

**Développement Fullstack + CI/CD avec GitHub Actions**

**Réalisé par :** Nada Belachqer

**2024-2025**

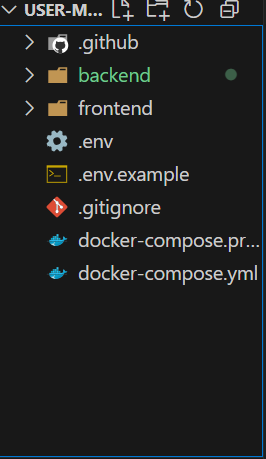
**1. Présentation Générale du Projet**

1. **Objectifs**

Ce projet vise à développer une application web fullstack permettant la gestion d'utilisateurs (CRUD) avec :

* Backend : API REST en Node.js/Express
* Frontend : Interface React avec Axios pour les requêtes HTTP
* Base de données : MySQL conteneurisée via Docker
* CI/CD : Automatisation des tests et déploiement avec GitHub Actions

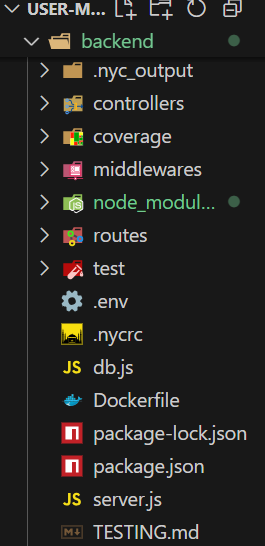
1. **Architecture Globale**

****

**Partie 1 : Développement Backend et Frontend**

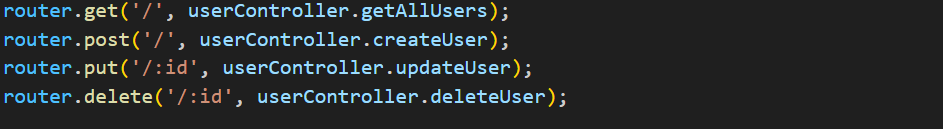
**1. Mise en Place du Backend (Express.js)**

* 1. **Structure des fichiers** :

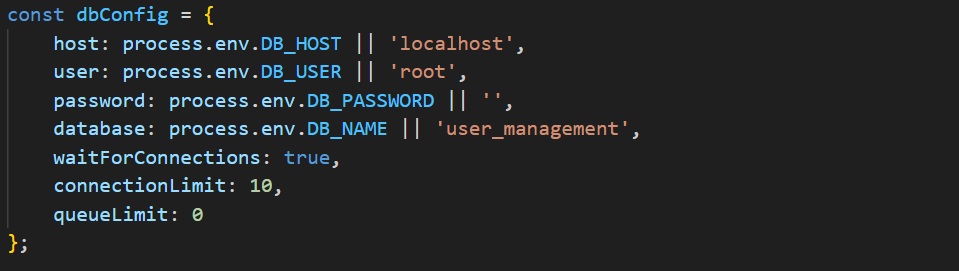


* 1. **Points Clés de l'Implémentation**

**a) Gestion des Routes**



**b) Connexion à MySQL**

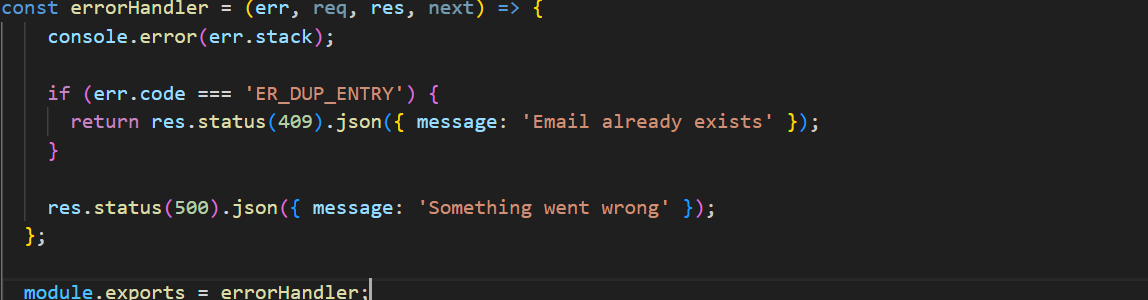


**Bonnes pratiques que j’ai adoptée** :

* Utilisation d'un pool de connexions
* Variables sensibles stockées dans .env

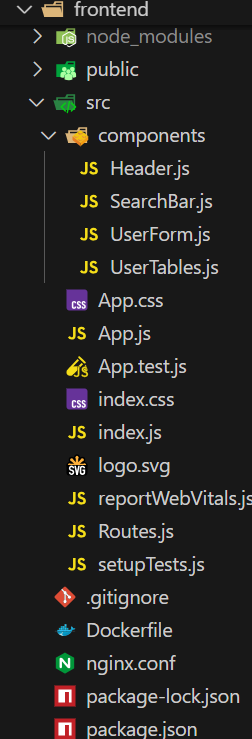
**c) Gestion des Erreurs**

Middleware dédié pour les erreurs :



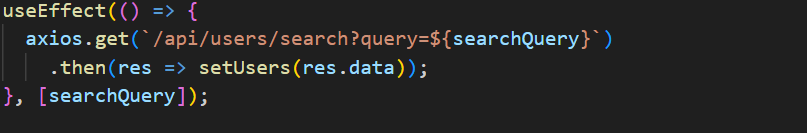
**2. Développement du Frontend (React.js)**

1. **Structure des Composants**



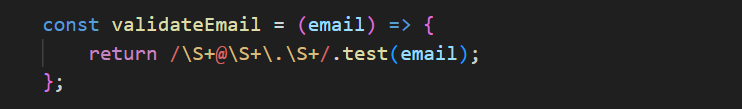
1. **Fonctionnalités Principales**

**a) Communication avec l'API**

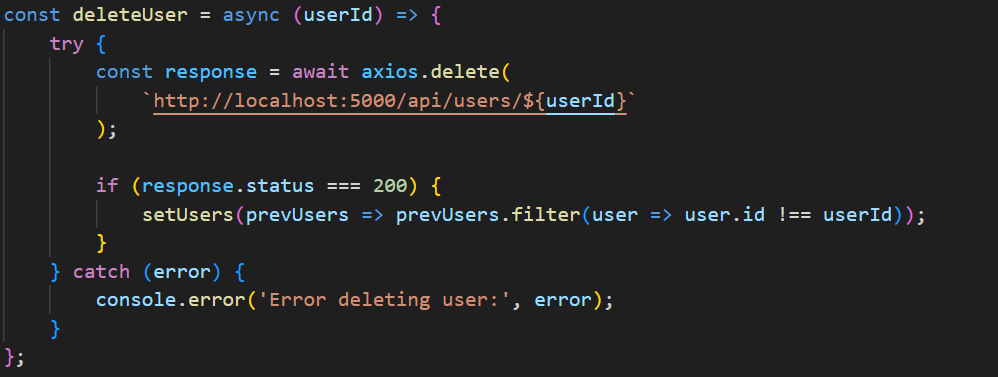


**b) Gestion des Formulaires**

* **Validation côté client** :



**c) Mise à Jour Optimiste**

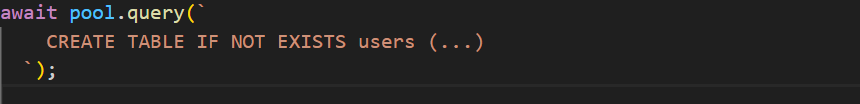


**3. Base de Données MySQL**

1. **Schéma de la Table**users



1. **Initialisation Automatisée**



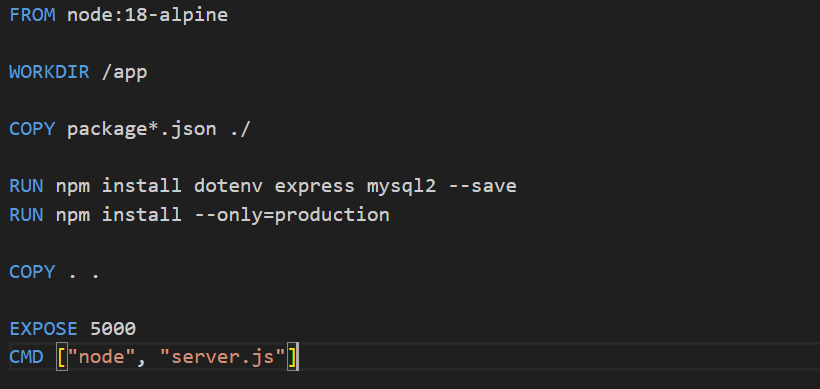
**Partie 2 : Dockerisation**

1. **Choix techniques** :

* **Multi-stage builds** pour réduire la taille des images
* **Nginx** pour servir le frontend (optimisation des assets)
* **Réseau Docker personnalisé** pour la communication inter-conteneurs

1. **Fichiers de Configuration**

**Backend (**Dockerfile**)**



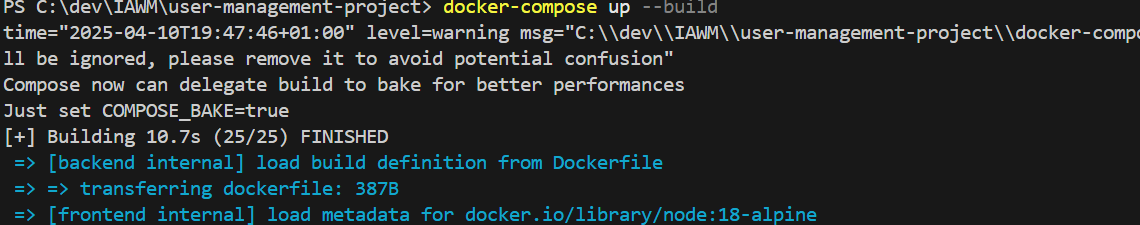
**Optimisations** :

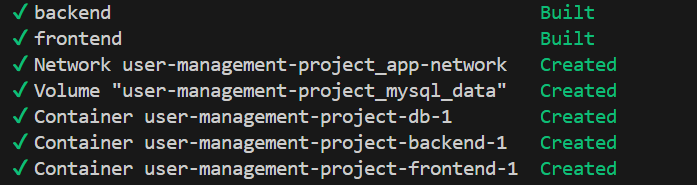
* Utilisation d'alpine pour l'image finale (-60% taille)
* Cache des dépendances via npm ci

**Frontend (**Dockerfile**)**



1. **Déploiement Local**





**Résultat** :

* Backend accessible sur http://localhost:5000
* Frontend sur http://localhost:80
* Base de données persistante via volume

**Partie 3 : Tests Automatisés**

1. **Tests Unitaires (Mocha/Chai)**

**Exemple de test pour la route GET /api/users :**



1. **Couverture de Code**

* Utilisation de nyc ou jest --coverage.

**Partie 4 : CI/CD avec Github Actions**

* **Étapes Clés**

1. **Lancement des tests :**
   * Exécution des tests unitaires avec Mocha/Chai
   * Couverture de code via nyc
2. **Build des images Docker :**
   * Construction parallèle des conteneurs

**Difficultés Rencontrées et Solutions**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | Problème | Solution | Cause/Résolution | | Connexion MySQL échoue dans Docker | Ajout de depends\_on + healthcheck | MySQL non prêt au démarrage du backend | | Port 5000 déjà utilisé | docker-compose down + netstat | Conflit avec un autre processus/contener | | Erreur MODULE\_NOT\_FOUND (dotenv) | npm install dotenv dans Dockerfile | Module manquant dans l'image Docker | | Échec des tests en CI/CD | Service MySQL dédié dans GitHub Actions | Environnement de test isolé requis | | Problème de build frontend | Flag --legacy-peer-deps | Incompatibilité des dépendances React | | Push Docker Hub échoue | Vérification des secrets GitHub | Token Docker Hub mal configuré/invalide | |

**Conclusion**

Ce projet a été ma **première expérience pratique avec Docker**, et j’ai rencontré plusieurs défis techniques, notamment :

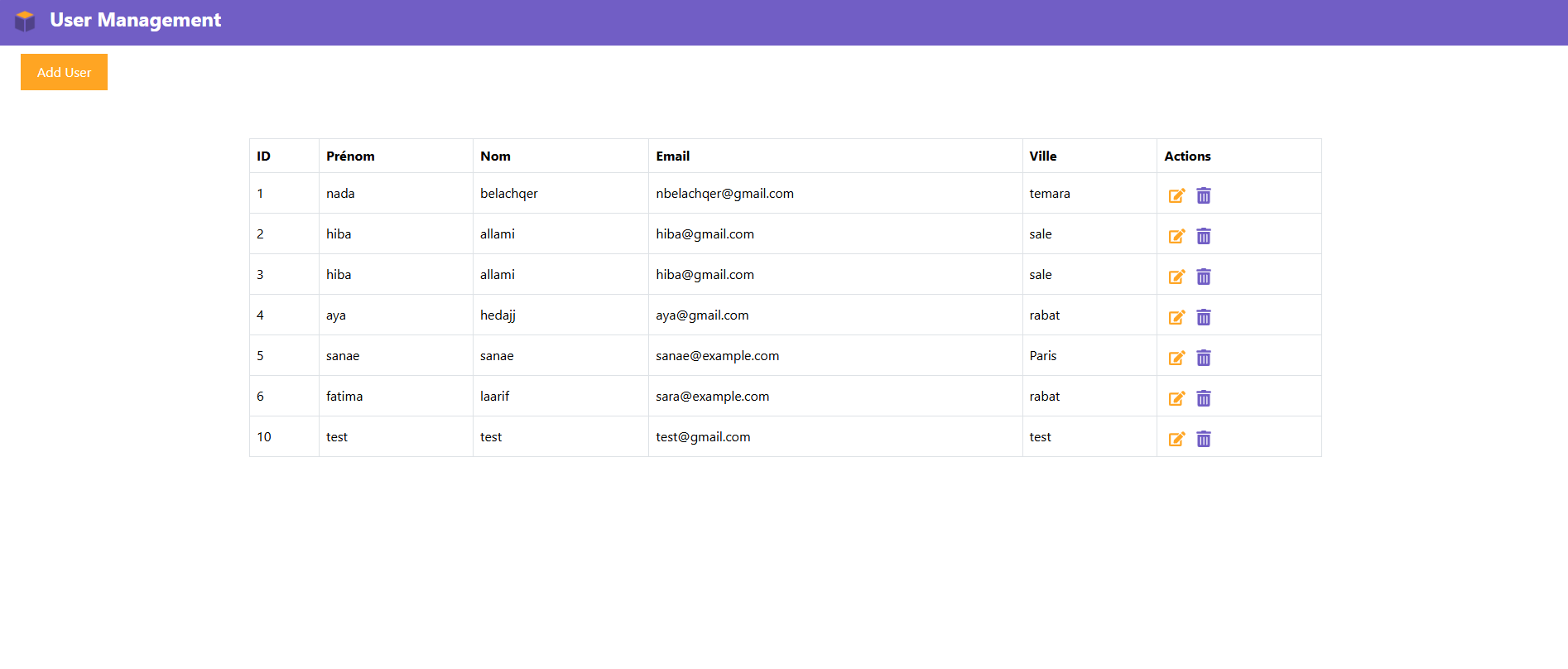
* Des erreurs de **build d’images** (dépendances manquantes, ports bloqués).
* Des problèmes de **tests automatisés** (configuration de la base de données, environnement isolé).
* Des difficultés avec **GitHub Actions** (secrets mal configurés, échecs de push Docker Hub).

Cette **première version**, livrée dans les délais, reste perfectible. À l’avenir, je compte :  
✅ **Optimiser les tests** (augmenter la couverture, ajouter des tests E2E).  
✅ **Améliorer le pipeline CI/CD** (déploiement blue-green, notifications Slack).  
✅ **Automatiser le déploiement** (intégration SSH, scaling avec Kubernetes).

Bien que frustrant par moments, ce TP m’a permis d’acquérir des **compétences solides en conteneurisation et intégration continue**. Je m’investirai davantage pour maîtriser ces outils et rendre mes prochains projets plus robustes.

**Annexe : Captures d'Écran du Frontend**

* 1. **Interface Principale**



*Fig. 1 - Page d'accueil listant les utilisateurs avec fonctionnalités CRUD*

* 1. **Formulaire d'Ajout**

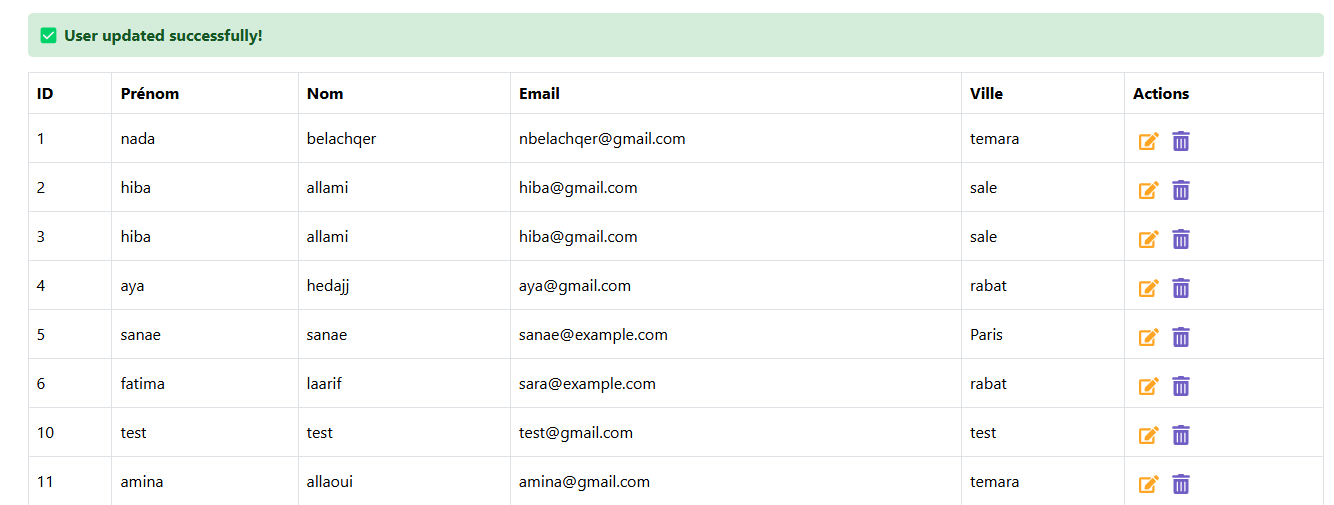


*Fig. 2 - Modal d'ajout d'utilisateur avec validation*

* 1. **Modification d’un utilisateur**

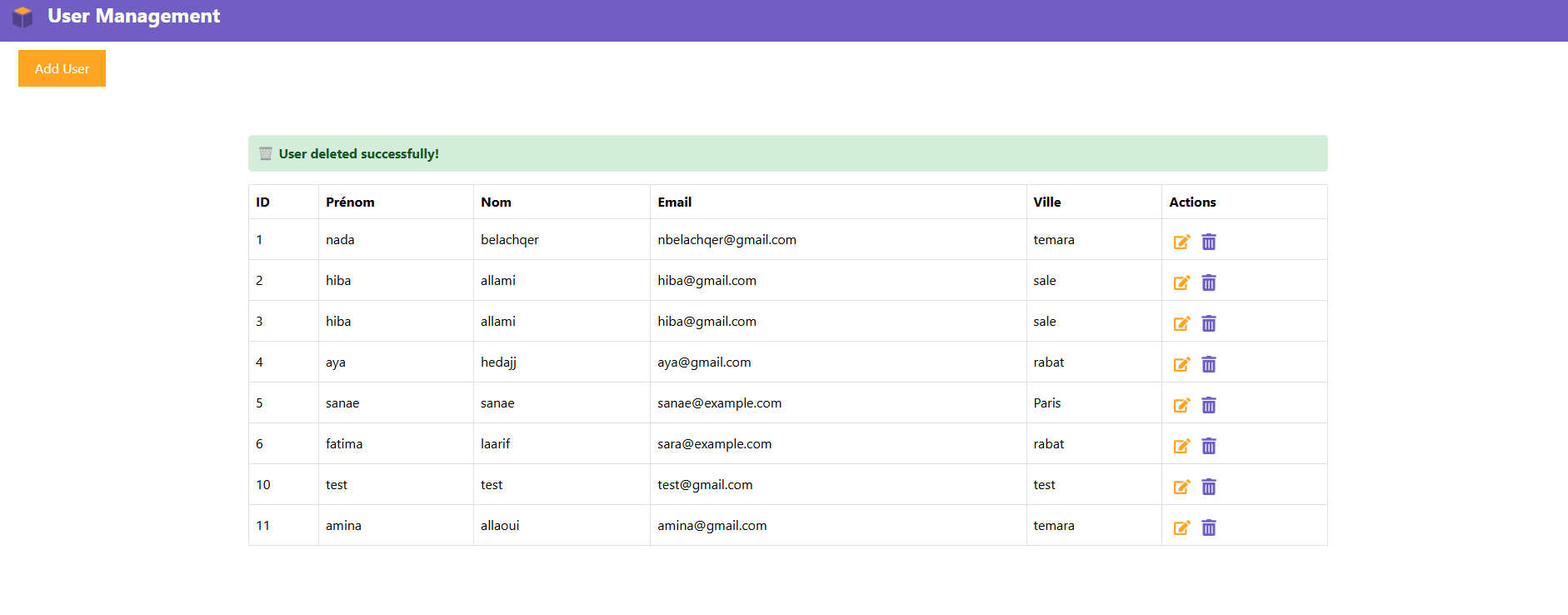


*Fig. 3 - Modal de modification d'utilisateur*



*Fig. 4 Message de mise à jour de la table après modification*

* 1. **Suppression d’un utilisateur**



*Fig. 5 Message de mise à jour de la table après suppression*