MAPA DA AULA — AUTOMATIZAR 100% DO WHATSAPP COM CLAUDE CODE

> *Aula F2A · 2026-05-20 · público: alunos pagantes (99% leigos em código) Formato: ao vivo Zoom/Meet + replay + PDF playbook auto-estudo Duração estimada: 4-5h ao vivo (com 2 intervalos e Q&A)*

META DA AULA

No fim das 3-4h, o aluno tem:

- WhatsApp dele respondendo lead 24/7 com IA real
- Decisão consciente: oficial vs não-oficial (e os porquês)
- Workflow no n8n criado VIA LINGUAGEM NATURAL pelo Claude Code
- Custos transparentes (sem maquiar)
- Caminho de upgrade quando crescer (não-oficial → oficial)

TESE PEDAGÓGICA (PORQUE ESSA ESTRUTURA)

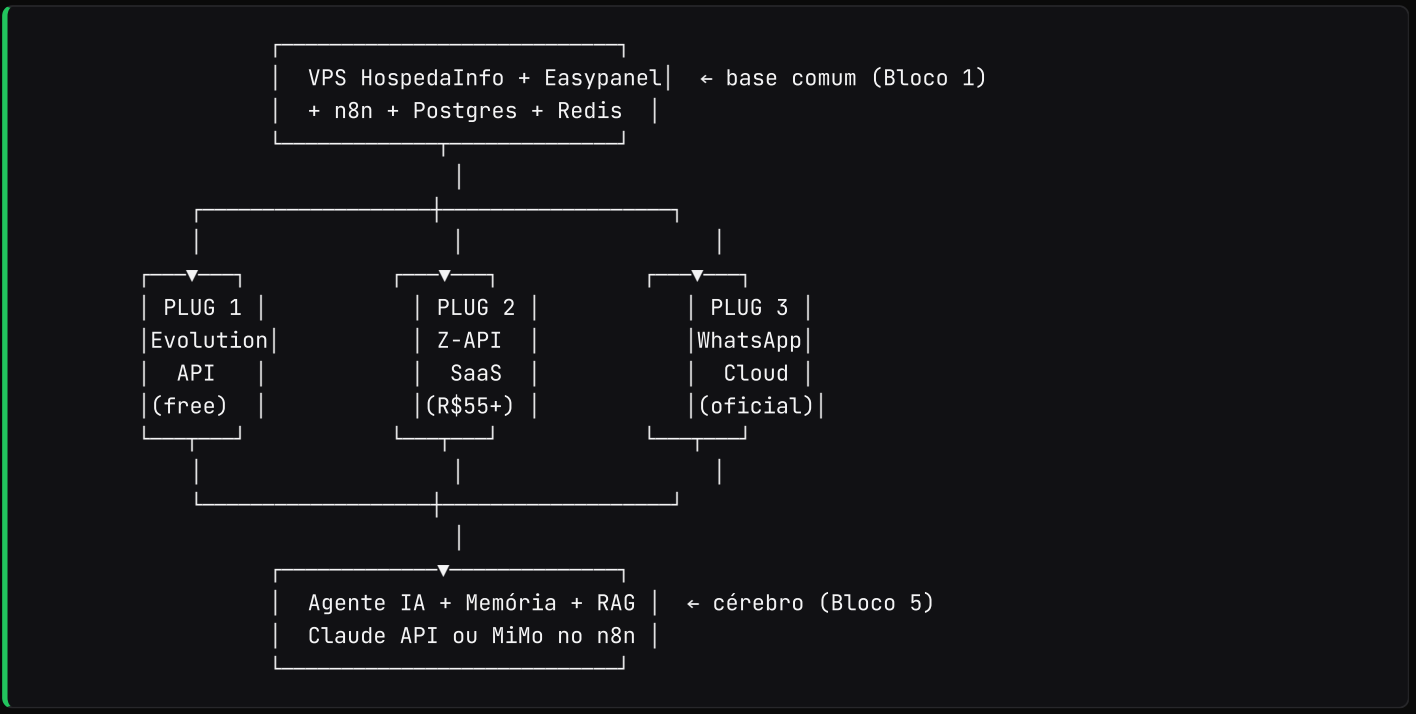
A aula tem 2 modos complementares — aluno aprende AMBOS:

```
MODO A — BOT 24/7 REATIVO (cliente inicia)
─────────────────────────────────────────
Cliente manda msg → bot responde sozinho com IA
Você não precisa estar na frente da tela
Stack: VPS + Easypanel + n8n + Plug WhatsApp + LLM
Use pra: SDR de plantão, atendimento noturno, primeira triagem

MODO B — OPERAÇÃO DIRETA (você inicia)
──────────────────────────────────────
Você fala em PT-BR no Claude Code:
  "dispara: oi galera, hoje tem live 20h"
  "kanban"
  "forecast"
  "qualifica 5511999... nome=Maria"
Agente Claude Code roda scripts Python + SQLite local
Stack: Python stdlib + SQLite + agente .md em .claude/agents/
Use pra: campanhas pra base, pipeline CRM, forecast,
         qualificação BANT, auditoria

AMBOS USAM O MESMO NÚMERO/INSTÂNCIA WhatsApp.
AMBOS PASSAM PELOS PLUGS (Evolution/Z-API/Zappfy/Cloud).
AMBOS PERSISTEM NO MESMO POSTGRES/SQLite.
```

A superestrutura técnica:



O aluno aprende a superestrutura UMA VEZ. Depois troca de plug em 10 minutos.

DECISION TREE — QUAL PLUG ESCOLHER

Ver `_diagramas/01-decision-tree.md` (ASCII).

Resumo:

VOCÊ TEM...	VAI PRA...	CUSTO/MÊS	SETUP	RISCO BAN
Quero zero custo extra, aceito gerenciar	**PLUG 1** Evolution	R$ 0 (só VPS)	30 min	Médio-alto
Quero plug-and-play, pago pra ter paz	**PLUG 2** Z-API	R$ 55-99	5 min	Médio
Tenho CNPJ, paciência, quero oficial	**PLUG 3** Cloud API	grátis (atendim.) + R$ 0,30/msg marketing	1-3 semanas	Zero

Regra de ouro: começou com Plug 1 ou 2 (não-oficial), **não migra o mesmo número** pra Plug 3 sem 7 dias offline. Decida antes.

ESTRUTURA DA AULA (6 BLOCOS)

BLOCO 0 — PRÉ-REQUISITOS UNIVERSAIS (60 MIN)

0.1 — Instalar Claude Code

- Download oficial, login Anthropic, primeiro `hello world`

- Diferença Claude Code (CLI) vs Claude API (token) vs Claude.ai (web)

0.2 — Conta Anthropic + créditos

- Criar conta plataform.claude.com
- Adicionar créditos (US$5 já dá pra fechar aula inteira)
- Pegar API key (`sk-ant-...`)
- Onde guardar com segurança (env var, NUNCA git)

0.3 — Conceito MCP em 5 minutos

- O que é: "plug pro Claude falar com outras coisas"
- Tipos: stdio (local) vs HTTP (remoto)
- Como adicionar: `claude mcp add --transport http <nome> <url>` (sem mexer em JSON)
- Demo ao vivo: adicionar 1 MCP simples (n8n ou Composio)

0.4 — AVISOS CRÍTICOS (3 que destroem aluno se ignorar)

> ⚠ ***AVISO 1: Plano Claude Code ≠ Backend 24/7*** *A assinatura Pro/Max do Claude Code é pra USO INTERATIVO seu. Usar como backend de bot em produção = ban da conta inteira. Pra agente respondendo 24/7: usa* ***Claude API key*** *(paga por token, US$3-15/M tokens). Custo real de 1000 conversas/mês de SDR: ~US$10-30/mês.*

> ⚠ ***AVISO 2: WhatsApp não-oficial pede chip dedicado*** *NUNCA conectar Evolution/Z-API no seu número pessoal. Compra chip novo R$30, ativa em celular Android velho, usa SÓ pra automação. Se banir, perde o chip — não a vida.*

> ⚠ ***AVISO 3: Decisão oficial vs não-oficial é IRREVERSÍVEL*** *Número usado em Evolution/Z-API antes não pode ir pra Cloud API sem 7 dias offline. Escolha de antemão. Mudar de ideia depois = perde 1 semana.*

0.5 — Custos reais por trilha (não vai mentir)

CUSTO MENSAL	PLUG 1 (EVOLUTION)	PLUG 2 (Z-API)	PLUG 3 (CLOUD)
VPS HospedaInfo (cupom KPA20)	R$ 48	R$ 48	R$ 48
Plug WhatsApp	R$ 0	R$ 55-99	R$ 0 atend / R$ 0,30/msg marketing
LLM (Claude API ~1k conv)	R$ 50-150	R$ 50-150	R$ 50-150
TOTAL	**R$ 90-190**	**R$ 145-290**	**R$ 90-340**

Tudo barato comparado a SDR humano (R$ 2.500-5.000/mês).

0.6 — WhatsApp Business App nativo (warm-up antes da automação via API)

> *Antes de mergulhar em VPS/API, mostrar pro aluno o que ele JÁ tem disponível de graça no celular dele. 80% das micro-empresas no Brasil nunca usaram esses recursos — começam usando WhatsApp pessoal pra negócio. Dá pra ganhar muito só ativando essas features ANTES de partir pra automação.*

O que é WhatsApp Business App

- App gratuito da Meta no Google Play / App Store (separado do WhatsApp comum)

- Pode usar mesmo número do pessoal? Não — número dedicado
- Pode usar mesmo chip do Cloud API? Não — escolher de antemão
- Diferença pro WhatsApp pessoal: tem ferramentas comerciais nativas

Features nativas que viram automação leve sem código:

1. **Mensagem de saudação automática**
   - Configurável em horários (ex: fora do expediente)
   - Resposta: "Oi! Recebi sua mensagem, respondo em até 1h útil"
   - Útil pra: setar expectativa, ganhar tempo
2. **Mensagem de ausência**
   - Dispara quando lead manda msg em horário não-comercial
   - Texto + link Calendly = lead agenda sozinho enquanto você dorme
3. **Respostas rápidas**
   - Crie até 50 respostas com `/atalho`
   - Ex: `/preco`, `/garantia`, `/comoFunciona`, `/endereco`
   - Você digita `/preco` no chat → expande texto completo
   - **Hack:** atalhos pra TODAS as objeções comuns do seu nicho
4. **Etiquetas (labels)**
   - Organize conversas por status: Novo Lead / Em Negociação / Cliente / Pós-Venda
   - Funciona como mini-CRM dentro do app
   - Limite: 20 etiquetas
5. **Catálogo de produtos**
   - Vitrine dentro do WhatsApp com fotos + descrição + preço
   - Lead pede catálogo → você manda link wa.me com produto pré-selecionado
   - Bom pra: e-commerce, infoprodutor com várias ofertas
6. **Link wa.me + QR code**
   - `wa.me/5511999999999?text=Quero saber sobre o F2A`
   - Lead clica → abre WhatsApp já com mensagem digitada
   - QR code da empresa → cola em LP, criativo, cartão
7. **Botão Web do WhatsApp Business** (limitado)
   - Versão web pra responder do desktop
   - 4 dispositivos conectados
   - **NÃO é WhatsApp Web normal** — UI separada com etiquetas, respostas rápidas

Quando aluno deve PARAR no Business App e NÃO partir pra automação completa

```
PARA NO BUSINESS APP SE:
- < 30 mensagens recebidas/dia
- Atendimento 100% manual já é fluido
- Sem equipe (só você responde)
- Sem orçamento R$ 90-340/mês pra infra

PARTE PRA AUTOMAÇÃO SE:
- > 50 mensagens/dia (não consegue mais responder tudo)
- Quer disparar Campanha pra base existente
- Quer pipeline CRM (forecast, qualificação BANT)
- Quer responder lead às 3h da manhã
- Tem orçamento + interesse técnico
```

Pedagógico: mostrar Business App PRIMEIRO frustra menos. Aluno que partia direto pra "monto VPS + n8n" às vezes nem precisava — Business App resolvia. Reduz expectativa ruim e dá vitória rápida pra alunos sem tempo/orçamento.

Demo ao vivo (5 min):

- Instalar Business App
- Configurar saudação automática
- Criar 3 respostas rápidas (`/preco` , `/horario` , `/agendar`)
- Criar 4 etiquetas
- Gerar QR code da empresa
- Compartilhar link wa.me

Aluno sai com isso JÁ funcionando, mesmo que decida não fazer o resto da aula.

BLOCO 1 — SUPERESTRUTURA: VPS + EASYPANEL + N8N + POSTGRES (45 MIN)

> *Esse bloco é comum a TODOS os plugs. Aluno faz UMA VEZ.*

1.1 — Comprar VPS HospedaInfo

- `cart.hospedainfo.com/aff/577` (link afiliado)
- No menu: **Hospedagem → Servidores VPS**
- Plano R$ 59,90/mês · aplicar cupom `KPA20` no checkout → R$ 47,92/mês (20% off · 3 meses)
- **CRÍTICO:** Easypanel já vem **pré-instalado**, zero configuração inicial
- Após 3 meses do cupom volta pra R$ 59,90/mês — avisar o aluno

> ***CADA passo tem 2 MODOS*** *— aluno escolhe:*
>
> - ***Modo Básico (98% · app):*** *tudo conversando com o Claude Code no aplicativo desktop. Sem terminal, sem npm, sem bash. O Claude conecta no Easypanel pela API e cria os serviços.*
> - ***Modo Avançado (CLI):*** *pra quem manja terminal e prefere rodar os comandos na mão.*

1.2 — Acessar Easypanel + pegar a API key

- Abrir `http://SEU_IP:3000` (Easypanel já vem instalado) → criar senha admin
- **Settings → API Tokens → Generate Token** → copiar o token

- Anotar URL do painel + token (vamos usar pra conectar o Claude)
- **CRÍTICO:** API key dá controle total da VPS — nunca postar print, nunca commitar

1.3 — Conectar Claude Code ao Easypanel (via API key, NÃO SSH)

- MCP oficial: `easypanel-mcp` (github.com/dray-supadev/easypanel-mcp — 40 tools)
- **Modo Básico (app):** Settings → Connectors → Add → Easypanel → cola URL + API token → Conectar
- **Modo Avançado (CLI):**
  - `claude mcp add --transport stdio easypanel -- npx -y easypanel-mcp`
  - `claude mcp add-env easypanel EASYPANEL_URL=http://SEU_IP:3000 EASYPANEL_TOKEN=tua_key`
- Teste (ambos): *"Claude, lista os serviços rodando na minha VPS e o uso de CPU/RAM"*

1.4 — Subir n8n + Postgres + Redis

- **Modo Básico (app):** 1 prompt só → *"Claude, no meu Easypanel cria n8n + Postgres (banco 'whatsapp') + Redis, conecta o banco no n8n e me devolve URL/login/connection string."* O Claude cria os 3 serviços via API e entrega os acessos. **Zero clique no painel.**
- **Modo Avançado (painel):** Easypanel → Create Service → From Template → n8n / Postgres / Redis (1-click cada) → conectar env vars no n8n

1.5 — Conectar Claude Code ao n8n via MCP

- No n8n: Settings → API → criar token
- **Modo Básico (app):** Settings → Connectors → Add → n8n → cola URL + token
- **Modo Avançado (CLI):** `claude mcp add --transport http n8n-vps https://SEU_DOMINIO/api/mcp --header "Authorization: Bearer SEU_TOKEN"`
- **Demo MÁGICA (WOW da aula):** aluno fala *"Claude, cria um workflow de teste que retorna hello world em webhook"* → Claude monta o JSON → sobe no n8n direto. **Sem digitar 1 linha de código.**

Ver diagrama: `_diagramas/02-superestrutura-comum.md`

BLOCO 2 — PLUG 1: EVOLUTION API (NÃO-OFICIAL, FREE) (45 MIN)

2.1 — Subir Evolution API no Easypanel

- Easypanel → "From Template" → buscar "Evolution API"
- **ATENÇÃO:** escolher repositório `EvolutionAPI/evolution-api` v2.3.x — NÃO o fork `evolution-foundation` (esse é pago/licenciado)
- Preencher: domínio, API key, redirecionamento webhook (n8n URL)
- Subir → ~1 min

2.2 — Criar instância WhatsApp

- Acessar `https://evolution.seu-dominio/manager`
- Criar instância "atendimento"
- Pegar QR code → escanear no chip dedicado (Android velho)
- Conectado ✓

2.3 — Configurar webhook → n8n

- Na instância Evolution, configurar webhook URL apontando pro n8n
- Eventos a escutar: `MESSAGES_UPSERT` (mensagens recebidas)

- Salvar

2.4 — Workflow n8n: Evolution → Claude → resposta

- No Claude Code: *"Cria workflow que recebe webhook Evolution, filtra grupos e fromMe, chama Claude API, manda resposta de volta"*
- Claude monta workflow com:
  - Webhook trigger (recebe POST Evolution)
  - IF (filtra `@g.us` e `fromMe`)
  - HTTP Request → Claude API (api.anthropic.com/v1/messages)
  - HTTP Request → Evolution API (POST /message/sendText)
- Testar end-to-end: mandar mensagem do celular → bot responde

2.5 — Boas práticas anti-ban Evolution

- Chip NOVO dedicado (R$30)
- Aquecimento 7-14 dias antes (mensagens orgânicas pra contatos reais)
- Delay aleatório 8-30s entre respostas
- Nunca templates 100% idênticos (varia frases)
- Não comprar lista fria
- Limite respostas/dia (~100 mensagens novas é seguro)

Ver diagrama: `_diagramas/03-plug-evolution.md`

BLOCO 3 — PLUG 2: Z-API (NÃO-OFICIAL, SAAS) (30 MIN)

> *Mesmo n8n da VPS, troca só o "plug" do WhatsApp. Setup mais rápido, paga R$ 55-99/mês.*

3.1 — Criar conta Z-API

- z-api.io (3 dias grátis sem cartão)
- "Nova instância" → ganha **Instance ID** + **Token**

3.2 — Conectar WhatsApp

- Abrir instância no painel → QR code → escanear chip dedicado
- Conectado ✓

3.3 — Configurar webhook → n8n

- Painel Z-API → instância → webhooks
- Apontar URL do n8n (mesma estrutura do Plug 1)

3.4 — Workflow n8n: Z-API → Claude → resposta

- No Claude Code: *"Adapta o workflow do Evolution pra usar Z-API"*
- Claude troca:
  - Endpoint webhook (Z-API tem JSON diferente)
  - Endpoint envio: `POST api.z-api.io/instances/{ID}/token/{TOKEN}/send-text`

- **Gotcha crítico:** webhook Z-API timeout em 5s. Workflow deve responder 200 imediato + processar em background. Claude resolve isso com Respond to Webhook + Wait.
- Testar

3.5 — Z-API vs Evolution: trade-offs

- Z-API: paga R$ 55-99/mês, mas time BR mitiga ban, atualiza protocolo, suporte PT-BR
- Evolution: free, mas você cuida de tudo
- Recomendação pro F2A: começa Z-API (paga paz), migra pra Evolution quando dominar

Ver diagrama: `_diagramas/04-plug-zapi.md`

BLOCO 4 — PLUG 3: WHATSAPP CLOUD API (OFICIAL) + COMPOSIO/RUBE (60 MIN)

> *Setup mais lento (1-3 semanas verificação CNPJ), mas SEM risco de ban. Atendimento reativo grátis.*

4.1 — Setup Meta Business

- business.facebook.com → criar conta
- Adicionar empresa: CNPJ, comprovante endereço, contrato social
- **Verificação Meta:** 2-10 dias úteis (sem isso, fica limitado 250 conv/24h)

4.2 — Configurar WhatsApp Business Platform

- developers.facebook.com → criar app → adicionar produto "WhatsApp"
- Cadastrar **número novo** (NÃO usar número que já passou por Evolution/Z-API)
- Pegar:
  - `WHATSAPP_BUSINESS_ACCOUNT_ID`
  - `PHONE_NUMBER_ID`
  - `ACCESS_TOKEN` (gerar token permanente — sistema user)

4.3 — Caminho A: via Composio/Rube no Claude Code

- Conta composio.dev → pegar API key
- `claude mcp add --transport http rube https://rube.composio.dev/mcp --header "X-API-Key: SUA_KEY"`
- Dentro do Claude: `@rube` → autenticar WhatsApp Cloud
- 17 tools disponíveis: `WHATSAPP_SEND_MESSAGE`, `WHATSAPP_SEND_TEMPLATE_MESSAGE`, `WHATSAPP_SEND_MEDIA`, etc
- **Limitação:** funciona enquanto Claude Code está aberto. **Não é 24/7.**
- **Uso real:** ferramenta de OPERAÇÃO (você dispara mensagens manualmente via Claude). Pra 24/7 reativo, vai pra 4.4.

4.4 — Caminho B: WhatsApp Cloud no n8n (24/7 real)

- No n8n: nó "WhatsApp Trigger" (webhook do Meta)
- Configurar webhook URL + verify token no painel Meta
- Workflow: webhook → IF (mensagem in vs status update) → Claude API → HTTP Request → Cloud API
- No Claude Code: *"Cria workflow n8n que escuta webhook WhatsApp Cloud, responde com Claude API"*

4.5 — Atendimento REATIVO (cliente inicia)

- Cliente manda primeira msg → abre janela de 24h
- Você responde livre dentro da janela (qualquer texto/mídia/botão)
- **Janela é GRÁTIS** — service conversations sem teto desde nov/2024

4.6 — Disparo ATIVO (empresa inicia) — Templates HSM PASSO A PASSO

> *O ponto crítico que você quis detalhar. Vai do zero ao disparo.*

4.6.1 Conceitos:

- HSM = "Highly Structured Message"
- Cobra por mensagem (não por conversa)
- Aprovação Meta: 24-72h
- 3 categorias:
  - **UTILITY** (confirmação pedido, lembrete) — mais barato
  - **AUTHENTICATION** (OTP, código) — preço médio
  - **MARKETING** (promo, lançamento) — mais caro, scrutínio pesado

4.6.2 Criar template (passo a passo Meta Business Manager)

- WhatsApp Manager → Account tools → Message Templates → "Create Template"
- Escolher: Categoria + Idioma (pt_BR) + Nome (snake_case, sem espaços)
- Body com placeholders `{{1}}`, `{{2}}` (variáveis)
- Adicionar Header (opcional): texto, imagem, vídeo, doc
- Footer (opcional): texto curto
- Buttons (opcional): Quick Reply ou Call-to-Action (URL/telefone)
- Submeter → status "In Review"

4.6.3 Aguardar aprovação

- Status: PENDING → APPROVED ou REJECTED
- Tempo médio: 1-3h pra utility, 24-72h pra marketing
- Se REJECTED: ver motivo, corrigir, ressubmeter

4.6.4 Disparar template via API

- POST `graph.facebook.com/v21.0/{PHONE_NUMBER_ID}/messages`
- Body:

```
{
  "messaging_product": "whatsapp",
  "to": "5511999999999",
  "type": "template",
  "template": {
    "name": "lembrete_pagamento",
    "language": { "code": "pt_BR" },
    "components": [
      { "type": "body", "parameters": [
        { "type": "text", "text": "João" },
        { "type": "text", "text": "29/05" }
      ]}
    ]
  }
}
```

- No n8n: nó HTTP Request fazendo esse POST
- No Claude Code: *"Cria workflow de disparo de template lembrete_pagamento pra lista de leads do Postgres"*

4.6.5 Tiers de mensagem (limite diário)

- Tier 1: 250 conversas únicas/24h
- Tier 2: 1k conv/24h
- Tier 3: 10k conv/24h
- Tier 4: 100k conv/24h
- Tier 5: ilimitado
- Subida automática conforme qualidade (rating do número alto + volume)
- **Cuidado:** rating baixo = downgrade pra Tier 1 e suspensão temporária

4.7 — Custos reais Cloud API 2026 (Brasil)

- Service conversations (cliente inicia, 24h): **GRÁTIS**
- Marketing message: ~R$ 0,30-0,40 por msg
- Utility: ~R$ 0,03-0,05 por msg
- Authentication: ~R$ 0,02 por msg
- **Exemplo:** 1.000 leads, disparar template marketing = R$ 300-400 + responder via API grátis (janela 24h aberta após resposta)

Ver diagrama: `_diagramas/05-plug-cloud-api.md`

BLOCO 5 — AGENTE IA + MEMÓRIA + RAG (45 MIN)

> *Comum a todos os 3 plugs. Aluno só faz uma vez.*

5.1 — Configurar nó LLM no n8n

- Opções:
  - **Claude API** (recomendado pra produção, mais caro mas melhor) — modelo `claude-opus-4-7` ou `claude-haiku-4-5`
  - **MiMo v2.5-pro** via OpenAI-compat (mais barato — US$1/3 por Mtok, contexto 1M)

- **OpenAI GPT-5** (similar Claude, ecossistema gigante)

- Configurar credencial no n8n: nó "OpenAI Chat Model" (base URL custom pra MiMo)
- Demo: enviar 1 prompt, ver resposta

5.2 — System prompt afinado pro caso

- Pra F2A: "Você é Lailla, SDR da Pedro Zampieri. Atende lead novo, qualifica (orçamento, urgência), agenda call ou direciona pro VSL..."
- Princípios: tom Tropa do Choque, anti-guru, direto, sem markdown pesado
- Limite: máximo 3 perguntas seguidas, sempre validar antes de fechar

5.3 — Memória curto-prazo (Redis no Easypanel)

- Easypanel → template Redis (1 clique)
- No n8n: nó "Redis" pra armazenar buffer de mensagens consecutivas (anti-fragmentação)
- Padrão "Smart Redis Buffer" (já usado no Bot SDR Polypus do Pedro)

5.4 — Memória longo-prazo (Postgres tabela chat_history)

- Tabela já criada no Postgres (Bloco 1)
- No n8n: nó "Postgres" lendo/escrevendo histórico
- Antes de cada resposta: puxa últimas 20 mensagens da conversa

5.5 — RAG 2 camadas (decisão arquitetural do Pedro no Bot SDR)

```
┌──────────────────────────────────────┐
│ Camada 1 — DETERMINÍSTICA (lookup exato) │
│ Tabela Postgres: clientes, produtos,     │
│ preços, números autorizados, status      │
│ Tool no n8n: SQL Query direta            │
│ NUNCA vetorizar (risco de pegar dado     │
│ do cliente errado)                       │
└──────────────────────────────────────┘

┌──────────────────────────────────────┐
│ Camada 2 — SEMÂNTICA (RAG vetor)         │
│ PGVector no Postgres (extensão)          │
│ Conteúdo: playbooks, FAQs, objeções,     │
│ processos da empresa                     │
│ Embedding 1x na ingestão, busca top-3    │
│ Tool no n8n: Vector Store Tool           │
└──────────────────────────────────────┘
```

- Aluno aprende a separar os 2 tipos de conhecimento
- Resultado: agente não alucina dado crítico (preço, ID cliente) E ainda tem contexto rico (objeções, casos)

5.6 — Filtros e proteções

- Filtra grupos (`@g.us` no remoteJid)
- Filtra `fromMe` (não responde a si mesmo)
- Whitelist de números (opcional pra ambiente de teste)
- Rate limit: max N respostas/minuto pra evitar spam
- Fallback: se Claude API falhar, manda "1 momento, vou chamar atendente" e cria ticket

Ver diagrama: `_diagramas/06-agente-ia-memoria.md`

BLOCO 6 — MODO OPERACIONAL: COMANDAR O WHATSAPP DIRETO PELO CLAUDE CODE (60 MIN)

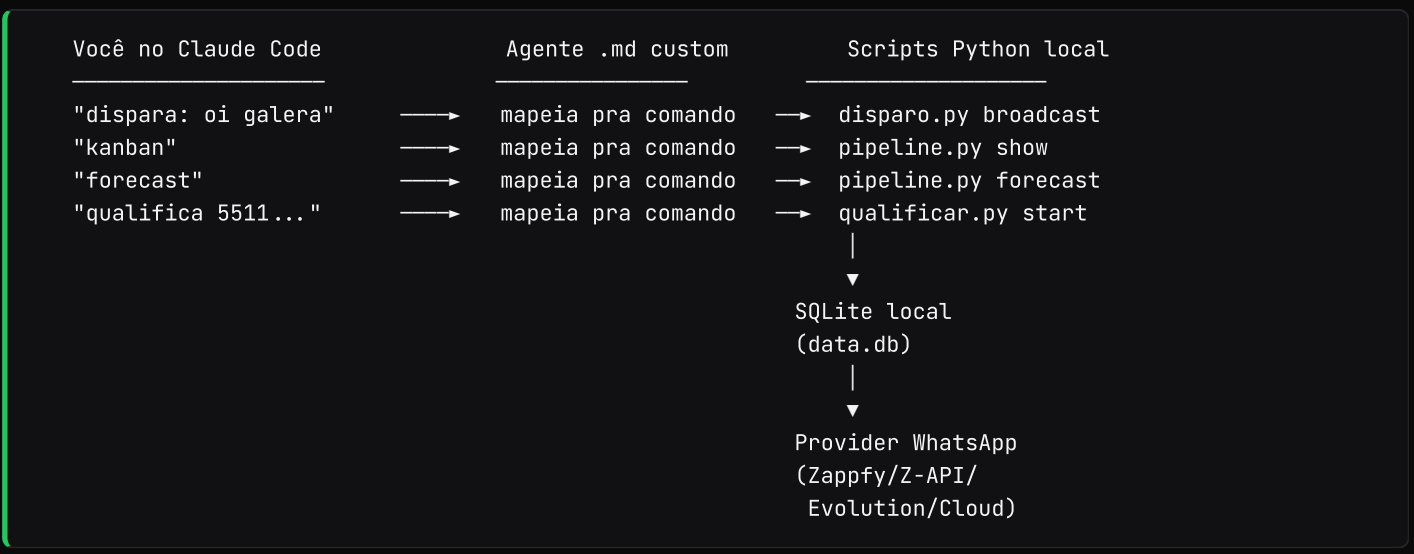
> *Esse bloco mostra como Pedro vive isso. Não é o bot 24/7 dos Blocos 1-5. É o Pedro abrindo o Claude Code de manhã e falando "kanban" → ver pipeline; "dispara: ..." → fazer campanha; "audita" → ver leads abandonados. Tudo em linguagem natural.*

Referência arquitetural: o padrão de "agente Claude Code + scripts Python stdlib + SQLite local" é maduro no mercado (ex: ASV Digital / Bravy expõem isso comercialmente pra clientes deles). Pedro entrega versão ORIGINAL Polypus do zero — inspirada nos mesmos padrões arquiteturais, código novo, sem redistribuição de pacote de terceiros.

Nome do pacote Polypus: `POW — Polypus WhatsApp Toolkit` (placeholder — Pedro decide).

6.1 — A arquitetura em 1 imagem

Ver `_diagramas/07-modo-operacional-arquitetura.md`.



Stack ULTRA leve:

- Python 3.8+ stdlib only (zero `pip install` — `urllib`, `sqlite3`, `csv`, `json`, `re`)
- SQLite local (`data.db`) com 7 tabelas: leads, touches, inbox, conversations, followup_jobs, pipeline_events, campaigns
- 1 arquivo agente `.md` que mora em `.claude/agents/` — instrui Claude Code a usar Bash pra rodar os Python scripts
- ZERO MCP customizado, ZERO Docker pra esse modo. Roda no laptop OU na VPS (instala Python).

6.2 — 21 capacidades em 4 camadas (panorama)

```
╔═══════════════════════════════════════════════════════════════════╗
║                                                                   ║
║  CAMADA 1 — DISPARO                                               ║
║  ──────────────────                                               ║
║   1  Listar grupos (admin vs membro)                              ║
║   2  Extrair leads dos grupos (E.164 + SHA-256 + dedup)           ║
║   3  Segmentar (por DDD, nº grupos, admin, blacklist)             ║
║   4  Broadcast em grupos (mention + jitter + retry)               ║
║   5  X1 (1:1) personalizado com {{placeholders}}                  ║
║   6  Agendar disparo pra horário ouro                             ║
║   7  A/B test com z-test 95%                                      ║
║                                                                   ║
║  CAMADA 2 — INBOX                                                 ║
║  ────────────────                                                 ║
║   8  Pull/watch de mensagens recebidas                            ║
║   9  Webhook HTTP em :8765 pra tempo real                         ║
║  10  Classificador 8 intents (regex+heurística, ZERO ML)          ║
║  11  Auto-triage: opt_out→blacklist, agendamento→Calendly, etc    ║
║                                                                   ║
║  CAMADA 3 — CRM                                                   ║
║  ──────────                                                       ║
║  12  SQLite local 7 tabelas                                       ║
║  13  Qualificação BANT/SPIN em 3 perguntas via x1                 ║
║  14  Cadências D+0..D+30 (4 built-in + custom CSV)                ║
║  15  Pipeline kanban 7 estágios                                   ║
║  16  Forecast ponderado (probabilidade × ticket)                  ║
║  17  Funil com taxa de conversão                                  ║
║  18  Audit de leads abandonados                                   ║
║                                                                   ║
║  CAMADA 4 — CONTEXTO                                              ║
║  ──────────────────                                               ║
║  19  Enrich CSV/Google Sheets/JSON cruzando por phone             ║
║  20  Fetch HTML → contexto pro lead                               ║
║  21  Relatório executivo (single OU semanal)                      ║
║                                                                   ║
╚═══════════════════════════════════════════════════════════════════╝
```

6.3 — Setup pra aluno (15 min)

1. Baixar pacote `POW-F2A.zip` (versão original Polypus, distribuição liberada pra aluno F2A)
2. `unzip` na pasta de projeto
3. `cp .env.example .env` → editar credenciais do provider escolhido (Zappfy/Z-API/Evolution/Cloud — variável `API_BASE` muda)
4. `python3 health_check.py` → ver semáforo 🟢/🟡/🔴
5. `cp whatsapp-*.md .claude/agents/` → instala agente
6. `/exit` + reabre Claude Code
7. No prompt: `> lista grupos` → agente roda Python → CSV gerado
8. Pronto pra operar

6.4 — Fluxo de dia comercial real (storytelling pra aula)

Ver `_diagramas/08-fluxo-campanha-via-claude-code.md`.

```
Pedro, 9h da manhã, abre Claude Code:

> health
🟢 API: 200 OK
🟢 Grupos visíveis: 18

> pull inbox && triage
✅ 12 novas · 3 duplicadas
triage: 3 promovido_sql · 2 blacklist · 1 agendamento + link enviado

> kanban
▮ NOVO (8) ▮ MQL (12) ▮ SQL (5)  ← os 3 quentes
▮ EM_CONVERSA (7) ▮ PROPOSTA (3) ▮ NEGOCIAÇÃO (2)

> forecast --ticket 2500
PROJEÇÃO PONDERADA  R$ 18.750
GANHO REALIZADO     R$ 25.000

> audit --days 7
[15 leads sem toque há ≥7d, ordenados por fit]

> enroll-csv leads_audit.csv em followup_padrao
✅ 15 enrolled — 75 jobs criados

> fire followup
✅ 12 disparados | 2 skipped (responderam) | 1 falha
```

Tempo total Pedro nessa rotina: ~10 minutos. Toca a base inteira.

6.5 — Fluxo seguro de disparo (4 passos OBRIGATÓRIOS)

```
P1   Captura copy + mídia + valida placeholders
     ↓
P2   Preview (mostra alvos, ETA, blacklist filtrada,
     exemplo renderizado com placeholders)
     ↓
P3   Teste no número pessoal (sem mention/personalização)
     Operador valida no celular → confirma
     ↓
P4   Broadcast/x1 com retry + jitter + log estruturado
     ↓
Relatório executivo .md gerado
Touches persistidos no SQLite
```

Esse fluxo é HARD-CODED no agente — agente RECUSA pular passos.

6.6 — Os 5 playbooks de campanha prontos

O pacote vem com 5 templates de campanha (estrutural — aluno adapta com produto/link/tom):

1. **Aquecimento de turma / lançamento** — D-7 a D-1, 1 msg/dia em horário ouro
2. **Oferta direta (carrinho aberto)** — D0/D1/D5/D6/D7, 5-7 dias com urgência crescente
3. **Reativação de base fria** — disparo único sábado 10h ou domingo 11h
4. **Pesquisa de feedback** — disparo único, dias úteis 14h
5. **Convite pra evento ao vivo** — D-3/D-1/D0 manhã/D0 1h antes/D0 ao vivo
6. **X1 (1:1) personalizado** — recuperação de carrinho, pós-evento, follow-up de orçamento

Cada um com copy estrutural + horário + cadência. Aluno PRECISA adaptar (não dispara cru).

6.7 — Quadro decoradinho de limites operacionais

```
┌──────────────────────┬──────────────────┬──────────────────┐
│                      │   GRUPOS         │   X1 (1:1)       │
├──────────────────────┼──────────────────┼──────────────────┤
│ Volume/min           │   40-50          │   25-30          │
│ Volume/hora          │   200            │   150            │
│ Volume/dia           │   1500-3000      │   800-1500       │
│ Delay mínimo         │   30s            │   45s            │
│ Delay padrão         │   60s ±20%       │   75s ±20%       │
│ Mention everyone     │   sim            │   n/a            │
│ Personaliz. obrigat. │   não            │   sim ({{name}}) │
└──────────────────────┴──────────────────┴──────────────────┘

HORÁRIOS BR
──────────
OURO        ter-qui  9-11h e 14-16h
PRATA       seg+sex  9:30-11h / ter-qui 18-20h
PROIBIDO    01-07h, sábado, domingo, feriado
```

6.8 — Intent Classifier (8 categorias com ação automática)

```
opt_out         → blacklist + responde "ok, parei" + status=perdido
agendamento     → envia CALENDLY_URL + state=aguardando_reuniao
interessado     → status=sql + tag 'quente' + alerta humano 🔥
objecao_preco   → status=em_conversa + tag 'objecao_preco' + alerta 💰
sem_interesse   → status=perdido + tag 'frio'
pergunta        → state=em_conversa + alerta humano ❓
saudacao        → responde com {{first_name}}
ruido           → ignora (kk, ok, emoji só)
```

Tudo regex + heurística — ZERO ML, ZERO custo de modelo. Roda em ~50ms por mensagem.

6.9 — LGPD compliant by design

```
[ ] Token nunca no código — só `.env`
[ ] Lista comprada/raspada = RECUSA do agente (spam ilegal)
[ ] Hash SHA-256 dos números em CSVs auditáveis
[ ] Opt-out automático via intent → blacklist persistente
[ ] Logs estruturados (timestamp|number|kind|status|err)
[ ] Aviso de opt-out obrigatório no rodapé de marketing:
    "Pra parar de receber, responde SAIR"
```

6.10 — Como conecta com Modo A (Blocos 1-5)

```
┌──────────────────────────────────────────────────────┐
│  MESMO NÚMERO. MESMA INSTÂNCIA. 2 MODOS RODANDO EM PARALELO. │
│                                                      │
│  Modo A (n8n bot 24/7)                               │
│    └─► Recebe mensagem entrada                       │
│        └─► Responde com Claude API                   │
│        └─► Grava em Postgres (chat_history)          │
│                                                      │
│  Modo B (Claude Code + Python local)                 │
│    └─► Você dispara campanha em massa                │
│        └─► Python lê SQLite local                    │
│        └─► Envia via mesmo provider                  │
│        └─► Webhook do provider entrega resposta no n8n │
│        └─► n8n grava em Postgres                     │
│        └─► (opcional) sync Postgres → SQLite local   │
│                                                      │
│  Resultado:                                          │
│  Modo A faz triagem 24/7, Modo B opera a base ativa. │
│  Ambos contribuem pro mesmo pipeline.                │
└──────────────────────────────────────────────────────┘
```

6.11 — Adaptação por provider

O pacote original usa Zappfy. Mesmo padrão funciona pra qualquer provider trocando 3 coisas:

PROVIDER	API_BASE	ENDPOINT ENVIO	ENDPOINT INBOX
Zappfy	`https://api.zappfy.io`	`/send/text`	`/chat/messages`
Z-API	`https://api.z-api.io`	`/instances/{id}/token/{tk}/send-text`	webhook in-bound
Evolution	`https://evolution.seu-dominio`	`/message/sendText/{instance}`	webhook MESSAGES_UPSERT
Cloud API	`https://graph.facebook.com/v21.0`	`/{phone_id}/messages`	webhook Meta

Pra aula F2A: oferecer pacote com **4 perfis prontos** (1 por provider) — aluno escolhe e edita só `.env`.

6.12 — Demo ao vivo planejada

```
Pedro mostra ao vivo:
1. Lista grupos da instância dele
2. Extrai 100 leads dedup
3. Mostra preview de campanha "live amanhã"
4. Dispara teste no número pessoal
5. (sem disparar real) mostra como seria o broadcast
6. Roda kanban — mostra pipeline real do Pedro
7. Roda forecast — projeta receita F2A
8. Roda audit — leads abandonados últimos 7 dias

Esse é o WOW visual da aula. Aluno vê tudo funcionando AO VIVO,
linguagem natural, comando em PT-BR.
```

BLOCO 7 — OPERAÇÃO E PRÓXIMOS PASSOS (15 MIN)

6.1 — Monitoramento

- Easypanel: uptime/CPU/RAM (free)
- n8n: histórico de execuções (free)
- Anthropic Console: gasto API real-time
- Uptime Kuma no Easypanel (template 1-click) → alerta WhatsApp se cair

6.2 — Backup

- Easypanel Pro: backup S3 automático (US$ 9.90/mês — vale)
- Manual: dump Postgres + export n8n workflows

6.3 — Escala

- Plug 1 (Evolution) crescendo? → migra pra Plug 2 (Z-API) ou direto Plug 3
- Aluno com 5+ clientes diferentes? → considerar MCP próprio (n8n consome via MCP Client Tool)
- Mais de 10k msg/mês? → upgrade VPS pra KVM 4 (16GB)

6.4 — Quando NÃO automatizar

- Resposta complexa exige humano: criar trigger "transferir pra atendente"
- Vendas high-ticket (PAV, mentoria 1:1): IA QUALIFICA, humano FECHA
- Suporte sensível (reclamação séria): escalation imediata

ANEXOS

ANEXO A — COMPARATIVO REPOS GITHUB WHATSAPP 2026

Não-oficiais (Baileys-based ou Puppeteer-based — risco de ban ALTO em todos)

REPO	STARS	ÚLTIMA RELEASE	DOCS PT-BR	SETUP LEIGO	LICENÇA	VEREDITO
Evolution API	8.4k	Dez/2025 (ativo)	Parcial	4/5	Apache 2.0	⭐ Caminho A recomendado
WAHA	6.6k	Mai/2026 (semanal)	Não	2/5	Apache 2.0 / Plus pago	Alt. mais polida pra dev
WPPConnect	3.3k	Mai/2026 (ativo)	**SIM (BR)**	3/5	LGPL v3	Alt. leigo BR
Baileys	9.5k	Mai/2026 (muito ativo)	Não	5/5	MIT	Só devs (Evolution usa por baixo)
whatsapp-web.js	21.9k	Abr/2026 (moderado)	Tutoriais	3/5	Apache 2.0	LEGADO — não recomendar pra novo

Oficiais (Meta Cloud API — risco zero)

REPO/TOOLKIT	ESTADO	SETUP LEIGO	LICENÇA	VEREDITO
Meta SDK Node oficial	🚫 ARQUIVADO desde 2023	—	Meta	NÃO USAR (descontinuado)
@great-detail/whatsapp (Node)	v8.4.0 Nov/2025 ativo	3/5	MIT	Caminho C Node moderno
filipporomani/whatsapp-python	v4.3.0 Nov/2024 ativo	3/5	AGPL-3.0 ⚠	Caminho C Python (atenção AGPL)
Composio WhatsApp Toolkit	Freemium ativo, 17 tools	1/5	Comercial	⭐ Caminho C + IA recomendado

Recomendação final da aula:

- Caminho A (DIY técnico): **Evolution v2.3.7** — comunidade BR, multi-engine, integra n8n nativo
- Caminho B (SaaS leigo): **Z-API** — BR consolidado, doc PT-BR, NF-e
- Caminho C (oficial + IA): **Composio WhatsApp** via Rube no Claude Code — abstrai OAuth, 17 tools prontas

Insight importante pro aluno: Meta arquivou o SDK Node oficial em 2023 — não existe SDK oficial Meta mantido. A integração oficial em 2026 é via REST direto na Graph API OU via wrapper de terceiro (Composio é o mais polido pra IA agent).

Nota sobre ban risk: todos os não-oficiais (Evolution/WAHA/WPPConnect/Baileys/whatsapp-web.js) vão por Baileys/Puppeteer no fim — mesmo risco subjacente. Diferença é COMUNIDADE/MANUTENÇÃO, não segurança. Tempo médio de vida de número em produção agressiva: 2-8 semanas. Pra automação séria a médio prazo, migrar pra Cloud API oficial.

ANEXO B — CUSTOS DETALHADOS POR TRILHA

Já no Bloco 0.5 acima.

ANEXO C — CHECKLIST ANTI-BAN WHATSAPP NÃO-OFICIAL

1. Chip novo dedicado (R$30)
2. Android antigo, não usar em iPhone primário
3. Aquecimento 7-14 dias antes
4. Delay aleatório 8-30s entre respostas
5. Nunca templates 100% idênticos
6. Não comprar lista fria
7. Limite ~100 mensagens novas/dia início
8. Não responder grupos
9. Backup do banco de contatos (se banir, perde)
10. Plano B: número 2 reserva já aquecido

ANEXO D — TROUBLESHOOTING COMUM

- "Easypanel não abre na porta 3000": firewall HospedaInfo bloqueando → liberar via painel da HospedaInfo
- "Evolution não conecta": tentar reiniciar instância, escanear QR de novo
- "Z-API timeout webhook": workflow respondendo > 5s → adicionar Respond to Webhook + processar em background
- "Cloud API: template REJECTED": ver categoria errada (marketing escrito como utility) ou idioma diferente

- "Claude API erro 401": API key vazada/revogada — regerar no console

ANEXO E — PRÓXIMAS AULAS SUGERIDAS

1. Aula 2: Aprofundar Modo Operacional — customizar cadências, A/B test avançado, integrar Calendly/Cal.com, webhooks em produção
2. Aula 3: Conectar CRM externo (HubSpot/Pipedrive/RD) ao bot via MCP
3. Aula 4: Bot multimodal (áudio com Whisper, imagem com Vision)
4. Aula 5: Disparo em massa Cloud API com segmentação por enrich (CSV/Sheets)
5. Aula 6: Agente fazendo ações reais (agendar, enviar boleto, criar pedido)
6. Aula 7: Adaptar pacote Agent-Whatsapp pra outros providers (Z-API, Evolution, Cloud)

ANEXO F — ATIVOS ENTREGÁVEIS DA AULA (O QUE O ALUNO LEVA)

1. **Pacote `POW-F2A` (Polypus WhatsApp Toolkit, original — sem código de terceiros)**
   - ~13 scripts Python stdlib (escritos do zero pela Polypus, inspirados em padrões maduros do mercado)
   - 1 agente Claude Code `.md` custom (linguagem natural → CLI)
   - 4 perfis de `.env` (Zappfy / Z-API / Evolution / Cloud API)
   - SQLite com 7 tabelas + schema versionado
   - 5 playbooks de campanha (lançamento, oferta, reativação, pesquisa, evento)
   - **Licença:** distribuição liberada pra alunos F2A pagantes
2. **Workflow n8n base** (template JSON exportado) — Evolution + Claude API + RAG 2 camadas
3. **Slides em HTML** (estrutura design system Polypus)
4. **Playbook PDF passo-a-passo** (auto-estudo)
5. **Vídeo replay** da aula ao vivo
6. **Comunidade WhatsApp** dedicada à aula (suporte pós-aula)

DIAGRAMAS (ARQUIVOS SEPARADOS)

Modo A — Bot 24/7 reativo:

- `_diagramas/01-decision-tree.md` — escolha plug A/B/C
- `_diagramas/02-superestrutura-comum.md` — VPS + n8n + Postgres
- `_diagramas/03-plug-evolution.md` — fluxo Evolution
- `_diagramas/04-plug-zapi.md` — fluxo Z-API
- `_diagramas/05-plug-cloud-api.md` — fluxo Cloud API + Composio
- `_diagramas/06-agente-ia-memoria.md` — agente + RAG 2 camadas

Modo B — Operação direta via Claude Code:

- `_diagramas/07-modo-operacional-arquitetura.md` — Python stdlib + SQLite + agente .md
- `_diagramas/08-fluxo-campanha-via-claude-code.md` — fluxo real de campanha (live, lançamento, etc)

CASE DE REFERÊNCIA (STORYTELLING PRA AULA)

> *Pedro vive isso na pele. Conta em 1ª pessoa.*

Bot SDR Polypus (em produção desde 2026-05-14):

- Stack: n8n.polypus.agency + Evolution v2.3.7 + MiMo v2.5-pro
- Template base: n8n.io #13407 ("Human-like Evolution API WhatsApp agent")
- Instância "gio" = Giovana Polypus (chip dedicado)
- Workflow ID `BJaWHReNaYFDSUIR`
- **Bug real que Pedro pegou:** template não filtrava grupos → respondia em todo grupo de WhatsApp. Fix: filter `@g.us` + `fromMe`. Bug está documentado em `Bot-SDR-Polypus/ARQUITETURA.md`.
- **Decisão real que Pedro fez:** descartou OpenClaw + plano Claude Code como backend (ToS, ban risk). Usa Claude API key paga.
- **Arquitetura RAG real:** 2 camadas (determinística + semântica) — Pedro documentou o porquê.

Esse case dá credibilidade total à aula: aluno vê que Pedro não tá falando teoria, tá ensinando o que ele JÁ FEZ.

CHECKLIST PEDAGÓGICO (VALIDAÇÃO ANTES DE GRAVAR)

- [ ] Cada Bloco tem demo ao vivo no Claude Code
- [ ] Cada passo tem screenshot/screencast no playbook PDF
- [ ] Cada comando tem cópia-cola pronta (sem digitar)
- [ ] Avisos críticos repetidos 3x (não 1x — leigo esquece)
- [ ] Diagrama visual antes de cada bloco técnico
- [ ] Custos transparentes (não maquia)
- [ ] Plano B documentado pra cada gotcha conhecido
- [ ] Q&A de objeções comuns no fim
- [ ] Próximos passos claros (não deixa aluno sem rumo)

> *Versão: MAPA v1 · 2026-05-20 Próxima fase: produção dos materiais (slides HTML, playbook PDF, scripts copy-paste, demo em vídeo)*