



Configuração de Produção - PostgreSQL



Por que PostgreSQL?

- ✓ **Superior para Analytics:** Funções analíticas avançadas
- ✓ **JSON Nativo:** Melhor que MySQL para dados estruturados
- ✓ **Performance:** Otimizado para queries complexas
- ✓ **Extensibilidade:** PostGIS, TimescaleDB para futuro
- ✓ **ACID Compliant:** Transações seguras
- ✓ **Ecosystem Python:** Melhor integração



Configuração Completa

1. 📦 Instalar PostgreSQL

Ubuntu/Debian:

```
bash

sudo apt update
sudo apt install postgresql postgresql-contrib
sudo systemctl start postgresql
sudo systemctl enable postgresql
```

macOS:

```
bash

brew install postgresql
brew services start postgresql
```

Windows:

- Baixe em: <https://www.postgresql.org/download/windows/>
- Instale com senha para usuário (postgres)

Docker (Recomendado para Desenvolvimento):

```
bash

# Usar docker-compose já configurado
docker-compose up -d db
```

2. Criar Banco e Usuário

```
sql

-- Conectar como postgres
sudo -u postgres psql

-- Criar usuário
CREATE USER pharmaletics_user WITH ENCRYPTED PASSWORD 'SuaSenhaSegura123!';

-- Criar banco
CREATE DATABASE pharmaletics_prod OWNER pharmaletics_user;

-- Conceder privilégios
GRANT ALL PRIVILEGES ON DATABASE pharmaletics_prod TO pharmaletics_user;

-- Conectar ao banco
\c pharmaletics_prod

-- Conceder privilégios no schema public
GRANT ALL ON SCHEMA public TO pharmaletics_user;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO pharmaletics_user;
GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA public TO pharmaletics_user;
ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT ALL ON TABLES TO pharmaletics_user;
ALTER DEFAULT PRIVILEGES IN SCHEMA public GRANT ALL ON SEQUENCES TO pharmaletics_user;

-- Sair
\q
```

3. Configurar Variáveis de Ambiente

Crie dois arquivos de ambiente:

.env (Desenvolvimento/Testes):

```
bash
```

Aplicação

APP_NAME=QSDPharmalitics

APP_VERSION=2.0.0

DEBUG=true

PostgreSQL - Desenvolvimento

POSTGRES_SERVER=localhost

POSTGRES_USER=pharmalitics_user

POSTGRES_PASSWORD=SuaSenhaSegura123!

POSTGRES_DB=pharmalitics_dev

POSTGRES_PORT=5432

URL construída automaticamente

DATABASE_URL=postgresql://pharmalitics_user:SuaSenhaSegura123!@localhost:5432/pharmalitics_dev

Configurações da API

API_V1_STR=/api/v1

SECRET_KEY=dev-secret-key-change-in-production

CORS

BACKEND_CORS_ORIGINS=["*"]

Cache e Performance

CACHE_TTL_MINUTES=30

ENABLE_CACHE=true

Logs

LOG_LEVEL=DEBUG

Relatórios

REPORTS_DIR=./reports

TIMEZONE=America/Sao_Paulo

.env.production (Produção):

bash

Aplicação

APP_NAME=QSDPharmalitics

APP_VERSION=2.0.0

DEBUG=false

PostgreSQL - Produção

POSTGRES_SERVER=seu-servidor-postgres.com

POSTGRES_USER=pharmalitics_user

POSTGRES_PASSWORD=SenhaUltraSeguraProd2024!@#

POSTGRES_DB=pharmalitics_prod

POSTGRES_PORT=5432

URL de produção

DATABASE_URL=postgresql://pharmalitics_user:SenhaUltraSeguraProd2024!@#@seu-servidor-postgres.com:5432/p

Segurança de Produção

SECRET_KEY=sua-chave-ultra-secreta-256-bits-aqui

API_V1_STR=/api/v1

CORS específico para produção

BACKEND_CORS_ORIGINS=["https://seu-dominio.com","https://app.seu-dominio.com"]

Performance de Produção

CACHE_TTL_MINUTES=60

ENABLE_CACHE=true

Logs de Produção

LOG_LEVEL=INFO

Configurações de Produção

REPORTS_DIR=/app/reports

TIMEZONE=America/Sao_Paulo

4. Configurar Alembic (Migrations)

alembic/env.py (Atualizado):

```
python
```

```
from logging.config import fileConfig
from sqlalchemy import engine_from_config
from sqlalchemy import pool
from alembic import context
import os
import sys
```

```
# Adicionar path para imports
```

```
sys.path.append(os.path.dirname(os.path.dirname(os.path.realpath(__file__))))
```

```
from app.core.config import settings
from app.models.base import BaseModel
```

```
# Interpretação da configuração do Alembic
```

```
config = context.config
```

```
# Configurar logging
```

```
if config.config_file_name is not None:
    fileConfig(config.config_file_name)
```

```
# Definir metadata para auto-geração
```

```
target_metadata = BaseModel.metadata
```

```
def get_url():
    return settings.database_url
```

```
def run_migrations_offline() -> None:
    """Executar migrations em modo offline."""
    url = get_url()
    context.configure(
        url=url,
        target_metadata=target_metadata,
        literal_binds=True,
        dialect_opts={"paramstyle": "named"},
        compare_type=True,
        compare_server_default=True,
    )
```

```
with context.begin_transaction():
    context.run_migrations()
```

```
def run_migrations_online() -> None:
    """Executar migrations em modo online."""
    configuration = config.get_section(config.config_ini_section)
    configuration["sqlalchemy.url"] = get_url()
```

```
connectable = engine_from_config(
    configuration,
    prefix="sqlalchemy.",
    poolclass=pool.NullPool,
)

with connectable.connect() as connection:
    context.configure(
        connection=connection,
        target_metadata=target_metadata,
        compare_type=True,
        compare_server_default=True,
    )

    with context.begin_transaction():
        context.run_migrations()

if context.is_offline_mode():
    run_migrations_offline()
else:
    run_migrations_online()
```

5. 🐳 Docker Compose para Produção

docker-compose.prod.yml:

yaml

version: '3.8'

services:

PostgreSQL

postgres:

image: postgres:15-alpine

container_name: pharmalitics_postgres

environment:

POSTGRES_DB: pharmalitics_prod

POSTGRES_USER: pharmalitics_user

POSTGRES_PASSWORD: SuaSenhaSegura123!

PGDATA: /var/lib/postgresql/data/pgdata

volumes:

- postgres_data:/var/lib/postgresql/data

- ./backups:/backups

ports:

- "5432:5432"

networks:

- pharmalitics-net

restart: unless-stopped

PgAdmin

pgadmin:

image: dpage/pgadmin4:latest

container_name: pharmalitics_pgadmin

environment:

PGADMIN_DEFAULT_EMAIL: admin@pharmalitics.local

PGADMIN_DEFAULT_PASSWORD: admin123

volumes:

- pgadmin_data:/var/lib/pgadmin

ports:

- "8080:80"

networks:

- pharmalitics-net

depends_on:

- postgres

restart: unless-stopped

Redis para Cache

redis:

image: redis:7-alpine

container_name: pharmalitics_redis

ports:

- "6379:6379"

volumes:

- redis_data:/data

networks:

- pharmalitics-net

restart: unless-stopped

command: redis-server --appendonly yes --requirepass SuaRedisPassword123

API

api:

build: .

container_name: pharmalitics_api

environment:

- DATABASE_URL=postgresql://pharmalitics_user:SuaSenhaSegura123!@postgres:5432/pharmalitics_prod
- REDIS_URL=redis://:SuaRedisPassword123@redis:6379/0

ports:

- "8000:8000"

volumes:

- ./reports:/app/reports
- ./logs:/app/logs

networks:

- pharmalitics-net

depends_on:

- postgres
- redis

restart: unless-stopped

volumes:

postgres_data:

pgadmin_data:

redis_data:

networks:

pharmalitics-net:

driver: bridge

Comandos de Setup

Desenvolvimento:

bash

1. Iniciar PostgreSQL

```
docker-compose up -d postgres pgadmin
```

2. Criar migrations iniciais

```
alembic revision --autogenerate -m "Initial migration"
```

3. Executar migrations

```
alembic upgrade head
```

4. Popular dados de teste

```
python scripts/populate_sample_data.py
```

5. Executar API

```
uvicorn app.main:app --reload --host 0.0.0.0 --port 8000
```

Produção:

```
bash
```

1. Configurar ambiente

```
cp .env.production .env
```

2. Iniciar todos os serviços

```
docker-compose -f docker-compose.prod.yml up -d
```

3. Executar migrations

```
docker-compose -f docker-compose.prod.yml exec api alembic upgrade head
```

4. Verificar status

```
docker-compose -f docker-compose.prod.yml ps
```

Migração de Dados (SQLite → PostgreSQL)

Quando tiver dados no SQLite e quiser migrar:

Script de Migração:

```
bash
```

1. Backup SQLite

```
cp pharmaletics.db pharmaletics_backup.db
```

2. Executar script de migração

```
python scripts/migrate_sqlite_to_postgres.py
```