Réaliser par Oubaha Rachid

Application Web le Suivi des Scores de Matchs



# Présentation générale du projet

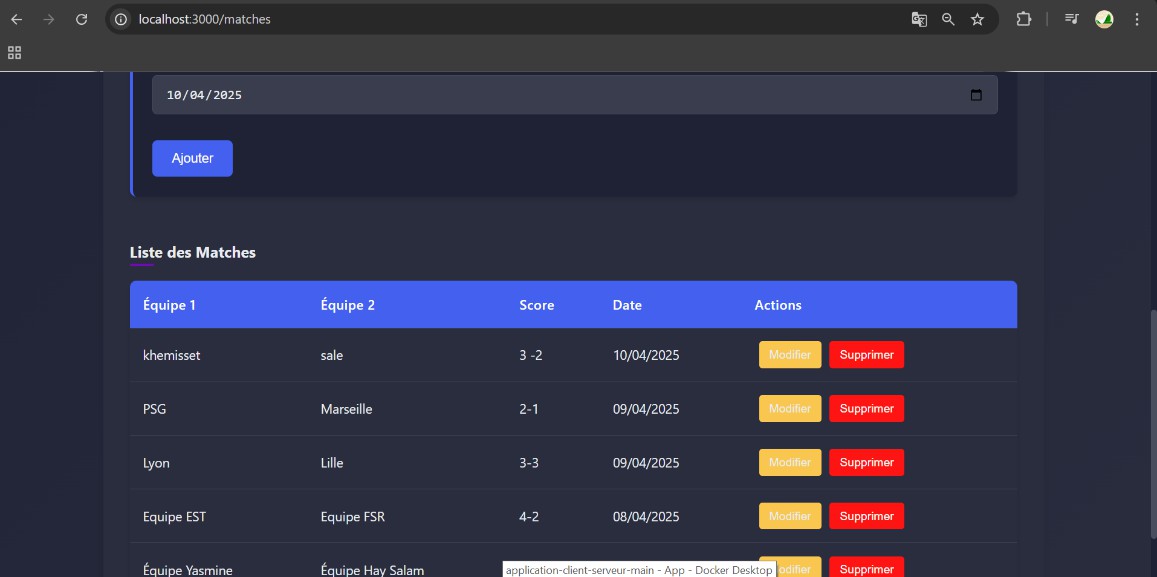
Ce projet consiste en une **application client-serveur** dédiée à la gestion des matchs, utilisée pour enregistrer, suivre, et gérer des matchs entre équipes ou joueurs. L'application est divisée en deux parties :

* + **Backend** : Développé en **Node.js** avec l'utilisation de **Express.js**. Le backend gère les opérations liées aux matchs, telles que la création, la modification, et la suppression de matchs.
  + **Frontend** : Développé en **React.js**, le frontend fournit une interface utilisateur permettant de consulter et gérer les matchs.
  + **Base de données** : La base de données utilisée est **MongoDB Atlas Cloud**, qui permet un stockage flexible des informations liées aux matchs, équipes, utilisateurs, etc.

**Home page :**

**Match liste page :**

****



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 de matchs, la gestion des utilisateurs, etc.

## Connexion à MongoDB Atlas :

* + Connexion du backend à la base de données MongoDB Cloud en utilisant URi.

## 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 des matchs et la gestion des interactions utilisateur.
  + Utilisation de **React Router** pour la gestion de la navigation entre les pages.

## Appels API depuis le frontend :

* + Utilisation de **Axios** pour effectuer des requêtes HTTP vers le backend et récupérer des données.

## Affichage dynamique des matchs :

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

1. Explication de la base de données

La base de données utilisée est **MongoDB Atlas**, une version cloud de MongoDB. Elle est utilisée pour stocker les informations relatives aux matchs, aux utilisateurs, et aux équipes.

La connexion entre l’application et MongoDB est réalisée à l’aide del’URI, Cela permet de définir des schémas de données, valider les entrées, et effectuer des requêtes facilement

## Modèles de données :

* + - **Match** : Contient des informations sur chaque match (date, équipes, scores).
    - **Utilisateur** : Stocke les informations des utilisateurs enregistrés, tels que leur nom, email, et rôle.
    - **Équipe** : Contient les informations sur les équipes participant aux matchs.

La connexion entre l’application et MongoDB est réalisée à l’aide de **Mongoose**, un ORM pour MongoDB. Cela permet de définir des schémas de données, valider les entrées, et effectuer des requêtes facilement.

# Dockerisation : étapes et choix faits

Backend :

## 1. Dockerfile pour le backend :

o Création d'un fichier Dockerfile pour dockeriser le backend, qui inclut l'installation de Node.js, la copie des fichiers du backend dans l'image Docker, et l'exécution de l'application.

Frontend :

## 1. Dockerfile pour le frontend :

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

**Build et exécution de l’image Docker** :Utilisation de Docker pour créer une image et l'exécuter dans un conteneur.

* **Docker compose build**
* **Docker compose up**

# GitHub Actions : pipeline expliqué étape par étape

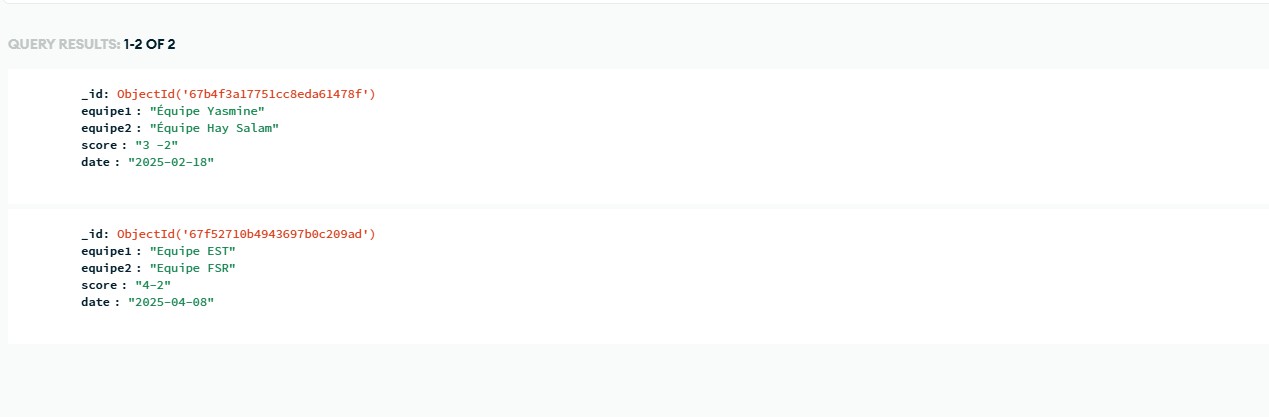
Le pipeline CI/CD pour ce projet est géré via **GitHub Actions**. Il permet d’automatiser les processus de test, build, et déploiement des images Docker vers Docker Hub.

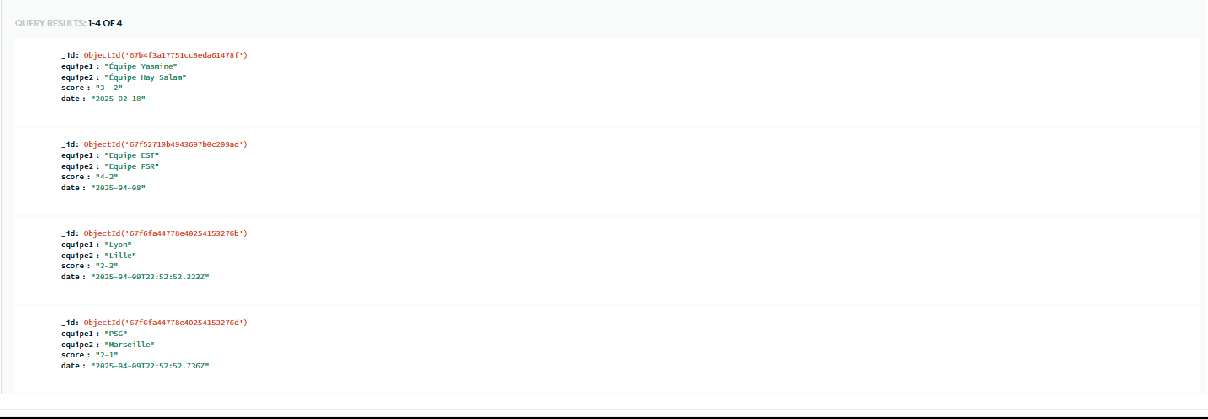
* + **Étapes du pipeline** :

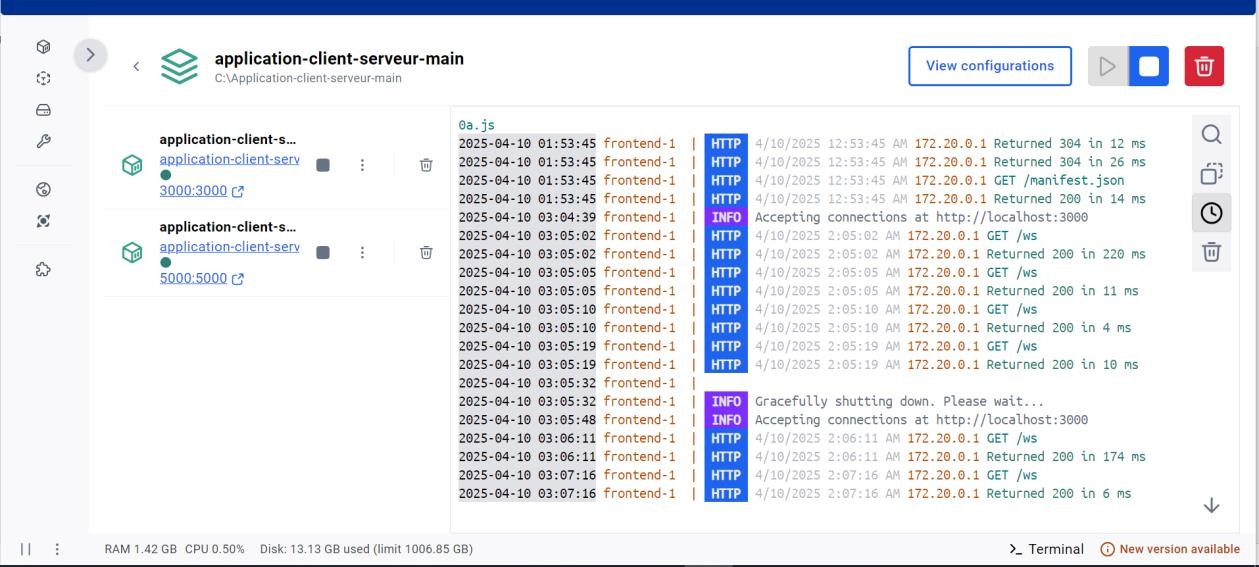
1. **Checkout du code** : La première étape consiste à récupérer le code du dépôt.
2. **Setup de Node.js** : Installation de Node.js pour permettre l'exécution des tests et la construction de l’application.
3. **Installation des dépendances** : Installation des dépendances nécessaires pour le backend et le frontend.
4. **Exécution des tests** : Lancement des tests unitaires pour vérifier que le backend et le frontend fonctionnent correctement.
5. **Build des images Docker** : Création des images Docker pour le backend et le frontend.
6. **Push vers Docker Hub** : Envoi des images Docker vers Docker Hub.

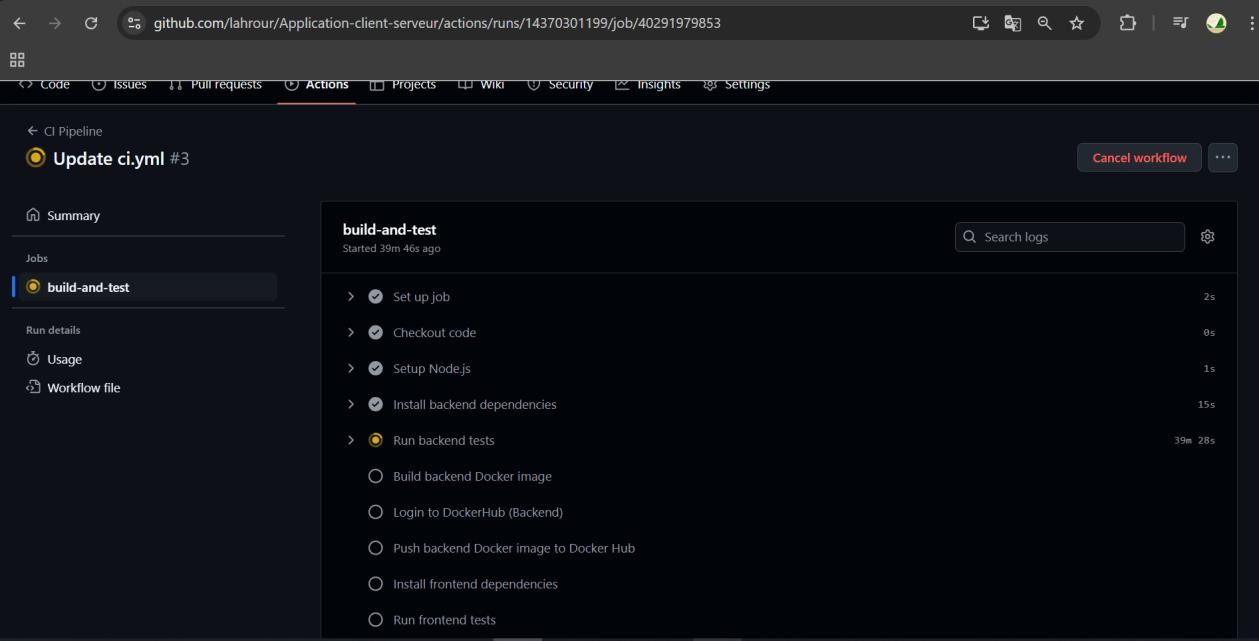
**NB : Les étapes sont définies dans le fichier ci.yml sous .github/workflows/.**

# Captures d'écran des tests, actions GitHub, conteneurs Docker

**La base de données avant le test**

**La base de données après le test**

**Concernant Docker**

**Et concernant action git hub il prend beaucoup de temp pour s’execute :**

# Difficultés rencontrées et solutions

Difficultés :

* + **Problèmes de connexion avec MongoDB Atlas** : Lors de la mise en place de la connexion, il a fallu configurer les variables d’environnement et les règles de sécurité dans MongoDB Atlas.
  + **Problèmes avec Docker** : La dockerisation a causé des conflits de ports entre le backend et le frontend. Il a fallu bien configurer les Dockerfiles et utiliser des ports non conflictuels.
  + **Erreurs dans les tests GitHub Actions** : Certaines étapes du pipeline échouaient à cause de la configuration de Node.js et des dépendances manquantes.

Documentation de l'API

# Endpoints

GET /api/matches

* + Description : Récupère tous les matches
  + Réponse : 200 OK avec un tableau des matches

POST /api/matches

* + Description : Ajoute un nouveau match
  + Corps de la requête : { "equipe1": "string", "equipe2": "string", "score": "string", "date": "string" }
  + Réponse : 201 Created avec les détails du match créé

PUT /api/matches/:id

* + Description : Met à jour un match existant
  + Paramètres : id - ID du match
  + Corps de la requête : { "equipe1": "string", "equipe2": "string", "score": "string", "date": "string" }
  + Réponse : 200 OK avec les détails du match mis à jour

DELETE /api/matches/:id

* + Description : Supprime un match
  + Paramètres : id - ID du match
  + Réponse : 200 OK avec un message de confirmation