

RAPPORT PROJET

Développement d’une Platforme d’offres d’emploi de avec python (django)

Réalisé par :

Badre elmouden

3IIR G22

Encadré par :

Pr.Tace

Table des matières

[1. Introduction 3](#_Toc197505297)

[1.1. Contexte du projet 3](#_Toc197505298)

[1.2. Problématique adressée 3](#_Toc197505299)

[1.3. Objectifs du projet 4](#_Toc197505300)

[1.4. Présentation de la solution développée 4](#_Toc197505301)

[2. Analyse des besoins 5](#_Toc197505302)

[2.1. Identification des utilisateurs 5](#_Toc197505303)

[a) Candidat 5](#_Toc197505304)

[b) Recruteur 5](#_Toc197505305)

[2.2. Besoins fonctionnels 6](#_Toc197505306)

[2.3. Besoins non fonctionnels 7](#_Toc197505307)

[2.4. Diagramme des cas d’utilisation 8](#_Toc197505308)

[3.3 Schéma de Base de Données: 10](#_Toc197505309)

[4. Développement et Implémentation 11](#_Toc197505310)

[4.1 Technologies Utilisées 11](#_Toc197505311)

[4.2 Défis Techniques Rencontrés 12](#_Toc197505312)

[7. Conclusion et Perspectives 18](#_Toc197505313)

# Introduction

## Contexte du projet

Dans un environnement professionnel en constante évolution, le processus de recrutement a considérablement changé au cours des dernières années, notamment avec l'émergence d'outils numériques performants. Les entreprises recherchent des moyens plus efficaces pour trouver des profils qualifiés, tandis que les candidats souhaitent accéder rapidement à des offres adaptées à leurs compétences et aspirations.

Les plateformes d’offres d’emploi sont devenues des intermédiaires clés dans ce processus. Elles permettent de publier, rechercher, filtrer et postuler à des offres à travers des interfaces accessibles et intuitives. Dans ce contexte, la digitalisation du recrutement représente un enjeu stratégique tant pour les entreprises que pour les chercheurs d’emploi.

C’est dans cette optique que s’inscrit ce projet de développement d’une plateforme web destinée à faciliter le processus de recrutement à travers une solution technique construite avec Django, un framework web Python puissant et structuré, reposant sur le modèle MVT (Model-View-Template).

## 1.2. Problématique adressée

Malgré la diversité des plateformes existantes, plusieurs limites subsistent :

* L’expérience utilisateur n’est pas toujours optimisée pour les deux types d’utilisateurs (recruteurs et candidats).
* Les processus de candidature sont parfois lourds et peu personnalisables.
* Le suivi des candidatures et les échanges entre utilisateurs manquent souvent de fluidité.
* L’absence de notifications pertinentes peut entraîner une perte d’opportunités.

De plus, pour les étudiants et jeunes diplômés, il est parfois difficile d’avoir une interface adaptée à leur profil débutant. D’un autre côté, les petites entreprises peinent à trouver une plateforme accessible pour publier et gérer leurs offres.

## 1.3. Objectifs du projet

Le développement de ce projet a plusieurs objectifs, à la fois pédagogiques et techniques :

* **Apprentissage et maîtrise de Django** : Comprendre et utiliser l’architecture MVT (Model-View-Template) de Django, apprendre à structurer un projet de manière modulaire.
* **Double interface utilisateur** : Mettre en place une gestion d'utilisateurs différenciée avec des interfaces spécifiques pour les **recruteurs** et les **candidats**, tout en assurant une expérience cohérente et fluide.
* **Fonctionnalités avancées** :
  + Formulaires personnalisés (inscription, connexion, ajout d’offre, candidature, etc.)
  + Publication d’offres avec critères multiples (compétences, type de contrat, lieu, etc.)
  + Système de candidature complet avec suivi et téléchargement de fichiers (CV, lettre de motivation)
  + Système de **recherche filtrée** et **recommandations basiques** d’offres
  + Statistiques pour les recruteurs (nombre de vues, candidatures reçues)
  + Notifications (email ou internes) pour informer des événements importants
* **Utilisation d’outils modernes** :
  + Base de données **MySQL**
  + Framework CSS **Bootstrap** pour un design réactif
  + **Django Authentication** pour la sécurité et la gestion des sessions

## 1.4. Présentation de la solution développée

La solution développée est une **application web** conçue pour connecter des candidats à la recherche d’un emploi avec des recruteurs souhaitant publier et gérer des offres. L’interface est **responsive**, conviviale et adaptée aux deux types de profils :

* **Côté candidat**, l’utilisateur peut :
  + Créer et personnaliser son profil
  + Rechercher des offres selon différents critères
  + Postuler en ligne avec ses documents (CV, lettre de motivation)
  + Suivre l’état de ses candidatures
  + Recevoir des notifications concernant les nouvelles offres ou les réponses des recruteurs
* **Côté recruteur**, l’entreprise peut :
  + Créer un profil entreprise
  + Publier des offres avec des champs complets et structurés
  + Gérer les candidatures reçues (accepter, refuser, contacter)
  + Accéder à un tableau de bord présentant des statistiques de consultation
  + Rechercher des profils candidats selon les critères souhaités

# 2. Analyse des besoins

## 2.1. Identification des utilisateurs

Le système s’adresse à deux types principaux d’utilisateurs ayant des rôles et des besoins distincts :

### a) Candidat

Le candidat est une personne à la recherche d’un emploi ou d’un stage. Ses actions principales sur la plateforme incluent :

* Création et gestion de son profil personnel
* Téléversement de son CV et autres documents justificatifs
* Navigation parmi les offres disponibles
* Filtrage des offres par mot-clé, lieu, type de contrat, etc.
* Candidature à une ou plusieurs offres
* Suivi des candidatures soumises
* Réception de notifications concernant les nouvelles offres ou les réponses des recruteurs

### b) Recruteur

Le recruteur est une entreprise ou une organisation qui souhaite publier des offres d’emploi pour recruter des candidats. Ses actions principales sont :

* Création et gestion du profil entreprise
* Publication de nouvelles offres d’emploi
* Consultation et gestion des candidatures reçues
* Recherche de profils de candidats selon différents critères
* Statistiques sur les offres (nombre de vues, nombre de candidatures)
* Communication avec les candidats via messagerie ou notifications

## 2.2. Besoins fonctionnels

Les besoins fonctionnels correspondent aux fonctionnalités que le système doit obligatoirement fournir. Voici les principaux :

**Pour les candidats :**

* S’inscrire avec un rôle « candidat »
* Se connecter et se déconnecter
* Compléter et modifier leur profil (nom, prénom, email, compétences, expérience, CV, etc.)
* Rechercher des offres selon des filtres multiples (mot-clé, contrat, localisation, etc.)
* Visualiser les détails d’une offre
* Postuler à une offre avec lettre de motivation et documents
* Suivre l’état de chaque candidature
* Recevoir des notifications (accusés de réception, réponse du recruteur, nouvelle offre)

**Pour les recruteurs :**

* S’inscrire avec un rôle « recruteur »
* Créer et mettre à jour le profil de l’entreprise
* Publier de nouvelles offres d’emploi
* Gérer les candidatures reçues (accepter, refuser, contacter)
* Consulter des statistiques sur les offres publiées
* Rechercher des profils de candidats

**Pour les deux :**

* Interface de gestion sécurisée (authentification, sessions)
* Tableau de bord personnalisé
* Messagerie ou zone d’échange entre recruteur et candidat
* Notifications en temps réel ou par mail

## 2.3. Besoins non fonctionnels

Les besoins non fonctionnels définissent les contraintes de qualité du système :

* **Sécurité** : gestion sécurisée des mots de passe, accès restreint aux interfaces selon les rôles
* **Ergonomie** : interfaces simples, claires et accessibles pour tous les utilisateurs
* **Réactivité** : chargement rapide des pages et filtres
* **Évolutivité** : possibilité d’ajouter facilement d’autres modules ou fonctionnalités (ex : API REST)
* **Maintenabilité** : code structuré selon le modèle MVT de Django
* **Compatibilité** : support des principaux navigateurs web et adaptation mobile (responsive design)
* **Fiabilité** : système stable avec gestion des erreurs et validation des formulaires

## 2.4. Diagramme des cas d’utilisation

**-diagramme de classe**

**Une image contenant texte, diagramme, capture d’écran, Police

Le contenu généré par l’IA peut être incorrect.**

Diagramme de cas d’utilisation:

Une image contenant texte, document, diagramme, reçu

Le contenu généré par l’IA peut être incorrect.

### ****3.3 Schéma de Base de Données****:

Une **base de données** est un ensemble organisé de données stockées et gérées de manière à être facilement accessibles, modifiées et mises à jour. Elle permet de stocker des informations structurées de manière à ce qu'elles puissent être facilement consultées

Voici les entité utilisé dans ce projet:



## **4. Développement et Implémentation**

### ****4.1 Technologies Utilisées****

Le développement du système de réservation de terrains s’appuie sur un ensemble de technologies modernes qui garantissent robustesse, évolutivité et maintenabilité :

* **Langage de programmation :** PHP  
  Utilisé pour le développement côté serveur grâce à sa popularité, sa documentation riche et sa compatibilité avec Symfony.
* **Framework :** Symfony  
  Un framework PHP robuste qui facilite la création d’applications web complexes en proposant une architecture MVC, des outils intégrés comme Twig pour les templates, et un gestionnaire de dépendances.
* **Base de données :** MySQL  
  Utilisé pour stocker les données des utilisateurs, des administrateurs et des réservations de manière relationnelle.
* **Interface utilisateur :** HTML, CSS, et JavaScript  
  Fournissent une interface web simple et intuitive permettant aux utilisateurs d'interagir avec le système.
* **Environnement de développement :**
  + Serveur local : XAMPP pour tester l’application localement.

### ****4.2 Défis Techniques Rencontrés****

Le développement a fait face à plusieurs défis techniques qui ont nécessité des solutions adaptées:

#### **Conflits de réservation**

* **Problème :** Empêcher deux utilisateurs de réserver le même terrain pour le même créneau horaire.
* **Solution :** Ajout d’une validation dans le backend pour vérifier la disponibilité du terrain avant la confirmation de la réservation.

#### **Gestion des rôles utilisateurs (client et administrateur)**

* **Problème :** Assurer une séparation claire entre les permissions des administrateurs et des clients.
* **Solution :** Mise en place du système de rôle intégré dans Symfony, avec des contrôles stricts dans les contrôleurs et la base de données.

#### **Optimisation des requêtes à la base de données**

* **Problème :** La gestion de grandes quantités de données pouvait ralentir le système, en particulier pour les recherches et l'affichage des réservations.
* **Solution :** Utilisation des QueryBuilders de Doctrine pour construire des requêtes optimisées et éviter les jointures inutiles.

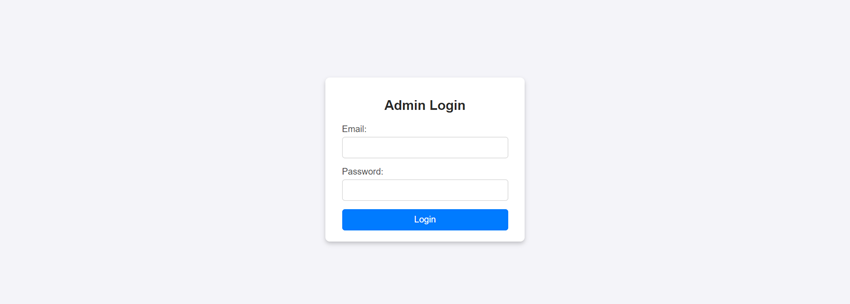
#### **Design réactif pour l’interface utilisateur**

* **Problème :** Garantir une expérience utilisateur fluide sur différentes tailles d’écran (PC, tablette, mobile).
* **Solution :** Utilisation de frameworks CSS comme Bootstrap et de media queries pour adapter l’affichage selon les appareils.

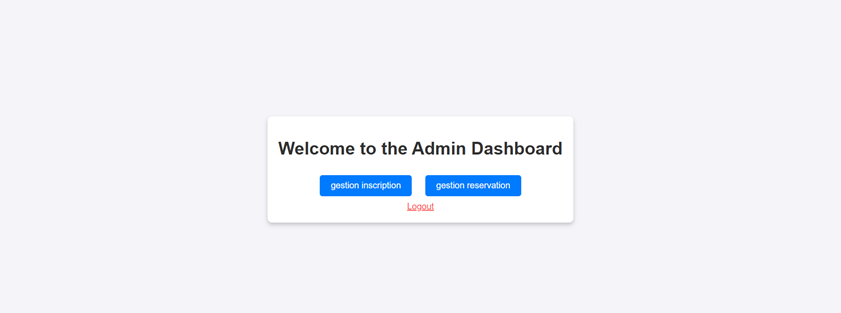
**6.Tests et Validation**

6.1 Méthodologie de Test parti administrateur

1-Admin login: email et mot de passe unique

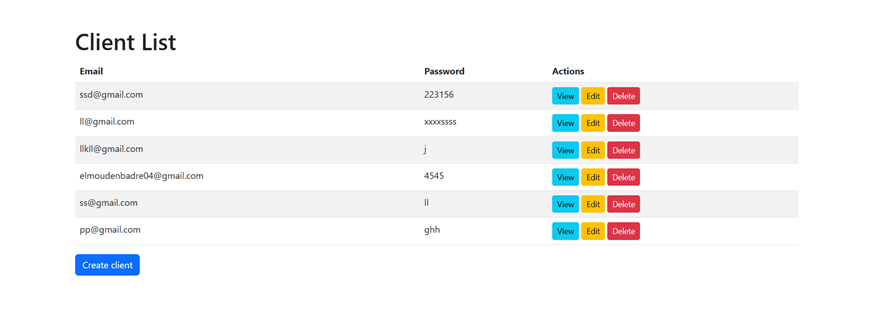


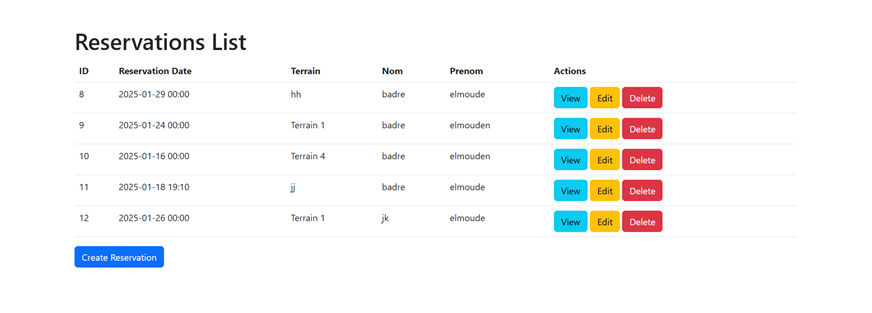
2-Choix de gestion:

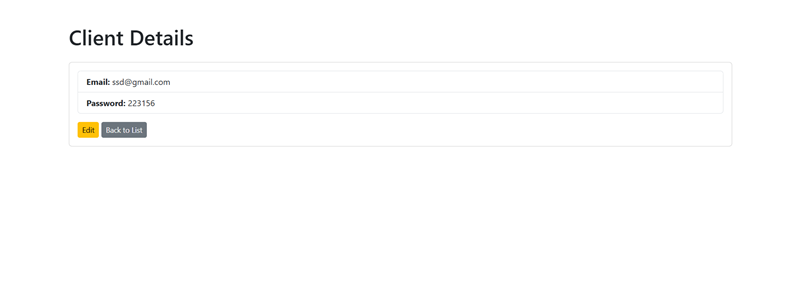


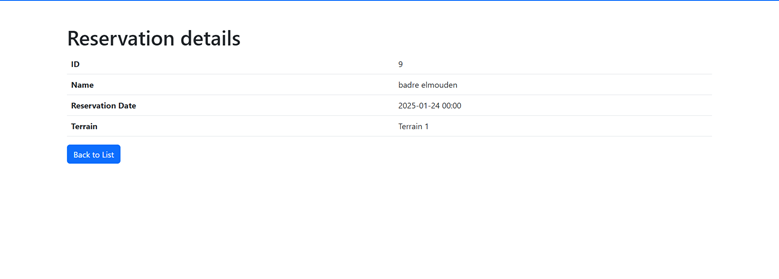
3-système CRUDE:

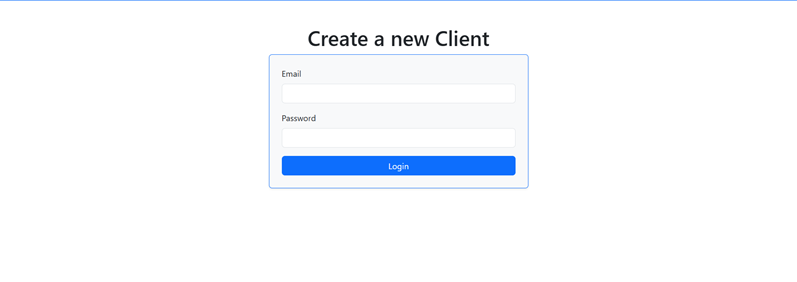
Le système **CRUD** (Create, Read, Update, Delete) représente les quatre opérations de base nécessaires pour gérer les données dans un système ou une base de données. Ces fonctionnalités sont fondamentales dans le développement d’applications. La création (**Create**) permet d’ajouter de nouvelles données, comme l’enregistrement d’un utilisateur ou d’une réservation. La lecture (**Read**) consiste à récupérer et afficher ces données, comme consulter les terrains disponibles ou les réservations. La mise à jour (**Update**) permet de modifier des informations existantes, par exemple changer les détails d’une réservation. Enfin, la suppression (**Delete**) permet de retirer des données du système, comme annuler une réservation ou supprimer un compte. Ces opérations sont généralement implémentées à l’aide de requêtesSQL telles que INSERT, SELECT, UPDATE et DELETE, et constituent la base de nombreux systèmes de gestion modernes

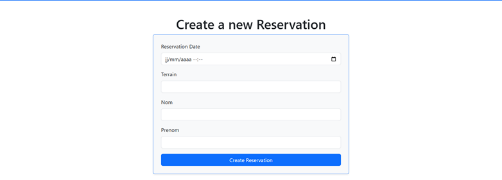


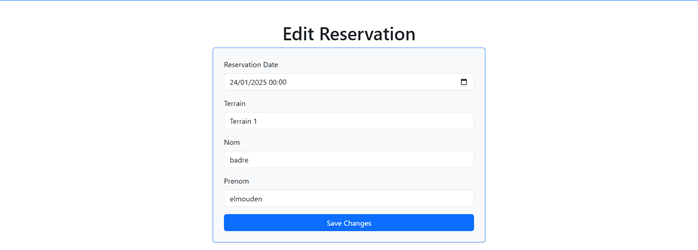


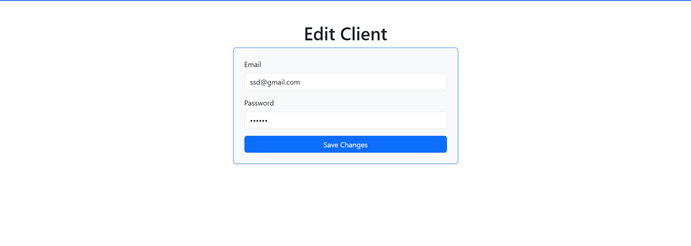


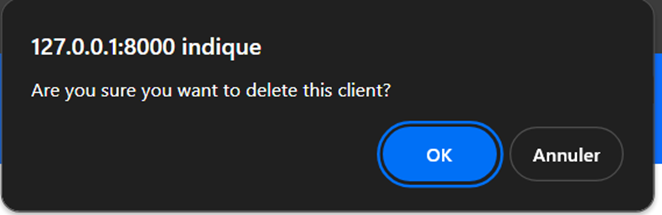


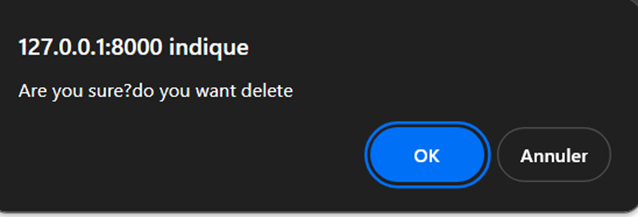






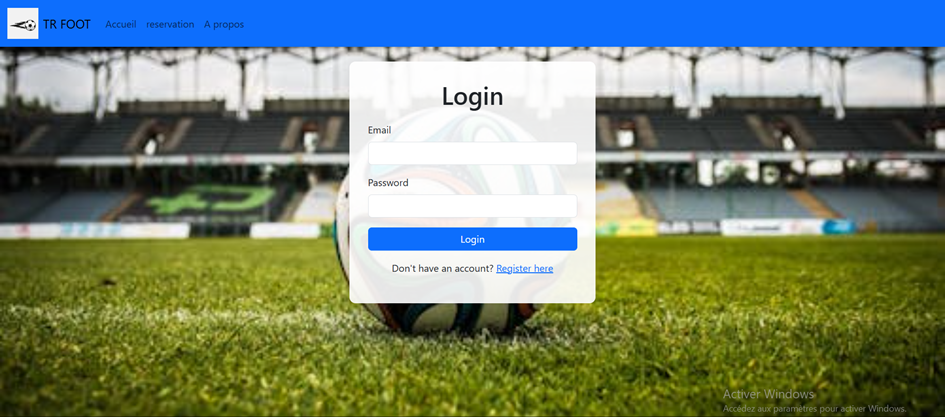


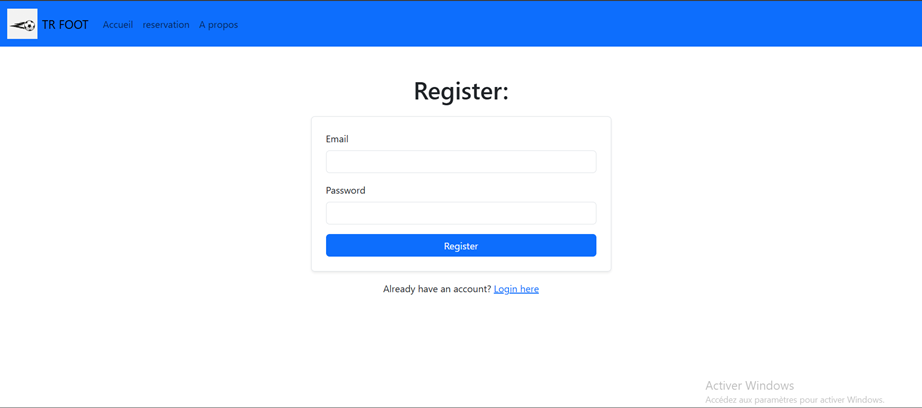


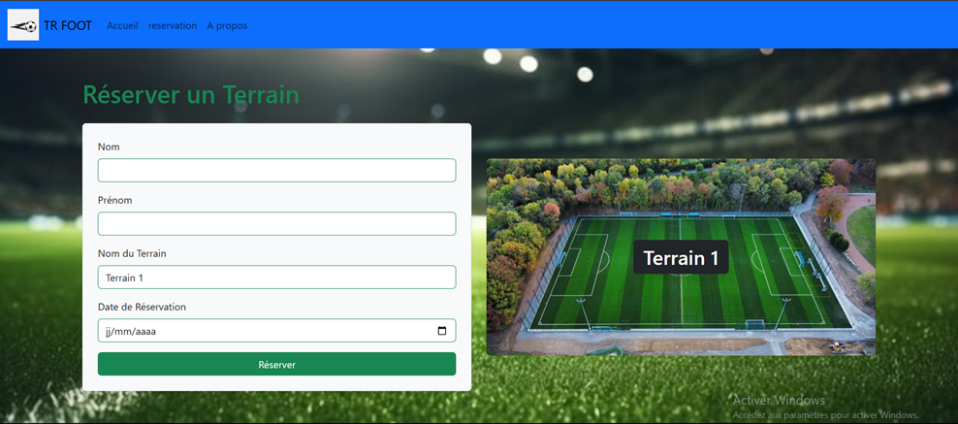


6.1 Méthodologie de Test parti utilisateur

Login utilisateur : Si l'email est incorrect, l'utilisateur est redirigé vers la page d'inscription où il peut créer un nouveau compte. Si le mot de passe est incorrect, un message d'erreur en rouge s'affiche. Dans le cas contraire, l'utilisateur est redirigé vers une autre page



Inscription utilisateur : Si l'email est déjà trouvé dans la base de données, un message d'erreur s'affiche pour informer que l'adresse email est déjà utilisée

Page réservation : Une photo change automatiquement à intervalles réguliers grâce à une fonction JavaScript. Si une date est déjà enregistrée dans le système, un message s'affiche demandant à l'utilisateur de saisir une autre date

Page a propos: contient des informations du site



### ****7. Conclusion et Perspectives****

#### **7.1 Synthèse**

Le projet de développement d'un système de réservation de terrains en PHP Symfony a permis d'automatiser et de simplifier le processus de réservation pour les utilisateurs et les administrateurs. Grâce à l'implémentation des fonctionnalités essentielles, telles que la gestion des utilisateurs, la réservation de terrains et l'administration des données, le système répond aux objectifs initiaux. L'utilisation de Symfony a facilité le développement grâce à son architecture MVC et à ses outils performants, garantissant ainsi une application robuste, sécurisée et évolutive. Les tests réalisés ont confirmé la fiabilité du système dans divers scénarios d'utilisation. Malgré quelques défis techniques rencontrés, comme la configuration de l’authentification JWT ou la gestion des conflits de réservation, ces obstacles ont été surmontés avec succès.

#### **7.2 Perspectives d’Amélioration**

Bien que le projet soit fonctionnel, plusieurs améliorations peuvent être envisagées :

1. **Amélioration de l'interface utilisateur** : Rendre l'interface plus conviviale et moderne grâce à des frameworks front-end comme React ou Vue.js.
2. **Ajout de notifications** : Mettre en place un système de notification (email ou SMS) pour informer les utilisateurs des confirmations ou modifications de leurs réservations.
3. **Gestion des paiements** : Intégrer une plateforme de paiement en ligne pour permettre aux utilisateurs de payer directement leurs réservations.
4. **Rapports avancés** : Ajouter une fonctionnalité permettant aux administrateurs de générer des rapports détaillés sur l'utilisation des terrains et les réservations.
5. **Accessibilité mobile** : Optimiser l'application pour une utilisation sur mobile ou développer une version native pour les appareils mobiles.
6. **Multi-langues** : Permettre le support de plusieurs langues pour rendre le système accessible à un public plus large.

**8-**Remerciments

Je tiens à exprimer ma sincère gratitude à toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce projet.

Tout d’abord, je remercie chaleureusement mon encadrant, Monsieur tace, pour son soutien constant, ses conseils précieux, et sa disponibilité tout au long du projet.

Je remercie également mes camarades de classe pour leurs retours constructifs et leur encouragement.

Enfin, je remercie l’EMSI pour m’avoir offert l’opportunité de travailler sur ce projet enrichissant