Application de Réservation de Terrains

Binome (Groupe 1):

- ✓ Ines OULD AMAR
- ✓ Salaheddin BEN HAMMICH

1. Vue Globale du Projet

Dans le cadre de ce projet, nous avons pour objectif de concevoir et de développer une application web de gestion et de réservation de terrains de football. Ce système permettra aux **utilisateurs** de consulter la disponibilité des terrains, d'effectuer des réservations en ligne, et aux **administrateurs** de gérer les ressources, les réservations et les utilisateurs de manière centralisée.

Le développement de cette application s'inscrit dans une approche **Full Stack** moderne, reposant sur les technologies **NestJS** pour le backend et **React** pour le frontend et **MongoDB** comme base de données. L'objectif principal est de mettre en place une architecture modulaire, sécurisée et évolutive, favorisant la clarté du code, la maintenabilité et la réutilisabilité des composants.

Sur le plan technique, l'application sera structurée autour de plusieurs modules, notamment :

- **User Module** : gestion des utilisateurs et des administrateurs (authentification, inscription, Gestion de Profil, etc…).
- **Stadium Module** : gestion des terrains disponibles.
- **Reservation Module** : gestion des réservations effectuées par les utilisateurs (création, modification, annulation, historique).
- **Sessions Module** : gestion des horaires associés à un terrain.
- **Payment Module** : gestion des paiements.

Ce projet vise à appliquer les principes d'ingénierie logicielle et les bonnes pratiques de développement web, tout en offrant une solution complète et intuitive aux utilisateurs. L'approche adoptée met l'accent sur la sécurité, la performance, et l'expérience utilisateur à travers une interface ergonomique et un backend robuste.

2. Spécification des besoins :

2.1. Besoins Fonctionnels :

➔ Pour l'interface utilisateur :

❖ **S'authentifier :**

L'utilisateur doit pouvoir créer un compte et se connecter à l'aide d'un email et d'un mot de passe.

L'application offre également la possibilité de se connecter via **Google** ou **Facebook**. Le système vérifie les informations d'authentification avant d'accorder l'accès à l'espace personnel.

❖ **Consulter la liste des stades :**

L'utilisateur pourra consulter l'ensemble des stades disponibles, avec des informations détaillées telles que :

- Nom du stade
- Localisation
- Type de terrain
- Disponibilités
- Prix de réservation

❖ **Réserver un stade :**

L'utilisateur pourra réserver un stade pour une date et une heure précise. Le système vérifie la disponibilité du créneau avant de confirmer la réservation.

❖ **Paiement en ligne :**

L'utilisateur pourra effectuer le paiement de sa réservation directement via la plateforme (carte bancaire ou autre moyen pris en charge).
Un reçu ou une confirmation lui sera envoyé après validation du paiement.

❖ **Gestion du profil utilisateur :**

Chaque utilisateur pourra consulter et modifier ses informations personnelles (nom, email, mot de passe, photo, etc.) à tout moment.

❖ **Consulter et gérer ses réservations :**

L'utilisateur pourra consulter la liste de ses réservations passées et à venir, ainsi qu'annuler une réservation avant la date prévue (selon les conditions définies).

❖ **Créer et gérer une liste de joueurs (“Player List”) :**

L’utilisateur pourra créer une liste de joueurs ou inviter d’autres utilisateurs à participer à un match, facilitant l’organisation d’événements sportifs entre amis ou membres d’un club.

❖ **Donner un avis sur un stade :**

Après chaque réservation, l’utilisateur pourra laisser un avis sous forme de note (étoiles) et de commentaire concernant le stade réservé.
Ces avis seront visibles par les autres utilisateurs pour les aider dans leur choix.

➔ Pour l’interface administrateur :

❖ **S’authentifier :**

L’administrateur devra pouvoir se connecter à l’aide de ses identifiants. Le système vérifie l’authentification et accorde l’accès à l’espace d’administration.

❖ **Gestion des utilisateurs :**

L’administrateur pourra gérer les comptes utilisateurs :

- Ajouter un nouvel utilisateur
- Modifier les informations d’un utilisateur
- Supprimer un utilisateur inactif ou indésirable
- Banner un utilisateur

❖ **Gestion des stades :**

L’administrateur aura la possibilité d’ajouter, de modifier ou de supprimer les stades présents sur la plateforme.
Il pourra également mettre à jour les informations concernant la disponibilité et les tarifs.

❖ **Gestion des réservations :**

L’administrateur pourra consulter toutes les réservations effectuées par les utilisateurs, les valider, les modifier ou les supprimer en cas de besoin.

❖ **Gestion des paiements :**

L’administrateur pourra suivre les paiements effectués par les utilisateurs et vérifier les transactions pour assurer la transparence et la sécurité du système.

❖ **Gestion des avis :**

L’administrateur pourra consulter, modérer ou supprimer les avis jugés inappropriés ou contraires aux règles de la plateforme.

2.2. Besoins non Fonctionnels :

Les besoins non fonctionnels sont des exigences qui ne concernent pas spécifiquement le comportement du système mais plutôt ils identifient des contraintes internes et externes du système.

Les principaux besoins non fonctionnels de cette application web sont les suivants :

- ✓ **Performance :**

L'application doit répondre rapidement aux actions des utilisateurs (chargement, réservation, paiement).

Les temps de réponse ne doivent pas dépasser quelques secondes.

- ✓ **Sécurité :**

Les données personnelles et financières doivent être protégées à l'aide de protocoles de sécurité (JWT, cryptage des mots de passe, etc.).

- ✓ **Ergonomie :**

L'interface doit être simple, intuitive et accessible depuis tout type d'appareil (ordinateur, tablette, smartphone).

- ✓ **Disponibilité :**

Le système doit être disponible 24h/24 et 7j/7, sauf en cas de maintenance planifiée.

- ✓ **Scalabilité :**

L'application doit pouvoir évoluer facilement pour supporter un nombre croissant d'utilisateurs et de stades.

- ✓ **Maintenabilité :**

Le code doit être structuré et commenté afin de faciliter les mises à jour et la correction d'éventuelles anomalies.

- ✓ **Compatibilité :**

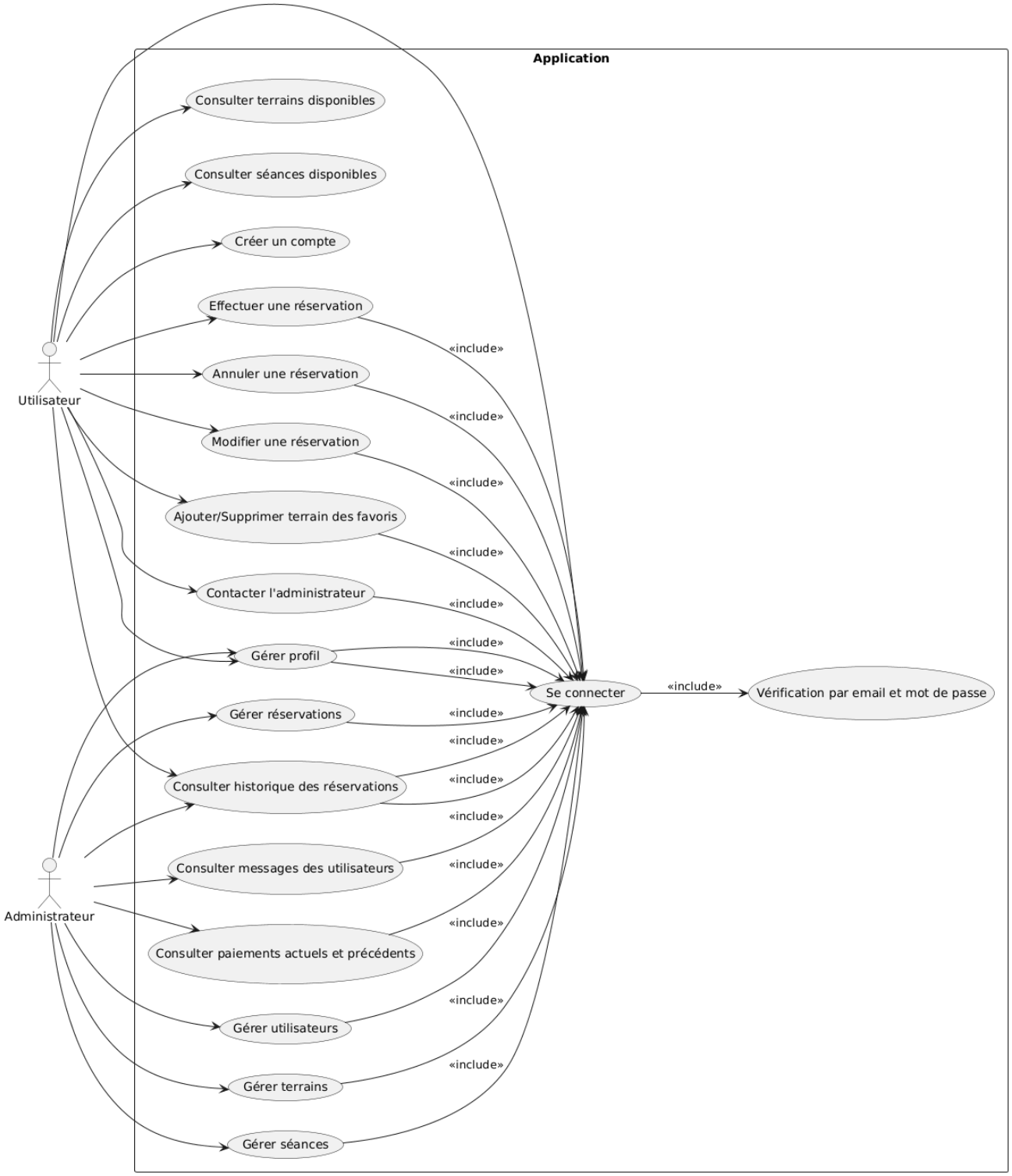
L'application doit être compatible avec les principaux navigateurs modernes (Chrome, Firefox, Edge, Safari).

3. Acteurs :

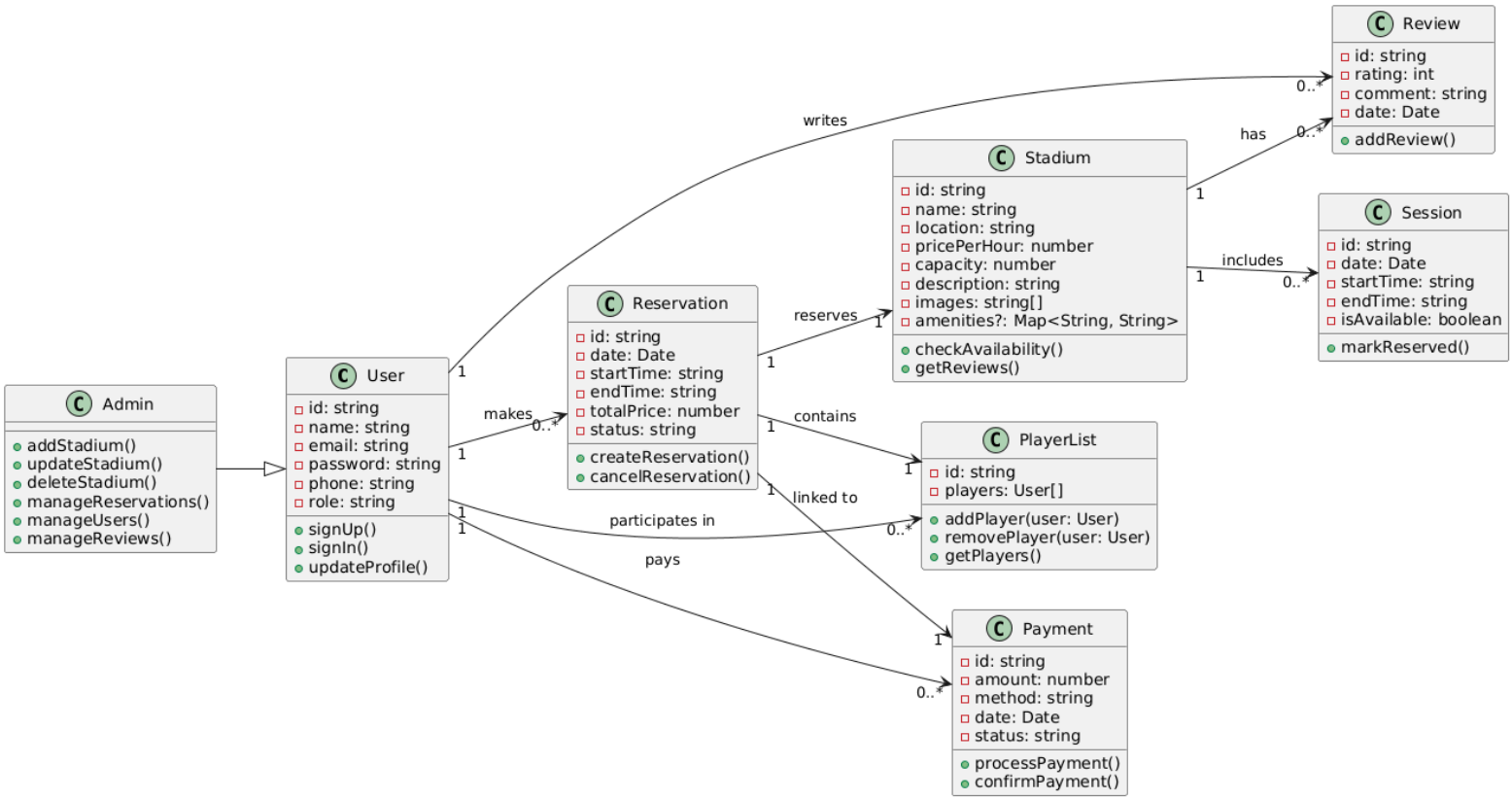
Un acteur est un rôle joué par une personne externe, un processus ou une chose interagit avec un système. Les acteurs qui peuvent interagir avec ce projet sont :

- **Utilisateur/Joueur :** c'est l'acteur le plus important, celui pour lequel le système existe.
- **Administrateur :** c'est l'acteur qui gère l'application web (gérer les utilisateurs, les terrains, etc …).

4. Diagramme des Cas d'utilisation (Provisoire):

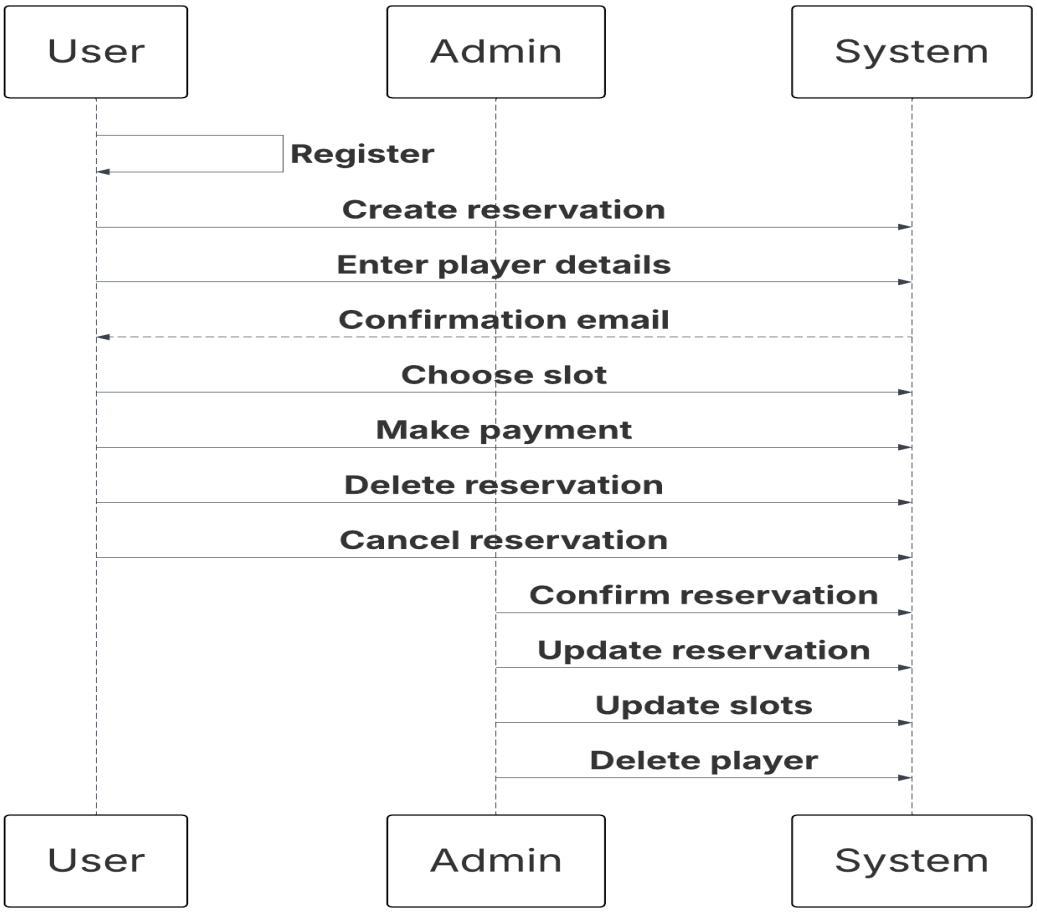


5. Diagramme de classes provisoire (Temporaire et on aura des modifications)



6. Diagramme de séquence (Provisoire):

On prend juste un exemple pour un scénario "Inscription et réservation"



1. **Utilisateur** clique sur "Sign Up".
2. Notre **Frontend (React)** envoie la requête au **Controller (NestJS)**.
3. Le **Controller** transmet les données au **Service**.
4. Le **Service** applique la logique (hash du mot de passe, vérification email unique).
5. Le **Service** sauvegarde le nouvel utilisateur dans **MongoDB** via le **Schema User**.
6. Une confirmation est renvoyée au **Frontend**.

Ensuite, pour la réservation :

1. L'utilisateur authentifié choisit un **Stadium** et une date.
2. Le **Controller Reservation** reçoit la requête.
3. Le **Service Reservation** vérifie la disponibilité du stade.
4. Si disponible → création d'une **Reservation** en base.
5. L'utilisateur procède au **paiement** via le **Payment Module**.

7. Organisation du travail entre les deux binômes

- **Collaboration** : réunions régulières pour synchroniser le code, discuter les points bloquants, utilisation de GitHub pour le versioning, et échanges réguliers sur la conception UML et l'architecture de l'application.
- **Flexibilité du travail** : les binômes peuvent **échanger ou basculer le travail** sur certains composants ou modules selon les besoins, afin d'assurer un apprentissage partagé et une meilleure cohésion du projet.

8. Conclusion

Ce projet a permis de mettre en pratique :

- La création d'une architecture modulaire avec **NestJS**.
- L'intégration d'une base de données **MongoDB**.
- L'utilisation de **React** (avec quelques autres frameworks comme tailwind css comme exemple) pour l'interface utilisateur.
- Le travail collaboratif entre deux binômes avec répartition claire des responsabilités.

L’application pourra être enrichie à l’avenir avec de nouvelles fonctionnalités comme la gestion des tournois, les notifications en temps réel, et l’ajout de paiements sécurisés via des APIs externes.