Jano.

Probar una v1 con Claude antes de mostrársela al cliente

Introducción

Con esta guía probás un agente o un workflow nuevo antes de que lo vea tu cliente. Primero le escribís vos, como lo haría un cliente, y Claude verifica cada paso en la ejecución. Después Claude hace de cliente y corre todos los casos cada vez que cambiás algo. Todo se suma a la carpeta **n8n-con-claude** del video 10.

Importante

Las pruebas corren de verdad: mandan mensajes, usan tu crédito de OpenAI y, si el workflow guarda, reserva o manda algo, lo hace. Probá con tu propio chat o con un contacto que se llame "Prueba", y borrá lo que quede al terminar.

Sección 1: El orden

1. **El caño.** Mandale "hola": el mensaje tiene que llegar, el agente responder y la respuesta salir. Si esto falla, no sigas.
2. **El pase.** Vos le escribís desde el canal y Claude verifica cada paso en la ejecución (Sección 3).
3. **Leerlo como el cliente.** Dónde se habría ido un cliente real. Separá lo que falló técnicamente de lo que salió mal para el cliente (Sección 4).
4. **Arreglar y repetir el paso.** Si se puede resolver en un nodo, se resuelve en un nodo y no en el prompt. Después repetís ese paso ("3 bis").
5. **Claude hace de cliente.** Los casos corren solos, todos, después de cada cambio (Secciones 5 a 7).
6. **Alguien más lo prueba.** Alguien del equipo, o alguien que no lo armó. Después, el cliente, con el bot de prueba y una guía corta (Sección 8).
7. **Cada falla, un caso fijo.** Lo que falló en una prueba, lo que reportó el cliente o un error de producción se suma como caso. Cada cambio, todos los casos.
8. **En producción, un barrido diario** de las ejecuciones con error (Sección 8).

Sección 2: Preparar la v1 de la clase

El recurso **asistente-v1.zip** trae **cotizar.json** (el subflujo que busca el precio en la planilla, lo pasa a la moneda que piden y aplica el descuento), **asistente-telegram.json** (el agente que conversa y usa Cotizar como herramienta) y **lista-de-precios.csv**, por si todavía no tenés la planilla del video 10.

1. **El bot.** En Telegram buscá **BotFather** y mandale /newbot. Te pide un nombre y un usuario que termine en "bot". Te responde con el token: tratalo como una contraseña y borrá ese mensaje del chat. Es el mismo bot que usás en el video 14.

2. **Las credenciales.** En n8n, **Create credential**: una de **Telegram API**, con el token del bot, y una de **OpenAI**, con una API key de platform.openai.com (necesita crédito cargado). La de Google es la del video 10.
3. **La planilla.** Una planilla de Google llamada "Ferretería El Tornillo" con una hoja "Lista de precios" que tenga las columnas producto, precio_usd y stock. Podés importar lista-de-precios.csv.
4. **Cotizar.** Creá un workflow nuevo, menú de los tres puntos → Import → From file → cotizar.json. En **Buscar producto** elegí tu credencial de Google, la planilla y la hoja Lista de precios. Guardá y tocá **Publish**.
5. **El asistente.** En otro workflow importá asistente-telegram.json. En el nodo **cotizar**, en Workflow, elegí **Cotizar** de la lista. Revisá que el disparador y el nodo que responde tengan la credencial del bot, y el modelo la de OpenAI. Tocá **Publish**. Cotizar va publicado antes: si no, la herramienta falla con "Workflow is not active and cannot be executed".
6. **El primer mensaje.** Abrí tu bot en Telegram y mandale "hola".

Sección 3: El pase

En la app de Claude, pestaña **Code**, abrí una sesión nueva en n8n-con-claude y pedile (cambiá lo que pidió tu cliente):

> Terminé la v1 de [workflow]. El cliente pidió [sus reglas]. Le voy a escribir al bot como si fuera un cliente. Cuando te diga "listo", buscá la ejecución por la hora y verificá qué herramienta usó, con qué datos y si lo que respondió cumple lo que pidió el cliente. No cambies nada.

Después le escribís al bot, por ejemplo *¿cuánto me salen 3 candados en dólares?*, y cuando llega la respuesta le escribís "listo" a Claude. Si querés una prueba más larga, pedile que te arme el pase: una tabla con lo que le escribís vos, lo que verifica él y la regla del cliente que queda probada.

1. Escribile "listo" **cuando te llegue la respuesta**. Si mira la ejecución antes de que termine, no ve la herramienta y te dice que falló.
2. Claude busca la ejecución por la hora y te dice pasa o falla, con la evidencia.
3. Una ejecución en verde no es un OK: mirá el resultado. Y en una v1, un error adentro de una verde puede ser lo esperado, como la herramienta que no encuentra un producto que no existe. En producción conviene que la herramienta devuelva "no encontrado" como respuesta y no como error.

Sección 4: Leerlo como el cliente

Cuando termines el pase, pedile:

> Leé toda la conversación como si fueras el cliente: ¿en qué mensaje habrías dejado de contestar o te habrías ido enojado? Separá lo que falló técnicamente de lo que salió mal para el cliente.

1. **Falla técnica:** no respondió, tardó, una herramienta dio error, el mensaje no salió.

2. **Lo que salió mal para el cliente:** todo corrió, pero le llegó algo equivocado. Puede faltar una regla en el flujo (le cotizó algo sin stock porque el stock nunca llegó al agente) o el agente puede decidir mal (inventó un dato, prometió algo que no puede cumplir, preguntó de más). No aparece en rojo en ningún lado: se ve leyendo.

Sección 5: El módulo de pruebas

1. Descomprimí **modulo-pruebas.zip**. Trae **probar-n8n** (la skill, con el programa que manda los mensajes) y **casos** (los casos del asistente y, como ejemplo de un webhook, los del Cotizador del video 10).
2. Arrastrá **casos** adentro de n8n-con-claude.
3. Arrastrá **probar-n8n** adentro de n8n-con-claude/.claude/skills. La carpeta .claude está oculta: en el Finder, Cmd + Shift + punto la muestra; en Windows, Explorador → Vista → Mostrar → Elementos ocultos. Más simple: abrí una sesión de Claude en n8n-con-claude y pedile "mové la carpeta probar-n8n adentro de .claude/skills".
4. Así tiene que quedar:

```
n8n-con-claude/
  .claude/skills/probar-n8n/
  casos/agente-telegram.md y cotizador.md
  CLAUDE.md
  conectar.py
  n8n.py
```

5. **Empezá por un webhook común.** Pedile "probá el Cotizador": Claude le manda cada pedido al webhook de producción, como un cliente, y después lee la ejecución por la API de n8n. La API no corre workflows: sirve para leer las ejecuciones y subir arreglos. Por eso las pruebas entran por el disparador.
6. **Telegram también es un webhook.** Al publicar, n8n le da a Telegram una dirección de tu n8n y un encabezado secreto, y Telegram le manda ahí cada mensaje. Claude manda el mismo paquete a esa dirección y n8n lo recibe por el disparador real.
7. Abrí una sesión nueva en la carpeta. Para un bot de Telegram, mandale "hola" al bot y pedile a Claude: "averiguá mi chat de Telegram y guardalo".

Sección 6: Cómo se escribe un caso

Un archivo por workflow, en **casos**. Arriba, el nombre del workflow y el disparador. Después, un caso por bloque: un título, lo que se manda y lo que tiene que pasar, en español.

````
- workflow: Asistente Telegram
- disparador: telegram

## Caso 8: Sin stock

```mensajes
¿cuánto me salen 3 candados en dólares?
```

Esperado: dice que no hay stock y no da un precio como si hubiera.
````

1. Los casos salen de lo que pidió el cliente (una regla, un caso, y uno justo en el borde), de los pasos del pase y de cada falla real.
2. Para un agente, escribí qué herramienta tiene que usar, con qué datos y qué no puede hacer. La redacción de la respuesta no se compara.
3. En Telegram todos los casos usan tu chat, y el agente recuerda lo que se habló antes. Si un caso falla por eso, corrélo solo.

Si no tenés los casos escritos

No hace falta un archivo de casos para arrancar. Pegale a Claude este pedido, con lo que sabés del cliente (lo que hace el agente o el flujo, sus reglas y lo que salió en la llamada):

```
Quiero probar [workflow] antes de mostrárselo al cliente. Esto es lo que tiene que hacer: [qué hace, para quién, con qué herramientas]. Las reglas del cliente: [reglas].
1. Armá los casos en casos/[workflow].md con el formato de la skill probar-n8n:
   - el camino feliz: lo que la mayoría de los clientes va a pedir;
   - cada regla del cliente: un caso que la cumple y uno justo en el borde;
   - los casos borde: lo que no está en la lista, lo que está fuera de tema, datos que faltan, dos mensajes seguidos, alguien que vuelve días después;
   - para un agente: qué herramienta tiene que usar, con qué datos y qué no puede hacer.
2. Mostrame la lista antes de correrla.
3. Con mi OK, corré todos los casos con la skill probar-n8n y armá la tabla con OK o FALLA, el nodo que falló y qué cambiarías.
[Rápido: 8 a 10 casos, los más importantes. Exhaustivo: todos los que encuentres.]
```

Sección 7: Pedidos que sirven

```
probá el [workflow]
sumá un caso: [lo que se manda]. Esperado: [lo que tiene que pasar]
mostrame qué le mandaste en el caso 2 y cómo leíste la ejecución
corré de nuevo los que fallaron y después todos
arreglalo y corré todo
armame la guía corta para que el cliente pruebe el bot
armame el mensaje para el cliente con lo que cambió, y antes revisá que cada punto esté así en producción
```

Cuando un caso falla, Claude te dice en qué nodo y qué cambiaría; con tu OK lo cambia (antes guarda una copia en **backups**) y vuelve a probar. Cada corrida deja su reporte en **reportes**.

Sección 8: Antes del cliente y en producción

1. **Alguien más lo prueba** antes que el cliente, aunque lleve dos horas.
2. **El cliente prueba en Telegram**, aunque el canal final sea otro: le pasás el usuario del bot y la guía corta que arma Claude (qué hace y qué todavía no, qué escribirle, qué tendría que pasar y qué reportar: con una captura del chat alcanza).
3. **Su pedido de cambio** lleva un caso nuevo, y antes de avisarle corren todos.
4. **Con su OK**, lo conectás a su canal y lo prendés.

5. **Todos los días**, pedile a Claude:

   > revisá las ejecuciones con error de las últimas 24 horas de [workflows]: qué pasó, qué corregirías y cómo verificás que el arreglo no rompa otra cosa

   Lo que se corrige va con su caso nuevo, y antes de subirlo corren todos. Para que la revisión corra sola, programala como en la hoja del video 10.

Sección 9: Si algo falla

1. **El bot no responde y no hay ejecución:** el workflow no está publicado, o otro workflow con el mismo bot se publicó después o quedó escuchando en el editor. Telegram le manda los mensajes de cada bot a un solo lugar: despublicá y volvé a publicar el asistente.
2. **401:** la API key está mal o vencida. Creá otra y corré python3 conectar.py.
3. **Sigue fallando después de corregir a mano:** te faltó Publish. Lo que cambia Claude por la API en un workflow publicado queda publicado.
4. **Cambiaste algo y desapareció un arreglo de Claude:** tocaste una pestaña vieja. Recargá la página antes de tocar un workflow que cambió Claude.

Clases filtradas de n8n · Jano Llosa · @one.jano · jano.my. Usalo libre en tus proyectos y con tus clientes; si lo compartís, nombrá la fuente.