Rapport Technique - Application de Gestion d'Utilisateurs

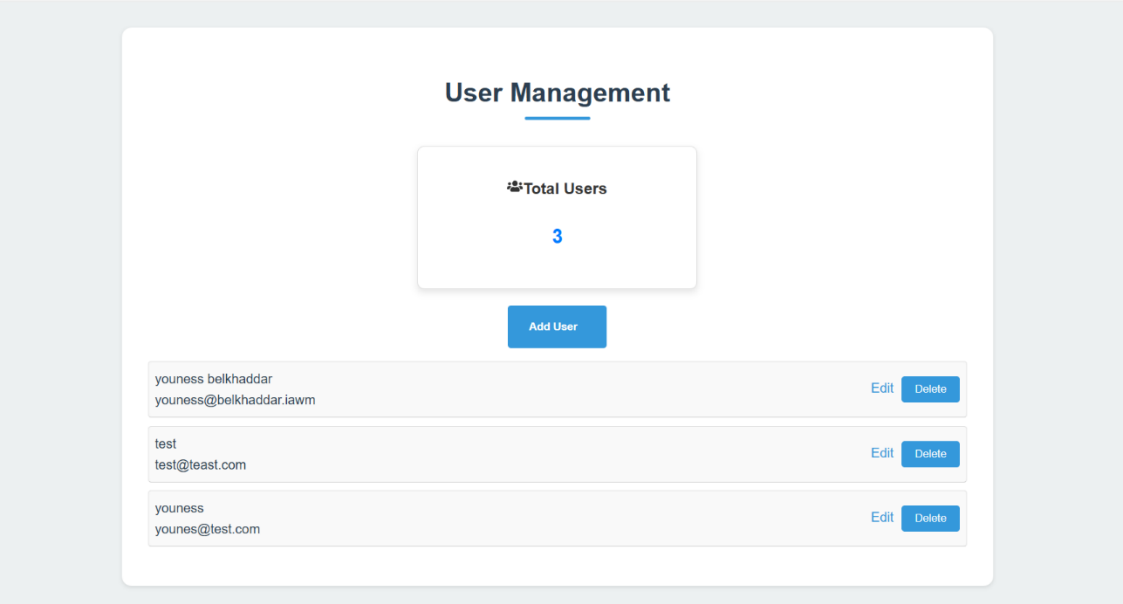


Table des matières

REALISÉ par : Youness Belkhaddar LICENCE **IAWM**

Présentation générale du projet

L'application de gestion d'utilisateurs est une solution web moderne permettant de gérer efficacement les utilisateurs d'une organisation. Elle offre une interface utilisateur intuitive et des fonctionnalités robustes pour la gestion des comptes utilisateurs.

Fonctionnalités principales

- Création et gestion des utilisateurs
- Interface utilisateur responsive et moderne
- API RESTful sécurisée
- Authentification des utilisateurs
- Tests automatisés
- Déploiement continu avec Docker

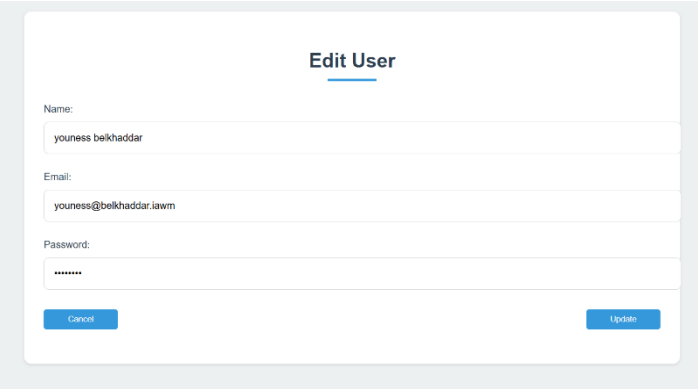


Figure 2 Add and Edit user

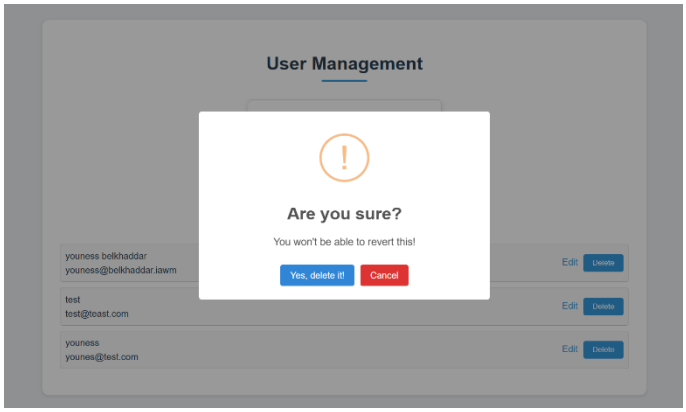


Figure 1 Delete user

Architecture technique

L'application suit une architecture moderne basée sur le stack MERN (MongoDB, Express.js, React, Node.js) avec une séparation claire entre le frontend et le backend.

Stack technologique

- **Frontend**: React.js avec Material-UI
- **Backend**: Node.js avec Express.js
- **Base de données**: MySQL
- **Tests**: Jest, Chai, Supertest

Conteneurisation: Docker
CI/CD: GitHub Actions

Mise en place du backend

Structure du backend

```
backend/
├── src/
│   ├── config/
│   ├── controllers/
│   ├── middleware/
│   ├── models/
│   ├── routes/
│   └── app.js
├── test/
└── package.json
```

Points clés de l'implémentation

1. Configuration de l'environnement avec dotenv
2. Mise en place des middlewares de sécurité
3. Implémentation des routes RESTful
4. Gestion des erreurs centralisée
5. Tests automatisés

Mise en place du frontend

Structure du frontend

```
frontend/
├── src/
│   ├── components/
│   ├── services/
│   ├── styles/
│   └── App.js
└── package.json
```

Caractéristiques principales

1. Interface utilisateur moderne avec Material-UI
2. Gestion d'état avec React Hooks
3. Appels API avec Axios
4. Styles modulaires avec CSS
5. Validation des formulaires

Base de données

Schéma de la base de données

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

Configuration

- Base de données: MySQL
- Port: 3306
- Nom de la base: users_management
- Utilisation de migrations pour la gestion du schéma

Dockerisation

Structure Docker

```
.
├── docker-compose.yml
├── backend/
│   └── Dockerfile
└── frontend/
    └── Dockerfile
```

Configuration

Docker

1. **Backend Dockerfile**
   - Image de base: Node.js
   - Installation des dépendances
   - Configuration de l'environnement

   Exposition du port 5000

2. **Frontend Dockerfile**
   - Image de base: Node.js
   - Build de l'application
   - Serveur nginx pour la production

   Exposition du port 80

3. **Docker Compose**
    - Orchestration des services
    - Configuration des réseaux
    - Gestion des volumes
    - Variables d'environnement

Pipeline CI/CD avec GitHub Actions

Étapes du pipeline

1. **Tests Backend**
    - Installation des dépendances
    - Exécution des tests unitaires
    - Vérification de la couverture
2. **Tests Frontend**
    - Installation des dépendances
    - Build de l'application
    - Tests des composants
3. **Build et Push Docker**
    - Construction des images
    - Push vers Docker Hub
    - Tagging des versions
4. **Déploiement**
    - Déploiement automatique
    - Vérification de la santé
    - Rollback en cas d'échec

Tests et qualité du code

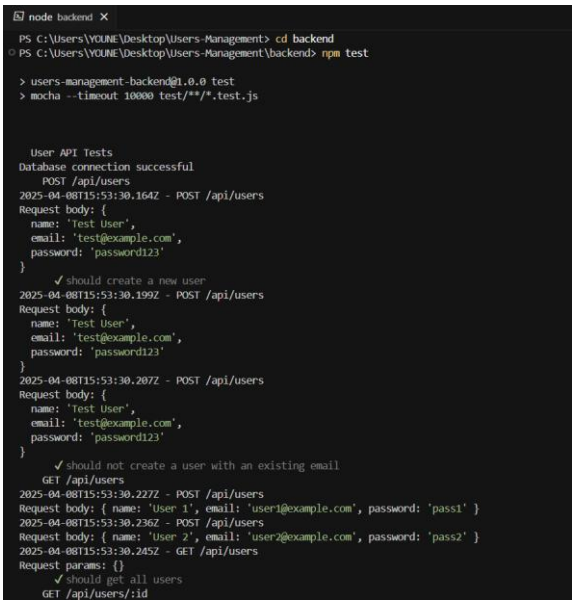
Stratégie de test

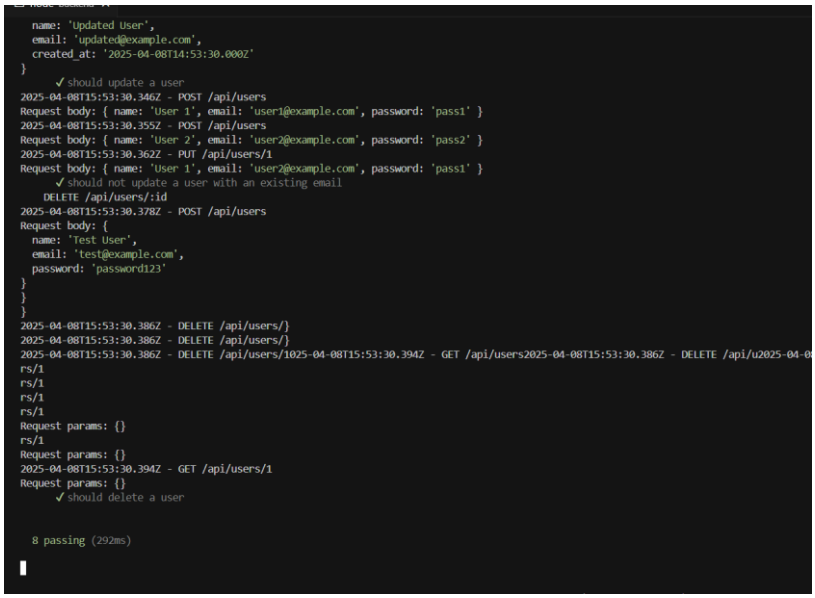
1. **Tests unitaires**
    - Contrôleurs
    - Services
    - Composants React
2. **Tests d'intégration**
    - API endpoints

- Flux utilisateur
- Base de données

3. **Tests de qualité**

- Linting (ESLint)
- Formatage (Prettier)
- Couverture de code





Difficultés rencontrées et solutions

1. Gestion des connexions à la base de données

Problème: Fuites de mémoire dans les tests **Solution:** Implémentation d'un pool de connexions et nettoyage approprié

2. Pipeline CI/CD

Problème: Tests qui ne se terminent pas **Solution:** Ajout de process.exit(0) après les tests

3. Dockerisation

Problème: Communication entre conteneurs **Solution:** Configuration correcte des réseaux Docker

REALISÉ par : Youness Belkhaddar LICENCE **IAWM**

Conclusion et axes d'amélioration

Points forts

- Architecture moderne et évolutive
- Tests automatisés robustes
- Pipeline CI/CD efficace
- Documentation claire

Axes d'amélioration

1. Performance
    - Mise en cache
    - Optimisation des requêtes
    - Compression des assets
2. **Sécurité**
    - Authentification JWT
    - Rate limiting
    - Validation des données
3. **Fonctionnalités**
    - Gestion des rôles
    - Journalisation avancée
    - Interface d'administration
4. **Monitoring**
    - Métriques de performance
    - Alertes
    - Logs centralisés