AGENTE DE WHATSAPP CON IA

WhatsApp *AgentKit*

Tu agente de WhatsApp con IA, armado por Claude Code mientras respondés preguntas sobre tu negocio.

INSTALACIÓN
5 minutos

AGENTE FUNCIONANDO EN TU COMPU
20–30 minutos

Antes de empezar

Qué es

WhatsApp AgentKit no es una plantilla para adaptar a mano. Es un set de instrucciones que Claude Code lee para **entrevistarte sobre tu negocio** y después escribir, probar y dejar listo para publicar el agente completo: el servidor, la conexión con WhatsApp, la memoria de cada cliente y el prompt que le da personalidad.

Vos no escribís código. Respondés preguntas.

REPO github.com/Hainrixz/whatsapp-agentkit

Qué necesitás tener a mano

	PARA QUÉ	DÓNDE SE CONSIGUE
Claude Code	Es quien construye todo	claude.com/claude-code
API key de Anthropic	Es el "cerebro" del agente cuando ya está andando. Empieza con `sk-ant-`	platform.anthropic.com → Settings → API Keys
Cuenta de Zernio recomendado	Conecta el agente con WhatsApp sin crear una app de Facebook	zernio.com — plan gratis, sin tarjeta
Info de tu negocio	Menú, lista de precios, FAQ, políticas, horarios	PDF, TXT, DOCX, CSV, JSON, Markdown o imágenes

NOTA

¿No tenés número de WhatsApp Business?

No hace falta para arrancar. Zernio tiene un número de pruebas compartido: 50 mensajes por día durante 7 días, gratis. Alcanza para ver el agente contestándote desde tu celular hoy mismo.

Cuánto cuesta de verdad

AgentKit es gratis (licencia MIT). Lo que se paga es el uso:

Zernio	2 cuentas conectadas gratis. Si necesitás que te den un número: entre USD 3 y 21 al mes según el país
Claude con el modelo por defecto, Sonnet	~USD 0,07 por conversación de 8 mensajes → ~USD 20 al mes con 300 conversaciones
Railway para tenerlo online 24/7	USD 5 de crédito de prueba, después USD 5 al mes

El costo de Claude crece con el tamaño de la información que le cargás. La forma más efectiva de bajarlo no es cambiar de modelo: es no meterle info que tus clientes nunca preguntan.

Instalación

REPO github.com/Hainrixz/whatsapp-agentkit

Abrí Claude Code en la carpeta donde quieras guardar el proyecto y pegá este prompt tal cual:

PROMPT · PEGAR EN CLAUDE CODE — copiar tal cual

```
Quiero instalar WhatsApp AgentKit desde https://github.com/Hainrixz/whatsapp-agentkit

1. Cloná el repo en una carpeta llamada whatsapp-agentkit dentro de esta carpeta.
2. Verificá que tengo Python 3.11 o superior. Ojo si estoy en Windows: "python3"
   puede ser un acceso directo falso de Microsoft Store que no hace nada. En ese
   caso usá "python" o "py" y decime qué versión encontraste. Si no tengo Python
   3.11+, decime exactamente cómo instalarlo y pará ahí.
3. Dentro de la carpeta del repo, creá la carpeta "knowledge" si no existe y
   copiá .env.example como .env (sin pisar un .env que ya exista).
4. No construyas el agente todavía. Cuando termines, decime en una línea cómo
   abrir Claude Code dentro de la carpeta whatsapp-agentkit para correr
   /build-agent.
```

Cuando termine, cerrá esa sesión y **abrí Claude Code dentro de la carpeta `whatsapp-agentkit`**. Esto importa: el comando `/build-agent` y las instrucciones que guían a Claude viven dentro de esa carpeta. Si abrís Claude Code en otro lado, el comando no aparece.

Ahí escribí: `/build-agent`

Construir el agente

`/build-agent` arranca un proceso de 5 fases. Claude te va haciendo **una pregunta por vez** y no avanza de fase sin tu confirmación.

FASE 1

Prepara el entorno

Revisa Python, crea las carpetas e instala las dependencias. No tenés que hacer nada; solo aprobar los comandos que te pida correr.

FASE 2

La entrevista (acá está todo el valor)

Son 10 preguntas. Lo que respondas acá es lo que el agente va a saber, así que vale la pena tomárselo en serio:

#	PREGUNTA	CÓMO RESPONDERLA BIEN
01	**Nombre del negocio**	Tal cual lo conocen tus clientes
02	**A qué se dedica**	Con detalle: qué vendés, qué servicios, quién te compra. Una línea genera un agente genérico
03	**Para qué querés el agente**	FAQ, agendar, calificar leads, tomar pedidos, soporte. Podés elegir varias
04	**Nombre del agente**	Lo que ven tus clientes: "Ana", "Soporte TuMarca"
05	**Tono**	Formal, casual, vendedor o cálido
06	**Horario de atención**	El agente avisa cuando le escriben fuera de horario
07	**Archivos del negocio**	Antes de contestar, tirá tus archivos en la carpeta `knowledge` y apretá Enter. Precios y horarios quedan literales en el agente
08	**API key de Anthropic**	Se guarda en el `.env`, que nunca se sube a GitHub
09	**Cómo conectar WhatsApp**	**Zernio** si querés el camino corto. Meta Cloud API solo si ya tenés una app de Facebook Business armada
10	**Credenciales del proveedor**	Con Zernio: tu API key (empieza con `sk_`) y un "webhook secret" que **inventás vos** (cualquier texto, ej. `mi-agente-2026`)

NOTA

¿Todavía no tenés cuenta de Zernio?

Decíselo a Claude en la pregunta 10 y te guía: crear cuenta → Connections → Connect new → WhatsApp → Settings → API Keys → Create. **Copiá la key apenas aparece: se muestra una sola vez.**

NOTA

¿Querés probar sin WhatsApp primero?

Podés dejar las credenciales de WhatsApp vacías y probar todo en la terminal. Las cargás después.

FASE 3

Lo construye

Con tus respuestas escribe el agente entero: el servidor, la conexión con Claude, la memoria de cada cliente, el adaptador de WhatsApp, la personalidad (`config/prompts.yaml`) y los archivos para publicarlo. Si cargaste archivos en `knowledge` , los lee y los incorpora.

Probarlo en tu compu

FASE 4

Chateás con tu agente

Claude abre un chat en la terminal donde **vos escribís como si fueras un cliente**:

```
Tu: Hola, qué horarios tienen?
Agente: Hola! Atendemos de lunes a viernes de 9 a 18.
        ¿Te ayudo con algo más?
```

Probalo como lo haría un cliente real, no como el dueño:

- Preguntá precios que **sí** cargaste y fijate que coincidan exacto.
- Preguntá algo que **no** cargaste. Tiene que decir que no tiene esa info y ofrecer pasarte con una persona, nunca inventar.
- Escribí con faltas, en una sola palabra, o enojado.
- Escribile fuera de horario.

Si algo no te gusta, decíselo a Claude en lenguaje normal: "está muy formal", "no ofrece el delivery", "responde demasiado largo". Lo ajusta y volvés a probar. No pasa a la siguiente fase hasta que vos digas que está bien.

Verlo en tu WhatsApp con el sandbox de Zernio

Si no tenés número propio, pedile a Claude:

PEDIDO A CLAUDE CODE

```
Activá el sandbox de Zernio para probar el agente con mi celular: +54 9 ...
```

Te llega un WhatsApp desde el número de pruebas. Lo respondés desde tu celular y quedás activado por 7 días, con 50 mensajes por día.

Ponerlo online

FASE 5

Railway

Para que conteste solo, las 24 horas, aunque tu compu esté apagada, hay que subirlo a un servidor. Claude te guía paso a paso; el recorrido es este:

1. **GitHub.** Creás un repo vacío en github.com/new y Claude sube tu agente ahí.
2. **Railway.** railway.app → New Project → Deploy from GitHub repo → elegís tu repo.
3. **Base de datos.** En el proyecto: New → Database → Add PostgreSQL. Es lo que hace que el agente recuerde a tus clientes.
4. **Variables.** Cargás tus keys en la pestaña Variables (Claude te dice cuáles y con qué valor).
5. **Dominio.** Tu servicio → Settings → Networking → Generate Domain. Te queda algo como `tu-agente.up.railway.app`.
6. **Webhook.** En Zernio → Webhooks → Create webhook: pegás `https://tu-agente.up.railway.app/webhook`, el mismo secret que inventaste, marcás `message.received` y usás "Send test" para confirmar.

Desde ahí, cualquiera que le escriba a tu número habla con tu agente.

NOTA

La regla de las 24 horas de WhatsApp

El agente responde a quien le escribe, así que siempre puede contestar. Si algún día querés que escriba **primero** (recordatorios, promos), WhatsApp exige plantillas aprobadas por Meta. Es una regla de WhatsApp, no de AgentKit.

Cambiarlo después

No hace falta tocar código. Abrí Claude Code en la carpeta del proyecto y pedile:

PEDIDOS A CLAUDE CODE

```
Agregamos servicio de delivery. Actualizá el agente.
```

```
El agente está muy formal. Hacelo más cercano.
```

```
Quiero migrar de Zernio a Meta Cloud API.
```

Lo que hay que saber antes de usarlo con clientes

LO QUE NOS PASÓ AL PROBARLO

En Windows, el arranque se corta sin explicar nada.

El script de inicio del repo busca `python3`, y en Windows ese nombre suele ser un acceso directo de Microsoft Store que no tiene Python adentro, aunque lo tengas instalado como `python`. El script muere en el paso 1. El prompt de instalación de esta guía ya le avisa a Claude Code para que use el Python correcto. Si igual te pasa, decile: *"python3 es el alias de la Store, usá python"*.

LO QUE ADVIERTE EL PROPIO REPO (Y CONVIENE NO SALTEAR)

Sin la base de datos, el agente se olvida de todos.

Si no agregás PostgreSQL en Railway y no cargás la variable `DATABASE_URL` en tu servicio, el agente arranca igual, pero guarda las conversaciones en un disco que se borra en cada actualización. Cada vez que cambies algo, pierde el historial de todos tus clientes.

El "webhook secret" no es opcional en producción.

Si lo dejás vacío, el agente funciona igual, pero acepta mensajes de cualquiera que conozca tu URL. Sirve para probar. Antes de atender clientes reales, cargalo.

Railway no te da la URL solo.

Hay que tocar "Generate Domain". Sin eso no hay dónde apuntar el webhook.

Habla de agendar, pero no agenda solo.

El agente conversa, responde con la info de tu negocio y toma los datos. Lo que todavía no hace de fábrica es *ejecutar* la acción del otro lado: escribir en tu calendario, descontar stock, cobrar. Ese es el siguiente paso y se lo pedís a Claude Code; no viene andando de la caja.

No subas a GitHub tu `.env`.

Ahí están tus keys. Claude lo deja excluido; no lo fuerces.

07 SIGUIENTE NIVEL

Lo que muestra el Reel

Hasta acá, tu agente responde solo. En el Reel hace tres cosas más: **pide tu visto bueno** antes de algo delicado, **anota a cada cliente** en una planilla y **hace seguimiento** si alguien deja de escribir. AgentKit no las trae de fábrica: se las pedís a Claude Code, una por vez, en la carpeta de tu agente.

Qué hace solo y qué no

RESPONDE SOLO

Lo que está en tu info

Precios, horarios, envíos, medios de pago y preguntas frecuentes.

AL CLIENTE

La respuesta, al instante.

PIDE APROBACIÓN

Lo que compromete plata

Descuentos, devoluciones, pagos, cambios de precio o cualquier promesa que no esté en tu info.

AL CLIENTE

"Lo consulto con el equipo y te confirmo."

ESCALA A UNA PERSONA

Lo que necesita a alguien

Enojo, insultos, amenaza de cancelar, problemas de pago o legales, o algo que no sabe.

AL CLIENTE

"Te paso con una persona del equipo." El agente se pausa.

Así te llega a vos

MP **Agente Mate Pampa**
bot

Pedido de aprobación
Cliente: 598 99 111 222
Escribió:

> Si compro 50 mates, ¿me hacen 20% de descuento?

Respuesta propuesta:

> Por 50 unidades te podemos hacer un 20%. ¿Te paso el total?

Motivo: pide 20%; la política es 10% desde 50 unidades.

Aprobar | Editar | Rechazar

Aprobar la manda tal cual.

Editar te pide el texto corregido y manda ese.

Rechazar no manda nada.

Ejemplo con un negocio ficticio.

07 SIGUIENTE NIVEL

Pedíselo a Claude Code

Abrí Claude Code en la carpeta de tu agente, la misma donde corriste `/build-agent`, y pegá los pedidos de a uno. Antes de pasar al siguiente, probá que el anterior funcione.

PASO 1

Aprobación por Telegram

Necesitás Telegram en el celular. El bot se crea gratis con @BotFather y Claude Code te guía.

PROMPT · PEGAR EN CLAUDE CODE — copiar tal cual

```
Mi agente de WhatsApp está hecho con AgentKit (https://github.com/Hainrixz/whatsapp-agentkit). Quiero que no mande nada delicado sin mi visto bueno:

1. Que Claude devuelva un JSON con la respuesta y una acción: "responder" (lo que está en la info del negocio), "pedir_aprobacion" (descuentos, devoluciones, pagos, cambios de precio o promesas que no estén en la info) o "escalar" (enojo, insultos, amenaza de cancelar, problema de pago o legal, o algo que no sabe). Si el JSON no se puede leer, va a aprobación.
2. pedir_aprobacion: al cliente le llega "Lo consulto con el equipo y te confirmo" y a mí, por Telegram, su mensaje, la respuesta propuesta y el motivo, con botones Aprobar, Editar y Rechazar.
3. escalar: al cliente le llega "Te paso con una persona del equipo"; a mí, un resumen, los últimos mensajes y una respuesta sugerida. El agente se pausa con ese cliente hasta que toque "Devolver al agente"; mientras, sus mensajes me llegan a Telegram.
4. Lo pendiente se guarda en la base de datos y solo mi chat puede aprobar. Cada envío posterior lleva su propia Idempotency-Key: si reusa la del mensaje original, Zernio no lo manda.
5. Guiame para crear el bot con @BotFather y sacar mi chat_id.

Probalo con mensajes simulados antes de terminar: una consulta, un descuento y un reclamo.
```

PASO 2

Planilla de clientes

Necesitás una cuenta de Google. La "cuenta de servicio" es un usuario técnico que escribe en tu planilla: Claude Code te guía para crearla.

PROMPT · PEGAR EN CLAUDE CODE — copiar tal cual

```
Sumale a mi agente de AgentKit (https://github.com/Hainrixz/whatsapp-agentkit) un registro de clientes en Google Sheets: una fila por cliente, buscada por teléfono, con nombre, teléfono, estado (nuevo, interesado, caliente, cliente, escalado), interés, resumen, fecha del último mensaje y próxima acción. Que Claude saque esos datos en el mismo JSON que ya devuelve. La base de datos es la fuente de verdad; la planilla es para que yo la mire. Si Sheets falla, el cliente igual recibe su respuesta.

Usá una cuenta de servicio con el JSON en una variable de entorno (en el .env, entre comillas simples) y guiame para crearla y compartirle la planilla. Probalo con una conversación simulada.
```

PASO 3

Seguimiento automático

Para el segundo seguimiento necesitás una plantilla aprobada por Meta. Se crea desde Zernio y Meta la revisa en hasta 24 h.

PROMPT · PEGAR EN CLAUDE CODE — copiar tal cual

```
Sumale seguimiento a mi agente de AgentKit (https://github.com/Hainrixz/whatsapp-agentkit). Si un cliente interesado o caliente deja de escribir, máximo 2 seguimientos:

1. Entre 18 y 23 h después de su último mensaje (dentro de la ventana de 24 h de WhatsApp): un texto corto escrito por Claude, solo entre las 9 y las 21 en mi zona horaria. Si no hay un momento así, se saltea.
2. A las 72 h, solo con una plantilla aprobada por Meta. Sin plantilla no se manda. Guiame para crearla.

No mandes nada si respondió, pidió que no le escriban o está escalado. Que corra en el mismo servidor cada 15 minutos, guarde lo enviado para no repetir y actualice la planilla. Sumá tzdata al requirements.txt. Probalo simulando el paso del tiempo.
```

07 SIGUIENTE NIVEL

Antes de activarlo

LO QUE NOS PASÓ AL PROBARLO · CON WHATSAPP, TELEGRAM Y SHEETS SIMULADOS

La respuesta aprobada no le llegaba al cliente.

Al aprobar, el envío reusaba la clave de idempotencia del aviso "Lo consulto con el equipo". Zernio la reconocía, contestaba OK y no mandaba nada: vos veías "Enviado" y el cliente, silencio. El paso 1 ya se lo avisa a Claude Code.

La credencial de Google desaparecía.

Pegada en el `.env` entre comillas dobles, python-dotenv no la puede leer y la variable no existe. Va entre comillas simples; en Railway se pega tal cual.

El seguimiento no salía nunca.

La imagen del Dockerfile de AgentKit (`python:3.11-slim`) no trae zonas horarias y cada vuelta fallaba al calcular la hora. Se arregla con `tzdata` en el `requirements.txt` ; el paso 3 ya lo pide.

REGLAS QUE CONVIENE SABER

Aprobá dentro de las 24 h.

Pasado ese tiempo desde el último mensaje del cliente, WhatsApp no deja mandarle texto libre. Si llegás tarde, el agente te avisa por Telegram que no se pudo.

El sandbox de Zernio no sirve para el segundo seguimiento.

Solo trae una plantilla fija y sin variables (`sandbox_start`). El primer seguimiento sí lo podés probar.

Contestá desde Telegram.

Si a un cliente escalado le respondés desde el inbox de Zernio, el agente no lo ve: cuando se lo devolvés, no sabe qué le dijiste.

Una sola copia del agente.

El seguimiento corre dentro del servidor: con dos instancias en Railway, cada cliente recibiría dos mensajes.

Empezá por la aprobación: es la que más tranquilidad da. Sumá la planilla y el seguimiento cuando ande bien.

NodeStudio

Automatizamos procesos de
negocio con IA y sistemas a medida.

nodestudioai.com