Rapport de Projet : Application Full Stack de Gestion des Utilisateurs

Présentation générale du projet

Ce projet est une application Full Stack composée d'un frontend en React, d'un backend en Express.js, et d'une base de données MySQL. Il permet de gérer une liste d'utilisateurs avec des fonctionnalités CRUD (Créer, Lire, Mettre à jour, Supprimer). Le projet a été entièrement conteneurisé avec Docker et déployé via GitHub Actions, offrant ainsi une solution DevOps complète avec CI/CD.

Étapes de mise en place du backend et frontend

 Backend (Express.js)

- Initialisation d'un projet Node.js avec Express.

- Création d'API RESTful pour gérer les utilisateurs : GET, POST, PUT, DELETE.

- Connexion à MySQL avec mysql2.

- Utilisation de middlewares pour erreurs et validation.

- Configuration via .env.

 Frontend (React)

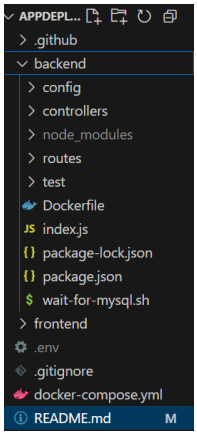
- Interface utilisateur avec React.

- Composants : UserList, UserForm.

- Intégration API avec axios.

- Notifications via react-toastify.

- Stylisation avec Bootstrap 5.



Explication de la base de données

- Base de données : MySQL

schema :

CREATE TABLE USERS (

ID INT AUTO_INCREMENT PRIMARY KEY,

NAME VARCHAR(255) NOT NULL,

EMAIL VARCHAR(255) NOT NULL

);

configuration :

- conteneur docker officiel mysql : 8.0
- variables d'environnement :

environnement :

mysql_root_password : rootpassword

mysql_database : users_db

- persistance via volume docker

sécurité :

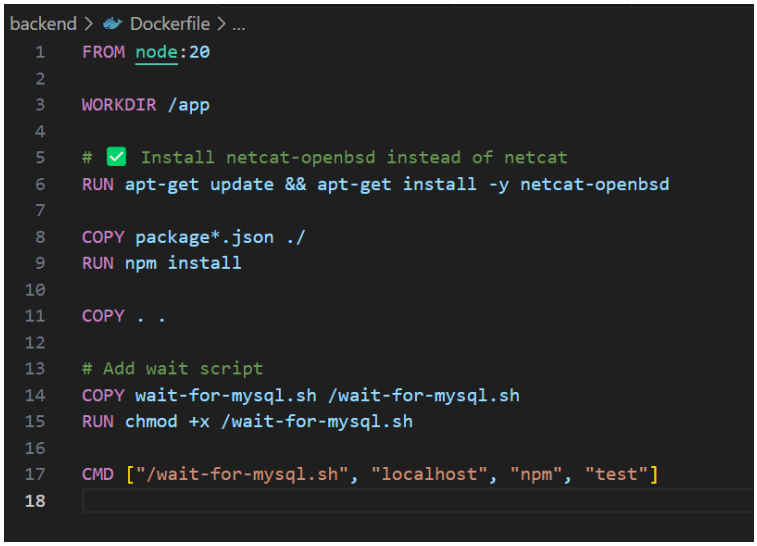
- utilisateur dédité

mot de passe chiffré

Dockerisation : étapes et choix faits

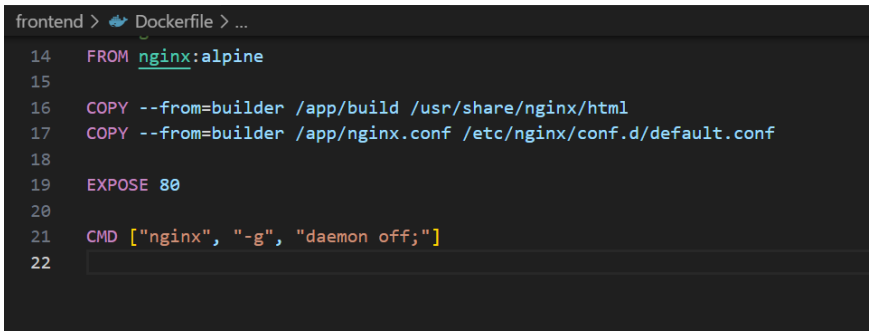
 Backend :

- Dockerfile basé sur Node, copie du code et lancement avec npm start.



 Frontend :

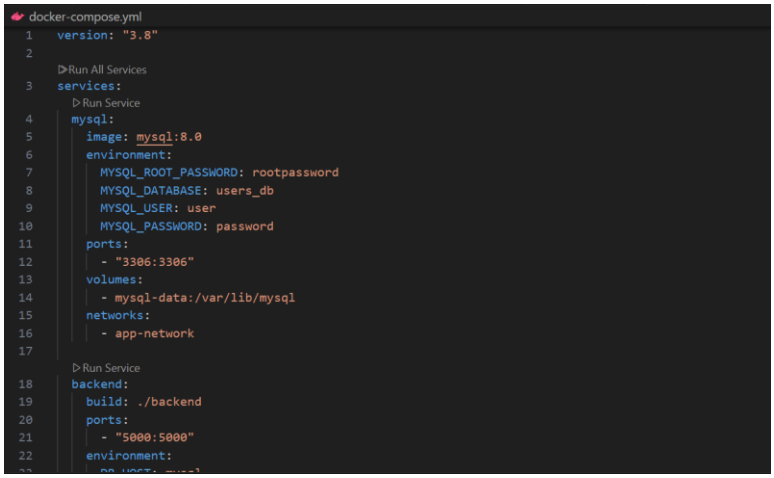
- Dockerfile avec build React + serve statique avec 'serve'.

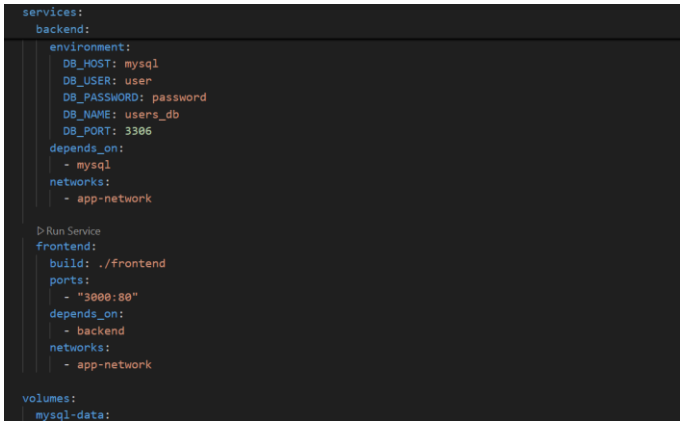


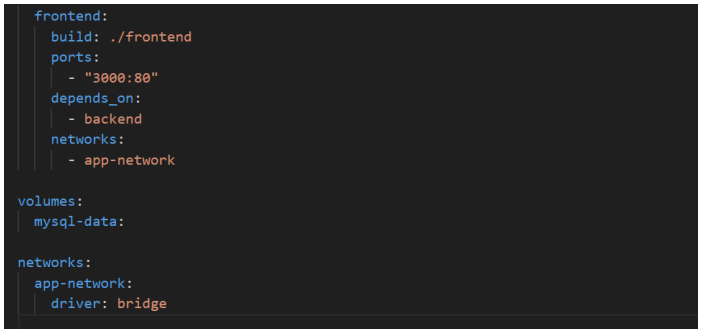
 Docker Compose :

- Services : frontend, backend, mysql, phpmyadmin.

- Configuration des réseaux et des volumes pour persistance.







GitHub Actions : pipeline expliqué

étape clés :

1. Déclenchement : push sur main
2. build-backend : install & test
3. build-frontend : install & build
4. Docker build & push vers Docker Hub
5. Déploiement via SSH + docker pull + docker-compose up -d

WORKFLOW:

NAME: CI/CD PIPELINE
ON: [PUSH]

JOBS:
  TEST:
    RUNS-ON: UBUNTU-LATEST
    STEPS:
      - USES: ACTIONS/CHECKOUT@V3
      - RUN: NPM TEST
  DEPLOY:
    NEEDS: TEST
    STEPS:
      - USES: DOCKER/LOGIN-ACTION@V2
      - RUN: DOCKER BUILD -T MONIMAGE

Difficultés rencontrées et solutions

Problème : Connexion backend-MySQL refusée ➤Solution : delay de démarrage ou dépendance docker-compose

Problème : Dépendances manquantes GitHub Actions ➤Solution : npm ci et cache

Problème : Build React échoué ➤Solution : installer 'serve'

Problème : Erreurs CORS ➤Solution : middleware cors()

Conclusion et axes d'amélioration

✅Projet fonctionnel, CI/CD intégré

 Améliorations futures :

- Authentification JWT

- Tests automatisés

- Système de rôles

- Monitoring avec Grafana + Prometheus