Portal de Vagas CGB Energia

1. Visão Geral

O Portal de Vagas CGB Energia é uma aplicação web completa, desenvolvida para modernizar e otimizar o processo de recrutamento e seleção da empresa. Ele oferece uma interface pública para candidatos buscarem e se candidatarem a vagas, e um portal administrativo robusto para a equipe de RH gerenciar todo o ciclo de vida do recrutamento.

O projeto foi construído com foco em usabilidade, segurança e escalabilidade, utilizando tecnologias modernas como React, TypeScript e Supabase.

---

2. Funcionalidades Principais

Para Candidatos (Público)

- **Página Inicial Dinâmica:** Apresenta um mapa interativo com a geolocalização das vagas, estatísticas e uma lista de vagas abertas.
- **Busca e Filtragem Avançada:** Os candidatos podem filtrar vagas por estado, cidade, área de atuação e tipo de contrato. A busca por cidade é feita através de um campo pesquisável para melhor usabilidade.
- **Detalhes da Vaga:** Cada vaga tem uma página dedicada com descrição completa, requisitos e benefícios.
- **Formulário de Candidatura Inteligente:**
  - Validação de campos e de anexo de currículo.
  - Campos de CNH e Veículo são tratados de forma condicional.
  - Suporte para upload de arquivos `.pdf`, `.doc`, `.docx` com sanitização de nome para evitar erros.
- **Banco de Talentos:** Um formulário dedicado para que candidatos possam enviar seus currículos para futuras oportunidades.

Para Administradores e RH (Portal Administrativo)

- **Autenticação Segura:** Sistema de login com proteção contra ataques de força bruta (rate limiting).

- **Dashboard de Gestão:** Painel completo para gerenciamento de:

  - **Vagas:** Criar, editar, ativar/inativar e excluir vagas. O formulário de criação possui busca de cidades e restringe a criação por região para usuários de RH não-administradores.

  - **Candidatos:** Visualizar todos os candidatos, com filtros por vaga, status, localização, CNH e tipo de veículo.

  - **Processos Seletivos:** Um quadro Kanban (arrastar e soltar) para mover candidatos entre as etapas do processo seletivo (Cadastrado, Análise, Entrevista, Aprovado, Reprovado).

  - **Gestão de Equipe de RH:** Um administrador pode criar, editar e remover outros usuários do sistema, definindo permissões de acesso (Admin ou Recrutador) e restringindo o acesso por estados e cidades.

  - **Contratados e Banco de Talentos:** Abas dedicadas para gerenciar candidatos aprovados e os que se inscreveram no banco de talentos.

- **Comunicação com Candidatos:** Funcionalidade para enviar e-mail e contatar via WhatsApp diretamente do painel do candidato.

---

3. Tecnologias Utilizadas

- **Frontend:**

  - **React:** Biblioteca para construção da interface de usuário.

  - **TypeScript:** Para um código mais seguro e manutenível.

  - **Vite:** Ferramenta de build extremamente rápida.

  - **Tailwind CSS:** Framework CSS para estilização.

  - **Shadcn/ui:** Componentes de UI pré-construídos e customizáveis.

  - **TanStack Query (React Query):** Para gerenciamento de estado do servidor, cache e sincronização de dados.

  - **Leaflet & React-Leaflet:** Para o mapa interativo.

  - **react-beautiful-dnd:** Para a funcionalidade de arrastar e soltar no quadro Kanban.

- **Backend (BaaS - Backend as a Service):**

- **Supabase:** Plataforma open-source que provê banco de dados PostgreSQL, autenticação, armazenamento de arquivos e Edge Functions.

- **Geocodificação:**

  - **Nominatim API (OpenStreetMap):** Para converter nomes de cidades em coordenadas geográficas dinamicamente.

---

4. Guia de Instalação e Configuração

Siga os passos abaixo para configurar e rodar o projeto em um ambiente de desenvolvimento local.

Pré-requisitos

- **Node.js:** Versão 18 ou superior.
- **npm:** Gerenciador de pacotes (geralmente vem com o Node.js).

Passo 1: Clonar o Repositório

```bash
git clone <URL_DO_SEU_REPOSITORIO>
cd <NOME_DA_PASTA_DO_PROJETO>
```

Passo 2: Instalar as Dependências

Execute o seguinte comando para instalar todas as bibliotecas e pacotes necessários:

```bash
npm install
```

Passo 3: Configurar as Variáveis de Ambiente

1. Na raiz do projeto, você encontrará um arquivo chamado `.env.example`.
2. Crie uma cópia deste arquivo e renomeie-a para `.env`.
3. Abra o arquivo `.env` e preencha com as suas chaves do Supabase. Você pode encontrá-las no painel do seu projeto em **Settings > API**.

```
VITE_SUPABASE_URL="SUA_URL_DO_SUPABASE_AQUI"
VITE_SUPABASE_ANON_KEY="SUA_ANON_KEY_DO_SUPABASE_AQUI"
```

Passo 4: Rodar o Projeto

Após a instalação e configuração, inicie o servidor de desenvolvimento:

```bash
npm run dev
```

O projeto estará disponível em `http://localhost:8080` (ou outra porta, se a 8080 estiver em uso).

Passo 5: Rodar a Build de Produção

Quando estiver pronto para fazer o deploy (enviar para a hospedagem), gere a versão otimizada do site com o comando:

```bash
npm run build
```

Isso criará uma pasta `dist` com todos os arquivos estáticos prontos para serem hospedados.

---

5. Documentação Técnica

Esta seção é destinada a desenvolvedores e detalha a arquitetura, as convenções e as principais lógicas do projeto.

5.1. Estrutura do Projeto

A estrutura de pastas do `src` foi organizada para separar as responsabilidades:

- `src/components`: Contém componentes React reutilizáveis.
  - `src/components/ui`: Componentes de UI base (gerados pelo Shadcn).
  - `src/components/admin`: Componentes complexos, específicos do portal administrativo.
- `src/hooks`: Hooks customizados, responsáveis por toda a lógica de negócio e acesso a dados.
  - `useAuth.tsx`: Gerencia o estado de autenticação do usuário.
  - `useJobs.tsx`: Lógica para CRUD de Vagas.
  - `useCandidates.tsx`: Lógica para CRUD de Candidatos.
  - `useRH.tsx`: Lógica para gerenciamento da equipe de RH.
- `src/pages`: Representa cada página da aplicação, montando os componentes e hooks.
- `src/lib`: Funções utilitárias (`utils.ts`), constantes (`constants.ts`) e validadores (`validators.ts`).
- `src/integrations/supabase`: Configuração do cliente Supabase e tipos do banco de dados.
- `supabase/functions`: Código-fonte das Edge Functions que rodam no backend.

5.2. Gerenciamento de Estado

A aplicação utiliza duas estratégias principais para gerenciamento de estado:

1. **Estado do Servidor (Server State):** `TanStack Query` (`@tanstack/react-query`) é a biblioteca principal para buscar, armazenar em cache e atualizar dados do Supabase. Todos os hooks em `src/hooks` que interagem com o banco de dados (`useJobs`, `useCandidates`, etc.) usam `useQuery` para leitura e `useMutation` para escrita (criação, atualização, exclusão). Isso nos dá, gratuitamente, cache, revalidação automática e uma UI otimista.

2. **Estado da UI (UI State):** Para o estado local de componentes (como a abertura de um modal, o conteúdo de um formulário, etc.), utilizamos os hooks padrão do React, principalmente `useState` e `useE ect`. A regra é manter o estado o mais local possível. Em casos de formulários complexos (como em `JobManagement.tsx`), o padrão "levantar o estado" (*lifting state up*) é utilizado, onde o componente pai gerencia o estado e o passa para o componente filho (o formulário).

5.3. Arquitetura e Lógica Chave

- **Autenticação (`useAuth` + `ProtectedRoute`):** O `useAuth.tsx` utiliza o `onAuthStateChange` do Supabase para ouvir o estado de autenticação em tempo real. O estado de `loading` é cuidadosamente gerenciado para evitar loops de renderização. O componente `ProtectedRoute.tsx` envolve as rotas administrativas, verificando o estado de `loading` e o `user` do `useAuth` antes de renderizar a página solicitada ou redirecionar para o login.

- **Permissões de RH (`RHManagement.tsx` e outros):** A lógica de permissão é centralizada no hook `useRHProfile`. Os componentes que precisam de restrição regional (como `JobManagement`, `SelectionProcess`, `CandidateManagement`) buscam o perfil do RH logado e filtram os dados (vagas, candidatos) com base nos campos `is_admin`, `assigned_states` e `assigned_cities`.

- **Geocodificação Dinâmica (`JobsMap.tsx`):** Para evitar uma lista estática de coordenadas, o mapa utiliza a API pública do Nominatim para buscar as coordenadas de cidades desconhecidas. As coordenadas encontradas são salvas no `localStorage` do navegador para otimizar carregamentos futuros e evitar chamadas excessivas à API.

5.4. Backend (Supabase)

- **Segurança (RLS):** **A segurança do sistema depende 100% das políticas de Row Level Security (RLS) ativadas no Supabase.** É fundamental que cada tabela tenha políticas que definam quem pode `SELECT`, `INSERT`, `UPDATE` e `DELETE`.

  - **Exemplo de Política:** Uma política na tabela `candidates` pode permitir que um usuário de RH (`rh_users`) veja um candidato (`SELECT`) apenas se o estado da vaga do candidato estiver na lista de `assigned_states` do perfil do RH.

- **Edge Functions:**

  - `create-user-direct`: Esta função é chamada pelo frontend para criar um novo usuário de RH. Ela utiliza a `service_role_key` do Supabase para ter privilégios de administrador e (1) criar o usuário no sistema de **Auth** do Supabase e (2) inserir o perfil correspondente na tabela `rh_users` com as permissões corretas. Isso é necessário porque a criação de usuários é uma ação privilegiada.

  - `send-email`: Uma função simples que recebe os detalhes do e-mail (destinatário, assunto, corpo) e utiliza as credenciais SMTP (configuradas como "Secrets" no Supabase) para enviar o e-mail.

5.5. Convenções de Código

- **Componentes:** Componentes complexos são divididos em subcomponentes menores para facilitar a manutenção.

- **Hooks:** Toda a lógica de acesso a dados e de negócio complexa é abstraída em hooks customizados na pasta `src/hooks`. Isso mantém os componentes de página (`src/pages`) mais limpos e focados na renderização.

- **Estilização:** A estilização é feita primariamente com **Tailwind CSS**. Componentes reutilizáveis de UI são criados a partir da biblioteca **Shadcn/ui**.