

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.

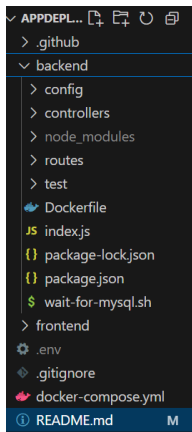
Étapes de mise en place du backend et frontend

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.
- Stylistation 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 dédié

mot de passe chiffré

Dockerisation : étapes et choix faits

Backend :

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

```
backend > Dockerfile > ...
1  FROM node:20
2
3  WORKDIR /app
4
5  # ✅ Install netcat-openbsd instead of netcat
6  RUN apt-get update && apt-get install -y netcat-openbsd
7
8  COPY package*.json ./
9  RUN npm install
10
11 COPY . .
12
13 # Add wait script
14 COPY wait-for-mysql.sh /wait-for-mysql.sh
15 RUN chmod +x /wait-for-mysql.sh
16
17 CMD ["/wait-for-mysql.sh", "localhost", "npm", "test"]
18
```

Frontend :

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

```
frontend > Dockerfile > ...
14 FROM nginx:alpine
15
16 COPY --from=builder /app/build /usr/share/nginx/html
17 COPY --from=builder /app/nginx.conf /etc/nginx/conf.d/default.conf
18
19 EXPOSE 80
20
21 CMD ["nginx", "-g", "daemon off;"]
22
```

Docker Compose :

- Services : frontend, backend, mysql, phpmyadmin.
- Configuration des réseaux et des volumes pour persistance.

```

1  version: "3.8"
2
3  services:
4    mysql:
5      image: mysql:8.0
6      environment:
7        MYSQL_ROOT_PASSWORD: rootpassword
8        MYSQL_DATABASE: users_db
9        MYSQL_USER: user
10       MYSQL_PASSWORD: password
11      ports:
12        - "3306:3306"
13      volumes:
14        - mysql-data:/var/lib/mysql
15      networks:
16        - app-network
17
18    backend:
19      build: ./backend
20      ports:
21        - "5000:5000"
22      environment:

```

```

services:
  backend:
    environment:
      DB_HOST: mysql
      DB_USER: user
      DB_PASSWORD: password
      DB_NAME: users_db
      DB_PORT: 3306
    depends_on:
      - mysql
    networks:
      - app-network

  frontend:
    build: ./frontend
    ports:
      - "3000:80"
    depends_on:
      - backend
    networks:
      - app-network

volumes:
  mysql-data:

```

```

frontend:
  build: ./frontend
  ports:
    - "3000:80"
  depends_on:
    - backend
  networks:
    - app-network

volumes:
  mysql-data:

networks:
  app-network:
    driver: bridge

```

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