🚀Projet FastAPI avec PostgreSQL et pgAdmin (Docker)

Bienvenue dans ce projet FastAPI déployé avec Docker ⚡!

Ce guide est conçu pour les **débutants** ❄️qui veulent démarrer rapidement un backend FastAPI avec une base PostgreSQL et une interface graphique pgAdmin pour manipuler la base de données facilement.

Prérequis

- Docker installé sur ta machine
- Un terminal (Command Prompt, Terminal macOS ou Linux, ou VS Code Terminal)

Structure du projet

```
mon_api_fastapi/
├── app/
│   ├── main.py
│   ├── models.py
│   ├── crud.py
│   ├── schemas.py
│   └── ...
├── tests/
│   └── test_main.py
├── Dockerfile
├── docker-compose.yml
├── requirements.txt
└── README.md
```

🏨Lancer le projet

Dans un terminal, place-toi dans le dossier du projet, puis exécute :

```
docker-compose up --build
```

Cela va :

- Construire l'image Docker de ton app FastAPI
- Démarrer PostgreSQL avec une base appelée `mydb`

• Démarrer pgAdmin pour voir la base en mode graphique

Accéder aux services

- 🦟 **FastAPI (docs Swagger)** : http://localhost:8000/docs
- 💐 **pgAdmin** : http://localhost:5050

Connexion à pgAdmin

- Email : `admin@admin.com`
- Mot de passe : `admin`

Ajouter un nouveau serveur dans pgAdmin

1. Clique droit sur "Servers" > *Create* > *Server...*
2. Onglet **General** :
3. Name : `Postgres Local`
4. Onglet **Connection** :
5. Host name/address : `db`
6. Username : `postgres`
7. Password : `postgres`

Contenu du fichier `docker-compose.yml`

```
version: "3.9"

services:
  fastapi_app:
    build: .
    container_name: fastapi_app
    depends_on:
      - db
    volumes:
      - .:/app
    ports:
      - "8000:8000"
    environment:
      - DATABASE_URL=postgresql://postgres:postgres@db:5432/mydb

  db:
    image: postgres:15
    container_name: fastapi_db
    restart: always
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
      POSTGRES_DB: mydb
    ports:
```

```
      - "5432:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data

  pgadmin:
    image: dpage/pgadmin4
    container_name: pgadmin
    restart: always
    environment:
      PGADMIN_DEFAULT_EMAIL: admin@admin.com
      PGADMIN_DEFAULT_PASSWORD: admin
    ports:
      - "5050:80"
    depends_on:
      - db

volumes:
  postgres_data:
```

🚀 Fichier `Dockerfile`

```
FROM python:3.11-slim

WORKDIR /app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY . .

CMD ["uvicorn", "app.main:app", "--host", "0.0.0.0", "--port", "8000", "--
reload"]
```

🧪 Tester l'API avec Pytest

Tu peux lancer les tests avec :

```
docker exec -it fastapi_app pytest
```

Ou créer un script personnalisé `run_tests.sh` pour tout automatiser (voir plus bas).

🏥 Script `run_tests.sh` (optionnel)

Crée un fichier `run_tests.sh` pour lancer les tests et enregistrer les logs :

```
#!/bin/bash
DATE=$(date +"%Y-%m-%d_%H-%M-%S")
LOG_FILE="logs/test_$DATE.log"
echo "🕊️ Log enregistré dans : $LOG_FILE"

STATUS=$(docker inspect -f '{{.State.Status}}' fastapi_app 2>/dev/null)
if [ "$STATUS" == "exited" ]; then
  echo "   Redémarrage du conteneur..."
  docker start fastapi_app
elif [ "$STATUS" != "running" ]; then
  echo "🦗 Le conteneur n'est pas lancé. Lancez: docker-compose up -d"
  exit 1
fi

sleep 3
echo "🍁 Lancement des tests..."
docker exec -it fastapi_app pytest | tee "$LOG_FILE"
echo "🦜 Tests terminés."
```

Rends le script exécutable :

```
chmod +x run_tests.sh
```

Puis exécute :

```
./run_tests.sh
```

🌐 Conseils bonus pour les débutants

- Pour arrêter les conteneurs :

  ```
  docker-compose down
  ```

- Pour relancer proprement :

  ```
  docker-compose down -v && docker-compose up --build
  ```

- Pour ouvrir un terminal dans le conteneur :

```
docker exec -it fastapi_app /bin/bash
```

🌳Tu es prêt à développer une API moderne, rapide et testable ✨!

N'hésite pas à consulter la doc officielle : https://fastapi.tiangolo.com/

Créé avec ❤️pour les développeurs qui démarrent avec FastAPI.