ChatGPT

Implementação da página Instâncias (WhatsApp via WAHA API)

Para criar e gerenciar instâncias do WhatsApp diretamente no site, usamos a API REST do WAHA (WhatsApp HTTP API) no backend. A arquitetura proposta é: Node.js (Express) no servidor, banco PostgreSQL com Drizzle ORM para armazenar as instâncias, e React + Vite no frontend (com Wouter para roteamento). O fluxo geral é: o frontend chama rotas customizadas do Express, que por sua vez invocam os endpoints do WAHA (usando as variáveis de ambiente `WAHA_API` e `WAHA_API_KEY`), armazenam a sessão/instância no banco, e retornam dados (incluindo o QR Code) para o cliente. Em seguida, o usuário escaneia o QR Code exibido no navegador para conectar o WhatsApp.

Passo a passo de implementação

1. **Configurar o banco de dados (Drizzle + PostgreSQL)**. Crie uma tabela (por exemplo, `instances`) para armazenar as instâncias criadas, com colunas como `id` (PK autoincrement), `name` (nome/identificador da sessão), `status` (ex.: `STARTING`, `SCAN_QR`, `WORKING`), `createdAt`, etc. No Drizzle ORM, por exemplo:

```
// src/db/schema.ts
import { integer, varchar, timestamp } from "drizzle-orm/pg-core";
export const instances = pgTable("instances", {
  id: integer().primaryKey().generatedAlwaysAsIdentity(),
  name: varchar({ length: 100 }).notNull(),
  status: varchar({ length: 50 }).notNull(),
  createdAt: timestamp("created_at").defaultNow(),
});
```

   Defina a configuração do Drizzle (`drizzle.config.ts`) e aplique a migration (ou use `drizzle-kit push`) para criar a tabela no banco [1].

2. **Endpoint para criar instância (Express)**. No servidor Express, crie uma rota `POST /api/instancias` que fará as seguintes ações:

3. Use o axios (ou fetch) para chamar o endpoint do WAHA: `POST /api/sessions` [2]. Envie no corpo JSON o nome (ou deixe o WAHA gerar automaticamente) e `config` básico. Não esqueça de incluir o cabeçalho `X-Api-Key: <WAHA_API_KEY>` (conforme a [documentação WAHA] [3]). Por exemplo:

```
// Exemplo com axios no Node.js
const url = `${process.env.WAHA_API}/sessions`;
const headers = { 'Content-Type': 'application/json', 'X-Api-Key':
process.env.WAHA_API_KEY };
const data = { name: "instancia1", config: {} };
const response = await axios.post(url, data, { headers });
```

```
const sessionInfo = response.data; // contém { name,
status:"STARTING", ... }
```

4. A resposta conterá o nome da sessão criada e status inicial (`STARTING`) [4]. Insira um registro no banco Drizzle com esse `name` (session name) e status inicial. Por exemplo: `await db.insert(instances).values({ name: sessionInfo.name, status: sessionInfo.status });`.

5. Retorne ao cliente os dados da nova instância (nome, status, etc.). Opcionalmente inicie a sessão imediatamente (por padrão WAHA já inicia a sessão após criação) ou, se desejar, crie com `"start": false` e inicie depois via `POST /api/sessions/{name}/start` [5].

6. **Endpoint para listar instâncias**. Implemente `GET /api/instancias` que busca todas as instâncias do banco (`SELECT * FROM instances`) e retorna ao frontend. Isso permite ao React exibir a lista atualizada de instâncias criadas.

7. **Endpoint para obter o QR Code**. Crie uma rota `GET /api/instancias/:name/qr` no backend. Nela, faça outra chamada ao WAHA: `GET /api/{session}/auth/qr` [6], onde `{session}` é o nome da sessão (instância) obtido do banco. Para facilitar, solicite o QR Code em formato *base64* definindo o header `Accept: application/json`. A documentação do WAHA mostra que esse endpoint pode retornar o QR Code em diversos formatos (imagem binária, JSON com base64, ou "raw") [7]. Um exemplo usando axios:

```
const url = `${process.env.WAHA_API}/${sessionName}/auth/qr`;
const headers = { 'Accept': 'application/json', 'X-Api-Key':
process.env.WAHA_API_KEY };
const qrResponse = await axios.get(url, { headers });
// qrResponse.data provavelmente terá { qr: "<base64 string>" }
```

   Em seguida, envie esse base64 ao cliente no JSON da resposta. (Opcionalmente, poderia retornar a imagem diretamente com `responseType: 'arraybuffer'`, mas o base64 simplifica para incorporar no <img> do frontend.)

8. **Atualização de status (WebSocket ou polling)**. Enquanto o QR Code não for escaneado, a sessão ficará em status `SCAN_QR`. Quando o usuário escanear, o WAHA mudará para `WORKING`. Para notificar o cliente dessas mudanças em tempo real, você pode usar WebSocket no servidor. Por exemplo, o WAHA emite eventos como `session.status`. No backend, conecte-se (via webhook ou outra técnica) a esses eventos e retransmita-os via WebSocket para o frontend. Assim, o React pode atualizar a interface (por exemplo, ocultar o QR Code e mostrar "Conectado") automaticamente. (Se não usar WebSocket, pode-se simplesmente permitir que o cliente requeira periodicamente o status pela rota de lista de instâncias.)

9. **Front-end (React)**. Implemente uma página `Instâncias` (rota com Wouter, e.g. `/instancias`). No carregamento inicial, faça `GET /api/instancias` para buscar e exibir todas as instâncias existentes (nome, status). Cada instância pode exibir: nome, status atual, e – se em `SCAN_QR` – o QR Code. Também inclua um botão "Criar nova instância" que envia `POST /api/instancias`. Ao criar, receba a instância criada e atualize a lista exibida (usando estado React ou contexto). Para exibir o QR Code, use o valor base64 retornado pelo endpoint `/qr`: `<img src={"" + qrBase64} />`. Isso mostra o código

diretamente na página sem sair para o site do WAHA. Use o WebSocket (ou um effect de polling) para ouvir mudanças de status: assim que a sessão for autenticada (`WORKING`), remova o QR Code e indique que a conexão foi estabelecida.

10. **Fluxo de autenticação**. Em resumo, o usuário no frontend cria a instância, o backend registra no WAHA (status `STARTING`) [2], então o WAHA passa para `SCAN_QR` e você obtém o QR code via `/auth/qr` [6] [7]. O usuário escaneia o código com o app WhatsApp, o WAHA atualiza para `WORKING` e então a instância está pronta para uso (envio/recebimento de mensagens). Todo esse processo ocorre no próprio site, sem precisar abrir o painel do WAHA. Use as rotas e headers conforme a documentação oficial do WAHA [2] [6], garantindo incluir sempre o `X-Api-Key` para autenticação nas chamadas.

Fontes: Documentação oficial do WAHA para gerenciamento de sessões e QR Code [2] [8], e guia do Drizzle ORM para criação de tabelas no PostgreSQL [1].

---

[1] Drizzle ORM - PostgreSQL
https://orm.drizzle.team/docs/get-started/postgresql-new

[2] [3] [4] [5] [6] [7] [8] ○ Sessions | WAHA
https://waha.devlike.pro/docs/how-to/sessions/