiento desactivado a un costo similar.

> **SI POR ALGÚN MOTIVO TENÉS QUE MANTENERLO DESACTIVADO**
>
> Una sola instrucción combinada mitiga los dos artefactos: dale permiso explícito para decir una oración breve antes de llamar a una herramienta, una alternativa para cuando ninguna herramienta encaja (que lo diga en vez de inventar), y una regla general contra etiquetas internas en la respuesta. Evitá nombrar "las etiquetas de pensamiento" en la instrucción — la forma general funciona mejor que señalar el artefacto puntual.

Resumen: qué mejoró frente a Opus 4.8, capacidad por capacidad

Programación agéntica
Más fuerte en tareas difíciles de varios archivos y refactors grandes; termina tareas completas en vez de dejar código a medio hacer.

Revisión de código
Alta precisión y exhaustividad encontrando bugs reales; mantiene precisión incluso con esfuerzo bajo.

Eficiencia
Esfuerzo bajo/medio rinde bien con una fracción de los tokens y la latencia de niveles altos.

Visión
Fuerte en gráficos, documentos, diagramas y réplica visual de interfaces.

Contexto largo
Ventana de 1M de tokens con seguimiento de instrucciones consistente en toda su extensión.

Documentos de oficina
Planillas complejas con fórmulas no triviales y presentaciones bien estructuradas.

Coordinación multiagente
Coordina equipos de subagentes con patrones de escritor-verificador, con pocos casos de agentes pisándose el trabajo.

Revisión literal de instrucciones
Si le pedís "solo reportá lo de alta severidad", lo va a tomar literal. Pedile que reporte todo y filtrá después vos.

Casos de uso prácticos

Cuatro guías de producción de Anthropic, resumidas y con la recomendación de modelo actualizada según dónde encaja Opus 5.

Enrutamiento de tickets de soporte

MODELO RECOMENDADO: HAIKU 4.5

Clasificar tickets por intención, urgencia o especialidad tiene sentido para Claude cuando tenés poca data etiquetada, las categorías cambian con el tiempo, o necesitás razonamiento explicable detrás de cada decisión. El patrón típico: un prompt que devuelve razonamiento e intención en etiquetas separadas (así podés extraerlas por separado), con 3 a 5 ejemplos de cada categoría. Con más de 20 categorías, conviene pasar a una jerarquía de clasificadores en cascada en vez de un prompt único gigante. Meta de referencia: ~95% de precisión y una reducción de costo relevante frente al método de enrutamiento anterior.

Agente de atención al cliente

MODELO RECOMENDADO: OPUS 5 (O HAIKU 4.5 SI PRIORIZA LATENCIA)

Para consultas complejas que requieren razonamiento profundo a lo largo de conversaciones largas de varios pasos, Anthropic recomienda directamente Opus 5. Si tu flujo tiene muchos prompts encadenados con RAG o uso intensivo de herramientas y la latencia es la variable crítica, Haiku 4.5 puede rendir mejor en la práctica. La receta de siempre sigue aplicando: definí la interacción ideal, desglosala en tareas concretas (saludo, información de producto, gestión de la conversación, generación de cotizaciones vía tool use), y sumá ejemplos de interacciones "buenas" al prompt.

Casos de uso prácticos – parte 2

Moderación de contenido

MODELO RECOMENDADO: HAIKU 4.5 (POR COSTO A ESCALA)

Un dato de costo real que vale la pena tener presente: para un millón de posteos moderados al mes, con un 3% marcado como problemático, Anthropic estima el costo mensual en **~US$36.100 con Haiku 4.5** contra **~US$180.500 con Opus 5 u Opus 4.8** (ambos al mismo precio por token). Para volumen alto, Haiku es casi siempre la elección correcta; Opus entra en juego cuando el matiz semántico del contenido lo justifica. Importante: Claude modera contenido particularmente peligroso según la Política de Uso Aceptable de Anthropic sin importar lo que diga tu prompt — no hay forma de "desactivar" esa capa vía instrucciones.

Definir las categorías con precisión importa mucho: agregar una definición explícita de "asesoramiento especializado" (financiero, médico, legal) hizo que un comentario como "es un buen momento para invertir en oro" pasara de "sin problema" a marcado, sin tocar el resto del prompt.

Resúmenes legales

MODELO RECOMENDADO: OPUS 5 (POR PRECISIÓN)

La guía oficial de Anthropic para este caso todavía recomienda Opus 4.8 por su precisión —fue escrita antes del lanzamiento de Opus 5—, pero como Opus 5 llega casi al nivel de Fable 5 al mismo precio que Opus 4.8, es el candidato natural para reemplazarlo en este flujo. Para volúmenes muy grandes de documentos, Haiku 4.5 sigue siendo la opción más económica. El patrón de prompt: pedí que la respuesta llegue en secciones XML por cada dato a extraer (partes involucradas, plazos, montos, obligaciones), con "No especificado" como valor por defecto cuando el documento no lo menciona. Para documentos que superan la ventana de contexto, la técnica de meta-resumen (resumir por partes y después combinar los resúmenes) sigue siendo la solución recomendada.

ADVERTENCIA LEGAL, SIEMPRE

Si automatizás resúmenes legales, sumá siempre un descargo dejando claro que el resumen fue generado por IA y debe ser revisado por una persona con formación legal antes de tomar decisiones.

Cómo saber si está funcionando

Antes de optimizar un prompt hace falta saber contra qué lo estás midiendo. Un buen criterio de éxito es específico ("clasificación de sentimiento precisa", no "buen rendimiento"), medible con una métrica cuantitativa o una escala cualitativa aplicada con consistencia, alcanzable según lo que hoy pueden hacer los modelos de punta, y relevante para lo que tu aplicación realmente necesita.

Criterios habituales

Fidelidad de la tarea · Consistencia · Relevancia y coherencia · Tono y estilo · Preservación de privacidad · Uso del contexto · Latencia · Precio

Cómo calificar tus evaluaciones

Elegí el método más rápido y confiable que tu caso permita:

- **Calificación por código** (exact match, string match): la más rápida, confiable y escalable, pero con poco margen para matices.
- **Calificación humana**: la más flexible y de mayor calidad, pero lenta y cara. Evitarla si hay alternativa.
- **Calificación por LLM**: rápida, flexible y escalable para juicios complejos. Probá primero que sea confiable antes de escalarla.

> **SI USÁS UN LLM COMO EVALUADOR**
>
> Dale una rúbrica clara y detallada ("la respuesta siempre debe mencionar 'Acme Inc.' en la primera oración; si no lo hace, es automáticamente incorrecta"), pedile un resultado empírico y específico (correcto/incorrecto, o una escala del 1 al 5), y hacé que razone antes de decidir la calificación — eso mejora el desempeño del evaluador, sobre todo en juicios complejos.

Principio general para diseñar los casos de prueba: priorizá el volumen por sobre la perfección. Muchos casos de prueba con calificación automática algo menos precisa suelen ganarle a pocos casos calificados a mano con altísima precisión — y no te olvides de sumar casos límite, no solo los casos típicos.

Blindar tu aplicación

Cuatro frentes de trabajo para que tu integración con Claude sea confiable en producción, no solo en la demo.

Reducir alucinaciones

- Dale permiso explícito para decir "no sé" — por sí solo reduce bastante la información inventada.
- En documentos largos (+20.000 tokens), pedile que extraiga citas textuales antes de responder: eso ancla la respuesta en el texto real.
- Pedile que cite la fuente de cada afirmación, y que retracte cualquier afirmación para la que no encuentre una cita de respaldo.
- Técnicas más avanzadas: verificación paso a paso (chain-of-thought), correr el mismo prompt varias veces y comparar resultados (best-of-N), y restringirlo explícitamente a usar solo la información de los documentos provistos, no su conocimiento general.

Ninguna de estas técnicas elimina las alucinaciones del todo. Para decisiones de alto impacto, siempre validá la información crítica por otra vía.

Aumentar la consistencia

- Especificá el formato de salida exacto (JSON, XML, una plantilla propia). Si necesitás conformidad garantizada con un schema, usá la función de Structured Outputs en vez de trucos de prompting.
- Restringí con ejemplos: mostrar el formato deseado funciona mejor que describirlo de forma abstracta.
- Para chatbots o bases de conocimiento, apoyate en retrieval para anclar las respuestas en un set de información fijo.
- Para mantener un personaje o rol consistente en el tiempo: definilo con detalle en el system prompt (personalidad, historia, particularidades) y anticipale escenarios comunes con la respuesta esperada.

Blindar tu aplicación – parte 2

Mitigar jailbreaks e inyecciones de prompt

Son dos amenazas distintas, con defensas distintas:

LA PERSONA USUARIA ES LA AMENAZA

Filtros livianos con un modelo rápido (Haiku 4.5) que preclasifican el input antes de que llegue a la conversación principal; validación de patrones de inyección conocidos; system prompts que marcan límites éticos y legales con claridad; y una política definida para usuarios reincidentes.

EL CONTENIDO DE TERCEROS ES LA AMENAZA

Contenido no confiable (mails, páginas web, resultados de herramientas) siempre dentro de bloques `tool_result`, nunca en el system prompt. Explicitá qué es ese contenido y de dónde viene. Codificalo en JSON para que un atacante no pueda "escapar" de las comillas. Nunca pongas tus propias instrucciones dentro de un tool_result.

Para flujos sensibles, aplicá el principio de mínimo privilegio: que el modelo no tenga acceso a datos o acciones que no necesita, corré herramientas en entornos aislados, y hacé "red-teaming" de tu propio agente antes de lanzarlo a producción — probalo deliberadamente con documentos y resultados de herramientas que contengan intentos de inyección.

Reducir la fuga de prompts

Antes de complicar tu prompt con técnicas anti-fuga, probá primero con monitoreo: filtrado de la salida y revisión periódica de outputs suele alcanzar. Si necesitás ir más allá: separá el contexto sensible de la consulta del usuario, filtrá la salida en busca de palabras clave que indiquen una fuga, y no incluyas detalle propietario que el modelo no necesita para la tarea — cuanto más contenido de sobra, más se distrae de la instrucción de "no filtrar".

EL EQUILIBRIO ES LA CLAVE

Una protección anti-fuga demasiado compleja puede degradar el resto de la tarea. El objetivo no es solo evitar la fuga, sino mantener el rendimiento del modelo. Priorizá la técnica más simple que resuelva tu caso.

Glosario de términos clave

Ventana de contexto
Cuánto texto puede "tener presente" el modelo al generar una respuesta. Es la memoria de trabajo, distinta de los datos con los que fue entrenado.

Latencia
El tiempo que pasa entre que enviás un prompt y recibís la respuesta generada.

MCP (Model Context Protocol)
Protocolo abierto que estandariza cómo las aplicaciones le dan contexto a un modelo — una especie de puerto USB-C para conectar IA con fuentes de datos y herramientas externas.

Temperatura
Parámetro que controla la aleatoriedad de las respuestas. Valores más altos generan salidas más creativas y variadas; valores más bajos, más conservadoras y predecibles.

RLHF
Aprendizaje por refuerzo con feedback humano — la técnica usada para entrenar a Claude a comportarse de forma más alineada con las preferencias humanas.

Token
La unidad mínima de texto que procesa el modelo. En Claude, un token equivale aproximadamente a 3,5 caracteres en inglés (varía según el idioma).

RAG (Retrieval Augmented Generation)
Técnica que combina la búsqueda de información externa con la generación de texto, para anclar las respuestas en una base de conocimiento en vez de depender solo de lo que el modelo memorizó.

Fine-tuning
Entrenar más un modelo ya entrenado con datos adicionales, para que adopte los patrones de ese dataset específico.

HHH
Los tres principios que guían a Claude: útil (helpful), honesto (honest) e inofensivo (harmless).

TTFT (Time to First Token)
Cuánto tarda el modelo en generar el primer token de la respuesta. Clave en aplicaciones interactivas donde el usuario espera una reacción rápida.

PARA SEGUIR PROFUNDIZANDO

Fuentes y recursos oficiales

Esta guía resume y traduce documentación pública de Anthropic. Para el detalle completo, con ejemplos de código y las guías sin recortar, estos son los enlaces originales en inglés:

Anuncio oficial — Introducing Claude Opus 5
anthropic.com/news/claude-opus-5

Guía de prompting para Claude Opus 5
platform.claude.com/docs/en/build-with-claude/prompt-engineering/prompting-claude-opus-5

Mejores prácticas de prompting (todos los modelos)
platform.claude.com/docs/en/build-with-claude/prompt-engineering/claude-prompting-best-practices

Guías de casos de uso (tickets, soporte, moderación, legal)
platform.claude.com/docs/en/about-claude/use-case-guides/overview

Cómo definir criterios de éxito y evaluaciones
platform.claude.com/docs/en/test-and-evaluate/develop-tests

Guías para blindar tu aplicación (guardrails)
platform.claude.com/docs/en/test-and-evaluate/strengthen-guardrails/reduce-hallucinations

Glosario completo
platform.claude.com/docs/en/about-claude/glossary