Guía de desarrollador · functional-scope

functional-scope: guía técnica

Cómo analiza el agente las fuentes de un proyecto, qué flujo sigue, cómo construye el documento Word y qué controles aplica antes de entregarlo.

3 nov 2026 | Guía de desarrollador | Claude Code | 18 min de lectura

Descripción general

El **Agente de Alcances Funcionales** (`functional-scope`) actúa como Product Manager senior, analista de negocio y consultor de soluciones de software. Lee las fuentes de un proyecto (propuestas, actas, cotizaciones, notas), extrae hechos y decisiones con su fuente, detecta contradicciones y vacíos, y redacta el **documento de alcance funcional** dirigido al cliente, **sin inventar compromisos**.

Su prioridad es un alcance comprensible para el cliente, verificable para el equipo técnico y delimitado para la gestión comercial. No reemplaza la aprobación del líder técnico, comercial, jurídico ni del cliente: la pide donde aplica.

Esta guía explica cómo está construido: las reglas del manual que lo gobiernan, el flujo de ocho etapas, el sistema de tres estados, la estructura del documento, el constructor del Word, el control de calidad y cómo mantenerlo. Está pensada para quien mantiene el agente.

3	8	16	14
estados siempre visibles: CONFIRMADO, PROPUESTO y PENDIENTE	etapas del flujo, de la recepción a la entrega	secciones de contenido en el documento al cliente	reglas obligatorias

LO ESENCIAL

Un archivo compartido **es todo lo que hay**: el agente no entrevista al usuario ni pregunta lo que el archivo no trae. Lo que falta queda marcado `[PENDIENTE DE VALIDACIÓN]` y asignado a un responsable, y se redacta el borrador **en el mismo turno** del diagnóstico.

Qué hace y qué no hace

El agente hace	El agente no hace
Analiza todas las fuentes y construye una matriz de requerimientos	Usar memoria ni «lo típico en proyectos así»
Redacta el borrador de inmediato, con los tres estados visibles	Presentar una opción sugerida como decisión tomada
Muestra ambas versiones cuando dos fuentes se contradicen	Elegir en silencio la versión más reciente
Entrega el documento editable, un resumen de cambios y el registro de pendientes	Declarar «listo para firma» con pendientes bloqueantes abiertos
Pide revisión técnica, comercial y jurídica donde corresponde	Redactar precio, impuestos, garantía, soporte o SLA «por defecto»
Distingue desarrollos específicos, componentes preexistentes y servicios de terceros	Convertir ideas de fases futuras en entregables actuales

Cómo corre: Claude Code y `CLAUDE.md`

El agente corre en **Claude Code** y se gobierna con `CLAUDE.md`. Su fuente normativa es el **Manual maestro para el agente generador de alcances funcionales** (`recursos/Manual_Agente_Alcances_Funcionales_Campuslands.docx`, versión 1.0, octubre de 2026), que `wiki/manual-operativo.md` resume para consulta rápida. Si algo de `CLAUDE.md` contradice el manual, el agente lo señala al usuario antes de actuar.

Hay un ajuste importante del usuario, del **2026-10-29**, que está por encima del manual (§2 etapa 4 y §10): si se comparte un archivo, es todo lo que hay, por lo que **no hay entrevistas por bloques ni preguntas sobre lo que el archivo no trae**. La wiki recoge además otras decisiones de interpretación del usuario (estructura del documento, portada sin versión ni clasificación, ausencia de la palabra «borrador» dentro del documento).

La instrucción maestra

Al recibir archivos o notas de un proyecto, el agente los analiza íntegramente; extrae hechos y decisiones con su fuente; identifica contradicciones y faltantes; construye la matriz de requerimientos y redacta el borrador de inmediato. Elabora un documento profesional con la

estructura del manual (fichas funcionales, matrices de integraciones, entregables, criterios de aceptación, exclusiones y responsabilidades). Si faltan decisiones críticas, entrega **solo borrador** y la lista de validaciones.

Estructura del repositorio

```
functional-scope/
  CLAUDE.md                       # reglas, flujo y diseño del documento
  recursos/                       # solo lectura
    Manual_Agente_Alcances_Funcionales_Campuslands.docx
    plantilla/                    # plantilla Word corporativa (membrete) y portada de referencia
    marca/                        # firmas escaneadas (representante legal y Gabriela Pedraza)
  plantillas/                     # tablas del manual y matrices de trabajo
    ficha-funcional.md · matriz-trazabilidad.md · matriz-riesgos-pendientes.md
    matriz-informacion.md · registro-decisiones.md · control-calidad.md
    Alcance Funcional - Plantilla.docx · Alcance Funcional - Muestra de formato.docx
  scripts/
    alcance_docx.py               # constructor del .docx (clase Alcance)
    previsualizar.py              # docx a PDF y a PNG por página
  alcances/
    <cliente-slug>/
      fuentes/                    # insumos del usuario; no se editan
      trabajo/                    # 01 inventario · 02 matriz · 03 pendientes · 04 decisiones ·
05 calidad
      entrega/                    # documento al cliente, una versión por archivo
  wiki/
    index.md · log.md             # catálogo de proyectos y bitácora
    manual-operativo.md           # resumen del manual
    ejemplos/                     # diagnósticos de muestra: solo método
```

DÓNDE VIVE EL AGENTE Y SUS RECURSOS COMPARTIDOS

El agente está en `agents/claude-agents/functional-scope/`. Los logos, el isotipo, el favicon, el `Brief Fullservice.pdf` y el Brandbook de Campuslands ya no se copian dentro de cada agente: están en `resources/campuslands-resources/`, en la raíz del repositorio (desde la carpeta del agente: `../../../resources/campuslands-resources/`). En `recursos/marca/` quedan solo las firmas escaneadas.

- **Un proyecto, una carpeta.** El slug va en minúsculas, sin espacios ni tildes, con guiones (`casa-blanca`). Si un cliente tiene varios proyectos: `<cliente>-<proyecto>`.
- **Las fuentes no se editan.** Son los insumos del usuario.

- **Los ejemplos son solo método.** Lo que hay en `wiki/ejemplos/` ilustra cómo se diagnostica, y nunca se copia a otro proyecto: no se trasladan módulos, integraciones, cifras, plazos ni condiciones.
- **El repositorio es privado** por decisión del usuario (2026-10-29), así que `alcances/<cliente>/` se puede versionar y subir sin exclusiones.

El flujo de ocho etapas



Cada etapa deja un archivo en `alcances/<slug>/trabajo/` o en `entrega/`:

Etapa	Qué hace el agente	Salida
1 · Recepción	Identifica proyecto, cliente, tipo de documento, archivos y la **versión vigente** de cada fuente	`01-inventario-fuentes.md`
2 · Lectura	Extrae requisitos, decisiones, cifras, exclusiones y contradicciones de **todas** las fuentes	`02-matriz-informacion.md`

Etapa	Qué hace el agente	Salida
3 · Diagnóstico	Clasifica los vacíos por criticidad y por su efecto en alcance, plazo y costo	`03-pendientes.md`
4 · Cierre de vacíos	Sin entrevista: lo que la fuente no dice queda `[PENDIENTE DE VALIDACIÓN]` y se asigna a un responsable; solo se registra lo que el usuario diga por iniciativa propia	`04-registro-decisiones.md`
5 · Resumen	Deja el alcance y los supuestos a la vista, **no como pregunta previa**	Resumen en el chat
6 · Redacción	Escribe el documento y los anexos justo después del diagnóstico	Borrador en `entrega/`
7 · Control de calidad	Aplica el checklist del manual §8 con evidencia	`05-control-calidad.md`
8 · Entrega	Entrega el documento editable y el resumen de pendientes; actualiza índice y bitácora	`entrega/` y `wiki/log.md`

La primera respuesta

Tras el diagnóstico, el agente responde con un formato fijo (que reemplaza al del manual §10): identifica el proyecto, indica cuántos módulos y cuántos pendientes bloqueantes encontró, nombra el borrador y la fuente en que se basa, resume tres a cinco hallazgos y apunta a la lista de validaciones por responsable.

Regla de preguntas

El agente **solo pregunta si, sin la respuesta, no puede ni empezar** (por ejemplo, dos archivos que se contradicen sin que se sepa cuál es la fuente principal, o no se entiende qué documento se quiere). Entonces formula **una** pregunta, con la opción que asumirá por defecto. Un «no sé» del usuario nunca detiene la redacción, y lo que el usuario diga por iniciativa propia se registra en `04-registro-decisiones.md` y actualiza la matriz.

Los tres estados y la política de faltantes

Todo dato del documento lleva uno de tres estados, siempre visibles:

Estado	Significado	Cómo se muestra
CONFIRMADO	Hay una fuente o una aprobación citable	Resaltado en verde
PROPUESTO	Es una recomendación del agente o del equipo, aún no aceptada	Resaltado en azul
PENDIENTE	Falta definirlo	En ámbar; y `[PENDIENTE DE VALIDACIÓN]` con fondo amarillo suave

Los faltantes se clasifican con una política de tres niveles:

Criticidad	Qué es	Consecuencia
BLOQUEANTE	Impide determinar alcance, viabilidad, costo, plazo, aceptación, privacidad o responsabilidades	Sin resolverlo, el documento es **solo borrador**
IMPORTANTE	Afecta la operación o la experiencia, pero no impide un borrador	Se lista y se asigna
EDITORIAL	Es asunto de presentación	Se corrige al revisar

UNA EXCEPCIÓN ACORDADA

Lo que la propia fuente dice que se definirá en una etapa posterior (por ejemplo, los criterios de aceptación medibles que una propuesta fija en el levantamiento de su Fase 1) se escribe como tal, «se definen en el levantamiento…», **sin** la marca de pendiente.

Las reglas duras

Regla	Qué significa
Solo lo aportado	Documentos, reuniones, cotizaciones y respuestas efectivamente entregados

Regla	Qué significa
Tres estados	Nunca presentar una opción sugerida como decisión tomada
Lo futuro no se contrata	Hipótesis, ejemplos y fases futuras van solo en «Fases futuras»
No inventar	Cifras, fechas, niveles de servicio, métricas de precisión, integraciones, propiedad intelectual ni responsabilidades
Contradicciones	Mostrar **ambas versiones** con documento, sección y fecha, en el documento y en la lista de pendientes; nunca elegir en silencio ni detener el trabajo
Redacción medible	«Permitirá consultar» no equivale a «garantizará ahorro»; toda funcionalidad tiene un **límite verificable** (no basta «gestionar», «analizar» o «automatizar»)
IA con límites	Sin exactitud perfecta ni entrenamiento de modelos ni recomendaciones infalibles; se declara la tarea, las fuentes autorizadas, la supervisión humana, los límites y la trazabilidad
Integraciones	No se presupone acceso ni capacidades de una API sin evidencia
Propiedad intelectual	Se distinguen desarrollos específicos, componentes preexistentes y servicios de terceros, y se solicita revisión jurídica
Términos comerciales	Precio, impuestos, hitos de pago, garantía, soporte y SLA solo si la fuente los dice; si no, `[PENDIENTE DE VALIDACIÓN]`
Versión para firma	Se bloquea mientras haya pendientes críticos; antes se verifica que alcance, costo, plazo, entregables y condiciones estén alineados con la propuesta aprobada
Los ejemplos son ejemplos	Nunca se copian módulos, cifras ni condiciones de `wiki/ejemplos/` a otro proyecto
Confidencialidad	Cada proyecto es de un cliente; no se mezclan datos entre documentos

El documento al cliente

El documento va **siempre en este orden**: portada, carta de presentación, tabla de contenido, contenido y apartado de firma. Cada una de las tres primeras ocupa **una sola hoja**.

Parte	Regla
Portada	Rótulo, título, cliente y dependencia, más dos tarjetas: «Preparado por» y «Fecha». **Sin** «Preparado para», estado, versión ni clasificación; nota legal al pie
Carta de presentación	Una hoja repartida por toda la página (cuerpo de 11 pt, interlineado 1,25): lugar y fecha, destinatario, asunto, dos o tres párrafos, la lista de lo que contiene el documento y la firma de quien presenta. Si la fuente trae una carta, se adapta, no se copia
Tabla de contenido	Una hoja, **ya generada**: nunca un campo vacío que haya que actualizar
Contenido	Sin límite de hojas; cada título 1 inicia sección
Aprobación y firmas	Última hoja: columna izquierda fija (representante legal de Campuslands) y columna derecha según el proyecto

LO QUE NUNCA APARECE DENTRO DEL DOCUMENTO

El número de versión ni la clasificación (viven en el nombre del archivo y en la wiki). La palabra «borrador» ni «no apto para firma» (el estado vive en el chat, `wiki/index.md` y `05-control-calidad.md`). Si la fuente llama «borrador» a un entregable, se escribe «propuesta». Ya no existe la página de control documental.

Las dieciséis secciones de contenido

#	Sección	Contenido
1	Resumen ejecutivo	Problema, solución, beneficios y alcance en lenguaje de negocio, sin garantías no aprobadas
2	Contexto y objetivos	Situación actual, objetivos, indicadores de éxito, restricciones y dependencias

#	Sección	Contenido
3	Alcance y límites	Módulos incluidos, fuera de alcance, fases futuras y mapa de procesos
4	Usuarios y permisos	Actores, perfiles, permisos y niveles de visibilidad
5	Requerimientos funcionales	**Ficha uniforme** por funcionalidad
6	Fuentes de datos e integraciones	Matriz por sistema: propietario, acceso, interfaz, datos, volumen, frecuencia, autenticación y errores
7	IA y automatización	Casos de uso, servicios confirmados, fuentes autorizadas, validación y límites
8	Requerimientos no funcionales	Seguridad, privacidad, auditoría, rendimiento, respaldo y ambientes, sin cifras no acordadas
9	Arquitectura de alto nivel	Componentes, flujo de información y servicios externos, con diagrama comprensible
10	Entregables y aceptación	Matriz por fase: entregable, evidencia, responsable de validación y criterios objetivos
11	Metodología y cronograma	Fases, hitos, supuestos, dependencias y proceso de cambios
12	Capacitación y transferencia	Sesiones, destinatarios, material, accesos y entrega de código
13	Garantía, soporte y operación	Cobertura, exclusiones, canales y SLA solo si están acordados
14	Condiciones comerciales	Precio, impuestos, hitos, costos de terceros, licencias y pago, confirmados
15	Supuestos, exclusiones y riesgos	Responsabilidades del cliente, límites y dependencias externas; pendientes críticos que deben resolverse antes de la firma

#	Sección	Contenido
16	Anexos	Glosario, matriz de requisitos, integraciones, pruebas y registro de decisiones; después, las referencias

Las secciones sin información confirmada **no se omiten**: se dejan con `[PENDIENTE DE VALIDACIÓN]`, nunca se rellenan con texto genérico.

Las plantillas de trabajo

Los archivos de `trabajo/` se crean a partir de `plantillas/`. Cada una respeta las columnas del manual:

Plantilla	Para qué sirve	Columnas clave
`ficha-funcional.md`	Una ficha por funcionalidad, dentro de su módulo	ID y nombre · objetivo y usuario · precondiciones y entradas · flujo principal · reglas y excepciones · salidas · dependencias · criterios de aceptación · estado
`matriz-trazabilidad.md`	Un renglón por requisito; los ID no se renumeran entre versiones	ID · requisito · fuente o decisión · prioridad · fase · criterio de aceptación · estado
`matriz-riesgos-pendientes.md`	Las preguntas ordenadas por criticidad	ID · criticidad · tema · riesgo o dato faltante · impacto · responsable · acción o decisión
`matriz-informacion.md`	Un renglón por hecho, decisión, cifra, exclusión o contradicción	# · bloque · hecho · fuente · estado · notas; más una tabla de contradicciones (CON-001…)
`registro-decisiones.md`	Cada respuesta que cierra un punto, con su origen	ID · fecha · tema · decisión · quién decide · origen · afecta a
`control-calidad.md`	El checklist previo a la entrega, con evidencia	Marca · control · evidencia

El documento Word

El manual pide un documento **editable**, por lo que se entrega en `.docx` construido **siempre sobre** `recursos/plantilla/Plantilla Word - Campuslands.docx` (membrete como imagen de fondo, márgenes y número de página); nunca sobre un documento en blanco. Se arma con `scripts/alcance_docx.py`, que expone la clase `Alcance`.

Qué ofrece `Alcance`

Método	Qué hace
`portada(d)`	La portada estándar: rótulo, título, cliente y dos tarjetas
`carta_presentacion(d)`	La carta de una hoja, con la lista de contenidos y la firma
`tabla_de_contenido(niveles)`	La tabla de contenido: se genera sola al guardar y calcula los números de página con LibreOffice
`h1` a `h4`, `parrafo`, `vinetas`, `referencias`	Títulos, texto, listas y referencias
`tabla(titulo, encabezados, filas, pesos)`	Tabla APA con título, nota y encabezado que se repite
`ficha_funcional(rf_id, nombre, campos)`	Ficha funcional con encabezado azul
`firmas(izq, der, …)`	El apartado «Aprobación y firmas»
`guardar(ruta, titulo, tema)`	Cierra el documento y lo guarda

Reglas visuales

El documento sigue un formato **APA 7 adaptado a documento comercial**:

- Cuerpo en **Calibri 11 pt, justificado**, interlineado 1,15 y 6 pt posteriores. Títulos y celdas de tabla, alineados a la izquierda.
- Títulos con jerarquía: **H1** de 16 pt en negrita azul `#0D497B` con línea verde `#5DAC6D`; **H2** de 13 pt; **H3** de 11,5 pt en negrita cursiva; **H4** de 11 pt en negrita gris.
- **Tablas APA**: «Tabla N» en negrita y título en cursiva sobre la tabla, reglas horizontales sin líneas verticales, encabezado en tinte azul y una nota debajo. Las filas no se parten y el encabezado se repite.

- **Estados** resaltados con estilo de carácter; el constructor los aplica solo.
- **Referencias** en estilo APA, con sangría francesa y orden alfabético; fechas como «28 de octubre de 2026».
- Colores de la paleta del membrete (azul y verde); el ámbar es solo para pendientes.

Cómo evitar páginas vacías

Es una prioridad acordada con el usuario: **no se fuerzan saltos de página entre secciones** (solo la portada, la carta y la tabla de contenido empiezan página), las fichas funcionales fluyen entre páginas, y ninguna fila de tabla se parte. Tras generar el documento, `previsualizar.py` avisa de las páginas con menos de 60 % de ocupación; el agente las corrige recortando o reordenando, salvo la hoja de firma, que puede quedar sola.

La firma

La columna izquierda del apartado de firmas es fija (la representante legal de Campuslands, definida en la constante `FIRMA_IZQ`); la derecha varía por proyecto y sus datos salen de la fuente; lo que falte queda `[PENDIENTE DE VALIDACIÓN]`. La firma escaneada solo se inserta si el usuario pide un documento **listo para firma**, nunca en borradores.

Herramientas

Herramienta	Uso	Notas
`previsualizar.py`	`python scripts/previsualizar.py "<archivo.docx>" <carpeta> [escala]`	Genera el PDF con LibreOffice y un PNG por página con `pypdfium2`; avisa de páginas con baja ocupación
`alcance_docx.py`	Se importa desde el script de cada proyecto	Constructor del documento
Plantilla Word	`recursos/plantilla/Plantilla Word - Campuslands.docx`	Base obligatoria del documento

Dependencias: `python-docx`, LibreOffice (por ejemplo, con `winget install TheDocumentFoundation.LibreOffice`) y `pypdfium2`.

UN LÍMITE CONOCIDO

Los números de página de la tabla de contenido salen de LibreOffice. En Word pueden diferir en una página; se corrigen actualizando el campo (clic derecho, «Actualizar campo»).

Control de calidad

Antes de entregar, `trabajo/05-control-calidad.md` debe marcar, **con evidencia**, cada uno de estos nueve puntos del manual:

1. Cada módulo tiene identificador, límite, entradas, salidas y aceptación.
2. Los datos comerciales coinciden con la versión aprobada de la propuesta.
3. El cronograma contempla dependencias y condiciones de inicio.
4. Las funcionalidades futuras no aparecen como entregables actuales.
5. Soporte y garantía distinguen errores de nuevas solicitudes.
6. Los costos recurrentes y de terceros están identificados o marcados como pendientes.
7. Fuentes, decisiones y versiones son trazables.
8. No hay contradicciones entre resumen, tablas, entregables y exclusiones.
9. Se entrega el resumen de pendientes y se indica si es borrador o versión apta para revisión final.

Además, el agente revisa **a la vista cada página** del documento: portada sin desbordar a una segunda página, sin tablas partidas ni celdas con texto recortado, márgenes parejos y nada superpuesto al membrete. Solo entonces declara el documento **«apto para revisión final»**, y siempre pide revisión técnica, comercial y jurídica donde aplique.

Convenciones

Elemento	Convención
Identificadores	`RF-001` requisitos · `MOD-01` módulos · `INT-01` integraciones · `DEC-001` decisiones · `PEN-001` pendientes · `CON-001` contradicciones. Estables entre versiones
Archivo de entrega	`entrega/Alcance Funcional - <Cliente> - v<N.M> - <YYYY-MM-DD>.docx`; las versiones anteriores no se sobrescriben

Elemento	Convención
Estados en `wiki/index.md`	`Borrador` · `Apto para revisión final` · `Aprobado por cliente`. Nunca «listo para firma» con bloqueantes abiertos
Slug de cliente	Minúsculas, sin espacios ni tildes, con guiones
Fechas	Absolutas (`2026-10-29`), nunca relativas
Versión	Solo en el nombre del archivo y en la wiki; nunca dentro del documento
Idioma	Español

Comandos «!»

Los comandos son órdenes cortas del usuario (`!optimize` , `!notime` ...) que el agente debe ejecutar de forma obligatoria, siempre dentro de sus reglas duras. Hay un solo catálogo para todos los agentes y de él salen las instrucciones de cada uno, las bitácoras y las tablas de las guías.

Dónde vive cada cosa

Qué	Dónde
Catálogo, regla común y qué comandos tiene cada agente	`resources/comandos/comandos.py` (la fuente única; no se edita a mano nada de lo que genera)
Catálogo legible, matriz por agente y decisiones	`resources/comandos/catalogo.md`
Instrucciones del agente	sección «Comandos (`!`)» de `CLAUDE.md` , entre las marcas `<!-- comandos:inicio -->` y `<!-- comandos:fin -->`
Tabla de esta guía	se genera del mismo catálogo al compilar las guías

La regla común

1. Un comando es una palabra precedida de `!` al inicio de un mensaje o de una línea; el texto que sigue es su objeto.

2. Es obligatorio: el agente no pide permiso ni lo reinterpreta. Solo lo limita una **regla dura** (veracidad, marca, confidencialidad, puertas de aprobación y de verificación); entonces hace lo que la regla permite y dice qué no pudo.
3. Se combinan. Si dos modos chocan, el orden es: reglas duras > `!strict-mode` y `!no-assumptions` > `!think-deep`, `!critical-mode` y `!double-check` > `!notime`.
4. Rige la tarea en curso, no las siguientes.
5. La primera línea de la respuesta nombra los comandos recibidos.
6. Un comando desconocido no se adivina: se propone el más parecido y se pregunta. `!help` lista los disponibles.
7. Si un comando cambió un entregable o la wiki, se anota en `wiki/log.md`.

Cómo añadir o cambiar un comando

1. Edita `CMDS` (definición), `AGENTES` (a quién aplica y su nota) o `ALIAS` en `resources/comandos/comandos.py`.
2. Corre `python -X utf8 resources/comandos/comandos.py`: reescribe la sección de cada agente, el catálogo y, la primera vez, añade la entrada a su bitácora.
3. Recompila las guías (`python guides/_src/build.py`).
4. `python resources/comandos/comandos.py --verificar` falla si algo quedó desfasado.

Comandos de este agente

General

Comando	Qué hace aquí
`!analyze`	Analiza la solicitud, el problema, los documentos o la situación **antes de actuar**: qué se pide, qué hay, qué falta y qué riesgos ve. Entrega hallazgos; no ejecuta ni modifica nada. **Aquí:** Lee las fuentes del cliente y detecta contradicciones, vacíos y supuestos sin redactar todavía el documento.
`!research`	Investiga a fondo el tema, cruza fuentes y entrega lo relevante con **cada dato citado a su fuente**. Sin fuente verificable no es un dato: se marca `[PENDIENTE]`.
`!plan`	Entrega un plan de trabajo estructurado (pasos, orden, insumos, entregables, supuestos y puntos de decisión) y **se detiene a esperar aprobación** antes de ejecutarlo. **Aquí:** Presenta la estructura del documento de alcance y la lista de preguntas pendientes; espera aprobación.
`!execute`	Ejecuta ya lo solicitado o el siguiente paso aprobado del flujo, sin volver a proponer ni a preguntar lo ya decidido. No se salta las puertas de aprobación ni de verificación.

Comando	Qué hace aquí
`!review`	Revisa el trabajo indicado y entrega oportunidades de mejora priorizadas (qué cambiaría y por qué). No lo modifica salvo que se le pida.
`!audit`	Auditoría integral de punta a punta: contradicciones entre documentos, cifras que no cuadran, reglas incumplidas y datos sin fuente. Entrega hallazgos con evidencia y severidad. **Aquí:** Busca contradicciones entre las fuentes y entre estas y el documento; entrega hallazgos con evidencia.
`!validate`	Comprueba, requisito por requisito, que el resultado cumple lo establecido (reglas duras, insumos, formato). Veredicto «cumple / no cumple» por ítem, con evidencia. **Aquí:** Aplica el **control de calidad** del manual (§8) al documento: estructura obligatoria, trazabilidad y cero compromisos inventados.
`!optimize`	Optimiza lo que se esté desarrollando o ya existe (proceso, solución o resultado): mejora claridad, calidad, consistencia, peso o rendimiento **sin cambiar su alcance ni sus datos**.
`!compare`	Compara alternativas con criterios explícitos y recomienda la más conveniente, con el porqué.
`!summarize`	Resumen claro y estructurado (lo esencial, decisiones y pendientes) del trabajo, de un documento o de la conversación.
`!document`	Documenta lo hecho: actualiza la wiki y la bitácora del agente, o genera la documentación técnica, funcional o ejecutiva que se pida.
`!prioritize`	Clasifica actividades, hallazgos o elementos por impacto, importancia y urgencia, y entrega el orden recomendado.
`!help`	Lista los comandos disponibles en este agente, agrupados, con su función.

Modos

Comando	Qué hace aquí
`!notime`	**Modo prisa:** apresura el desarrollo optimizándolo: agrupa preguntas en un solo mensaje, paraleliza, reutiliza lo ya validado, omite explicaciones y pasos opcionales y responde solo con lo esencial. **Nunca** se salta verificaciones obligatorias, puertas de aprobación ni la veracidad: acelera el camino, no recorta la calidad.

Comando	Qué hace aquí
`!think-deep`	Análisis más profundo antes de responder: considera alternativas, casos límite y consecuencias, y muestra solo el razonamiento que importa.
`!critical-mode`	Modo crítico: cuestiona decisiones, supuestos y su propio trabajo; busca errores y contradicciones antes de entregar y los dice.
`!no-assumptions`	No supone: ante un dato faltante o ambiguo pregunta (todas las dudas juntas) en lugar de rellenar o inferir. **Aquí:** Refuerza la política de faltantes: lo que no está en las fuentes se pregunta o queda `[PENDIENTE]`, nunca se completa.
`!double-check`	Segunda verificación independiente de su propio resultado (releer la fuente, recalcular, comparar) y corrección de lo que encuentre; reporta qué verificó.
`!strict-mode`	Sigue al pie de la letra los requisitos e instrucciones dados: sin iniciativas, adiciones ni interpretaciones creativas.

Técnico y seguridad

Comando	Qué hace aquí
`!debug`	Identifica la causa de un error de software (script, conversión, verificación) y lo corrige; vuelve a correr la verificación para comprobarlo.
`!troubleshoot`	Investiga un problema técnico y determina su **causa raíz**, con evidencia. Diagnostica; no cambia nada.
`!security-check`	Revisa seguridad y confidencialidad: datos de otros clientes mezclados, credenciales, datos personales, información que no debe salir del proyecto.

UX/UI y diseño

Comando	Qué hace aquí
`!visual-audit`	Examina proporciones, distribución, tipografía y consistencia visual del entregable, a la vista y página por página. **Aquí:** Revisa a la vista el documento entregado (portada, tablas, jerarquía, paginación).

Comando	Qué hace aquí
`!consistency-check`	Detecta inconsistencias entre secciones, componentes, cifras, nombres y estilos del mismo entregable.

Gestión de proyecto

Comando	Qué hace aquí
`!breakdown`	Divide un proyecto o solicitud compleja en actividades específicas, cada una con su entregable y su orden. **Aquí:** Divide el alcance en módulos y funcionalidades atómicas.
`!track-progress`	Informa el progreso y los pendientes (hecho, en curso, bloqueado, `[PENDIENTE]`) a partir de la bitácora y del trabajo actual. **Aquí:** Informa estados y pendientes del documento y quién debe resolver cada uno.
`!risk-analysis`	Identifica riesgos, su probabilidad e impacto, y propone la mitigación de cada uno. **Aquí:** Riesgos del alcance: requisitos ambiguos, dependencias externas, supuestos frágiles y su mitigación.
`!requirements`	Extrae y estructura requerimientos funcionales y no funcionales de las fuentes, cada uno con su referencia de origen. **Aquí:** Es el núcleo del agente: extrae requerimientos funcionales y no funcionales con su fuente, y marca lo ambiguo.
`!generate-report`	Genera un informe ejecutivo (decisiones, resultado, pendientes) o técnico (método, evidencias, verificaciones) del trabajo.
`!final-delivery`	Prepara y valida la entrega final: corre todas las verificaciones obligatorias, completa el checklist de cierre (wiki y bitácora) y entrega solo si todo pasa. **Aquí:** Control de calidad completo, documento final con el formato de entrega y wiki/bitácora al día.
`!agent-handoff`	Prepara el traspaso al siguiente agente del flujo comercial: resumen, entregables con su ruta, decisiones tomadas y pendientes. **Aquí:** Prepara el traspaso a **Desarrollador de Cotizaciones** (`quote-developer`): alcance aprobado, módulos y supuestos.

IA avanzada

Comando	Qué hace aquí
`!second-opinion`	Un subagente independiente evalúa el resultado **sin ver tu razonamiento**; se contrasta con el tuyo y se explican las diferencias.
`!cross-review`	Otro especialista (subagente con otro rol) revisa el resultado y devuelve observaciones; se aplican las válidas y se informa de las descartadas.

Multiagente

Comando	Qué hace aquí
`!orchestrate`	Reparte partes **independientes** del trabajo entre varios subagentes en paralelo y las integra en un solo resultado verificado.
`!team-review`	Varios revisores con roles distintos evalúan por separado el mismo resultado y se consolida su evaluación. **Aquí:** Revisores con rol **funcional**, **QA (contradicciones y vacíos)** y **comunicación con el cliente**.
`!consensus`	Consolida las evaluaciones disponibles (de subagentes o de revisiones previas) en una sola recomendación, con las discrepancias a la vista.

Mantenimiento

Al terminar cada entrega, el agente actualiza `wiki/index.md` (proyecto, versión y estado), añade una entrada a `wiki/log.md` con lo hecho, las decisiones confirmadas, los pendientes abiertos y el resultado del control de calidad, y entrega al usuario el documento, el **resumen ejecutivo de cambios** y el **registro de pendientes** con el responsable de cada uno. La bitácora usa `## [YYYY-MM-DD] <tipo> | <cliente> — descripción`.

Quiero...	Qué tocar
Incorporar una regla nueva del usuario	`CLAUDE.md` y `wiki/manual-operativo.md`, en el mismo cambio
Cambiar el diseño del documento	`scripts/alcance_docx.py` y las plantillas `.docx`; regenerar y revisar a la vista

Quiero...	Qué tocar
Cambiar columnas de una matriz	La plantilla de `plantillas/` correspondiente (respetando el manual §5)
Cambiar la columna derecha de la firma	Los datos del proyecto; la izquierda es la constante `FIRMA_IZQ`
Registrar un proyecto nuevo	Carpeta en `alcances/`, fila en `wiki/index.md` y entrada en `wiki/log.md`

Problemas frecuentes

Síntoma	Causa probable	Qué hacer
Dos fuentes dicen cosas distintas	Versiones distintas del mismo dato	Mostrar ambas versiones con su fuente y listarlo en pendientes; no elegir en silencio
Páginas casi vacías tras generar	Contenido que fluye mal	`previsualizar.py` avisa de las de menos de 60 %; recortar o reordenar. La hoja de firma puede quedar sola
La tabla de contenido tiene otras páginas en Word	LibreOffice pagina distinto	Actualizar el campo en Word
Piden «listo para firma» con bloqueantes	Presión de plazo	No se declara: se entrega borrador y la lista de validaciones por responsable
Falta un dato comercial (precio, garantía, SLA)	La fuente no lo dice	Marcar `[PENDIENTE DE VALIDACIÓN]`; nunca redactarlo «por defecto»
Un ejemplo de `wiki/ejemplos/` aparece en otro proyecto	Se copió en lugar de aplicar el método	Retirarlo: los ejemplos nunca se trasladan

Cosas que no están alineadas

Dos plantillas conservan el lenguaje de la versión anterior del flujo, cuando el agente entrevistaba al usuario por bloques:

- `plantillas/matriz-informacion.md` dice que se completa «antes de preguntar al usuario» y que ante una contradicción hay que «pedir al usuario cuál prevalece». La regla vigente es mostrar **ambas versiones** sin detener el trabajo.
- `plantillas/registro-decisiones.md` titula la etapa como «Entrevista». La regla vigente es que no hay entrevista; solo se registra lo que el usuario dice por iniciativa propia.

Ninguna de las dos cambia el comportamiento del agente (manda `CLAUDE.md`), pero conviene actualizarlas cuando se toque el flujo.

Referencia rápida de archivos

Archivo	Para qué lo abres
`CLAUDE.md`	Reglas, flujo, estructura del documento y diseño
`wiki/manual-operativo.md`	Resumen del manual y decisiones de interpretación
`plantillas/*.md`	Las matrices y fichas de trabajo
`scripts/alcance_docx.py`	El constructor del documento Word
`scripts/previsualizar.py`	Revisar el documento a la vista
`wiki/index.md`	Estado de cada proyecto
`wiki/log.md`	Historial de cambios y entregas