RÉALISER PAR ANAS ALIKY

Application Client-Serveur de Gestion des Utilisateurs

1. Présentation générale du projet

Ce projet consiste en une application client-serveur dédiée à la gestion des utilisateurs. Il permet d'ajouter, modifier, supprimer et afficher les utilisateurs à travers une interface simple. L'application est divisée en deux parties :

- Backend : Développé en Node.js avec l'utilisation de Express.js. Le backend gère les opérations liées aux utilisateurs, telles que la création, la modification et la suppression d'utilisateurs.
- Frontend : Développé en React.js, le frontend fournit une interface utilisateur permettant de consulter et gérer les utilisateurs.

Base de données : La base de données utilisée est SQLite pour une gestion légère et simple des données locales des utilisateurs.

L'application est également dockerisée pour une gestion simplifiée des environnements de développement et de production.

2. Étapes de mise en place du backend et frontend

Backend :

Installation de Node.js et Express :

Création d'un serveur Express.js pour la gestion des API.

Mise en place des routes pour les fonctionnalités telles que la création, la gestion et la suppression des utilisateurs.

Connexion à SQLite :

Connexion du backend à la base de données SQLite pour le stockage des utilisateurs.

Ajout de tests unitaires :

Utilisation de Mocha et Chai pour tester les routes API et la logique backend.

Frontend :

Création de l'interface utilisateur avec React :

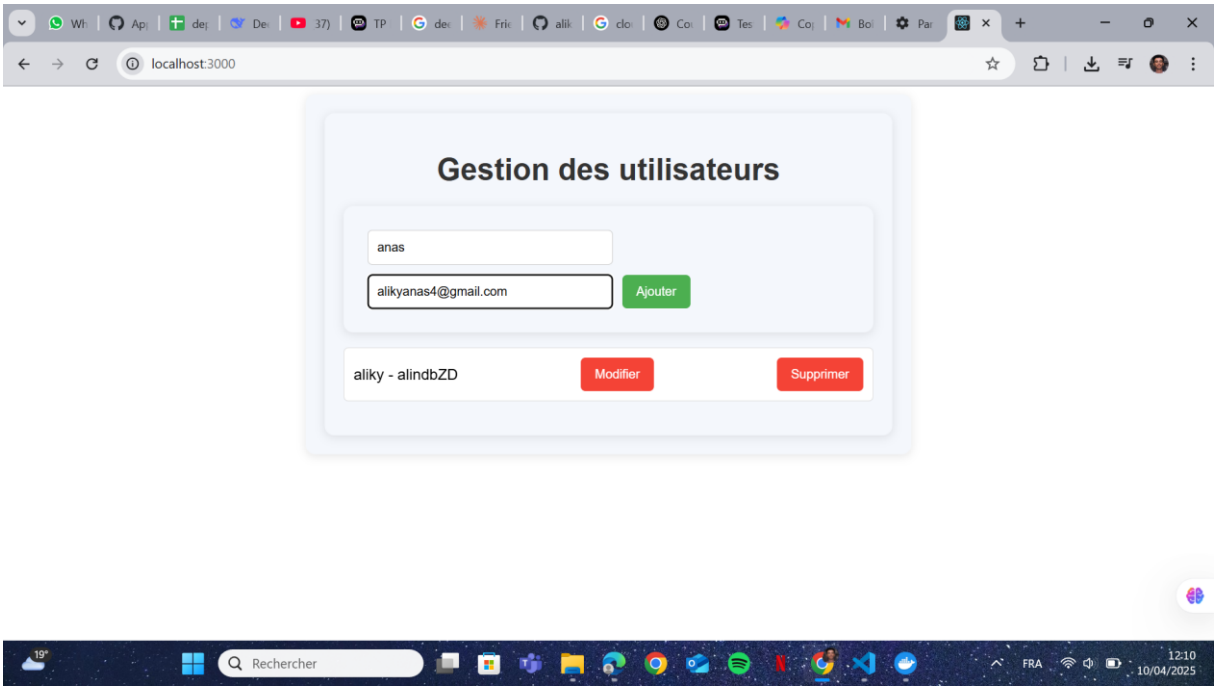
Mise en place des composants React pour l'affichage et la gestion des utilisateurs.

Utilisation de React Router pour la gestion de la navigation entre les pages (par exemple, liste des utilisateurs, formulaire d'ajout).

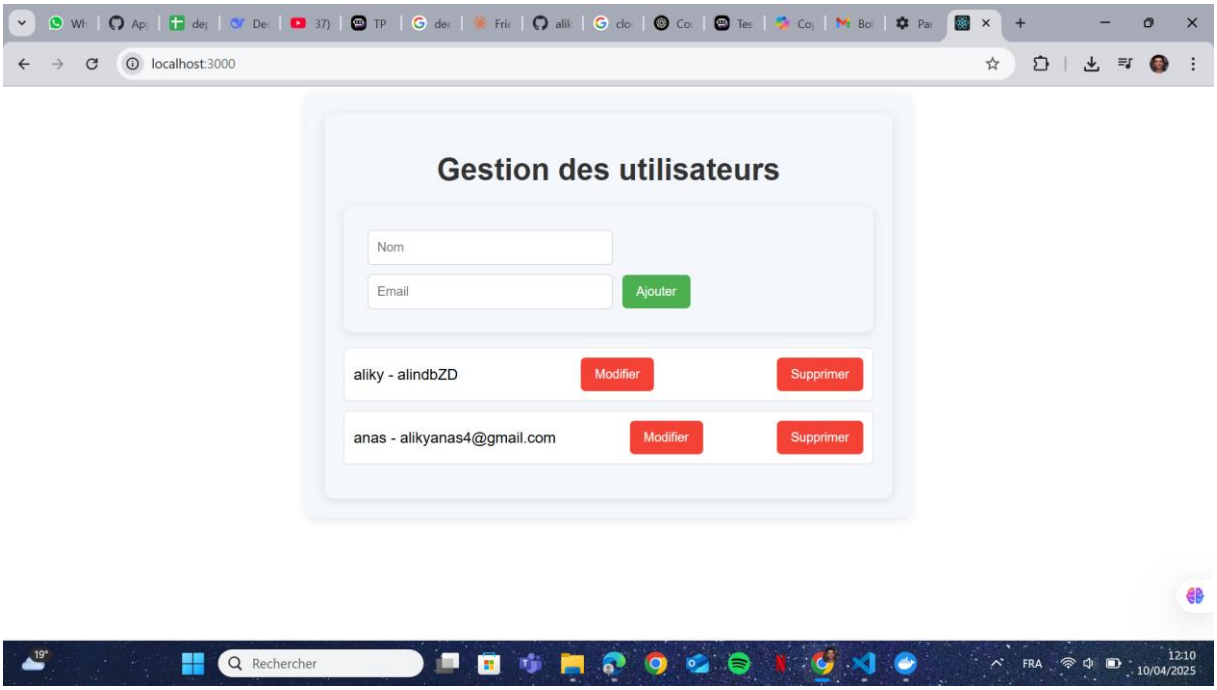
Appels API depuis le frontend :

Utilisation de Axios pour effectuer des requêtes HTTP vers le backend et récupérer les données des utilisateurs.

Affichage dynamique des utilisateurs :



Mise à jour dynamique de l'interface utilisateur en fonction des données récupérées depuis le backend.



3. Explication de la base de données

La base de données utilisée est SQLite, une base de données légère pour le stockage local des utilisateurs. Elle permet de stocker les informations relatives aux utilisateurs, telles que leur nom, email, et numéro de téléphone.

Modèles de données :

Utilisateur : Contient des informations sur chaque utilisateur (nom, prénom, email, etc.).

La connexion entre l'application et SQLite est réalisée via le module sqlite3 de Node.js. Cela permet de gérer les requêtes et d'effectuer les opérations CRUD (Créer, Lire, Mettre à jour, Supprimer).

4. Dockerisation : étapes et choix faits

Backend :

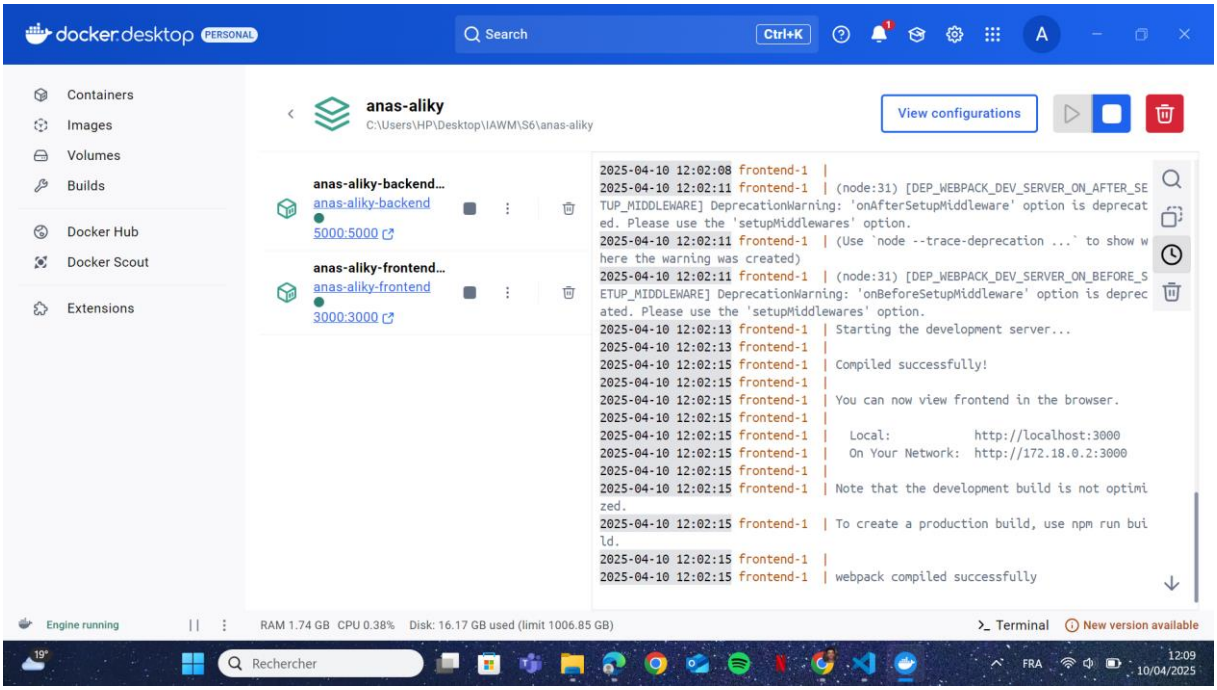
Dockerfile pour le backend :

Création d'un fichier Dockerfile pour dockeriser le backend, avec l'installation de Node.js, la copie des fichiers du backend, et l'exécution de l'application.

Frontend :

Dockerfile pour le frontend :

Création d'un Dockerfile pour construire une image statique du frontend à partir de React et la déployer avec Nginx.



Commandes Docker :

docker-compose build

docker-compose up

5. GitHub Actions : pipeline expliqué étape par étape

Le pipeline CI/CD pour ce projet est géré via GitHub Actions, ce qui permet d'automatiser les processus de test, build, et déploiement des images Docker.

Étapes du pipeline :

Checkout du code : La première étape consiste à récupérer le code du dépôt.

Setup de Node.js : Installation de Node.js pour l'exécution des tests et la construction de l'application.

Installation des dépendances : Installation des dépendances nécessaires pour le backend et le frontend.

Exécution des tests : Lancement des tests unitaires pour vérifier que le backend et le frontend fonctionnent correctement.

Build des images Docker : Création des images Docker pour le backend et le frontend.

Push vers Docker Hub : Envoi des images Docker vers Docker Hub.

Les étapes sont définies dans le fichier ci.yml sous .github/workflows/.

6. Captures d'écran des tests, actions GitHub, conteneurs Docker

7. Difficultés rencontrées et solutions

Difficultés :

Problèmes de connexion à SQLite : Des erreurs de connexion ont été rencontrées lors de la configuration de la base de données, mais elles ont été résolues en ajustant les permissions et en vérifiant le chemin d'accès au fichier SQLite.

Problèmes avec Docker : La dockerisation a causé des conflits de ports entre le backend et le frontend. Pour résoudre cela, des ports distincts ont été assignés et les Dockerfiles ont été ajustés.

Erreurs de test avec GitHub Actions : Certaines étapes du pipeline échouaient en raison de configurations manquantes pour les dépendances Node.js. Ces erreurs ont été corrigées en installant les dépendances correctement dans le fichier ci.yml.