Sistema de Control de Calidad de Workflows

Rúbrica de auditoría, canal de lanzamiento y lista de verificación intensiva — listo para Gumroad · 14 de julio de 2026

Canal de lanzamiento para Gumroad

Lleva un workflow desde el “borrador” hasta “listo para subir a Gumroad” a través de una compuerta de lanzamiento que nada atraviesa con defectos conocidos.

Una nota honesta sobre el “100 % sin errores”: ningún proceso puede garantizar que un workflow sea perfecto — lo que este canal garantiza es que **nada se publica con un error detectado**. Combina verificaciones mecánicas deterministas (que detectan defectos mecánicos con certeza) con una auditoría por rúbrica y una verificación independiente (que detectan defectos de criterio), y la compuerta solo se abre cuando las tres vuelven limpias. Un workflow que no puede dejarse limpio se devuelve al propietario con exactamente lo que lo bloquea — nunca se deja pasar.

Vista general del canal

```
Intake → Preflight → Auditoría → Corrección ⟲ (hasta quedar limpio)
→ Verificación independiente → Empaquetado → Registro de
lanzamiento
```

Ejecuta cada etapa por workflow. Al lanzar varios workflows, pasa cada uno por el canal completo de forma independiente — la aprobación de un workflow nunca cubre a otro.

Etapa 1 — Intake (recepción)

Identifica el conjunto completo de archivos del workflow (SKILL.md o la definición del workflow, `references/`, `scripts/`, `assets/`). Confirma con el propietario: nombre del producto, nivel de precio previsto (los productos gratuitos en Gumroad están limitados a 250 MB; los archivos de pago pueden ser mucho más grandes) y los términos de licencia. Haz una copia instantánea de los archivos antes de tocar nada, para que cada cambio del bucle de corrección sea comparable con un diff.

Etapa 2 — Preflight (verificaciones deterministas)

Ejecuta el verificador incluido:

```
python3 scripts/preflight.py <ruta-a-la-carpeta-del-workflow>
```

Verifica por máquina lo que un revisor humano pasa por alto: que el frontmatter se pueda analizar y tenga `name` y `description`, que cada archivo mencionado en las instrucciones exista realmente en el paquete, que no haya residuos personales (nombres de usuario del autor, correos electrónicos, rutas `/Users/<nombre>`), que no haya marcadores TODO/FIXME/placeholder, que no haya enlaces locales rotos y que el paquete se comprima en zip sin errores. Código de salida 0 = aprobado; cualquier otro imprime los fallos exactos. **Un fallo de preflight es siempre un hallazgo Crítico.** Corrige y vuelve a ejecutar hasta que salga 0 — este es el piso del “sin errores”, verificado por código y no por criterio.

Etapa 3 — Lista de verificación intensiva + auditoría por rúbrica

Recorre `references/release-checklist.md` punto por punto — alrededor de 90 propiedades binarias que abarcan metadatos, estructura, instrucciones, entradas, fiabilidad, salidas, código incluido, seguridad, portabilidad y empaquetado. Cada punto es aprobado, fallido o N/A-con-motivo; no existe el “casi”. Los puntos marcados [auto] ya los cubre el preflight; el resto se verifica leyendo y ejecutando el workflow.

En paralelo, puntúa el workflow con `references/rubric.md` (8 dimensiones ponderadas, con evidencia obligatoria por cada deducción) — la lista de verificación encuentra los defectos, la rúbrica los dimensiona. Si el skill complementario `workflow-quality-audit` está instalado, usa también sus recorridos por personas y su plantilla de informe. Resultado: la lista de verificación completada más una lista de hallazgos con severidades y correcciones concretas.

Etapa 4 — Bucle de corrección

Aplica las correcciones, primero las más severas. Después de cada ronda, vuelve a ejecutar **tanto** el preflight como la auditoría — las correcciones introducen defectos nuevos con frecuencia, y una aprobación obsoleta no vale nada.

Condición de salida: el preflight sale con 0, **todos** los puntos de la lista de verificación pasan (o son N/A con motivo declarado), **y** la auditoría reporta **cero hallazgos Críticos y cero Mayores**. Los Menores se registran en el registro de lanzamiento pero no bloquean la publicación.

Si tres rondas no convergen, detente y escala al propietario con los hallazgos restantes — un workflow que resiste tres rondas de corrección suele tener un problema de diseño que el propietario debe decidir, y seguir insistiendo desperdicia esfuerzo en la capa equivocada.

Etapa 5 — Verificación independiente

Quien corrige no debe calificar su propio trabajo. Haz que un agente nuevo (un subagente sin memoria del bucle de corrección) vuelva a ejecutar el preflight y la auditoría por rúbrica desde cero. La compuerta solo se abre si este pase independiente también encuentra cero hallazgos Críticos/Mayores. Si encuentra algo que el bucle de corrección pasó por alto, vuelve a la Etapa 4 — y anota cuál fue el descuido, porque suele revelar un punto ciego que vale la pena añadir a la rúbrica.

Etapa 6 — Empaquetado para Gumroad

Lee `references/gumroad-packaging.md` para las especificaciones vigentes y arma el paquete:

1. **Zip del producto** — la carpeta del workflow, empaquetada (un archivo `.skill` para skills de Cowork/Claude, más un `.zip` normal de respaldo, ya que los compradores pueden no saber qué es un archivo `.skill`).
2. **README del comprador, inglés + español** — instalación, primer uso, requisitos, soporte. Plantilla en `references/listing-template.md`. Escribe ambos para alguien que nunca ha usado un skill de Claude. La versión en español (`01-README.es.md`) debe reflejar la inglesa sección por sección; mantén sin traducir los comandos, el código y los nombres de archivo (la sección L de la lista de verificación tiene todas las reglas de localización).
3. **LICENSE.txt** — los términos que el propietario eligió en el intake.
4. **Texto del listado** — título, subtítulo, descripción, viñetas, FAQ. Plantilla en `references/listing-template.md`.
5. **Especificación de la imagen de portada** — PNG/JPEG de 1280×720 (16:9), con el contenido clave centrado para que el recorte cuadrado de la miniatura de Gumroad no lo corte. Genera la imagen si te lo piden, o entrega la especificación al diseñador del propietario.

Etapa 7 — Registro de lanzamiento

Escribe `release-record.md` en la carpeta superior del paquete: nombre y versión del workflow, fecha, puntuaciones finales de la rúbrica, salida del preflight, confirmación de la verificación independiente, SHA-256 del zip del producto y cualquier hallazgo Menor aceptado. Esta es la

prueba del control de calidad — es lo que distingue “lo verificamos” de “creemos que lo verificamos”, y es contra lo que se compara cuando salga la siguiente versión.

Lista de entregables

Cada ejecución completada entrega al propietario exactamente cinco cosas:

- [ ] `<nombre>.skill` + `<nombre>.zip` (el producto)
- [ ] `01-README.md` + `01-README.es.md` para compradores (dentro del zip)
- [ ] `LICENSE.txt` (dentro del zip)
- [ ] `listing-copy.md` (+ `listing-copy.es.md` si se vende a compradores hispanohablantes)
- [ ] `release-record.md` (la prueba de QC, incluido el recuento de localización de la sección L)

Si alguna casilla queda sin marcar, la ejecución no está terminada.

Lista de verificación intensiva de lanzamiento

Cada propiedad de abajo es binaria: pasa o falla. Un workflow está listo para lanzarse solo cuando **todos** los puntos pasan o están marcados explícitamente como N/A con un motivo. Recórrela de arriba abajo durante las Etapas 3–4; el verificador independiente (Etapa 5) vuelve a comprobar cada punto marcado como aprobado.

Los puntos etiquetados **[auto]** los verifica `scripts/preflight.py`; el resto se comprueba leyendo y ejecutando el workflow.

A. Metadatos y frontmatter

- [ ] A1 [auto] SKILL.md existe en la raíz del paquete
- [ ] A2 [auto] El frontmatter se analiza correctamente (bloque `---` válido)
- [ ] A3 [auto] `name` presente, no vacío, en kebab-case, coincide con el nombre de la carpeta
- [ ] A4 [auto] `description` presente, de 15 a 1500 caracteres
- [ ] A5 La description dice QUÉ hace el workflow en la primera frase
- [ ] A6 La description dice CUÁNDO usarlo (contextos, no solo capacidad)
- [ ] A7 La description incluye las frases literales que los usuarios realmente dicen
- [ ] A8 La description lo distingue de workflows adyacentes (“no para X — usa Y”)
- [ ] A9 Toda la información de cuándo-usarlo vive en la description, nada solo en el cuerpo
- [ ] A10 El nombre está tan orientado al resultado que un comprador puede adivinar qué hace

B. Estructura e integridad de archivos

- [ ] B1 [auto] Todo archivo referenciado en cualquier .md existe en el paquete
- [ ] B2 [auto] No hay archivos de cero bytes
- [ ] B3 [auto] No hay stubs de expulsión `.icloud`
- [ ] B4 [auto] El paquete se comprime en zip y el zip se verifica limpio
- [ ] B5 El cuerpo de SKILL.md tiene menos de ~500 líneas; el exceso va a `references/`
- [ ] B6 Cada archivo de referencia se señala desde SKILL.md con guía de cuándo leerlo
- [ ] B7 Los archivos de referencia de más de 300 líneas tienen índice
- [ ] B8 No hay archivos huérfanos (presentes en el paquete pero nunca mencionados ni usados)
- [ ] B9 Los nombres de carpeta siguen la anatomía estándar (`scripts/`, `references/`, `assets/`)
- [ ] B10 No hay basura de compilación (.DS_Store, **pycache**, .git, archivos swap del editor)

C. Calidad de las instrucciones

- [ ] C1 Las instrucciones son imperativas (“Ejecuta X”), no divagación descriptiva
- [ ] C2 Los pasos están numerados donde el orden importa
- [ ] C3 Cada término que un novato no conocería se define en su primer uso
- [ ] C4 Las instrucciones no obvias explican el PORQUÉ (la razón viaja mejor que la regla)
- [ ] C5 No hay contradicciones entre ninguna pareja de secciones
- [ ] C6 Ninguna instrucción depende de contexto que solo existe en la cabeza del autor
- [ ] C7 Hay ejemplos para todo lo sensible al formato
- [ ] C8 Un lector primerizo podría ejecutar de principio a fin sin hacer una pregunta
- [ ] C9 La configuración de una sola vez está claramente separada del uso cotidiano
- [ ] C10 Las expectativas del primer uso se establecen donde el

comportamiento difiere del régimen normal

D. Entradas y prerrequisitos

- [ ] D1 Cada entrada requerida está enumerada
- [ ] D2 Para cada entrada: dónde la encuentra el usuario si no la conoce
- [ ] D3 Requisitos del entorno declarados (SO, herramientas, cuentas, niveles de plan)
- [ ] D4 Credenciales requeridas listadas, con la forma segura de suministrar cada una
- [ ] D5 Comportamiento definido cuando falta una entrada o tiene formato incorrecto
- [ ] D6 Sección “Cuándo NO usar esto” presente y precisa
- [ ] D7 No se asumen ubicaciones de carpetas sin un paso de descubrimiento

E. Fiabilidad y manejo de errores

- [ ] E1 Modos de fallo conocidos documentados como síntoma → causa → solución
- [ ] E2 Las operaciones destructivas son flags de aceptación explícita, nunca predeterminadas
- [ ] E3 Ejecutar dos veces seguidas es seguro (idempotente o advertido explícitamente)
- [ ] E4 Una interrupción a mitad de ejecución deja un estado recuperable (o el riesgo está documentado)
- [ ] E5 Los fallos de dependencias de red/servicio tienen guía
- [ ] E6 Un paso de verificación explícito confirma el éxito después de la acción principal
- [ ] E7 El error más probable del usuario está previsto y manejado
- [ ] E8 Los mensajes de error que verá el usuario están citados para que pueda reconocerlos

F. Definición de la salida

- [ ] F1 El entregable está nombrado: formato, estructura, ubicación
- [ ] F2 Se proporcionan plantillas para cualquier salida estructurada
- [ ] F3 Se incluye al menos un ejemplo de salida correcta donde el formato importa
- [ ] F4 Al usuario se le dice de antemano qué va a recibir
- [ ] F5 La salida de éxito y la de advertencia son distinguibles por el usuario

G. Código incluido (`scripts/`) — omite la sección si no hay scripts

- [ ] G1 Cada script corre en una máquina limpia (solo stdlib, o dependencias declaradas)
- [ ] G2 Los scripts se ejecutan sin error con entrada de muestra válida
- [ ] G3 Los scripts salen con código distinto de cero al fallar (sin éxito silencioso)
- [ ] G4 Texto de uso/ayuda presente (`--help` o docstring del módulo)
- [ ] G5 Sin rutas, hosts, puertos ni credenciales codificados en duro
- [ ] G6 La configuración usa un archivo de ejemplo (`*.example.json`), nunca uno real relleno
- [ ] G7 Cada script que la documentación menciona está incluido; cada script incluido está mencionado

H. Seguridad, privacidad y confianza

- [ ] H1 [auto] Sin direcciones de correo en ninguna parte del paquete
- [ ] H2 [auto] Sin rutas `/Users/<nombre>` ni otras rutas personales
- [ ] H3 [auto] Sin el nombre del autor ni palabras bloqueadas (ejecuta preflight con `--blocklist`)
- [ ] H4 Sin claves de API, tokens, contraseñas ni hostnames reales — tampoco en los ejemplos
- [ ] H5 La guía de secretos usa indirección por variable de entorno/llavero, nunca pegar-en-el-chat
- [ ] H6 Se instruye proteger con permisos los archivos con credenciales

(chmod 600)
- [ ] H7 Cualquier dato que salga de la máquina del usuario se señala explícitamente
- [ ] H8 El workflow no hace nada que su description no diga (sin sorpresas)
- [ ] H9 Las acciones destructivas o irreversibles requieren confirmación explícita del usuario

I. Portabilidad y generalidad

- [ ] I1 Cero contexto específico del autor en los archivos publicados (nombres, hábitos, anécdotas)
- [ ] I2 Funciona para todo el rango de entradas que la description promete, no solo el caso de demostración
- [ ] I3 Los ejemplos son ilustrativos, no estructurales
- [ ] I4 Los comandos específicos de un SO están señalados como tales, con alternativas donde sea razonable
- [ ] I5 [auto] Sin marcadores TODO/FIXME/PLACEHOLDER/TBD
- [ ] I6 Nada referencia otros archivos privados, sesiones o memoria del autor

J. Empaquetado para el comprador (Gumroad)

- [ ] J1 El zip del producto contiene: README del comprador, archivo .skill, LICENSE, copia skill-source
- [ ] J2 El README del comprador cubre instalación, primer uso, requisitos y contacto de soporte
- [ ] J3 README escrito para alguien que nunca ha instalado un skill de Claude
- [ ] J4 LICENSE.txt presente y coincide con lo que promete el listado
- [ ] J5 Archivos nombrados para ordenarse con sentido (prefijos 01-, 02-, 03-)
- [ ] J6 Tamaño total menor de 250 MB si el producto puede obtenerse por $0
- [ ] J7 Imagen de portada de 1280×720, contenido clave dentro del cuadrado central, <1 MB
- [ ] J8 La descripción del listado incluye requisitos explícitos (evita reseñas por reembolso)
- [ ] J9 El listado no promete nada que el workflow no haga — audita las afirmaciones contra A5–A8
- [ ] J10 Canal de soporte y expectativa de respuesta declarados

L. Documentación en español (localización)

Los documentos de cara al comprador se publican en inglés **y** en español. Las instrucciones internas del workflow (el cuerpo de SKILL.md) permanecen solo en inglés salvo que el propietario decida otra cosa — esas las lee Claude; estas las leen los compradores.

- [ ] L1 Existe el README del comprador en español (`01-README.es.md`) junto al inglés
- [ ] L2 Paridad sección por sección: cada encabezado, paso y advertencia del README inglés aparece en el español — sin contenido eliminado en silencio
- [ ] L3 La traducción es español natural, no salida automática palabra por palabra (léela en voz alta: ¿la escribiría así un hablante nativo?)
- [ ] L4 Los comandos, el código, los nombres de archivo y las etiquetas de botones NO se traducen — solo la prosa a su alrededor (los comandos traducidos rompen el copiar-pegar)
- [ ] L5 Las etiquetas de interfaz que el comprador debe pulsar se dan tal como aparecen en la interfaz real del producto, con la glosa en español entre paréntesis donde ayude
- [ ] L6 Sección de solución de problemas traducida, con los mensajes de error citados mantenidos textualmente en su idioma original para que el comprador pueda reconocerlos
- [ ] L7 Texto del listado en español preparado (`listing-copy.es.md`) si el propietario vende a compradores hispanohablantes en el mismo

listado o en uno segundo
- [ ] L8 La sección de soporte indica en qué idiomas se responde — nunca implicar que existe soporte en español si no existe
- [ ] L9 Números, fechas y moneda con formato del locale (estilo 1.234,56 €) en la prosa — pero nunca dentro de código o configuración
- [ ] L10 Paridad re-verificada después de cada ronda del bucle de corrección que tocó los documentos ingleses: una edición en inglés sin la edición española correspondiente es un hallazgo **Mayor**
- [ ] L11 [auto] El preflight ejecutado con `--require-spanish` sale con 0 (comprueba que cada documento de cara al comprador tenga su contraparte `.es.md` no vacía y no idéntica)

K. Compuerta final

- [ ] K1 preflight.py sale con 0 sobre el paquete final
- [ ] K2 La auditoría completa por rúbrica muestra cero hallazgos Críticos y cero Mayores
- [ ] K3 El verificador independiente reconfirmó cada sección anterior desde cero
- [ ] K4 release-record.md escrito (puntuaciones, checksums, fecha, Menores aceptados)
- [ ] K5 El zip exacto que se sube es el zip exacto auditado (coincidencia SHA-256)

Regla de conteo: el registro de lanzamiento debe declarar el recuento, p. ej. “95 verificados / 3 N/A (sin scripts) / 0 fallando”. Cualquier punto fallando = la compuerta permanece cerrada. No existe el “aprobado en su mayoría”.

La sección L solo puede marcarse N/A en su conjunto, con la decisión explícita y registrada del propietario de publicar solo en inglés — nunca omitirse en silencio.

Rúbrica de calidad de workflows

Puntúa cada dimensión de 1 a 5. Registra evidencia (una cita o una carencia nombrada) por cada puntuación inferior a 5. Total ponderado = Σ(puntuación × peso) / Σ(pesos).

Los pesos reflejan lo que realmente impulsa las valoraciones de los usuarios: un workflow que nunca se activa o que falla en silencio recibe una reseña de 1 estrella por muy elegante que sea su prosa.

#	Dimensión	Peso
1	Activación	3
2	Claridad	2
3	Completitud	3
4	Fiabilidad	3
5	Definición de la salida	2
6	Eficiencia	1
7	Seguridad y confianza	2
8	Generalidad	2

1. Activación (peso 3)

La description del frontmatter es lo *único* que Claude ve antes de decidir usar el workflow. Si no se activa, nada más importa.

Comprueba:

- ¿La description dice tanto *qué hace* COMO *cuándo usarlo*?
- ¿Lista las frases/contextos concretos que los usuarios realmente dicen (“sube mi sitio”, “carga estos archivos”), no solo la capacidad abstracta (“sincronización de archivos”)?
- ¿Es apropiadamente “insistente”? Claude se queda corto activando por defecto; las descriptions deben decirle que se active incluso cuando el usuario no nombra el workflow.
- ¿Se desambiguan los casos límite (cuándo debe ganar un workflow *parecido*)?
- ¿Toda la información de cuándo-usarlo está en la description y no enterrada en el cuerpo?

5 = la description cubre qué + cuándo + frases del usuario + desambiguación. **3** = lo que hace está claro pero los contextos de activación son escasos o viven en el cuerpo. **1** = description de solo capacidad; Claude rara vez lo invocará sin que se lo pidan.

2. Claridad (peso 2)

Comprueba:

- Instrucciones imperativas (“Ejecuta X, luego comprueba Y”), no descripción pasiva.
- Cada término que un novato no conocería se explica en su primer uso o se enlaza a una referencia.
- Los pasos están ordenados y numerados donde el orden importa.
- El *porqué* de las instrucciones no obvias está explicado (tanto los modelos como los humanos siguen mejor las instrucciones cuando entienden la razón).
- Sin contradicciones entre secciones.

5 = un lector primerizo puede ejecutar sin una sola pregunta de seguimiento. **3** = ejecutable pero exige adivinar en 1 o 2 puntos. **1** = orden ambiguo, jerga sin explicar o guía contradictoria.

3. Completitud (peso 3)

Comprueba:

- Cada entrada requerida enumerada (qué pedirle al usuario, y dónde

encuentra cada dato si no lo conoce).
- Prerrequisitos y configuración única separados del uso cotidiano.
- Casos límite cubiertos: entrada vacía/enorme/malformada, archivos faltantes, primera ejecución frente a repetida, dependencias sin conexión.
- Sección “Cuándo NO usar esto” presente — el mal uso es una de las principales fuentes de malas valoraciones.
- Cada archivo, script o recurso que las instrucciones mencionan está realmente incluido o su ubicación está especificada con precisión.

5 = entradas, configuración, límites y anti-casos cubiertos; sin referencias colgantes. **3** = camino feliz completo; límites escasos. **1** = asume contexto que el usuario no tiene; referencia archivos inexistentes.

4. Fiabilidad (peso 3)

Comprueba:

- Modos de fallo conocidos listados como síntoma → causa → solución (una tabla de solución de problemas es el estándar de oro).
- Las operaciones destructivas son de aceptación explícita, nunca predeterminadas.
- Idempotencia: ¿qué pasa si se ejecuta dos veces? ¿Y si se interrumpe a medias?
- Las dependencias externas (red, credenciales, servicios) tienen guía de fallo.
- Paso de verificación: ¿cómo *confirma* el usuario que funcionó?

5 = solución de problemas presente, valores predeterminados no destructivos, paso de verificación definido. **3** = algo de guía de fallos pero quedan rutas de fallo silencioso. **1** = sin manejo de fallos; los errores dejarán varado al usuario.

5. Definición de la salida (peso 2)

Comprueba:

- El entregable está fijado: formato, estructura, ubicación, nomenclatura.
- Plantillas proporcionadas para salidas estructuradas (esqueletos de informe, ejemplos de configuración).
- Ejemplos de buena salida incluidos donde el formato importa.
- Al usuario se le dice qué recibirá antes de que empiece el trabajo (fijar expectativas es la mitad de una buena valoración).

5 = plantilla/ejemplo exacto de la salida presente. **3** = salida descrita en prosa pero formato dejado al azar. **1** = salida sin especificar en absoluto; cada ejecución produce algo distinto.

6. Eficiencia (peso 1)

Comprueba:

- SKILL.md por debajo de ~500 líneas; el detalle empujado a `references/` con indicaciones claras de cuándo leer cada archivo.
- El trabajo repetitivo/determinista capturado en `scripts/` incluidos en lugar de regenerarse en cada ejecución.
- Sin peso muerto: cada sección se gana su lugar.
- Los archivos de referencia grandes (>300 líneas) tienen índice.

5 = cuerpo esbelto, divulgación progresiva, scripts incluidos donde procede. **3** = funciona pero está inflado o pierde oportunidades obvias de incluir scripts. **1** = muro monolítico de texto; tokens desperdiciados en cada invocación.

7. Seguridad y confianza (peso 2)

Comprueba:

- Los secretos nunca se pegan en el chat ni se guardan en texto plano cuando hay disponible un patrón de variable de entorno o

llavero; permisos de archivo restringidos donde corresponda.
- Sin comportamiento sorprendente: el workflow hace lo que dice su description, nada más.
- Los datos que salen de la máquina del usuario (subidas, llamadas a API) se señalan explícitamente.
- Las acciones destructivas o irreversibles requieren confirmación explícita.

5 = higiene de credenciales, flujos de datos explícitos, acciones destructivas confirmadas. **3** = mayormente seguro pero con una práctica floja (p. ej., configuración en texto plano sugerida primero). **1** = maneja secretos con descuido o realiza acciones destructivas/de red no reveladas.

8. Generalidad (peso 2)

Un workflow que se vende o comparte debe funcionar para *cualquier* comprador, no solo para su autor.

Comprueba:

- Sin nombres codificados, rutas personales ni contexto específico del autor en las instrucciones (las notas personales pertenecen a la memoria/preferencias del propio usuario, no a un workflow distribuido).
- Los ejemplos son ilustrativos, no estructurales — el workflow generaliza más allá de ellos.
- Suposiciones del entorno declaradas (SO, herramientas, tipos de cuenta) en lugar de silenciosas.
- Comportamiento razonable en todo el *rango* de entradas que la description promete.

5 = totalmente portable; un desconocido podría ejecutarlo hoy. **3** = generaliza con ediciones menores; algunos residuos específicos del autor. **1** = solo funciona en la configuración exacta del autor.

Definiciones de severidad para los hallazgos

- **Crítico** — causará ejecuciones fallidas o falta de activación para un usuario típico (activación rota, archivo incluido faltante, valor predeterminado destructivo silencioso, contexto personal codificado en un producto vendido).
- **Mayor** — el usuario completa la tarea pero con fricción real o salida degradada (falta solución de problemas para un fallo común, formato de salida sin definir, prerrequisito sin explicar).
- **Menor** — pulido (tono, orden, redundancia, formato).

Un solo hallazgo Crítico limita el veredicto general a “Corregir mayores” sin importar la puntuación.

Especificaciones de empaquetado para Gumroad

Verificadas contra el centro de ayuda de Gumroad, julio de 2026. Si un listado se comporta de forma inesperada, vuelve a consultar gumroad.com/help — las especificaciones cambian.

Archivos

- Productos gratuitos ($0 / $0+): límite de **250 MB** en total. Los productos de pago permiten archivos mucho más grandes (varios GB); los paquetes de workflows de Claude son diminutos, así que esto solo importa si incluyes vídeos explicativos.
- Sube el producto como **zip**. Incluye tanto el archivo `.skill` como una copia descomprimida de la carpeta — los compradores que no conocen los archivos `.skill` aún pueden inspeccionar el contenido.
- Gumroad ofrece a los compradores una experiencia de “Download all”; nombra los archivos para que se ordenen con sentido: `01-README.md`, `02-<nombre>.skill`, `03-LICENSE.txt`.

Imagen de portada

- **1280 × 720 px** (16:9), PNG o JPEG, idealmente por debajo de 1 MB.
- Gumroad recorta un **cuadrado centrado** para las miniaturas — mantén el nombre del producto y el arte clave dentro del cuadrado central o quedarán cortados en las vistas de navegación.
- Banner del perfil (si también actualizas la tienda): 1600 × 400 px.

Qué va dentro del zip del producto

```
<nombre-del-producto>/
├── 01-README.md          ← para el comprador: instalación, primer uso, requisitos, soporte
├── 01-README.es.md       ← lo mismo, en español (la sección L de la lista rige su calidad)
├── 02-<nombre>.skill     ← el paquete de skill instalable
├── 03-LICENSE.txt        ← términos de uso
└── skill-source/         ← copia descomprimida de la carpeta del skill (la transparencia vende)
```

Verifica la localización mecánicamente antes de publicar:

```
python3 scripts/preflight.py <carpeta-del-paquete> --require-spanish
```

Campos del listado a preparar (listos para pegar en listing-copy.md)

- **Nombre** — orientado al resultado, no al mecanismo (“Despliega tu sitio en Hostinger con un comando”, no “Herramienta de sincronización FTPS”).
- **Descripción** — problema → qué hace → qué incluye → requisitos → política de reembolso/soporte. A los compradores de productos de workflows los queman a menudo; los requisitos explícitos (“necesita Claude con soporte de skills; macOS/Linux”) evitan los reembolsos y las reseñas de 1 estrella que los listados vagos se ganan.
- **Precio** — recuerda el límite de 250 MB del nivel gratuito si usas precio $0+.
- **FAQ / soporte** — dónde contactan los compradores al propietario; expectativa de respuesta.

Plantillas: README del comprador y texto del listado de Gumroad

Plantilla del README del comprador (va dentro del zip como 01-README.md)

```
# [Nombre del producto]

¡Gracias por tu compra! Esto es un skill de Claude — un workflow empaquetado
que le enseña a Claude a [resultado en una línea].

## Qué necesitas

- [Requisito de plan/app de Claude, p. ej. "app de escritorio de Claude o Claude Code con soporte de skills"]
- [Requisitos de SO, si los hay]
- [Cuentas/credenciales que usa el workflow, si las hay]

## Instalación (2 minutos)

1. Descomprime esta descarga.
2. [Pasos exactos de instalación para la plataforma del comprador — pulsar
   Save skill / copiar la carpeta al directorio de skills / etc.]
3. Confirma que está instalado: pregúntale a Claude "[una frase que debería activarlo]".

## Primer uso

[Guía al comprador hasta su primer uso exitoso, incluida la configuración de
una sola vez si la hay. Termina con cómo se ve el éxito.]

## Solución de problemas

[Los 3 problemas principales y sus soluciones. Enlaza a la sección de solución
de problemas del propio workflow para el resto.]

## Soporte

Preguntas o problemas: [canal de contacto]. Suelo responder en [expectativa].
```

Plantilla del texto del listado de Gumroad (listing-copy.md, pegar en Gumroad)

```
# Título
[El resultado que obtiene el comprador, ≤60 caracteres — "…con un comando", "…sin los errores"]

# Subtítulo / frase única
[Para quién es + el dolor que elimina]

# Descripción

**El problema:** [2-3 frases que el comprador objetivo reconoce al instante]

**Qué hace esto:** [3-4 frases, concretas. El mecanismo solo después del resultado.]

**Qué incluye:**
- [El skill en sí]
- [README del comprador con guía de instalación y primer uso]
- [Scripts/plantillas incluidos, si los hay]
- [Resumen de los términos de licencia]

**Requisitos:** [Todos los requisitos, explícitos. Esta línea evita reembolsos
y reseñas de 1 estrella — no la suavices.]

**FAQ**
- *¿Funciona con [cosa adyacente común]?* — [respuesta honesta]
- *¿Y si no me funciona?* — [política de reembolso/soporte]
```

```
        # Etiquetas
        [3-6 términos de búsqueda del marketplace que los compradores
escribirían]
```

Versiones en español (01-README.es.md, listing-copy.es.md)

Usa las mismas plantillas de arriba, traducidas — no adaptadas con soltura. Reglas que importan (lista completa en `references/release-checklist.md`, sección L):

- Refleja la estructura inglesa encabezado por encabezado; un comprador que compare ambas debe encontrar cada advertencia y cada paso en las dos.
- Nunca traduzcas comandos, código, nombres de archivo ni lo que el comprador deba escribir o pulsar — solo la prosa a su alrededor. Un comando traducido rompe el copiar-pegar.
- Mantén los mensajes de error citados textualmente en su idioma original (eso es lo que mostrará la pantalla del comprador), con la explicación en español después.
- Indica con claridad en qué idiomas responde el soporte.
- Escribe español natural. Si una frase se lee como una frase inglesa con las palabras sustituidas, reescríbela.

Reglas de redacción para ambos

- Las afirmaciones deben coincidir con la auditoría: todo lo prometido aquí debe corresponder a un punto aprobado de la lista de verificación. El listado es parte de la superficie de QC — un listado sobrevendido convierte un producto aprobado en una reseña de 1 estrella.
- Escribe los requisitos también como exclusiones (“no es para X”) — que el comprador equivocado pida reembolso es peor que no vender.
- Mantén el README del comprador autosuficiente: asume que el comprador nunca contacta a soporte y nunca vuelve a leer el listado después de comprar.

Plantilla del informe de auditoría

Usa esta estructura exacta para cada informe por workflow. Un informe por workflow, un archivo por informe, llamado `audit-<nombre-del-workflow>.md`.

Auditoría de calidad: [nombre del workflow]

Auditado: [fecha] · **Versión auditada:** [hash del archivo, fecha o versión si se conoce] **Puntuación ponderada:** X.X / 5 · **Veredicto:** Publicar / Corregir menores / Corregir mayores / Rehacer

Tabla de puntuación

Dimensión	Peso	Puntuación	Motivo en una línea
Activación	3		
Claridad	2		
Completitud	3		
Fiabilidad	3		
Definición de la salida	2		
Eficiencia	1		
Seguridad y confianza	2		
Generalidad	2		
Total ponderado		**X.X**	

Qué funciona bien

[2–4 viñetas. Fortalezas reales, no relleno — el propietario necesita saber qué NO tocar durante la corrección.]

Hallazgos

[Ordenados por severidad, luego por dimensión. De 5 a 12 hallazgos es el rango sano.]

F1 · [Crítico/Mayor/Menor] · [Dimensión] — [título en una línea]

Evidencia: [cita la línea infractora, o nombra la carencia específica] **Por qué daña las valoraciones:** [la consecuencia visible para el usuario, una frase] **Solución:** [acción concreta — texto de reemplazo, sección a añadir, archivo a incluir. Escrita para que el propietario pueda aplicarla en menos de una hora sin preguntas de seguimiento.]

F2 · …

Victorias rápidas

[Las 2–3 correcciones con mejor impacto-en-valoraciones-por-minuto, extraídas de los hallazgos de arriba. Esto es lo que el propietario debería hacer hoy.]

Disparador de re-auditoría

[Cuándo volver a ejecutar esta auditoría: tras aplicar correcciones Críticas/Mayores, antes del siguiente lanzamiento, o con cadencia si el workflow cambia a menudo.]

Plantilla del resumen de cartera (solo auditorías de varios workflows)

Cartera de workflows: resumen de calidad

Auditado: [fecha] · **Workflows:** N

Workflow	Puntuación	Veredicto	Corrección principal

Patrones sistémicos

[Problemas que aparecen en 2 o más workflows — son problemas de proceso. Arregla el proceso (p. ej., una lista de verificación de autoría) una vez, no cada workflow por separado.]

Orden de trabajo sugerido

[Qué workflows corregir primero, sopesando la severidad frente a cuánto se usa cada uno o cuán visiblemente está valorado.]