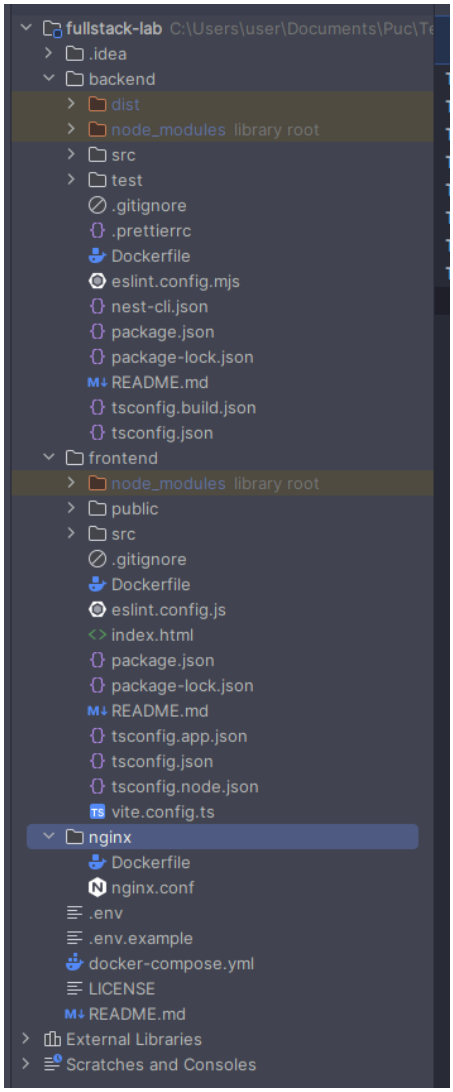
Evidências - LAB02: Ambiente de Desenvolvimento com Docker Compose

Este documento reúne as evidências práticas da execução do LAB02, que consistiu na criação de um ambiente de desenvolvimento full-stack utilizando Docker Compose. O ambiente contempla os serviços de frontend, backend, banco de dados PostgreSQL, administração via pgAdmin e proxy reverso com Nginx.

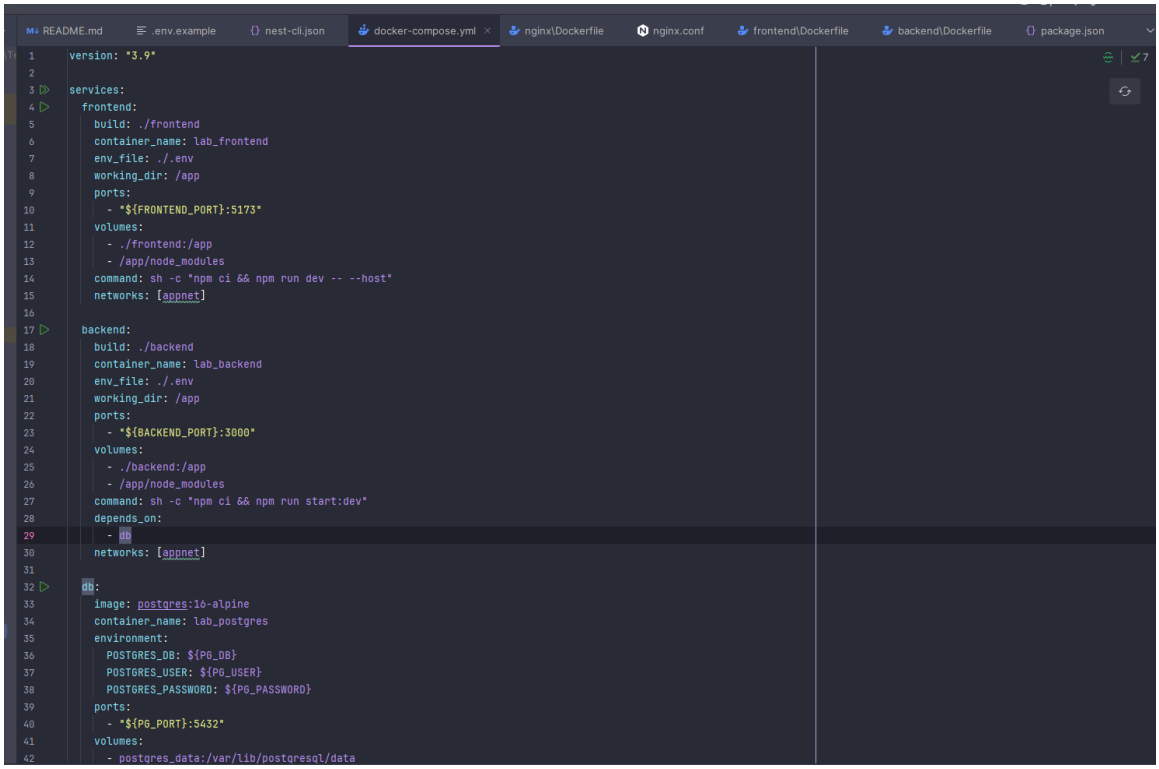
1. Estrutura de Diretórios do Projeto

Foi criada a seguinte estrutura de diretórios no repositório 'fullstack-lab':
- frontend/: aplicação React com Vite
- backend/: aplicação NestJS
- nginx/: configuração do Nginx (Dockerfile e nginx.conf)
- docker-compose.yml e arquivos .env/.env.example na raiz



2. Arquivo docker-compose.yml

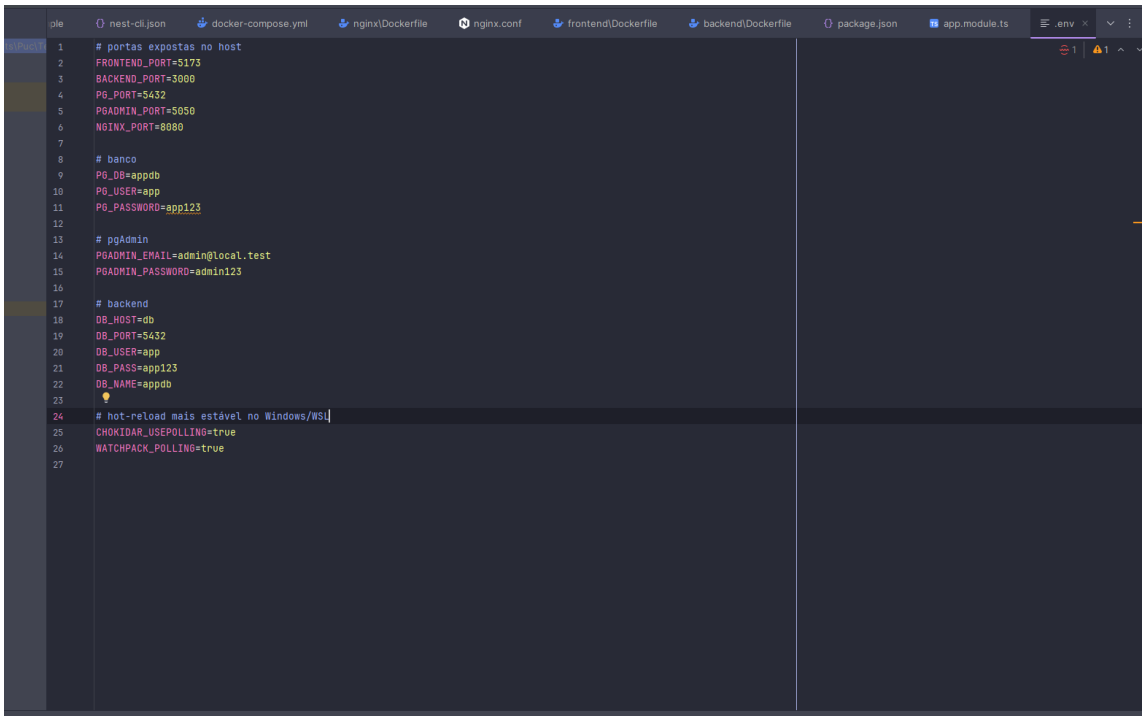
O arquivo docker-compose.yml orquestra todos os serviços: frontend, backend, banco de dados, pgAdmin e Nginx. Cada serviço utiliza volumes e variáveis de ambiente conforme definido no arquivo .env.



```yaml
version: "3.9"

services:
  frontend:
    build: ./frontend
    container_name: lab_frontend
    env_file: ./.env
    working_dir: /app
    ports:
      - "${FRONTEND_PORT}:5173"
    volumes:
      - ./frontend:/app
      - /app/node_modules
    command: sh -c "npm ci && npm run dev -- --host"
    networks: [appnet]

  backend:
    build: ./backend
    container_name: lab_backend
    env_file: ./.env
    working_dir: /app
    ports:
      - "${BACKEND_PORT}:3000"
    volumes:
      - ./backend:/app
      - /app/node_modules
    command: sh -c "npm ci && npm run start:dev"
    depends_on:
      - db
    networks: [appnet]

  db:
    image: postgres:16-alpine
    container_name: lab_postgres
    environment:
      POSTGRES_DB: ${PG_DB}
      POSTGRES_USER: ${PG_USER}
      POSTGRES_PASSWORD: ${PG_PASSWORD}
    ports:
      - "${PG_PORT}:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data
```

3. Variáveis de Ambiente (.env)

As variáveis de ambiente foram centralizadas no arquivo .env, incluindo portas expostas, credenciais do banco, credenciais do pgAdmin e parâmetros do backend.

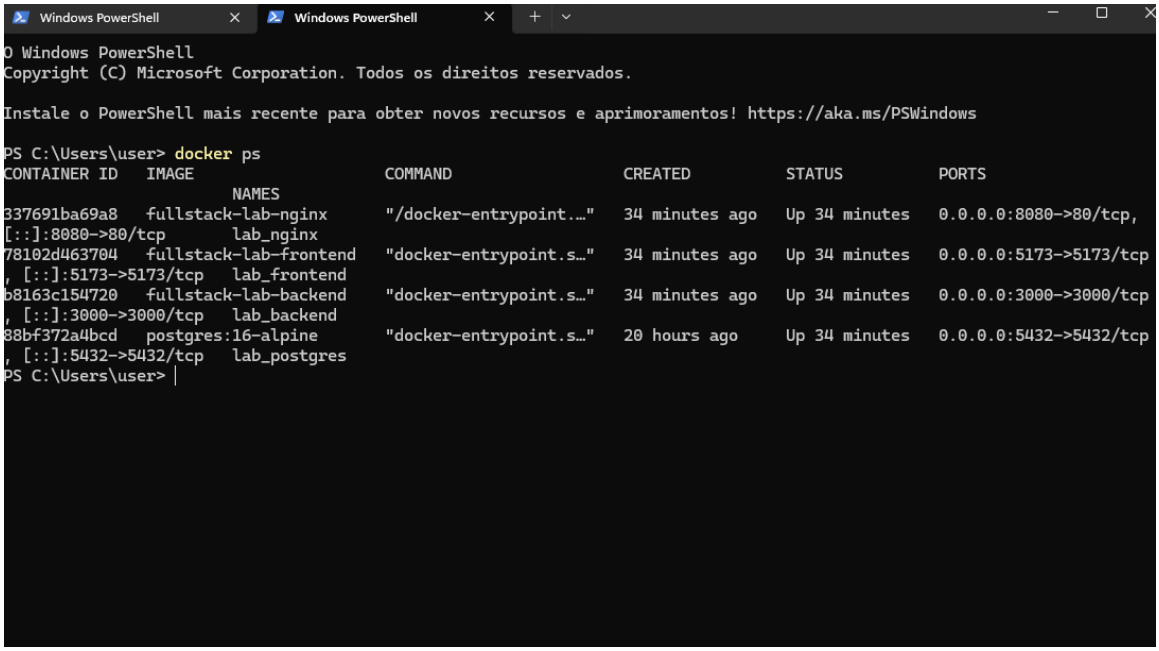


```
# portas expostas no host
FRONTEND_PORT=5173
BACKEND_PORT=3000
PG_PORT=5432
PGADMIN_PORT=5050
NGINX_PORT=8080

# banco
PG_DB=appdb
PG_USER=app
PG_PASSWORD=app123

# pgAdmin
PGADMIN_EMAIL=admin@local.test
PGADMIN_PASSWORD=admin123

# backend
DB_HOST=db
DB_PORT=5432
DB_USER=app
DB_PASS=app123
DB_NAME=appdb

# hot-reload mais estável no Windows/WSL
CHOKIDAR_USEPOLLING=true
WATCHPACK_POLLING=true
```

4. Build e Execução dos Containers

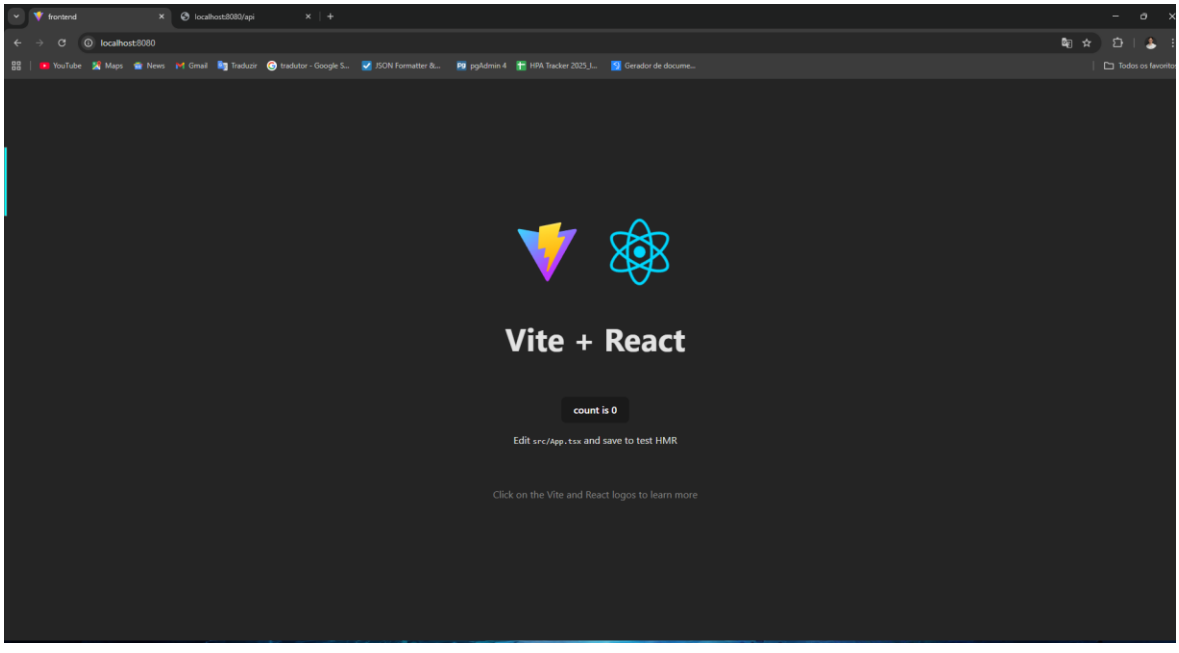
Os containers foram construídos e inicializados utilizando os comandos:
docker compose build --no-cache
docker compose up

Todos os serviços foram corretamente inicializados e podem ser verificados com o comando 'docker compose ps'.



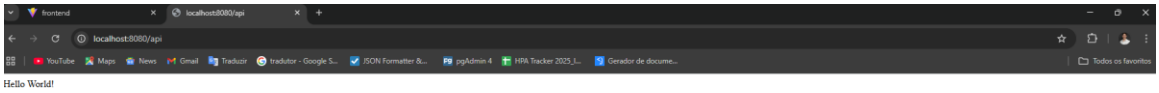
5. Frontend (React + Vite)

O frontend foi implementado em React com Vite. O serviço está disponível através do proxy reverso Nginx na URL:
http://localhost:8080/



6. Backend (NestJS)

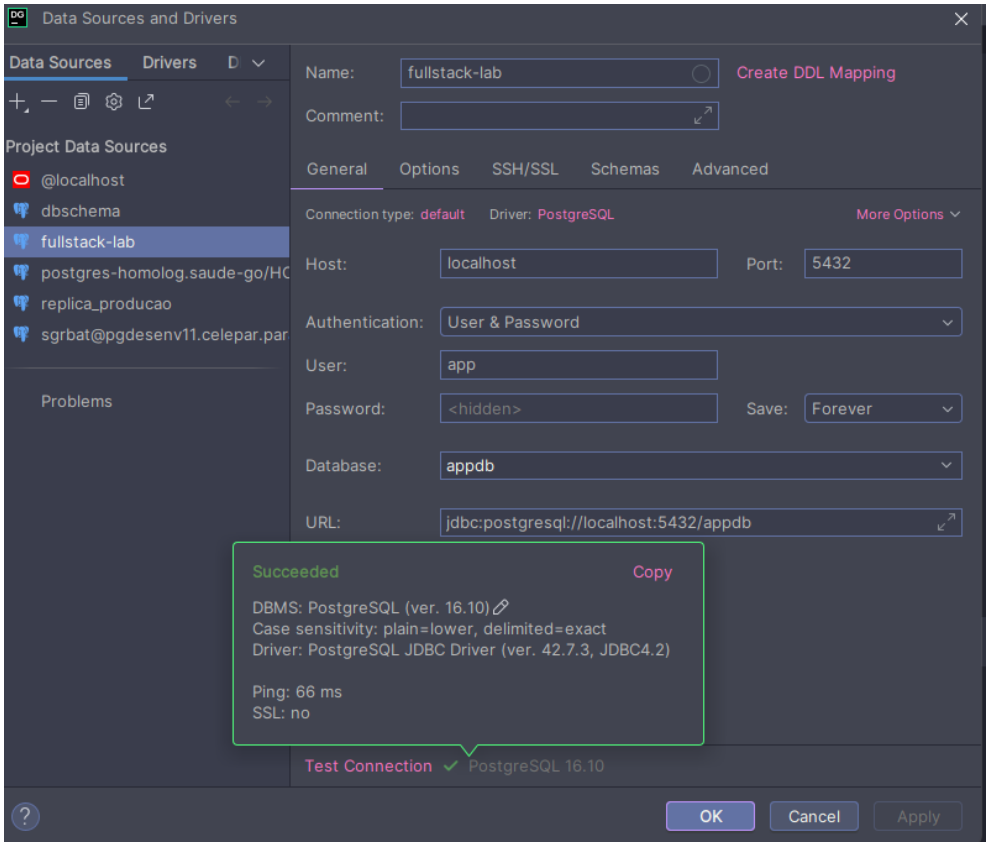
O backend foi desenvolvido em NestJS e exposto pelo Nginx utilizando o prefixo '/api'. As requisições realizadas em http://localhost:8080/api são corretamente encaminhadas para o serviço backend.



7. Banco de Dados (PostgreSQL)

O banco de dados PostgreSQL foi configurado para utilizar volume persistente, garantindo que os dados sejam mantidos mesmo após reinicialização dos containers. A conexão foi validada utilizando o DataGrip:

- Host: localhost
- Porta: 5432
- Usuário: app
- Senha: app123
- Database: appdb



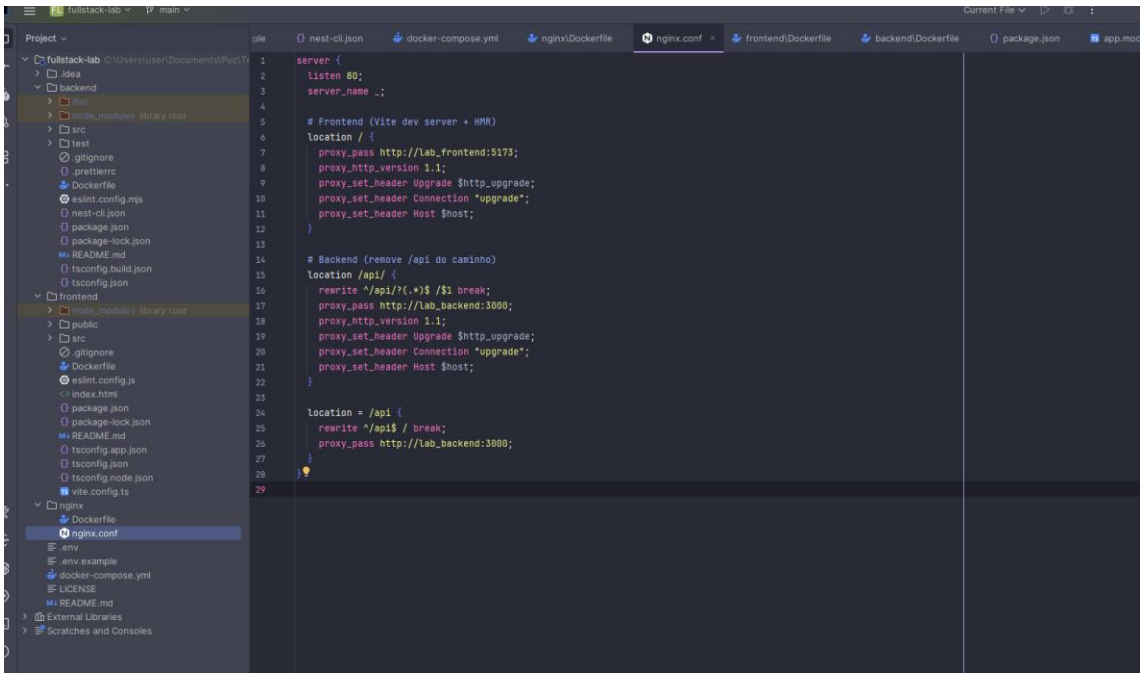
8. pgAdmin

O serviço pgAdmin também foi configurado e está acessível em http://localhost:5050/. Entretanto, a evidência principal será feita com o DataGrip.

9. Proxy Reverso (Nginx)

O Nginx foi configurado como proxy reverso para encaminhar as requisições:

- '/' → frontend (React)
- '/api' → backend (NestJS), removendo o prefixo /api do caminho.



10. Conclusão

Com isso, todos os critérios de avaliação foram atendidos:
- O ambiente completo é inicializado com um único comando.
- Hot reload funciona para frontend e backend.
- Todos os serviços estão acessíveis em suas respectivas portas.
- Regras de roteamento do Nginx funcionam conforme especificado.
- Dados do PostgreSQL persistem entre reinicializações.

As evidências acima comprovam o funcionamento do ambiente full-stack com Docker Compose.