Documentação Técnica - Infraestrutura e Dados (RH ZELLO)

Esta documentação detalha a infraestrutura de banco de dados, modelagem de dados e práticas de segurança implementadas para o projeto RH ZELLO.

Responsável Técnico: Victor (Assistente de IA)

Destinatário: Kaua (Especialista Backend)

Data: 05/02/2026

---

1. Visão Geral da Arquitetura

O projeto utiliza uma arquitetura containerizada para o banco de dados, garantindo isolamento e reprodutibilidade. A aplicação backend comunica-se com o banco através do ORM SQLAlchemy, utilizando Alembic para gerenciamento de versões do esquema (migrações).

- **Banco de Dados:** PostgreSQL 15 (Docker)
- **Linguagem:** Python 3.10+
- **ORM:** SQLAlchemy
- **Migrações:** Alembic
- **Segurança:** Passlib + Bcrypt

2. Estrutura de Diretórios

```
backend_db/
├── migrations/          # Scripts de migração do Alembic
│   ├── versions/        # Histórico de alterações do banco
│   ├── env.py           # Configuração do ambiente de migração
│   └── script.py.mako   # Template para novas migrações
├── .env                 # Variáveis de ambiente (NÃO COMITAR)
├── .gitignore           # Arquivos ignorados pelo Git
```

```
├── alembic.ini          # Configuração principal do Alembic (Traduzido)
├── create_user.py       # Script utilitário para criar admins
├── database.py          # Configuração de conexão com o banco
├── docker-compose.yml   # Definição do container PostgreSQL
├── models.py            # Definição das tabelas (ORM)
├── requirements.txt     # Dependências do projeto
└── security.py          # Lógica de hash e verificação de senhas
```

3. Configuração do Ambiente

3.1 Pré-requisitos

- Docker e Docker Compose instalados.
- Python 3.10 ou superior.

3.2 Variáveis de Ambiente (.env)

O arquivo `.env` deve estar na raiz de `backend_db/` com o seguinte conteúdo (segurança):

```
# Configurações do Banco de Dados
DATABASE_URL=postgresql://admin_rh:senha_segura_123@localhost:5433/db_rh
```

> **Nota:** A porta externa do banco foi configurada para `5433` para evitar conflitos com instalações locais padrão do PostgreSQL na porta 5432.

4. Banco de Dados (Docker)

O serviço de banco de dados é definido no `docker-compose.yml`.

- **Container Name:** `pg_rh_inteligente`
- **Imagem:** `postgres:15`
- **Porta:** `5433` (Host) -> `5432` (Container)
- **Volume:** `pgdata_rh` (Persistência de dados)

Comandos Úteis:

```
# Iniciar o banco em segundo plano
docker-compose up -d

# Parar o banco
docker-compose down

# Verificar logs
docker logs pg_rh_inteligente
```

5. Modelagem de Dados (`models.py`)

O esquema foi otimizado para performance em consultas de dashboard e integridade referencial.

5.1 Tabela `usuarios`

Gerencia o acesso ao sistema.

- `id` : PK, Integer.
- `email` : String(255), Unique. (Login)
- `senha_hash` : String(255). (Armazena hash bcrypt, nunca texto plano)
- **Método** `__repr__` : Implementado para facilitar debug.

5.2 Tabela `colaboradores` (Mestre RH)

Tabela principal com dados demográficos e contratuais.

- `id` : PK, Integer.
- `nome` , `sexo` , `idade` , `regiao` .
- `tipo_contrato` : (CLT, PJ, Temporário).
- `departamento` : String(100).
- `data_atualizacao` : Timestamp automático na atualização.

Índices de Performance (Otimização):

- `idx_departamento`: Acelera filtros por setor no dashboard.
- `idx_data_atualizacao`: Otimiza consultas de dados recentes.

5.3 Tabela `historico_contratos` (Auditoria/RPA)

Rastreia mudanças contratuais para fins de auditoria.

- `id`: PK, Integer.
- `colaborador_id`: FK -> `colaboradores.id`.
- `tipo_antigo`, `tipo_novo`: Registro da mudança.
- `data_mudanca`: Timestamp do evento.

6. Segurança (`security.py`)

A segurança das senhas é garantida utilizando a biblioteca `passlib` com o algoritmo `bcrypt` (padrão de mercado).

- **Hash:** As senhas são transformadas em hash antes de salvar no banco.
- **Verificação:** O login compara a senha fornecida com o hash armazenado.
- **Contexto:** `CryptContext(schemes=["bcrypt"], deprecated="auto")`

7. Migrações (Alembic)

O Alembic gerencia a evolução do esquema do banco de dados de forma segura e versionada.

Fluxo de Trabalho:

1. **Fazer alterações** nos modelos em `models.py`.
2. **Gerar uma nova migração:**

```bash
alembic revision --autogenerate -m "descricao_da_mudanca"
```

3. **Aplicar as alterações no banco:**

```bash
alembic upgrade head
```

Configuração Importante:

O arquivo `migrations/env.py` foi refatorado para ler a `DATABASE_URL` do arquivo `.env`. Isso garante que as credenciais não fiquem expostas no código fonte.

8. Scripts Utilitários

Criar Usuário Admin (`create_user.py`)

Script robusto para criação inicial de usuários, com verificação de duplicidade.

Uso:

```
python create_user.py --email admin@rhzello.com --password minhasenha123
```

Ou modo interativo (apenas rodar o script):

```
python create_user.py
```

Próximos Passos (Sugestão para o Backend)

1. **API:** Implementar rotas (FastAPI/Flask) consumindo os modelos definidos.
2. **Auth:** Criar sistema de autenticação JWT utilizando a verificação de senha do `security.py`.
3. **CRUD:** Desenvolver rotas para gerenciar `colaboradores`.

```
Esta documentação cobre **todas** as partes implementadas:

1.  **Infraestrutura Docker** (com a porta correta 5433).

2.  **Modelagem Otimizada** (com índices explicados).

3.  **Segurança** (explicação do `security.py`).
```

```
4.  **Alembic** (como usar as migrações).



**
```