Guia de Migração da Aplicação Lacrei Saúde Insights para Novo Ambiente

Este documento descreve, de forma completa e detalhada, o processo para migrar a aplicação **Lacrei Saúde Insights** — atualmente acessível em `http://44.197.116.219/login` — para qualquer outro ambiente interno ou servidor da empresa.

A aplicação está disponível no repositório oficial:

GitHub:
https://github.com/ICEI-PUC-Minas-PMV-SI/pmv-si-2025-2-pe4-t3-lacrei_insights.git

---

1. Visão Geral da Arquitetura

A aplicação é composta pelos seguintes elementos:

- Backend em **Python (Flask)**
- Banco de dados **PostgreSQL**
- ETL automatizado em Python
- Dashboard construído em **Metabase**
- Infraestrutura via **Docker e Docker Compose**
- Frontend em HTML, CSS e JavaScript

Toda a aplicação foi projetada para ser executada de forma containerizada, garantindo portabilidade e facilidade na migração.

---

2. Pré-requisitos do Novo Ambiente

Antes de iniciar a migração, certifique-se de que o ambiente de destino possui:

2.1. Dependências

- Linux (Ubuntu 20.04+, Debian, Amazon Linux, Rocky, etc.)
- Docker 24+
- Docker Compose 2.x+
- Porta 3000 liberada (Metabase)
- Porta 5432 liberada (PostgreSQL)
- Porta 80 ou 8080 (caso use Nginx)
- Acesso ao GitHub

2.2. Acesso e Permissões

- Usuário com permissão `sudo`
- Permissão para criar volumes e containers Docker
- Acesso ao servidor/VM

---

3. Clonando o Repositório

No servidor de destino, execute:

```
sudo git clone
https://github.com/ICEI-PUC-Minas-PMV-SI/pmv-si-2025-2-pe4-t3-lacrei
_insights.git
cd pmv-si-2025-2-pe4-t3-lacrei_insights
```

---

4. Configuração das Variáveis de Ambiente

Dentro da pasta do projeto, existe um arquivo `.env.example`.
Use-o como base:

```
cp .env.example .env
```

Depois, edite:

```
nano .env
```

Configurações importantes:

4.1. Banco de Dados

```
POSTGRES_USER=seu_usuario
POSTGRES_PASSWORD=sua_senha
POSTGRES_DB=lacrei_db
```

4.2. Backend Flask

```
FLASK_ENV=production
FLASK_SECRET_KEY=sua_chave_segura
```

4.3. Metabase

```
METABASE_SITE_URL=http://SEU_IP/metabase
```

IMPORTANTE:
O Metabase só gera links públicos corretos se o parâmetro `METABASE_SITE_URL` estiver adequado ao novo ambiente.

---

5. Subindo a Aplicação com Docker

Na raiz do projeto, execute:

```
sudo docker compose up -d --build
```

Isso iniciará:

- Container do **PostgreSQL**
- Container do **Flask API**

- Container do **Metabase**
- Volumes persistentes de banco e metabase

Para verificar:

```
docker ps
```

---

6. Estrutura de Containers Esperada

Você deverá visualizar algo como:

Serviço	Porta	Função
Flask Backend	5000	API da aplicação
PostgreSQL	5432	Banco de dados
Metabase	3000	Dashboard

---

7. Configuração do Metabase no Novo Ambiente

Acesse:

```
http://SEU_IP:3000
```

Faça a configuração inicial:

1. Crie um usuário admin
2. Configure o idioma (pt-BR)
3. Aponte o Metabase para o PostgreSQL do novo ambiente
4. Importe ou recrie dashboards conforme necessário

Se você possuir backups do Metabase, basta restaurar o volume:

```
docker volume inspect metabase_data
```

E movê-lo conforme necessário.

---

8. Carregando os Dados no Banco

A aplicação contém scripts ETL na pasta:

```
/app/services/etl/
```

Para executar manualmente:

```
docker exec -it nome_do_container_flask python etl/main.py
```

Ou pode-se configurar um CRON containerizado (caso queira automação).

---

9. Testando a Aplicação Após a Migração

Após o deploy:

9.1. Aplicação Web

Acesse:

```
http://SEU_IP/login
```

9.2. API

Teste:

```
http://SEU_IP/api/health
```

9.3. Banco

```
docker exec -it postgres_container psql -U usuario
```

9.4. ETL

Confirme:

- Conexão ao banco
- Execução dos scripts
- Geração e atualização de tabelas

9.5. Metabase

Verifique se os dashboards:

- Estão carregando corretamente
- Enxergam o novo banco
- Estão atualizando os dados

---

10. Configuração de Proxy Reverso (opcional)

Para expor a aplicação em domínio próprio, use Nginx:

```
server {
    listen 80;
    server_name SEU_DOMINIO;

    location / {
        proxy_pass http://localhost:5000;
    }

    location /metabase/ {
        proxy_pass http://localhost:3000/;
    }
}
```

Reinicie:

```
sudo systemctl restart nginx
```

---

11. Backup e Restauração (recomendado)

11.1. Backup do Banco

```
docker exec -t postgres_container pg_dump -U usuario lacrei_db > backup.sql
```

11.2. Backup do Metabase

```
docker run --rm -v metabase_data:/data alpine tar -cvzf metabase_backup.tar.gz /data
```

11.3. Restauração

```
docker exec -i postgres_container psql -U usuario lacrei_db < backup.sql
```

---

12. Conclusão

Após seguir todas as etapas acima, a aplicação estará totalmente migrada e funcionando no novo ambiente.
Este processo garante:

- Banco restaurado
- API operando
- ETL funcionando
- Dashboards ativos

- Ambiente idêntico ao original