AstraCalls

- Instalação e Configurações AstraCalls docker
  - AstraCalls no N8N e WhatsApp
  - Instalação AstraCalls padrão
- Funcionalidades da AstraCalls
  - IA que atende e faz ligações no WhatsApp com AstraCalls
  - N8N e WhatsApp: node grátis com botões, listas, ligações e IA no AstraCalls
  - Tutorial: AstraCalls no n8n com fluxos para botões, listas e outros recursos

Instalação e Configurações AstraCalls docker

AstraCalls no N8N e WhatsApp

Link vídeo Youtube: https://www.youtube.com/watch?v=Hs-xV2V-i1g

Github: https://github.com/AstraOnlineWeb/AstraCalls/blob/main/guias/n8n.md#-10-fluxos-de-exemplo-prontos

Neste tutorial você conecta o AstraCalls ao **n8n** e monta um robô que responde mensagens de WhatsApp. Leva uns 10 minutos e não precisa programar.

Node: `n8n-nodes-astracalls`. A documentação completa, com as 55 operações e os 15 eventos, está na página do npm.

O que você precisa

- Um AstraCalls rodando, com **pelo menos uma sessão conectada** (número pareado pelo QR Code no painel).
- A **API Key** do servidor, que é o valor de `WACALLS_API_KEY`.
- Um n8n onde você seja o dono (owner) da instância, porque só o owner instala community nodes.

Passo 1: instalar o node

1. No n8n, abra **Configurações** (⚙) → **Community Nodes** → **Install a community node**.
2. Digite `n8n-nodes-astracalls`, marque a ciência dos riscos e clique em **Install**.
3. Ao adicionar um node, procure **AstraCalls**. Aparecem dois:
   - **AstraCalls**: faz ações, como enviar mensagem, ligar ou criar grupo.
   - **AstraCalls Trigger**: inicia o fluxo quando algo acontece, como uma mensagem recebida.

> **n8n em Docker:** se preferir, adicione `N8N_COMMUNITY_PACKAGES=n8n-nodes-astracalls` nas variáveis do serviço e reinicie.

Passo 2: criar a credencial

1. **Credentials → Add credential → AstraCalls API**.
2. Preencha os dois campos:

Campo	Exemplo
URL do Servidor	`https://calls.seudominio.com`, o endereço do painel, sem `/api` no final
API Key	o valor de `WACALLS_API_KEY`

3. Clique em **Save**. Se aparecer □ , está conectado.
   - **401**: a API Key está errada.
   - **Não conecta**: a URL está errada ou o servidor está fora do ar.

Passo 3: enviar a primeira mensagem

1. Crie um fluxo com um **Manual Trigger** e adicione o node **AstraCalls** depois dele.
2. Configure o node:
   - **Recurso** `Mensagem` · **Operação** `Enviar Texto`
   - **Sessão**: escolha o número na lista
   - **Para**: `5511999999999`. Use DDI + DDD + número; com ou sem o 9º dígito funciona.
   - **Texto**: `Olá! Primeira mensagem pelo n8n □□`
3. Clique em **Test step**. A mensagem chega no WhatsApp, e a resposta do node traz o `id` dela.

Passo 4: responder mensagens automaticamente

>

> ⚡ **Atalho:** baixe `n8n/responder-mensagem.json` e importe no n8n pelo menu ⋯ → **Import from File**. Depois escolha a credencial e a sessão nos dois nodes.

Para montar na mão:

1. Adicione o **AstraCalls Trigger**:
   - **Sessão**: o número que vai receber
   - **Eventos**: `Mensagem (Recebida ou Enviada)`
   - Deixe **Ignorar Mensagens Enviadas por Mim** ligado, para o fluxo não responder a si mesmo em loop.
2. Depois do gatilho, adicione o node **AstraCalls** (`Mensagem → Enviar Texto`):
   - **Sessão**: `{{ $json.sessao }}`
   - **Para**: `{{ $json.responderPara }}`. Esse campo já traz o contato certo, ou o grupo quando a mensagem veio de um.
   - **Texto**: `Oi {{ $json.nome }}! Recebi: "{{ $json.texto }}"`
3. **Ative** o fluxo e mande uma mensagem para o número para testar.

Para responder conforme o conteúdo, coloque um **Switch** entre os dois nodes, olhando o campo `texto`. Por exemplo: se contém "preço", envia a tabela.

Passo 5 (opcional): agente de IA

Para que uma IA atenda pelo WhatsApp:

1. Use o fluxo do passo 4, mas troque o envio por um node **AI Agent**, com o modelo de sua preferência e uma memória.
2. Ligue o **AstraCalls** como **ferramenta (Tool)** do agente, por exemplo com a operação "Enviar Texto". O agente decide sozinho quando enviar mensagens, consultar contatos etc.

> Outra opção é a ferramenta **MCP Client** do n8n apontando para o `/mcp` do AstraCalls, que dá ao agente acesso a todas as ferramentas, sem instalar o node. Veja a seção "MCP para agentes de IA" no README.

□□ 10 fluxos de exemplo prontos

Baixe o `.json`, importe no n8n pelo menu ⋯ → **Import from File**, escolha a credencial e a **Sessão** em cada node AstraCalls e troque os números de exemplo (`5511999999999`). Cada fluxo tem uma nota amarela explicando como funciona.

#	Fluxo	O que faz
01	Resposta fora do horário	Responde automaticamente a quem escreve fora do expediente
02	Menu com botões	Envia um menu com botões e segue um caminho para cada opção (Switch)
03	Lista de serviços e agendamento	Envia uma lista interativa, confirma a opção escolhida e reage com
04	Atendente de IA com ferramentas	AI Agent com memória que usa o AstraCalls como ferramenta (requer credencial OpenAI)
05	Pesquisa de satisfação (enquete)	Envia uma enquete para os clientes, agradece cada voto e avisa o gestor se a nota for ruim
06	Boas-vindas no grupo	Dá boas-vindas a quem entra no grupo, mencionando a pessoa
07	Disparo de ligações com áudio	Liga para uma lista tocando um áudio gravado e acompanha o status
08	Validar lista antes da campanha	Confere quais números têm WhatsApp e só envia para eles
09	Lembrete diário com toque	Todo dia, no horário marcado, envia um lembrete e faz o telefone tocar
10	Alerta de desconexão e restrição	Avisa quando a sessão cai ou recebe restrição do WhatsApp

> Os fluxos com **AstraCalls Trigger** (01 a 06 e 10) trocam o webhook da sessão quando ativados. Ative **um por sessão**, ou junte os caminhos num só fluxo (veja o aviso abaixo).

⚠ Antes de usar em produção

- **Cada sessão tem um único webhook.** Ao ativar um fluxo com o AstraCalls Trigger, o node coloca o webhook do n8n no lugar do que estava configurado. Ao desativar, restaura o anterior. **Use um só gatilho por sessão** e separe os caminhos com o **Switch**. A integração nativa com o Chatwoot não é afetada.

- **Ligações e disparos em massa** só para quem autorizou o contato. Envio não solicitado bloqueia o número e viola a LGPD e as regras da Anatel.

Problemas comuns

Sintoma	Solução
A sessão não aparece na lista	Confira a credencial e se a sessão está conectada no painel
A sessão aparece como *aguardando QR Code*	Abra o painel do AstraCalls e pareie o número
O gatilho não dispara	O fluxo precisa estar **ativo**, e o n8n precisa ser acessível pelo servidor do AstraCalls (URL pública ou mesma rede)
Outro fluxo parou de receber eventos	Dois gatilhos na mesma sessão: o último ativado vence (veja o aviso acima)

Instalação AstraCalls padrão

Link: https://github.com/AstraOnlineWeb/AstraCalls/tree/main

git clone https://github.com/AstraOnlineWeb/AstraCalls.git

□□ AstraCalls

Chamadas de voz e vídeo do WhatsApp em Go puro, direto do navegador — prontas para produção SaaS.

VoIP nativo (áudio + vídeo, 1:1 e em grupo), gateway **SIP/PBX**, suíte completa de **mensagens**, **disparos**, **webhooks** confiáveis, integração com **Chatwoot** e deploy em **Docker Swarm + Traefik**.

Go React whatsmeow pion PostgreSQL Docker n8n License

Visão Geral · Funcionalidades · Arquitetura · Início Rápido · Configuração · Deploy · API · n8n · Suporte

---

□□ Visão Geral

> O AstraCalls pareia uma ou mais contas do WhatsApp (QR code, **código de pareamento** ou **passkey**) e permite **fazer e receber chamadas de voz e vídeo** de qualquer navegador, ligar para **tronco/ramal SIP**, e operar a conta como canal de **mensagens** (texto, mídia, botões, listas, carrossel, enquete, formulário, pix, produto, status…).
>
> O microfone e a câmera do navegador chegam ao servidor Go por **WebRTC** (ou por **WebSocket**, com fallback automático quando o WebRTC não passa no firewall). O servidor transcodifica para os codecs do WhatsApp (**MLow** para voz, **H.264** para vídeo) e injeta a mídia na malha de **relay SRTP** do WhatsApp — e o

> caminho inverso traz o áudio e o vídeo do outro lado de volta ao navegador.
>
> Toda a pilha VoIP roda **nativamente em Go**: codec MLow, RTP/SRTP/SRTCP, STUN, transporte SCTP/DTLS do relay, pipeline H.264 e a sinalização `<call>`, integrados ao **whatsmeow** e servidos a um painel **React 19**. A única dependência em C é o codec `opus_mlow` (via cgo) — e é opcional: sem ela o servidor roda em modo **somente sinalização**.
>
> Várias contas podem ser pareadas e operadas lado a lado, cada uma com QR, status, histórico e banco próprios. Uma conta mantém **várias chamadas 1:1 simultâneas** (uma por atendente), roteadas por call ID.

> **Status:** estável e em produção (última versão estável no CHANGELOG). Chamada em **grupo** é funcional (áudio + vídeo, 3+ participantes) mas segue marcada como experimental, atrás da flag `WACALLS_GROUP_CALLS`.

□□ Funcionalidades

☎ Chamadas de voz

- **Saída e entrada 1:1** com áudio bidirecional, várias chamadas simultâneas por conta.
- **Espera (hold/resume) com música**, **atender em espera** e **transferência cega** entre atendentes (troca de dono + ponte, sem tocar a perna do WhatsApp).
- **Mudo / mão levantada** (`mute_v2`, `raise_hand`) nos dois sentidos.
- **Timeout de toque**, recusa que para o telefone de quem ligou, consent freshness (RFC 7675) — a chamada não cai sozinha aos ~20s.
- **Toque fantasma** (`POST /calls/fake`): liga e desliga antes de atender (aviso/alerta).
- **Transporte de áudio no navegador:** WebRTC com **fallback automático para WebSocket** (`WACALLS_DEFAULT_TRANSPORT=auto|webrtc|websocket`); WS de áudio **PCM16 full-duplex** (`GET /calls/{id}/ws`) pronto para agentes de voz/IA.
- **Gravação opt-in por sessão**: MP3 mono mixado dos dois lados, vira nota privada no Chatwoot e evento `recording` no webhook.

□□ Chamadas de vídeo

- **Vídeo 1:1** (câmera do atendente e do cliente), **upgrade/downgrade** no meio da chamada (áudio → vídeo e volta), religar a câmera, **MediaClock** (áudio e vídeo no mesmo relógio).
- **Orientação automática** (lê a rotação do celular no RTP e no stanza `<video>`) com **botão de girar** como ajuste manual, no painel e no widget.
- **Vídeo no widget do Chatwoot** (WebCodecs, data channel H.264).
- **Disparo de ligações de VÍDEO** com arquivo gravado (`POST /broadcast {video:true}`).

□□ Chamada em grupo *(experimental, `WACALLS_GROUP_CALLS=1`)*

- Liga **para um grupo existente** (resolve os membros) com **áudio e vídeo**; o atendente ouve o **mix** de todos e fala para todos; vê a **câmera de cada participante** em tiles rotulados com número + nome.
- Protocolo completo do grupo: offer com `group_info`, chave de **epoch compartilhada com rotação** a cada entrada, relay de grupo (DTLS direto, SFU com HBH-FEC), **SRTCP** (Sender Reports + **PLI**/pedido de keyframe nos dois sentidos), anúncio `<video state>`, participante que sai some do painel.
- Endpoints `POST /calls/group`, `/calls/group/end`, WS `/calls/group/ws` (áudio PCM16) e `/calls/group/video-ws` (H.264), `GET /calls/group/roster`.

☎ Gateway SIP / PBX

- **Servidor SIP** (porta 5060 UDP/TCP) para o PBX/softphone do cliente **se registrar** no AstraCalls (credenciais por sessão, mostradas no painel) — tronco ↔ WhatsApp nos dois sentidos, áudio G.711 ↔ MLow.
- **Registro em PBX externo** (o AstraCalls vira um ramal do Asterisk/FreePBX/tronco hospedado), com **servidor, domínio e outbound proxy** separados, re-REGISTER automático e status no painel.
- RTP em **faixa fixa de portas** (`WACALLS_RTP_PORT_MIN/MAX`), publicável em `mode=host`.

□□ Conector SIP (PABX IP → AstraChat) — serviço separado

- Binário `sipconnector` na mesma imagem: liga **qualquer PABX IP** ao AstraChat, sem WhatsApp.
- Linha por caixa (o PABX registra no conector **ou** o conector registra no PABX) e **um ramal por atendente** (e-mail → ramal).

- Expõe a **mesma API de chamadas do wacalls** (SSE, accept/reject/DELETE, `/webrtc` para o navegador em PCMU sem transcodificar, `/ws` pcm16 para o agente de voz por IA) e transferência para ramal por **SIP REFER**.
- Detalhes, variáveis e deploy: `cmd/sipconnector/README.md`.

□□ Mensagens (API estilo WAHA, ~60 endpoints)

- **Envio:** texto, imagem, áudio/nota de voz (com waveform), vídeo, **nota de vídeo (PTV)**, documento, sticker, contato, localização, link com preview, **enquete** (e voto), **evento** (e resposta), **pix (BR Code)**, **produto**, **status** (texto/imagem/vídeo/áudio).
- **Interativas que renderizam no número não-oficial:** `interactive` (CTA: abrir URL, copiar código, ligar, resposta rápida), `buttons`, `list`, `carousel` e **formulário (webview)** com respostas no Chatwoot + webhook `form_response`.
- **Mensagem que some após leitura** (`disappearing`), mensagens temporárias por chat.
- **Editar, reagir, encaminhar, marcar como lida, digitando/gravando, presença.**
- **Agendamento** (`/schedule`, persiste e sobrevive a restart), **envio idempotente** (`X-Message-ID`), **busca** no histórico, **download de mídia recebida**.
- **Contatos, grupos, canais, perfil, privacidade, bloqueio, proxy por sessão**.

□□ Disparos em massa

- `POST /blast`: campanha de texto/imagem com **pacing anti-ban**, status e cancelamento.
- `POST /broadcast`: **disparo de ligações** (áudio ou vídeo) tocando um arquivo gravado.
- `GET /restriction-status`: detecta **restrição/shadow-ban** da conta (`reachout_timelock`).

□□ Webhooks confiáveis

- Eventos por sessão (`message`, `receipt`, `call_*`, `recording`, `form_response`, `group_participants`, `restriction`...), **filtro de eventos**, **assinatura HMAC-SHA256** (`X-Webhook-Signature`), **retry com backoff**, **circuit breaker**, **DLQ** com replay.

□□ Chatwoot

- Contato/conversa por telefone, mensagens **WhatsApp ↔ Chatwoot** (texto + mídia) com **1 QR só** por número; grupos e canais (opcionais); **assinatura do atendente**, **always online**, **marcar lidas**; **importar histórico sob demanda**; mensagem editada, origem de anúncio (Click to WhatsApp), nota privada da gravação.

- **Widget de chamada** (`widget.js`): botão de telefone na conversa, toca e abre sozinho na chamada recebida, áudio por WebRTC ou WebSocket, **vídeo**, token de widget por conta.
- Correção do **9º dígito BR / LID** em todos os envios; sem contato "lixo" com LID.

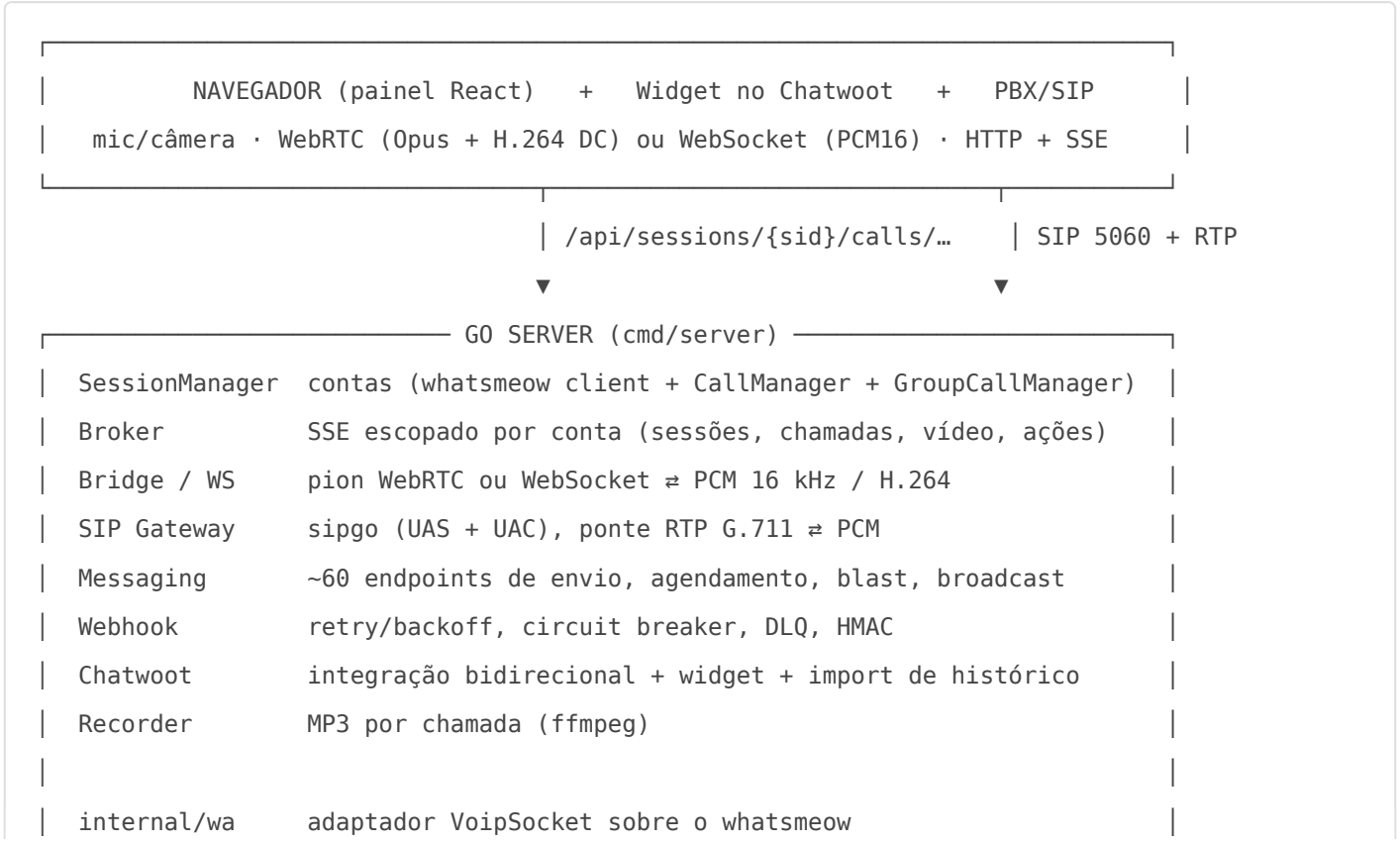
□□ Pareamento e segurança

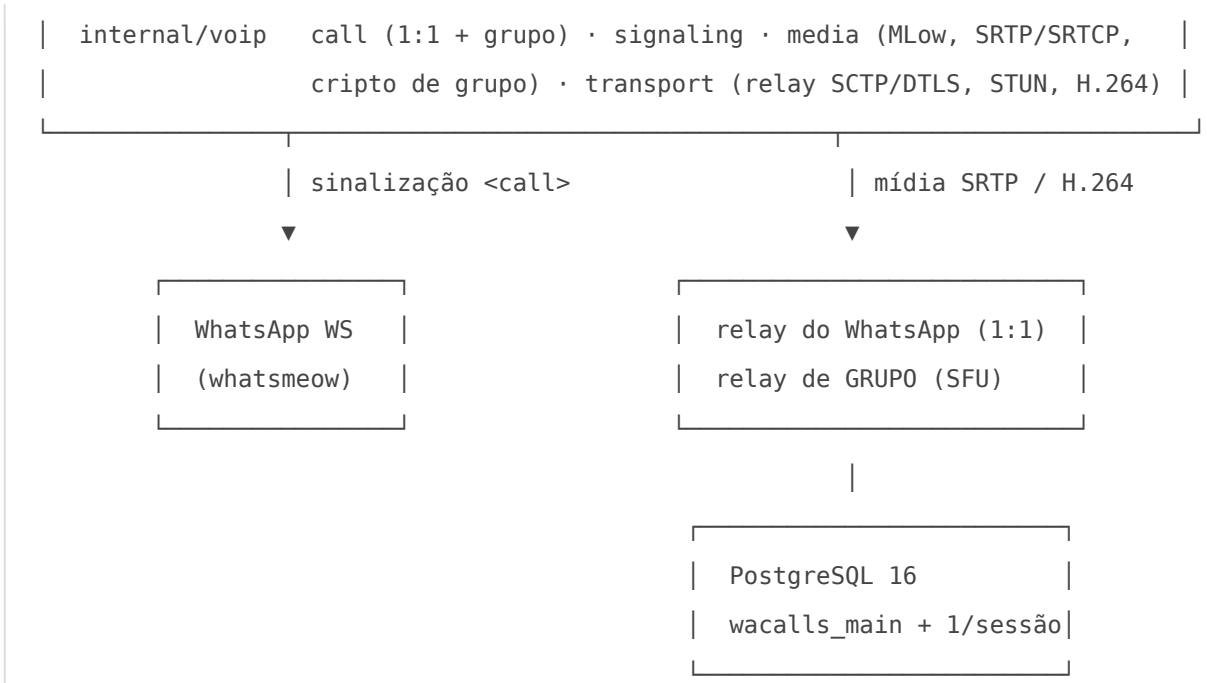
- Pareamento por **QR**, **código** (`/pair-code`) e **passkey** (`/pair-passkey`, com extensão de navegador própria em `passkey-extension/`).
- **API key** (`X-API-Key`), **tokens de widget** assinados, guarda **anti-SSRF** no download de mídia por URL, **escopo por conta** nos eventos SSE.

□□ Observabilidade e infra

- `GET /livez`, `/readyz`, `/metrics` (Prometheus), versão/commit do build em `GET /api/config` e no painel.
- **PostgreSQL por sessão** (estilo WAHA), **imagem multi-arch** (amd64 + arm64) publicada pelo CI em Docker Hub.

□□ Arquitetura





Estrutura

Caminho	Responsabilidade
`cmd/server`	HTTP/SSE, sessões + store (Postgres), ponte WebRTC/WS, auth, mensagens, agendamento, blast/broadcast, webhook, Chatwoot, gravação, **SIP** (`sip_gateway.go`, `sip_uac.go`, `sip_rtp.go`), **grupo** (`groupcall.go`)
`internal/wa`	`VoipSocket` — envia/recebe stanzas `<call>` via whatsmeow
`internal/voip/core`	Tipos de domínio, constantes, interface `VoipSocket`
`internal/voip/wanode`	Helpers de node WhatsApp e JID
`internal/voip/media`	Codec MLow, RTP, **SRTP/SRTCP**, SSRC, resampling, derivação de chaves (1:1 e **epoch de grupo**)
`internal/voip/transport`	Relay SCTP (1:1), **relay de grupo** (DTLS direto + Allocate de grupo), STUN, empacotamento H.264
`internal/voip/signaling`	Build/parse de `<call>` (offer/accept/vídeo/grupo/rekey), cripto da call-key
`internal/voip/call`	`CallManager` (1:1) e `GroupCallManager` (grupo), pipeline de vídeo, mixer de áudio
`client/`	Painel React 19 + Vite + Tailwind v4 + shadcn/ui (discador, chamadas, vídeo, grupo, sessões, SIP, Chatwoot, histórico, login)
`client/public/widget.js`	Widget de chamada (áudio + vídeo) embutível no Chatwoot
`client/public/openapi.yaml` · `api-docs.html`	Documentação OpenAPI/Swagger da API (`/api-docs.html` no servidor)
`passkey-extension/`	Extensão Chrome para pareamento por passkey

⚙ Início Rápido

```
git clone https://github.com/AstraOnlineWeb/AstraCalls.git
cd AstraCalls
git submodule update --init          # codec opus_mlow (necessário p/ áudio ao vivo)

go mod download                      # dependências Go
cd client && npm install && cd ..    # dependências do painel React
```

É preciso um **PostgreSQL** (usuário com `CREATEDB`):

```
export WACALLS_PG_URL='postgres://wacalls:senha@127.0.0.1:5432/postgres?sslmode=disable'
```

Rodar (somente sinalização — sem compilador C; pareia e chama, sem áudio)

```
go run ./cmd/server -addr :8080          # adicione -debug para logs verbosos
```

Rodar (áudio ao vivo — codec MLow nativo via cgo)

```
CGO_ENABLED=1 \
CGO_LDFLAGS="-L$PWD/native -Wl,-rpath,$PWD/native" \
go run -tags mlow ./cmd/server -addr :8080 -debug
```

> O áudio real exige cgo + `native/libopus_mlow.so` (compilada de opus_mlow). A imagem Docker já vem com tudo.

Abra `http://localhost:8080`, clique em **Nova sessão** e escaneie o QR em **WhatsApp → Aparelhos conectados** (ou use código de pareamento / passkey).

Painel React em modo dev

```
cd client
npm run dev      # Vite na :5173, faz proxy de /api → http://localhost:8080
```

⚙️ Configuração

Flags

Flag	Padrão	Significado
`-addr`	`:8080`	Endereço HTTP
`-pg-url` / `-pg-namespace`	env	URL do Postgres e prefixo dos bancos por sessão
`-static`	`client/dist`	Diretório do painel estático
`-debug`	`false`	Logs verbosos (inclui o log interno do whatsmeow)
`-max-calls-per-session`	`8`	Chamadas simultâneas por sessão (`0` = ilimitado)

Variáveis de ambiente

Env	Padrão	Significado
`WACALLS_PG_URL`	—	URL de manutenção do Postgres (usuário com `CREATEDB`)
`WACALLS_PG_NAMESPACE`	`wacalls`	Prefixo dos bancos (`wacalls_main` + `wacalls_<id>`)
`WACALLS_API_KEY`	—	Se setada, exige `X-API-Key` em `/api/*` (**use em produção**)
`WACALLS_PUBLIC_IP`	—	IP público p/ NAT 1:1 / ICE-TCP / SDP do SIP. `auto` descobre o IP real mesmo dentro do Docker/NAT de nuvem (interface pública → STUN → HTTPS); o IP achado e a fonte aparecem em `GET /api/config` (`publicIp`, `publicIpSource`)

Env	Padrão	Significado
`WACALLS_UDP_PORT`	—	Porta de mídia WebRTC (UDP + ICE-TCP)
`WACALLS_DEFAULT_TRANSPORT`	`auto`	Transporte de áudio do navegador: `auto` (WebRTC → fallback WS), `webrtc`, `websocket`
`WACALLS_MAX_CALLS`	`8`	Equivalente a `-max-calls-per-session`
`WACALLS_RING_TIMEOUT_SECONDS`	—	Expira chamada recebida que nunca é encerrada
`WACALLS_UNIFIED_SESSION_MINUTES`	`5`	Intervalo mínimo entre os sinais "unified session" enviados ao conectar, como faz o WhatsApp Web (`0` desliga)
`WACALLS_DEVICE_NAME`	`Windows`	Nome do aparelho em "Dispositivos conectados" (vale para pareamentos novos)
`WACALLS_DEVICE_BROWSER`	`chrome`	Ícone/plataforma do aparelho: `chrome`, `edge`, `firefox`, `safari` ou `desktop`
`WACALLS_RECORDING_DIR`	`$TMPDIR/wacalls-recordings`	Onde ficam os MP3s (retenção ~48h)
`WACALLS_PUBLIC_BASE_URL`	—	Base pública p/ URLs de gravação, formulários e widget (ex.: `https://call.seudominio.com`)
`WACALLS_WIDGET_KEY`	—	Chave-mestra para emitir tokens de widget (`POST /api/widget-tokens`)
`WACALLS_WIDGET_SIGNING_SECRET`	—	Segredo de assinatura dos tokens de widget
`WACALLS_SIP_ADDR`	`0.0.0.0:5060`	Endereço do servidor SIP (UDP + TCP)
`WACALLS_SIP_ADVERTISE_PORT`	`5060`	Porta SIP anunciada no `Contact` (registro em PBX externo)
`WACALLS_RTP_PORT_MIN` / `_MAX`	`40000` / `40049`	Faixa fixa de portas RTP do SIP (publique em `mode=host`)
`WACALLS_SIP_DEBUG`	off	Logs detalhados de mídia SIP/grupo (**só para diagnóstico** — gera muito log)
`WACALLS_GROUP_CALLS`	off	`1` liga a chamada em grupo (experimental)

> **Gravação de chamada (opt-in por sessão).** Ligue em `PUT /api/sessions/{sid}/recording {"enabled":true}` (ou pelo toggle "Gravar" no painel). O áudio dos dois lados é mixado num MP3 mono 16 kHz que vira **nota privada** no Chatwoot e evento `recording` no webhook; o arquivo fica em `GET /recordings/{callId}.mp3`. Requer `ffmpeg` (já na imagem).

□□ Deploy em produção (Docker Swarm + Traefik)

A imagem oficial é **multi-arquitetura** (`linux/amd64` + `linux/arm64`) e é publicada pelo CI a cada push: `astraonline/astracalls:develop` (contínua) e `astraonline/astracalls:vX.Y.Z`

- `:latest` (releases).

```
# deploy da stack (Postgres + servidor em rede de host + proxy Traefik)
docker stack deploy -c sua-stack.yml astracalls

# atualizar para a imagem mais nova
docker service update --image astraonline/astracalls:develop --force astracalls_astracalls
```

Notas de produção:

- O servidor roda em **rede de host** para a mídia WebRTC/SIP enxergar a interface real.
- Um serviço **socat** com labels do Traefik publica o HTTP em **HTTPS** (necessário porque `getUserMedia` só funciona em contexto seguro).
- O **PostgreSQL** dedicado escuta apenas em `127.0.0.1` (não exposto à internet).
- Defina `WACALLS_PUBLIC_IP` (ou `auto`), `WACALLS_UDP_PORT`, `WACALLS_PG_URL`, `WACALLS_PUBLIC_BASE_URL` e uma `WACALLS_API_KEY` forte.
- Para o **SIP**, publique `5060/udp`, `5060/tcp` e a faixa `WACALLS_RTP_PORT_MIN..MAX/udp` em `mode=host`, e aponte `WACALLS_PUBLIC_IP` para o IP real da máquina.
- Em nuvens que bloqueiam UDP de entrada, o **ICE-TCP** (WebRTC) e o transporte **WebSocket** (`WACALLS_DEFAULT_TRANSPORT=auto`) mantêm o áudio funcionando.
- Para buildar a própria imagem: `docker build -t astraonline/astracalls:develop .` (o Dockerfile compila o codec a partir do submódulo `opus_mlow`).

□□ API

A documentação completa (OpenAPI 3, **179 rotas**) é servida pelo próprio servidor em **`/api-docs.html`** (Swagger UI) e está em `client/public/openapi.yaml`.

Todas as rotas são escopadas por sessão (`/api/sessions/{sid}/…`). Os eventos chegam por um único canal SSE (`GET /api/events?clientId=…`), marcados com o `sessionId` de origem. Se `WACALLS_API_KEY` estiver setada, envie `X-API-Key` (ou `?apiKey=` no SSE/WS).

Área	Rotas (resumo)
Sessões	`GET/POST /api/sessions`, `DELETE /{sid}`, `/logout`, `/pair`, `/pair-code`, `/pair-passkey`, `/proxy`, `/recording`, `/restriction-status`
Chamadas	`POST /calls` (`{phone, video?, record?}`), `/calls/{id}/webrtc` (SDP), `GET /calls/{id}/ws` (PCM16), `/accept`, `/reject`, `DELETE /calls/{id}`, `/hold`, `/resume`, `/pickup`, `/transfer`, `/mute`, `/hand`, `/video/{request|accept|reject|stop}`, `/spy` (espiar: WebRTC recvonly / `GET /spy/ws` PCM16), `/volume` (ganho 0..3), `/silence` (silenciar o toque só na tela), `POST /calls/fake`, `GET /calls`, `GET /history`
Grupo	`POST /calls/group`, `POST /calls/group/end`, `GET /calls/group/ws`, `GET /calls/group/video-ws`, `GET /calls/group/roster`
Disparos	`POST /broadcast` (ligações, áudio/vídeo), `POST /blast` + `GET /blasts`, `/blasts/{id}`, `/blasts/{id}/cancel`
Mensagens	`POST /messages/{text|image|audio|video|ptv|document|sticker|contact|location|link-preview|poll|poll-vote|event|event-response|pix|product|product-native|interactive|buttons|list|carousel|form|disappearing|seen|typing|forward}`, `PUT /messages/{edit|react}`, `GET /messages`, `/messages/search`, `/messages/{id}/media`, `/messages/new-message-id`, `DELETE /messages`, `/schedule`
Status / presença	`POST /status/{text|image|video|audio}`, `POST /presence`, `/presence/{jid}/subscribe`
Contatos / grupos / canais	`GET /contacts`, `/contacts/check`, `/contacts/{jid}`, `/picture`, `/business`, `/block`, `/unblock`, `GET /blocklist`; `GET/POST /groups`, `/groups/{gid}/…` (participantes, convite, pedidos, configurações, foto, assunto, descrição, sair; `POST /groups/{gid}/history/share` e `shareHistory` no add: compartilhar o histórico recente com quem entrou); `GET/POST /communities`, `/communities/{cid}` (+ grupos), `/communities/{cid}/groups` (listar/criar), `/groups/link`, `/groups/unlink`, `/participants` (+ `/participants/add`); `GET /channels`, `/channels/{id}/…` (seguir, mutar, mensagens)
Chats / perfil / privacidade	`GET /chats`, `/chats/{chatId}/messages`, `PUT /chats/{chatId}/disappearing`, `PUT /disappearing`; `GET /profile`, `/profile/qr`, `PUT /profile/status`; `GET/PUT /privacy`, `GET /privacy/status`
Webhooks	`GET/POST/DELETE /webhook` (eventos, `secret` HMAC), `GET /webhooks/status`, `/webhooks/dlq`, `POST /webhooks/dlq/{id}/replay`, `/webhooks/reenable`
Chatwoot	`GET/POST/DELETE /chatwoot`, `POST /chatwoot/webhook`, `/chatwoot/import-history`, `/chatwoot/groups/{gid}/open`, `/chatwoot/channels/{id}/open`, `GET /api/chatwoot/resolve`

Área	Rotas (resumo)
SIP	`GET /api/sip/status`, `GET/POST /sip` (credenciais p/ o PBX registrar aqui), `GET/POST /sip-ext` (registrar num PBX externo: host, porta, usuário, senha, ramal, proxy)
Widget / formulários	`GET /api/widget-key`, `POST /api/widget-tokens`, `GET /forms/{token}`, `POST /forms/{token}/submit` (públicas, token como capability)
Infra	`GET /api/config` (flags, versão, commit), `GET /livez`, `/readyz`, `/metrics`, `GET /recordings/{id}`

□□ Integração com n8n

Automatize o WhatsApp no **n8n sem escrever código**: o node oficial `n8n-nodes-astracalls` já está publicado no npm, todo em português.

Instalação: no n8n, abra **Configurações → Community Nodes → Install**, digite `n8n-nodes-astracalls` e clique em **Install**. No Docker, use `N8N_COMMUNITY_PACKAGES=n8n-nodes-astracalls`.

Credencial: a URL do seu AstraCalls e a API Key (`WACALLS_API_KEY`).

Node	O que faz
AstraCalls	55 operações em 8 recursos: **Mensagem** (texto, mídia, documento, enquete, botões, lista, formulário nativo, reagir, editar, apagar…), **Chamada** (ligar, ligação fantasma, disparo de ligações com áudio/vídeo), **Grupo**, **Comunidade**, **Contato**, **Sessão**, **Formulário** e **Requisição Personalizada** (qualquer uma das 179 rotas)
AstraCalls Trigger	Inicia o fluxo com 15 eventos: mensagem recebida, leitura, reação, voto em enquete, formulário respondido, entrada/saída de grupo, gravação de chamada, desconexão da sessão… Configura o webhook da sessão sozinho ao ativar

- **Fácil de usar:** a sessão é escolhida numa lista (nome + número), o destino aceita número com ou sem o 9º dígito e a mídia vem por URL ou de um arquivo de etapa anterior.
- **Pronto para responder:** o gatilho entrega `de`, `nome`, `texto` e `responderPara` prontos para usar no envio, e ignora as mensagens da própria conta (sem loop).
- **Agentes de IA:** o node pode ser usado como ferramenta pelos agentes de IA do n8n.

□□ **Tutorial passo a passo**: instalar, criar a credencial, enviar a primeira mensagem e montar um robô que responde, com fluxo pronto para importar e 10 fluxos de exemplo (menu com botões,

atendente de IA, enquete, disparo de ligações…).

□□ MCP para agentes de IA

O AstraCalls traz um **servidor MCP** (Model Context Protocol) embutido: Claude, ChatGPT, Cursor, VS Code, n8n e outros agentes de IA usam o WhatsApp direto, sem instalar nada.

Chaves por sessão (recomendado): no painel, botão **MCP** da sessão → **Nova chave**. A chave fica presa àquele número, pode ser **somente leitura** e é revogada a qualquer momento. Ela não permite parear, desconectar nem mexer em webhooks, Chatwoot, SIP ou proxy. O painel mostra, uma única vez, a URL e as configurações prontas para copiar. Pela API: `GET/POST /api/sessions/{sid}/mcp-keys` e `DELETE /api/sessions/{sid}/mcp-keys/{id}`.

Cliente	Como conectar
claude.ai, ChatGPT (conector personalizado)	URL `https://SEU-ASTRACALLS/mcp/<chave>`
Claude Code	`claude mcp add --transport http astracalls https://SEU-ASTRACALLS/mcp --header "Authorization: Bearer <chave>"`
Claude Desktop, Cursor, VS Code	`mcpServers` → `{"type":"http","url":"https://SEU-ASTRACALLS/mcp","headers":{"Authorization":"Bearer <chave>"}}`
n8n	Ferramenta **MCP Client** do agente de IA, endereço `/mcp` e autenticação *Bearer*

Acesso de administrador: `https://SEU-ASTRACALLS/api/mcp` com a API Key da instância (`WACALLS_API_KEY`) enxerga todas as sessões; o agente escolhe o número por nome, telefone ou ID.

44 ferramentas: enviar texto, mídia, documento, localização, contato, enquete, botões e lista; reagir, editar, apagar, marcar como lida e "digitando..."; **ler conversas e buscar mensagens**; ligar, encerrar, toque fantasma e disparo de ligações; contatos, grupos e comunidades; limite de conversas novas e restrição; e `api_request` para as demais rotas da sessão.

> Quem tem a chave usa o número: trate-a como senha e revogue as que não usa mais.

□□ Testes

```
go test ./...                # mídia (SRTP/SRTCP, STUN, RTP, H.264, cripto de grupo), sinalizaç
cd client && npm run build   # type-check + build de produção do painel
```

□□ Segurança

- Em produção, **sempre** defina `WACALLS_API_KEY` — sem ela qualquer um com acesso HTTP pode criar contas, fazer chamadas, enviar mensagens e ler histórico.
- O banco de cada sessão guarda **credenciais do WhatsApp**. Mantenha o Postgres protegido e fora da internet.
- Exponha sempre por **HTTPS** (`getUserMedia` exige contexto seguro).
- Use `secret` no webhook e verifique `X-Webhook-Signature` no consumidor.
- O SIP na 5060 recebe varredura da internet o tempo todo: só sessões com credenciais válidas registram (Digest MD5); considere restringir a porta por firewall aos IPs dos PBXs.

□□ Suporte Profissional

Precisa de ajuda para melhorar, customizar ou implementar o projeto?

□□ **WhatsApp:** +55 61 9 9687-8959

□□ Temos uma equipe especializada para:

□ Customizações e melhorias □ Implementação e deploy completo □ Configuração de arquitetura SaaS □ Integração com outras APIs □ Desenvolvimento de features específicas □ Suporte técnico dedicado □ Consultoria em automação WhatsApp □ Treinamento e documentação

---

□□ Licença

O AstraCalls é distribuído sob a licença **GNU AGPL-3.0** — veja LICENSE.

Isso significa que qualquer uso em rede (inclusive SaaS) exige disponibilizar o código-fonte das modificações aos usuários do serviço.

O AstraCalls é um fork do WaCalls, que é licenciado sob **MIT**. Conforme exigido pela MIT, o aviso de copyright original (© 2026 jotadev66) é preservado em LICENSE.WaCalls. As porções originais permanecem sob os termos MIT; o trabalho derivado, como um todo, é licenciado sob AGPL-3.0.

Funcionalidades da AstraCalls

Lista de funcionalidades possíveis com a AstraCalls

IA que atende e faz ligações no WhatsApp com AstraCalls

Link: https://chatwoot.astraonline.com.br/blog/ia-atende-ligacao-whatsapp/

Vídeo Youtube: https://www.youtube.com/watch?v=8tilgREKqv8

Download dos códigos: https://astraonline.com.br/wp-content/uploads/2026/10/Disparador-de-Ligacoes-e-Atendimento-com-IA.zip

Por Raphael Popiolek · 28 de outubro de 2026 · Atualizado em 03 de novembro de 2026

Uma IA que atende ligação no WhatsApp recebe a chamada, escuta o cliente e responde por voz, e também pode ligar sozinha para uma lista de números, deixando cada chamada gravada e transcrita num painel.

Muita empresa ainda perde ligação no WhatsApp porque ninguém está livre para atender, ou gasta horas de alguém ligando para cobrar e confirmar horário. Uma **IA que atende ligação no WhatsApp** resolve as duas pontas.

No vídeo eu monto esse sistema do zero, numa VPS limpa, usando o AstraCalls para a parte de chamadas e o Google Gemini com voz em tempo real para a conversa. Quem escreveu o código foi o Claude Code, rodando direto no servidor. Aqui eu explico o que foi feito, o que você precisa ter em mãos e os cuidados que ninguém deveria pular.

https://www.youtube.com/embed/8tilgREKqv8?rel=0

O que esse sistema faz na prática

O sistema montado no vídeo é um painel web com login e senha, rodando no seu próprio subdomínio. Dentro dele:

- **Conexões:** você conecta um ou mais números de WhatsApp lendo o QR Code;
- **Ligações:** a lista de chamadas recebidas e feitas, com a gravação para ouvir e a transcrição da conversa;
- **Disparador:** você cola uma lista de números e o sistema liga para cada um, podendo alternar entre as conexões;

- **Configurações:** o prompt do agente, a mensagem inicial, a chave da API do Gemini, o modelo e a voz, além das opções de atender automaticamente e gravar as chamadas.

No teste do vídeo eu liguei para o número conectado e pedi um agendamento. A IA atendeu, perguntou o assunto, pediu meu nome e empresa e confirmou o horário. Depois disso a chamada já apareceu no painel, gravada e transcrita.

As peças por trás do painel

Vale entender o que cada parte faz, porque é isso que você vai ajustar depois:

Peça	Função
AstraCalls	API de WhatsApp open source e não oficial que faz as chamadas de áudio (WebRTC)
Google Gemini (modelo de voz em tempo real)	Escuta e responde durante a ligação
SQLite	Banco simples para guardar conexões, ligações e configurações
Docker, Docker Swarm, Traefik e Portainer	Sobem o projeto e cuidam do SSL e do subdomínio
Claude Code	Escreve e publica o projeto a partir de um prompt

Escolhi o SQLite de propósito. O projeto é para uso de uma empresa, não para revenda, então um banco leve deixa a VPS menor e mais barata. Se a ideia for vender como serviço para vários clientes, aí a conversa é outra.

Sobre o modelo de voz: eu uso o Gemini porque gosto da voz e o custo é bom. A OpenAI também flui muito bem, mas no meu uso sai mais caro. Se for trocar, confira se o modelo responde por voz em tempo real. Numa ligação não dá para converter texto em áudio com atraso, senão a conversa fica travada.

O que você precisa antes de começar

1. Uma VPS com **2 a 4 GB de RAM** e Ubuntu 22. No vídeo usei uma de 4 GB na Hetzner, mas qualquer provedor serve.
2. Acesso SSH como root (IP, porta 22 e senha). Eu uso o Bitvise para conectar.
3. Uma conta paga do Claude. O projeto é pequeno, mas a conta gratuita provavelmente vai travar no meio pedindo upgrade.

4. Um domínio com acesso ao DNS (eu uso o Cloudflare) para criar um subdomínio apontando para o IP da VPS.
5. Uma chave de API do Gemini, criada no Google AI Studio.
6. Um número de WhatsApp de teste para conectar.

O prompt completo e o tutorial com os comandos estão na página de apoio do vídeo, para baixar.

Passo a passo resumido

1. Crie a VPS e acesse pelo SSH.
2. Instale o Claude Code com o comando do tutorial e faça a autenticação pelo navegador. Ao copiar o código, aperte Ctrl+C uma vez só; duas vezes fecha o programa.
3. Crie uma pasta para o projeto (eu chamei de `atendimento`) e entre nela.
4. Abra o Claude Code no modo que não pede confirmação a cada passo, para ele executar até o fim sem você ficar apertando enter.
5. No DNS, crie o subdomínio apontando para o IP da VPS.
6. Troque, dentro do prompt, o subdomínio de exemplo pelo seu. Se preferir acessar por IP e porta, peça isso no prompt.
7. Cole o prompt e espere. No vídeo levou por volta de **10 minutos**.
8. Entre no endereço com o e-mail e a senha que o Claude Code gerou.
9. Em Configurações, cole a chave do Gemini, escolha modelo e voz e salve.
10. Em Conexões, crie uma conexão e leia o QR Code com o número de teste.
11. Ligue para esse número de outro celular e veja a IA atender.

O prompt que deixo é a estrutura. O texto do agente vem com um padrão genérico ("você é um atendente virtual simpático e objetivo"), e é você quem escreve o que ele deve fazer na sua empresa.

Por que rodar numa VPS e não no seu computador

Dá para rodar local, mas eu não recomendo. O AstraCalls é sensível a ambiente, e cada computador tem um sistema diferente, o que gera conflito. Na VPS o ambiente é padronizado e o sistema fica no ar 24 horas: se cair a luz ou a internet do escritório, as ligações continuam. E no mesmo servidor você ainda pode colocar outros projetos.

Um erro que apareceu no teste: o nono dígito

Quando fui disparar uma ligação, a chamada não tocou. O problema era o nono dígito do celular. Tirei o 9 do número e a ligação saiu na hora. O painel marcou certinho uma chamada de saída (do disparador) e uma de entrada (a que eu fiz antes).

A lição: peça no prompt que o sistema trate o nono dígito e limpe os números antes de disparar. Eu já deixei esse ajuste no prompt salvo.

Onde isso ajuda e onde exige cuidado

Os usos mais diretos são cobrança, confirmação e marcação de horário e prospecção ativa. A IA liga, conversa e deixa tudo gravado e transcrito para alguém conferir depois.

Agora o lado honesto:

- **Risco de bloqueio.** Disparo em massa sempre pode levar a banimento. Use números separados para isso, nunca o número principal da empresa, e não coloque em risco o número de um cliente seu. Os cuidados para ligar para uma lista estão no artigo sobre disparo de ligações no WhatsApp com n8n. A Meta costuma ficar mais rigorosa em ano de eleição.
- **API não oficial.** O AstraCalls não é a API oficial da Meta. Funciona bem, mas o risco de bloqueio do número existe.
- **O prompt é com você.** Sem um prompt bem escrito, a IA vai atender de forma genérica. Ela não sabe seus preços, horários ou regras se você não contar.
- **Infraestrutura.** Mesmo com o Claude Code fazendo o grosso, você precisa saber acessar uma VPS, mexer em DNS e conferir se o projeto subiu.

Se o que você precisa é que os atendentes da equipe façam e recebam ligações dentro do Chatwoot, sem IA no meio, veja o artigo sobre ligações do WhatsApp no Chatwoot com o AstraCalls. E se quiser um agente de IA que responda por texto, com memória e transferência para humano, o caminho é este: chatbot com IA no WhatsApp.

Checklist antes de colocar em produção

- ☐ Subdomínio apontado e com SSL funcionando
- ☐ Prompt do agente escrito para o seu negócio, com o que ele pode e não pode prometer
- ☐ Tratamento do nono dígito testado com um número real
- ☐ Gravação e transcrição conferidas em algumas chamadas
- ☐ Número usado no disparo separado do número principal
- ☐ Backup da VPS configurado

Perguntas frequentes

Existe IA que atende ligação no WhatsApp?

Sim. O painel montado no vídeo usa o AstraCalls para as chamadas e o Google Gemini com voz em tempo real para a conversa. A IA atende, escuta o cliente, responde por voz e a chamada fica gravada e transcrita no painel.

Como fazer uma IA ligar para clientes pelo WhatsApp?

No mesmo painel há um disparador: você cola uma lista de números e o sistema liga para cada um, podendo alternar entre as conexões. Os usos mais diretos são cobrança, confirmação de horário e prospecção ativa.

Qual VPS usar para uma IA de voz no WhatsApp?

Uma VPS com 2 a 4 GB de RAM e Ubuntu 22, com acesso SSH como root. No vídeo foi usada uma de 4 GB, e o projeto usa SQLite justamente para manter o servidor menor e mais barato.

Qual modelo de IA usar para atender ligações?

O vídeo usa o Gemini com voz em tempo real, pela voz e pelo custo. A OpenAI também flui bem, mas sai mais cara no uso mostrado. Se trocar, confira se o modelo responde por voz em tempo real, senão a conversa fica travada.

Posso rodar a IA de ligação no meu computador?

Dá, mas não é recomendado. O AstraCalls é sensível a ambiente e cada computador tem um sistema diferente, enquanto a VPS tem ambiente padronizado e fica no ar 24 horas, mesmo se cair a luz ou a internet do escritório.

Quer isso junto do seu atendimento?

Esse painel é um projeto à parte. Se você quer as ligações e a IA dentro do mesmo lugar onde a equipe atende, o **AstraChat**, a nossa versão do Chatwoot, já traz ligações pelo WhatsApp, agentes de IA, Kanban, disparador e flows, no seu servidor, com a sua marca e sem limite de atendentes.

Ele vem na Comunidade Astra Online, com código-fonte, atualizações, grupo VIP e a primeira instalação feita pelo especialista em Chatwoot Raphael Popiolek. Se quiser um projeto de voz sob medida, integrado ao seu sistema, chame a gente no WhatsApp e conte o que precisa automatizar.

Falar com um especialistaConhecer o AstraChat

Mais artigos de IA no atendimento e o Guia completo do Chatwoot, com todos os assuntos organizados.

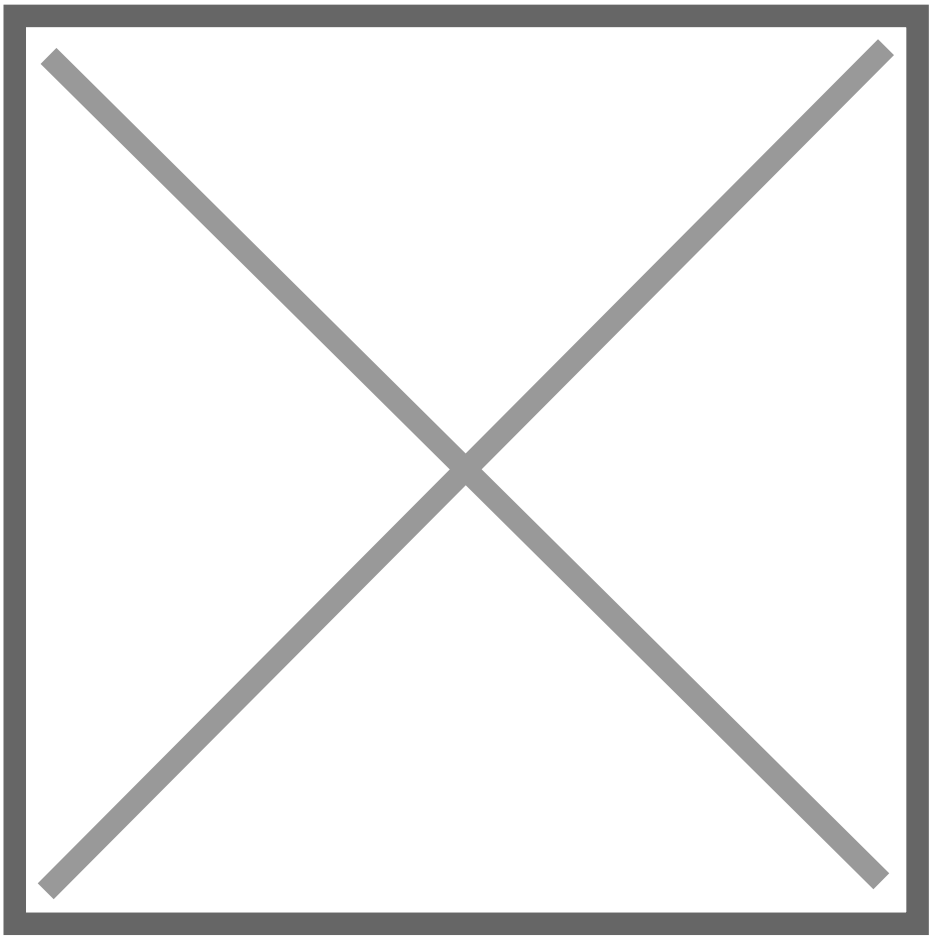
Leia também



28 de outubro de 2026

Disparo de ligações WhatsApp com n8n: áudio e vídeo gravados

Disparo de ligações WhatsApp com n8n e AstraCalls: o sistema liga, toca seu áudio ou vídeo gravado e marca quem atendeu. Veja o fluxo e os cuidados.



28 de outubro de 2026

Ligação WhatsApp no Chatwoot: o atendente liga e atende na tela

Ligação WhatsApp no Chatwoot com o AstraCalls: instale a API gratuita e ative um widget para o atendente ligar e receber chamadas sem sair da conversa.



05 de novembro de 2026

n8n e WhatsApp: node grátis com botões, listas, ligações e IA

Conecte o WhatsApp ao n8n com o node do AstraCalls: instalação, credencial, gatilho, botões, listas, ligações, agente de IA e 10 fluxos prontos.

N8N e WhatsApp: node grátis com botões, listas, ligações e IA no AstraCalls

Link: https://chatwoot.astraonline.com.br/blog/n8n-whatsapp-node-astracalls/

Vídeo Youtube: https://youtu.be/Hs-xV2V-i1g

Fluxos download: https://github.com/AstraOnlineWeb/AstraCalls/blob/main/guias/n8n.md#-10-fluxos-de-exemplo-prontos

Tutorial: AstraCalls n... | Portal de Conhecimento

Por Raphael Popiolek · 05 de novembro de 2026

Para ligar o **WhatsApp no n8n** sem montar requisição HTTP, instale o community node `n8n-nodes-astracalls`, crie uma credencial com a URL e a API Key do seu AstraCalls e use os dois nodes que aparecem: o **AstraCalls Trigger**, que inicia o fluxo quando chega uma mensagem, e o **AstraCalls**, que envia texto, botões, listas, enquetes e faz ligações. O AstraCalls é a nossa API de WhatsApp open source, e o node é gratuito.

No vídeo eu instalo o node, mostro a credencial, o gatilho, os recursos e testo ao vivo uma lista interativa chegando no WhatsApp. Aqui está o passo a passo escrito, com os 10 fluxos prontos que deixamos para você importar.

https://www.youtube.com/embed/Hs-xV2V-i1g?rel=0

O que você precisa

- Um **AstraCalls** instalado, com pelo menos um número conectado pelo QR Code no painel.
  A instalação está no artigo sobre ligações do WhatsApp no Chatwoot.
- A **API Key** do servidor, que é o valor da variável `WACALLS_API_KEY` da stack.

- Um **n8n** em que você seja o dono (owner) da instância. Só o owner instala community nodes.

O que o node faz

São dois nodes, os dois em português:

Node	Função
AstraCalls	55 operações em 8 recursos: Mensagem, Chamada, Grupo, Comunidade, Contato, Sessão, Formulário e Requisição Personalizada, que chama qualquer uma das 179 rotas da API
AstraCalls Trigger	Inicia o fluxo com 15 eventos, como mensagem recebida, leitura, reação, voto em enquete, formulário respondido, entrada e saída de grupo, gravação de chamada e desconexão da sessão

A vantagem sobre o node HTTP Request é que você não monta URL, cabeçalho e corpo a cada chamada. A sessão é escolhida numa lista com nome e número, o destino aceita número com ou sem o nono dígito, e o gatilho já entrega os campos prontos para responder.

Passo 1: instalar o node no n8n

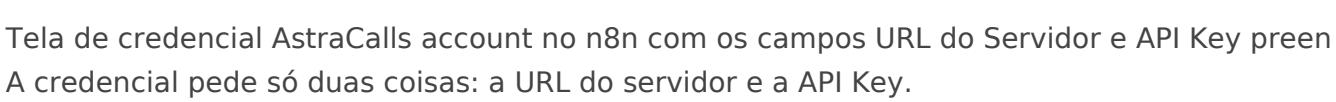
1. No n8n, abra **Settings** e depois **Community Nodes**.
2. Clique em **Install**, digite `n8n-nodes-astracalls`, marque a ciência dos riscos e confirme.
3. Ao adicionar um node no fluxo, procure por **AstraCalls**. Os dois nodes aparecem.

> Janela Install community nodes do n8n com o pacote n8n-nodes-astracalls digitado e o botão I
> Instalação pelo Community Nodes. Ao fundo, o pacote já instalado com os dois nodes.

Se o seu n8n roda em Docker, outra forma é colocar `N8N_COMMUNITY_PACKAGES=n8n-nodes-astracalls` nas variáveis do serviço e reiniciar. Quando sair versão nova, o próprio Community Nodes mostra o botão de atualizar.

Passo 2: criar a credencial

1. Vá em **Credentials**, **Add credential** e escolha **AstraCalls API**.
2. Em **URL do Servidor**, coloque o endereço do painel do AstraCalls, sem `/api` no final.
3. Em **API Key**, cole o valor de `WACALLS_API_KEY`.
4. Salve. Se der erro 401, a chave está errada. Se não conectar, confira a URL e se o servidor está no ar.

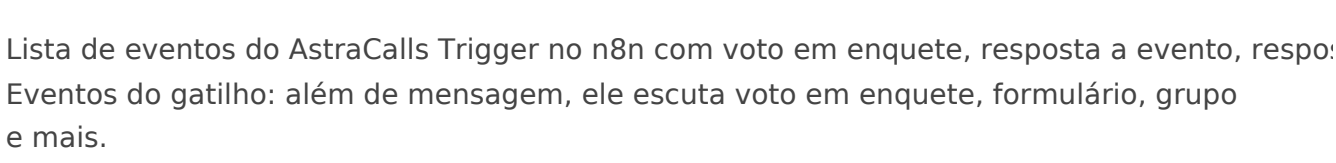


Tela de credencial AstraCalls account no n8n com os campos URL do Servidor e API Key preen
A credencial pede só duas coisas: a URL do servidor e a API Key.

Passo 3: o gatilho e a primeira resposta automática

O fluxo mais simples responde quem manda mensagem:

1. Adicione o **AstraCalls Trigger**, escolha a sessão e o evento **Mensagem (Recebida ou Enviada)**.
2. Deixe ligadas as opções **Ignorar Mensagens Enviadas por Mim**, para o fluxo não responder a si mesmo em loop, e **Ignorar Mensagens de Grupos**, se não quiser responder em grupo.
3. Depois do gatilho, adicione o node **AstraCalls** com Recurso **Mensagem** e Operação **Enviar Texto**.
4. Em **Para**, use `{{ $json.responderPara }}`. Esse campo já traz o contato certo, ou o grupo quando a mensagem veio de um.
5. No texto, dá para usar `{{ $json.nome }}` e `{{ $json.texto }}` para personalizar.
6. Ative o fluxo e mande uma mensagem para o número.



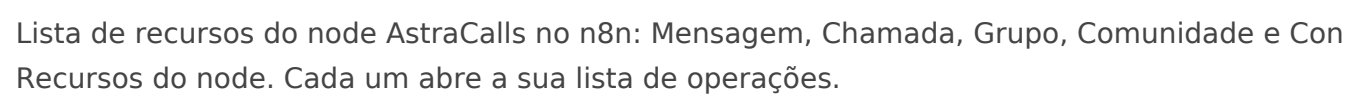
Lista de eventos do AstraCalls Trigger no n8n com voto em enquete, resposta a evento, respos
Eventos do gatilho: além de mensagem, ele escuta voto em enquete, formulário, grupo e mais.

Para responder conforme o conteúdo, coloque um **Switch** entre o gatilho e o envio, olhando o campo `texto`. Se contém "preço", manda a tabela; se contém "horário", manda o expediente.

Uma dica do vídeo: antes de responder, use a operação **Mostrar "Digitando..."** e um node **Wait** de alguns segundos. A resposta parece mais humana.

Recursos e operações

Dentro do node AstraCalls, você escolhe um recurso e depois a operação:



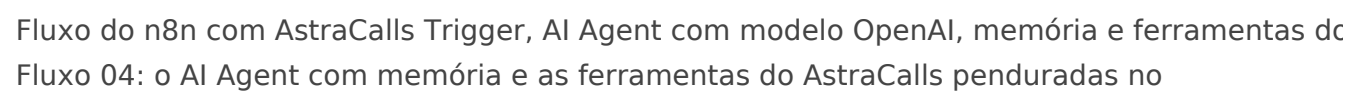
Lista de recursos do node AstraCalls no n8n: Mensagem, Chamada, Grupo, Comunidade e Con
Recursos do node. Cada um abre a sua lista de operações.

- **Mensagem:** texto, imagem, vídeo, áudio, figurinha, localização, contato, botões, listas, enquetes, formulários nativos, reação, edição, apagar e mostrar digitando.
- **Chamada:** ligar, encerrar, ligação fantasma (o telefone do cliente toca e a chamada não completa, só para chamar a atenção) e disparo de ligações com áudio ou vídeo gravado, com status e histórico.
- **Contato:** verificar se um número tem WhatsApp, buscar foto e bloquear.
- **Grupo e Comunidade:** criar e administrar grupos e comunidades.
- **Sessão:** operações da conta de WhatsApp conectada.
- **Formulário:** formulários nativos do WhatsApp Business.
- **Requisição Personalizada:** qualquer rota da API que não tenha operação pronta.

A documentação completa, em Swagger, fica no próprio painel do AstraCalls, no ícone de livro no alto da tela.

Agente de IA no n8n que usa o WhatsApp como ferramenta

O node AstraCalls também pode ser ligado como **ferramenta (Tool)** do node **AI Agent** do n8n. O agente decide sozinho quando enviar mensagem, mandar localização, verificar se um número tem WhatsApp ou ligar para o cliente.



Fluxo do n8n com AstraCalls Trigger, AI Agent com modelo OpenAI, memória e ferramentas do
Fluxo 04: o AI Agent com memória e as ferramentas do AstraCalls penduradas no conector Tool.

Para montar:

1. Parta do fluxo do passo 3 e troque o envio direto por um node **AI Agent**, com o modelo de sua preferência e uma memória.
2. No conector **Tool** do agente, adicione nodes AstraCalls com as operações que ele pode usar, por exemplo Enviar Texto, Enviar Localização e Ligar.
3. Escreva na descrição de cada ferramenta quando o agente deve usá-la.
4. Depois do agente, um node AstraCalls envia a resposta para `{{ $json.responderPara }}`.

No n8n self-hosted, nodes da comunidade só funcionam como ferramenta com a variável `N8N_COMMUNITY_PACKAGES_ALLOW_TOOL_USAGE=true` no serviço.

Outra opção é a ferramenta **MCP Client** do n8n apontando para o endereço `/mcp` do AstraCalls, com autenticação Bearer. Assim o agente acessa todas as ferramentas sem instalar o node. O mesmo MCP funciona fora do n8n, como mostro no artigo sobre WhatsApp no ChatGPT e no Claude com MCP. Para a parte de prompt, contexto e passagem para humano, veja como criar um agente de IA para WhatsApp.

Os 10 fluxos prontos para importar

Deixamos 10 fluxos de exemplo no guia do n8n no GitHub. Baixe o `.json`, importe pelo menu **Import from File**, escolha a credencial e a sessão em cada node AstraCalls e troque os números de exemplo (`5511999999999`). Cada fluxo tem uma nota amarela explicando como funciona.

#	Fluxo	O que faz
01	Resposta fora do horário	Responde quem escreve fora do expediente
02	Menu com botões	Envia um menu e segue um caminho para cada botão
03	Lista de serviços e agendamento	Envia uma lista interativa, confirma a opção e reage à mensagem
04	Atendente de IA com ferramentas	AI Agent com memória que usa o AstraCalls como ferramenta
05	Pesquisa de satisfação	Envia uma enquete, agradece cada voto e avisa o gestor se a nota for ruim
06	Boas-vindas no grupo	Dá boas-vindas a quem entra, mencionando a pessoa

#	Fluxo	O que faz
07	Disparo de ligações com áudio	Liga para uma lista tocando um áudio gravado e acompanha o status
08	Validar lista antes da campanha	Confere quais números têm WhatsApp e só envia para eles
09	Lembrete diário com toque	Envia um lembrete no horário marcado e faz o telefone tocar
10	Alerta de desconexão e restrição	Avisa quando a sessão cai ou recebe restrição do WhatsApp

No teste do vídeo, disparei o fluxo 03 e a lista chegou assim no celular:

> WhatsApp no celular mostrando a lista interativa Ver serviços com as opções Consulta, Retorn
> Lista interativa enviada pelo fluxo 03: o cliente escolhe e a opção volta para o n8n.

No vídeo eu destaco o fluxo 10 para quem dá suporte a clientes: você fica sabendo que o número caiu ou foi restringido antes do cliente reclamar. O fluxo 07 está detalhado no artigo sobre disparo de ligações no WhatsApp com n8n.

Antes de usar em produção

- **Um gatilho por sessão.** Cada sessão do AstraCalls tem um único webhook. Ao ativar um fluxo com o AstraCalls Trigger, ele assume o webhook da sessão e devolve o anterior quando é desativado. Se ativar dois fluxos com gatilho no mesmo número, o último vence. Junte os caminhos num fluxo só, com o Switch. A integração nativa com o Chatwoot não é afetada.
- **Listas e botões dependem do aparelho.** A nota do fluxo 03 avisa que a lista aparece no celular e pode não aparecer no WhatsApp Web. Teste no aparelho do cliente.
- **Ligações e disparos só para quem autorizou.** Envio em massa não solicitado bloqueia o número e viola a LGPD e as regras da Anatel. O AstraCalls é uma API não oficial, então use um número separado do principal para disparos.

Problemas comuns

Sintoma	O que fazer
A sessão não aparece na lista	Confira a credencial e se o número está conectado no painel
A sessão aparece aguardando QR Code	Abra o painel do AstraCalls e pareie o número
O gatilho não dispara	O fluxo precisa estar ativo e o n8n precisa ser acessível pelo servidor do AstraCalls
Outro fluxo parou de receber eventos	Há dois gatilhos na mesma sessão; deixe só um
O agente não enxerga o AstraCalls como ferramenta	Falta a variável `N8N_COMMUNITY_PACKAGES_ALLOW_TOOL_USAGE=true`

Perguntas frequentes

Como conectar o WhatsApp no n8n?

Instale o community node `n8n-nodes-astracalls` em Settings, Community Nodes, crie uma credencial com a URL e a API Key do seu AstraCalls e use o AstraCalls Trigger para receber mensagens e o node AstraCalls para enviar. O número fica conectado no AstraCalls pelo QR Code.

O node do AstraCalls para n8n é grátis?

Sim. O node é gratuito e o AstraCalls é open source, sem cobrança por mensagem. Você paga só o servidor onde instala a API. Por ser uma API não oficial, existe risco de bloqueio do número em disparos em massa.

Dá para mandar botões e listas no WhatsApp pelo n8n?

Sim. O recurso Mensagem do node tem as operações de botões, listas, enquetes e formulários nativos, e o gatilho recebe a opção que o cliente escolheu. Os fluxos 02 e 03 do guia trazem exemplos prontos de menu com botões e de lista de serviços.

Como fazer um agente de IA no n8n responder pelo WhatsApp?

Use o AstraCalls Trigger para receber a mensagem, um node AI Agent com modelo e memória e ligue nodes AstraCalls no conector Tool do agente. No n8n self-hosted, ative a variável `N8N_COMMUNITY_PACKAGES_ALLOW_TOOL_USAGE=true`. O fluxo 04 do guia já vem montado.

Dá para fazer ligação de WhatsApp pelo n8n?

Sim. O recurso Chamada do node liga, encerra, faz ligação fantasma e dispara ligações com áudio ou vídeo gravado para uma lista, com status e histórico de cada chamada.

Quer isso rodando sem montar servidor?

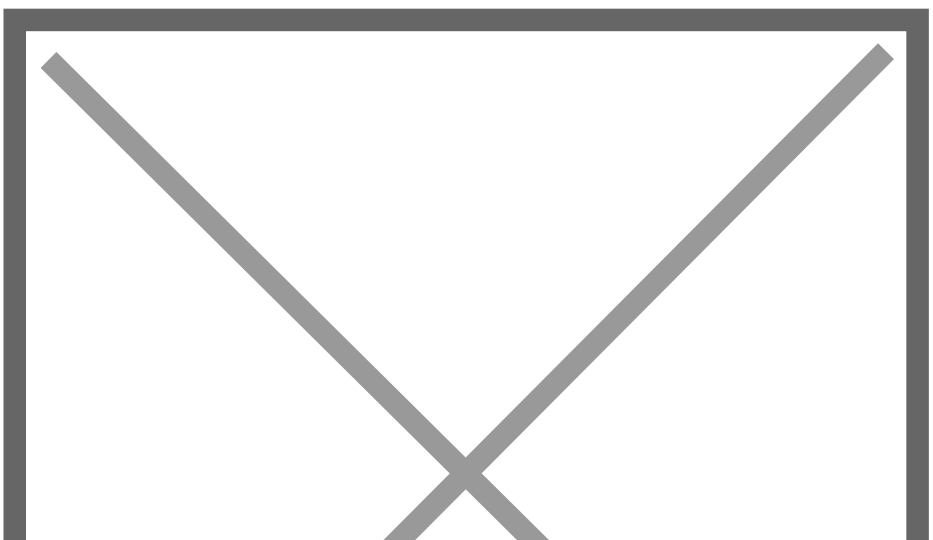
Instalamos o AstraCalls no seu servidor com tudo configurado e também fazemos projetos sob medida com a nossa API, para empresas que querem automatizar atendimento, cobrança ou agendamento pelo WhatsApp.

Se você atende com uma equipe, o **AstraChat**, a nossa versão do Chatwoot, junta multiatendimento, Kanban, agentes de IA e ligações pelo WhatsApp numa tela só. Ele vem na Comunidade Astra Online, com a primeira instalação feita pelo especialista em Chatwoot Raphael Popiolek. Chame no WhatsApp e conte o que você quer automatizar.

Falar com um especialistaConhecer o AstraChat

Mais artigos de API do WhatsApp e o Guia completo do Chatwoot, com todos os assuntos organizados.

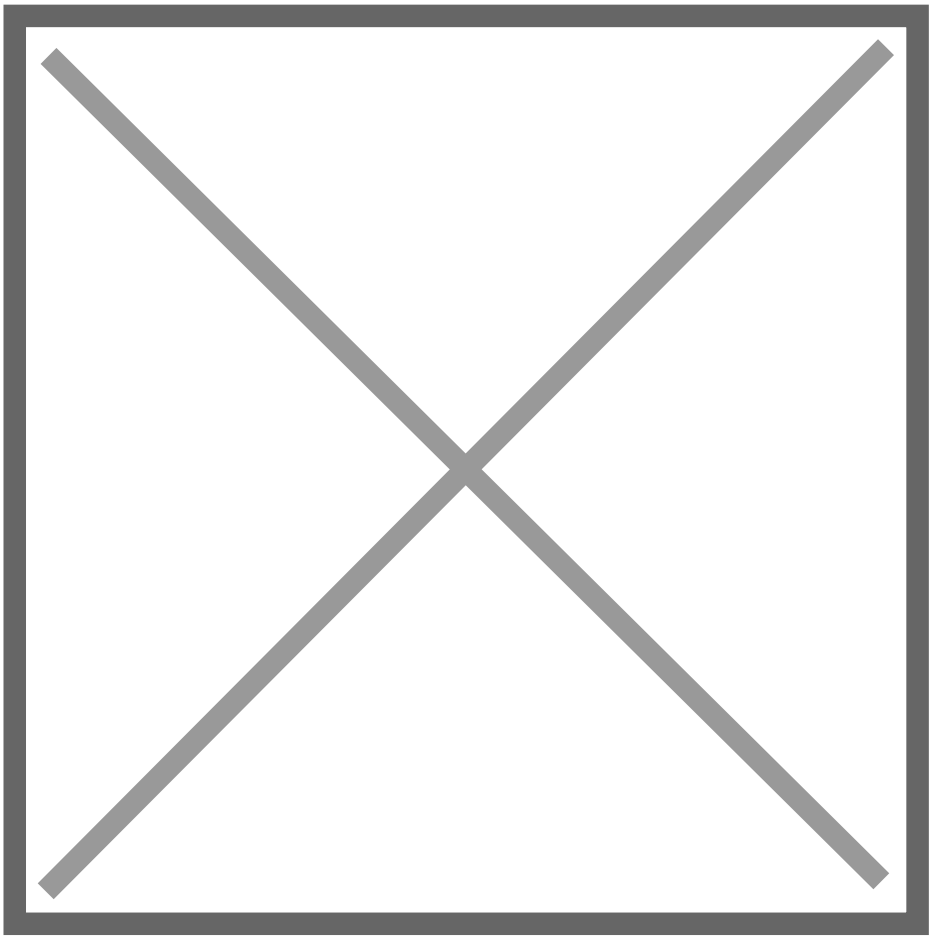
Leia também



02 de novembro de 2026

Agente de IA do WhatsApp Business: é grátis, como ligar e limites

Agente de IA do WhatsApp Business (Meta Business Agent): o que faz, se é gratuito, como ativar e desativar e quando vale ter um agente próprio.



02 de novembro de 2026

Como criar um agente de IA para WhatsApp: método e 3 caminhos

Como criar um agente de IA para WhatsApp: função, tom, base de conhecimento, regra de passar para humano e testes. Sem programar, com n8n ou com código.



28 de outubro de 2026

Disparo de ligações WhatsApp com n8n: áudio e vídeo gravados

Disparo de ligações WhatsApp com n8n e AstraCalls: o sistema liga, toca seu áudio ou vídeo gravado e marca quem atendeu. Veja o fluxo e os cuidados.

© 2026 Astra Online · AstraChat · Blog · Canal no YouTube

Tutorial: AstraCalls no n8n com fluxos para botões, listas e outros recursos

Link: https://github.com/AstraOnlineWeb/AstraCalls/blob/main/guias/n8n.md#-10-fluxos-de-exemplo-prontos

Neste tutorial você conecta o AstraCalls ao **n8n** e monta um robô que responde mensagens de WhatsApp. Leva uns 10 minutos e não precisa programar.

□□ Node: `n8n-nodes-astracalls`. A documentação completa, com as 55 operações e os 15 eventos, está na página do npm.

Configurações iniciais: N8N e WhatsApp: node g... | Portal de Conhecimento

---

O que você precisa

- Um AstraCalls rodando, com **pelo menos uma sessão conectada** (número pareado pelo QR Code no painel).
- A **API Key** do servidor, que é o valor de `WACALLS_API_KEY`.
- Um n8n onde você seja o dono (owner) da instância, porque só o owner instala community nodes.

Passo 1: instalar o node

1. No n8n, abra **Configurações** (⚙) → **Community Nodes** → **Install a community node**.
2. Digite `n8n-nodes-astracalls`, marque a ciência dos riscos e clique em **Install**.
3. Ao adicionar um node, procure **AstraCalls**. Aparecem dois:
   - **AstraCalls**: faz ações, como enviar mensagem, ligar ou criar grupo.

- **AstraCalls Trigger**: inicia o fluxo quando algo acontece, como uma mensagem recebida.

> **n8n em Docker:** se preferir, adicione `N8N_COMMUNITY_PACKAGES=n8n-nodes-astracalls` nas variáveis do serviço e reinicie.

Passo 2: criar a credencial

1. **Credentials → Add credential → AstraCalls API**.
2. Preencha os dois campos:

Campo	Exemplo
URL do Servidor	`https://calls.seudominio.com`, o endereço do painel, sem `/api` no final
API Key	o valor de `WACALLS_API_KEY`

3. Clique em **Save**. Se aparecer □ , está conectado.
   - **401**: a API Key está errada.
   - **Não conecta**: a URL está errada ou o servidor está fora do ar.

Passo 3: enviar a primeira mensagem

1. Crie um fluxo com um **Manual Trigger** e adicione o node **AstraCalls** depois dele.
2. Configure o node:
   - **Recurso** `Mensagem` · **Operação** `Enviar Texto`
   - **Sessão**: escolha o número na lista
   - **Para**: `5511999999999`. Use DDI + DDD + número; com ou sem o 9º dígito funciona.
   - **Texto**: `Olá! Primeira mensagem pelo n8n □□`
3. Clique em **Test step**. A mensagem chega no WhatsApp, e a resposta do node traz o `id` dela.

Passo 4: responder mensagens automaticamente

> ⚡ **Atalho:** baixe `n8n/responder-mensagem.json` e importe no n8n pelo menu ⋯ → **Import from File**. Depois escolha a credencial e a sessão nos dois nodes.

Para montar na mão:

1. Adicione o **AstraCalls Trigger**:
   - **Sessão**: o número que vai receber
   - **Eventos**: `Mensagem (Recebida ou Enviada)`
   - Deixe **Ignorar Mensagens Enviadas por Mim** ligado, para o fluxo não responder a si mesmo em loop.
2. Depois do gatilho, adicione o node **AstraCalls** (`Mensagem → Enviar Texto`):
   - **Sessão**: `{{ $json.sessao }}`
   - **Para**: `{{ $json.responderPara }}`. Esse campo já traz o contato certo, ou o grupo quando a mensagem veio de um.
   - **Texto**: `Oi {{ $json.nome }}! Recebi: "{{ $json.texto }}"`
3. **Ative** o fluxo e mande uma mensagem para o número para testar.

Para responder conforme o conteúdo, coloque um **Switch** entre os dois nodes, olhando o campo `texto`. Por exemplo: se contém "preço", envia a tabela.

Passo 5 (opcional): agente de IA

Para que uma IA atenda pelo WhatsApp:

1. Use o fluxo do passo 4, mas troque o envio por um node **AI Agent**, com o modelo de sua preferência e uma memória.
2. Ligue o **AstraCalls** como **ferramenta (Tool)** do agente, por exemplo com a operação "Enviar Texto". O agente decide sozinho quando enviar mensagens, consultar contatos etc.

> Outra opção é a ferramenta **MCP Client** do n8n apontando para o `/mcp` do AstraCalls, que dá ao agente acesso a todas as ferramentas, sem instalar o node. Veja a seção "MCP para agentes de IA" no

> README.

□□ 10 fluxos de exemplo prontos

Baixe o `.json`, importe no n8n pelo menu **⋯ → Import from File**, escolha a credencial e a **Sessão** em cada node AstraCalls e troque os números de exemplo (`5511999999999`). Cada fluxo tem uma nota amarela explicando como funciona.

#	Fluxo	O que faz
01	Resposta fora do horário	Responde automaticamente a quem escreve fora do expediente
02	Menu com botões	Envia um menu com botões e segue um caminho para cada opção (Switch)
03	Lista de serviços e agendamento	Envia uma lista interativa, confirma a opção escolhida e reage com □
04	Atendente de IA com ferramentas	AI Agent com memória que usa o AstraCalls como ferramenta (requer credencial OpenAI)
05	Pesquisa de satisfação (enquete)	Envia uma enquete para os clientes, agradece cada voto e avisa o gestor se a nota for ruim
06	Boas-vindas no grupo	Dá boas-vindas a quem entra no grupo, mencionando a pessoa
07	Disparo de ligações com áudio	Liga para uma lista tocando um áudio gravado e acompanha o status
08	Validar lista antes da campanha	Confere quais números têm WhatsApp e só envia para eles
09	Lembrete diário com toque	Todo dia, no horário marcado, envia um lembrete e faz o telefone tocar
10	Alerta de desconexão e restrição	Avisa quando a sessão cai ou recebe restrição do WhatsApp

> Os fluxos com **AstraCalls Trigger** (01 a 06 e 10) trocam o webhook da sessão quando ativados. Ative **um por sessão**, ou junte os caminhos num só fluxo (veja o aviso abaixo).

⚠ Antes de usar em produção

- **Cada sessão tem um único webhook.** Ao ativar um fluxo com o AstraCalls Trigger, o node coloca o webhook do n8n no lugar do que estava configurado. Ao desativar, restaura o anterior. **Use um só gatilho por sessão** e separe os caminhos com o **Switch**. A integração nativa com o Chatwoot não é afetada.
- **Ligações e disparos em massa** só para quem autorizou o contato. Envio não solicitado bloqueia o número e viola a LGPD e as regras da Anatel.

Problemas comuns

Sintoma	Solução
A sessão não aparece na lista	Confira a credencial e se a sessão está conectada no painel
A sessão aparece como *aguardando QR Code*	Abra o painel do AstraCalls e pareie o número
O gatilho não dispara	O fluxo precisa estar **ativo**, e o n8n precisa ser acessível pelo servidor do AstraCalls (URL pública ou mesma rede)
Outro fluxo parou de receber eventos	Dois gatilhos na mesma sessão: o último ativado vence (veja o aviso acima)

Arquivos do projeto: