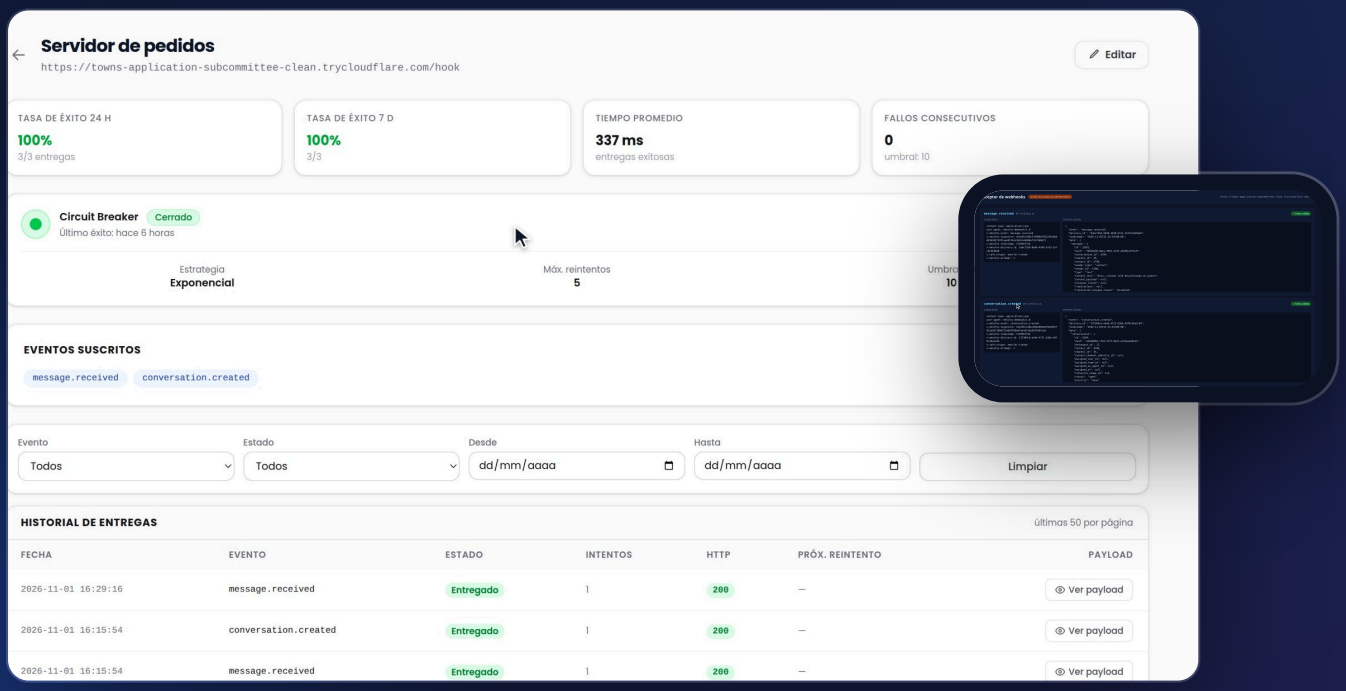
Omnifox

INTEGRACIONES Y API · GUÍA COMPLETA

# Integraciones y API

Claves API, llamadas reales, webhooks firmados, n8n, MCP y los SDK de Node, Python y PHP.




Guía paso a paso para equipos técnicos y dueños de negocio

Versión 1.0 · noviembre de 2026 · omnifox.io

CONTENIDO

# Índice

### Cómo leer este manual

Las capturas son pantallas reales de Omnifox con un negocio de ejemplo: **Café Montañero**, una tostaduría ficticia de Loja que vende café por su web. Todo lo que se muestra se hizo de verdad: la clave API se creó en la grabación, los comandos se ejecutaron contra Omnifox, los webhooks llegaron a un servidor receptor de pruebas, y n8n y el servidor MCP corrieron sus versiones publicadas. Los clientes son inventados.

Los nombres de botones y campos van en **negrita**. Los comandos usan `curl` y `jq`, y dos variables de entorno: `OMNIFOX_API_KEY` (tu clave) y `API=https://app.omnifox.io/api/v1`. Este manual sigue el video «Integraciones y API · Guía completa» del canal de YouTube de Omnifox.


01 Las formas de conectar Omnifox 3
02 Requisitos: plan y permisos 4
03 Crear una clave API 5
04 Alcances: qué puede hacer la clave 6
05 Copiar la clave una sola vez 7
06 La lista de claves: editar, regenerar y eliminar 8
07 La documentación de la API 9
08 Tu primera llamada 9
09 Contactos por API 10
10 Alcances en acción y errores 12
11 Conversaciones y mensajes por API 13
12 CRM y calendario por API 14
13 Paginación y límites 16
14 Webhooks: crear una suscripción 16
15 Secreto de firma, reintentos y encabezados 18
16 Entrega de prueba 19
17 Un evento real 21
18 Verificar la firma en tu servidor 22
19 Historial de entregas 23
20 La página de integraciones 24
21 Webhook personalizado: entrada y salida 25
22 n8n: instalar el nodo de Omnifox 26
23 n8n: la credencial de Omnifox 27
24 n8n: responder solo cuando preguntan el precio 28
25 n8n: todas las operaciones 30
26 MCP: que una IA opere Omnifox 31
27 Probar el servidor MCP con el Inspector 32
28 SDK oficiales: Node, Python y PHP 33
29 Buenas prácticas de seguridad 34
30 Solución de problemas 35

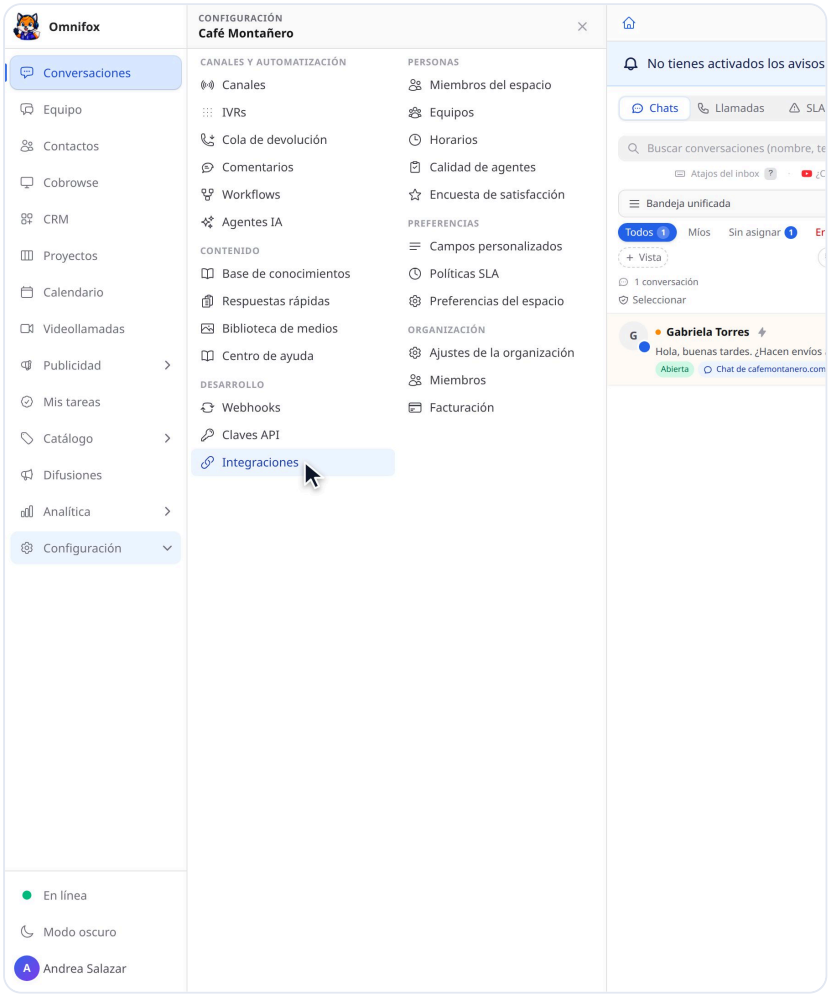
01

CAPÍTULO 1

# Las formas de conectar Omnifox

Tu sistema le habla a Omnifox con la API; Omnifox le avisa a tu sistema con los webhooks.

Todo lo de desarrolladores está en **Configuración → Desarrollo**: **Webhooks**, **Claves API** e **Integraciones**.




Configuración → Desarrollo: Webhooks, Claves API e Integraciones.

| Herramienta | Sentido | Para qué |
|---|---|---|
| **API REST** | Tu sistema → Omnifox | Crear y actualizar contactos, responder conversaciones, crear oportunidades, reservar citas, leer reportes... |
| **Webhooks** | Omnifox → tu sistema | Recibir un aviso al instante cuando entra un mensaje, se crea un contacto, se agenda una cita... |
| **Integraciones** | Los dos | Conectores listos: n8n, Make, Zapier, Webhook personalizado, Azur, Cal.com. |
| **n8n** | Los dos | Nodos oficiales de Omnifox (140 operaciones + un disparador) para automatizar sin programar. |
| **MCP** | IA → Omnifox | Que un asistente de IA (Claude, Cursor, ChatGPT...) opere Omnifox con 72 herramientas. |
| **SDK** | Tu código → Omnifox | Librerías oficiales para Node, Python y PHP. |

**CONSEJO**

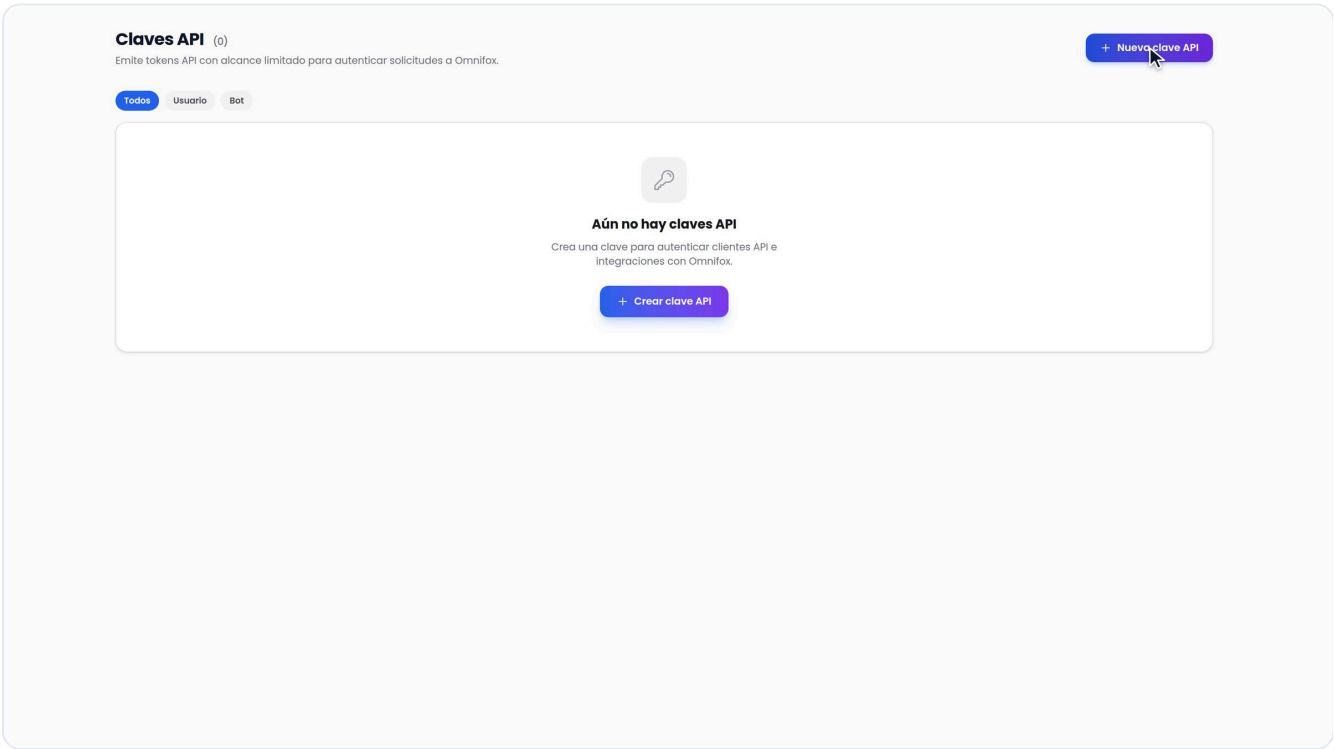
Todos usan la misma clave API. Si creas una clave por integración, puedes revocar una sin afectar a las demás.

02

CAPÍTULO 2

# Requisitos: plan y permisos

La API viene en Crece, Escala y Enterprise.




Claves API, todavía vacía.

- La API y los webhooks están en los planes **Crece**, **Escala** y **Enterprise**. En Lite, Impulsa y Conecta, la API responde que tu plan no la incluye.
- Para crear claves API hace falta el permiso de gestionar la configuración del espacio. Los webhooks solo los crea un **administrador**.
- Cada clave queda atada a **un** espacio de trabajo: lo que hagas con ella ocurre en ese espacio.

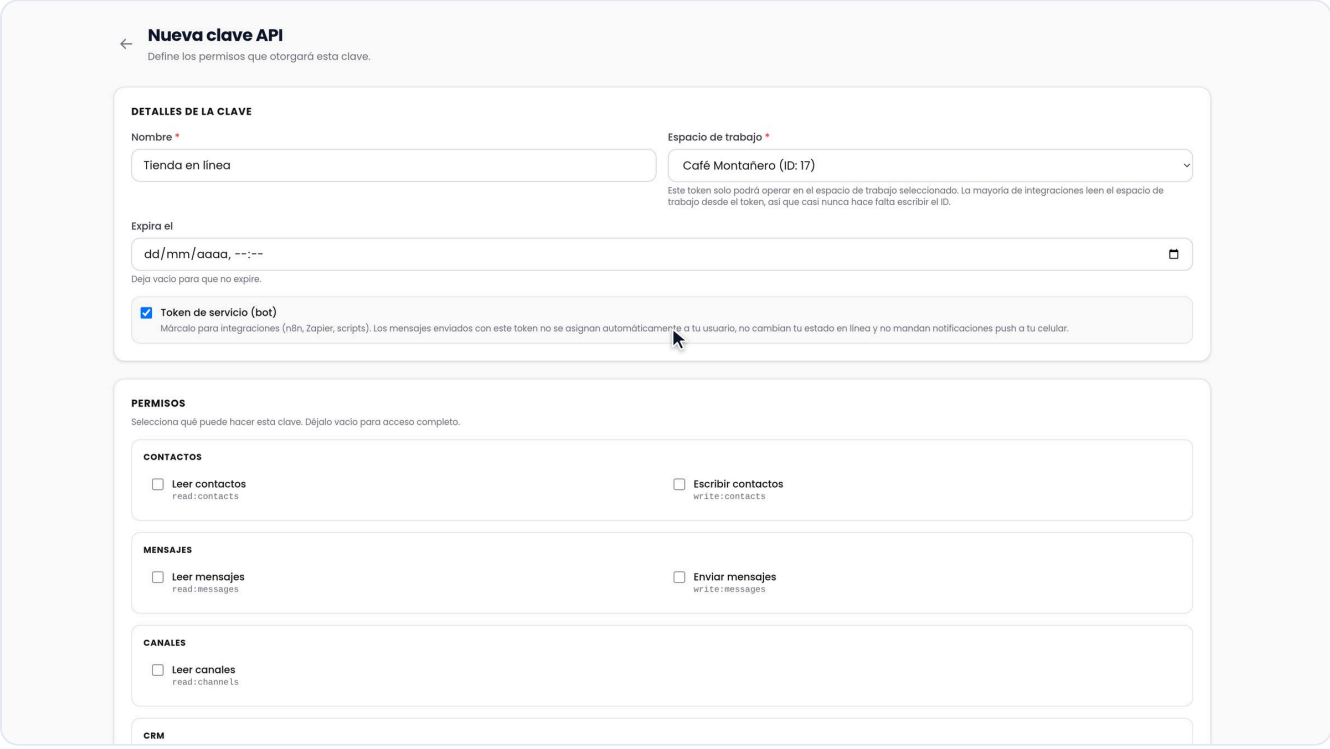
# 03

CAPÍTULO 3

# Crear una clave API

Nombre, espacio de trabajo, vencimiento y tipo.

1. Ve a **Configuración → Claves API** y pulsa **Nueva clave API**.
2. Escribe un **Nombre** que diga para qué es (por ejemplo, «Tienda en línea»).
3. Elige el **Espacio de trabajo**. La clave solo podrá trabajar en ese espacio, por eso casi nunca hace falta enviar el número del espacio en las llamadas.
4. Opcional: **Expira el**. Vacío = no vence. Para un proveedor externo o una prueba, ponle fecha.
5. Marca **Token de servicio (bot)** para integraciones.




Nueva clave API: nombre, espacio de trabajo, vencimiento y token de servicio (bot).

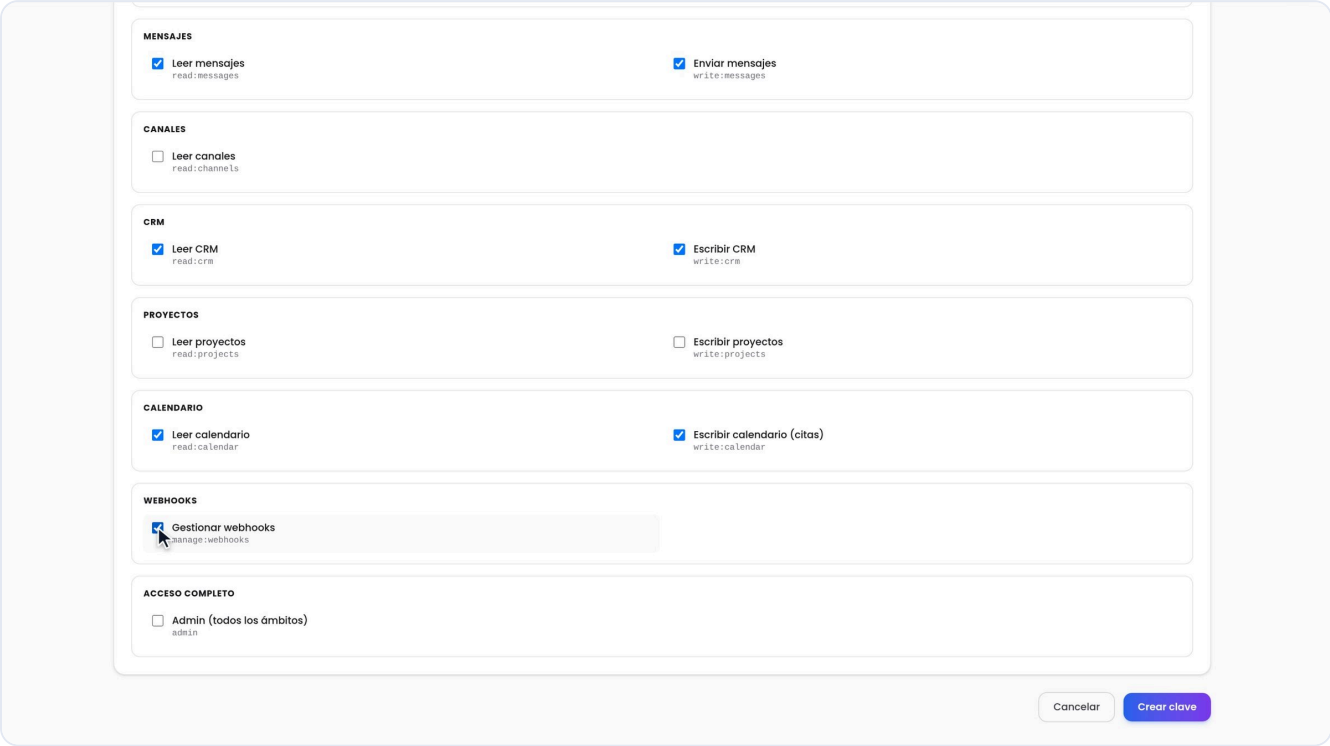
| Tipo | Comportamiento |
|---|---|
| **Usuario** | Actúa como tú: lo que envía se asigna a tu usuario y cuenta como actividad tuya. |
| **Bot** (token de servicio) | Lo que envía no se asigna a tu usuario, no cambia tu estado en línea y no te manda notificaciones push. Recomendado para n8n, Zapier, scripts y servidores. |

# 04

CAPÍTULO 4

# Alcances: qué puede hacer la clave

Dale a cada clave solo lo que necesita.




Los alcances marcados para la clave «Tienda en línea». Canales queda sin marcar a propósito.

| Módulo | Alcances | Cubre |
|---|---|---|
| Contactos | `read:contacts` · `write:contacts` | Contactos, búsqueda, etiquetas del contacto, fusionar. |
| Mensajes | `read:messages` · `write:messages` | Conversaciones y mensajes: leer, enviar, asignar, cerrar. |
| Canales | `read:channels` | Ver los canales del espacio. |
| CRM | `read:crm` · `write:crm` | Embudos, etapas, oportunidades, empresas y actividades. |
| Proyectos | `read:projects` · `write:projects` | Tableros, elementos y columnas. |
| Calendario | `read:calendar` · `write:calendar` | Calendarios, servicios, horarios libres y citas. |
| Webhooks | `manage:webhooks` | Crear, ver, rotar y borrar suscripciones (lo usa el disparador de n8n). |
| Acceso completo | `admin` | Todos los alcances. |

- Si no marcas ninguno, la clave tiene **acceso completo**.

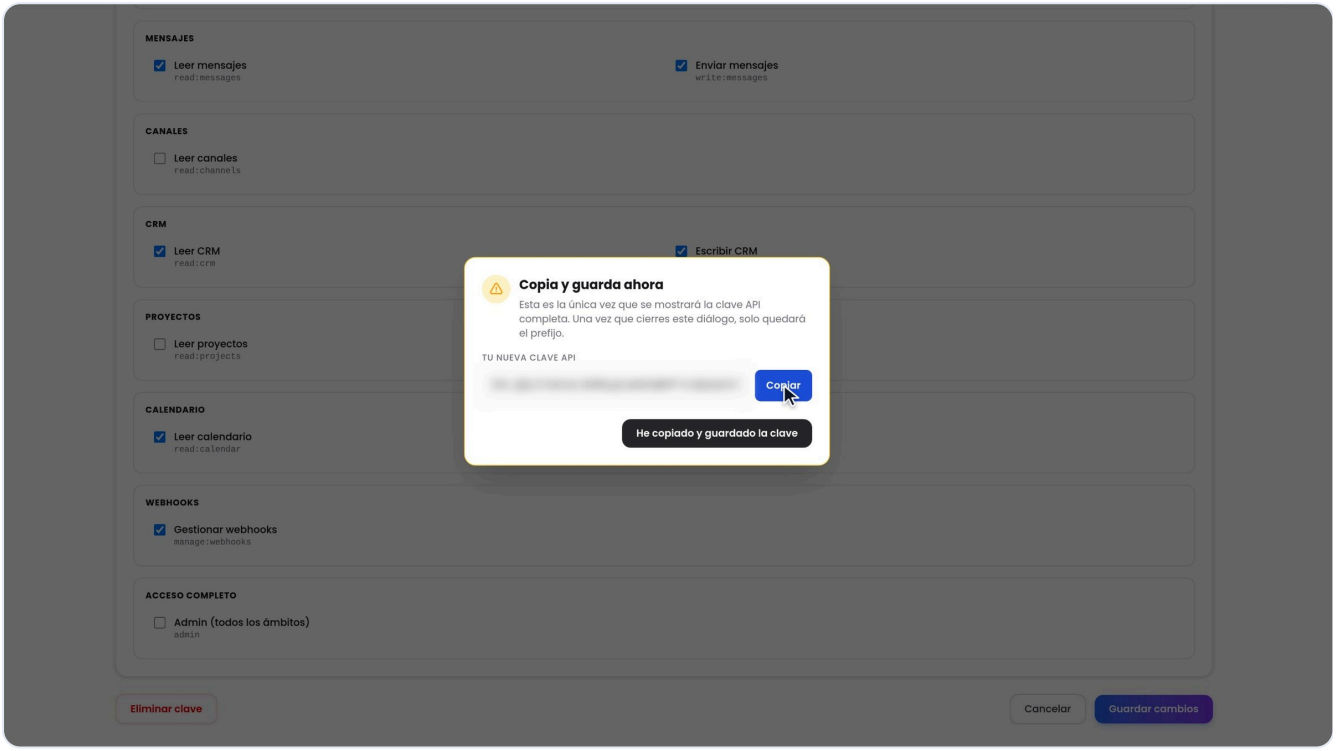
- **Escribir** incluye **leer** en el mismo módulo.
- Una llamada sin el alcance necesario responde **403** con el alcance que falta (capítulo 10).

05

CAPÍTULO 5

# Copiar la clave una sola vez

Se ve una vez. Guárdala bien.




La clave se muestra una sola vez (en las capturas aparece difuminada).

Al pulsar **Crear clave**, Omnifox muestra la clave completa **una única vez**. Pulsa **Copiar**, guárdala en el gestor de secretos o en las variables de entorno de tu servidor y confirma con **He copiado y guardado la clave**. Después solo se ve su prefijo.

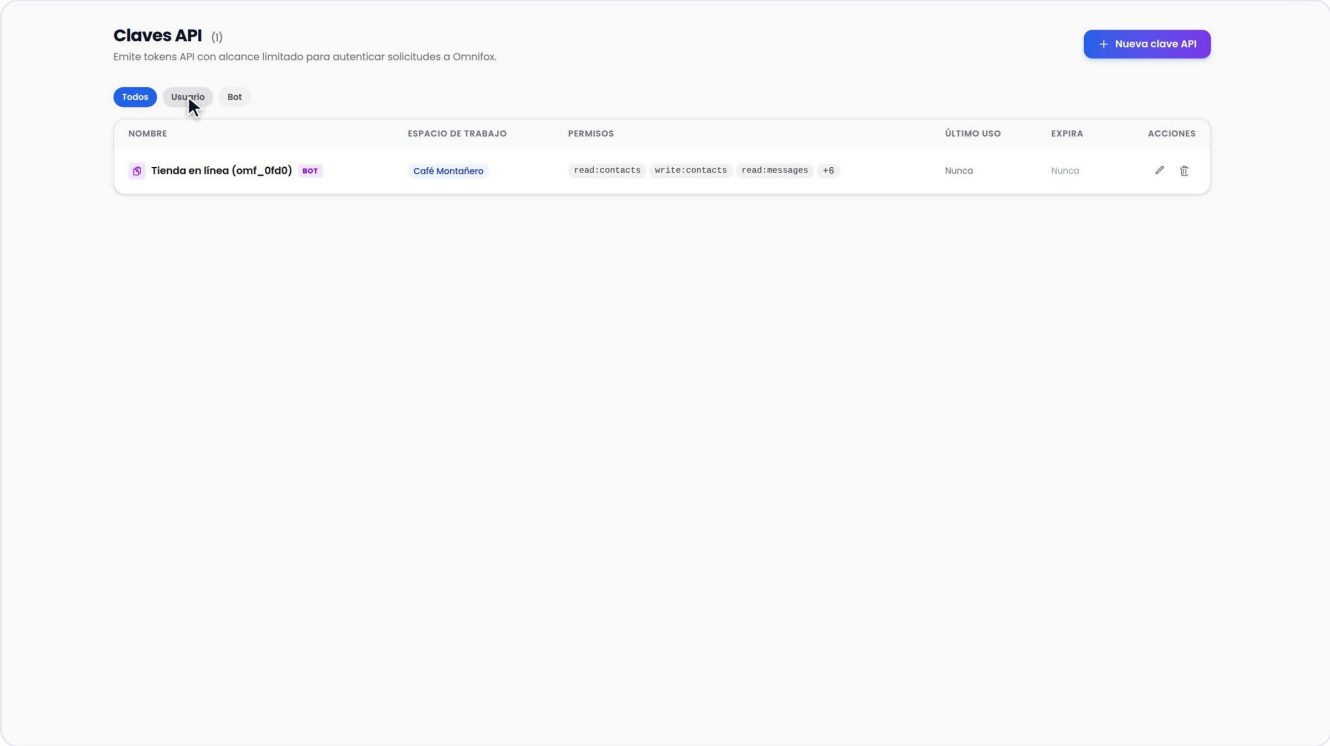
> **OJO**
>
> Si la pierdes, no la puedes recuperar: regenérala (capítulo 6). La anterior deja de funcionar en ese momento.

06

CAPÍTULO 6

# La lista de claves: editar, regenerar y eliminar

Editar sin romper, regenerar si se filtró, eliminar para cortar el acceso.




La lista de claves: tipo, espacio, alcances, último uso y vencimiento.

| Acción | Qué hace |
|---|---|
| Filtros **Todos / Usuario / Bot** | Muestra las claves de cada tipo. |
| **Editar** (lápiz) | Cambia nombre, alcances o vencimiento. El token sigue siendo el mismo. |
| **Regenerar** (dentro de editar) | Crea un token nuevo con los mismos permisos y lo muestra una vez. El anterior deja de funcionar. |
| **Eliminar** (papelera) | Revoca la clave en el acto. |
| **Último uso** | Cuándo se usó por última vez: útil para detectar claves abandonadas. |

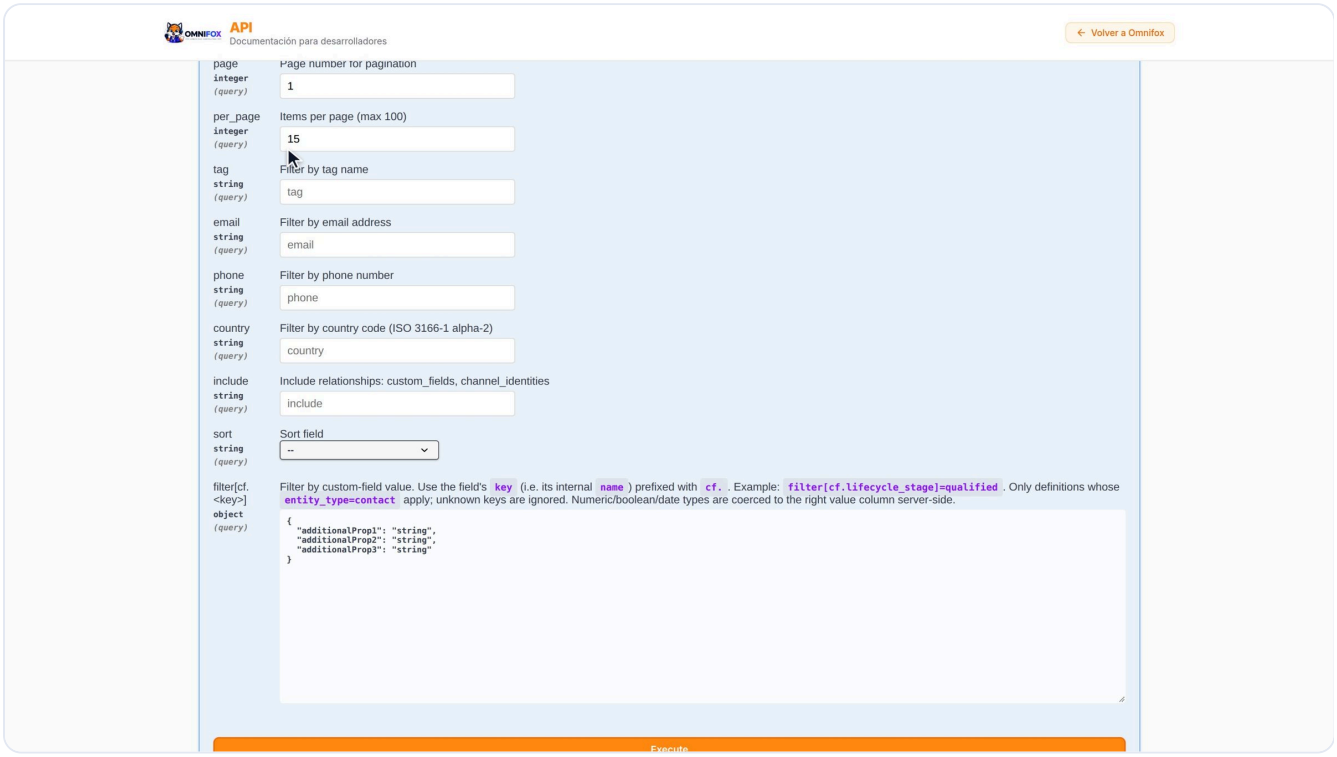
07

CAPÍTULO 7

# La documentación de la API

Cada endpoint, con parámetros, ejemplos y errores.

La documentación está en **omnifox.io/docs**. Arriba, accesos a los nodos de n8n, al servidor MCP y a los SDK. Debajo, la referencia completa agrupada por módulo.




La referencia de la API en /docs, con cada endpoint y su ejemplo.

- **Authorize**: pega tu clave y prueba cualquier llamada desde la misma página.
- **openapi.json**: descárgalo para Postman, Insomnia o un generador de código.

08

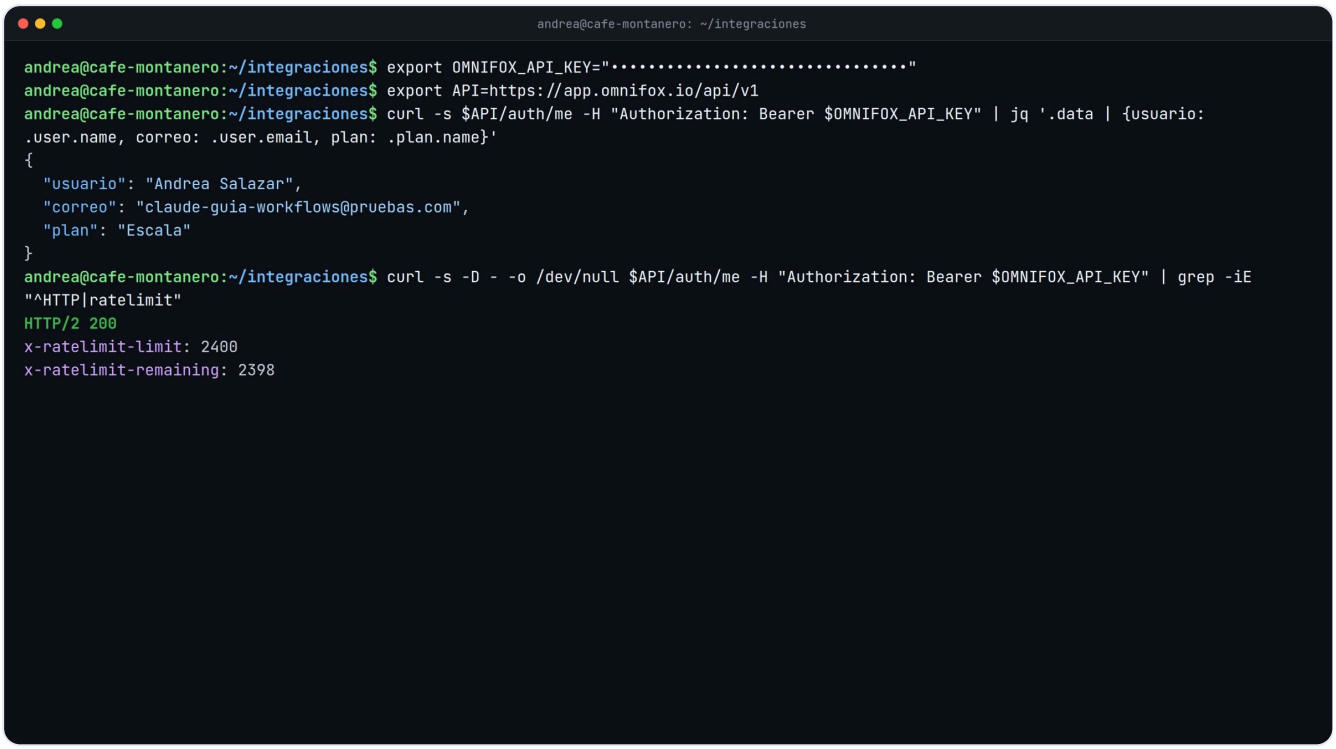
CAPÍTULO 8

# Tu primera llamada

Una cabecera, y la API te responde.

```
export OMNIFOX_API_KEY="tu-clave"
export API=https://app.omnifox.io/api/v1

curl -s $API/auth/me -H "Authorization: Bearer $OMNIFOX_API_KEY" | jq '.data | {usuario: .user.name,
plan: .plan.name}'
```




Primera llamada desde la terminal y las cabeceras de límite de uso.

Todas las respuestas correctas tienen la misma forma: `success`, `message` y `data`. Las listas traen además la paginación (capítulo 13). Las cabeceras `X-RateLimit-Limit` y `X-RateLimit-Remaining` dicen cuántas llamadas por minuto te da tu plan y cuántas te quedan.

09

CAPÍTULO 9

# Contactos por API

Crear, buscar y actualizar.

```
curl -s -X POST $API/contacts -H "Authorization: Bearer $OMNIFOX_API_KEY" -H "Content-Type:
application/json" \
  -d '{"display_name":"Valentina
Ríos","phone":"+593990710055","email":"valentina.rios@example.com","city":"Quito"}'

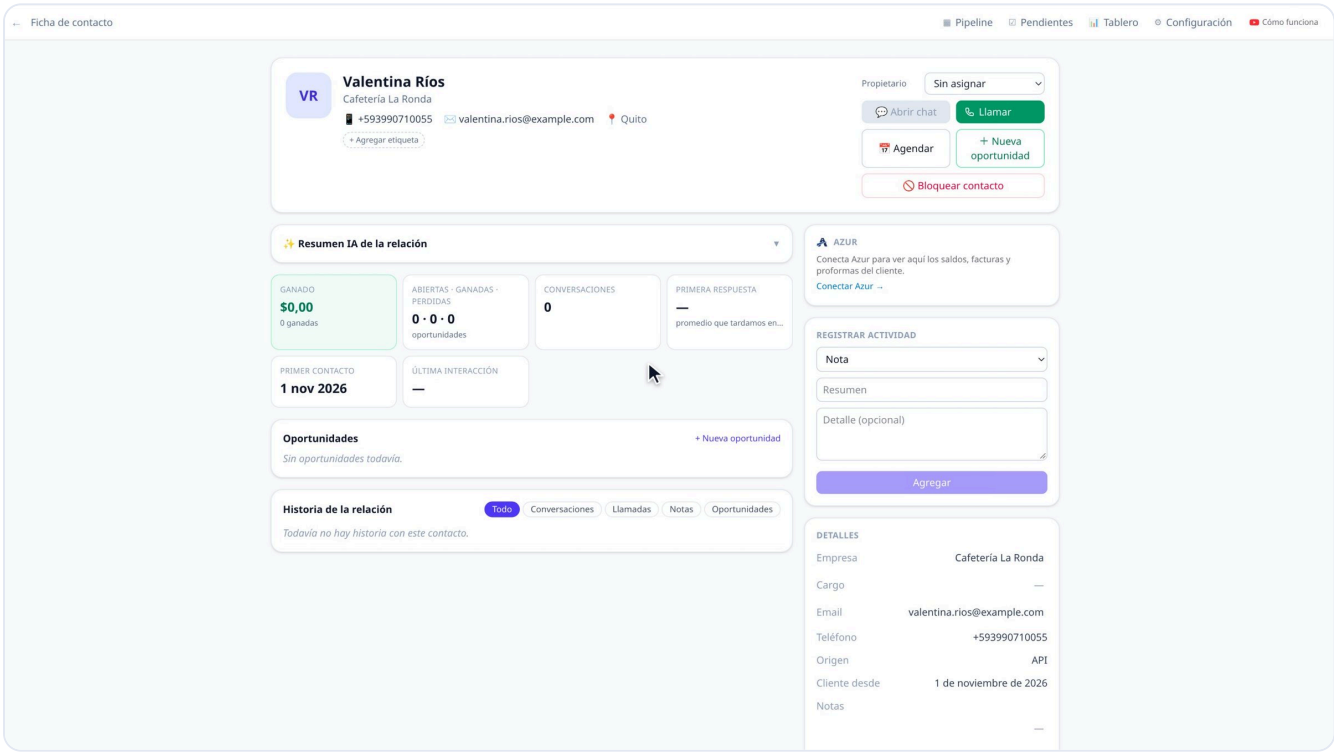
curl -s "$API/contacts/search?q=Valentina" -H "Authorization: Bearer $OMNIFOX_API_KEY"

curl -s -X PUT $API/contacts/$ID -H "Authorization: Bearer $OMNIFOX_API_KEY" -H "Content-Type:
application/json" -d '{"company":"Cafetería La Ronda"}'
```

```
andrea@cafe-montanero: ~/integraciones

andrea@cafe-montanero:~/integraciones$ curl -s -X POST $API/contacts -H "Authorization: Bearer $OMNIFOX_API_KEY" -H "Content-Type:
application/json" -d '{"display_name":"Valentina
Ríos","first_name":"Valentina","last_name":"Ríos","phone":"+593990710055","email":"valentina.rios@example.com","city":"Quito"}' | jq
'.data | {id, display_name, phone, city}'
{
  "id": 5699,
  "display_name": "Valentina Ríos",
  "phone": "+593990710055",
  "city": "Quito"
}
andrea@cafe-montanero:~/integraciones$ ID=$(curl -s "$API/contacts/search?q=Valentina" -H "Authorization: Bearer $OMNIFOX_API_KEY" | jq
'.data[0].id'); echo "id: $ID"
id: 5699
andrea@cafe-montanero:~/integraciones$ curl -s -X PUT $API/contacts/$ID -H "Authorization: Bearer $OMNIFOX_API_KEY" -H "Content-Type:
application/json" -d '{"company":"Cafetería La Ronda"}' | jq '.data | {id, display_name, company}'
{
  "id": 5699,
  "display_name": "Valentina Ríos",
  "company": "Cafetería La Ronda"
}
```

Crear, buscar y actualizar un contacto por API.




El contacto creado por API, en Contactos.

> **CONSEJO**
>
> Para no duplicar clientes usa `POST /contacts/upsert` : crea o actualiza según el teléfono o el correo.

10

CAPÍTULO 10

# Alcances en acción y errores

Los errores te dicen exactamente qué pasó.




Un 403 con el alcance que falta, un 401 con una clave inválida y un 422 de validación.

| Código | Cuándo | Qué hacer |
|---|---|---|
| 401 | La clave no existe, venció o fue revocada/regenerada. | Revisa que esté bien copiada; crea o regenera una. |
| 403 | Falta un alcance ( `required_scope` ) o tu plan no incluye esa función. | Agrega el alcance a la clave o revisa el plan. |
| 404 | El recurso no existe en el espacio de la clave. | Revisa el ID y que la clave sea del espacio correcto. |
| 422 | Un dato no es válido. | La respuesta trae el campo y el motivo. |
| 429 | Superaste el límite por minuto. | Espera unos segundos y reintenta (los SDK lo hacen solos). |

11

CAPÍTULO 11

# Conversaciones y mensajes por API

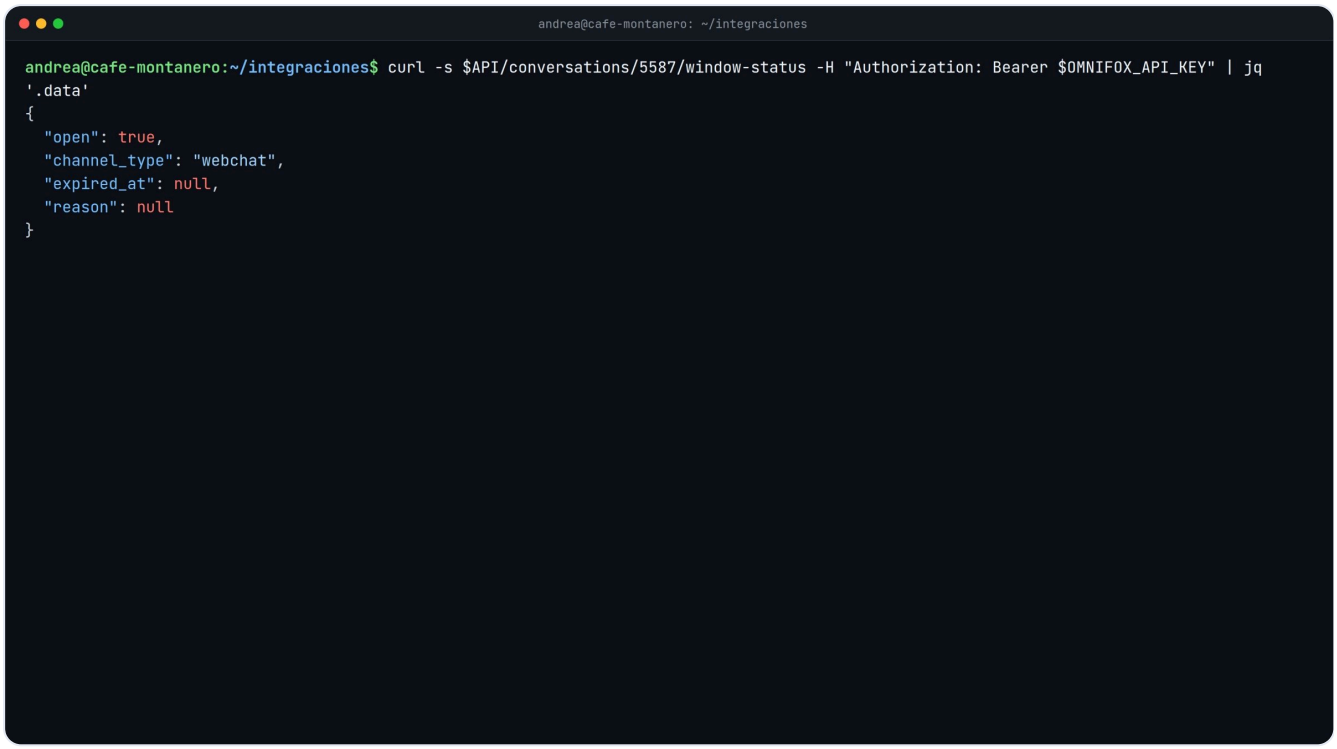
Responde a tus clientes desde tu sistema, por el mismo canal.

```
curl -s "$API/conversations?status=open" -H "Authorization: Bearer $OMNIFOX_API_KEY"
curl -s "$API/conversations/$CID/messages" -H "Authorization: Bearer $OMNIFOX_API_KEY"
curl -s -X POST $API/conversations/$CID/messages -H "Authorization: Bearer $OMNIFOX_API_KEY" -H
"Content-Type: application/json" \
  -d '{"type":"text","content_text":"¡Hola, Gabriela! Sí, enviamos a Quito en 48 horas."}'
curl -s $API/conversations/$CID/window-status -H "Authorization: Bearer $OMNIFOX_API_KEY"
```




Leer la conversación y responder por API.

Omnifox envía la respuesta por el canal por el que escribió el cliente (en el ejemplo, el chat de la web, donde la respuesta aparece al instante).




La ventana de 24 horas (window-status) de la conversación.

**OJO**

En WhatsApp solo puedes escribir texto libre dentro de la ventana de 24 horas desde el último mensaje del cliente. Consulta `window-status` antes de enviar: si está cerrada, usa una plantilla aprobada.

12

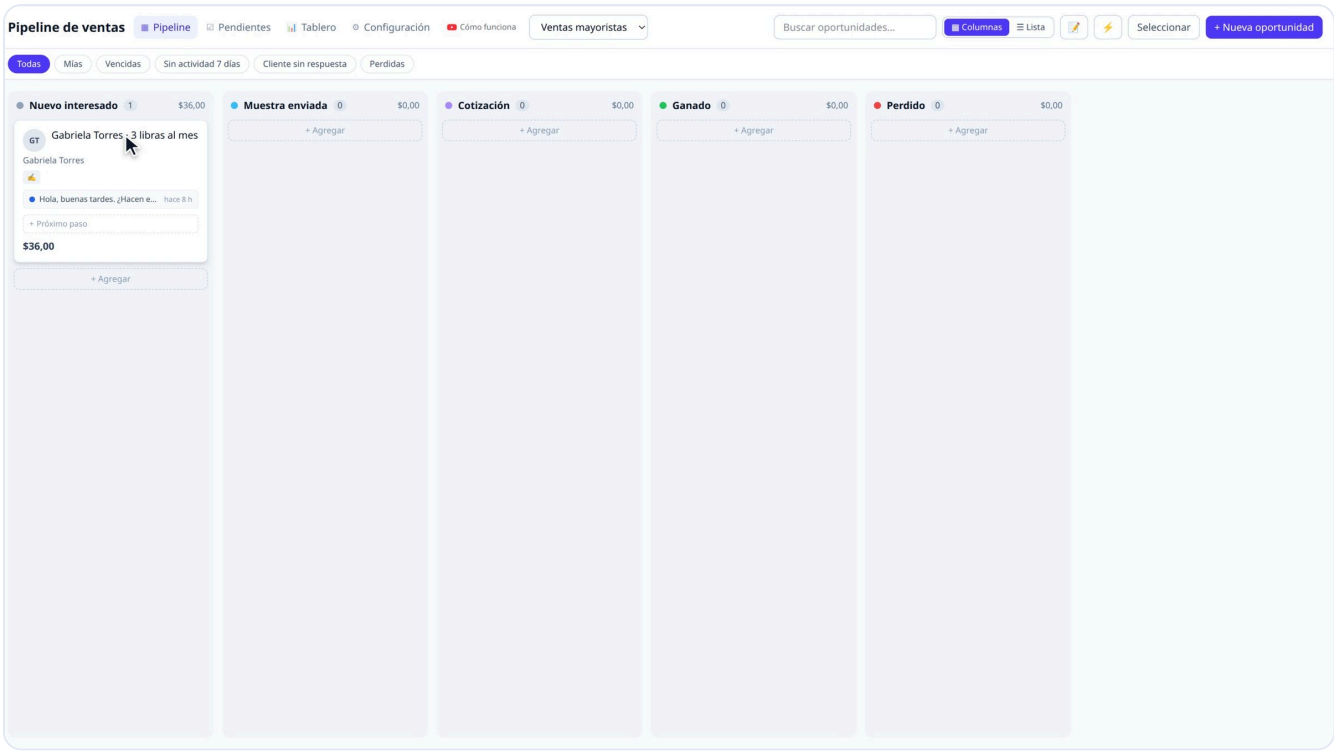
CAPÍTULO 12

# CRM y calendario por API

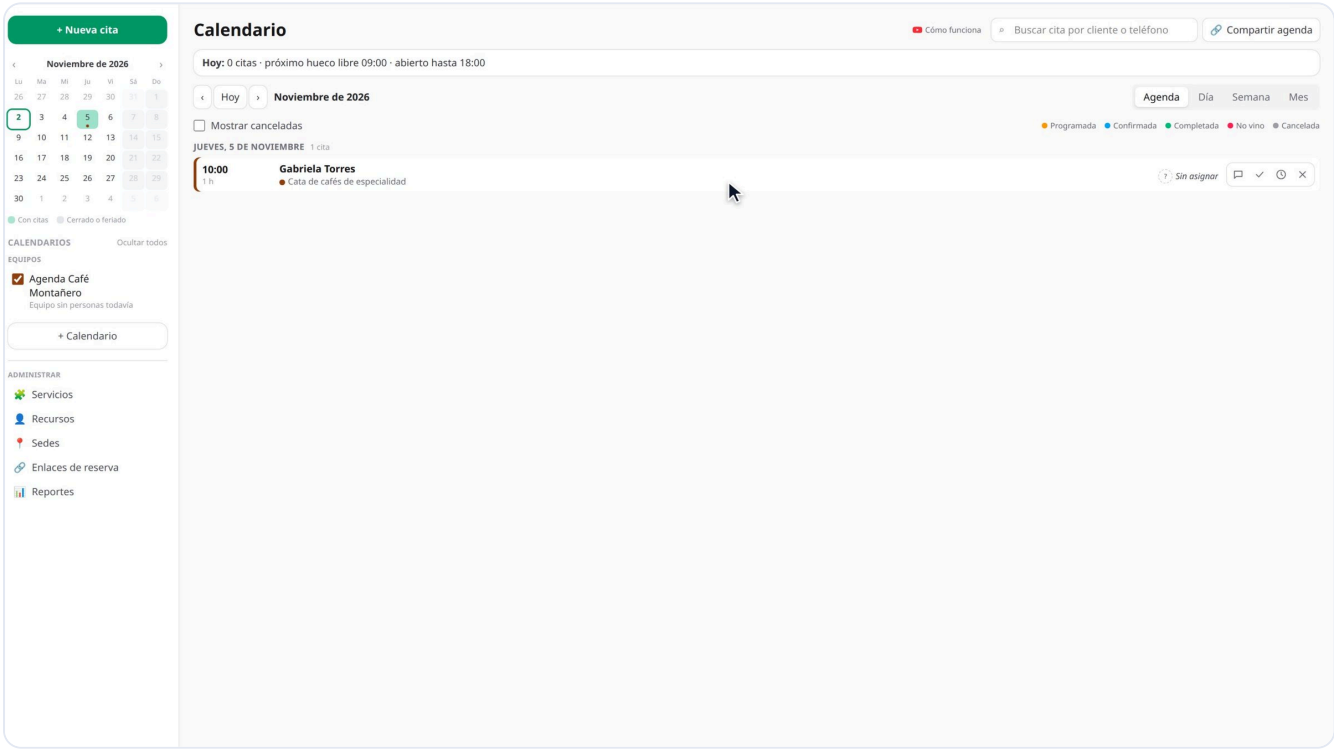
La API llega a toda la plataforma.

```
curl -s -X POST $API/crm/deals -H "Authorization: Bearer $OMNIFOX_API_KEY" -H "Content-Type: application/json" \
  -d '{"title":"Gabriela Torres · 3 libras al mes","pipeline_id":62,"stage_id":266,"amount":36,"contact_id":5051}}'

curl -s "$API/calendar/slots?calendar_id=139&service_id=141&from=2026-11-05&to=2026-11-05" -H "Authorization: Bearer $OMNIFOX_API_KEY"
curl -s -X POST $API/calendar/appointments -H "Authorization: Bearer $OMNIFOX_API_KEY" -H "Content-Type: application/json" \
  -d '{"calendar_id":139,"service_id":141,"contact_id":5051,"starts_at":"2026-11-05T10:00:00-05:00"}'
```




La oportunidad creada por API, en el embudo del CRM.




La cita reservada por API, en el calendario.

Las citas también se reprograman ( `PUT /calendar/appointments/{id}` ) y se cancelan ( `POST /calendar/appointments/{id}/cancel` ). Lo mismo vale para proyectos, catálogo, pedidos, reportes, equipos, etiquetas y difusiones: todo está en /docs.

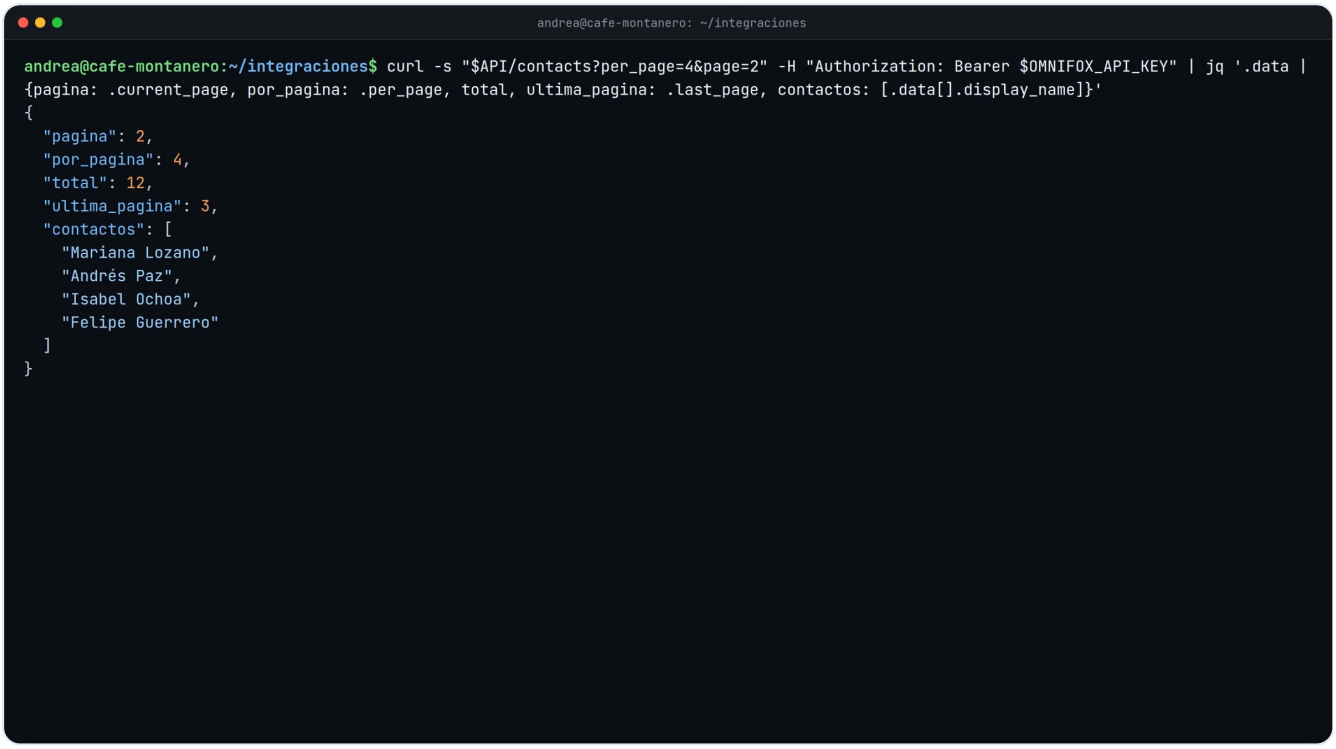
13

CAPÍTULO 13

# Paginación y límites

Recorre las páginas y respeta el límite.

```
curl -s "$API/contacts?per_page=4&page=2" -H "Authorization: Bearer $OMNIFOX_API_KEY"
```




Paginación: página, por página, total y última página.

La respuesta trae `current_page`, `per_page`, `total` y `last_page`. Avanza hasta `last_page`. Si recibes **429**, espera y reintenta.

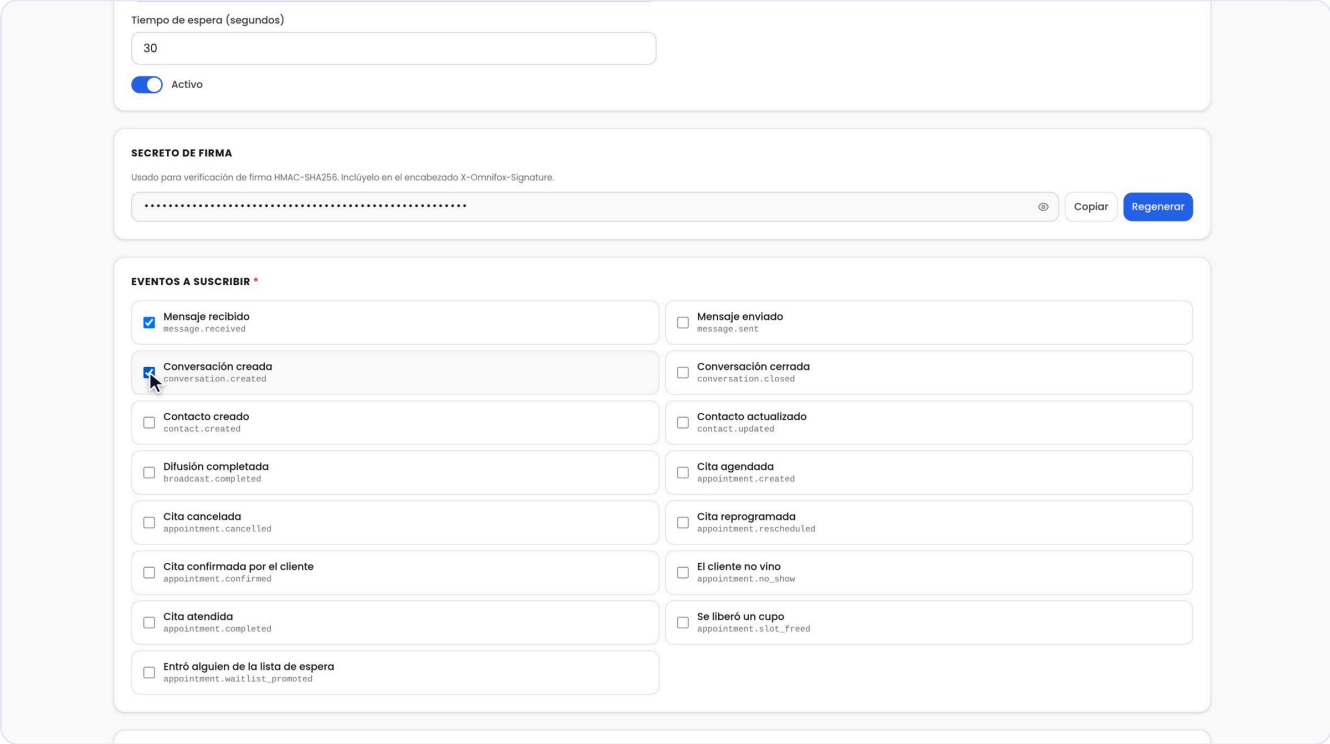
14

CAPÍTULO 14

# Webhooks: crear una suscripción

Que Omnifox te avise al instante.

1. Ve a **Configuración** → **Webhooks** y pulsa **Nuevo webhook**.
2. Escribe un **Nombre** y la **URL de destino** de tu servidor (debe ser **https** y pública).
3. Marca los **Eventos a suscribir**.




Nuevo webhook: nombre, URL https y eventos.

| Evento | Cuándo llega |
|---|---|
| `message.received` | Entra un mensaje de un cliente. |
| `message.sent` | Tu equipo, la IA o un flujo envía un mensaje. |
| `conversation.created` · `conversation.closed` | Se abre o se cierra una conversación. |
| `contact.created` · `contact.updated` | Se crea o cambia un contacto. |
| `broadcast.completed` | Terminó una difusión. |
| `appointment.*` | Cita agendada, cancelada, reprogramada, confirmada por el cliente, el cliente no vino, atendida, se liberó un cupo, entró alguien de la lista de espera. |

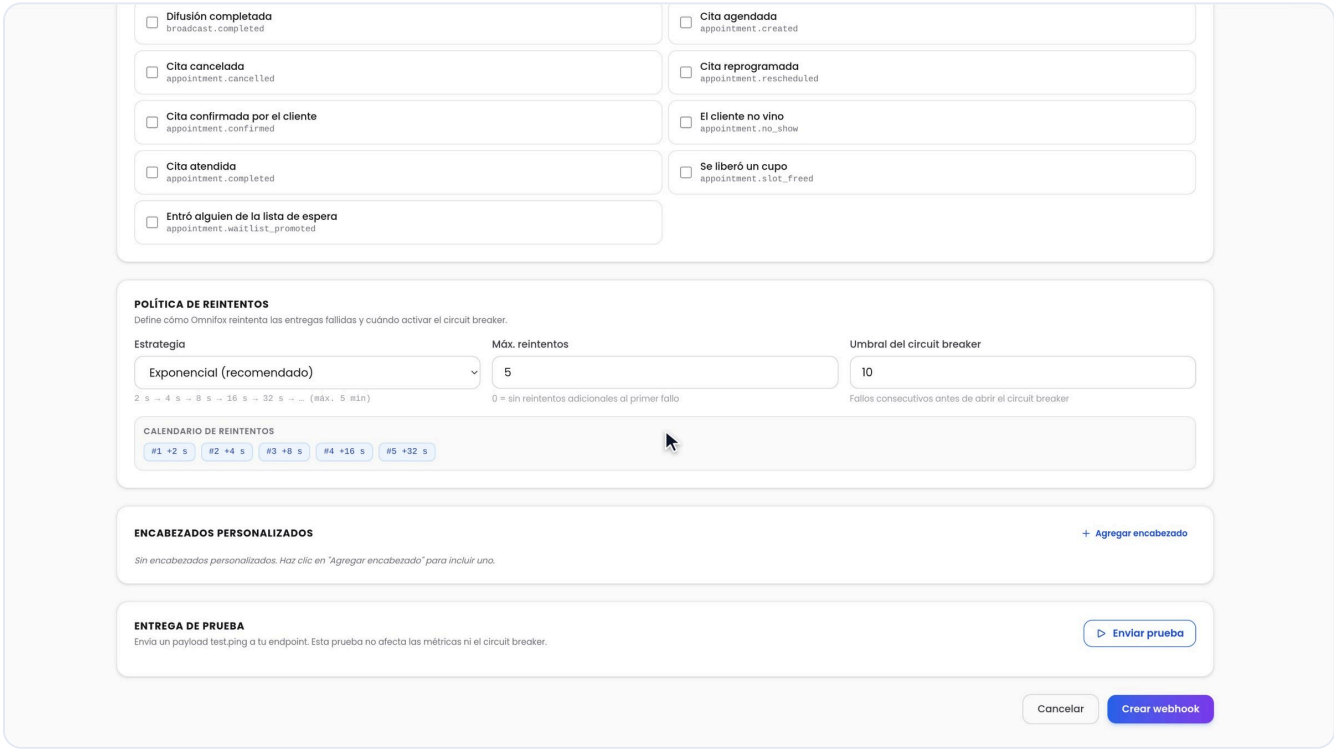
15

CAPÍTULO 15

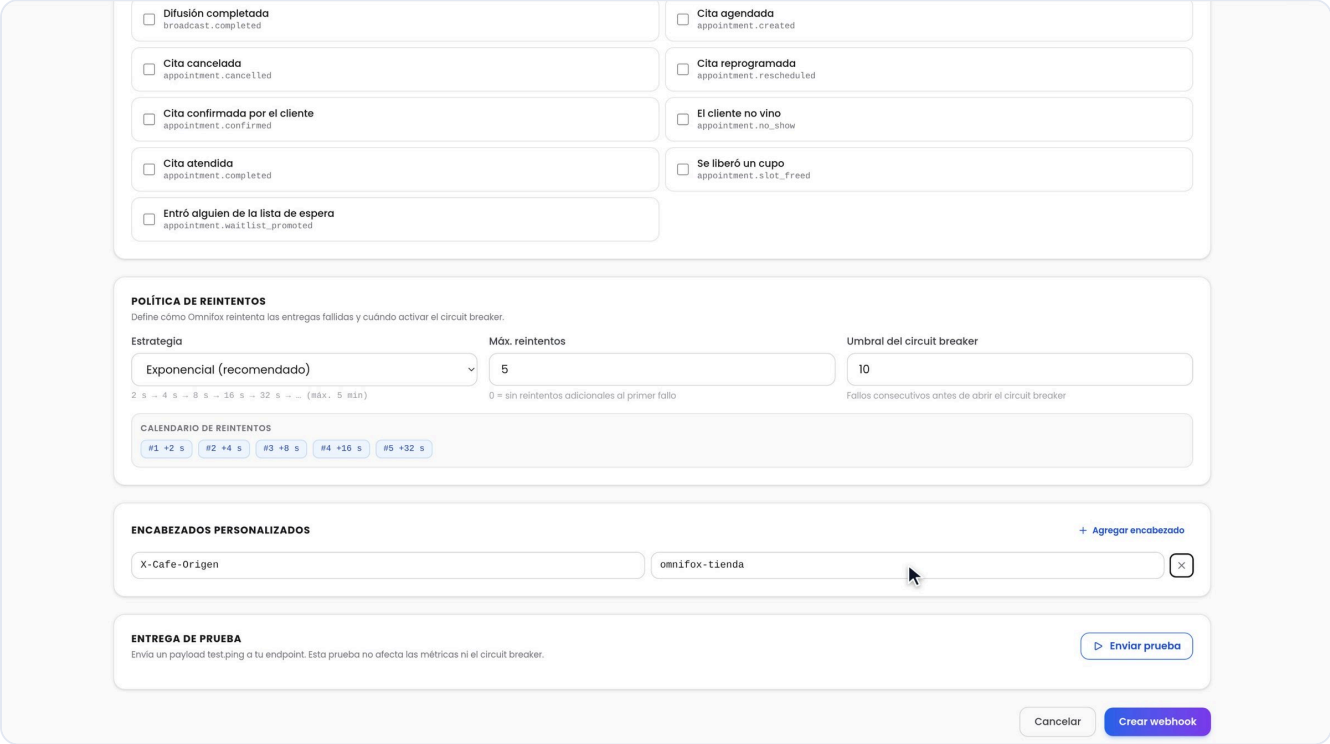
# Secreto de firma, reintentos y encabezados

Firmado, con reintentos y protegido.

| Ajuste | Para qué |
|---|---|
| **Secreto de firma** | Omnifox firma cada envío con este secreto. Cópialo a tu servidor para verificar (capítulo 18). **Regenerar** crea uno nuevo. |
| **Estrategia** | **Exponencial** (recomendada: 2 s, 4 s, 8 s, 16 s… hasta 5 min), **Lineal** (cada 60 s) o **Sin reintentos**. |
| **Máx. reintentos** | Cuántas veces reintentar una entrega fallida. |
| **Umbral del circuit breaker** | Tras tantos fallos seguidos, Omnifox deja de enviar un rato y luego prueba de nuevo. |
| **Tiempo de espera** | Segundos que espera la respuesta de tu servidor. |
| **Encabezados personalizados** | Cabeceras propias en cada envío (por ejemplo, una clave de tu lado). |




Política de reintentos y calendario de reintentos.




Encabezados personalizados.

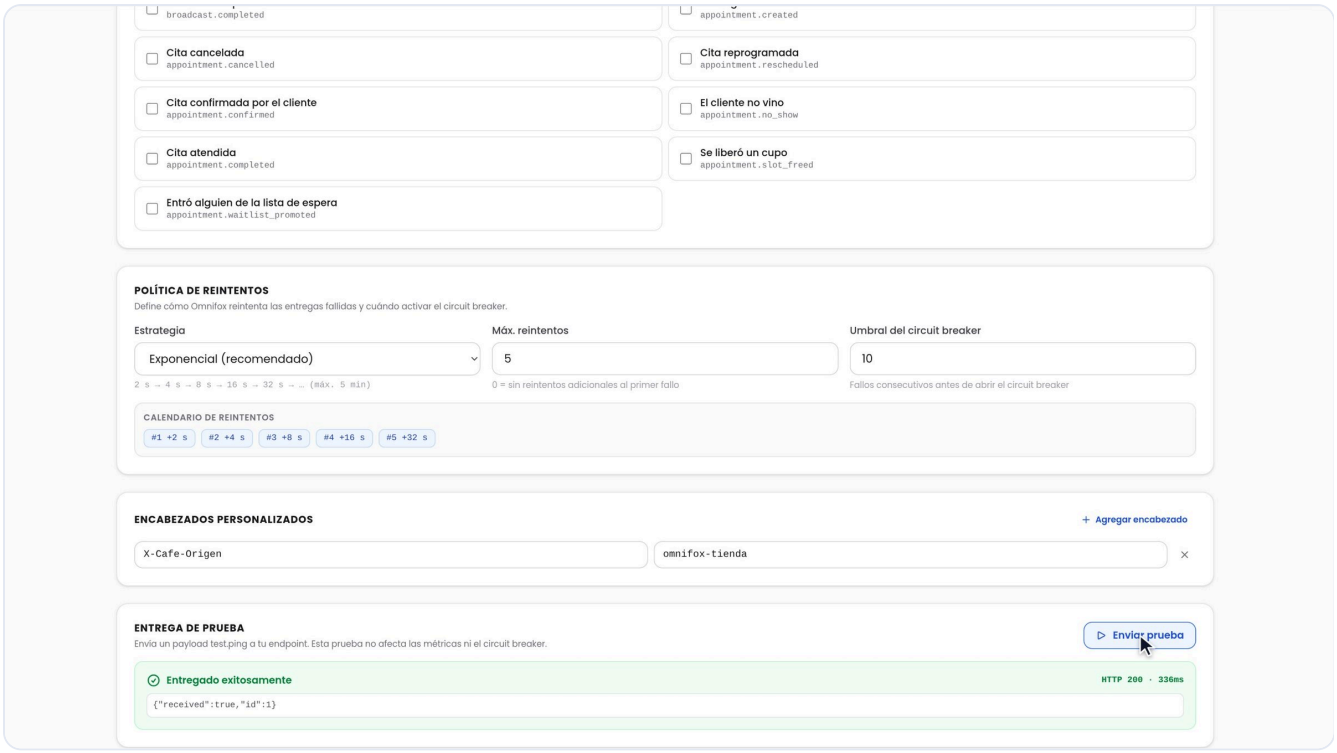
16

CAPÍTULO 16

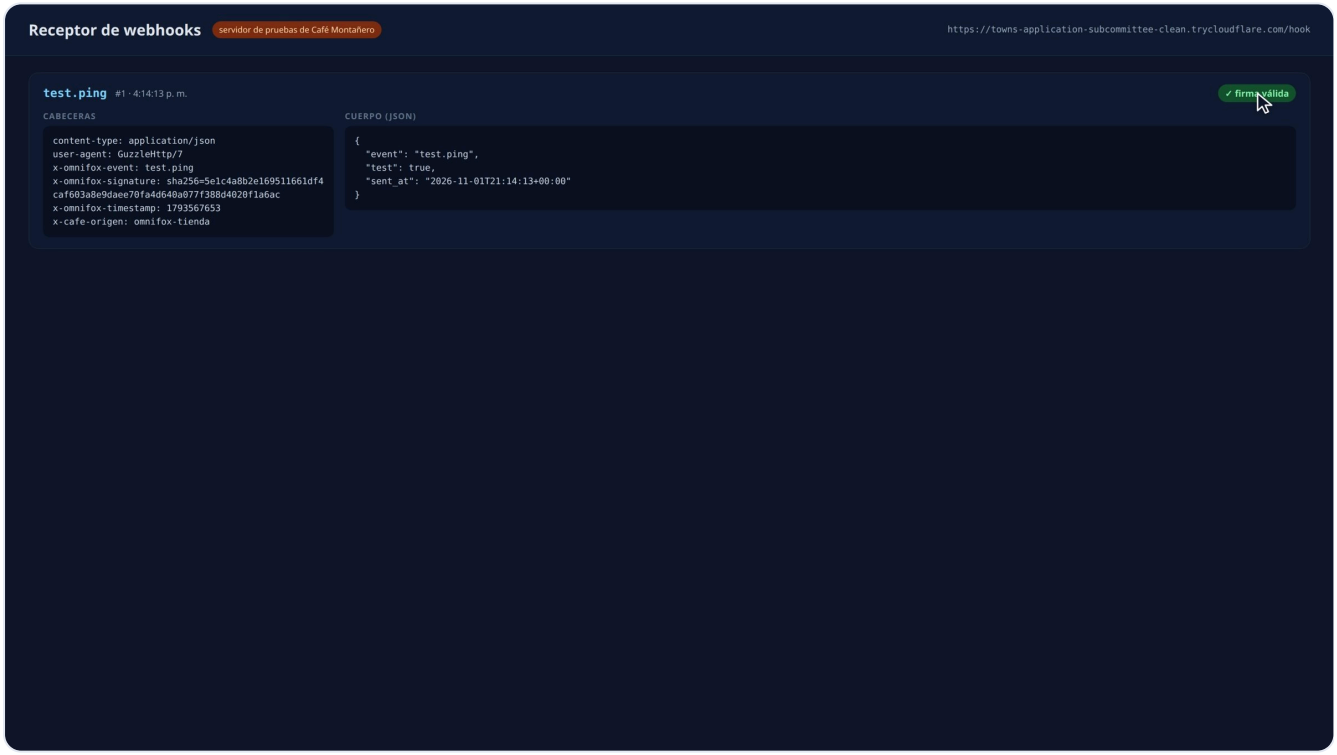
# Entrega de prueba

Prueba antes de guardar.

**Enviar prueba** manda un evento `test.ping` a tu URL y muestra el código HTTP, el tiempo y la respuesta. No cuenta para las métricas ni para el circuit breaker.




Resultado de la entrega de prueba.



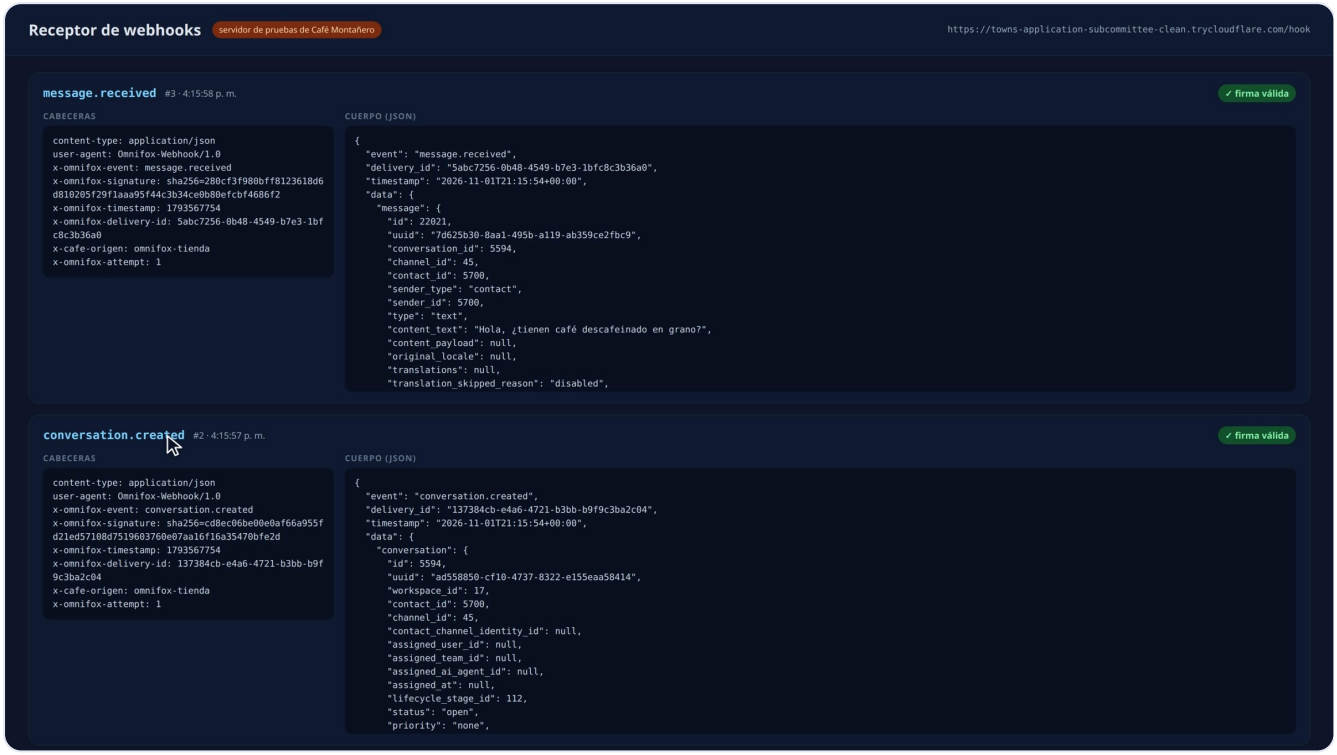

El ping en el receptor, con la firma verificada.

17

CAPÍTULO 17

# Un evento real

Un cliente escribe y tu servidor se entera.




Los eventos message.received y conversation.created en el receptor.

| Cabecera | Contenido |
|---|---|
| `X-Omnifox-Event` | El evento ( `message.received` ...). |
| `X-Omnifox-Delivery-Id` | Identificador único de la entrega: guárdalo para no procesar dos veces el mismo evento. |
| `X-Omnifox-Attempt` | Número de intento (1 la primera vez). |
| `X-Omnifox-Timestamp` | Marca de tiempo Unix de la firma. |
| `X-Omnifox-Signature` | `sha256=` + HMAC-SHA256 de «timestamp.cuerpo». |

El cuerpo trae `event` , `delivery_id` , `timestamp` y `data` (en `message.received` : el mensaje con `content_text` , el contacto, la conversación y el canal).

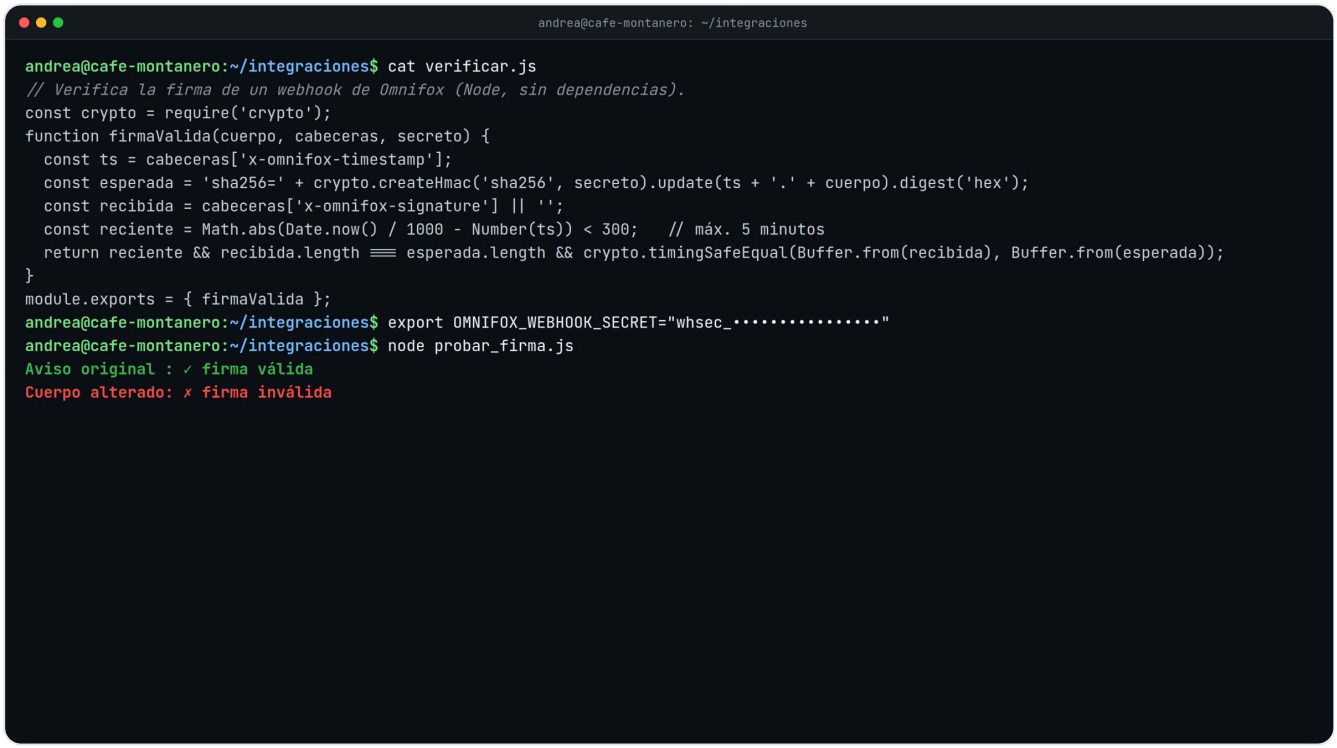
18

CAPÍTULO 18

# Verificar la firma en tu servidor

Diez líneas para saber que el aviso es auténtico.

```
const crypto = require('crypto');
function firmaValida(cuerpo, cabeceras, secreto) {
  const ts = cabeceras['x-omnifox-timestamp'];
  const esperada = 'sha256=' + crypto.createHmac('sha256', secreto).update(ts + '.' +
cuerpo).digest('hex');
  const recibida = cabeceras['x-omnifox-signature'] || '';
  const reciente = Math.abs(Date.now() / 1000 - Number(ts)) < 300;   // máx. 5 minutos
  return reciente && recibida.length === esperada.length &&
         crypto.timingSafeEqual(Buffer.from(recibida), Buffer.from(esperada));
}
```




Verificación de la firma: el aviso original pasa, el alterado no.

**OJO**

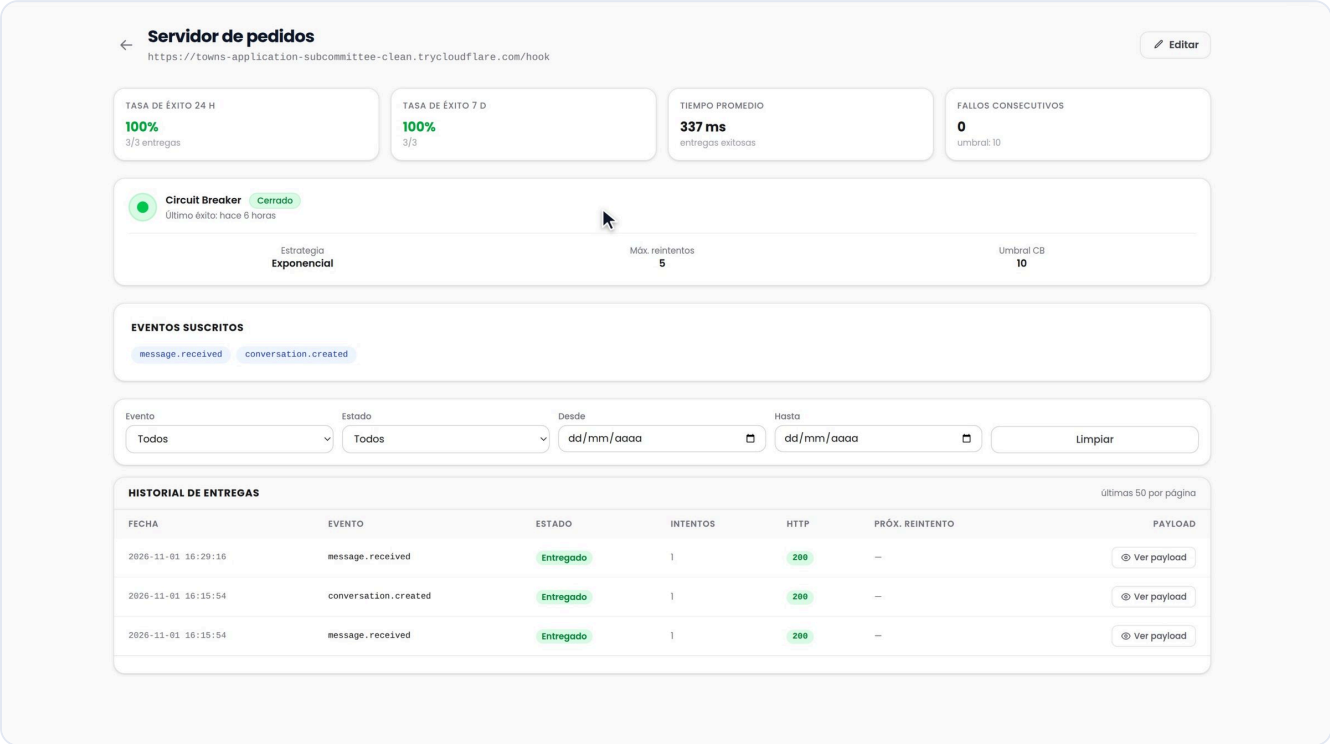
Usa el cuerpo EXACTO que recibiste (sin volver a serializar el JSON) y una comparación de tiempo constante.

19

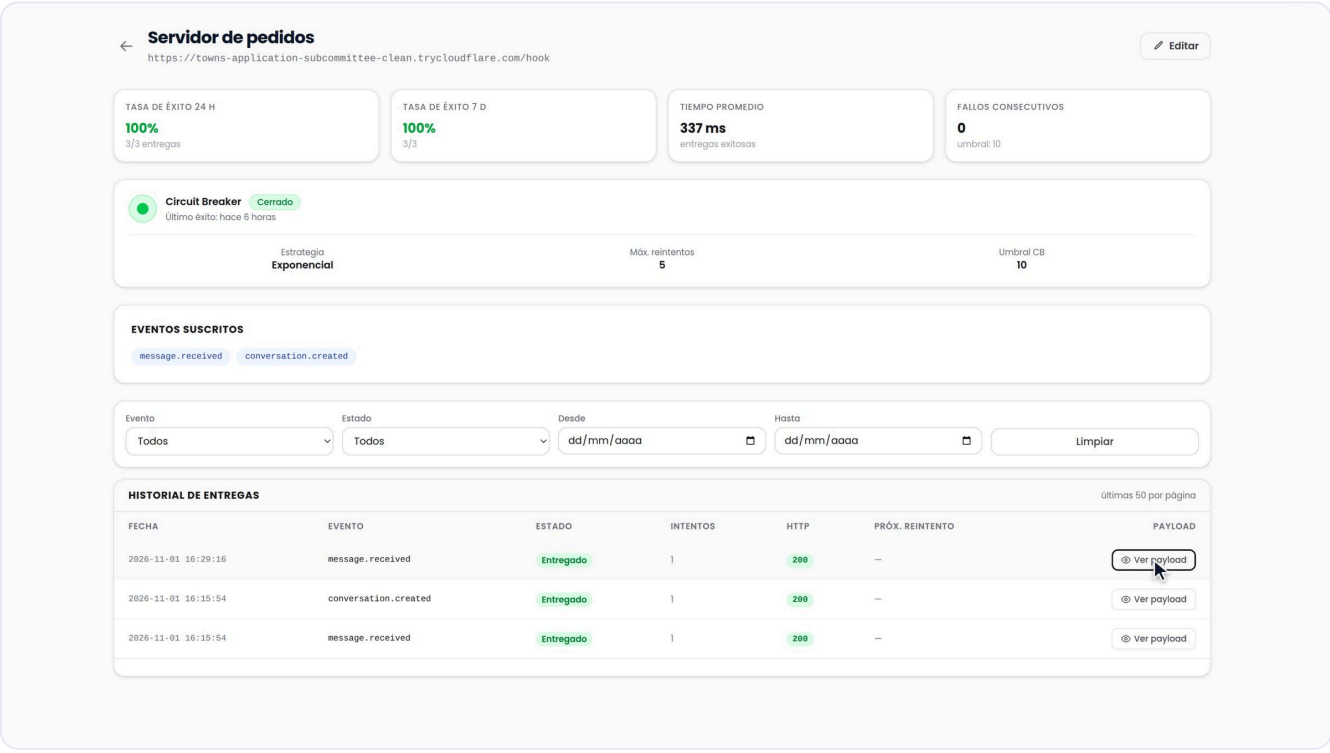
CAPÍTULO 19

# Historial de entregas

Qué se envió, qué respondió tu servidor y cuándo.




Detalle del webhook: tasas de éxito, tiempo promedio y fallos seguidos.




El historial de entregas y el payload de una entrega.

- Tasa de éxito de 24 horas y 7 días, tiempo promedio y fallos seguidos.

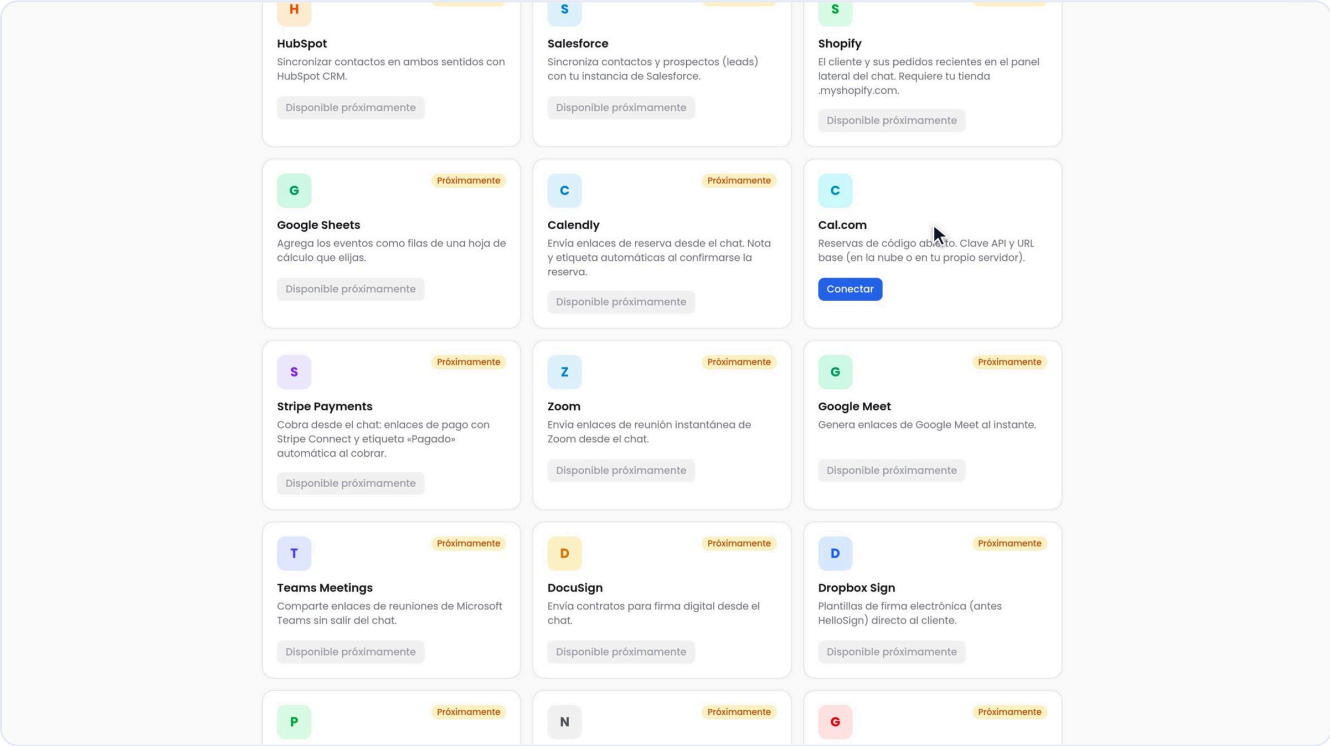
- Estado del **circuit breaker**; si quedó abierto, **Forzar cerrar circuit breaker** cuando tu servidor vuelva.
- **Historial de entregas** con filtros por evento, estado y fecha; **Ver payload** muestra lo enviado y la respuesta.
- Desde la lista: pausar, probar, regenerar el secreto, editar o eliminar.

# 20

CAPÍTULO 20

# La página de integraciones

Conectores listos.




La página de integraciones.

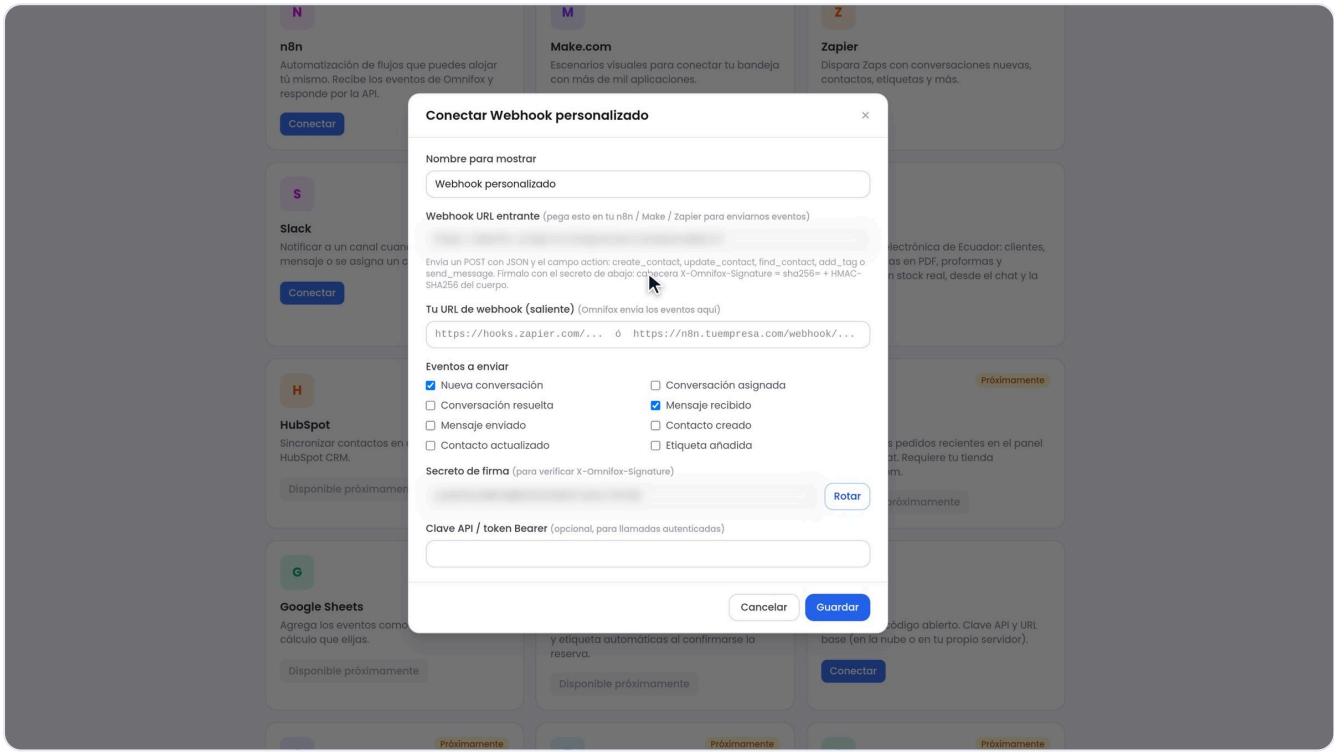
| Integración | Cómo funciona |
|---|---|
| **n8n, Make, Zapier, Webhook personalizado** | Webhooks en los dos sentidos: una URL entrante con acciones y eventos salientes firmados (capítulo 21). |
| **Azur** | Facturación electrónica de Ecuador: clientes, saldos, facturas, proformas y productos, desde el chat y la IA. |
| **Cal.com** | Tus reservas de Cal.com en el calendario de Omnifox. |
| **Próximamente** | HubSpot, Salesforce, Shopify, Google Sheets, Calendly, Stripe Payments, Zoom, Google Meet, Teams, DocuSign, Dropbox Sign, PandaDoc, Notion, Gmail y Outlook. Mientras tanto: API + webhooks + n8n. |

21

CAPÍTULO 21

# Webhook personalizado: entrada y salida

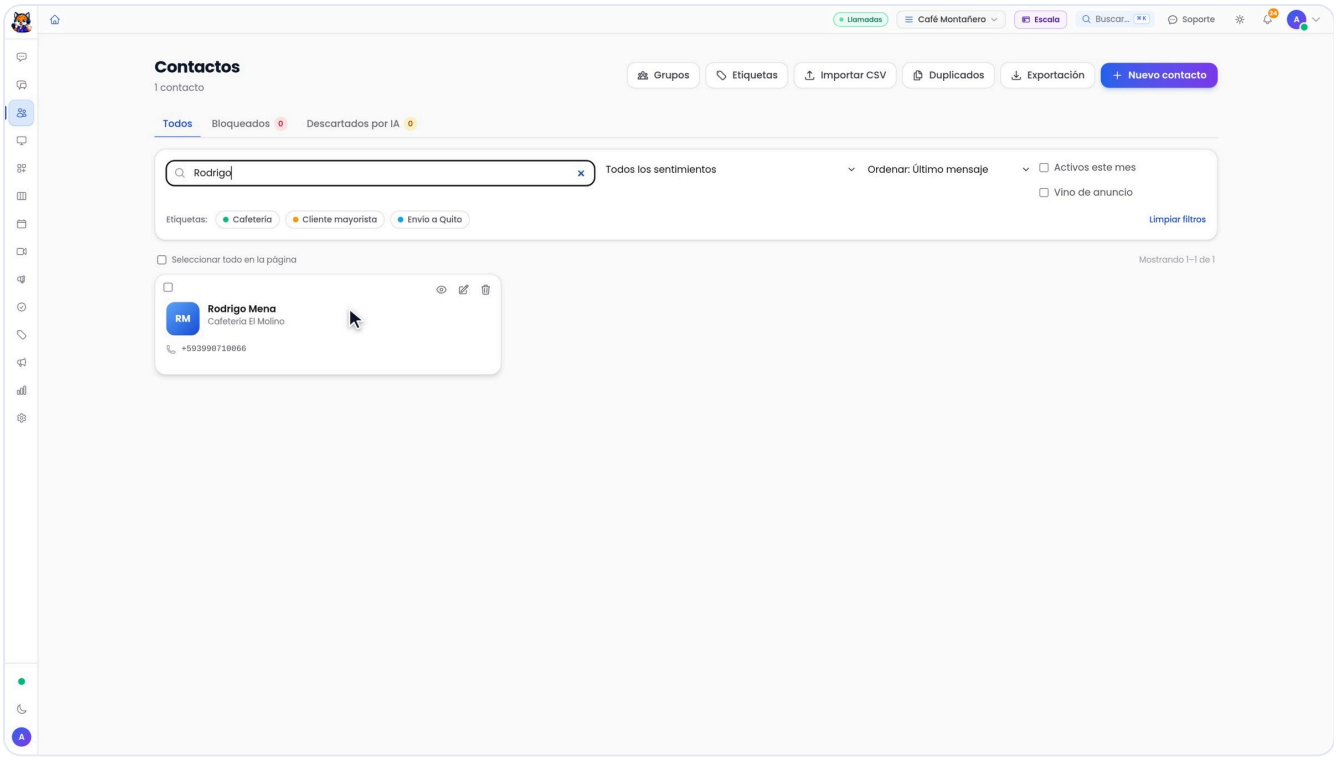
Una URL para recibir acciones y otra para enviar eventos.




Webhook personalizado: URL entrante, URL saliente, eventos y secreto de firma.

| Acción entrante ( `action` ) | Campos |
|---|---|
| `create_contact` | `display_name` , `email` , `phone` , `company` , `notes` , `external_id` |
| `update_contact` | El contacto y los campos a cambiar |
| `find_contact` | Teléfono o correo |
| `add_tag` | Contacto y etiqueta |
| `send_message` | Contacto o conversación y el texto |

```
CUERPO='{"action":"create_contact","display_name":"Rodrigo Mena","phone":"+593990710066"}'
FIRMA=$(printf '%s' "$CUERPO" | openssl dgst -sha256 -hmac "$SECRETO" | cut -d' ' -f2)
curl -s -X POST https://omnifox.io/api/v1/integrations/custom/<organización>/<espacio> \
  -H "Content-Type: application/json" -H "X-Omnifox-Signature: sha256=$FIRMA" -d "$CUERPO"
```




Una acción entrante firmada (create_contact) desde la terminal.

La **URL saliente** y los **Eventos a enviar** crean una suscripción de webhook real, firmada con el mismo **Secreto de firma**. **Pausar** la detiene y **Desconectar** la borra.

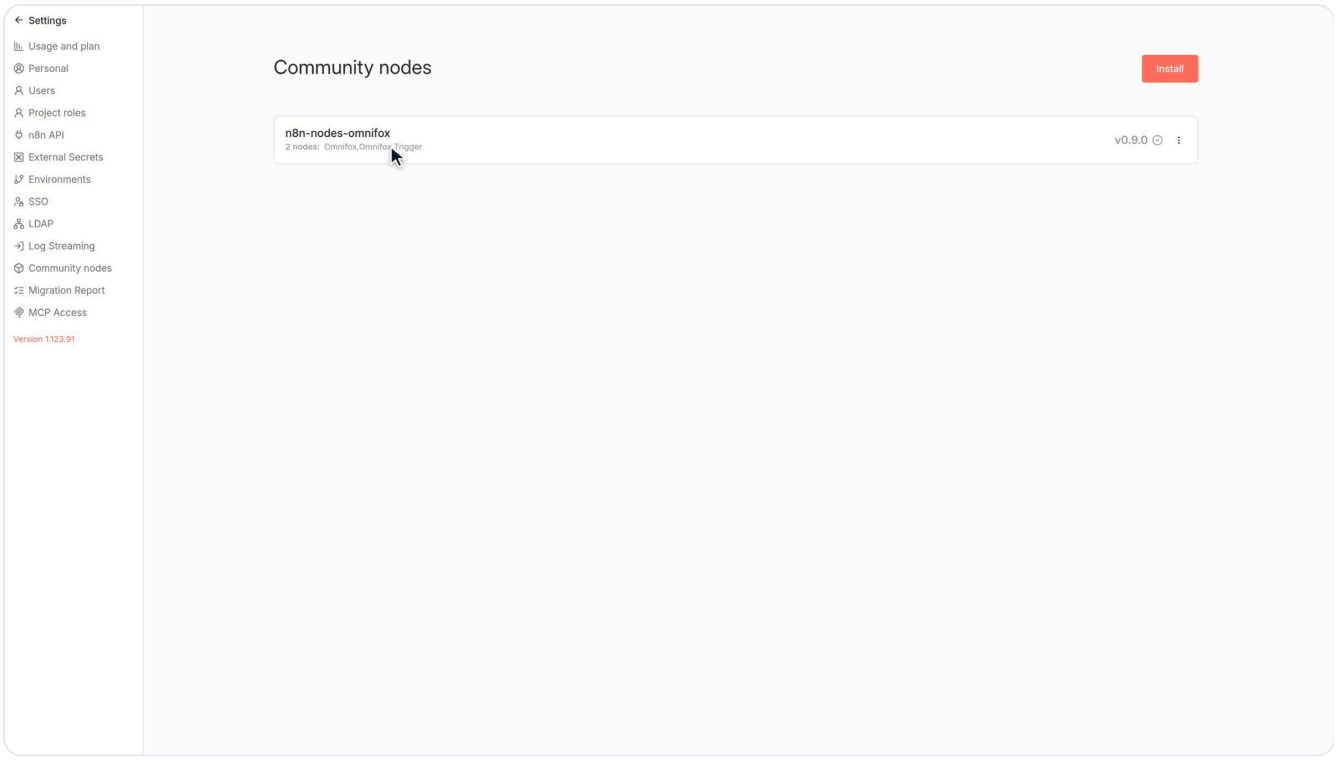
22

CAPÍTULO 22

# n8n: instalar el nodo de Omnifox

Un paquete de la comunidad, oficial de Omnifox.

1. En n8n ve a **Settings** → **Community nodes** y pulsa **Install**.
2. Escribe `n8n-nodes-omnifox`, acepta el aviso de riesgos y pulsa **Install**.
3. Quedan dos nodos: **Omnifox** (acciones) y **Omnifox Trigger** (eventos).




n8n → Settings → Community nodes con n8n-nodes-omnifox instalado.

**NOTA**

Funciona en n8n Cloud y en una instalación propia. Para el disparador, tu n8n debe ser accesible por https desde internet.

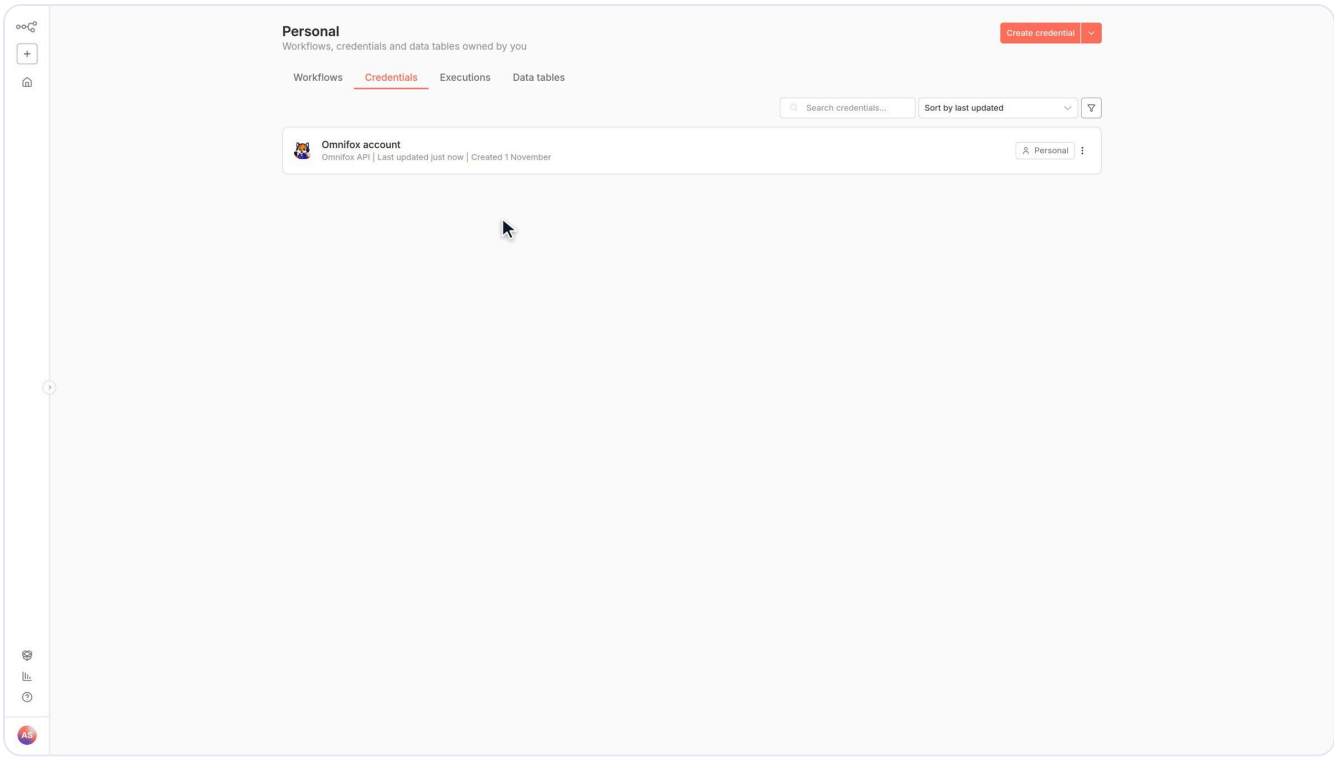
23

CAPÍTULO 23

# n8n: la credencial de Omnifox

Una credencial, todas las operaciones.

| Campo | Valor |
|---|---|
| **API Token** | Tu clave API (mejor de tipo bot y propia de n8n). |
| **Workspace ID** | 0: se toma de la clave. |
| **Base URL** | `https://app.omnifox.io` (sin /api: el nodo lo agrega). |

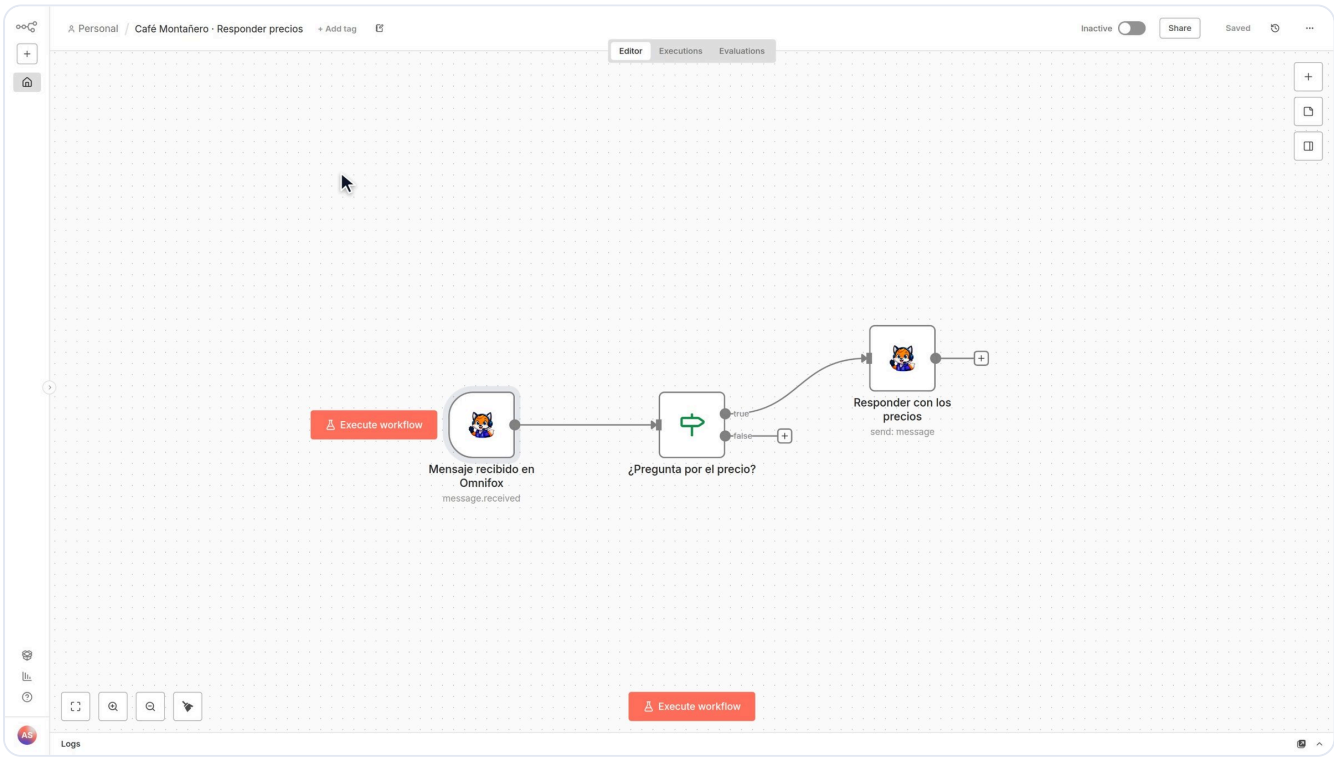



La credencial «Omnifox API» en n8n, con la conexión probada.

CAPÍTULO 24

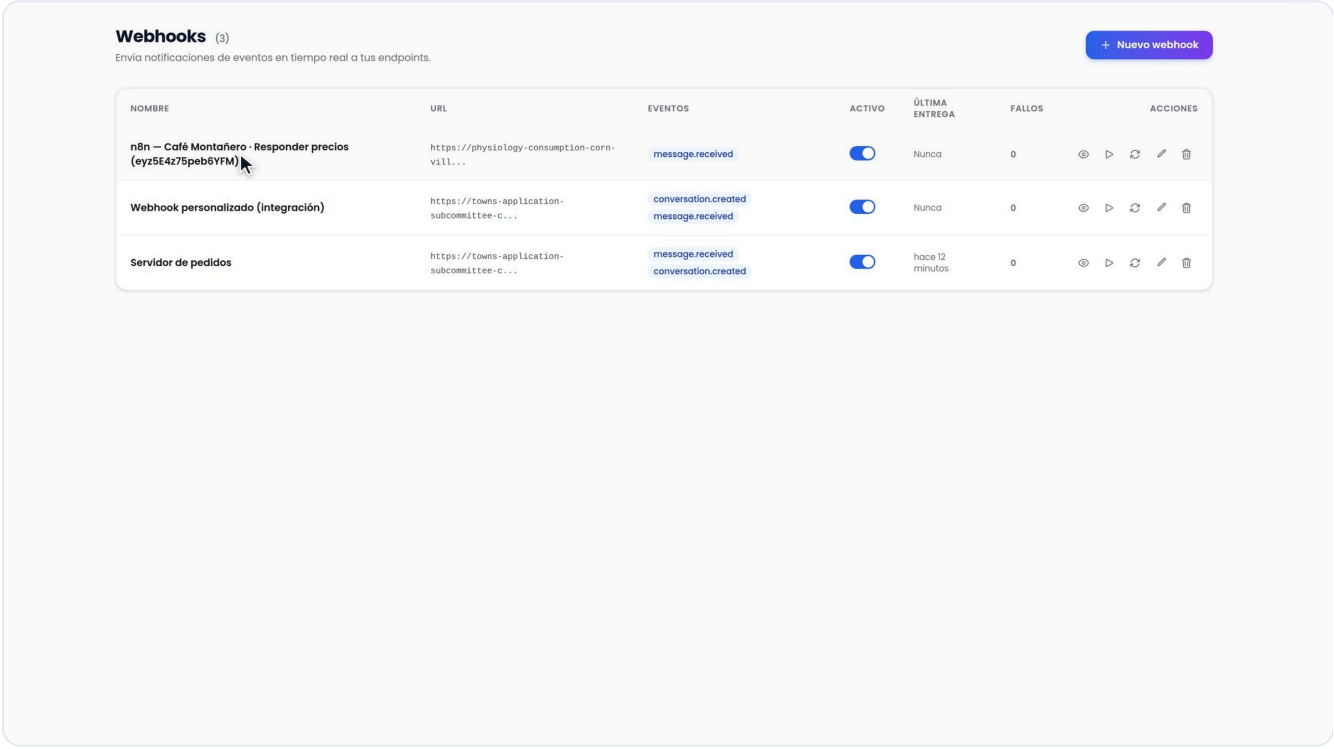
# 24 n8n: responder solo cuando preguntan el precio

Un flujo de tres nodos, probado de verdad.

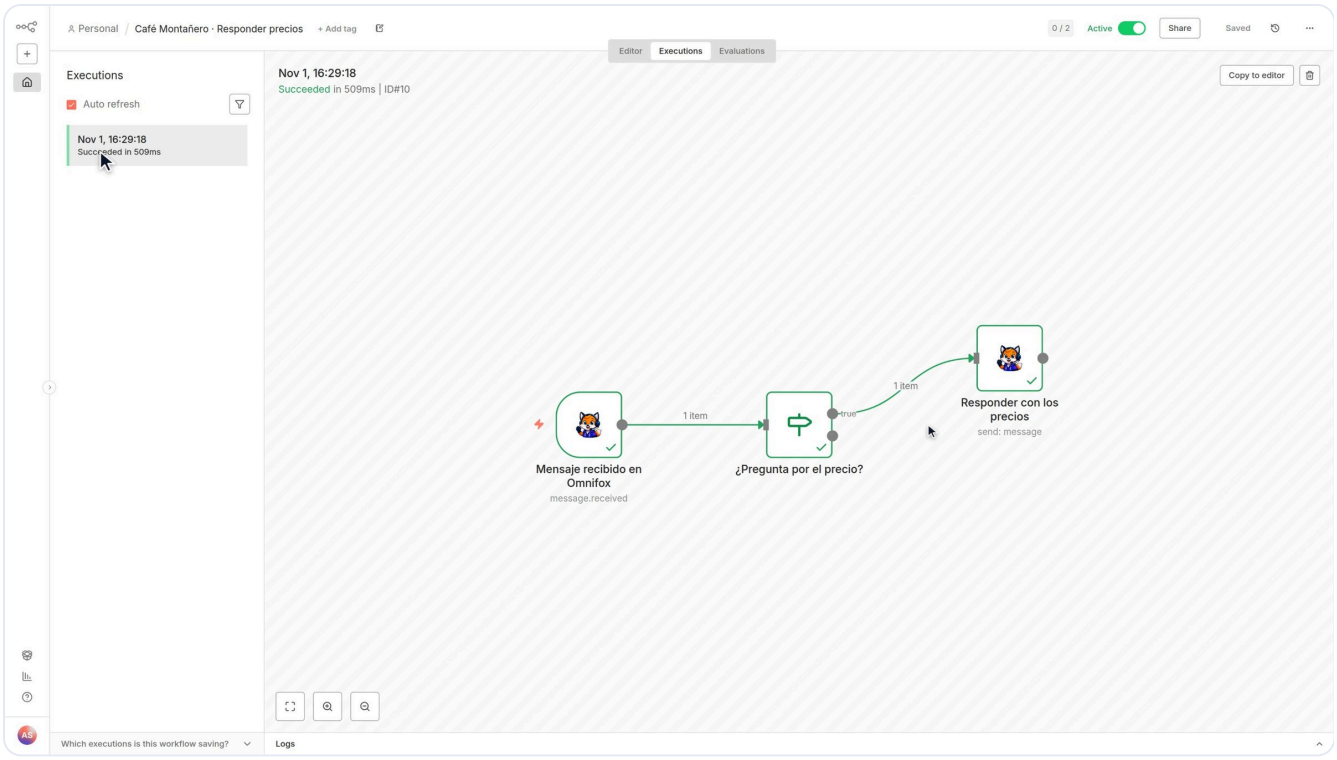



El flujo de n8n: disparador de Omnifox, condición y respuesta.

| Nodo | Configuración |
| --- | --- |
| **Omnifox Trigger** | Evento `message.received`. Al activarse crea la suscripción en Omnifox y verifica la firma de cada aviso. |
| **If** | Si `{{ $json.data.message.content_text }}` contiene «precio» (sin distinguir mayúsculas). |
| **Omnifox** | Recurso **Message**, operación **Send**, conversación `{{ ...data.conversation.id }}`, texto con la lista de precios. |




La suscripción que creó n8n, en Webhooks de Omnifox.



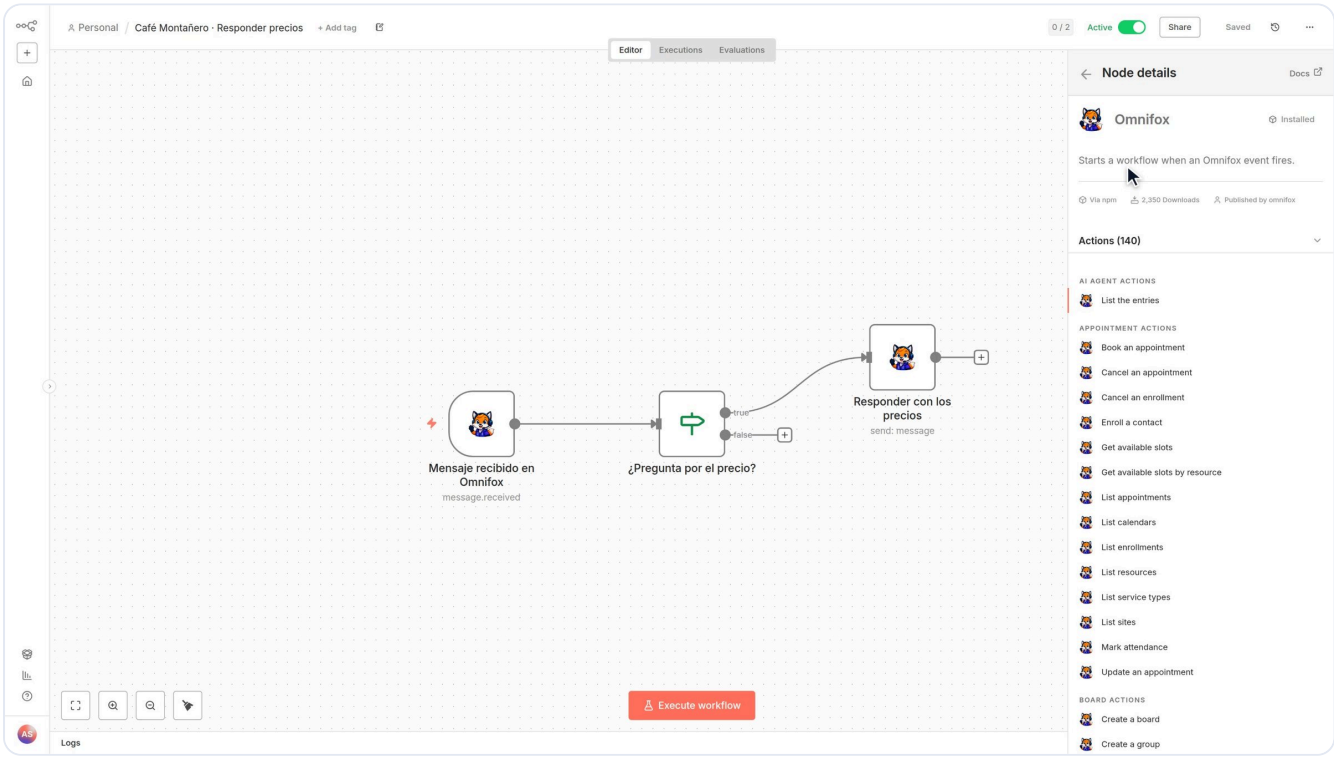

La ejecución del flujo en n8n.

25

CAPÍTULO 25

# n8n: todas las operaciones

140 operaciones en un solo nodo.




Las acciones del nodo Omnifox en n8n.

Conversaciones, contactos, mensajes, etiquetas, campañas, canales, equipos, agentes de IA, CRM, tableros, calendario, catálogo y pedidos. Las listas (canales, etiquetas, equipos, plantillas) se cargan desde tu cuenta.

| Error típico | Solución |
|---|---|
| Datos vacíos o 403 | La clave es de otro espacio de trabajo o le falta un alcance. |
| 404 en todas las llamadas | Base URL con `/api` al final: quítalo. |

26

CAPÍTULO 26

# MCP: que una IA opere Omnifox

72 herramientas para tu asistente de IA.

El servidor MCP de Omnifox ( `@omnifox/mcp-server` ) expone 72 herramientas: contactos, conversaciones, mensajes, etiquetas, campos, calendario y citas, cursos, catálogo, pedidos, flujos, difusiones, webhooks y reportes. Se ejecuta con `npx` y necesita dos variables:

```
OMNIFOX_API_KEY=tu-clave
OMNIFOX_BASE_URL=https://app.omnifox.io/api/v1
```

**OJO**

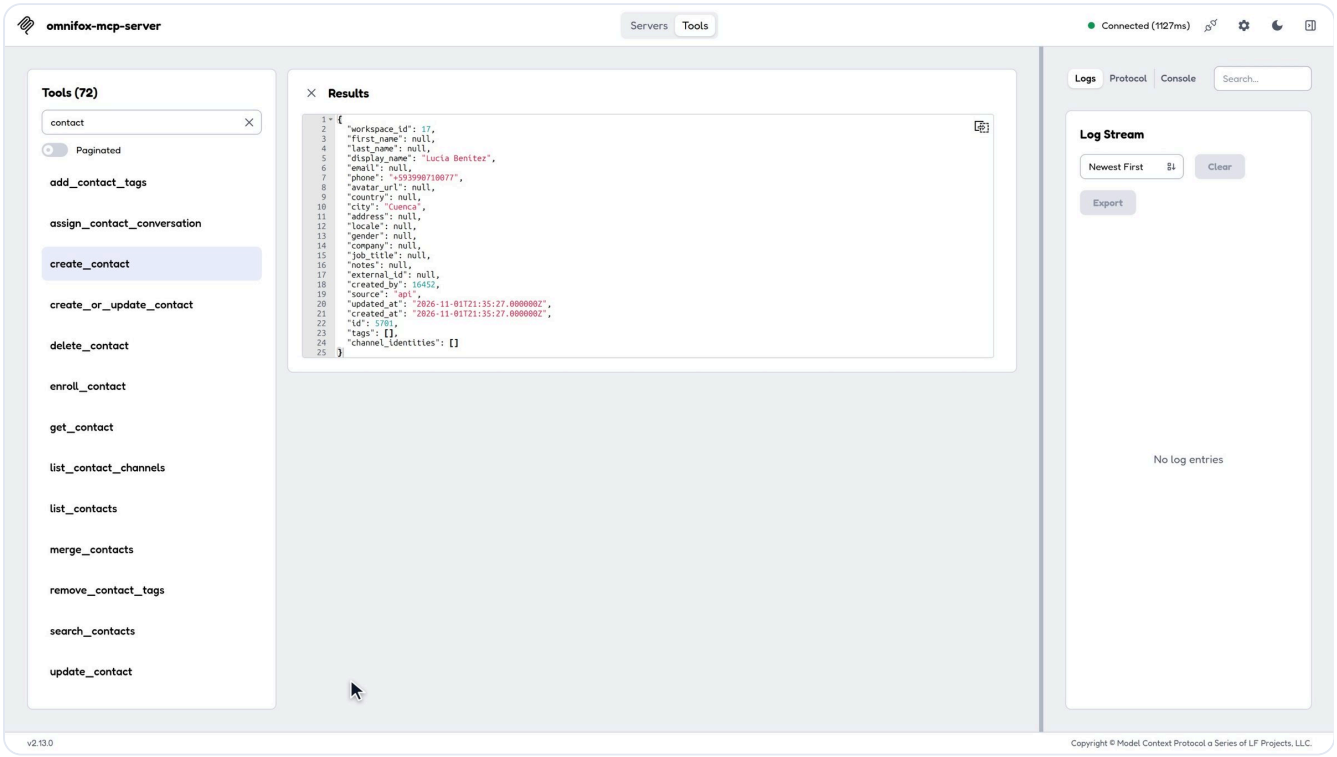
La IA solo puede hacer lo que permitan los alcances de la clave. Para un asistente, empieza con alcances de lectura.

27

CAPÍTULO 27

# Probar el servidor MCP con el Inspector

Prueba las herramientas sin conectar un asistente.




MCP Inspector: herramientas del servidor de Omnifox y el resultado de una llamada.

```
{
  "mcpServers": {
    "omnifox": {
      "command": "npx",
      "args": ["-y", "@omnifox/mcp-server"],
      "env": { "OMNIFOX_API_KEY": "tu-clave-api-de-omnifox", "OMNIFOX_BASE_URL":
"https://app.omnifox.io/api/v1" }
    }
  }
}
```

Ese bloque va en la configuración MCP de Claude Desktop, Cursor u otro cliente. Para servidores remotos, el MCP también funciona por HTTP.

28

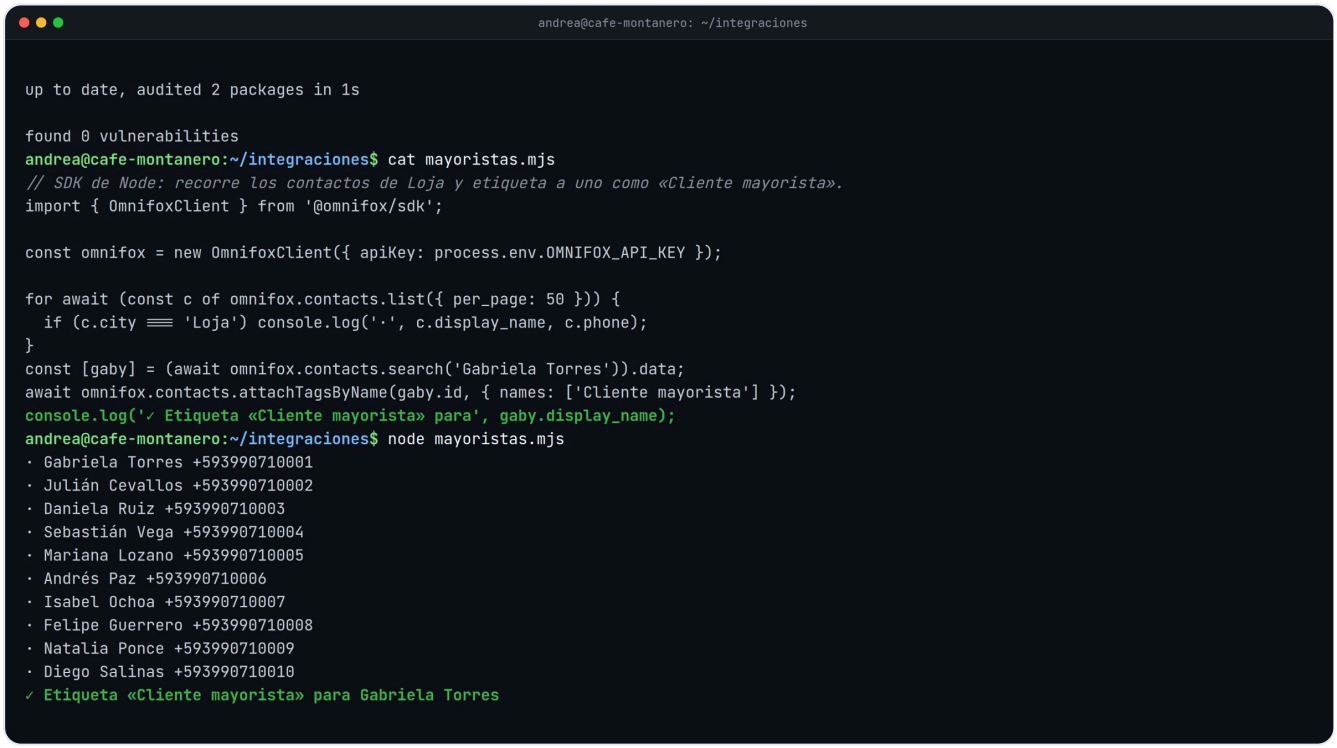
CAPÍTULO 28

# SDK oficiales: Node, Python y PHP

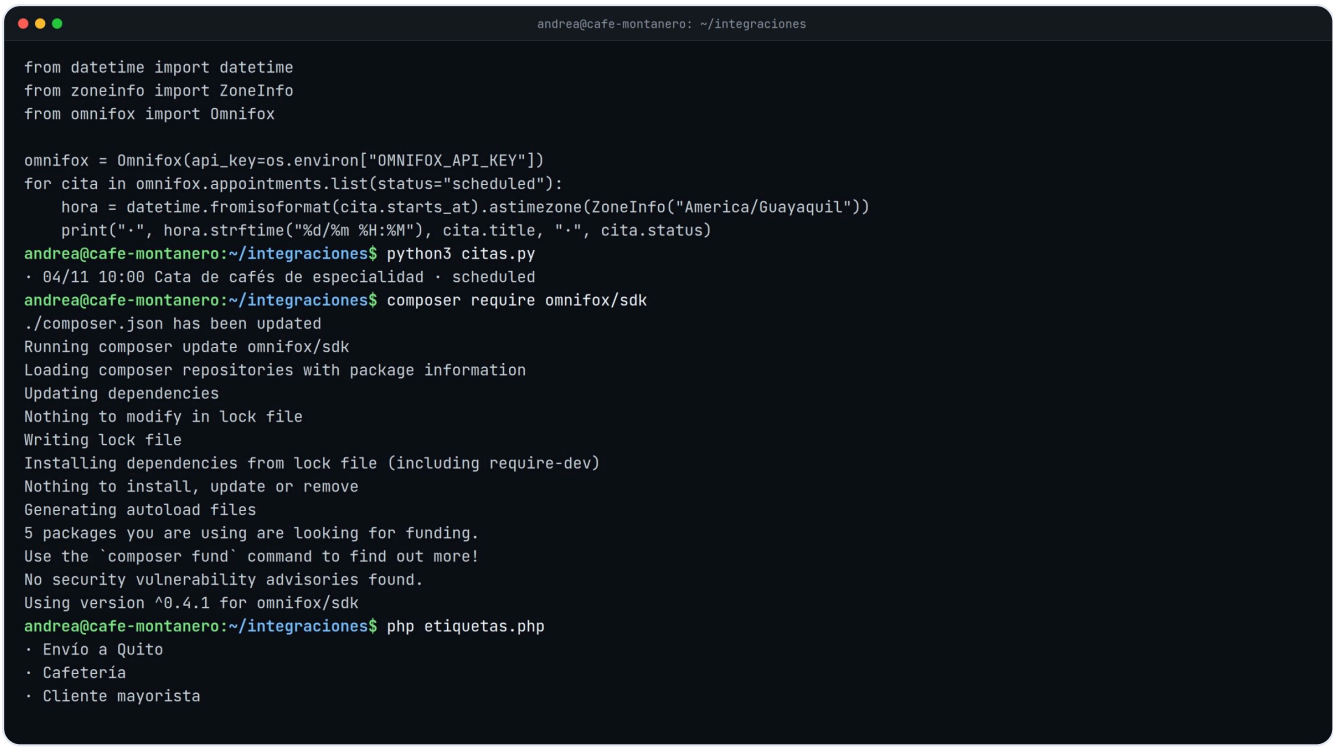
Autenticación, paginación, reintentos y errores, resueltos.

| Lenguaje | Instalar | Cliente |
|---|---|---|
| Node | `npm install @omnifox/sdk` | `new OmnifoxClient({ apiKey })` |
| Python | `pip install omnifox` | `Omnifox(api_key=...)` y `AsyncOmnifox` |
| PHP | `composer require omnifox/sdk` | `new \\Omnifox\\OmnifoxClient(['apiKey' => ...])` |

```
import { OmnifoxClient } from '@omnifox/sdk';
const omnifox = new OmnifoxClient({ apiKey: process.env.OMNIFOX_API_KEY });
for await (const c of omnifox.contacts.list({ per_page: 50 })) {
  if (c.city === 'Loja') console.log('·', c.display_name, c.phone);
}
const [gaby] = (await omnifox.contacts.search('Gabriela Torres')).data;
await omnifox.contacts.attachTagsByName(gaby.id, { names: ['Cliente mayorista'] });
```




El SDK de Node en acción.




Los SDK de Python y PHP.

# 29

CAPÍTULO 29

# Buenas prácticas de seguridad

Menor privilegio, secretos fuera del código y verificación siempre.

- Una clave por integración, con el mínimo de alcances y de tipo bot.
- Guárdala en variables de entorno o en un gestor de secretos; nunca en el código, en el repositorio ni en el navegador.
- Ponle vencimiento si es temporal y regenérala si alguien que la conocía se va.
- Webhooks: verifica la firma, responde 200 rápido y procesa en una cola; usa `X-Omnifox-Delivery-Id` para no duplicar.

30

CAPÍTULO 30

# Solución de problemas

Lo que suele fallar y cómo arreglarlo.

| Síntoma | Causa probable | Solución |
|---|---|---|
| 401 Unauthenticated | Clave mal copiada, vencida o regenerada | Copia de nuevo o crea una clave |
| 403 con `required_scope` | Falta el alcance | Edita la clave y agrégalo |
| 403 de plan | Tu plan no incluye la API o el módulo | Crece o superior; CRM y Proyectos según plan |
| Listas vacías | Clave de otro espacio de trabajo | Crea la clave en el espacio correcto |
| El webhook no llega | URL no pública/https, servidor caído o circuito abierto | Revisa el historial de entregas y el código HTTP; cierra el circuito |
| Firma inválida | Cuerpo re-serializado o secreto viejo | Usa el cuerpo crudo y el secreto actual |

¿Necesitas ayuda? Escríbenos desde el chat de soporte de la app o en **omnifox.io/help**.