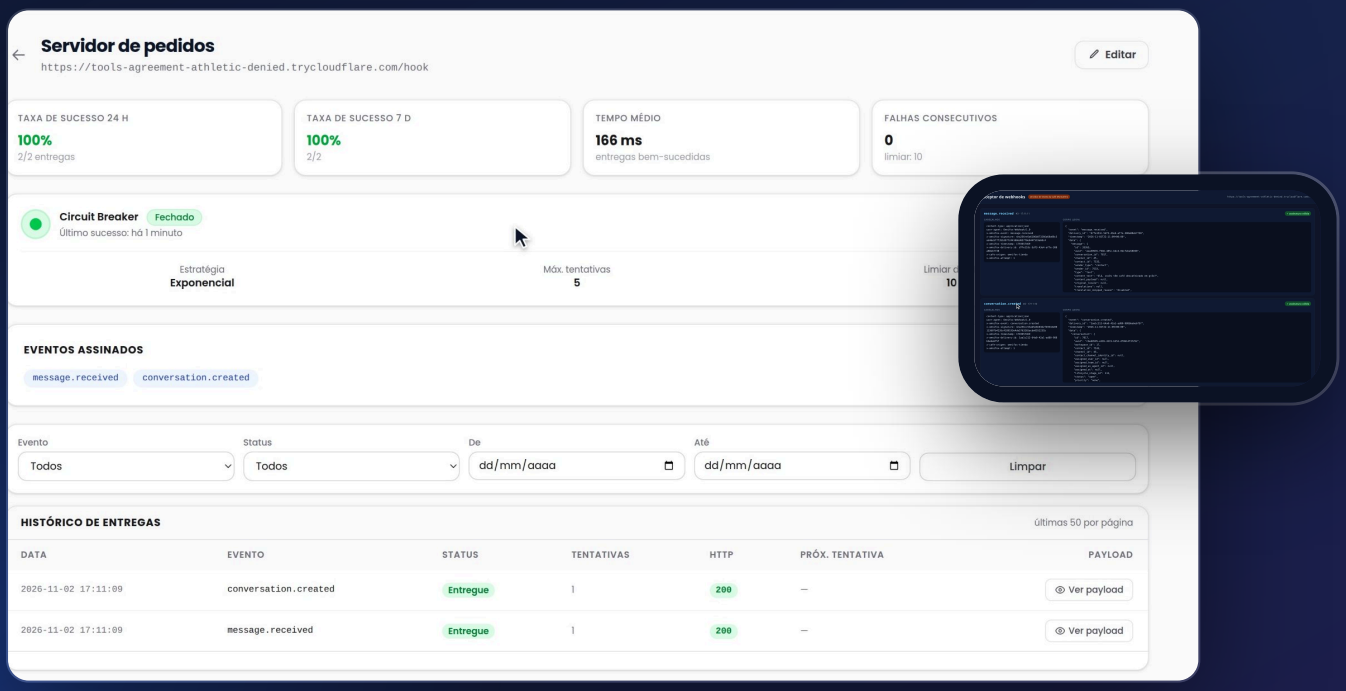
Omnifox

INTEGRAÇÕES E API · GUIA COMPLETO

Integrações e API

Chaves de API, chamadas reais, webhooks assinados, n8n, MCP e os SDKs de Node, Python e PHP.



CONTEÚDO

Índice

Como ler este manual

As capturas são telas reais do Omnifox, em português, com um negócio de exemplo: a **Café Montañero**, uma torrefação fictícia de Loja que vende café pelo site. Tudo o que aparece foi feito de verdade: a chave de API foi criada durante as capturas, os comandos rodaram contra o Omnifox, os webhooks chegaram a um servidor receptor de testes, e o n8n e o servidor MCP rodaram as versões publicadas. Os clientes são inventados.

Os nomes de botões e campos estão em **negrito**. Os comandos usam `curl` e `jq`, e duas variáveis de ambiente: `OMNIFOX_API_KEY` (a sua chave) e `API=https://app.omnifox.io/api/v1`. Este manual acompanha o vídeo «Integraciones y API · Guía completa» do canal do Omnifox no YouTube (em espanhol).

01 As formas de conectar o Omnifox 3
02 Requisitos: plano e permissões 4
03 Criar uma chave de API 5
04 Escopos: o que a chave pode fazer 6
05 Copiar a chave uma única vez 7
06 A lista de chaves: editar, regenerar e excluir 8
07 A documentação da API 9
08 A sua primeira chamada 9
09 Contatos pela API 10
10 Escopos em ação e erros 12
11 Conversas e mensagens pela API 13
12 CRM e calendário pela API 14
13 Paginação e limites 16
14 Webhooks: criar uma assinatura 16
15 Segredo de assinatura, novas tentativas e cabeçalhos 18
16 Entrega de teste 19
17 Um evento real 21
18 Verificar a assinatura no seu servidor 22
19 Histórico de entregas 23
20 A página de integrações 24
21 Webhook personalizado: entrada e saída 25
22 n8n: instalar o nó do Omnifox 26
23 n8n: a credencial do Omnifox 27
24 n8n: responder só quando perguntam o preço 28
25 n8n: todas as operações 30
26 MCP: que uma IA opere o Omnifox 31
27 Testar o servidor MCP com o Inspector 32
28 SDKs oficiais: Node, Python e PHP 33
29 Boas práticas de segurança 34
30 Solução de problemas 35

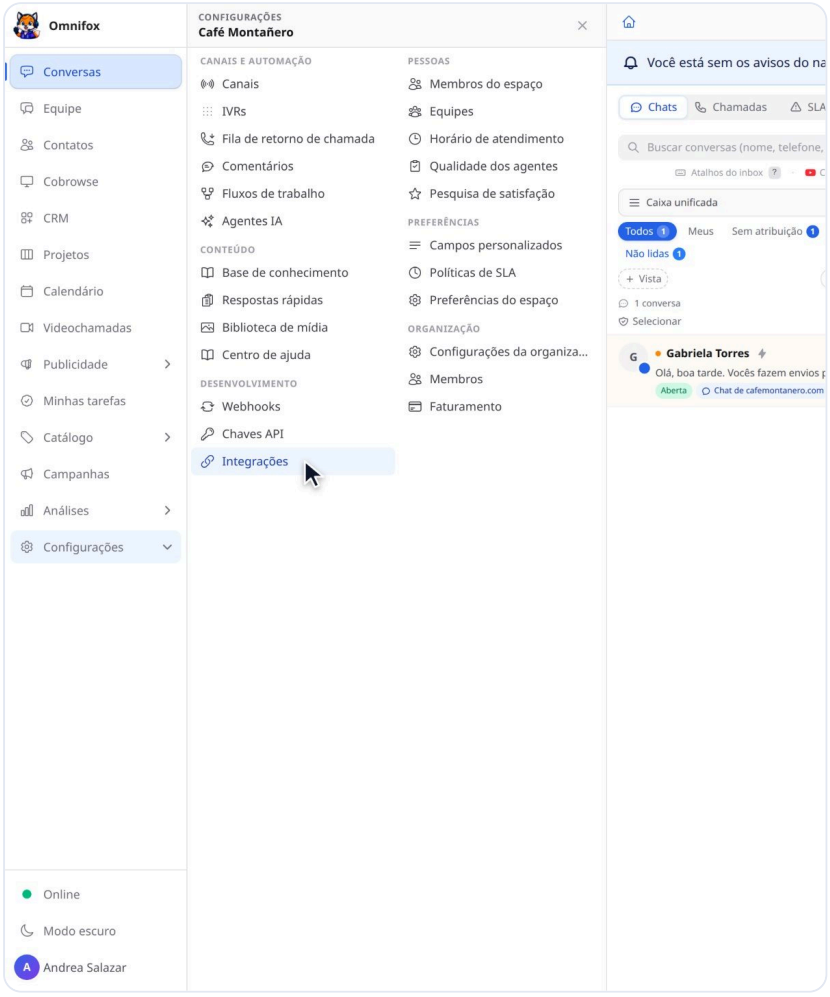
01

CAPÍTULO 1

As formas de conectar o Omnifox

O seu sistema fala com o Omnifox pela API; o Omnifox avisa o seu sistema pelos webhooks.

Tudo o que é para desenvolvedores está em **Configurações → Desenvolvimento**: **Webhooks**, **Chaves de API** e **Integrações**.



Configurações → Desenvolvimento: Webhooks, Chaves de API e Integrações.

Ferramenta	Sentido	Para quê
API REST	Seu sistema → Omnifox	Criar e atualizar contatos, responder conversas, criar oportunidades, agendar horários, ler relatórios...
Webhooks	Omnifox → seu sistema	Receber um aviso na hora quando chega uma mensagem, um contato é criado, um horário é agendado...
Integrações	Os dois	Conectores prontos: n8n, Make, Zapier, Webhook personalizado, Azur, Cal.com.
n8n	Os dois	Nós oficiais do Omnifox (140 operações + um gatilho) para automatizar sem programar.
MCP	IA → Omnifox	Que um assistente de IA (Claude, Cursor, ChatGPT...) opere o Omnifox com 72 ferramentas.
SDK	Seu código → Omnifox	Bibliotecas oficiais para Node, Python e PHP.

DICA

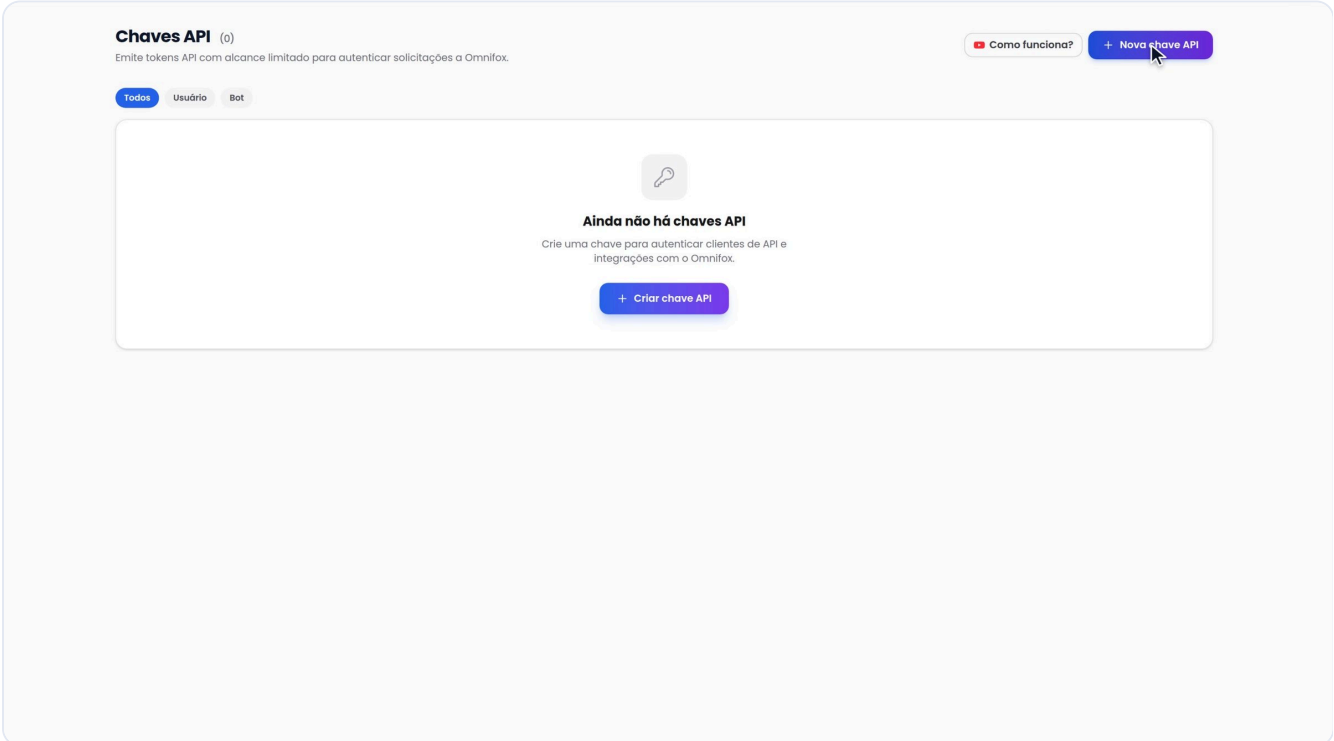
Todos usam a mesma chave de API. Se você criar uma chave por integração, pode revogar uma sem afetar as outras.

02

CAPÍTULO 2

Requisitos: plano e permissões

A API vem no Crece, no Escala e no Enterprise.



Chaves de API, ainda vazia.

- A API e os webhooks estão nos planos **Crece**, **Escala** e **Enterprise**. No Lite, Impulsa e Conecta, a API responde que o seu plano não a inclui.
- Para criar chaves de API é preciso a permissão de gerenciar a configuração do espaço. Os webhooks só são criados por um **administrador**.
- Cada chave fica presa a **um** espaço de trabalho: o que você fizer com ela acontece nesse espaço.

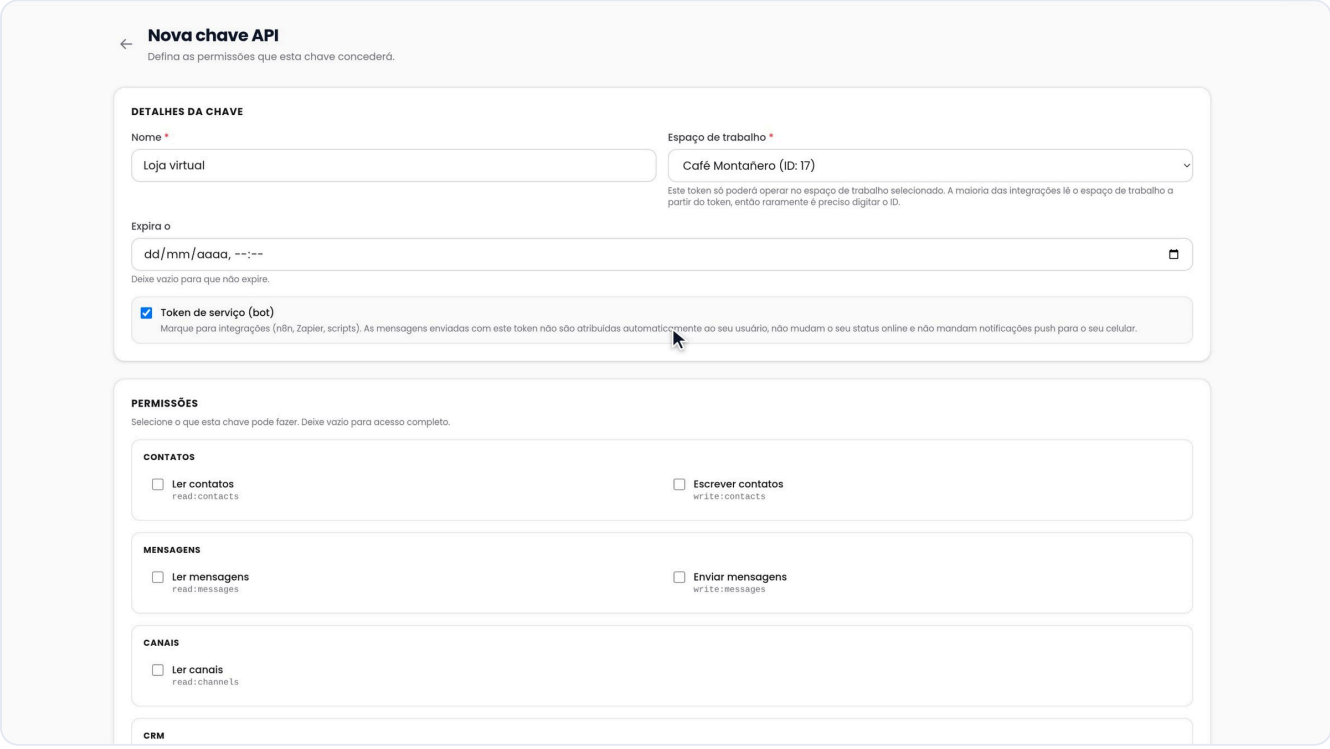
03

CAPÍTULO 3

Criar uma chave de API

Nome, espaço de trabalho, vencimento e tipo.

1. Vá em **Configurações → Chaves de API** e clique em **Nova chave de API**.
2. Escreva um **Nome** que diga para que serve (por exemplo, «Loja virtual»).
3. Escolha o **Espaço de trabalho**. A chave só poderá trabalhar nesse espaço, por isso quase nunca é preciso enviar o número do espaço nas chamadas.
4. Opcional: **Expira em**. Vazio = não vence. Para um fornecedor externo ou um teste, coloque uma data.
5. Marque **Token de serviço (bot)** para integrações.



Nova chave de API: nome, espaço de trabalho, vencimento e token de serviço (bot).

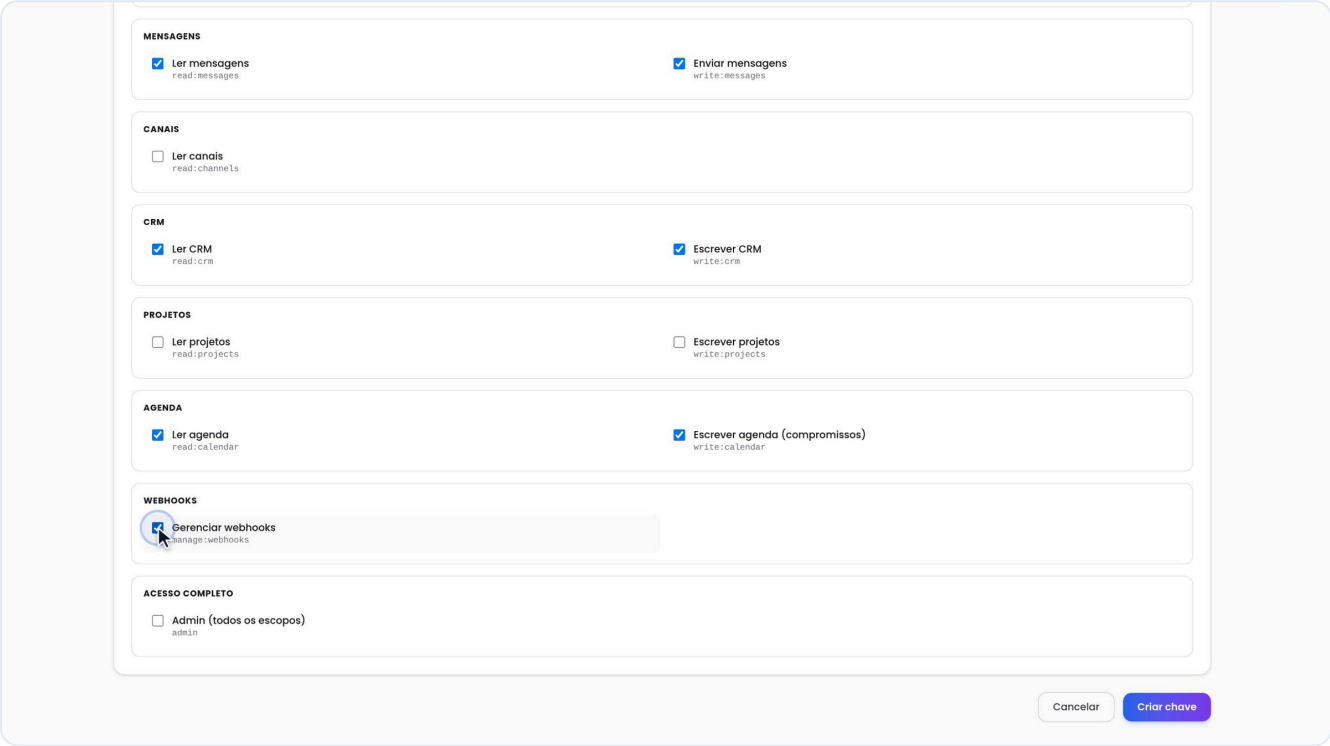
Tipo	Comportamento
Usuário	Age como você: o que ela envia é atribuído ao seu usuário e conta como atividade sua.
Bot (token de serviço)	O que ela envia não é atribuído ao seu usuário, não muda o seu status online e não manda notificações push. Recomendado para n8n, Zapier, scripts e servidores.

04

CAPÍTULO 4

Escopos: o que a chave pode fazer

Dê a cada chave só o que ela precisa.



Os escopos marcados para a chave «Loja virtual». Canais fica desmarcado de propósito.

Módulo	Escopos	Cobre
Contatos	`read:contacts` · `write:contacts`	Contatos, busca, etiquetas do contato, mesclar.
Mensagens	`read:messages` · `write:messages`	Conversas e mensagens: ler, enviar, atribuir, fechar.
Canais	`read:channels`	Ver os canais do espaço.
CRM	`read:crm` · `write:crm`	Funis, etapas, oportunidades, empresas e atividades.
Projetos	`read:projects` · `write:projects`	Quadros, itens e colunas.
Calendário	`read:calendar` · `write:calendar`	Calendários, serviços, horários livres e agendamentos.
Webhooks	`manage:webhooks`	Criar, ver, rotacionar e excluir assinaturas (usado pelo gatilho do n8n).
Acesso completo	`admin`	Todos os escopos.

- Se você não marcar nenhum, a chave tem **acesso completo**.

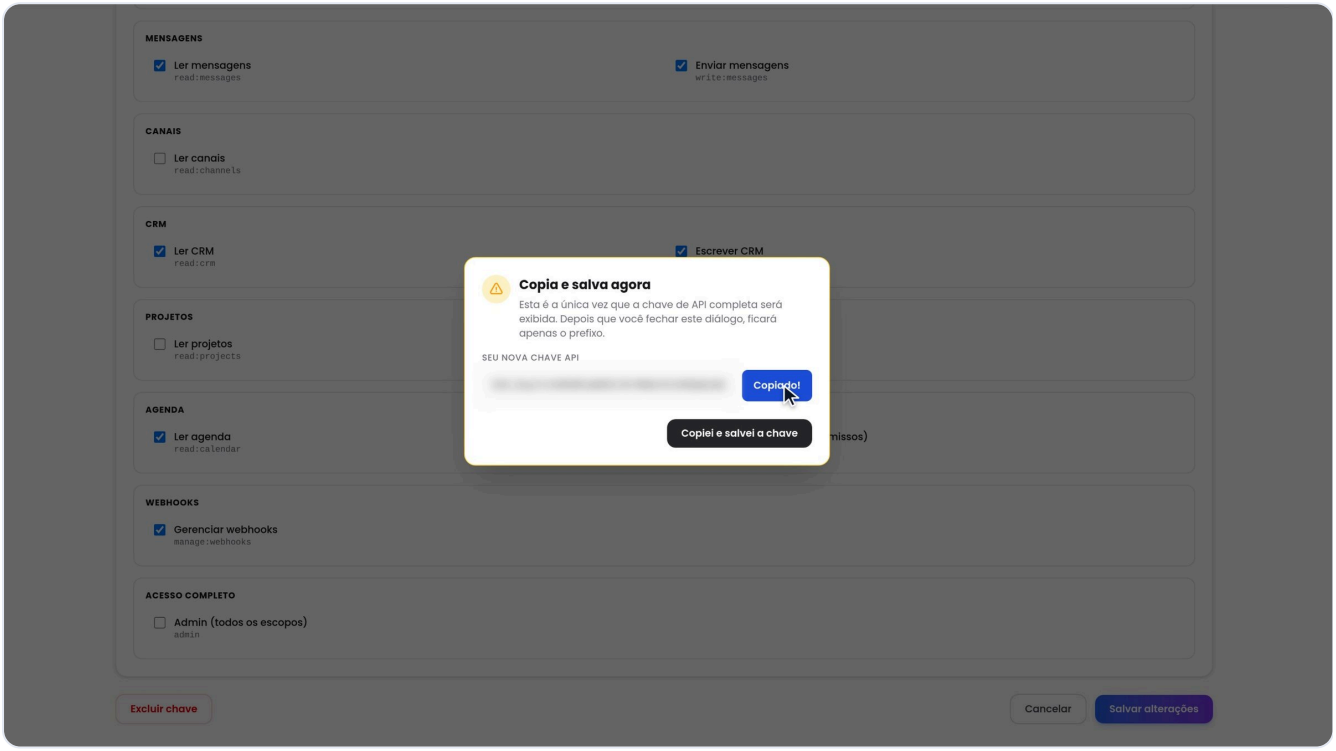
- **Escrever** inclui **ler** no mesmo módulo.
- Uma chamada sem o escopo necessário responde **403** com o escopo que falta (capítulo 10).

05

CAPÍTULO 5

Copiar a chave uma única vez

Aparece uma vez. Guarde bem.



A chave aparece uma única vez (nas capturas, desfocada).

Ao clicar em **Criar chave**, o Omnifox mostra a chave completa **uma única vez**. Clique em **Copiar**, guarde no gerenciador de segredos ou nas variáveis de ambiente do seu servidor e confirme que copiou e guardou a chave. Depois só aparece o prefixo.

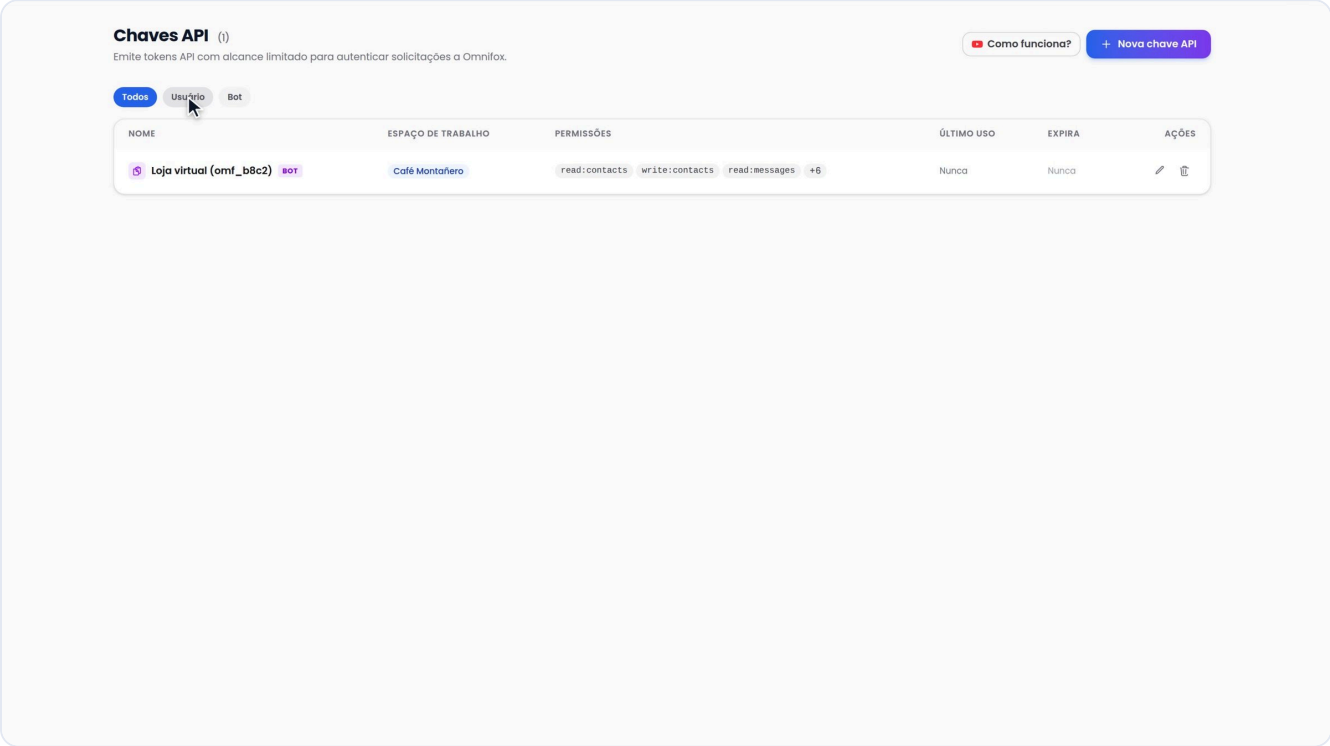
> **ATENÇÃO**
> Se você a perder, não dá para recuperar: regenere-a (capítulo 6). A anterior deixa de funcionar nesse momento.

06

CAPÍTULO 6

A lista de chaves: editar, regenerar e excluir

Editar sem quebrar, regenerar se vazou, excluir para cortar o acesso.



A lista de chaves: tipo, espaço, escopos, último uso e vencimento.

Ação	O que faz
Filtros **Todos / Usuário / Bot**	Mostra as chaves de cada tipo.
Editar (lápis)	Muda o nome, os escopos ou o vencimento. O token continua o mesmo.
Regenerar (dentro de editar)	Cria um token novo com as mesmas permissões e o mostra uma vez. O anterior deixa de funcionar.
Excluir (lixeira)	Revoga a chave na hora.
Último uso	Quando foi usada pela última vez: útil para detectar chaves abandonadas.

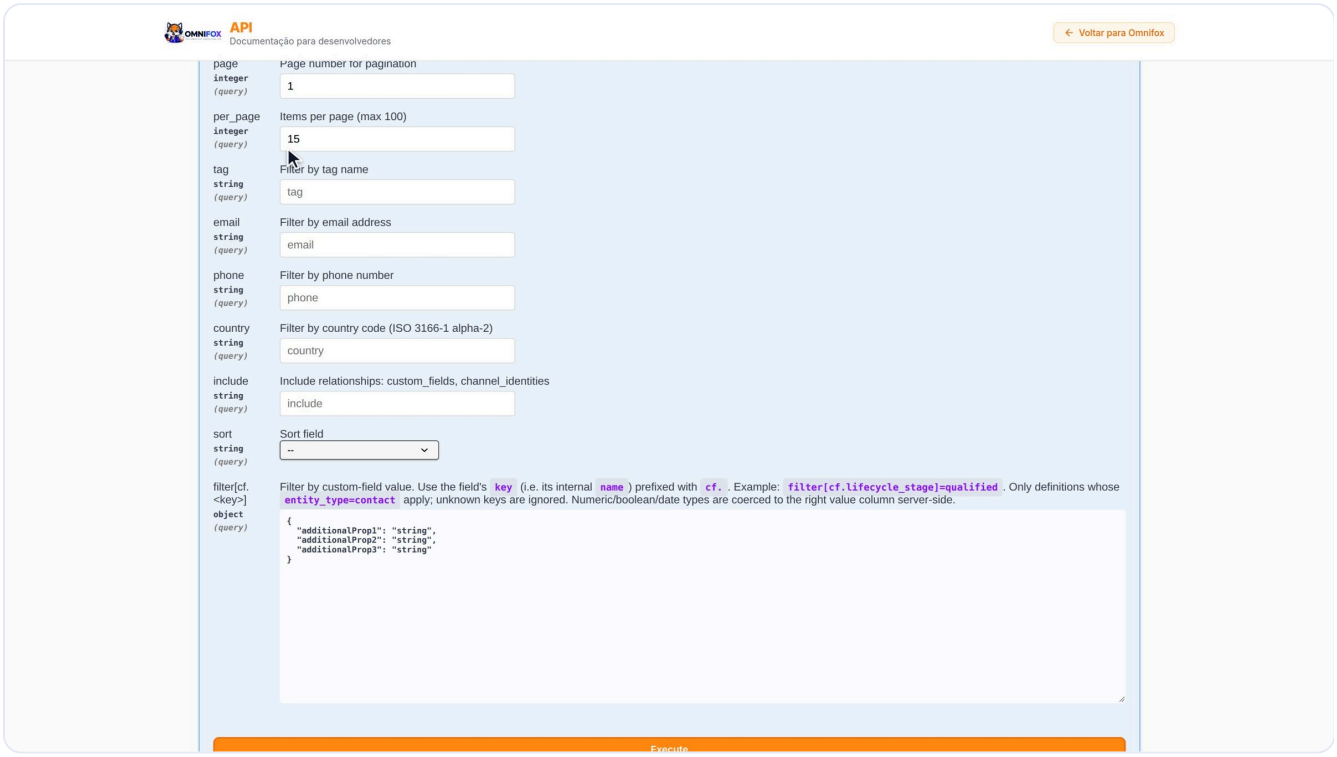
07

CAPÍTULO 7

A documentação da API

Cada endpoint, com parâmetros, exemplos e erros.

A documentação está em **omnifox.io/docs**. No topo, atalhos para os nós do n8n, o servidor MCP e os SDKs. Abaixo, a referência completa agrupada por módulo.



A referência da API em /docs, com cada endpoint e o seu exemplo.

- **Authorize**: cole a sua chave e teste qualquer chamada na mesma página.
- **openapi.json**: baixe para Postman, Insomnia ou um gerador de código.

08

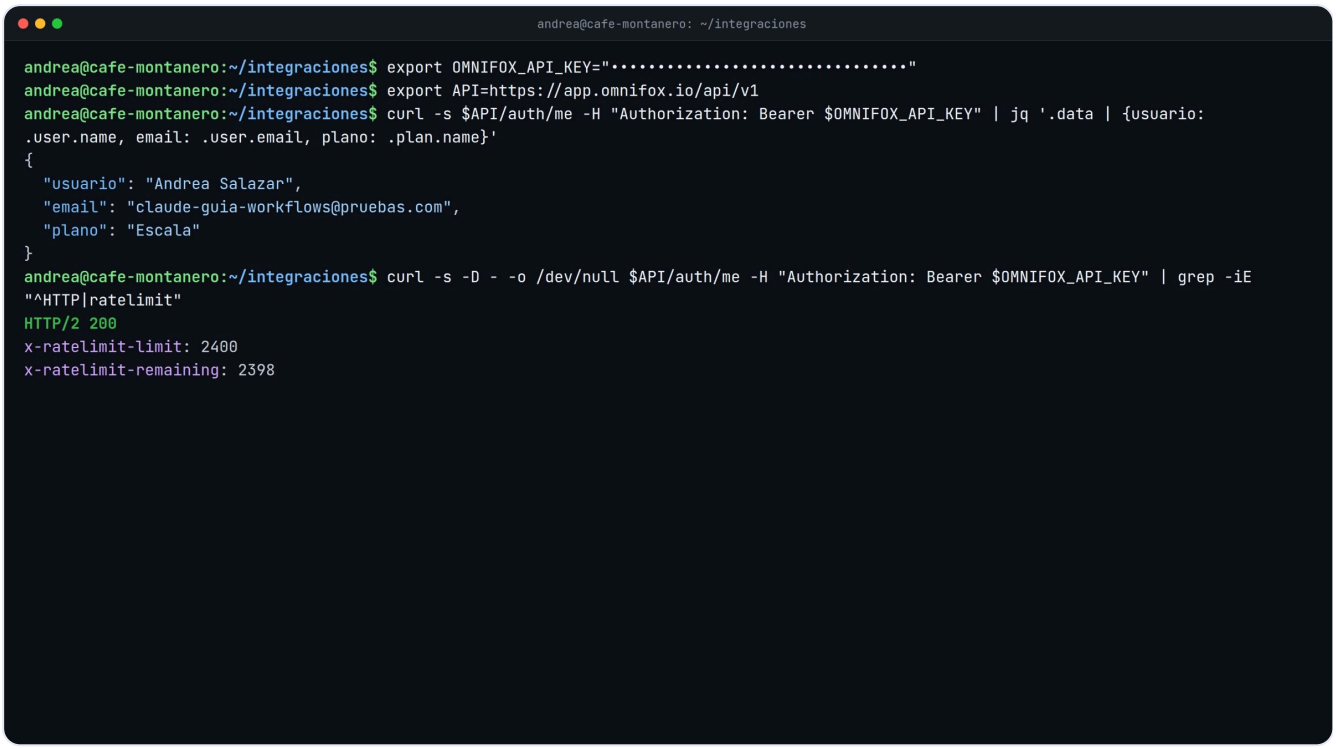
CAPÍTULO 8

A sua primeira chamada

Um cabeçalho, e a API responde.

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

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



Primeira chamada pelo terminal e os cabeçalhos de limite de uso.

Todas as respostas corretas têm o mesmo formato: `success`, `message` e `data`. As listas trazem também a paginação (capítulo 13). Os cabeçalhos `X-RateLimit-Limit` e `X-RateLimit-Remaining` dizem quantas chamadas por minuto o seu plano permite e quantas restam.

09

CAPÍTULO 9

Contatos pela API

Criar, buscar e atualizar.

```
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"}'

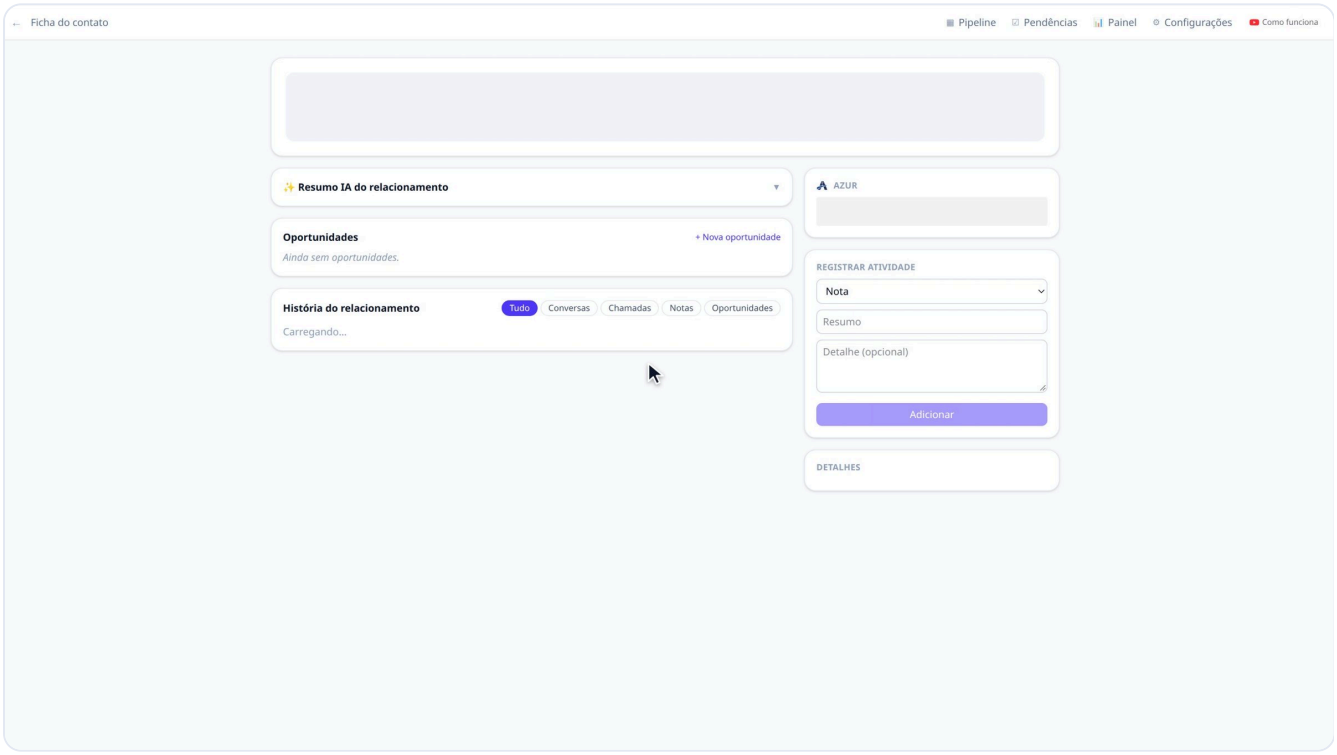
curl -s "$API/contacts/search?q=Valentina" -H "Authorization: Bearer $OMNIFOX_API_KEY"

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 $OMNIFOX_API_KEY" -H "Content-Type:
application/json" -d '{"display_name":"Valentina
Ríos","first_name":"Valentina","last_name":"Ríos","phone":"+593990710055","email":"valentina.rios@example.com","city":"Quito"}' | jq
'.data | {id, display_name, phone, city}'
{
  "id": 7532,
  "display_name": "Valentina Ríos",
  "phone": "+593990710055",
  "city": "Quito"
}
andrea@cafe-montanero:~/integraciones$ ID=$(curl -s "$API/contacts/search?q=Valentina" -H "Authorization: Bearer $OMNIFOX_API_KEY" | jq
'.data[0].id'); echo "id: $ID"
id: 7532
andrea@cafe-montanero:~/integraciones$ 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": 7532,
  "display_name": "Valentina Ríos",
  "company": "Cafetería La Ronda"
}
```

Criar, buscar e atualizar um contato pela API.



O contato criado pela API, em Contatos.

DICA

Para não duplicar clientes use `POST /contacts/upsert` : cria ou atualiza conforme o telefone ou o e-mail.

10

CAPÍTULO 10

Escopos em ação e erros

Os erros dizem exatamente o que aconteceu.



Um 403 com o escopo que falta, um 401 com uma chave inválida e um 422 de validação.

Código	Quando	O que fazer
401	A chave não existe, venceu ou foi revogada/regenerada.	Confira se foi bem copiada; crie ou regenere uma.
403	Falta um escopo (`required_scope`) ou o seu plano não inclui essa função.	Adicione o escopo à chave ou confira o plano.
404	O recurso não existe no espaço da chave.	Confira o ID e se a chave é do espaço certo.
422	Um dado não é válido.	A resposta traz o campo e o motivo.
429	Você passou do limite por minuto.	Espere alguns segundos e tente de novo (os SDKs fazem sozinhos).

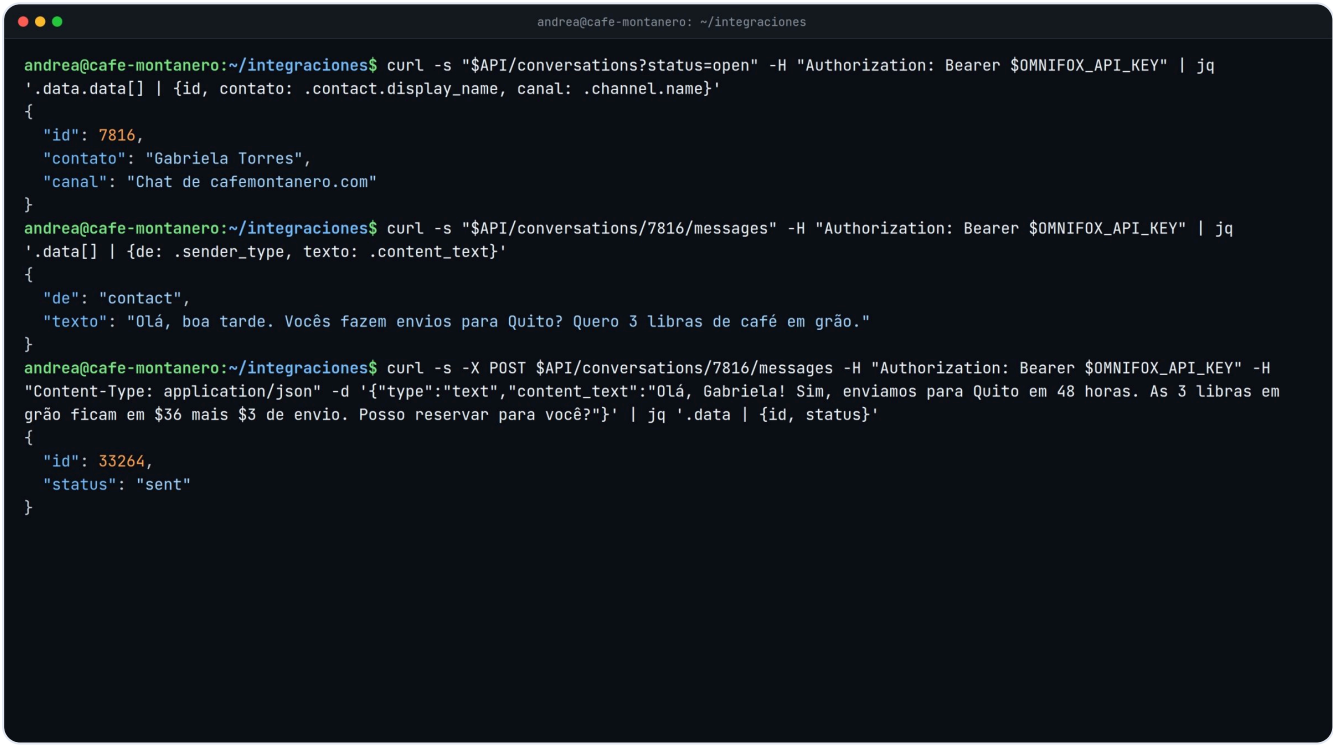
11

CAPÍTULO 11

Conversas e mensagens pela API

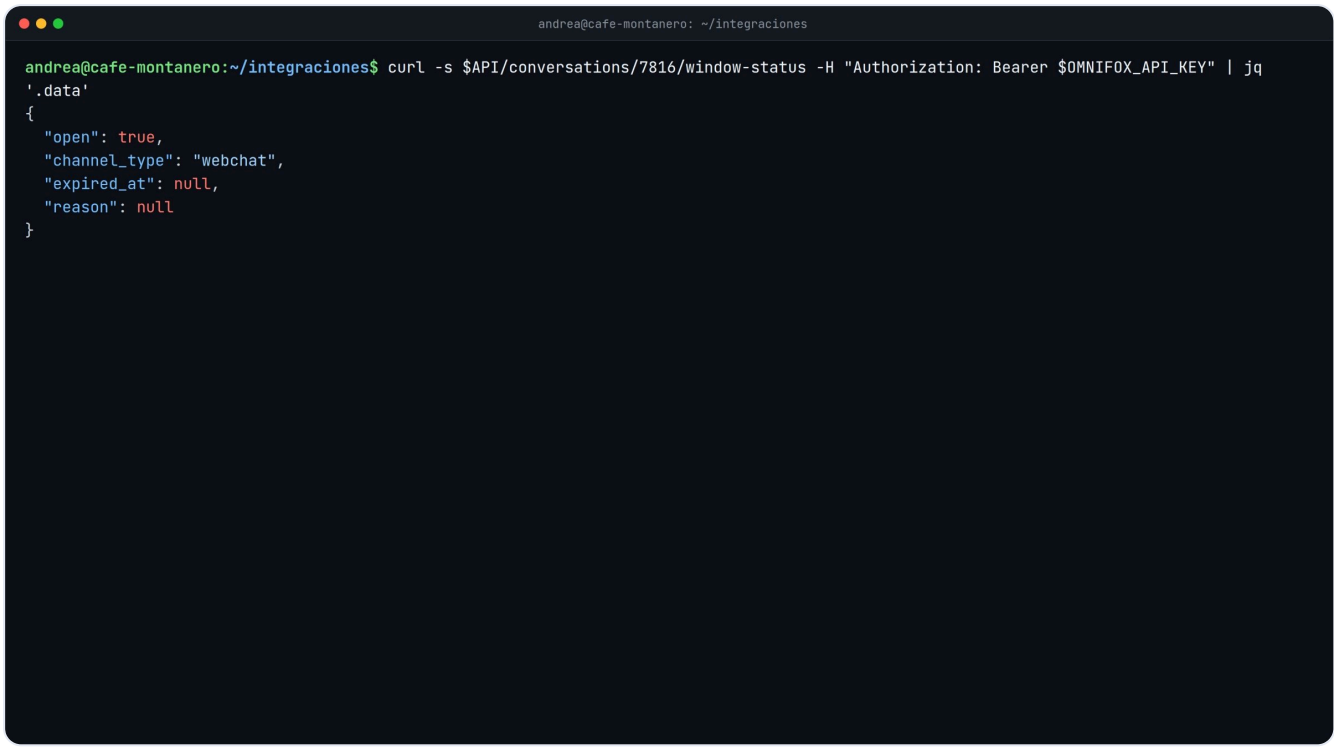
Responda aos seus clientes pelo seu sistema, pelo mesmo canal.

```
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":"Olá, Gabriela! Sim, enviamos para Quito em 48 horas."}'
curl -s $API/conversations/$CID/window-status -H "Authorization: Bearer $OMNIFOX_API_KEY"
```



Ler a conversa e responder pela API.

O Omnifox envia a resposta pelo canal por onde o cliente escreveu (no exemplo, o chat do site, onde a resposta aparece na hora).



A janela de 24 horas (window-status) da conversa.

ATENÇÃO

No WhatsApp você só pode mandar texto livre dentro da janela de 24 horas desde a última mensagem do cliente. Consulte o `window-status` antes de enviar: se estiver fechada, use um modelo aprovado.

12

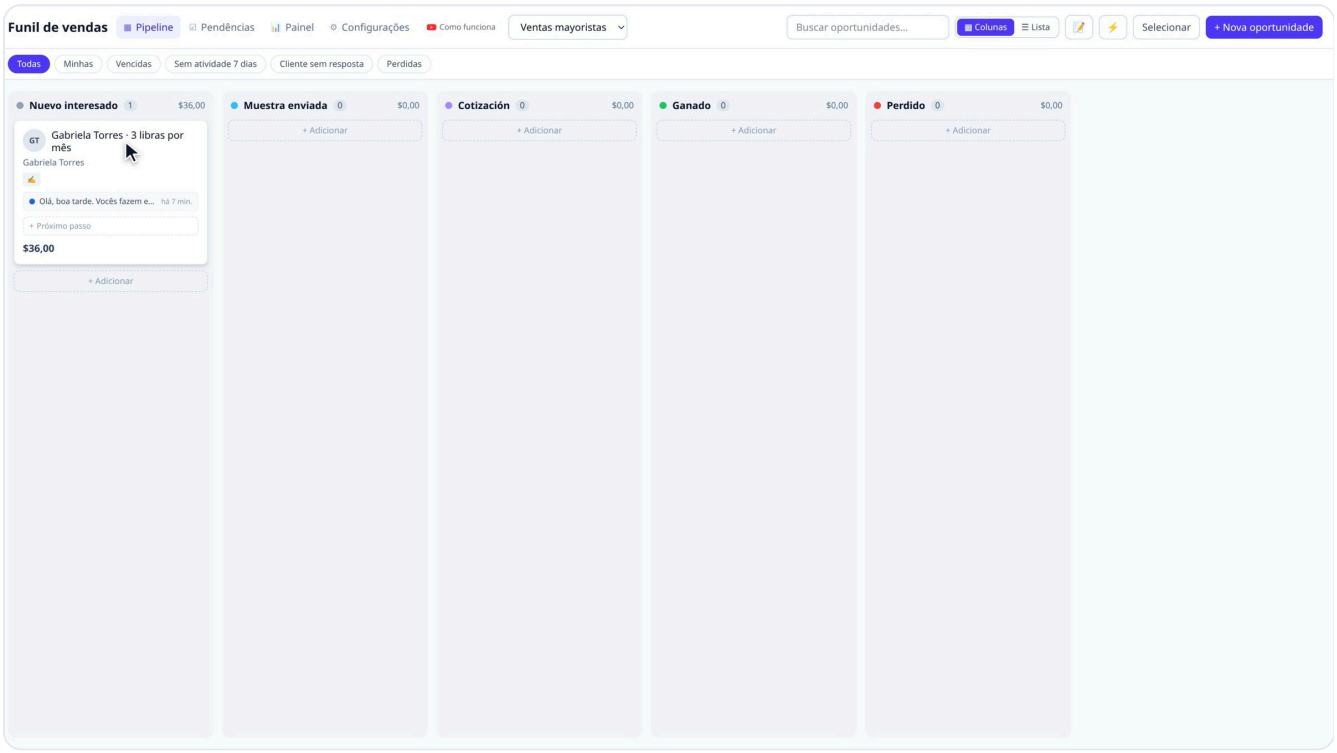
CAPÍTULO 12

CRM e calendário pela API

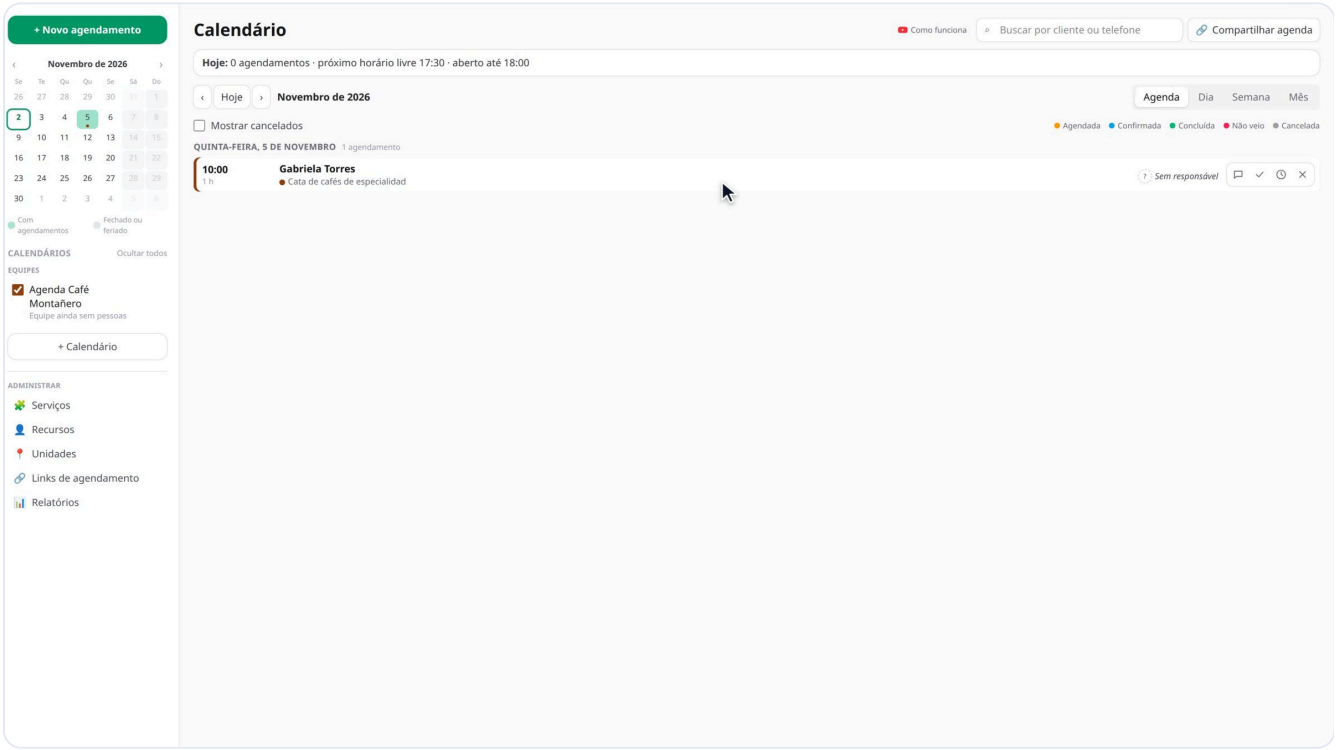
A API chega à plataforma inteira.

```
curl -s -X POST $API/crm/deals -H "Authorization: Bearer $OMNIFOX_API_KEY" -H "Content-Type: application/json" \
  -d '{"title":"Gabriela Torres · 3 libras por mês","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"}'
```



A oportunidade criada pela API, no funil do CRM.



O agendamento feito pela API, no calendário.

Os agendamentos também podem ser remarcados (`PUT /calendar/appointments/{id}`) e cancelados (`POST /calendar/appointments/{id}/cancel`). O mesmo vale para projetos, catálogo, pedidos, relatórios, equipes, etiquetas e campanhas: está tudo em /docs.

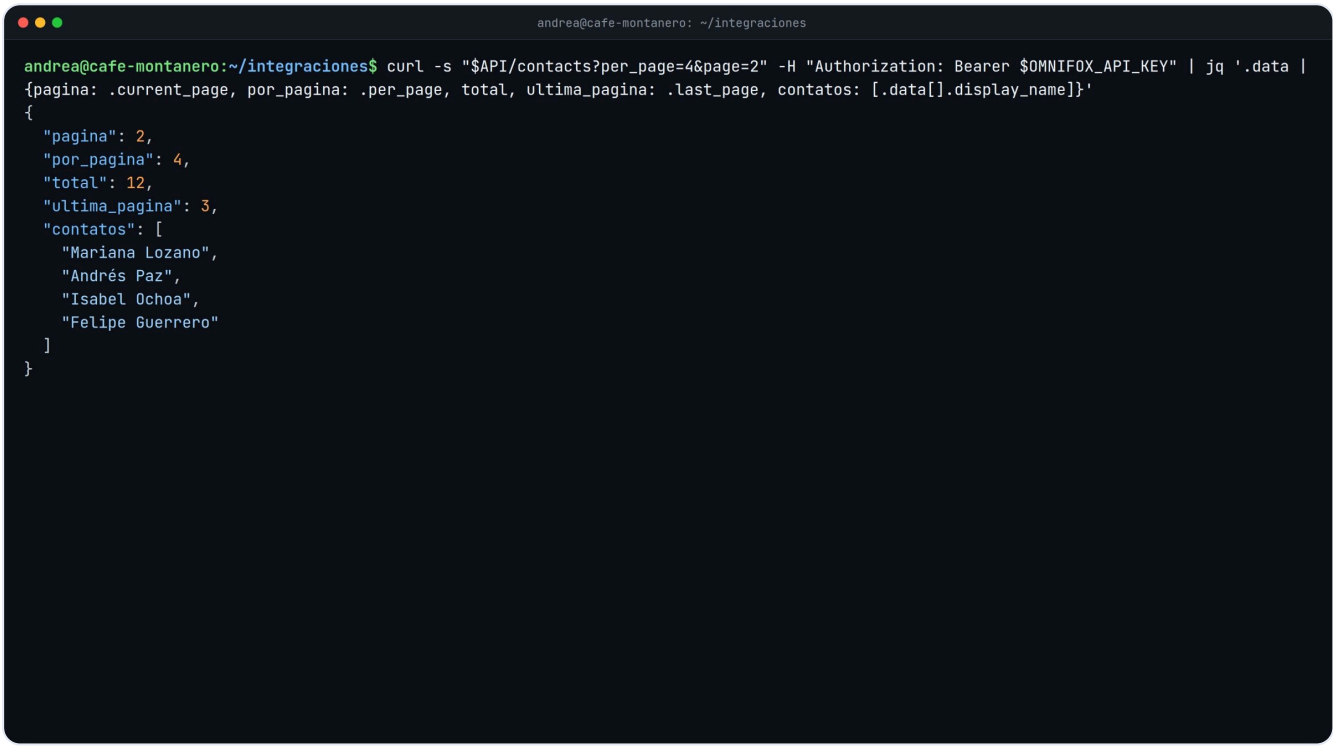
13

CAPÍTULO 13

Paginação e limites

Percorra as páginas e respeite o limite.

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



Paginação: página, por página, total e última página.

A resposta traz `current_page`, `per_page`, `total` e `last_page`. Avance até `last_page`. Se receber **429**, espere e tente de novo.

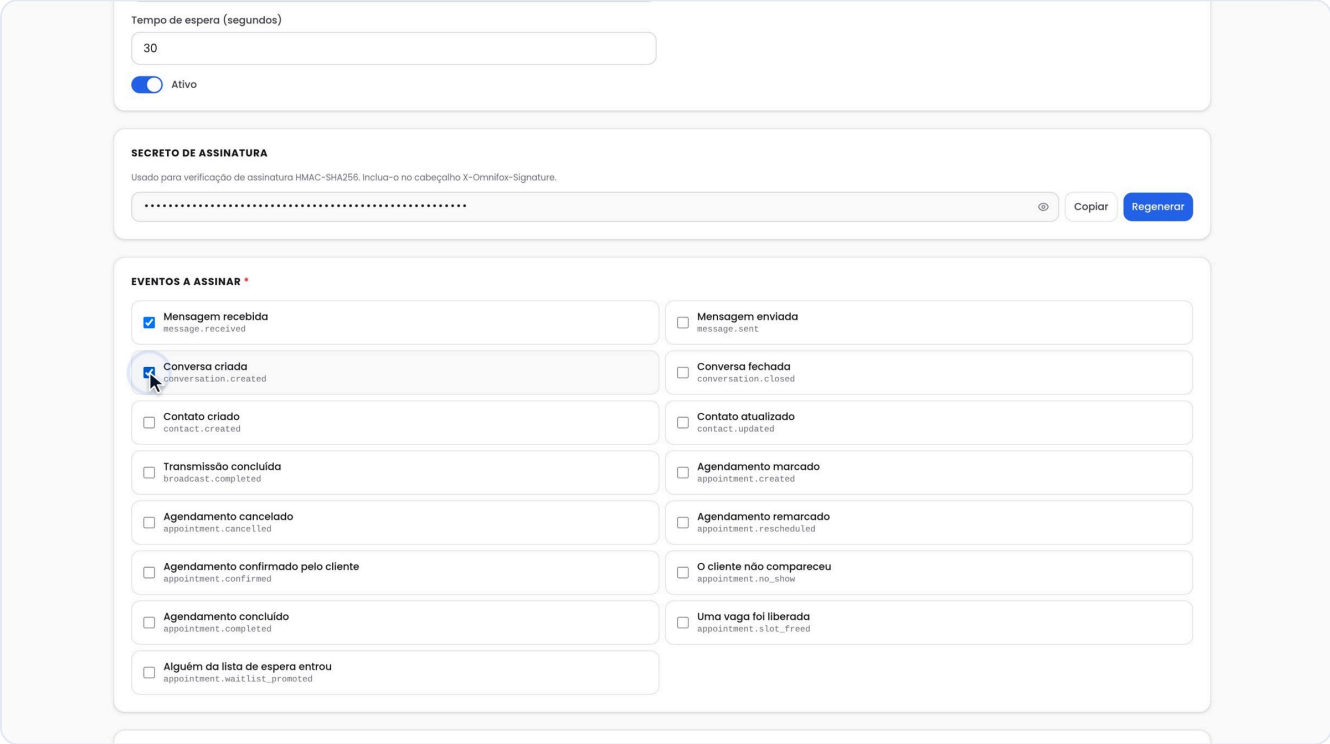
14

CAPÍTULO 14

Webhooks: criar uma assinatura

Que o Omnifox avise você na hora.

1. Vá em **Configurações** → **Webhooks** e clique em **Novo webhook**.
2. Escreva um **Nome** e a **URL de destino** do seu servidor (precisa ser **https** e pública).
3. Marque os **Eventos para assinar**.



Novo webhook: nome, URL https e eventos.

Evento	Quando chega
`message.received`	Chega uma mensagem de um cliente.
`message.sent`	A sua equipe, a IA ou um fluxo envia uma mensagem.
`conversation.created` · `conversation.closed`	Uma conversa se abre ou se fecha.
`contact.created` · `contact.updated`	Um contato é criado ou muda.
`broadcast.completed`	Uma campanha terminou.
`appointment.*`	Agendamento criado, cancelado, remarcado, confirmado pelo cliente, o cliente não veio, atendido, liberou-se uma vaga, entrou alguém da lista de espera.

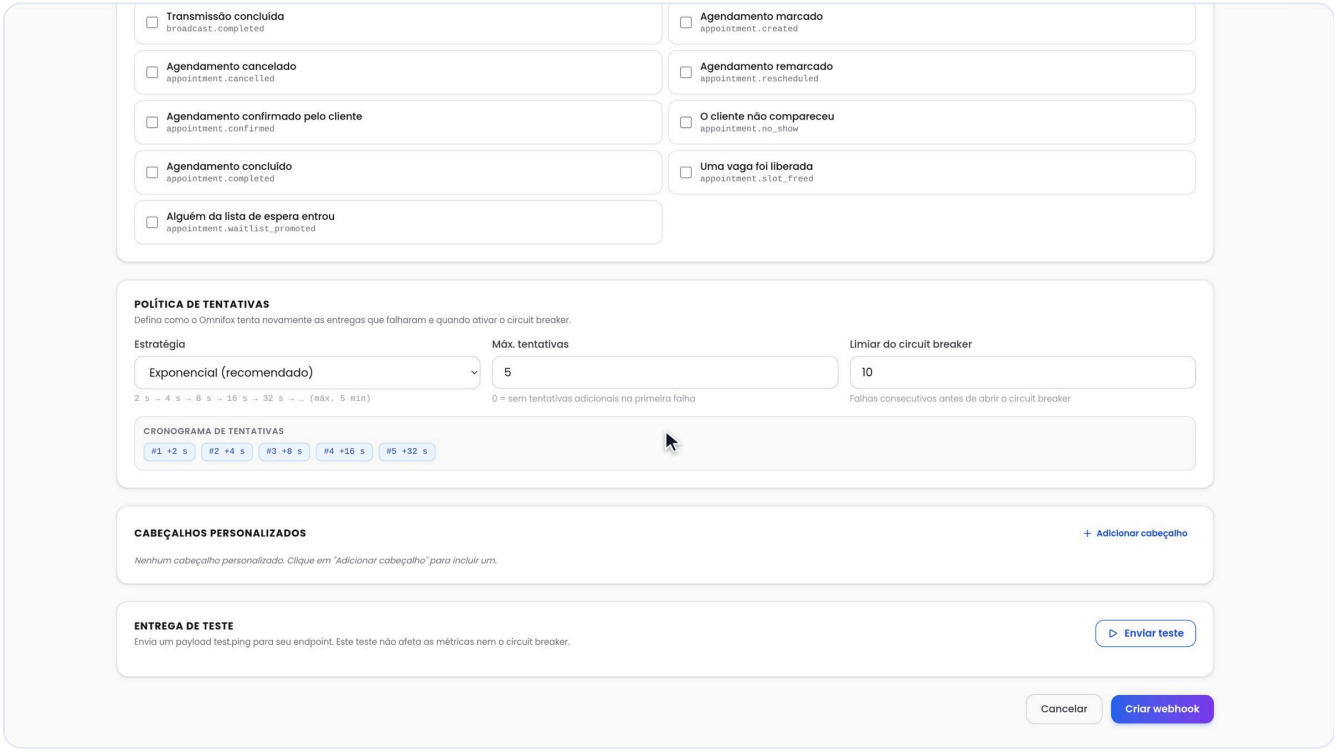
15

CAPÍTULO 15

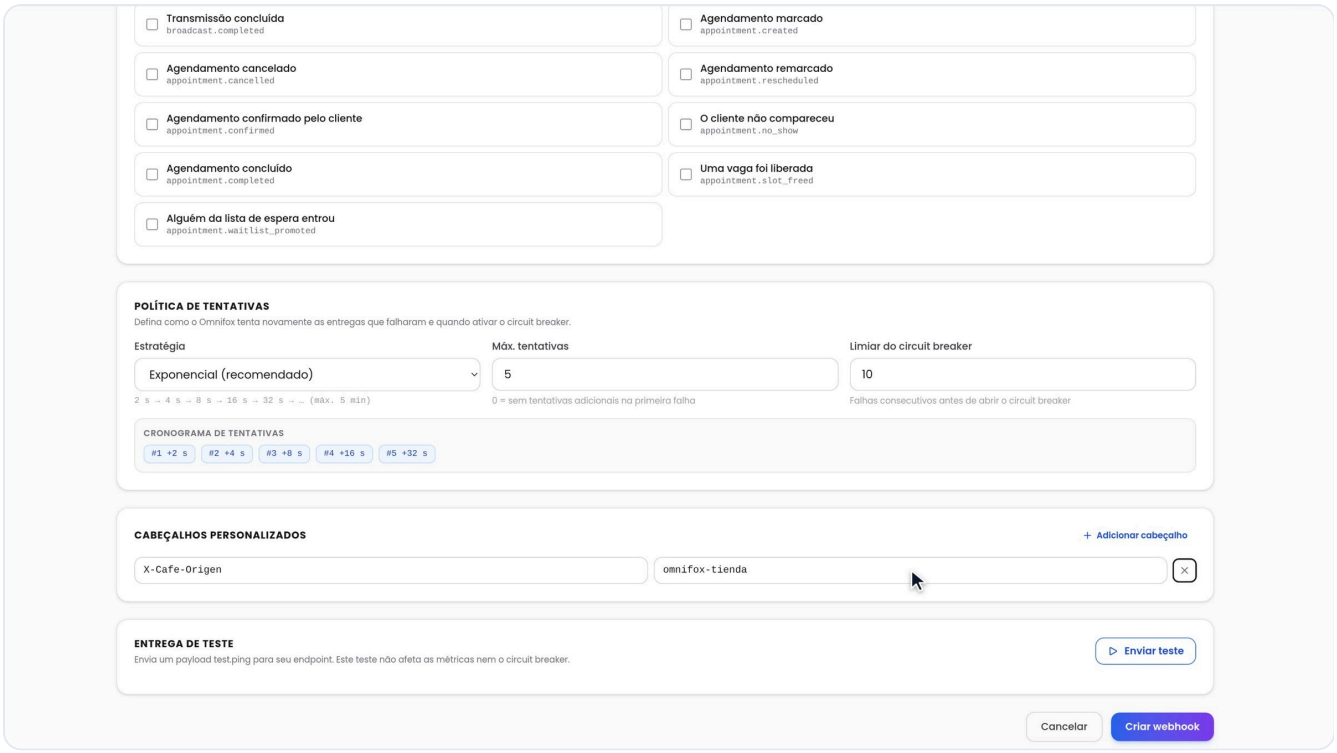
Segredo de assinatura, novas tentativas e cabeçalhos

Assinado, com novas tentativas e protegido.

Ajuste	Para quê
Segredo de assinatura	O Omnifox assina cada envio com este segredo. Copie para o seu servidor para verificar (capítulo 18). **Regenerar** cria um novo.
Estratégia	**Exponencial** (recomendada: 2 s, 4 s, 8 s, 16 s... até 5 min), **Linear** (a cada 60 s) ou **Sem novas tentativas**.
Máx. de tentativas	Quantas vezes tentar de novo uma entrega com falha.
Limite do circuit breaker	Depois de tantas falhas seguidas, o Omnifox para de enviar por um tempo e depois tenta de novo.
Tempo limite	Segundos de espera pela resposta do seu servidor.
Cabeçalhos personalizados	Cabeçalhos próprios em cada envio (por exemplo, uma chave do seu lado).



Política de novas tentativas e calendário de tentativas.



Cabeçalhos personalizados.

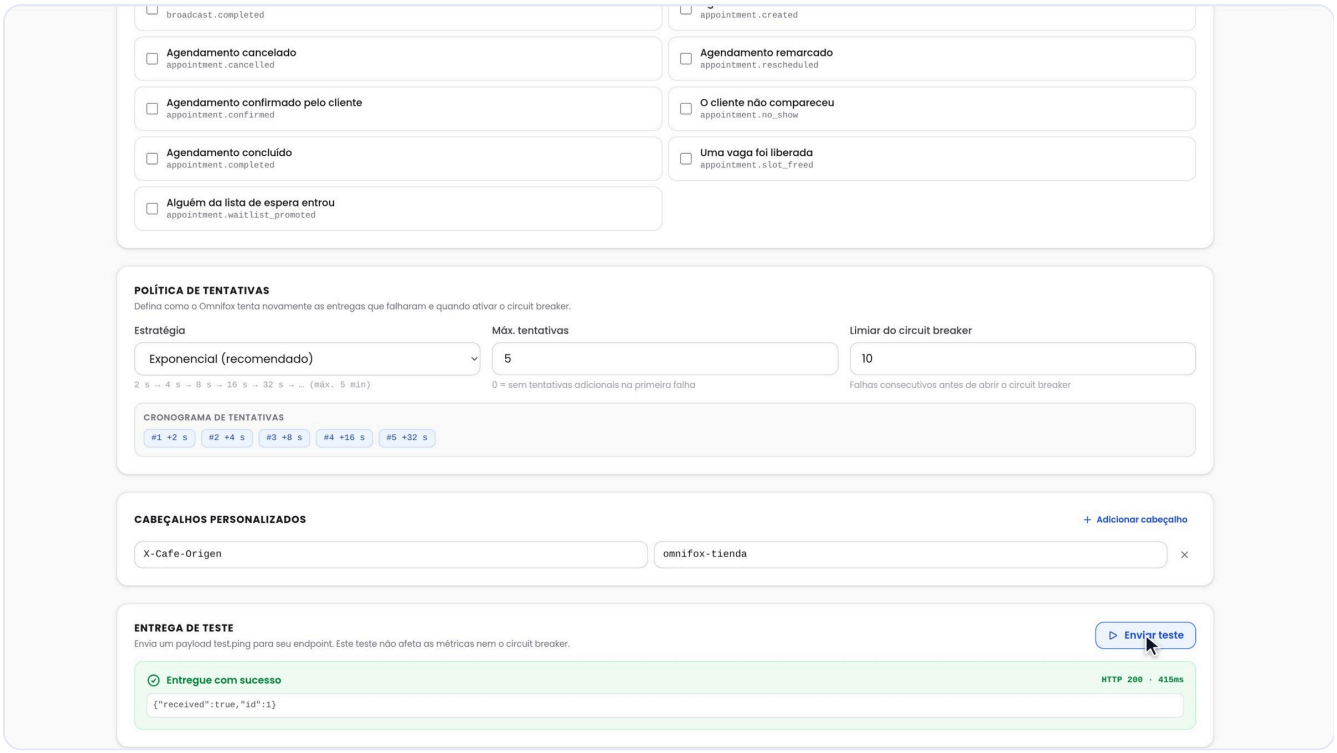
16

CAPÍTULO 16

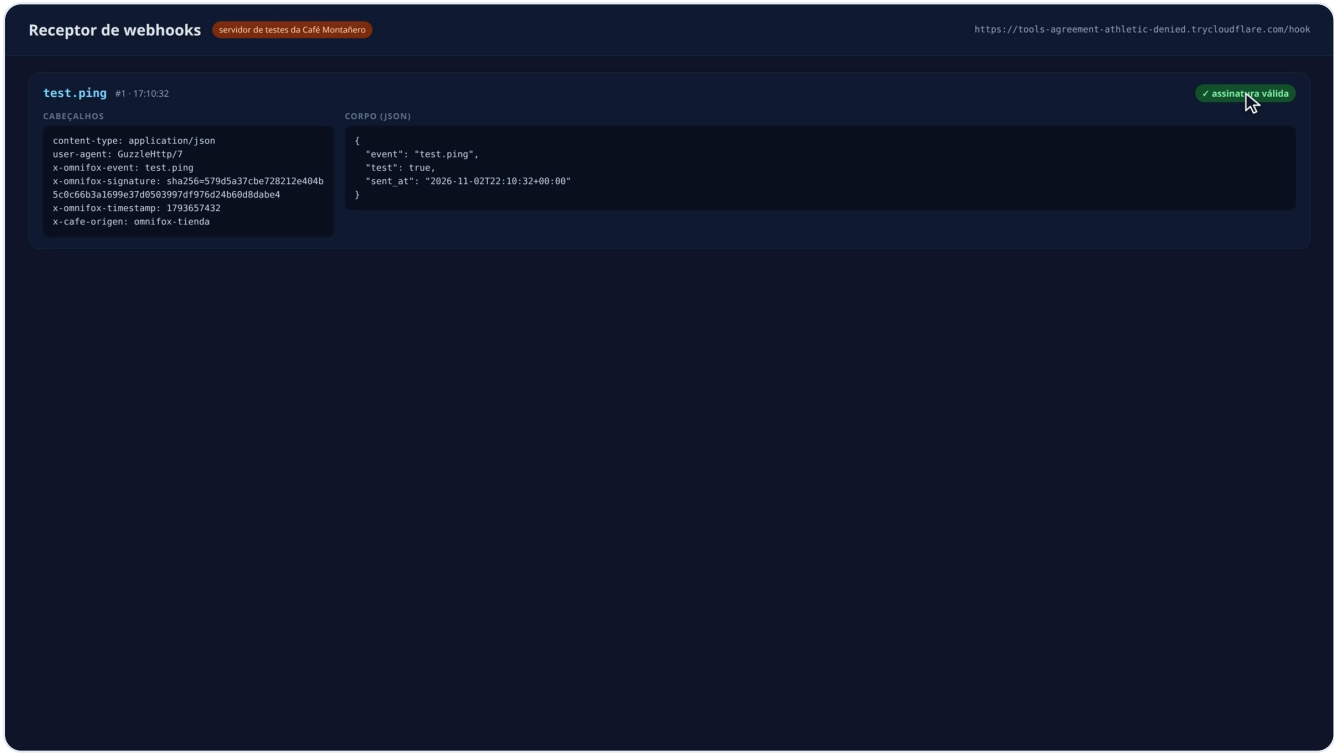
Entrega de teste

Teste antes de salvar.

Enviar teste manda um evento `test.ping` para a sua URL e mostra o código HTTP, o tempo e a resposta. Não conta para as métricas nem para o circuit breaker.



Resultado da entrega de teste.



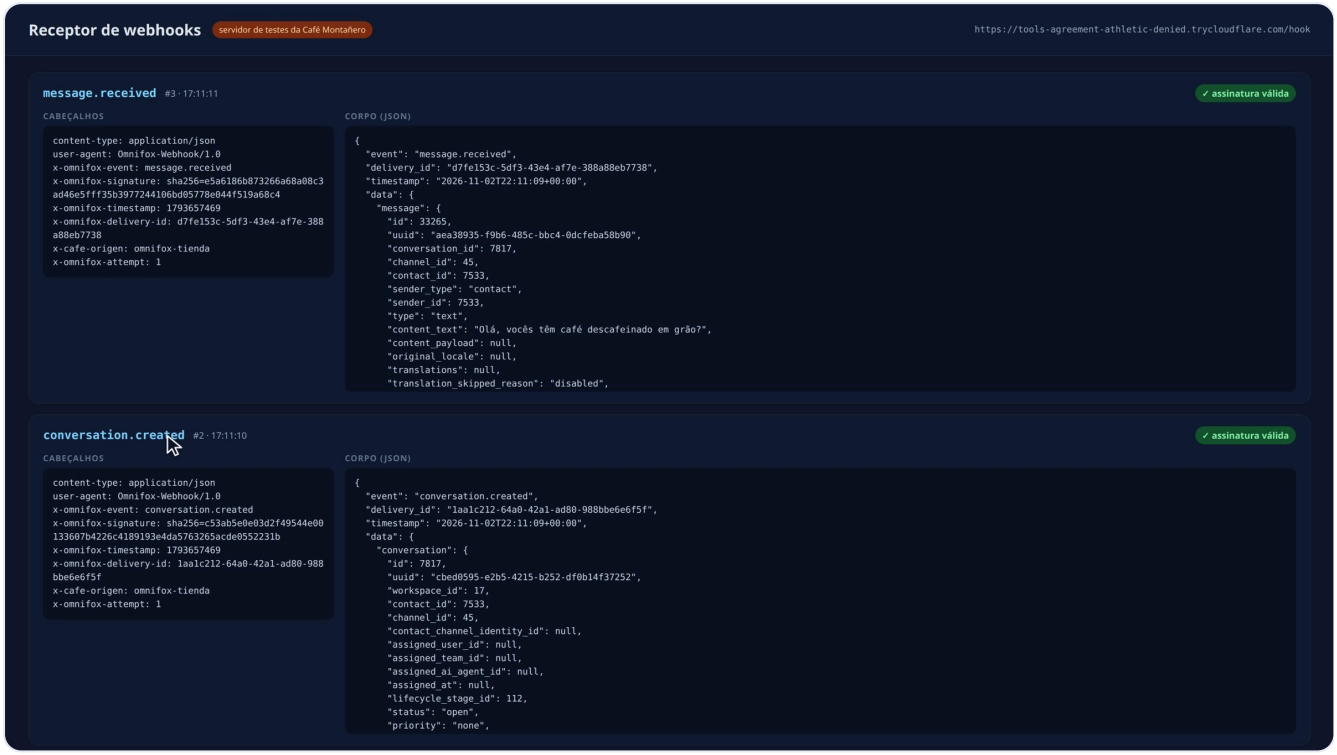
O ping no receptor, com a assinatura verificada.

17

CAPÍTULO 17

Um evento real

Um cliente escreve e o seu servidor fica sabendo.



Os eventos message.received e conversation.created no receptor.

Cabeçalho	Conteúdo
`X-Omnifox-Event`	O evento (`message.received` ...).
`X-Omnifox-Delivery-Id`	Identificador único da entrega: guarde-o para não processar duas vezes o mesmo evento.
`X-Omnifox-Attempt`	Número da tentativa (1 na primeira vez).
`X-Omnifox-Timestamp`	Carimbo de tempo Unix da assinatura.
`X-Omnifox-Signature`	`sha256=` + HMAC-SHA256 de «timestamp.corpo».

O corpo traz `event` , `delivery_id` , `timestamp` e `data` (em `message.received` : a mensagem com `content_text` , o contato, a conversa e o canal).

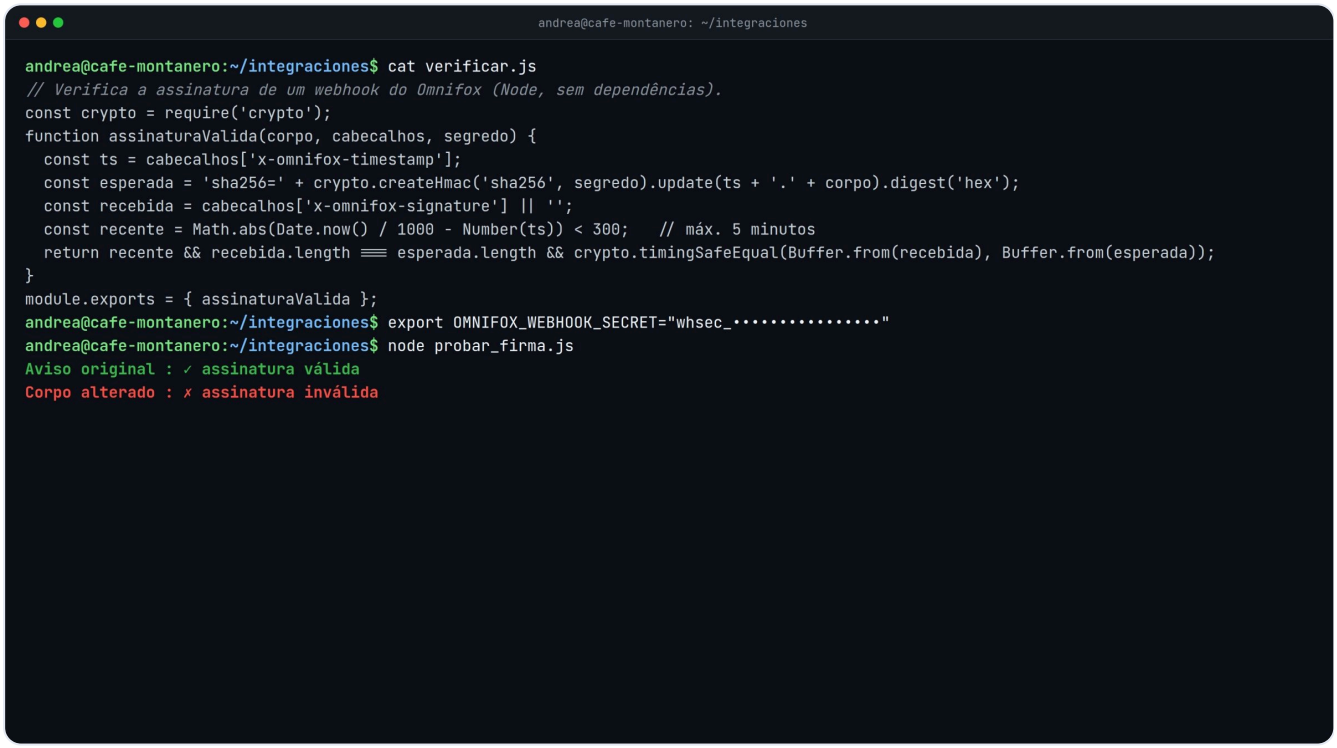
18

CAPÍTULO 18

Verificar a assinatura no seu servidor

Dez linhas para saber que o aviso é autêntico.

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



Verificação da assinatura: o aviso original passa, o alterado não.

ATENÇÃO

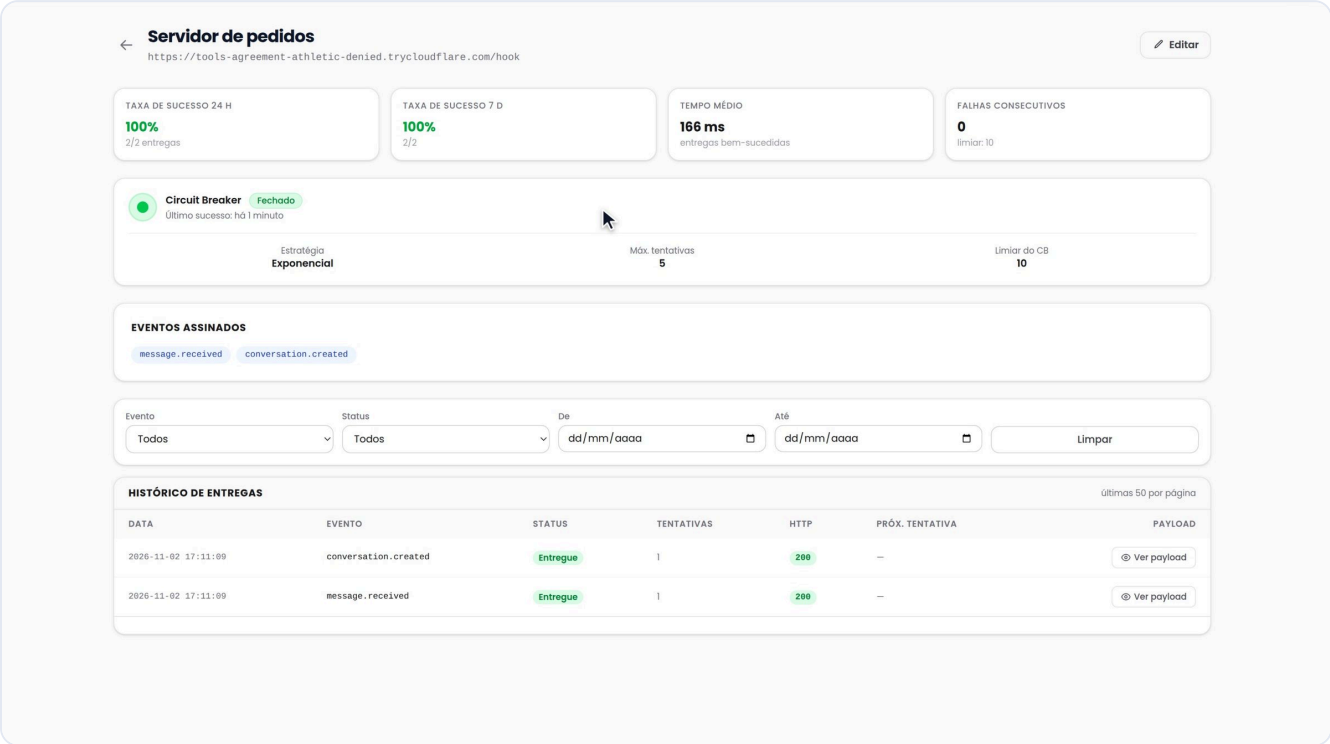
Use o corpo EXATO que você recebeu (sem re-serializar o JSON) e uma comparação de tempo constante.

19

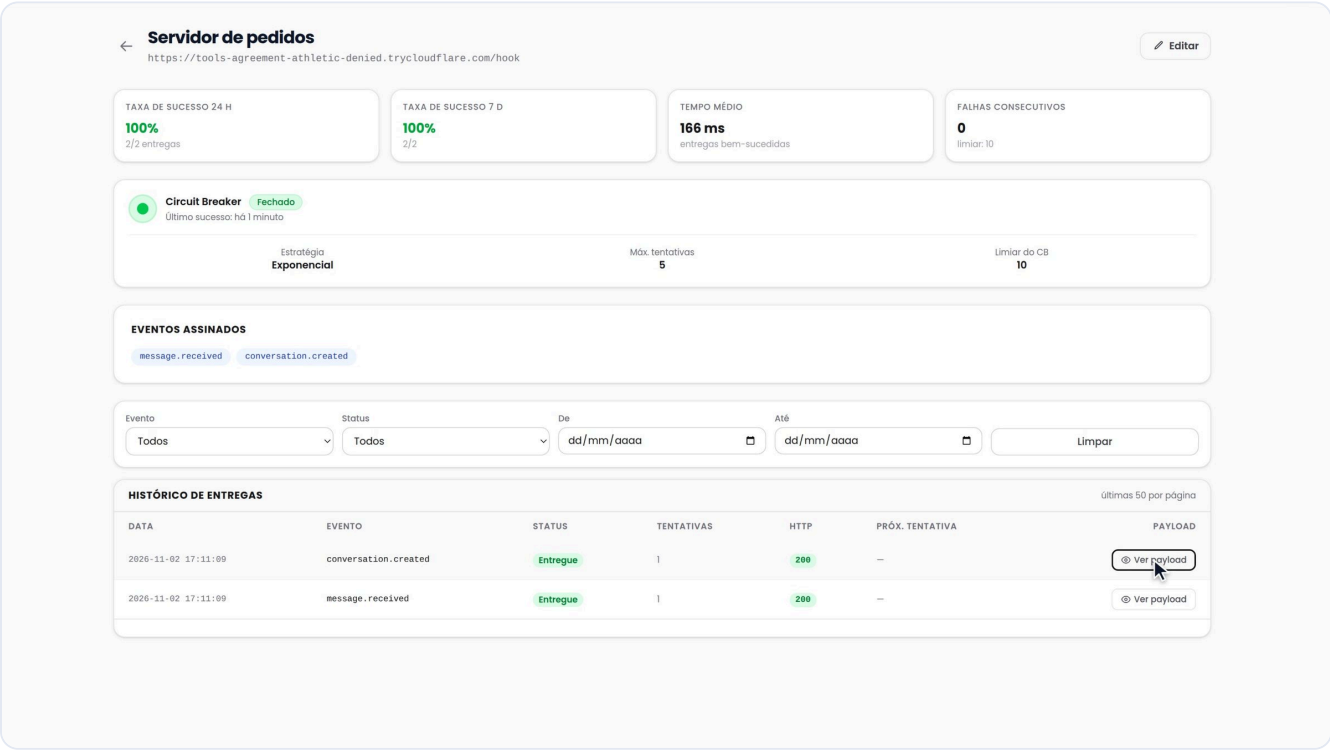
CAPÍTULO 19

Histórico de entregas

O que foi enviado, o que o seu servidor respondeu e quando.



Detalhe do webhook: taxas de sucesso, tempo médio e falhas seguidas.



O histórico de entregas e o payload de uma entrega.

- Taxa de sucesso de 24 horas e 7 dias, tempo médio e falhas seguidas.

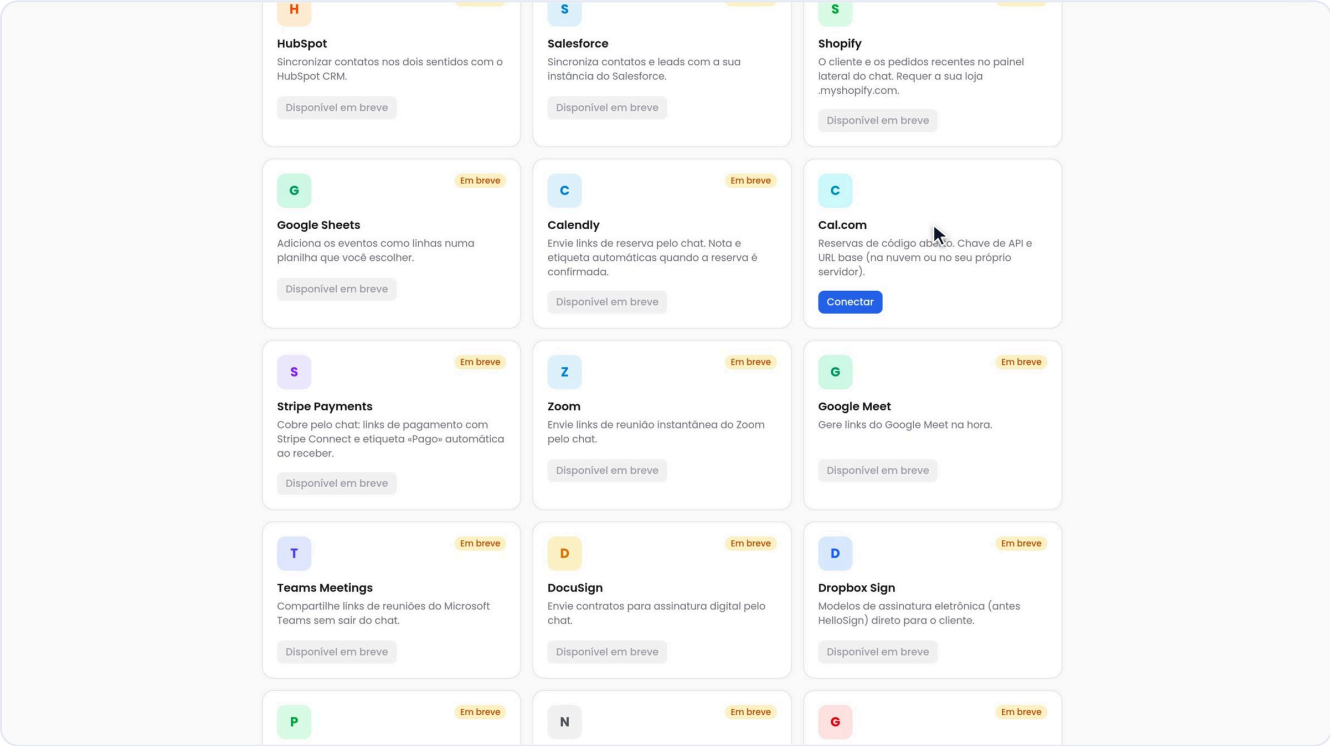
- Status do **circuit breaker**; se ficou aberto, force o fechamento quando o seu servidor voltar.
- **Histórico de entregas** com filtros por evento, status e data; **Ver payload** mostra o que foi enviado e a resposta.
- Pela lista: pausar, testar, regenerar o segredo, editar ou excluir.

20

CAPÍTULO 20

A página de integrações

Conectores prontos.



A página de integrações.

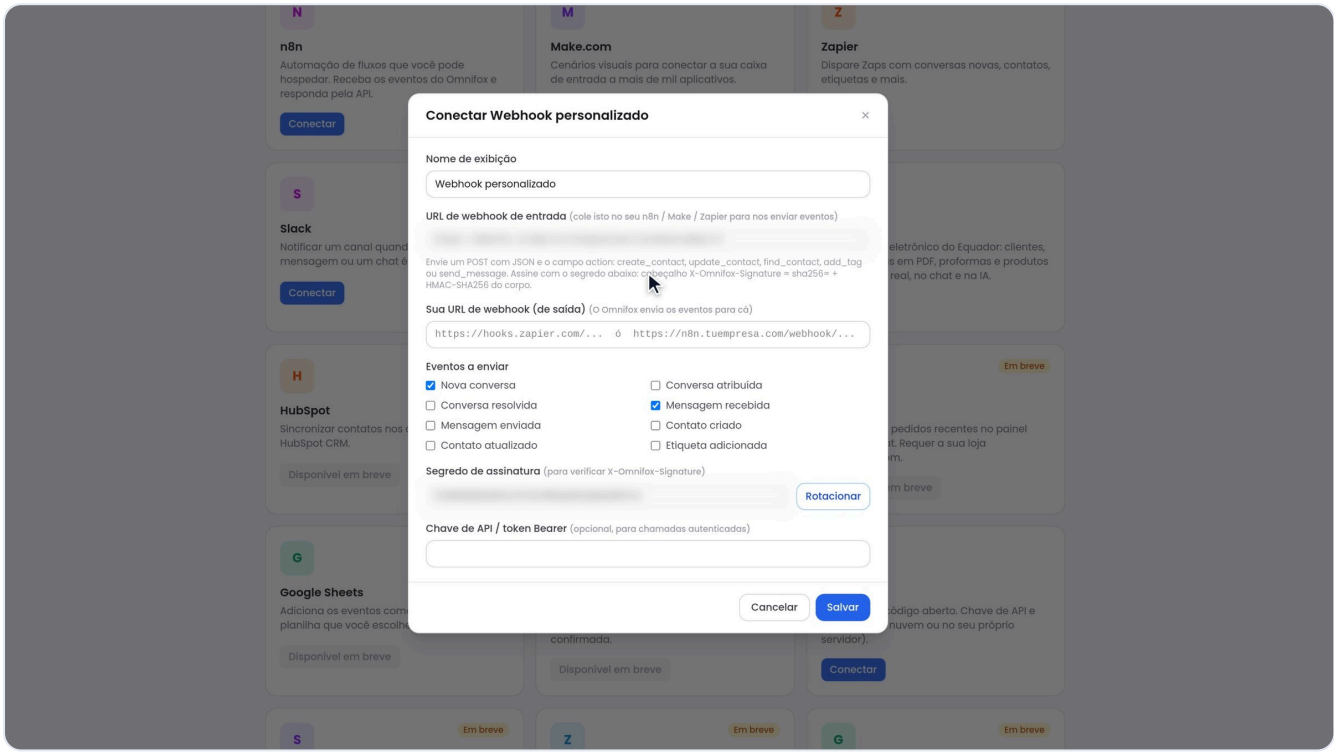
Integração	Como funciona
n8n, Make, Zapier, Webhook personalizado	Webhooks nos dois sentidos: uma URL de entrada com ações e eventos de saída assinados (capítulo 21).
Azur	A nota fiscal eletrônica do Equador: clientes, saldos, notas, orçamentos e produtos, pelo chat e pela IA.
Cal.com	As suas reservas do Cal.com no calendário do Omnifox.
Em breve	HubSpot, Salesforce, Shopify, Google Sheets, Calendly, Stripe Payments, Zoom, Google Meet, Teams, DocuSign, Dropbox Sign, PandaDoc, Notion, Gmail e Outlook. Enquanto isso: API + webhooks + n8n.

21

CAPÍTULO 21

Webhook personalizado: entrada e saída

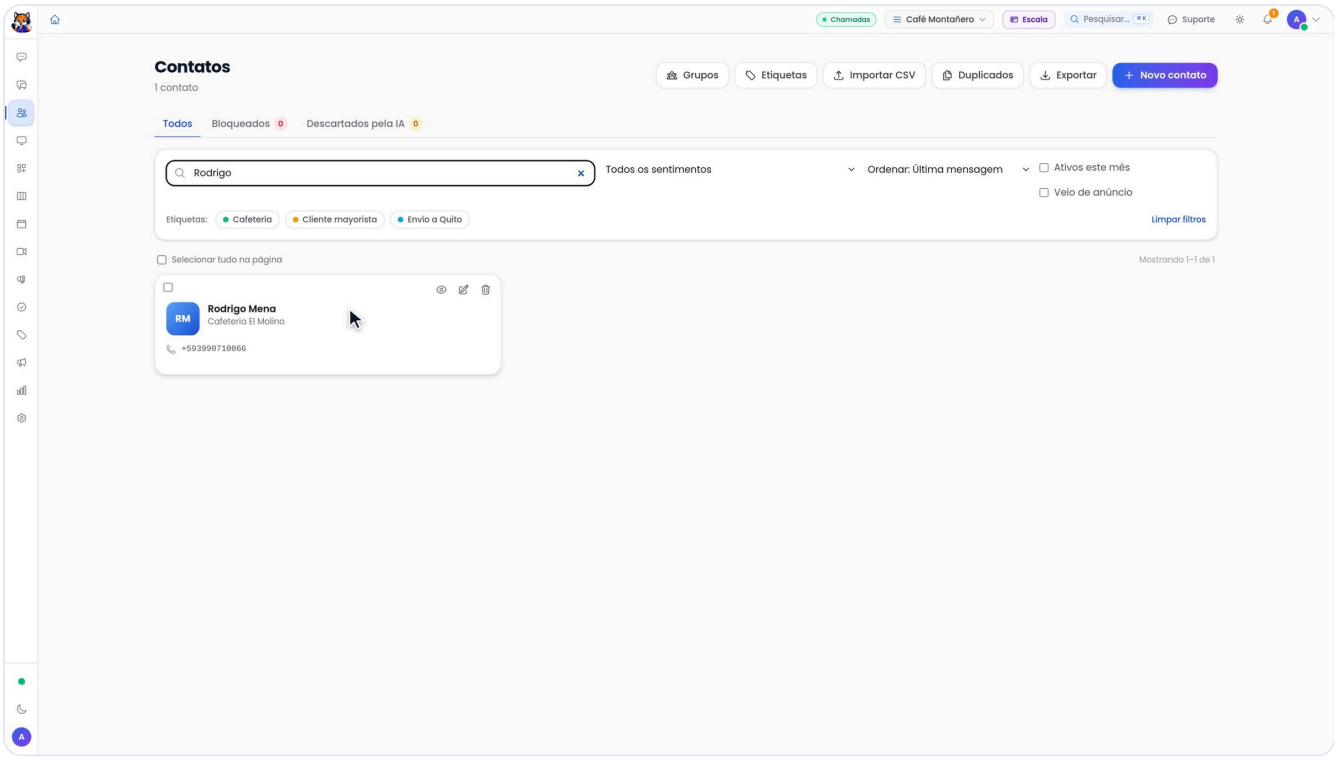
Uma URL para receber ações e outra para enviar eventos.



Webhook personalizado: URL de entrada, URL de saída, eventos e segredo de assinatura.

Ação de entrada (`action`)	Campos
`create_contact`	`display_name` , `email` , `phone` , `company` , `notes` , `external_id`
`update_contact`	O contato e os campos a mudar
`find_contact`	Telefone ou e-mail
`add_tag`	Contato e etiqueta
`send_message`	Contato ou conversa e o texto

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



Uma ação de entrada assinada (create_contact) pelo terminal.

A **URL de saída** e os **Eventos a enviar** criam uma assinatura de webhook real, assinada com o mesmo **Segredo de assinatura**. **Pausar** a interrompe e **Desconectar** a apaga.

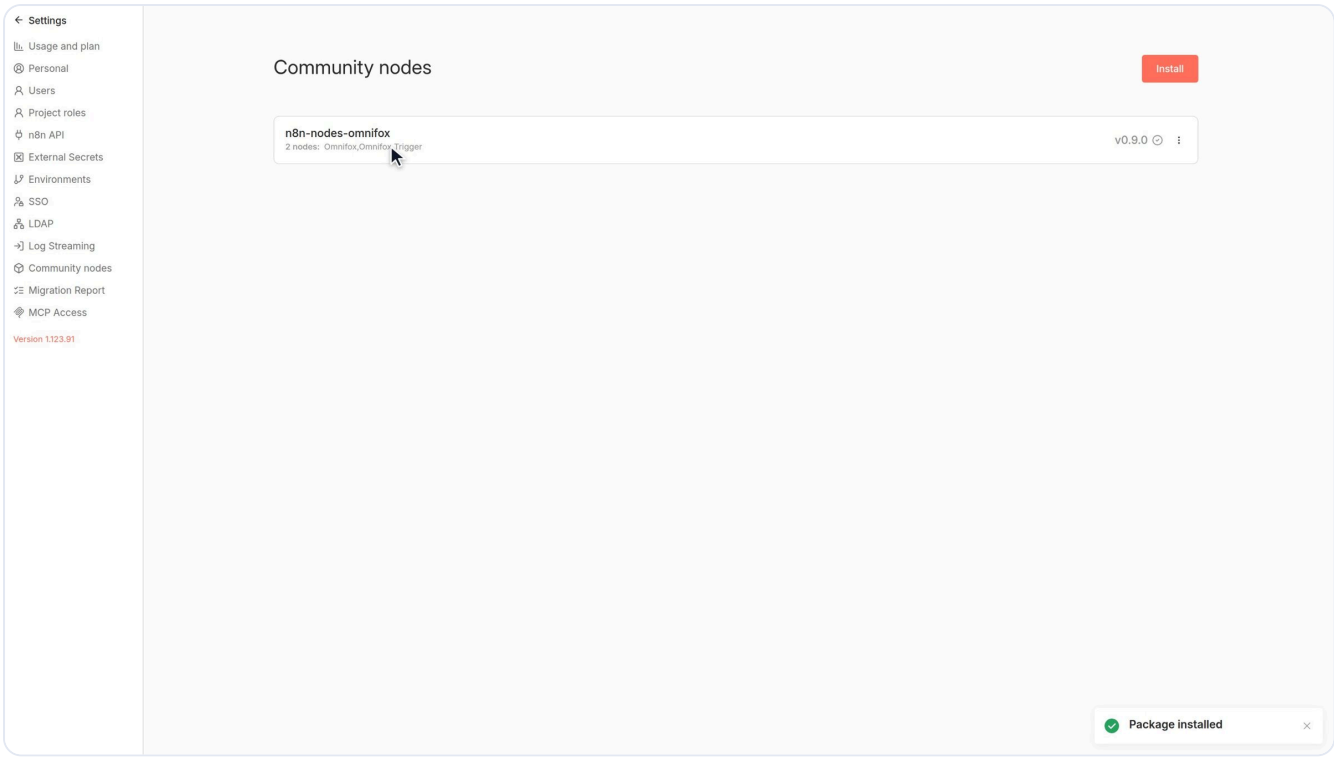
22

CAPÍTULO 22

n8n: instalar o nó do Omnifox

Um pacote da comunidade, oficial do Omnifox.

1. No n8n vá em **Settings** → **Community nodes** e clique em **Install**.
2. Digite `n8n-nodes-omnifox`, aceite o aviso de riscos e clique em **Install**.
3. Ficam dois nós: **Omnifox** (ações) e **Omnifox Trigger** (eventos).



n8n → Settings → Community nodes com o n8n-nodes-omnifox instalado.

NOTA

Funciona no n8n Cloud e numa instalação própria. Para o gatilho, o seu n8n precisa ser acessível por https pela internet.

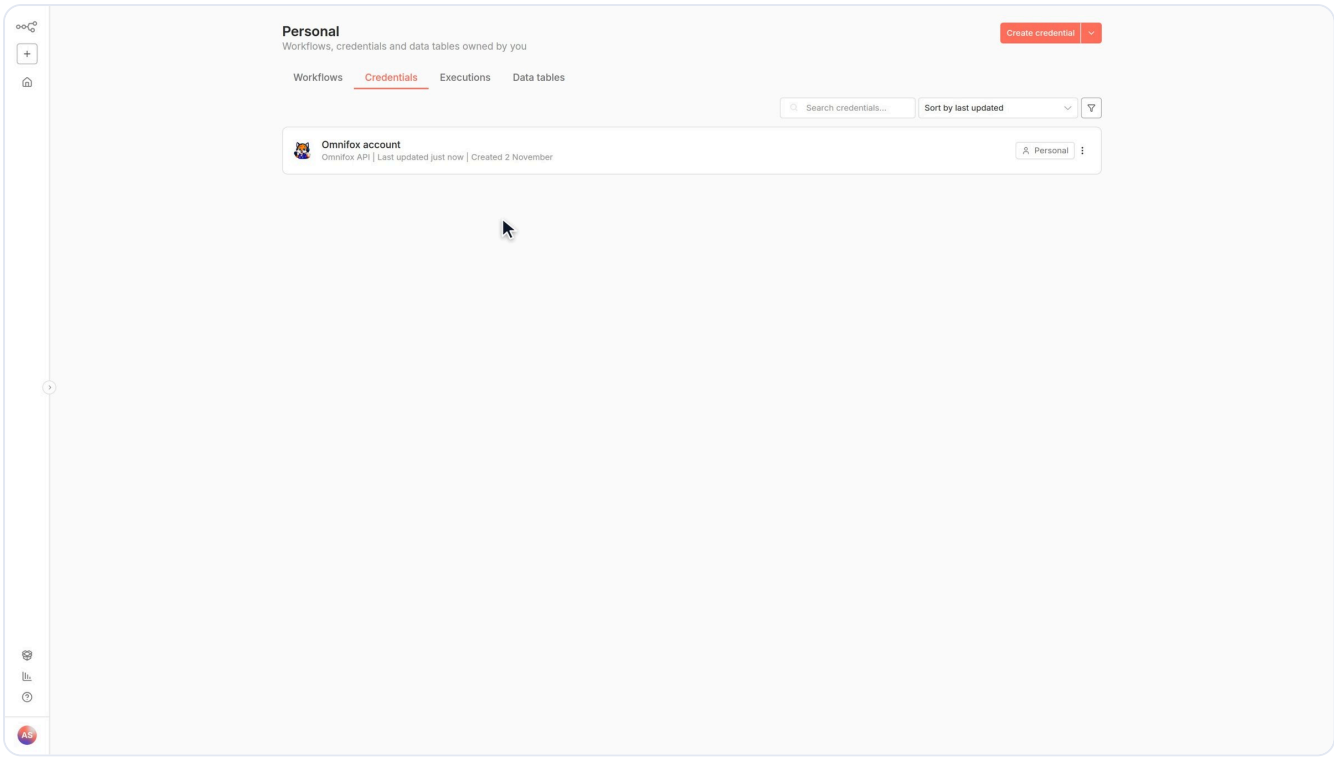
23

CAPÍTULO 23

n8n: a credencial do Omnifox

Uma credencial, todas as operações.

Campo	Valor
API Token	A sua chave de API (melhor do tipo bot e só para o n8n).
Workspace ID	0: vem da chave.
Base URL	`https://app.omnifox.io` (sem /api: o nó adiciona).



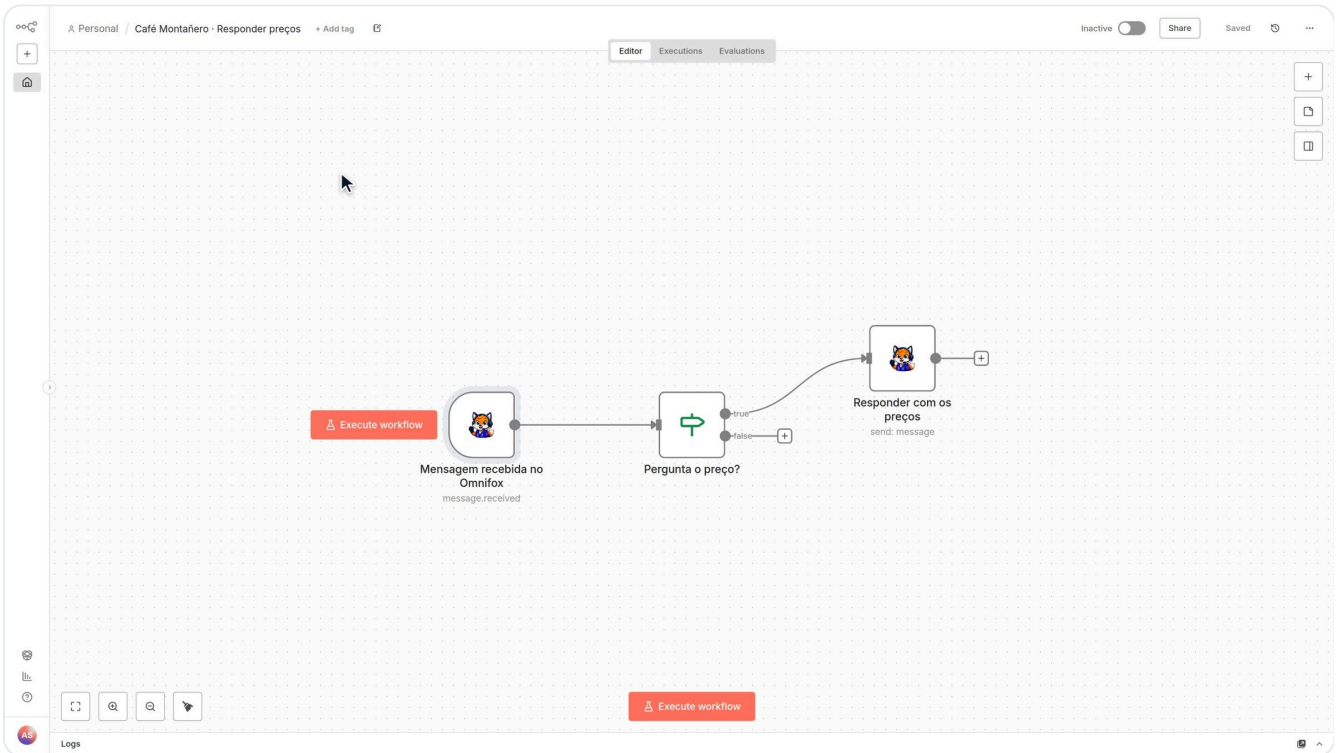
A credencial «Omnifox API» no n8n, com a conexão testada.

24

CAPÍTULO 24

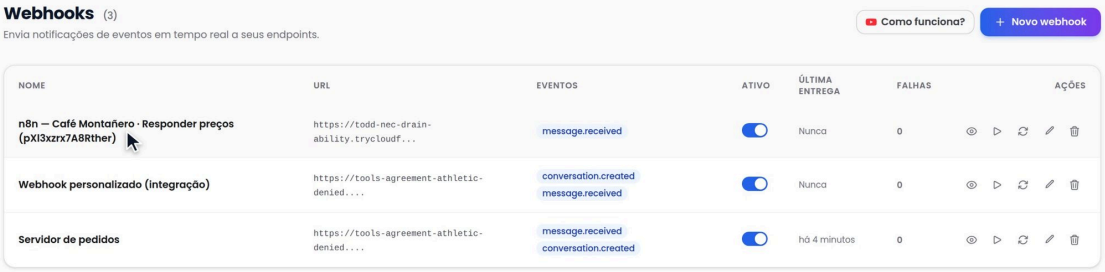
n8n: responder só quando perguntam o preço

Um fluxo de três nós, testado de verdade.

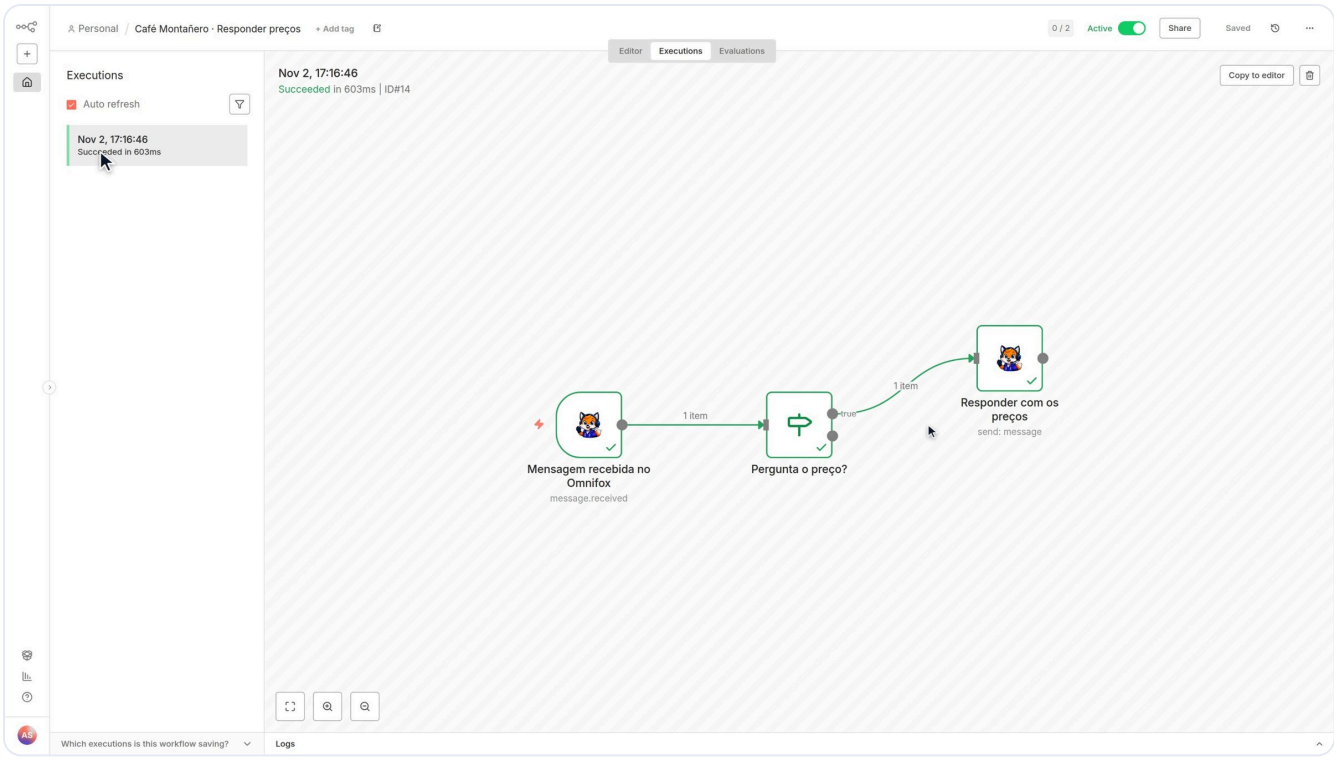


O fluxo do n8n: gatilho do Omnifox, condição e resposta.

Nó	Configuração
Omnifox Trigger	Evento `message.received`. Ao ativar, cria a assinatura no Omnifox e verifica a assinatura de cada aviso.
If	Se `{{ $json.data.message.content_text }}` contém «preço» (sem diferenciar maiúsculas).
Omnifox	Recurso **Message**, operação **Send**, conversa `{{ ...data.conversation.id }}`, texto com a lista de preços.



A assinatura que o n8n criou, em Webhooks do Omnifox.



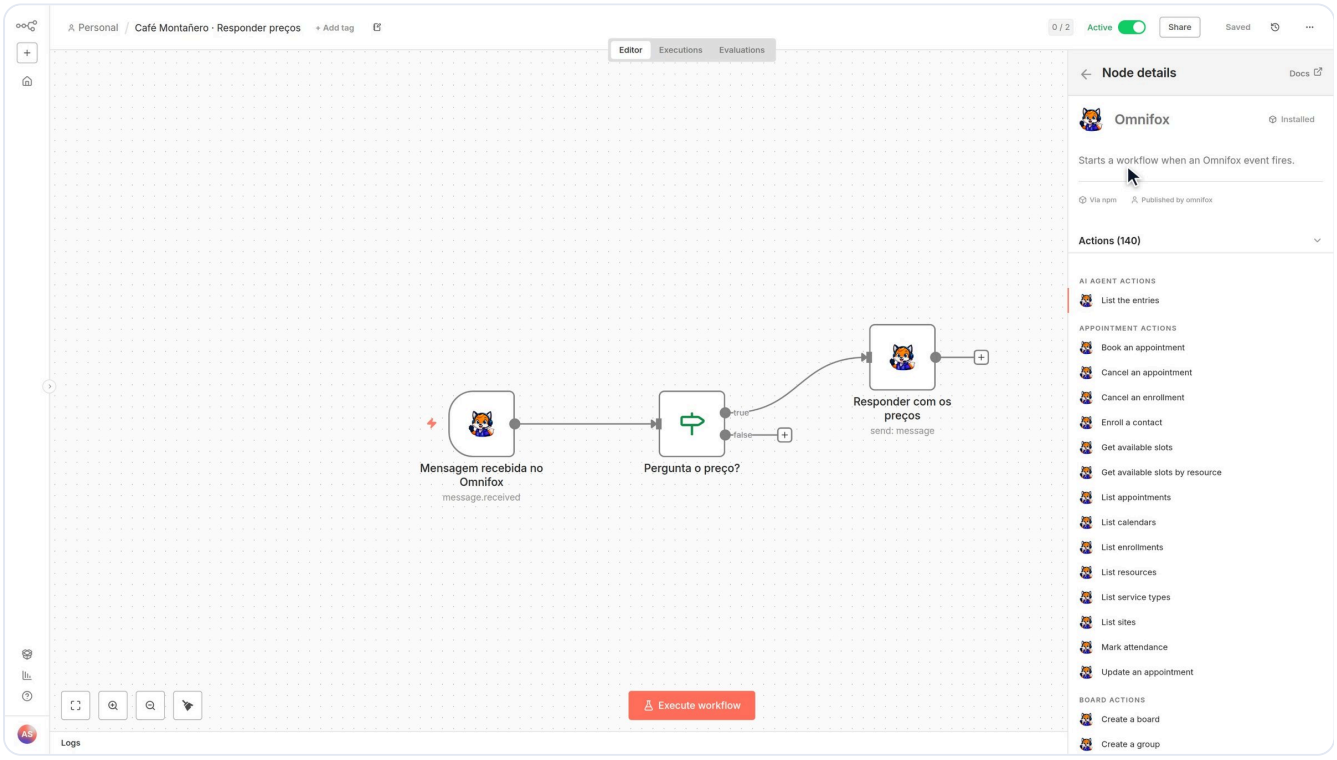
A execução do fluxo no n8n.

25

CAPÍTULO 25

n8n: todas as operações

140 operações num único nó.



As ações do nó Omnifox no n8n.

Conversas, contatos, mensagens, etiquetas, campanhas, canais, equipes, agentes de IA, CRM, quadros, calendário, catálogo e pedidos. As listas (canais, etiquetas, equipes, modelos) carregam da sua conta.

Erro típico	Solução
Dados vazios ou 403	A chave é de outro espaço de trabalho ou falta um escopo.
404 em todas as chamadas	Base URL com `/api` no final: remova.

26

CAPÍTULO 26

MCP: que uma IA opere o Omnifox

72 ferramentas para o seu assistente de IA.

O servidor MCP do Omnifox (`@omnifox/mcp-server`) expõe 72 ferramentas: contatos, conversas, mensagens, etiquetas, campos, calendário e agendamentos, cursos, catálogo, pedidos, fluxos, campanhas, webhooks e relatórios. Roda com `npx` e precisa de duas variáveis:

```
OMNIFOX_API_KEY=sua-chave
OMNIFOX_BASE_URL=https://app.omnifox.io/api/v1
```

ATENÇÃO

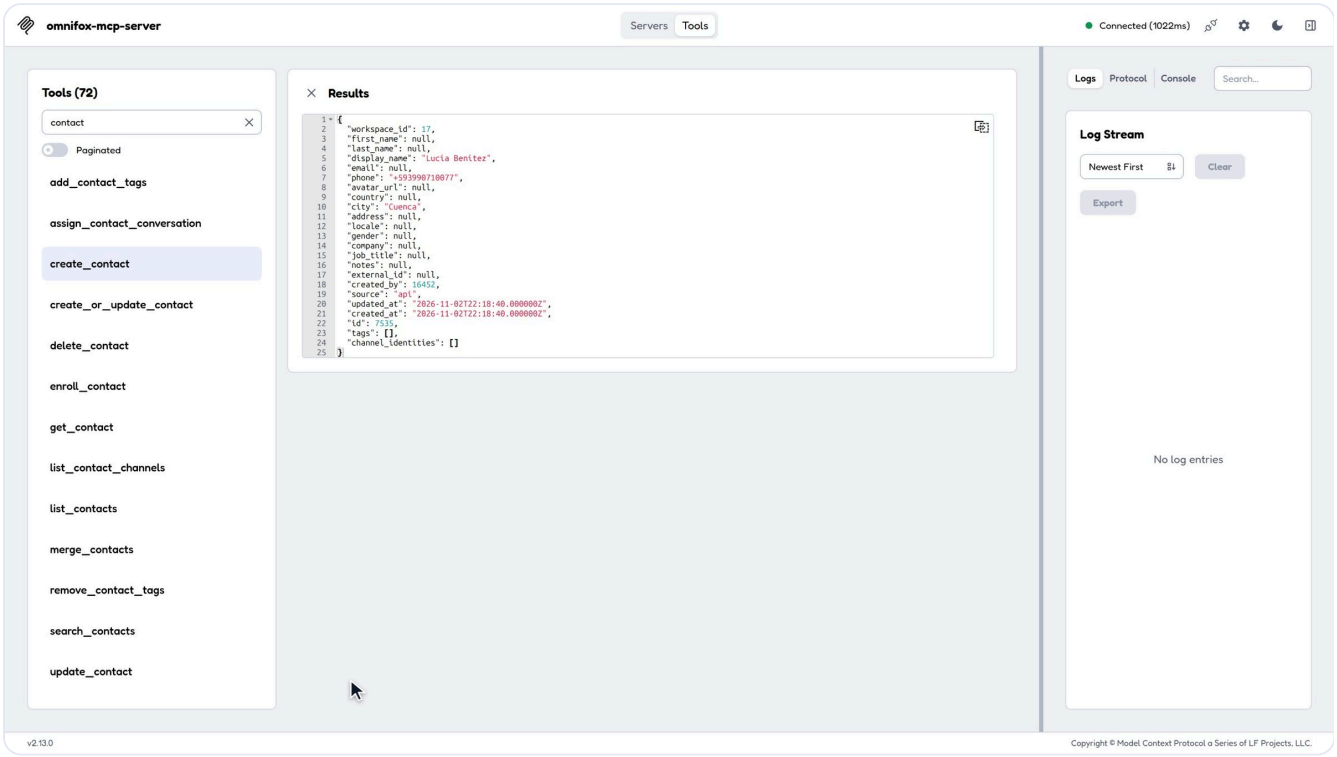
A IA só pode fazer o que os escopos da chave permitirem. Para um assistente, comece com escopos de leitura.

27

CAPÍTULO 27

Testar o servidor MCP com o Inspector

Teste as ferramentas sem conectar um assistente.



MCP Inspector: as ferramentas do servidor do Omnifox e o resultado de uma chamada.

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

Esse bloco vai na configuração MCP do Claude Desktop, do Cursor ou de outro cliente. Para servidores remotos, o MCP também funciona por HTTP.

28

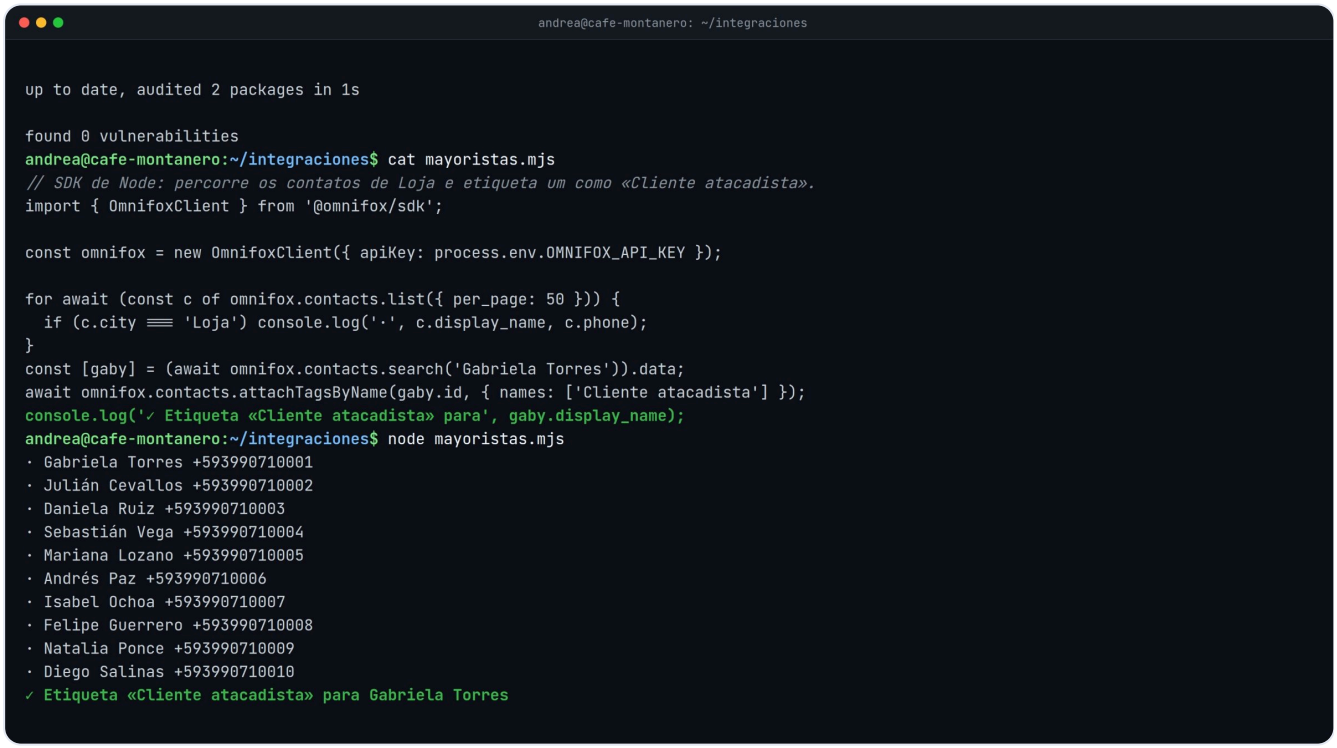
CAPÍTULO 28

SDKs oficiais: Node, Python e PHP

Autenticação, paginação, novas tentativas e erros, resolvidos.

Linguagem	Instalar	Cliente
Node	`npm install @omnifox/sdk`	`new OmnifoxClient({ apiKey })`
Python	`pip install omnifox`	`Omnifox(api_key=...)` e `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 atacadista'] });
```



O SDK de Node em ação.

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

from zoneinfo import ZoneInfo
from omnifox import Omnifox

omnifox = Omnifox(api_key=os.environ["OMNIFOX_API_KEY"])
for ag in omnifox.appointments.list(status="scheduled"):
    hora = datetime.fromisoformat(ag.starts_at).astimezone(ZoneInfo("America/Guayaquil"))
    print("·", hora.strftime("%d/%m %H:%M"), ag.title, "·", ag.status)
andrea@cafe-montanero:~/integraciones$ python3 citas.py
· 05/11 10:00 Cata de cafés de especialidad · scheduled
andrea@cafe-montanero:~/integraciones$ composer require omnifox/sdk
./composer.json has been updated
Running composer update omnifox/sdk
Loading composer repositories with package information
Updating dependencies
Nothing to modify in lock file
Writing lock file
Installing dependencies from lock file (including require-dev)
Nothing to install, update or remove
Generating autoload files
5 packages you are using are looking for funding.
Use the `composer fund` command to find out more!
No security vulnerability advisories found.
Using version ^0.4.1 for omnifox/sdk
andrea@cafe-montanero:~/integraciones$ php etiquetas.php
· Cliente atacadista
· Envío a Quito
· Cafetería
· Cliente mayorista
```

Os SDKs de Python e PHP.

CAPÍTULO 29

29 Boas práticas de segurança

Menor privilégio, segredos fora do código e verificação sempre.

- Uma chave por integração, com o mínimo de escopos e do tipo bot.
- Guarde-a em variáveis de ambiente ou num gerenciador de segredos; nunca no código, no repositório nem no navegador.
- Coloque vencimento se for temporária e regenere se alguém que a conhecia sair.
- Webhooks: verifique a assinatura, responda 200 rápido e processe numa fila; use `X-Omnifox-Delivery-Id` para não duplicar.

30

CAPÍTULO 30

Solução de problemas

O que costuma falhar e como resolver.

Sintoma	Causa provável	Solução
401 Unauthenticated	Chave copiada errada, vencida ou regenerada	Copie de novo ou crie uma chave
403 com `required_scope`	Falta o escopo	Edite a chave e adicione
403 de plano	O seu plano não inclui a API ou o módulo	Crece ou superior; CRM e Projetos conforme o plano
Listas vazias	Chave de outro espaço de trabalho	Crie a chave no espaço certo
O webhook não chega	URL não pública/https, servidor fora do ar ou circuito aberto	Confira o histórico de entregas e o código HTTP; feche o circuito
Assinatura inválida	Corpo re-serializado ou segredo antigo	Use o corpo cru e o segredo atual

Precisa de ajuda? Escreva pelo chat de suporte do aplicativo ou em **omnifox.io/help**.