Guia de Deploy do Sistema de Gestão de Mudanças (Streamlit + Supabase)

Este guia detalha os passos necessários para hospedar o aplicativo Streamlit no **Streamlit Community Cloud** (ou Render) e conectá-lo ao banco de dados **Supabase (PostgreSQL)** de forma segura.

1. Arquivos Fornecidos

Você recebeu os seguintes arquivos:

1. `app.py` : O script principal do Streamlit, modificado para usar a conexão com o banco de dados.
2. `connection.py` : O módulo Python que gerencia a conexão com o PostgreSQL (Supabase) usando `st.secrets` e contém as funções CRUD (Create, Read, Update, Delete) e de inicialização do banco de dados.
3. `requirements.txt` : Lista de dependências Python necessárias para o ambiente de hospedagem.

2. Configuração do Banco de Dados Supabase

O aplicativo está configurado para usar um banco de dados PostgreSQL. O Supabase é a opção recomendada.

2.1. Criar Projeto Supabase

1. Acesse o Supabase Dashboard e crie um novo projeto.
2. Anote a **Senha do Banco de Dados** (Database Password) durante a criação.

2.2. Obter Credenciais de Conexão

Após a criação do projeto:

1. No painel lateral, vá para **Project Settings** (Configurações do Projeto) > **Database** (Banco de Dados).
2. Na seção **Connection String** (String de Conexão), selecione **URI**.
3. Você precisará extrair as seguintes informações da string de conexão (ou da seção **Connection Parameters**):
   - `host` : O endereço do host do seu banco de dados.
   - `database` : O nome do banco de dados (geralmente `postgres`).

- `user` : O usuário do banco de dados (geralmente `postgres`).
- `port` : A porta de conexão (geralmente `5432`).
- `password` : A senha que você definiu na criação do projeto.

3. Configuração de Credenciais Seguras (`secrets.toml`)

O Streamlit Community Cloud usa um arquivo `.streamlit/secrets.toml` para gerenciar credenciais de forma segura através de `st.secrets` .

3.1. Criar o Arquivo `secrets.toml`

Crie uma pasta chamada `.streamlit` na raiz do seu repositório (junto com `app.py` e `requirements.txt`). Dentro dela, crie o arquivo `secrets.toml` com o seguinte formato, substituindo os valores pelos seus:

Plain Text

```
# .streamlit/secrets.toml

[postgres]
host = "seu_host_do_supabase.supabase.co"
database = "postgres"
user = "postgres"
password = "sua_senha_do_banco_de_dados"
port = 5432
```

Importante: Ao fazer o deploy no Streamlit Community Cloud, você deve inserir essas credenciais diretamente na interface de **Secrets** do aplicativo, e não no arquivo `secrets.toml` do repositório (que deve ser ignorado pelo Git).

4. Deploy no Streamlit Community Cloud

4.1. Preparar o Repositório Git

1. Crie um novo repositório Git (GitHub, GitLab, etc.).
2. Adicione os arquivos `app.py` , `connection.py` e `requirements.txt` ao repositório.
3. Crie um arquivo `.gitignore` e adicione a linha `.streamlit/secrets.toml` para garantir que suas credenciais não sejam enviadas acidentalmente.

4.2. Configurar o Deploy

1. Acesse o Streamlit Community Cloud e faça login.

2. Clique em **New app** (Novo aplicativo).
3. Selecione o repositório onde você salvou os arquivos.
4. Em **Advanced settings** (Configurações avançadas), vá para a seção **Secrets**.
5. Adicione as credenciais do Supabase no formato TOML (o mesmo conteúdo do `secrets.toml`):

   Plain Text

   ```
   [postgres]
   host = "seu_host_do_supabase.supabase.co"
   database = "postgres"
   user = "postgres"
   password = "sua_senha_do_banco_de_dados"
   port = 5432
   ```

6. Clique em **Deploy!**

O Streamlit fará o download das dependências listadas em `requirements.txt` (incluindo `psycopg2-binary`) e executará o `app.py`. Na primeira execução, o código de inicialização do banco de dados em `app.py` chamará `init_db_structure(conn)` para criar as tabelas e inserir os dados de demonstração.

5. Próximos Passos

- **Segurança:** O código de demonstração usa senhas em texto simples. Em um ambiente de produção real, **você deve implementar hash de senhas** (ex: usando `bcrypt` ou `passlib`) e ajustar a lógica de login em `app.py` e `connection.py`.
- **Dados:** O aplicativo agora persiste os dados no Supabase. O `st.session_state.data` é atualizado a partir do banco de dados a cada operação de escrita (insert/update).
- **Escopo:** A lógica de escopo (filtragem por `secretaryId`) foi mantida e agora usa os IDs gerados pelo banco de dados.

---

*Documento gerado por **Manus AI***