Carla — Bot de Vendas no WhatsApp (FSM Blindada)

Vendedora virtual empática e persuasiva integrada ao WhatsApp via Baileys, com fluxos de conversa (FSM) orientados a conversão, telemetria de funil, sessão persistente no Railway e prontinha para clonar em novas personas.

📦Estrutura do Projeto

```
src/
  bot.js                # orquestra FSM e guarda-chuva de intents/guardrails
  model.js              # wrapper do provedor de IA (chat)
  telemetry.js          # eventos do funil (JSON/console)
  memory.js             # memória leve por user (TTL)
  prompts/
    base.js             # persona + style rules + productPrompt
  flows/
    greet.js            # abertura calorosa (2 frases + 1 pergunta)
    qualify.js          # mapeia tipo de cabelo/dor (SEM loops)
    offer.js            # oferta curta, benefícios, urgência (sem link)
    close.js            # só envia link com intenção clara + pós-venda
    postsale.js         # confirma comprovante, agradece e libera cupom
public/
  product.jpg           # foto do produto exibida na primeira interação
.env.example            # variáveis de ambiente (modelo)
```

🔧Requisitos

- **Node 18.x** (Baileys estável) — recomenda-se NVM
- Conta WhatsApp dedicada
- Git + GitHub
- (Prod) Conta Railway com **volume persistente**

⚙️Variáveis de Ambiente (`.env`)

Crie um `.env` na raiz do projeto:

```
OPENAI_API_KEY=coloque_sua_chave_aqui
MODEL_NAME=gpt-4o
PRICE_TARGET=170
PRICE_ORIGINAL=197
```

```
WPP_AUTH_DIR=./.wpp-auth
MEMORY_TTL_DAYS=7
PORT=8080
CHECKOUT_LINK=https://entrega.logzz.com.br/pay/memmpxgmg/progcreme170
COUPON_CODE=TOP-AGO2025-PROGRVG-150
```

> **Railway (produção):** use `WPP_AUTH_DIR=/app/baileys-auth` e monte um volume nesse caminho.

📩 Rodando Localmente

```
nvm use 18
npm install
mkdir .wpp-auth  # guarda a sessão do WhatsApp localmente
npm start
```

Parear WhatsApp: abra `http://127.0.0.1:8080/wpp/qr` e escaneie o QR pelo app (Dispositivos Conectados). A sessão fica salva e não precisa repetir.

Health check: `curl http://127.0.0.1:8080/health` → `{ "ok": true }`

Teste de webhook:

```
curl -X POST "http://127.0.0.1:8080/webhook"
  -H "Content-Type: application/json"
  -d '{"userId":"c001","text":"Oi, Carla!","context":{}}'
```

🗜️ Deploy no Railway (produção)

1) Conecte o repo do GitHub.

2) **Variables** (Settings → Variables):

```
OPENAI_API_KEY=...
MODEL_NAME=gpt-4o
PRICE_TARGET=170
PRICE_ORIGINAL=197
WPP_AUTH_DIR=/app/baileys-auth
MEMORY_TTL_DAYS=7
PORT=8080
CHECKOUT_LINK=...
COUPON_CODE=...
```

3) **Persistent Storage / Volumes** → **Add Volume** - **Mount path:** `/app/baileys-auth`

4) Deploy automático pela `main`.

Health (prod): `curl https://<SEU-APP>.up.railway.app/health`

Webhook (prod):

```
curl -X POST "https://<SEU-APP>.up.railway.app/webhook"
  -H "Content-Type: application/json"
  -d '{"userId":"c001","text":"Oi, Carla!","context":{}}'
```

FSM (Fluxo de Conversa)

1) greet - Abertura em até **2 frases + 1 pergunta** - Pede **nome** e registra em `memory` - Empatia + validação emocional + tom humano

2) qualify - Pergunta **uma vez** o **tipo de cabelo** caso ainda não saiba - Se já souber, mapeia **dor** (frizz, volume, chapinha, definição) - Sem loops, sem repetição

3) offer - Sempre inclui: **R$197 → R$170**, **COD (paga na entrega)**, **prazo** (24h capitais; 2 dias), **urgência** (estoque limitado) - Personaliza benefício por tipo de cabelo - **Sem link** nesta etapa - Trata dúvida vaga ("não sei... será que...") como objeção leve

4) close - **Só envia link** se houver **intenção clara** (comprar/checkout/fechar pedido) - Dúvidas informativas: responde **sem link** - **Pós-venda**: ao detectar comprovante, agradece e **libera cupom**

5) postsale - Confirma comprovante, agradece e envia `COUPON_CODE` - Encerramento com **1 pergunta leve** (ex.: aviso de saída para entrega)

📊 Telemetria (eventos)

- `session_start`, `midia_detectada`, `resposta_direta`, `erro`
- `abertura_enviada` (greet)
- `mapeamento_pergunta_enviada` / `dor_pergunta_enviada` (qualify)
- `oferta_mostrada` (offer)
- `checkout_enviado`, `pos_pagamento_enviado`, `cupom_liberado` (close/postsale)

Use esses eventos para achar gargalos: onde os leads somem? oferta? fechamento? ajuste mensagens por etapa.

🧱 Guardrails

- Não revelar identidade técnica (nada de "sou IA").

- Link **apenas** no close com intenção clara.
- Pedir **resumo em texto** quando receber mídia.
- Respeitar **2 frases + 1 pergunta** e **1 pergunta única** (sem interrogatório).

🧪 Roteiro de Testes (checklist rápido)

1. `/health` OK local e prod
2. QR exibido em `/wpp/qr` e sessão conectada (persistência ok)
3. `greet` pede **nome** (sem 2 perguntas)
4. `qualify` pergunta **tipo de cabelo** só 1x e depois mapeia **dor**
5. `offer` com **preço, COD, prazo, urgência** e **sem link**
6. `close` envia **link** apenas com intenção (simular “quero comprar”)
7. `postsale` reconhece **comprovante** e libera **cupom**
8. Telemetria registrando os eventos acima

🛠️ Git Workflow

Fluxo rápido (solo):

```
git add .
git commit -m "feat(flow): FSM blindada + telemetria"
git push origin main
```

Fluxo seguro (evita conflito):

```
git checkout main
git pull --rebase origin main
git add -A
git commit -m "feat: ajustes flows e guardrails"
git push origin main
```

🕯️ Troubleshooting

QR não aparece - Confirme `npm start` sem erros - Atualize `/wpp/qr` a cada 2–3s - Verifique Node 18 via `node -v`

Cai a sessão no Railway - Garanta `WPP_AUTH_DIR=/app/baileys-auth` **e** volume montado nesse path

Respostas repetitivas/loops - Confirme que está usando os **flows blindados** (`greet/qualify/offer/close/postsale`) - Cheque env `MODEL_NAME`, `PRICE_TARGET` e logs da `telemetry`

Link saindo antes da hora - Revise `close.js`: só envia link quando `shouldOfferLink()` detectar intenção

---

🧬Clonar para nova persona

1) Duplicar pasta da Carla 2) Ajustar `prompts/base.js` (persona + estilo) 3) Atualizar `productPrompt` (nome, preço, claims, foto) 4) Manter guardrails, telemetria e FSM 5) Subir nova instância (número próprio)

📄Licença

Uso interno da TopOfertas Express. Ajuste conforme necessidade.