Facsímil 06

Construir y operar

Capítulo 08

Handoffs operativos y revisión humana

686f6c61 · 2026

Capítulo 08: Handoffs operativos y revisión humana

Qué deberías poder hacer al terminar

En el capítulo 07 aprendimos a exponer cambios poco a poco. Ahora aparece una situación igual de importante: **qué ocurre cuando una run no debería seguir sola**.

Un sistema de IA real no vive solo en el modelo. Vive en colas, tickets, contratos, herramientas, permisos, revisiones, trazas y decisiones. A veces el sistema puede responder. A veces debe abstenerse. A veces debe preparar una acción, pero no ejecutarla. Y a veces debe entregar el caso a una persona con todo lo necesario para decidir sin reconstruir la historia desde cero.

Al terminar, deberías poder hacer esto:

RESULTADO DE APRENDIZAJE	EVIDENCIA DE QUE LO SABES HACER
Diseñar un handoff operativo.	Sabes qué campos mínimos debe entregar una run al pausarse.
Decidir cuándo pedir revisión.	Usas reglas por confianza, efecto, coste, contrato, evidencia y permisos.
Construir una cola de revisión.	Priorizas casos por impacto, antigüedad, SLA y falta de evidencia.
Diseñar una tarjeta de decisión.	La persona ve entrada, propuesta, fuentes, trazas, coste y acciones disponibles.
Reanudar una run.	Sabes continuar con `approve`, `reject` o `needs_more_info` sin perder estado.
Medir la cola.	Defines SLI, SLO, backlog, aging, tasa de acuerdo y coste humano.
Convertir revisión en aprendizaje.	Los casos revisados vuelven a evals, prompts, contratos y runbooks.

La idea central: **pedir revisión no es fracasar; fracasar es pedirla sin contexto, sin prioridad y sin reanudación clara**.

Cuando el sistema no debe seguir solo

Imagina un asistente interno que prepara respuestas para soporte universitario. La mayoría de preguntas son sencillas: horarios, plazos, enlaces, estado de una solicitud. Pero un día llega un caso con documentos incompletos, una norma que ha cambiado, un dato que contradice el expediente y una respuesta propuesta que podría cerrar el ticket.

El sistema puede tener una respuesta plausible. Eso no basta. La pregunta operativa es otra:

> ¿Tenemos evidencia suficiente para que esta acción siga automáticamente, o debemos entregar el caso a revisión?

Un handoff operativo evita dos extremos malos. El primero: automatizar de más y descubrir tarde que faltaba una condición. El segundo: mandar todo a revisión y convertir el sistema en una bandeja de espera. La ingeniería está en diseñar el punto medio.

Qué no es revisión humana

Revisión humana no es añadir un botón de “aprobar” al final de una pantalla. Si la persona no ve el motivo, los datos usados, la salida propuesta, las fuentes, los límites y el efecto de cada botón, está firmando a oscuras.

Tampoco es revisar todo. Si cada respuesta trivial pide intervención, la cola crece, el tiempo de respuesta empeora y la revisión pierde valor. La persona debe entrar donde aporta criterio: falta evidencia, hay efecto persistente, la confianza no alcanza el umbral, el contrato falla, el coste se sale del presupuesto o la acción afecta a un recurso sensible.

Y no es “que lo decida soporte” como cajón de sastre. Una cola sin SLO, prioridad, dueño y estado es deuda operativa. El sistema debe explicar qué necesita y qué pasará después de la decisión.

CONFUSIÓN	QUÉ FALTA
“Lo revisa una persona y ya está”	Falta evidencia estructurada, estado y reanudación.
“Todo lo dudoso va a revisión”	Falta priorización y criterios de entrada.
“La persona corrige a mano”	Falta convertir la corrección en dataset, regla o cambio de contrato.
“Aprobar es pulsar sí”	Falta mostrar efecto, alcance y alternativa.
“La cola es una bandeja”	Falta SLI, SLO, aging, backlog y dueño.

Qué sí es un handoff operativo

Un handoff operativo es una pausa con contrato. La run entrega un paquete de evidencia, queda en un estado recuperable y espera una decisión explícita.

Un handoff se compone de estos campos. No es una ecuación de la literatura, es una lista de ingeniería:

SÍMBOLO	SIGNIFICADO	EJEMPLO
x	Entrada original.	Pregunta del usuario y documentos consultados.
y	Salida propuesta.	Respuesta, JSON, diff o acción preparada.
a	Acción pendiente.	Enviar, publicar, actualizar, derivar, cerrar ticket.
e	Evidencia disponible.	Fuentes, trazas, validaciones, métricas, contrato.
r	Razón de revisión.	Baja confianza, efecto persistente, falta de fuente, coste alto.
q	Cola o persona responsable.	`soporte_n2`, `legal_ops`, `ai_platform`.
t	Tiempo objetivo de resolución.	30 minutos, 4 horas, 1 día laboral.
s	Estado serializable de la run.	`run_state`, `trace_id`, `resume_token`.

El sistema no debería decir solo “necesita revisión”. Debería decir: “necesita revisión por estas razones, con esta evidencia, antes de este tiempo, y estas son las decisiones posibles”.

Fecha de corte del estado del arte

Fecha de corte: 28 de mayo de 2026.
Fuentes consultadas: documentación oficial de OpenAI Agents SDK sobre human-in-the-loop y handoffs; documentación de Anthropic Claude Agent SDK sobre permisos, hooks y aprobaciones; documentación de Google ADK sobre callbacks y controles para agentes; LangGraph interrupts; OpenTelemetry Traces; NIST AI RMF; y trabajos académicos sobre ingeniería de ML, model cards y datasheets.

OpenAI Agents SDK describe un flujo donde una herramienta puede marcarse como pendiente de aprobación, la ejecución se pausa con interrupciones, el estado de la run se serializa y después se reanuda con aprobación o rechazo.[1] También documenta handoffs como mecanismo para transferir control entre agentes especializados.[2]

Anthropic documenta permisos, reglas de allow/deny, modos de permiso, hooks y callbacks de aprobación para controlar cuándo una herramienta se ejecuta, se bloquea o pide una decisión.[3][4][5]

Google ADK presenta callbacks para observar, personalizar y controlar el comportamiento de agentes, y recomienda tratar los agentes como sistemas con controles explícitos, evaluación y límites.[6][7] LangGraph usa interrupts para pausar grafos, pedir decisión humana y reanudar desde un checkpoint.[8]

Lo estable no es una API concreta. Lo estable es el patrón: detectar, pausar, empaquetar evidencia, decidir, reanudar, registrar y aprender.

Cuándo disparar revisión

Un buen sistema no pregunta a una persona por vergüenza. Pregunta porque una regla dice que ahí la revisión tiene valor.

DISPARADOR	SEÑAL CONCRETA	DECISIÓN TÍPICA
Baja confianza	`confidence < 0.72`.	Revisar antes de responder.
Contrato roto	JSON inválido, campo obligatorio ausente, enum inesperado.	Bloquear y pedir corrección.
Falta de evidencia	No hay fuente, cita o documento suficiente.	Pedir más información.
Fuentes en tensión	Dos documentos sostienen respuestas distintas.	Revisar con expediente completo.
Efecto persistente	Enviar, publicar, cerrar, modificar o borrar.	Aprobar antes de ejecutar.
Coste alto	La ruta supera presupuesto por tarea.	Revisar si merece usar ruta cara.
Permiso insuficiente	La tool requiere scope que la run no tiene.	Derivar a persona responsable.
Cola saturada	El sistema entraría en demora no aceptable.	Responder con estado claro o degradar.
Canary	Candidate toca casos nuevos.	Muestrear revisión para aprender.
Usuario lo pide	La persona quiere contacto humano.	Crear handoff explícito.

La regla práctica: cada disparador debe tener una salida concreta. Si una condición solo dice “mirar”, no es una política; es una nota suelta.

Fórmula sencilla de decisión

La decisión de revisar o automatizar es una comparación de coste esperado, el criterio de la teoría de la decisión sensible al coste: se revisa cuando el coste esperado de un fallo automático (su probabilidad por su impacto) supera el coste de revisar.[9]

$$R(x) = \begin{cases} \text{review} & \text{si } p_f(x) \cdot C_f(x) > C_h(x) + C_d(x) \\ \text{auto} & \text{si } p_f(x) \cdot C_f(x) \leq C_h(x) + C_d(x) \end{cases}$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
$R(x)$	Decisión para el caso x.	`review` o `auto`.
$p_f(x)$	Probabilidad estimada de fallo relevante.	0,18.
$C_f(x)$	Coste si el sistema se equivoca.	80 EUR equivalentes o impacto operativo alto.
$C_h(x)$	Coste de revisión humana.	6 minutos de una persona.
$C_d(x)$	Coste de demora.	Retraso sobre SLA o experiencia.

Ejemplo numérico:

VARIABLE	VALOR
$p_f(x)$	0,18
$C_f(x)$	80
$C_h(x)$	6
$C_d(x)$	3

Calculamos:

$$p_f(x) \cdot C_f(x) = 0{,}18 \cdot 80 = 14{,}4$$

$$C_h(x) + C_d(x) = 6 + 3 = 9$$

Como $14{,}4 > 9$, revisamos. No porque “dé miedo”, sino porque el coste esperado de seguir automáticamente supera el coste de parar y decidir mejor.

Esta fórmula no sustituye criterio profesional. Lo hace discutible. Si alguien quiere cambiar el umbral, tiene que hablar de probabilidades, costes, demoras y evidencia.

¿Coinciden los revisores? La kappa de Cohen. Si dos personas revisan los mismos casos y no se ponen de acuerdo, tu «verdad» no es fiable. El coeficiente kappa mide el acuerdo entre revisores descontando el que ocurriría por puro azar, que es lo que la simple coincidencia porcentual oculta.[10]

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
p_o	Acuerdo observado entre revisores.	0,90
p_e	Acuerdo esperable por azar.	0,60
κ	Acuerdo corregido por azar.	$(0{,}9-0{,}6)/(1-0{,}6)=0{,}75$.

En palabras: dos revisores que coinciden el 90% pueden tener una kappa modesta si ese tema es fácil y el azar ya explicaba el 60%. Una kappa baja avisa de que la rúbrica es ambigua o los revisores entienden cosas distintas: antes de fiarte de las etiquetas humanas que alimentan tus evals, conviene medir si esas etiquetas son consistentes.

¿Cuándo molestar a un humano? Riesgo y cobertura. Mandar a revisión todo no escala; no mandar nada es temerario. La predicción selectiva formaliza el equilibrio: el sistema responde solo cuando su confianza supera un umbral y delega el resto, y se mide con dos cantidades, cuánto cubre y con qué error.[11]

$$\text{cobertura} = \frac{1}{N}\sum_{i=1}^{N} 1[\,g(x_i) \geq \tau\,], \qquad \text{riesgo} = \frac{\sum_i 1[\,g(x_i)\geq\tau\,]\,\ell_i}{\sum_i 1[\,g(x_i)\geq\tau\,]}$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
$g(x_i)$	Confianza del sistema en el caso i.	0,2 a 0,99.
τ	Umbral por encima del cual responde solo.	0,85
cobertura	Fracción que el sistema atiende sin humano.	0,70
riesgo	Tasa de error entre los casos atendidos.	0,02

En palabras: subir el umbral baja la cobertura pero también el error de lo que se responde solo. La curva riesgo-cobertura te deja elegir el punto de operación con datos, por ejemplo «automatizo el 70% con un 2% de error y delego el 30% dudoso», en vez de decidir el umbral por intuición.

¿Significa algo la confianza del modelo? La puntuación de Brier. La confianza solo sirve para decidir handoffs si está calibrada: si el modelo dice 0,9, debería acertar el 90% de esas veces. La puntuación de Brier mide esa calibración como el error cuadrático entre la probabilidad anunciada y lo que de verdad ocurre.[12]

$$\text{BS} = \frac{1}{N}\sum_{i=1}^{N}(f_i - o_i)^2$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
f_i	Confianza anunciada para el caso i.	0,9
o_i	Resultado real: 1 si acertó, 0 si no.	0 o 1.
BS	Puntuación; más baja es mejor.	0 perfecto, 0,25 es azar.

En palabras: un Brier bajo significa que la confianza es de fiar y se puede usar como umbral de handoff; uno alto significa que el «0,9» del modelo no quiere decir 90%, y entonces enrutar por esa confianza es enrutar por una ilusión. Calibrar antes de delegar es lo que evita mandar a revisión los casos equivocados.

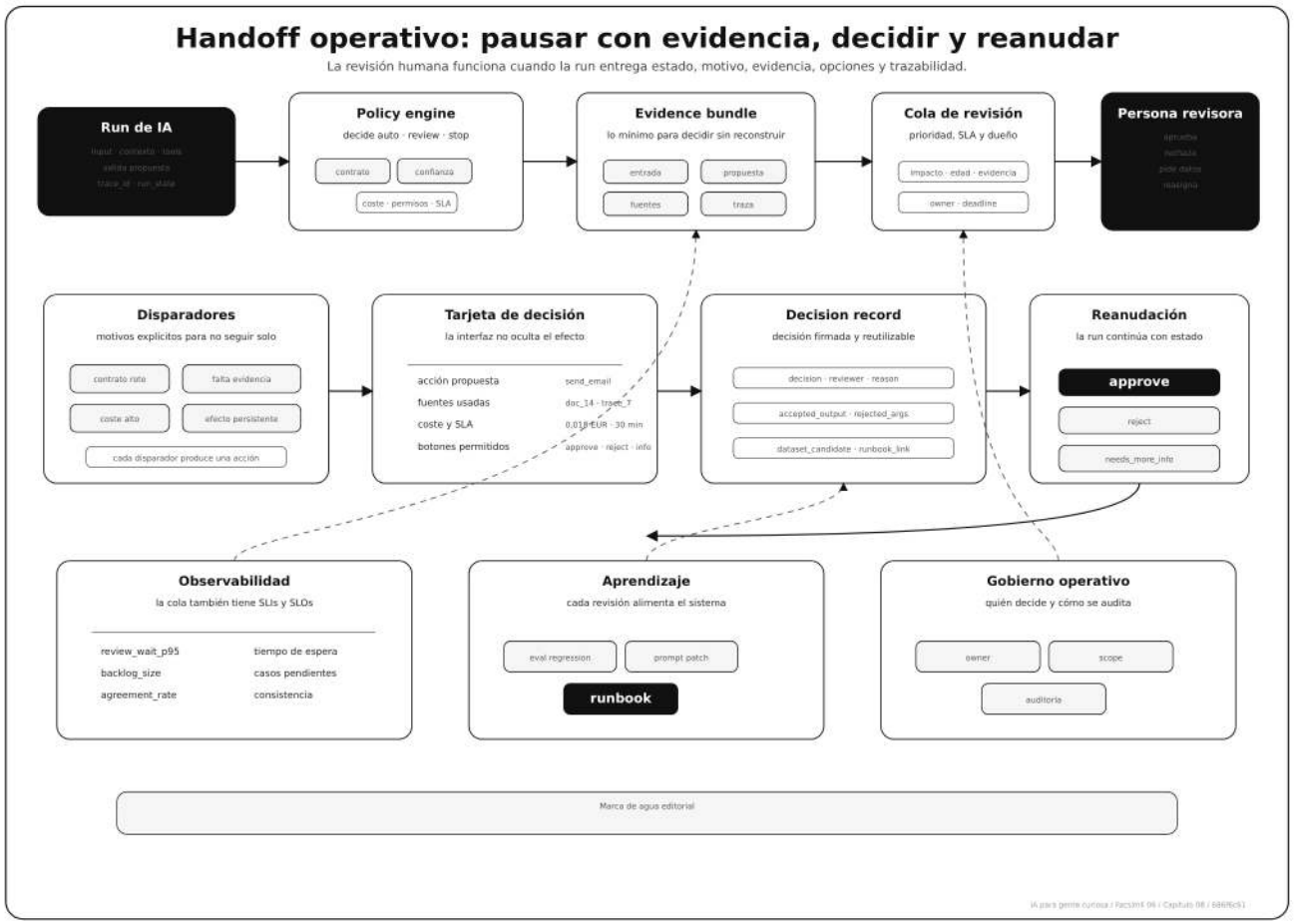
¿Se va a saturar la cola de revisión? La ley de Little. Una bandeja de revisión es una cola, y obedece la misma ley que las colas del runtime (la vimos en el capítulo 2): el trabajo acumulado es la tasa de entrada por el tiempo que cada caso pasa esperando.[13]

$$L = \lambda W$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
L	Casos esperando revisión en media.	60
λ	Casos que llegan a revisión por hora.	30 por hora.
W	Tiempo medio que un caso espera.	2 horas.

En palabras: si llegan 30 handoffs por hora y cada uno tarda 2 horas en revisarse, siempre hay unos 60 en cola. Si quieres que la espera baje a media hora con esa entrada, necesitas capacidad para que solo haya 15 acumulados, lo que fija cuántos revisores hacen falta. Sin esta cuenta, una cola de revisión sin caducidad ni dimensionado se convierte en el cuello de botella que vacía de sentido la supervisión humana.

Anatomía visual de un handoff operativo



El contrato mínimo del handoff

Un handoff serio debería poder serializarse. Si no puede guardarse, moverse de cola, auditarse y reanudarse, dependerá demasiado de la memoria de una persona.

Un contrato útil contiene:

CAMPO	MOTIVO	EJEMPLO
`request_id`	Identifica la revisión.	`rev_20260528_001`.
`run_id`	Une revisión con ejecución.	`run_7f31`.
`trace_id`	Permite abrir la traza completa.	`trace_abc`.
`task`	Agrupa métricas por caso de uso.	`support_reply`.
`proposed_action`	Dice qué se quiere hacer.	`send_email`.
`effect`	Describe impacto técnico.	`external_message`, `db_update`, `none`.
`reason_codes`	Explica por qué se revisa.	`low_confidence`, `missing_source`.
`evidence`	Resume pruebas disponibles.	Fuentes, validaciones, diffs, score.
`missing_evidence`	Dice qué falta.	`policy_doc`, `user_confirmation`.
`options`	Limita decisiones posibles.	`approve`, `reject`, `needs_more_info`.
`deadline_at`	Hace visible el SLO.	`2026-05-28T15:30:00Z`.
`resume_token`	Permite continuar la run.	Token opaco o estado serializado.

Ejemplo:

```
{
  "request_id": "rev_20260528_001",
  "run_id": "run_7f31",
  "trace_id": "trace_abc",
  "task": "support_reply",
  "proposed_action": "send_email",
  "effect": "external_message",
  "confidence": 0.68,
  "cost_eur": 0.018,
  "reason_codes": ["low_confidence", "missing_source"],
  "evidence": {
    "draft": "Hemos revisado tu solicitud...",
    "sources": ["doc_matricula_2026"],
    "schema_valid": true,
    "policy_version": "handoff_policy@3"
  },
  "missing_evidence": ["expediente_actualizado"],
  "options": ["approve", "reject", "needs_more_info"],
  "deadline_at": "2026-05-28T15:30:00Z",
  "resume_token": "resume_run_7f31_step_12"
}
```


La persona no recibe “un caso raro”. Recibe un objeto de trabajo.

Máquina de estados del handoff

Para ingeniería, un handoff no debería ser un booleano `needs_review=true`. Debe tener una máquina de estados. Si no la tiene, aparecen dudas en producción: ¿se puede aprobar dos veces?, ¿qué pasa si caduca?, ¿quién puede reasignar?, ¿qué ocurre si la run ya se reanudó?

Una máquina mínima:

ESTADO	QUÉ SIGNIFICA	TRANSICIONES VÁLIDAS
`running`	La run sigue trabajando.	`needs_review`, `completed`, `failed`.
`needs_review`	La política exige revisión.	`queued`, `cancelled`.
`queued`	El caso está en cola con prioridad y deadline.	`assigned`, `expired`, `cancelled`.
`assigned`	Una persona o equipo lo tiene asignado.	`in_review`, `reassigned`, `expired`.
`in_review`	La tarjeta está abierta o en edición.	`approved`, `approved_with_edits`, `rejected`, `needs_more_info`.
`needs_more_info`	Falta evidencia concreta.	`queued`, `cancelled`.
`approved`	La acción puede ejecutarse como venía.	`resuming`.
`approved_with_edits`	La acción puede continuar con cambios humanos.	`resuming`.
`rejected`	La acción no debe ejecutarse.	`resuming`.
`resuming`	Un worker está aplicando la decisión.	`closed`, `resume_failed`.
`resume_failed`	La decisión existe, pero reanudar falló.	`resuming`, `closed_manual`.
`expired`	Se incumplió el tiempo objetivo sin decisión.	`reassigned`, `cancelled`, `fallback_response`.
`closed`	El caso terminó con traza y decision record.	Estado terminal.

Reglas que conviene escribir:

REGLA	POR QUÉ IMPORTA
Una revisión cerrada no se modifica; se crea una revisión nueva.	Conserva auditoría y evita reescribir historia.
`approved` no ejecuta directamente; pasa por `resuming`.	Permite idempotencia y control de efectos.
`expired` no decide por sí solo.	La caducidad activa una política: reasignar, degradar o responder con estado claro.
`needs_more_info` debe nombrar qué falta.	Evita que la run vuelva a la cola sin mejora.
Cada transición emite un evento.	La cola se puede auditar y medir.

Modelo de datos persistente

El contrato JSON sirve para transportar el caso. En producción hace falta persistencia. Un modelo relacional mínimo podría ser:

```
create table handoff_requests (
  request_id text primary key,
  run_id text not null,
  trace_id text not null,
  task text not null,
  queue_name text not null,
  state text not null,
  proposed_action text not null,
  effect text not null,
  confidence numeric not null,
  cost_eur numeric not null,
  reason_codes jsonb not null,
  evidence jsonb not null,
  missing_evidence jsonb not null,
  resume_token text not null,
  idempotency_key text not null unique,
  created_at timestamptz not null,
  deadline_at timestamptz not null,
  updated_at timestamptz not null
);

create table review_decisions (
  decision_id text primary key,
  request_id text not null references handoff_requests(request_id),
  decision_version integer not null,
  reviewer text not null,
  decision text not null,
  reason text not null,
  edited_output text,
  created_at timestamptz not null,
  unique (request_id, decision_version)
);

create table review_events (
  event_id text primary key,
  request_id text not null references handoff_requests(request_id),
  event_type text not null,
  actor text not null,
  payload jsonb not null,
  created_at timestamptz not null
);

create table review_assignments (
  assignment_id text primary key,
  request_id text not null references handoff_requests(request_id),
  queue_name text not null,
  assignee text,
  assigned_at timestamptz not null,
  released_at timestamptz
);
```

Consultas que un equipo debería poder hacer:

```
select state, count(*) as n
from handoff_requests
where created_at >= now() - interval '24 hours'
group by state
order by n desc;
```

```
select
  queue_name,
  percentile_cont(0.95) within group (order by extract(epoch from updated_at - created_at) /
60) as age_p95_minutes,
  avg(case when now() > deadline_at and state not in ('closed', 'cancelled') then 1 else 0
end) as active_breach_rate
from handoff_requests
where created_at >= now() - interval '7 days'
group by queue_name;
```

La base de datos no es burocracia. Es lo que permite reanudar después de un crash, medir la cola, auditar decisiones y aprender de patrones repetidos.

Idempotencia: aprobar dos veces no debe ejecutar dos veces

El punto más delicado para un ingeniero de IA no es pedir revisión. Es aplicar la decisión sin duplicar efectos.

Ejemplo de fallo realista:

1. Una persona aprueba `send_email`.
2. El worker ejecuta la tool.
3. El proceso cae antes de marcar `closed`.
4. El sistema reintenta.
5. Si no hay idempotencia, envía dos correos.

Campos que evitan ese problema:

CAMPO	QUÉ PROTEGE
`idempotency_key`	La misma acción aprobada no se ejecuta dos veces.
`effect_id`	Identifica el efecto externo creado: email, ticket, cambio, publicación.
`decision_version`	Evita aplicar una decisión antigua tras una corrección.
`resume_token`	Reanuda el punto correcto de la run.
`state_version`	Bloquea carreras entre workers.
`executed_at`	Marca cuándo se produjo el efecto.

Patrón recomendado:

```
1. Leer request y decisión dentro de una transacción.
2. Comprobar `state=resuming` y `decision_version` vigente.
3. Reservar `idempotency_key`.
4. Ejecutar tool con esa clave.
5. Guardar `effect_id`.
6. Marcar `closed`.
7. Emitir evento `handoff.closed`.
```

Si la tool externa soporta claves de idempotencia, úsala. Si no, crea tu propia tabla de efectos y no ejecutes dos veces la misma combinación `(request_id, decision_version, proposed_action)`.

Colas, SLI y SLO de revisión

Una cola de revisión también se opera. Si no la medimos, solo veremos su dolor cuando alguien se queje.

SLI	QUÉ MIDE	EJEMPLO
`review_wait_p95`	Tiempo p95 hasta primera decisión.	38 minutos.
`backlog_size`	Casos pendientes por cola.	71 casos.
`deadline_breach_rate`	Proporción fuera de SLO.	4,2%.
`needs_more_info_rate`	Casos que llegaron con evidencia insuficiente.	19%.
`approval_rate`	Porcentaje aprobado.	63%.
`reversal_rate`	Decisiones corregidas después.	1,1%.
`review_cost_per_case`	Coste humano estimado por revisión.	3,40 EUR.
`agreement_rate`	Consistencia entre revisores en muestra doble.	0,86.

Un SLO posible:

$$SLO_{review} = P(T_{decision} \leq 30\ \text{min}) \geq 0{,}95$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
$T_{decision}$	Tiempo desde que se crea la revisión hasta primera decisión.	18 minutos.
$30\ \text{min}$	Objetivo máximo para la cola.	Media hora.
$0{,}95$	Proporción mínima dentro del objetivo.	95 de cada 100 casos.

El presupuesto de revisión no es infinito. Si la cola crece, tienes tres palancas:

PALANCA	QUÉ CAMBIA	CUIDADO
Mejorar evidencia	Menos `needs_more_info`.	Exige tocar prompts, retrieval o contrato.
Ajustar umbrales	Menos casos llegan a cola.	Puede aumentar automatización en casos discutibles.
Cambiar capacidad	Más personas o turnos.	Cuesta dinero y puede variar criterio.

Backpressure: qué hacer si la cola se satura

Una cola saturada no es solo un problema de soporte. Cambia el comportamiento del sistema de IA. Si el sistema sigue metiendo casos en revisión cuando nadie puede atenderlos, acumula deuda, incumple expectativas y oculta el fallo real.

Política de saturación:

SEÑAL	UMBRAL EJEMPLO	ACCIÓN
`backlog_size`	Más de 200 casos.	Bajar automatización que genera revisiones dudosas.
`review_wait_p95`	Más de 45 min.	Reasignar cola o ampliar capacidad temporal.
`deadline_breach_rate`	Más de 5%.	Responder con estado claro en vez de prometer resolución inmediata.
`needs_more_info_rate`	Más de 25%.	Mejorar bundle antes de crear revisión.
`agreement_rate`	Menos de 0,75.	Pausar decisiones automáticas basadas en esa rúbrica.

Estrategias de degradación:

ESTRATEGIA	QUÉ HACE	CUÁNDO USARLA
`queue_only_critical`	Solo entran casos de alto impacto.	Backlog alto y poco personal.
`auto_reject_low_evidence`	No crea revisión si falta evidencia básica.	Muchos casos llegan vacíos.
`fallback_response`	Informa que el caso queda pendiente.	La respuesta inmediata sería engañosa.
`batch_review`	Agrupa casos parecidos.	Muchos casos con mismo motivo.
`threshold_raise`	Exige más confianza para actuar solo o más evidencia para revisar.	Canary o incidente operativo.
`route_to_specialist`	Cambia cola según tema.	La cola general bloquea casos especializados.

La regla: **si la cola está llena, el sistema debe cambiar de comportamiento**. No basta con quejarse del backlog.

La tarjeta de decisión

La persona revisora no debería tener que buscar en diez sistemas. La tarjeta debe responder rápido a siete preguntas:

PREGUNTA	LO QUE DEBE VER
¿Qué quiere hacer el sistema?	Acción concreta y efecto.
¿Por qué llegó a revisión?	Motivos codificados y explicación corta.
¿Qué información usó?	Fuentes, documentos, campos y trazas.
¿Qué falta?	Evidencia ausente o dato que debe comprobarse.
¿Qué pasa si apruebo?	Acción exacta y recurso afectado.
¿Qué pasa si rechazo?	Mensaje a la run y siguiente estado.
¿Qué debo dejar escrito?	Motivo, corrección, caso de regresión o enlace a runbook.

Botones sanos:

BOTÓN	QUÉ HACE
`approve`	Reanuda la run y ejecuta la acción propuesta.
`approve_with_edits`	Reanuda con una salida corregida por la persona.
`reject`	No ejecuta la acción y devuelve motivo al sistema.
`needs_more_info`	Pide datos concretos y deja la run pausada.
`reassign`	Cambia cola o persona responsable.

Botones peligrosos por diseño:

BOTÓN	POR QUÉ LO EVITARÍA
`ok`	No dice qué ocurre.
`continuar`	Oculta si ejecuta, responde o solo guarda.
`resolver`	Mezcla decisión, ejecución y cierre.
`saltar`	No deja claro si rechaza, pospone o ignora.

Datos visibles, datos ocultos y privacidad

La tarjeta de revisión debe mostrar lo suficiente para decidir, pero no todo lo que el sistema sabe. Minimizar datos también es ingeniería.

TIPO DE DATO	MOSTRAR POR DEFECTO	MOTIVO
Acción propuesta	Sí.	Sin esto no hay decisión.
Efecto técnico	Sí.	La persona debe saber qué cambia.
Fuentes usadas	Sí, si tiene permiso.	Permite comprobar evidencia.
Datos personales completos	No.	Mostrar solo campos necesarios o redactados.
Prompt completo	Normalmente no.	Puede contener instrucciones internas o datos no necesarios.
Traza completa	No en tarjeta; sí enlazada.	La tarjeta debe ser legible y la auditoría completa debe existir.
Coste y latencia	Sí.	Afecta decisión operativa.
Identidad de otras personas	Solo si es necesario.	Reduce exposición innecesaria.

Campos recomendados:

```
visible_to_reviewer:
  - proposed_action
  - effect
  - draft
  - sources
  - missing_evidence
  - reason_codes
  - cost_eur
  - deadline_at
redacted_by_default:
  - raw_prompt
  - full_trace_payload
  - credentials
  - unrelated_personal_data
  - provider_internal_metadata
```

Una revisión útil no necesita enseñar todo. Necesita enseñar lo necesario, con enlaces auditables para quien tenga permiso.

RACI operativo: quién puede decidir qué

La revisión humana se vuelve frágil cuando cualquiera puede aprobar cualquier cosa. Un RACI ayuda a separar responsabilidad.

CASO	RESPONSIBLE	ACCOUNTABLE	CONSULTED	INFORMED
Respuesta de soporte común	Soporte nivel 1	Responsable de soporte	Producto	Usuario final
Caso con norma ambigua	Soporte nivel 2	Responsable académico	Legal o coordinación	Usuario final
Cambio de configuración IA	Plataforma IA	Lead técnico	Producto y soporte	Equipo
Tool con efecto externo	Equipo dueño del sistema	Owner del proceso	Plataforma IA	Auditoría
Incumplimiento de SLO	Guardia operativa	Responsable de operación	Soporte	Producto

Traducción práctica:

ROL	PUEDE APROBAR	NO DEBERÍA APROBAR
Soporte nivel 1	Respuestas con fuente y efecto bajo.	Cambios persistentes o casos ambiguos.
Soporte nivel 2	Casos con expediente, excepción o criterio.	Cambios de infraestructura.
Plataforma IA	Reanudación técnica, rutas, políticas y gates.	Interpretaciones de negocio sin owner.
Producto	Cambios de experiencia y criterio funcional.	Ejecución técnica sin evidencia.
Guardia operativa	Acciones para proteger SLO y cola.	Decisiones de contenido especializado.

Handoffs en SDKs y runtimes

La forma concreta cambia según la herramienta, pero el patrón se repite.

ECOSISTEMA	MECANISMO RELEVANTE	CÓMO LO TRADUCIRÍA A OPERACIÓN
OpenAI Agents SDK	Tools con aprobación, interrupciones, `RunState` serializable y reanudación.	Guardar pending approvals como cola y reanudar desde estado versionado.
OpenAI Agents SDK	Handoffs entre agentes.	Separar especialización de revisión: no todo handoff es humano.
Claude Agent SDK	Permisos, allow/deny, modos y `canUseTool`.	Declarar qué se permite, qué se bloquea y qué pide decisión en runtime.
Claude Agent SDK	Hooks.	Interceptar tool calls para añadir política propia, logs y gates.
Google ADK	Callbacks.	Insertar controles antes/después de agente, modelo o tool.
LangGraph	Interrupts y checkpoints.	Pausar grafo, guardar estado y reanudar con decisión.
OpenTelemetry	Trazas y spans.	Unir revisión, run, tool, coste y decisión en una historia consultable.

OpenTelemetry define las trazas como una forma de seguir el camino de una petición mediante spans relacionados.[14] En un handoff, la traza debería permitir responder: qué agente iba actuando, qué tool se propuso, qué política bloqueó, quién decidió y cómo se reanudó.

Calidad de la revisión

La revisión humana también se evalúa. Una persona puede tener prisa, interpretar una rúbrica de forma distinta o aprobar casos porque la interfaz no explica bien el efecto.

CONTROL	QUÉ DETECTA	CÓMO SE APLICA
Doble revisión muestral	Inconsistencia entre personas.	5% de casos van a dos revisores.
Rúbrica versionada	Cambios de criterio.	`review_rubric@4`.
Casos calibrados	Deriva del equipo.	Casos fijos con respuesta esperada.
Revisión de reversals	Decisiones corregidas después.	Analizar por motivo y persona.
Tiempo por decisión	Decisiones demasiado rápidas o lentas.	Mirar distribución, no solo media.
Feedback al sistema	Mismo motivo aparece muchas veces.	Abrir trabajo de prompt, RAG o contrato.

El acuerdo simple se calcula así:

$$A = \frac{N_{same}}{N_{double}}$$

SÍMBOLO	SIGNIFICADO	EJEMPLO
A	Tasa de acuerdo simple.	0,86.
N_{same}	Casos donde dos revisores tomaron la misma decisión.	86.
N_{double}	Casos revisados por dos personas.	100.

No es una métrica perfecta, pero ayuda a detectar si la cola está midiendo el criterio del sistema o el humor del día.

Los trabajos sobre model cards y datasheets enseñan una idea que aquí encaja muy bien: documentar estructura, límites, datos, supuestos y uso previsto reduce ambigüedad operativa.[15][16] En revisión humana ocurre lo mismo: si la tarjeta no dice qué se está decidiendo, cada persona rellena huecos con su interpretación.

Qué te llevas para poner en práctica

Al terminar este capítulo, el lector debería poder montar una cola mínima de revisión para un sistema de IA.

ARTEFACTO	PARA QUÉ SIRVE EN UN PROYECTO REAL	QUÉ DEBERÍA CONTENER
`handoff_request.schema.json`	Contrato de entrada a la cola.	IDs, acción, efecto, evidencia, motivos, deadline y opciones.
`handoff_policy.yaml`	Reglas de revisión.	Umbrales de confianza, coste, efectos y evidencia obligatoria.
`handoff_queue.py`	Simulador operativo.	Prioridad, decisión, SLA y mensaje de reanudación.
`handoff_state_machine.md`	Estados y transiciones.	Qué estados existen, quién los cambia y qué eventos emiten.
`handoff_tables.sql`	Persistencia mínima.	Requests, decisiones, eventos y asignaciones.
`review_decision.md`	Registro humano.	Decisión, motivo, cambios y caso de regresión.
`review_dashboard.sql`	Consultas de cola.	Backlog, aging, breaches y tasa de aprobación.
`privacy_review.yaml`	Minimización de datos.	Campos visibles, redactados y enlazados.
`raci_handoff.md`	Ownership operativo.	Quién puede aprobar cada tipo de caso.
`runbook_handoff.md`	Procedimiento del equipo.	Quién revisa, cuándo, cómo escalar y cómo cerrar.

Práctica recomendada:

1. Define dos acciones: una que pueda seguir sola y otra que requiera revisión.
2. Escribe tres motivos de revisión con códigos estables.
3. Diseña el contrato JSON.
4. Ejecuta el script de cola con casos de ejemplo.
5. Genera una decisión `approve`, una `reject` y una `needs_more_info`.
6. Añade una clave de idempotencia para que una aprobación no duplique efectos.

7. Define qué ocurre si la cola supera el SLO.
8. Convierte al menos un caso revisado en entrada de regresión.

El resultado práctico no es “tenemos humanos en el bucle”. Es poder demostrar qué entra, por qué entra, quién decide, cuánto tarda y cómo se reanuda.

Reto de capítulo: una cola de revisión en 45 minutos

Este es el ejercicio que me gustaría que alguien pudiera llevarse del capítulo y hacer de verdad. No es leer el concepto: es montar una mini cola reproducible.

Escenario: tienes un asistente de soporte que prepara respuestas. Algunas pueden guardarse como nota interna; otras quieren enviar un mensaje externo. Tu trabajo es decidir qué sigue solo, qué va a revisión y qué decisión deja preparada la reanudación.

Archivos que vas a crear:

```
mi-proyecto/
  ops/
    ai/
      handoff_queue.py
  data/
    handoff_examples.jsonl
  output/
    handoff_queue_result.json
```

Datos de entrada en `data/handoff_examples.jsonl`:

```
{"request_id":"rev_001","run_id":"run_7f31","trace_id":"trace_abc","task":"support_reply","p
roposed_action":"send_email","effect":"external_message","confidence":0.68,"cost_eur":0.018,
"created_at":"2026-05-28T13:45:00+00:00","sla_minutes":30,"evidence":{"draft":"Tu solicitud
queda pendiente de revisar el expediente actualizado.","sources":
["doc_matricula_2026"],"schema_valid":true,"contract_version":"support_reply_schema@4"},"mis
sing_evidence":["expediente_actualizado"],"resume_token":"resume_run_7f31_step_12"}
{"request_id":"rev_002","run_id":"run_8a20","trace_id":"trace_def","task":"ticket_summary","
proposed_action":"save_summary","effect":"internal_note","confidence":0.91,"cost_eur":0.006,
"created_at":"2026-05-28T14:05:00+00:00","sla_minutes":240,"evidence":{"draft":"Resumen
interno del caso con próximos pasos.","sources":
["ticket_991"],"schema_valid":true,"contract_version":"summary_schema@2"},"missing_evidence"
:[],"resume_token":"resume_run_8a20_step_05"}
{"request_id":"rev_003","run_id":"run_9c10","trace_id":"trace_xyz","task":"policy_answer","p
roposed_action":"answer_user","effect":"external_message","confidence":0.74,"cost_eur":0.041
,"created_at":"2026-05-28T13:20:00+00:00","sla_minutes":60,"evidence":{"draft":"Según la
política vigente, el plazo ordinario termina el viernes.","sources":
[],"schema_valid":true,"contract_version":"policy_answer_schema@7"},"missing_evidence":
["fuente_normativa"],"resume_token":"resume_run_9c10_step_09"}
```

Comandos:

```
mkdir -p ops/ai data output
python ops/ai/handoff_queue.py --write
cat output/handoff_queue_result.json
```

Qué debería ocurrir:

CASO	RESULTADO ESPERADO	POR QUÉ
`rev_001`	Entra en cola.	Baja confianza, efecto externo y falta expediente.
`rev_002`	Sigue automáticamente.	Es nota interna, tiene evidencia y supera umbrales.
`rev_003`	Entra primero en cola.	Efecto externo, coste alto y falta fuente normativa.

La entrega mínima del alumno:

1. Captura o salida de `handoff_queue_result.json`.
2. Una decisión escrita para `rev_003`.
3. Una frase explicando por qué `rev_002` no necesita revisión.
4. Una modificación de la política: subir o bajar `min_confidence` y justificar el efecto.
5. Un caso nuevo añadido al JSONL que pruebe `contract_failed`.

En el día a día

El handoff operativo aparece en cuanto un sistema de IA hace algo que un humano debe poder revisar, aprobar o retomar. La escena habitual: el agente prepara una respuesta, una acción o una propuesta, pero antes de ejecutarla con efecto se la pasa a una persona. Lo que parece un detalle se convierte en ingeniería en cuanto te preguntas lo concreto: ¿qué le llega exactamente a quien revisa? ¿Tiene el contexto, la evidencia y el diff para decidir en treinta segundos, o tiene que reconstruir media historia? ¿Qué pasa si nadie lo revisa en una hora, en un día? Un handoff sin cola, sin prioridad y sin caducidad no es supervisión humana, es un cuello de botella esperando a saturarse.

La segunda situación es la del relevo entre turnos o entre personas. Un sistema operable no para de funcionar porque cambie el turno; lo que cambia es quién atiende la cola de revisiones. Para que ese relevo no pierda casos, el handoff necesita estado: qué está pendiente, qué se aprobó, qué se rechazó y por qué. Sin ese estado, cada relevo es una pequeña amnesia, y las amnesias en operación se pagan con casos que se quedan colgados.

Por qué debería importarte

Porque la supervisión humana es la red de seguridad de casi todo sistema de IA con efecto real, y una red mal tejida da una falsa sensación de seguridad. Diseñar el handoff (qué información viaja, con qué prioridad, con qué caducidad y a quién escala si nadie responde) es lo que hace que «hay un humano en el bucle» signifique de verdad que alguien puede y va a revisar a tiempo, y no que hay un buzón donde las cosas se acumulan hasta que alguien se queja.

Y hay una razón que se entiende el día de una auditoría o de un incidente. Un handoff bien diseñado deja rastro: quién aprobó qué, con qué evidencia y cuándo. Ese rastro es lo que te permite responder con calma cuando alguien pregunta por qué el sistema hizo algo, en vez de encogerte de hombros. La revisión humana sin trazabilidad es casi tan frágil como no tener revisión.

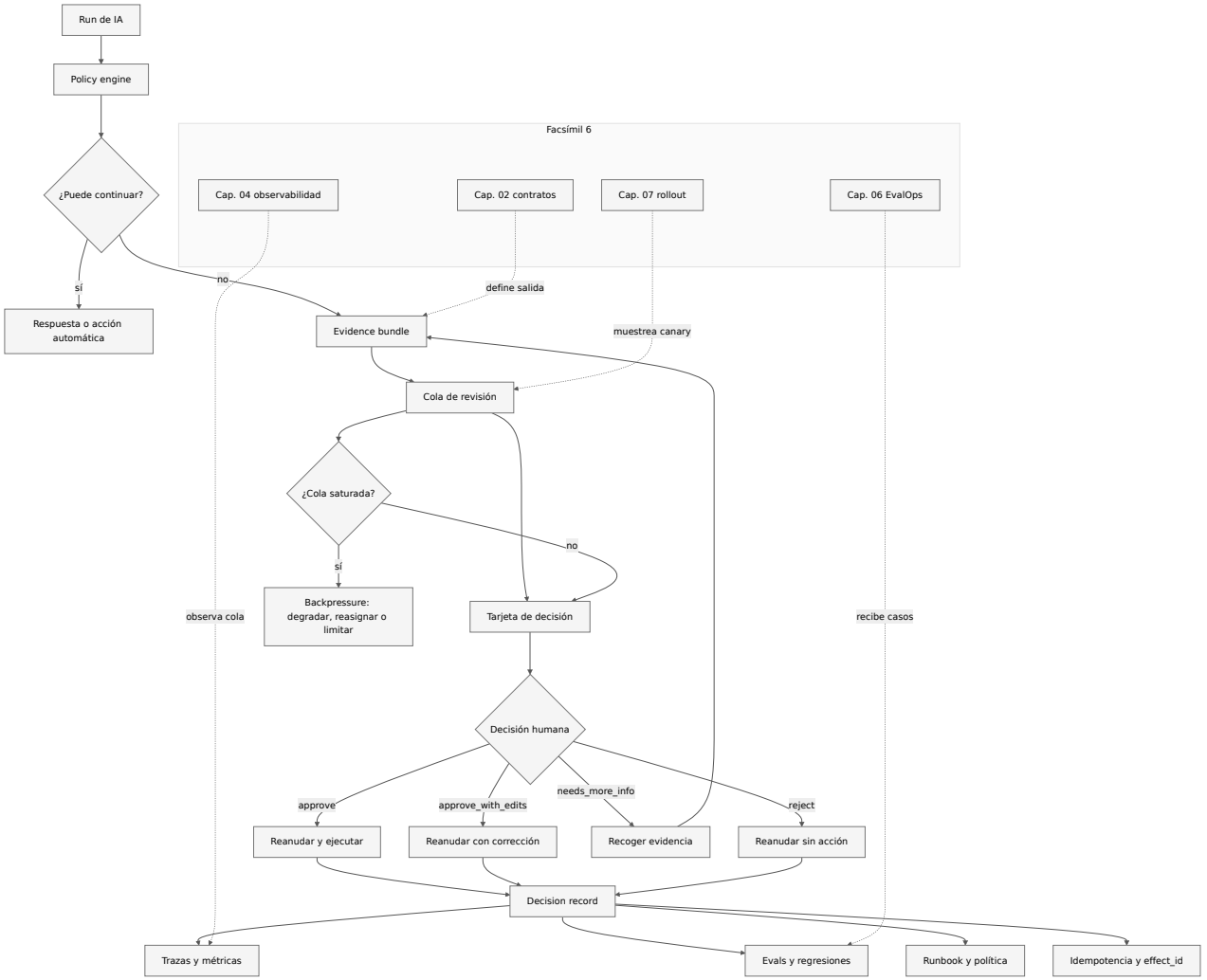
Dónde solía tropezar yo

Me costó entender que revisión humana no es una red de seguridad automática. Si llega tarde, sin contexto o sin criterio, solo traslada el problema a otra persona.

TROPIEZO	ANTÍDOTO
Mandar demasiadas cosas a revisión.	Codificar motivos y medir tasa por motivo.
Mandar casos sin evidencia suficiente.	Exigir `evidence` y `missing_evidence`.
No medir la cola.	Crear SLI/SLO de espera, backlog y breaches.
No reanudar bien.	Guardar `resume_token`, decisión y siguiente estado.
No aprender de revisiones.	Convertir casos en evals, reglas y runbooks.

La frase que me habría ahorrado muchas vueltas: **un handoff no es un mensaje; es un contrato de continuidad**.

Cómo encaja todo



Relación con otros capítulos

CAPÍTULO	QUÉ APORTA AQUÍ
F5 · Capítulo 08	Permisos, autonomía y aprobación de tools.
F6 · Capítulo 02	Estados `needs_review`, contratos de API y respuesta estructurada.
F6 · Capítulo 04	Trazas, métricas, SLIs y SLOs.
F6 · Capítulo 06	Casos revisados que vuelven a datasets y gates.
F6 · Capítulo 07	Canary y revisión muestreada antes de ampliar exposición.

Amershi y colaboradores mostraron que los sistemas de ML en producción traen retos propios de datos, operación, monitorización y evolución.[17] El NIST AI RMF organiza la gestión de sistemas de IA alrededor de gobernar, mapear, medir y gestionar.[18] Un handoff operativo es una forma concreta de llevar esas ideas al día a día: no basta con confiar en el sistema; hay que diseñar cómo se detiene, cómo pide criterio y cómo aprende.

Para entenderlo

Tres situaciones concretas:

SITUACIÓN	MALA SALIDA	HANDOFF OPERATIVO
Respuesta de soporte sin fuente.	“Creo que el plazo es el viernes”.	Pausa, pide fuente normativa y deja `needs_more_info`.
Tool quiere cerrar un ticket.	Cierra porque el resumen parece correcto.	Crea tarjeta con acción, evidencia, efecto y botón `approve`.
Canary de prompt nuevo.	Se mira el promedio y se sube al 50%.	Muestrea revisiones de casos nuevos y añade regresiones.

La diferencia no está en “meter una persona”. Está en que la persona entre justo donde el sistema necesita criterio, y salga dejando datos que mejoran el siguiente ciclo.

Vocabulario aprendido

TÉRMINO	DEFINICIÓN BREVE
Handoff operativo	Pausa estructurada de una run para que otra persona o sistema pueda decidir.
Evidence bundle	Paquete de entrada, propuesta, fuentes, trazas, motivos y estado.
Approval request	Solicitud concreta de aprobación sobre una acción.
Review queue	Cola priorizada de casos pendientes de decisión.
Decision record	Registro de qué se decidió, quién lo hizo y por qué.
Resume token	Identificador o estado que permite reanudar una run.
SLO de revisión	Objetivo medible de tiempo o calidad de la cola.
Agreement rate	Proporción de decisiones iguales entre revisores en muestra doble.

Antes de pasar página

Comprueba que puedes responder:

1. ¿Qué diferencia hay entre revisión humana y handoff operativo?
2. ¿Qué campos mínimos debe incluir un `evidence_bundle`?
3. ¿Cuándo pedirías revisión por efecto persistente?
4. ¿Qué SLI usarías para medir una cola de revisión?
5. ¿Por qué `approve` y `approve_with_edits` no deberían ser el mismo botón?
6. ¿Cómo se reanuda una run después de una decisión?
7. ¿Qué caso revisado añadirías a un dataset de regresión?

En resumen

IDEA	PARA LLEVARTE
Revisar no es improvisar.	La revisión debe tener contrato, evidencia, SLA y reanudación.
La cola se opera.	Backlog, p95, breaches, acuerdo y coste humano son métricas del sistema.
La interfaz decide calidad.	Si la tarjeta no muestra efecto y evidencia, la persona decide a ciegas.
Cada revisión debe enseñar algo.	Los casos vuelven a evals, prompts, políticas y runbooks.

Para saber más

- Amershi, S. y otros *Software Engineering for Machine Learning: A Case Study*. https://doi.org/10.1109/ICSE-SEIP.2019.00042
- Anthropic. *Claude Agent SDK: Handle Approvals and User Input*. https://code.claude.com/docs/en/agent-sdk/user-input
- Anthropic. *Claude Agent SDK: Hooks*. https://code.claude.com/docs/en/agent-sdk/hooks
- Anthropic. *Claude Agent SDK: Permissions*. https://code.claude.com/docs/en/agent-sdk/permissions
- Gebru, T. y otros *Datasheets for Datasets*. https://doi.org/10.1145/3458723
- Google. *Callbacks: Observe, Customize, and Control Agent Behavior*. https://adk.dev/callbacks/
- Google. *Safety and Security for AI Agents*. https://adk.dev/safety/
- LangChain. *LangGraph Interrupts*. https://docs.langchain.com/oss/python/langgraph/human-in-the-loop
- Mitchell, M. y otros *Model Cards for Model Reporting*. https://doi.org/10.1145/3287560.3287596
- OpenAI. *Agents SDK: Handoffs*. https://openai.github.io/openai-agents-python/handoffs/
- OpenAI. *Agents SDK: Human-in-the-loop*. https://openai.github.io/openai-agents-python/human_in_the_loop/
- OpenTelemetry. *Traces*. https://opentelemetry.io/docs/concepts/signals/traces/
- Tabassi, E. *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. https://doi.org/10.6028/NIST.AI.100-1

Notas

1. OpenAI. (2026). *Agents SDK: Human-in-the-loop*. https://openai.github.io/openai-agents-python/human_in_the_loop/. Consultado el 28 de mayo de 2026. ↩

2. OpenAI. (2026). *Agents SDK: Handoffs*. https://openai.github.io/openai-agents-python/handoffs/. Consultado el 28 de mayo de 2026. ↩

3. Anthropic. (2026). *Claude Agent SDK: Permissions*. https://code.claude.com/docs/en/agent-sdk/permissions. Consultado el 28 de mayo de 2026. ↩

4. Anthropic. (2026). *Claude Agent SDK: Hooks*. https://code.claude.com/docs/en/agent-sdk/hooks. Consultado el 28 de mayo de 2026. ↩

5. Anthropic. (2026). *Claude Agent SDK: Handle Approvals and User Input*. https://code.claude.com/docs/en/agent-sdk/user-input. Consultado el 28 de mayo de 2026. ↩

6. Google. (2026). *Callbacks: Observe, Customize, and Control Agent Behavior*. https://adk.dev/callbacks/. Consultado el 28 de mayo de 2026. ↩

7. Google. (2026). *Safety and Security for AI Agents*. https://adk.dev/safety/. Consultado el 28 de mayo de 2026. ↩

8. LangChain. (2026). *LangGraph Interrupts*. https://docs.langchain.com/oss/python/langgraph/human-in-the-loop. Consultado el 28 de mayo de 2026. ↩

9. Elkan, C. (2001). The Foundations of Cost-Sensitive Learning. *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI)*, 973-978. Define el umbral de decisión óptimo comparando el coste esperado de cada acción. ↩

.0. Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1), 37-46. https://doi.org/10.1177/001316446002000104. Define kappa como acuerdo corregido por azar. ↩

.1. Geifman, Y., & El-Yaniv, R. (2017). Selective Classification for Deep Neural Networks. *Advances in Neural Information Processing Systems*, 30. https://papers.nips.cc/paper/2017/hash/4a8423d5e91fda00bb7e46540e2b0cf1-Abstract.html. Formaliza la clasificación selectiva mediante la curva riesgo-cobertura. ↩

.2. Brier, G. W. (1950). Verification of Forecasts Expressed in Terms of Probability. *Monthly Weather Review*, 78(1), 1-3. Define la puntuación cuadrática para pronósticos probabilísticos. ↩

.3. Little, J. D. C. (1961). A Proof for the Queuing Formula: L = λW. *Operations Research*, 9(3), 383-387. https://doi.org/10.1287/opre.9.3.383. La relación se aplica a cualquier sistema estable, incluida una cola de revisión humana. ↩

.4. OpenTelemetry. (2026). *Traces*. https://opentelemetry.io/docs/concepts/signals/traces/. Consultado el 27 de mayo de 2026. ↩

.5. Mitchell, M. y otros (2019). *Model Cards for Model Reporting*. Proceedings of the Conference on Fairness, Accountability, and Transparency, 220-229. https://doi.org/10.1145/3287560.3287596 ↩

.6. Gebru, T. y otros (2021). *Datasheets for Datasets*. *Communications of the ACM*, 64(12), 86-92. https://doi.org/10.1145/3458723 ↩

.7. Amershi, S. y otros (2019). *Software Engineering for Machine Learning: A Case Study*. Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice, 291-300. https://doi.org/10.1109/ICSE-SEIP.2019.00042 ↩

.8. Tabassi, E. (2023). *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. NIST AI 100-1. https://doi.org/10.6028/NIST.AI.100-1 ↩