RAPPORT TECHNIQUE

DÉPLOIEMENT D'APPLICATION
FULLSTACK AVEC CI/CD

ÉLABORÉ PAR : LAAOUAN ISMAIL

ANNÉE UNIVERSITAIRE : 2024 / 2025

ENCADRÉ PAR : MR M O H A M E D E L F A R O U K I

1. PRÉSENTATION GÉNÉRALE

OBJECTIF :

Réalisation d'une application Fullstack (React + Express + MySQL) avec :

- Conteneurisation via Docker
- Pipeline CI/CD automatisé avec GitHub Actions
- Tests unitaires automatisés

FONCTIONNALITÉS CLÉS :

- CRUD complet des utilisateurs
- Interface utilisateur en ReactJS responsive
- API RESTful avec gestion des erreurs
- Tests Mocha/Chai (backend)
- Déploiement local avec Docker Compose

2. MISE EN PLACE TECHNIQUE

2.1 BACKEND (NODE.JS/EXPRESS)

Architecture :

server.js → Routes intégrées → Gestion base de données MySQL

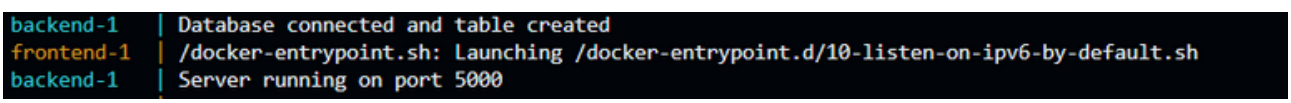
Endpoints implémentés :

- **GET /api/users** : Récupère tous les utilisateurs
- **POST /api/users** : Ajoute un utilisateur
- **PUT /api/users/:id** : Met à jour un utilisateur
- **DELETE /api/users/:id** : Supprime un utilisateur

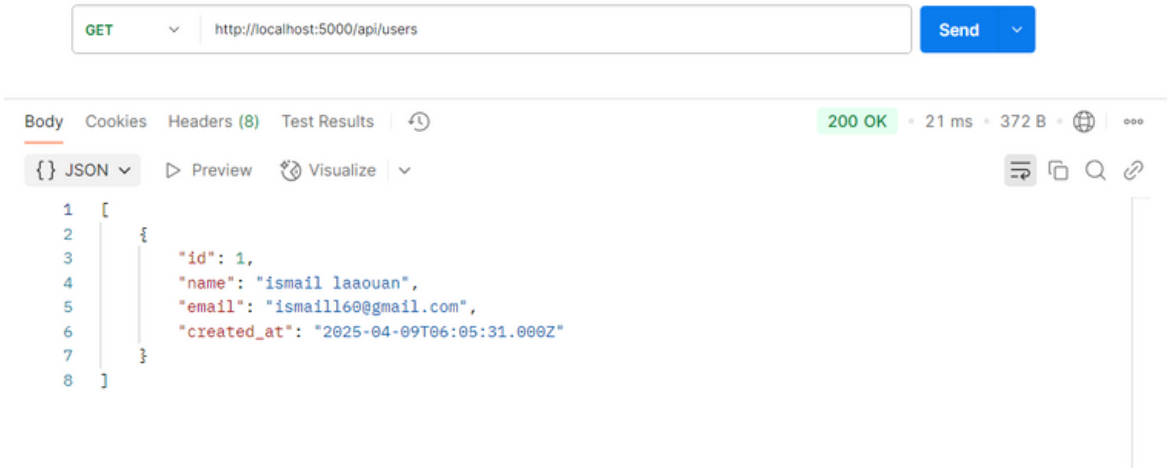
Technos :

- Express.js
- MySQL (mysql2 + pool)
- body-parser, dotenv, cors

Le terminal avec le backend lancé en utilisant docker-compose up --build :



Le résultat de la requête dans Postman :

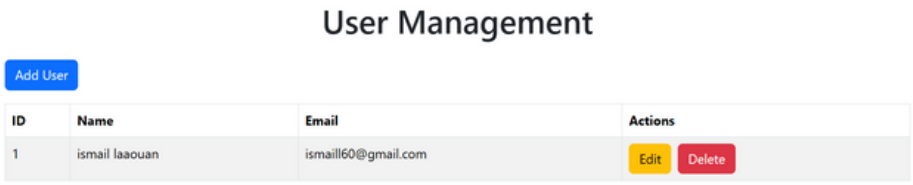


2.2 FRONTEND (REACT)

Structure de l'application :

- Components : UserList, UserForm, etc.
- Services : api.js pour les requêtes HTTP (via Axios)
- Gestion d'état : Utilisation simple via useState ou éventuellement Context API si besoin d'état global
- UI : Bootstrap 5 + React-Bootstrap
- Notifications : react-toastify

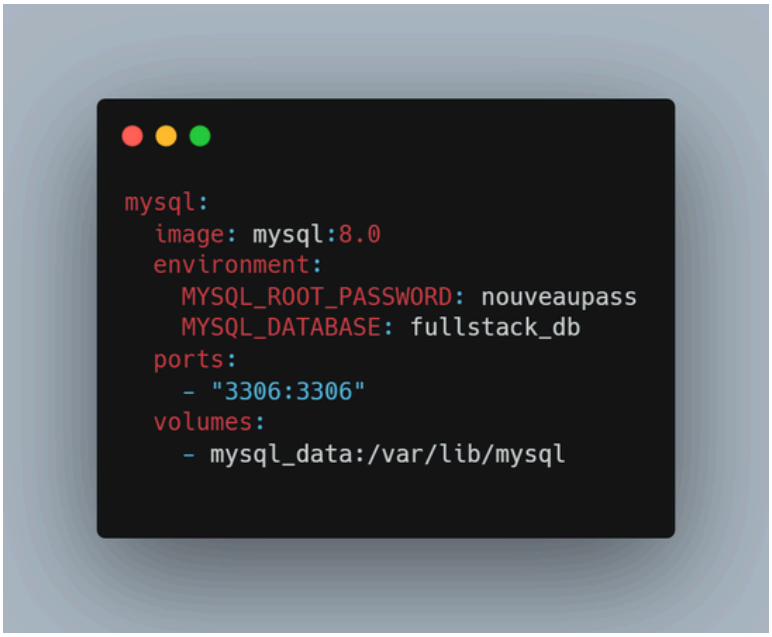
La page d'accueil de l' interface React



3- BASE DE DONNÉES MYSQL

Configuration :

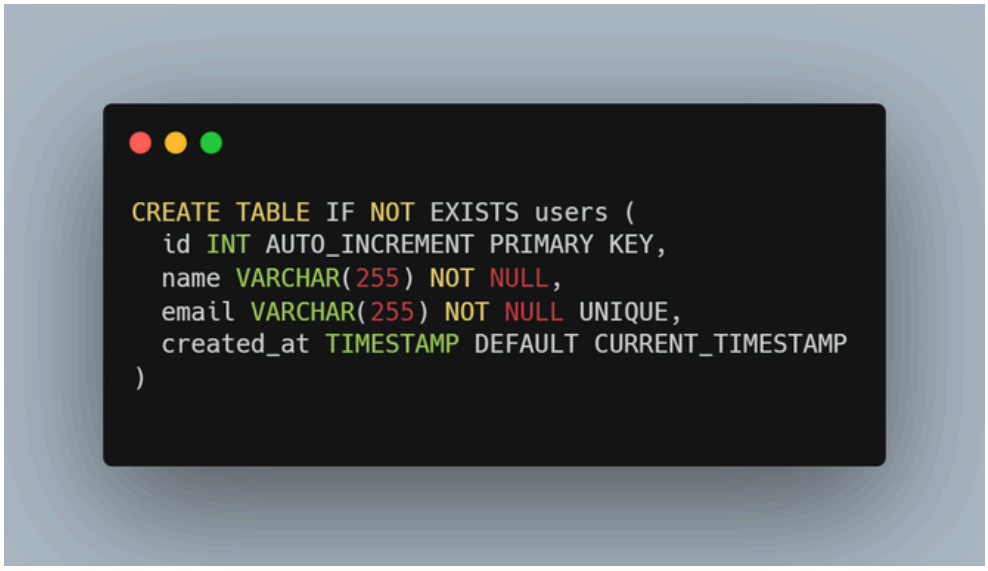
La base de données MySQL est lancée automatiquement avec Docker via le service suivant dans docker-compose.yml :



```yaml
mysql:
  image: mysql:8.0
  environment:
    MYSQL_ROOT_PASSWORD: nouveaupass
    MYSQL_DATABASE: fullstack_db
  ports:
    - "3306:3306"
  volumes:
    - mysql_data:/var/lib/mysql
```

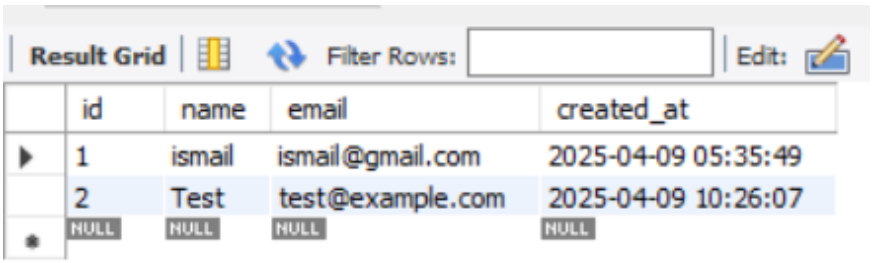
Table utilisée :

La table users est automatiquement créée au lancement grâce au script dans server.js :



```sql
CREATE TABLE IF NOT EXISTS users (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  email VARCHAR(255) NOT NULL UNIQUE,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
)
```

Une vue de la structure de la table users



Result Grid | Filter Rows: | Edit:

id	name	email	created_at
1	ismail	ismail@gmail.com	2025-04-09 05:35:49
2	Test	test@example.com	2025-04-09 10:26:07
NULL	NULL	NULL	NULL

4. DOCKERISATION

4.1 Configuration

Le projet utilise Docker pour conteneuriser à la fois :

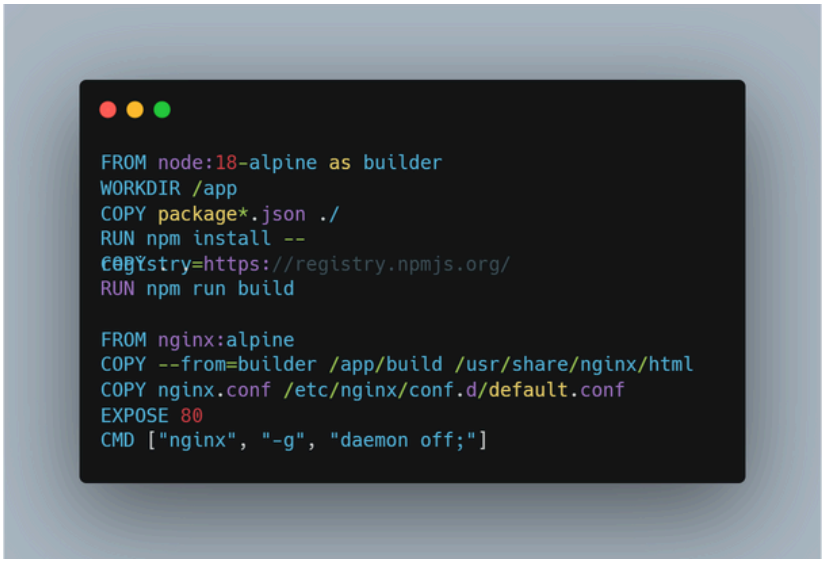
- Le backend (Node.js/Express)
- Le frontend (React)
- La base de données MySQL

Fichiers clés :

✅ Dockerfile (Backend) :

```
FROM node:18-alpine

WORKDIR /app

# Install wait-for-it pour gérer les dépendances entre conteneurs
RUN apk add --no-cache bash
ADD https://github.com/vishnubob/wait-for-it/raw/master/wait-for-it.sh /wait-for-
RUN chmod +x /wait-for-it.sh

COPY package*.json ./
RUN npm install

COPY . .

EXPOSE 5000

# Commande modifiée pour attendre que MySQL soit prêt
CMD ["/bin/sh", "-c", "/wait-for-it.sh mysql:3306 --timeout=30 -- node server.js"]
```

✅ Dockerfile (Frontend) :



```
FROM node:18-alpine as builder
WORKDIR /app
COPY package*.json ./
RUN npm install --
COPY . .
RUN npm run build

FROM nginx:alpine
COPY --from=builder /app/build /usr/share/nginx/html
COPY nginx.conf /etc/nginx/conf.d/default.conf
EXPOSE 80
CMD ["nginx", "-g", "daemon off;"]
```

✅ docker-compose.yml – Orchestration :

4.2 Optimisations

- Utilisation d'images multi-stage (frontend) pour réduire la taille
- Déclaration d'un volume persistant pour MySQL
- Configuration réseau simplifiée via docker-compose

La liste des conteneurs avec docker ps

```
C:\fullstack-app\backend>docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED         STATUS                   PORTS                              NAMES
a81ae96214e5   fullstack-app-frontend   "/docker-entrypoint.…"   32 minutes ago   Up 9 seconds             0.0.0.0:3000->80/tcp               fullstack-app-fronte
nd-1
e23d3999f438   fullstack-app-backend    "docker-entrypoint.s…"   32 minutes ago   Up 9 seconds             0.0.0.0:5000->5000/tcp             fullstack-app-backen
d-1
f771b7439909   mysql:8.0              "docker-entrypoint.s…"   34 hours ago     Up 15 seconds (healthy)  0.0.0.0:3306->3306/tcp, 33060/tcp  fullstack-app-mysql-
1
```

5. INTÉGRATION CONTINUE AVEC GITHUB ACTIONS

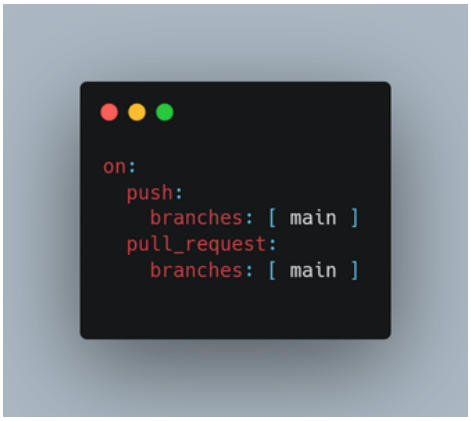
Objectif :

Mettre en place une pipeline CI/CD qui automatise :

- Les tests backend
- Le build des images Docker
- Le push vers Docker Hub

Fichier utilisé : .github/workflows/ci.yml

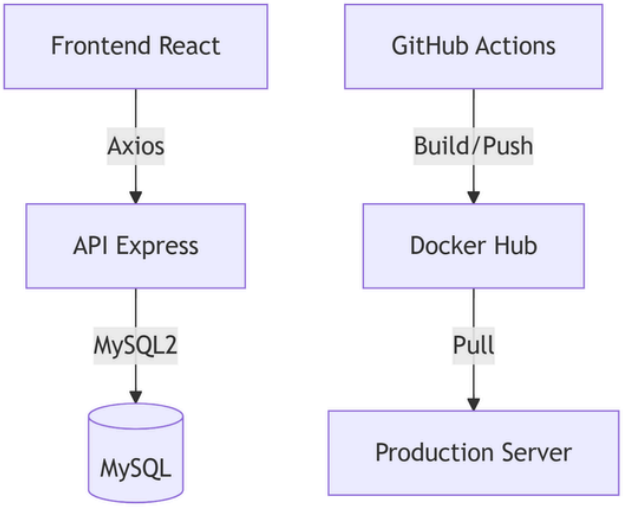
Déclencheurs :



Étapes de la pipeline :

- 📦 Installation de Node.js & des dépendances
- 🧪 Exécution des tests Mocha/Chai (npm test)
- 🐳 Build des images Docker (docker build)
- ☁️ Push sur Docker Hub

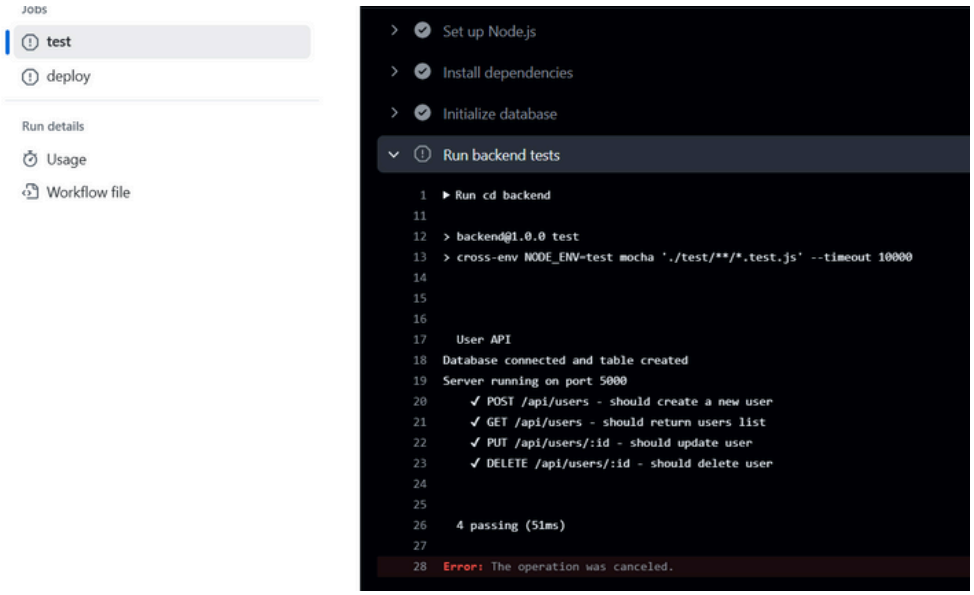
Schéma d'Architecture



Extrait Concis du Workflow CI/CD

```yaml
name: CI/CD Backend Pipeline

on:
  push:
    branches: [main, master]
  pull_request:
    branches: [main, master]

jobs:
  test:
    runs-on: ubuntu-latest

    services:
      mysql:
        image: mysql:8.0
        env:
          MYSQL_ROOT_PASSWORD: nouveaupass
          MYSQL_DATABASE: fullstack_db
        ports:
          - 3306:3306
        options: --health-cmd="mysqladmin ping" --health-interval=10s --health-timeout=5s --health-
retries=3

    steps:
      - name: Checkout repository
        uses: actions/checkout@v3

      - name: Set up Node.js
        uses: actions/setup-node@v3
        with:
          node-version: 20

      - name: Install dependencies
        run: |
          cd backend
          npm install
          npm install cross-env --save-dev
```

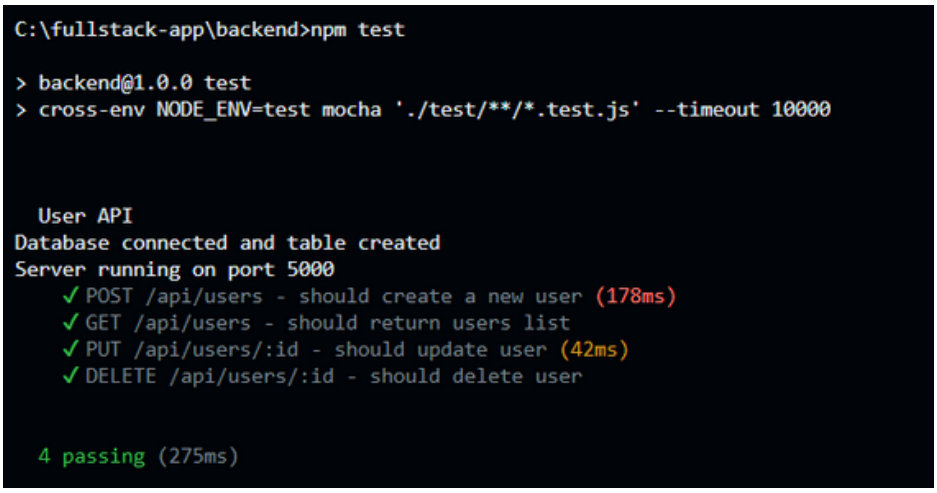
Une erreur "The operation was canceled" est apparue lors de l'exécution de la CI. Malgré cela, les tests sont tous passés (cf. image ci-dessus). Une piste probable serait un timeout GitHub Actions ou une interruption due au Docker service.



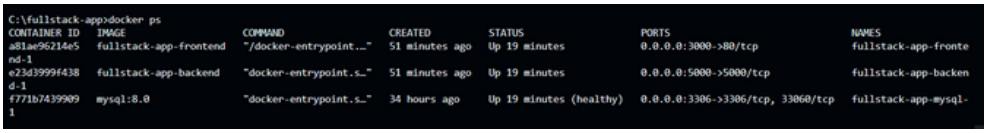
6. CAPTURES D'ÉCRAN

6.1 Tests Automatisés

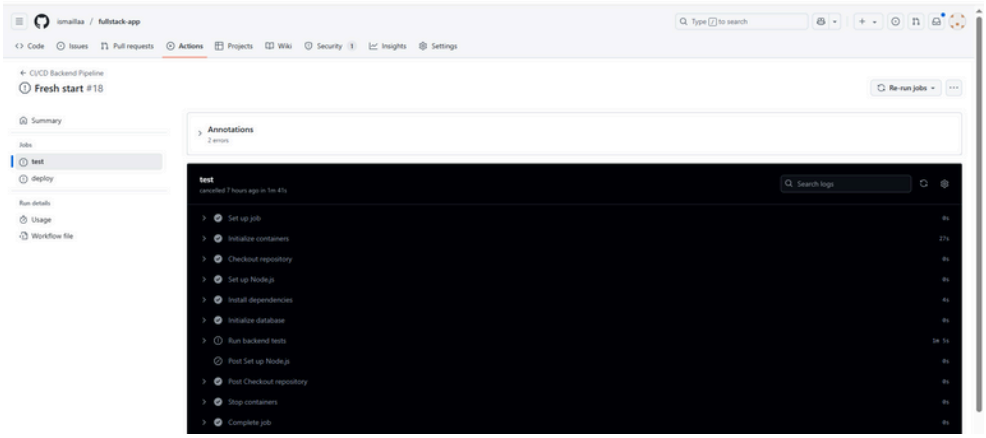
Capture montrant l'exécution des tests via Mocha/Chai
Résultats :



6.2 Conteneurs Docker



6.3 GitHub Actions



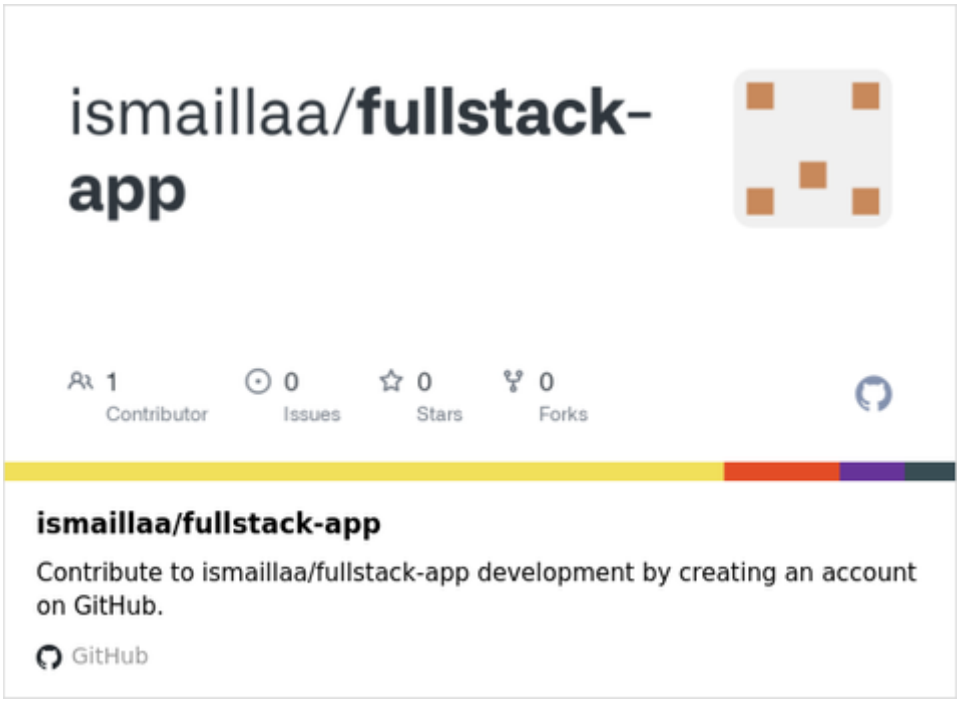
Comme je l'ai mentionné précédemment, l'exécution des actions GitHub s'est interrompue brutalement sans message d'erreur explicite. Contraint par le temps, je n'ai pas pu identifier la cause exacte du problème.

7. CONCLUSION ET PERSPECTIVESN

✅ Bilan du projet
L'objectif de créer une application fullstack conteneurisée a été atteint :
➤ Backend Express connecté à une base MySQL
➤ Frontend React responsive
➤ Pipeline CI/CD automatisée via GitHub Actions
➤ Tests automatisés validés avec Mocha/Chai
➤ Conteneurisation et orchestration maîtrisées via Docker & Docker Compose

🔍 Difficultés rencontrées
Une erreur "The operation was canceled" a été relevée lors de l'exécution du workflow CI sur GitHub Actions.
➤ Malgré cela, tous les tests sont passés localement, prouvant la stabilité du backend.
➤ La cause probable pourrait être un timeout ou une ressource non disponible dans l'environnement distant.

🚀 Perspectives d'amélioration
Ajout de tests End-to-End avec Cypress pour valider tout le parcours utilisateur
Intégration d'un déploiement automatique sur un VPS ou serveur cloud
Mise en place d'un système de logs centralisés + monitoring via Prometheus/Grafana
Sécurisation de l'API avec des tokens (JWT)
Authentification + autorisation pour différents niveaux d'utilisateurs



https://github.com/ismaillaa/fullstack-app