Mémoire de Projet de Fin d'Études

Digitalisation de la Ferme Expérimentale

Soutenu par : Marwa El ABIDI

Jury :

Pr. Abdelfattah Charif CHEFCHAOUNI
Pr. Mounir CHAOUKI
M. Younes JNAOUI

Année universitaire : 2024–2025

Remerciements

Je tiens tout d'abord à exprimer ma sincère gratitude à l'Université Mohammed VI Polytechnique, et en particulier à l'Institut des Sciences, Technologies et Innovations (IST&I), pour la richesse des enseignements qui m'ont été dispensés tout au long de mon cursus de Licence.

Je remercie chaleureusement mon encadrant académique Pr. Abdelfattah Charif CHEFCHAOUNI, aussi coordinateur de la filière. Son accompagnement rigoureux, sa grande disponibilité et ses précieux conseils ont été des piliers pour la réalisation de ce mémoire.

Je tiens également à remercier M. Younes JNAOUI, responsable de la Ferme Expérimentale UM6P, pour m'avoir accueilli au sein de son équipe avec bienveillance et professionnalisme. Ma reconnaissance s'étend également à M. Brahim ELKAJAF, pour son encadrement rigoureux, sa confiance et son soutien constant tout au long du stage.

Je remercie toute l'équipe de la ferme expérimentale pour leur collaboration, leur disponibilité et leur esprit d'équipe, qui ont grandement facilité mon intégration et la réalisation de ce projet.

Je souhaite exprimer ma profonde gratitude et mon plus grand respect au Professeur Radouane BOUKHARFANE. Son soutien et ses encouragements constants tout au long de ma formation ont été pour moi une véritable source de motivation.

Enfin, j'adresse un remerciement particulier envers tout le corps professoral et le personnel administratif du département College Of Computing de l'UM6P qui nous ont accompagnés durant ces 3 dernières années, pour tous leurs efforts et leur travail visant à maintenir et améliorer la qualité de la formation au sein de notre université.

Résumé

Dans le cadre de ma formation à l'Université Mohammed VI Polytechnique, j'ai eu l'opportunité de réaliser un stage de fin d'études au sein de la ferme expérimentale de l'université. Ma mission principale consistait à concevoir et développer une plateforme numérique de gestion destinée à centraliser et structurer les données agricoles, expérimentales et techniques, afin de remplacer le système existant basé sur des formulaires papier.

Le projet a été entièrement développé sur une architecture moderne, avec un backend robuste en Node.js et Express.js, et une base de données relationnelle PostgreSQL gérée via l'ORM Prisma. L'interface utilisateur est une application web dynamique et interactive construite en React et TypeScript. Cette plateforme permet une gestion efficace des utilisateurs avec des rôles distincts (Administrateur, Superviseur, Client) grâce à un système d'authentification sécurisé par JWT. Elle digitalise également les flux de travail clés, incluant les réservations de parcelles, le suivi de projets, et la gestion de l'inventaire.

En adoptant une approche itérative, le projet a progressé d'une phase d'analyse des besoins des utilisateurs jusqu'au déploiement d'un prototype fonctionnel. Tout au long du processus, des défis techniques liés à la modélisation des données, à la sécurisation de l'API RESTful et à l'ergonomie de l'interface ont été surmontés. Ces obstacles m'ont permis de renforcer mes compétences en conception d'architecture logicielle, en développement full-stack et en gestion de bases de données.

Ce rapport présente en détail les différentes étapes du projet, les choix technologiques effectués, ainsi que les perspectives d'évolution à long terme. Cette expérience m'a permis d'appliquer concrètement mes connaissances théoriques et de contribuer activement à la transformation numérique du secteur agricole.

Mots clés : Digitalisation, Ferme expérimentale, Node.js, React, PostgreSQL, Prisma, JWT, Gestion de projet agricole, Architecture web.

Abstract

As part of my academic program at Mohammed VI Polytechnic University, I had the opportunity to undertake a final year project at the university's experimental farm. My primary mission was to design and develop a digital management platform to centralize and structure the agricultural, experimental, and technical data, replacing the existing paper-based system.

The project was fully developed on a modern architecture, featuring a robust backend built with Node.js and Express.js, and a relational PostgreSQL database managed via the Prisma ORM. The user interface is a dynamic and interactive web application built with React and TypeScript. This platform enables efficient user management with distinct roles (Admin, Supervisor, Client) through a secure JWT-based authentication system. It also digitizes key workflows, including field reservations, project tracking, and inventory management.

Adopting an iterative approach, the project progressed from a user requirements analysis phase to the deployment of a functional prototype. Throughout the process, technical challenges related to data modeling, securing the RESTful API, and interface usability were overcome. These obstacles allowed me to strengthen my skills in software architecture design, full-stack development, and database management.

This report details the various stages of the project, the technological choices made, and the outlook for long-term evolution. This experience enabled me to apply my theoretical knowledge in a practical context and to actively contribute to the digital transformation of the agricultural sector.

Keywords : Digitalization, Experimental Farm, Node.js, React, PostgreSQL, Prisma, JWT, Agricultural Project Management, Web Architecture.

Liste des abréviations

ACID	Atomicity, Consistency, Isolation, Durability
AITTC	Agricultural Innovation and Technology Transfer Center
API	Application Programming Interface
CORS	Cross-Origin Resource Sharing
CRUD	Create, Read, Update, Delete
CSV	Comma-Separated Values
ERD	Entity–Relationship Diagram
ES6	ECMAScript 2015 (ES6)
FK	Foreign Key
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
IETF	Internet Engineering Task Force
IoT	Internet of Things
ISTI	Institute of Science, Technology and Innovation
JSON	JavaScript Object Notation
JWT	JSON Web Token
MDN	Mozilla Developer Network
MUI	Material-UI
MVC	Model–View–Controller
ORM	Object-Relational Mapping
PDF	Portable Document Format
PFE	Projet de Fin d'Études
PK	Primary Key
RBAC	Role-Based Access Control
REST	Representational State Transfer
RFC	Request for Comments
SMTP	Simple Mail Transfer Protocol

SPA	Single-Page Application
SQL	Structured Query Language
UI	User Interface
UK	Unique Key
UML	Unified Modeling Language
UM6P	Université Mohammed VI Polytechnique
UUID	Universally Unique Identifier
UX	User Experience
W3C	World Wide Web Consortium
WAI/WCAG21	Web Accessibility Initiative / WCAG 2.1
WCAG	Web Content Accessibility Guidelines
XSS	Cross-Site Scripting

Table des figures

1.1 Diagramme Entité-Association (ERD) du système de gestion 18

2.1 Architecture générale de la plateforme de gestion de la ferme expérimentale . . 22
2.2 Diagramme de cas d'utilisation des acteurs du système. 29
2.3 Diagramme de Classes. 30
2.4 Diagramme de Séquence. 31
2.5 Page de connexion de l'application 32
2.6 Interface de gestion des champs 34
2.7 Interface de gestion des projets 35
2.8 Interface de gestion des utilisateurs avec tableau de données et actions contextuelles 35
2.9 Fenêtre d'examen et de validation d'une réservation par l'administrateur 36
2.10 Interface de gestion des inventaires 37
2.11 Interface de réservation d'un champ 37
2.12 Interface de la gestion des projets (clients) 38
2.13 Interface de la gestion des champs (clients) 38
2.14 Interface de gestion des offres prix 39
2.15 Vue d'ensemble du tableau de bord administrateur avec statistiques et métriques clés 40
2.16 Navigation adaptative selon le rôle utilisateur (ADMIN) 41
2.17 Représentation visuelle des statuts par codes couleur (PENDING, APPROVED, REJECTED) 41
2.18 Barre de navigation principale affichant les sections accessibles au client 42
2.19 Message d'erreur affiché à l'utilisateur 42
2.20 Message de succès affiché à l'utilisateur 42

A.1 Fiche F.10 - Inventaire hebdomadaire (document fourni) 63
A.2 Fiche F.47 - Fiche offre de prix (document fourni) 64
A.3 Fiche F.75 - Fiche management du comité de pilotage (document fourni) 65
A.4 Fiche F.84 - Fiche ordre de service (document fourni) 66

Liste des tableaux

1.1 Récapitulatif des droits par rôle utilisateur 16

Table des matières

Résumé **3**

Abstract **4**

Introduction Générale **12**

1 Analyse des Besoins et Spécifications **15**
1.1 Description des utilisateurs et des rôles 15
1.2 Besoins fonctionnels 16
1.3 Besoins Non Fonctionnels 17
1.4 Modélisation du système 18

2 Conception technique et modélisation de la solution **21**
2.1 Architecture générale du système 21
2.1.1 Schéma d'architecture globale 22
2.1.2 Description des couches 23
2.1.3 Pattern architectural 23
2.2 Choix technologiques 24
2.2.1 Frontend 25
2.2.2 Backend 25
2.2.3 Base de données et persistance 26
2.2.4 Justification des choix 27
2.3 Modélisation technique 28
2.3.1 Diagramme de cas d'utilisation 28
2.3.2 Diagramme de Classes 30
2.3.3 Diagramme de Séquence : 30
2.4 Sécurité et gestion des accès : 31
2.4.1 Gestion de l'authentification : 31
2.4.2 Gestion des rôles et permissions (RBAC) : 32
2.4.3 Protection des routes et des données : 33
2.5 Conception de l'interface utilisateur 33

2.5.1 Fonctionnalités principales accessibles à l'administrateur 34
2.6 Fonctionnalités principales accessibles à l'utilisateur 37
2.6.1 Maquettes et wireframes 40
2.6.2 Principes d'ergonomie et d'accessibilité 40
2.6.3 Navigation et notifications 41

3 Implémentation 44
3.1 Structure et organisation du projet 44
3.1.1 Architecture des dossiers 44
3.1.2 Standards de codage et conventions 45
3.1.3 Gestion des dépendances et configuration 45
3.2 Développement du backend 46
3.2.1 Configuration du serveur Express et middlewares 46
3.2.2 Implémentation des routes API principales 46
3.2.3 Gestion de l'authentification et des autorisations 47
3.2.4 Intégration de Prisma et gestion de la base de données 47
3.2.5 Services utilitaires 47
3.3 Défis techniques et solutions apportées 48
3.3.1 Gestion de l'état global et synchronisation 48
3.3.2 Sécurité et validation des données 48
3.3.3 Performance et optimisation 48
3.3.4 Gestion des erreurs et robustesse 49

4 Résultats et évaluation 50
4.1 Fonctionnalités réalisées 50
4.2 Évaluation technique 51
4.2.1 Performance et stabilité 51
4.2.2 Sécurité et robustesse 51
4.2.3 Qualité du code et maintenabilité 51
4.2.4 Points d'amélioration identifiés 52
4.3 Évaluation fonctionnelle 52
4.3.1 Expérience utilisateur 52
4.3.2 Gestion des cas d'erreur 53
4.3.3 Fonctionnalités manquantes ou limitées 53
4.3.4 Satisfaction des objectifs 53
4.4 Limites et difficultés rencontrées 53
4.4.1 Contraintes techniques 53
4.4.2 Limitations fonctionnelles 54
4.4.3 Défis de développement 54
4.4.4 Contraintes de ressources 54

4.5 Conclusion de l'évaluation 55
4.5.1 Synthèse des résultats 55
4.5.2 Satisfaction des objectifs 55
4.5.3 Points forts et acquis 56
4.5.4 Perspectives d'amélioration 56
4.6 Bilan du projet 56
4.6.1 Objectifs atteints 56
4.6.2 Compétences développées 56
4.6.3 Limites et apprentissages 57

A Annexes **62**
A.1 Annexe A : Documents de référence 63
A.1.1 Fiche F.10 - Inventaire hebdomadaire 63
A.1.2 Fiche F.47 - Fiche offre de prix 64
A.1.3 Fiche F.75 - Fiche management du comité de pilotage 65
A.1.4 Fiche F.84 - Fiche ordre de service 66
A.2 Annexe B : Structure de la base de données 67
A.2.1 Schéma de base de données (Prisma) 67
A.3 Annexe C : Extraits de code source représentatif 69
A.3.1 Configuration du serveur Express (extrait principal) 69
A.3.2 Composant React principal (extrait) 70

Introduction Générale

En 2023, le secteur agricole marocain poursuit une transition nécessaire vers une modernisation durable, portée par les enjeux croissants de productivité, d'optimisation des ressources et d'adaptation aux changements climatiques. Dans ce contexte, l'Université Mohammed VI Polytechnique (UM6P) s'impose comme un acteur stratégique à travers ses infrastructures de recherche appliquée, dont la ferme expérimentale constitue un pilier central.

Installée sur Ben Guerir, la ferme expérimentale de l'UM6P s'étend sur plusieurs hectares et fait partie d'un écosystème d'innovation regroupant également le Green Energy Park, le Data Center Toubkal et d'autres plateformes technologiques de pointe. Ce dispositif unique permet de développer et de tester, en conditions réelles, des solutions agricoles intelligentes fondées sur l'usage des données, de l'IoT, de l'intelligence artificielle ou encore des énergies renouvelables. Véritable laboratoire vivant, la ferme expérimentale offre un terrain d'expérimentation aux chercheurs, ingénieurs et étudiants pour co-construire l'agriculture de demain.

Cependant, malgré ce potentiel technologique, la gestion des projets agricoles et expérimentaux au sein de la ferme reste fortement dépendante de processus manuels, dispersés et en grande partie basés sur des documents papier. Cette fragmentation engendre des lenteurs administratives, un manque de traçabilité dans le suivi des projets, une perte d'information et une difficulté dans la coordination des acteurs.

Face à ce constat, la digitalisation apparaît comme une nécessité stratégique. Le présent projet de fin d'études s'inscrit pleinement dans cette vision en proposant la conception et le développement d'une plateforme web centralisée dédiée à la gestion des projets de la ferme expérimentale. Cette solution numérique vise à automatiser l'ensemble du processus, depuis la soumission des demandes jusqu'à leur validation, en passant par le suivi d'avancement et la gestion des utilisateurs.

En transformant un workflow complexe en un système fluide, transparent et accessible, cette plateforme ambitionne de renforcer l'efficacité opérationnelle de la ferme, de fluidifier la prise de décision, de valoriser les données générées sur le terrain et de favoriser une gouvernance moderne alignée sur les objectifs de durabilité et d'innovation portés par l'UM6P et le secteur agricole marocain.

Problématique : Workflow Actuel

Actuellement, la gestion administrative des projets au sein de la ferme repose principalement sur des procédures manuelles et des outils bureautiques non intégrés, un mode de fonctionnement qui présente de multiples inconvénients limitant l'efficacité globale. Le processus débute par une démarche entièrement physique : les intervenants doivent se déplacer pour soumettre une demande, qui est ensuite consignée sur un formulaire papier. Cette dépendance au format physique se poursuit tout au long du cycle de validation, où la circulation des documents entre les différents services se fait manuellement, induisant une lenteur systémique et un manque de fiabilité.

Cette fragmentation ne se limite pas au flux de travail, mais affecte également l'information elle-même. La multiplicité des données à suivre (telles que les superficies, les statuts d'avancement, les responsables ou les dates clés) est gérée de manière éclatée, rendant toute consolidation ou analyse globale extrêmement difficile. Par conséquent, il devient quasi impossible d'obtenir une vision claire et en temps réel de l'état d'avancement d'un projet ou d'une demande. Cette opacité est aggravée par le risque tangible de perte ou de détérioration des formulaires papier, qui sont par nature vulnérables.

Enfin, l'ensemble de ce système engendre une charge administrative considérable pour le personnel. Les tâches répétitives de saisie manuelle, de vérification des informations et d'archivage physique des documents mobilisent un temps et des ressources importants, qui pourraient être alloués à des activités à plus forte valeur ajoutée. Ces défis combinés (délais importants, manque de traçabilité, et lourdeur administrative) justifient pleinement la nécessité d'une transition vers une solution numérique intégrée.

Objectifs du Projet

L'objectif principal de ce projet est de concevoir et de développer une plateforme web centralisée, dédiée à la digitalisation et à l'automatisation des procédures administratives de la ferme expérimentale. Cette solution vise à offrir un environnement sécurisé grâce à un système d'authentification robuste, permettant de distinguer clairement les rôles et les permissions des différents utilisateurs, qu'il s'agisse des administrateurs, des superviseurs ou des clients. Au cœur de la plateforme, un module de gestion des demandes de parcelles a été mis en place, intégrant une vérification automatique de la disponibilité afin d'optimiser l'allocation des ressources foncières. L'automatisation s'étend également à l'ensemble du cycle de vie des projets, depuis la soumission des demandes jusqu'à leur validation ou leur rejet, tout en assurant un suivi rigoureux et un archivage systématique des opérations réalisées. Par ailleurs, la plateforme intègre la gestion des ordres de service, permettant de planifier, suivre et valider les interventions sur le terrain de manière structurée. Un module dédié à la création et au suivi des offres de

prix a également été développé, avec la génération automatique de documents conformes aux exigences locales, notamment en ce qui concerne la TVA et la devise marocaine (dirham). La gestion de l'inventaire a été centralisée afin d'assurer un meilleur suivi des ressources matérielles et des stocks disponibles. Enfin, l'exploitation des données issues de l'ensemble de ces modules permet d'alimenter des tableaux de bord dynamiques et des rapports synthétiques, facilitant ainsi la prise de décision et le pilotage des activités. L'interface utilisateur a été conçue pour être à la fois intuitive et accessible, garantissant une adoption rapide et une utilisation efficace par l'ensemble des acteurs de la ferme expérimentale.

Démarche Générale

Pour atteindre ces objectifs, une démarche méthodologique structurée a été adoptée tout au long de ce projet de fin d'études. Elle a débuté par une analyse approfondie des besoins fonctionnels et non fonctionnels, réalisée en concertation avec les différents acteurs de la ferme expérimentale. Cette phase a permis de définir précisément les attentes et les contraintes du futur système. Sur la base de cette analyse, une conception technique détaillée a été élaborée, incluant la modélisation des données, l'architecture logicielle et le choix des technologies adaptées. La phase de développement s'est ensuite articulée autour de la mise en œuvre du backend (API, base de données, logique métier) et du frontend (interface utilisateur, gestion des rôles, expérience utilisateur), en veillant à garantir la sécurité, la performance et l'évolutivité de la solution. Des tests fonctionnels et techniques ont été menés afin de valider la conformité du système aux exigences initiales. Enfin, une réflexion sur les perspectives d'évolution a été intégrée, afin d'anticiper les besoins futurs de la ferme expérimentale et d'assurer la pérennité de la plateforme.

Structure du Rapport

Pour présenter de manière claire et structurée notre démarche, ce rapport s'articule autour de plusieurs chapitres complémentaires. Nous débuterons par une analyse approfondie des besoins et des spécifications du système, afin de poser les fondations du projet. Le chapitre suivant sera consacré à la conception technique et à la modélisation de la solution, où nous détaillerons les choix d'architecture et de technologies. Ensuite, la phase d'implémentation et de développement de la plateforme sera décrite, illustrant la manière dont la conception a été traduite en un produit fonctionnel. Une fois la solution présentée, nous aborderons les résultats obtenus, leur validation et leur évaluation au regard des objectifs initiaux. Enfin, nous conclurons ce rapport en esquissant les perspectives d'évolution possibles et en dressant un bilan global du travail accompli.

Chapitre 1

Analyse des Besoins et Spécifications

Introduction

Ce chapitre présente une analyse détaillée des besoins du système de gestion de la ferme expérimentale, en s'appuyant sur les attentes des différents utilisateurs et sur les contraintes du contexte métier. Il expose d'abord les profils utilisateurs et leurs rôles, puis détaille les besoins fonctionnels et non fonctionnels, avant de présenter la modélisation du système à travers le modèle entité-association et l'analyse des flux opérationnels.

1.1 Description des utilisateurs et des rôles

La plateforme s'organise autour de trois profils d'utilisateurs, aux prérogatives nettement définies afin d'assurer la sécurité des données et l'efficacité des opérations.

La plateforme s'adresse à trois profils d'utilisateurs principaux, chacun disposant de droits et de responsabilités spécifiques afin d'assurer une gestion sécurisée et efficace de la ferme expérimentale. L'administrateur occupe un rôle central, supervisant l'ensemble du système. Il gère les comptes utilisateurs, valide les demandes de réservation de terrain ainsi que les différentes fiches administratives, attribue les superviseurs aux projets, et pilote la gestion de l'inventaire, des offres de prix et des ordres de service. Il bénéficie d'un accès complet à tous les tableaux de bord, statistiques et fonctionnalités d'export de données. Le superviseur, quant à lui, assure le suivi opérationnel des projets qui lui sont confiés. Il peut consulter et suivre l'avancement des projets, ajouter des notes de suivi, mettre à jour les informations relatives aux projets, et accéder à un tableau de bord dédié regroupant les échéances et statistiques pertinentes. Il a également la possibilité de consulter les réservations qui lui sont attribuées. Enfin, l'utilisateur final (qu'il s'agisse d'un chercheur, d'un technicien ou d'un étudiant) utilise la plateforme pour mener à bien ses projets. Il peut s'inscrire et se connecter, soumettre des demandes de réservation de parcelles ou de champs, consulter l'historique de ses propres

demandes et projets, suivre l'état d'avancement de ses démarches, et recevoir des notifications concernant le statut de ses demandes. Grâce à l'intégration des modules d'offres de prix et d'ordres de service, l'utilisateur final bénéficie d'un environnement complet pour la gestion de ses activités au sein de la ferme expérimentale.

Tableau récapitulatif des droits par rôle

Fonctionnalité	Administrateur	Superviseur	Utilisateur final
Gérer les utilisateurs	✓	–	–
Gérer les projets	✓	✓*	✓**
Gérer l'inventaire	✓	–	–
Gérer les réservations	✓	✓*	✓**
Gérer les offres de prix	✓	–	–
Gérer les ordres de service	✓	–	–
Accès au tableau de bord	✓	✓	✓

* Superviseur : uniquement sur les projets/réservations qui lui sont attribués.

** Utilisateur final : uniquement sur ses propres projets/réservations.

TABLE 1.1 – Récapitulatif des droits par rôle utilisateur

1.2 Besoins fonctionnels

L'analyse des besoins fonctionnels a été menée en tenant compte des trois profils d'utilisateurs principaux, afin de définir précisément les services et interactions attendus de la plateforme. Pour l'utilisateur final, la solution permet l'inscription, la connexion et la gestion de son profil personnel. Il peut soumettre de nouvelles demandes de projet à travers un formulaire en ligne détaillé, incluant la description du projet, le choix de la parcelle, les dates prévisionnelles et l'ajout de pièces jointes si nécessaire. L'utilisateur final a également la possibilité de consulter l'historique de ses demandes, de suivre en temps réel le statut de chacune ("Soumise", "Approuvée", "Rejetée") et de recevoir des notifications automatiques à chaque changement d'état. Un tableau de bord personnel lui offre une vue synthétique de ses activités et de l'avancement de ses projets.

Du côté du superviseur, la plateforme met à disposition une interface dédiée pour consulter la liste des projets et réservations qui lui sont attribués. Il peut suivre l'avancement des projets, ajouter des notes de suivi, mettre à jour le statut des différentes étapes et accéder à un tableau de bord spécifique regroupant les échéances à venir et les statistiques pertinentes. Le superviseur a également accès aux informations détaillées sur les utilisateurs et les parcelles associées à ses projets, ce qui facilite la coordination et le pilotage opérationnel.

Pour l'administrateur, la plateforme offre une gestion complète des comptes utilisateurs, incluant la création, la modification, la suppression et l'attribution des rôles. Il est responsable de la validation ou du rejet des demandes, et peut également demander des informations complémentaires si nécessaire. L'administrateur attribue les superviseurs aux projets, génère les ordres de service, gère les offres de prix et supervise l'inventaire (ajout, modification, suppression d'articles). Enfin, il dispose d'un accès total à l'ensemble des tableaux de bord, statistiques et fonctionnalités d'export de données, lui permettant d'assurer un suivi global et une gestion optimale de la ferme expérimentale.

1.3 Besoins Non Fonctionnels

Au-delà des fonctionnalités spécifiques, le succès de la plateforme dépend également du respect d'un ensemble de critères qualitatifs, ou besoins non fonctionnels. Ces derniers définissent la manière dont le système doit opérer et garantissent une expérience utilisateur de haute qualité.

Performance

Le système doit offrir une expérience utilisateur fluide et réactive. Les temps de chargement des pages et de réponse aux requêtes doivent être optimaux, même en cas de forte charge.

Sécurité

La sécurité est une priorité absolue. La plateforme doit garantir la confidentialité, l'intégrité et la disponibilité des données des utilisateurs et des projets. Cela inclut la mise en place d'un système d'authentification robuste (tel que JWT), le chiffrement des données sensibles comme les mots de passe, et la protection contre les vulnérabilités web courantes (par exemple, les injections SQL ou les attaques XSS).

Ergonomie et Utilisabilité

L'interface utilisateur doit être intuitive, claire et facile à prendre en main pour tous les profils. Les formulaires doivent être simples à remplir et les informations présentées de manière compréhensible.

Compatibilité et Accessibilité

Étant une application web, la plateforme devra être entièrement compatible avec les principaux navigateurs web modernes (Google Chrome, Mozilla Firefox, Safari). De plus, une conception adaptative (responsive design) est requise pour garantir une expérience utilisateur satisfaisante non seulement sur ordinateur de bureau, mais également sur les appareils mobiles tels que les

tablettes et les smartphones, permettant ainsi une utilisation pratique en situation de mobilité sur le terrain.

Maintenabilité et Évolutivité

Le code source doit être propre, bien documenté et modulaire pour faciliter la maintenance, la correction de bugs et l'ajout de nouvelles fonctionnalités. L'architecture doit permettre une montée en charge sans refonte majeure.

1.4 Modélisation du système

Présentation du modèle entité-association (ERD)

La conception du modèle de données est une étape fondamentale de notre projet, car elle constitue le squelette architectural sur lequel repose l'ensemble de la plateforme. Pour garantir la cohérence, l'intégrité et l'évolutivité des données, nous avons élaboré un modèle Entité-Association (ERD) détaillé. Ce modèle (présenté à la Figure 1.1) structure l'information en six entités principales, dont les relations logiques ont été définies pour refléter précisément le flux opérationnel de la gestion des projets au sein de la ferme expérimentale.

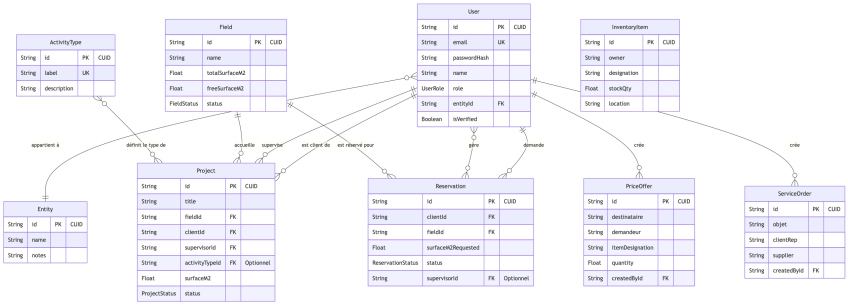

FIGURE 1.1 – Diagramme Entité-Association (ERD) du système de gestion

Description des Entités Principales

Le modèle de données a été conçu comme un écosystème intégré pour piloter l'ensemble des activités de la ferme expérimentale. Chaque entité joue un rôle spécifique tout en interagissant

pour assurer une gestion cohérente et efficace.

Acteurs et Espaces

La gestion des acteurs est centralisée dans l'entité Utilisateur, qui authentifie chaque personne interagissant avec le système et définit ses permissions via un rôle (client, superviseur, admin). Parallèlement, l'entité Parcelle (Field) modélise l'espace physique de la ferme, offrant une vue en temps réel sur la disponibilité et l'occupation des surfaces cultivables.

Du Concept à la Réalisation : Projets et Réservations

Toute nouvelle initiative est initiée par une Demande de Réservation, qui matérialise la requête d'un client pour une parcelle spécifique. Une fois validée, cette demande peut évoluer en Projet. L'entité Projet est la véritable clé de voûte opérationnelle : elle concrétise l'activité en liant le client, le superviseur assigné, la parcelle allouée et le type d'activité, tout en suivant son avancement.

Gestion Administrative et Financière

Pour assurer la traçabilité et le suivi, deux entités documentaires sont cruciales. L'Ordre de Service planifie les interventions concrètes sur le terrain (fournisseur, dates, objet), tandis que l'Offre de Prix formalise les aspects financiers en générant des devis et en suivant les coûts associés à chaque prestation.

Gestion des Ressources Matérielles

Enfin, aucune opération ne serait possible sans un suivi rigoureux du matériel. L'entité Inventaire remplit ce rôle en centralisant l'état des stocks, la localisation des équipements et leurs conditions, permettant ainsi d'anticiper les besoins et d'éviter les ruptures.

Analyse du flux opérationel et des relations

Le schéma relationnel mis en place permet de suivre l'intégralité du cycle de vie d'un projet, depuis la soumission d'une demande jusqu'à la réalisation effective des interventions sur le terrain. Le processus débute lorsqu'un utilisateur soumet une demande de réservation pour une parcelle spécifique. Après validation par l'administrateur, cette demande peut donner lieu à la création d'un ou plusieurs projets, chacun étant supervisé et associé à une activité précise. Chaque projet peut être décomposé en plusieurs ordres de service, permettant de planifier et de suivre les différentes tâches opérationnelles nécessaires à sa réalisation. À chaque ordre de service peuvent être rattachées des offres de prix pour la gestion des aspects financiers, ainsi que des éléments d'inventaire pour le suivi des ressources matérielles mobilisées. Cette architecture relationnelle assure une traçabilité fine des actions, des coûts et des ressources,

tout en garantissant l'intégrité des données. Elle offre ainsi une base solide et évolutive pour le pilotage précis des activités de la ferme expérimentale, en répondant efficacement aux enjeux de centralisation, de suivi et d'optimisation des processus identifiés lors de l'analyse initiale.

Conclusion du chapitre

Ce chapitre a permis de poser les bases du projet en identifiant précisément les utilisateurs, les besoins fonctionnels et non fonctionnels, ainsi que la structure des données et les principaux flux opérationnels. Ces éléments serviront de fondation à la conception technique et à l'implémentation détaillée présentées dans les chapitres suivants.

Chapitre 2

Conception technique et modélisation de la solution

Introduction

La réussite d'un projet de digitalisation repose sur une conception technique rigoureuse, capable de garantir la robustesse, la sécurité et l'évolutivité du système. Ce chapitre présente l'architecture générale de la solution développée pour la gestion de la ferme expérimentale, en détaillant les choix technologiques, la modélisation technique et les principes de sécurité adoptés. Après avoir analysé les besoins fonctionnels et non fonctionnels, une attention particulière a été portée à la structuration des différentes couches du système, à la définition des interactions entre les composants (frontend, backend, base de données) et à l'ergonomie de l'interface utilisateur. Les schémas et diagrammes présentés dans ce chapitre illustrent la vision globale de la solution et servent de fondation à l'implémentation détaillée qui sera exposée dans le chapitre suivant.

2.1 Architecture générale du système

La solution développée repose sur une architecture en couches, favorisant la séparation des responsabilités, la maintenabilité et l'évolutivité du système. L'application est structurée autour de trois couches principales : la couche présentation (frontend), la couche logique métier (backend/API) et la couche données (base de données).

2.1.1 Schéma d'architecture globale

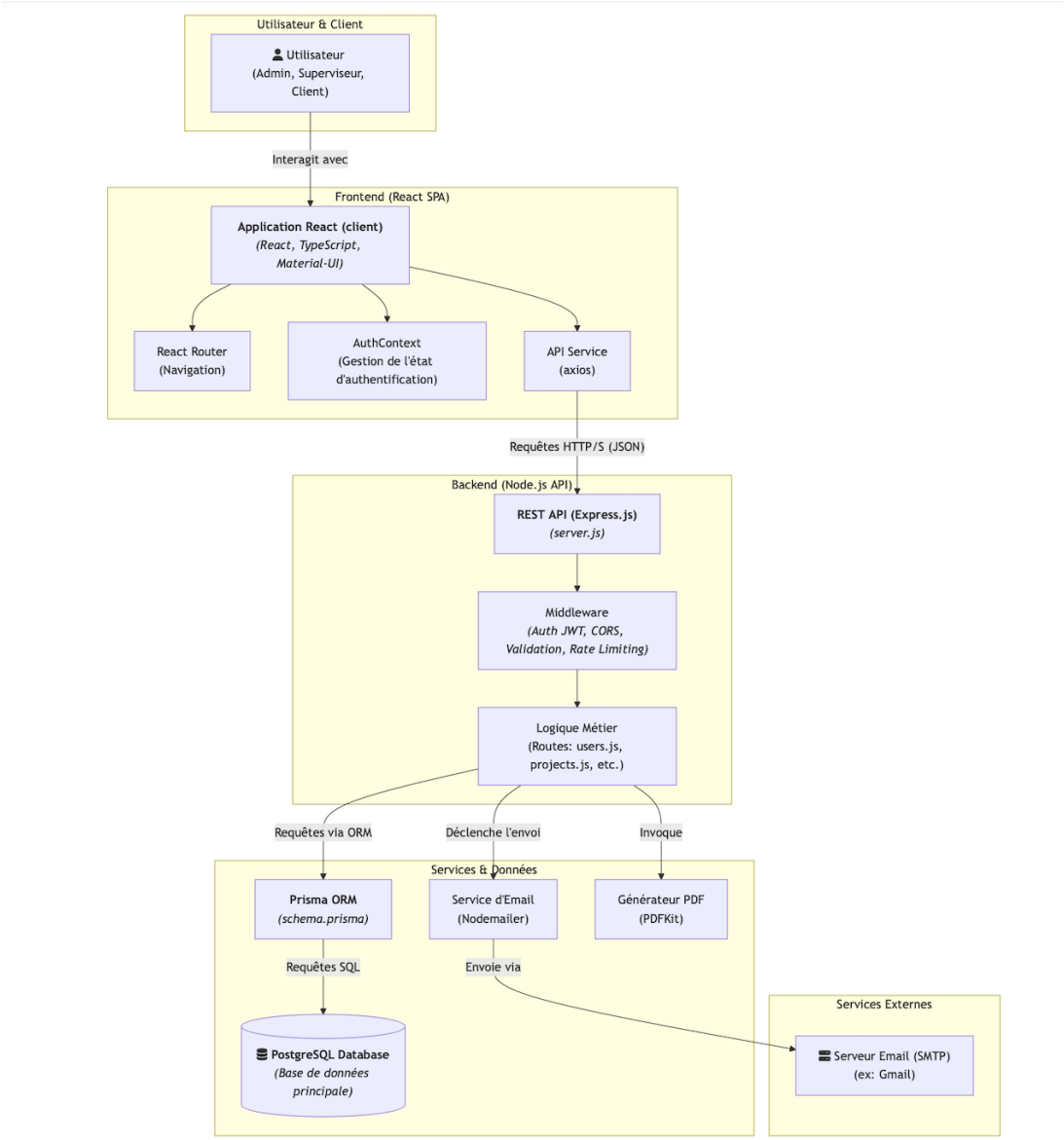

FIGURE 2.1 – Architecture générale de la plateforme de gestion de la ferme expérimentale

La figure ci-dessus illustre l'architecture trois-tiers de notre application. L'Utilisateur interagit avec un Frontend en React, qui communique via des requêtes HTTP/S avec un Backend en Node.js/Express.js. Ce dernier orchestre la logique métier, gère l'authentification et interagit avec une couche de persistance composée d'une base de données PostgreSQL gérée par l'ORM Prisma, ainsi que des services utilitaires pour l'envoi d'emails et la génération de PDF.

2.1.2 Description des couches

Couche présentation (Frontend) :

La couche de présentation (Frontend) constitue le point d'interaction direct avec l'utilisateur. Développée en tant que Single-Page Application (SPA) avec React et TypeScript, elle offre une expérience utilisateur réactive et moderne. Cette couche est responsable de l'affichage dynamique des vues, de la gestion des formulaires et de la navigation via React Router. La communication avec le serveur est centralisée par un service API basé sur axios, tandis que l'état d'authentification de l'utilisateur est maintenu de manière globale grâce au AuthContext.

Couche logique métier (Backend/API) :

Au cœur du système se trouve la couche applicative (Backend), une API RESTful robuste construite avec Node.js et le framework Express.js. Elle sert de chef d'orchestre en traitant toutes les requêtes provenant du frontend. Chaque requête entrante traverse une série de middlewares qui assurent des fonctions cruciales : la sécurité (avec Helmet et CORS), l'authentification basée sur les jetons JWT, la validation des données et la limitation du taux de requêtes pour prévenir les abus. La logique métier, encapsulée dans des modules de routes distincts (pour les utilisateurs, les projets, etc.), prend ensuite le relais pour exécuter les opérations demandées.

Couche données (Base de données) :

Enfin, la couche de services et de persistance soutient l'ensemble des opérations du backend. L'interaction avec la base de données PostgreSQL, où toutes les données de l'application sont stockées de manière persistante, est gérée de manière sécurisée et efficace par l'ORM Prisma. Ce dernier fait office de pont entre le code JavaScript et les requêtes SQL, tout en garantissant l'intégrité des données. Cette couche inclut également des services utilitaires essentiels, tels que Nodemailer pour l'envoi de notifications par email et PDFKit pour la génération à la volée de documents officiels comme les offres de prix. Ce découplage clair entre les couches assure une grande flexibilité et facilite la maintenance du

2.1.3 Pattern architectural

L'application repose sur une architecture web moderne, fondée sur une séparation claire entre le frontend et le backend, selon le modèle "SPA + API REST". Ce choix garantit la flexibilité, la maintenabilité et l'évolutivité de la plateforme.

Frontend (React/TypeScript)

La couche frontend est développée en React avec TypeScript. Elle prend en charge l'interface utilisateur, la navigation (React Router), la gestion de l'état d'authentification (AuthContext), et

la communication avec le backend via des appels API HTTP sécurisés (Axios). Toute la logique d'affichage, la collecte des saisies utilisateurs et l'expérience utilisateur sont gérées côté client. Le frontend ne contient aucune logique métier critique : il se contente de présenter les données et de transmettre les actions de l'utilisateur au backend.

Backend (Node.js/Express)

Le backend est construit avec Node.js et Express. Il expose des API REST qui reçoivent les requêtes du frontend, appliquent la logique métier (validation, autorisation, règles spécifiques), orchestrent les interactions avec la base de données via Prisma ORM, et renvoient les réponses appropriées. Le backend gère également la sécurité (authentification JWT, contrôle d'accès par rôle), la génération de documents PDF, l'envoi d'emails et la gestion des fichiers. Les contrôleurs Express regroupent la logique métier et la validation, mais il n'y a pas de séparation stricte "Modèle-Vue-Contrôleur" comme dans le pattern MVC classique.

Base de données et services

La persistance des données est assurée par une base PostgreSQL, accessible via Prisma ORM. Les opérations de lecture, d'écriture et de mise à jour sont centralisées dans cette couche, garantissant la cohérence et la sécurité des transactions. Des services utilitaires (génération de PDF, emailing) et des services externes (serveur SMTP, stockage de fichiers) complètent l'architecture.

Découplage et avantages

La communication entre le frontend et le backend s'effectue exclusivement via des API REST sécurisées. Ce découplage strict permet à chaque couche d'évoluer indépendamment, de renforcer la sécurité, et de faciliter l'intégration de nouveaux modules ou clients (applications mobiles, interfaces d'administration dédiées, etc.).

Ce modèle "SPA + API REST" est aujourd'hui la norme pour les applications web modernes, car il offre une grande souplesse, une meilleure testabilité et une évolutivité optimale.

2.2 Choix technologiques

Le choix des technologies utilisées pour la réalisation de la plateforme a été guidé par des critères stricts de robustesse, de maintenabilité, de sécurité et d'évolutivité. Chaque composant de l'architecture a été sélectionné non seulement pour ses mérites individuels, mais aussi pour sa capacité à s'intégrer harmonieusement au sein d'un écosystème technique cohérent et moderne, capable de répondre aux besoins fonctionnels et non-fonctionnels identifiés lors de l'analyse préalable.

2.2.1 Frontend

Le frontend a été conçu comme une Single-Page Application (SPA), offrant une expérience utilisateur fluide et réactive, sans rechargements de page.

React & TypeScript :

La synergie entre ces deux technologies constitue le fondement de notre interface. React a été privilégié pour son modèle de composants réutilisables et son écosystème mature, qui permettent de construire des interfaces complexes de manière structurée. L'intégration de TypeScript était un choix stratégique majeur. Pour un système gérant des données aussi critiques et des vues différentes selon les rôles, TypeScript apporte une robustesse indispensable en garantissant un typage fort de bout en bout. Cela permet une détection précoce des erreurs lors du développement, une autocomplétion intelligente et une documentation implicite du code, facilitant grandement la maintenance et l'évolution de l'application

Material-UI (MUI) :

Pour accélérer le développement et garantir une cohérence visuelle, nous avons adopté la bibliothèque de composants MUI. Elle offre un ensemble complet de composants UI (tableaux de données, formulaires, modales, etc.) respectant les meilleures pratiques en matière de design, d'accessibilité et de réactivité (responsive design), nous permettant de nous concentrer sur la logique métier plutôt que sur le style de base.

Axios & React Router :

Ces deux outils sont essentiels à l'écosystème. React Router gère la navigation et le routage côté client de manière transparente. Axios a été choisi pour sa simplicité et sa puissance dans la gestion des requêtes HTTP vers l'API. Nous avons notamment tiré parti de ses intercepteurs pour attacher automatiquement le jeton d'authentification JWT à chaque requête, centralisant ainsi la logique d'authentification des appels API.

2.2.2 Backend

Le backend est architecturé comme une API RESTful sécurisée, performante et modulaire.

Node.js & Express.js :

L'environnement d'exécution Node.js a été choisi pour son modèle d'entrées/sorties non bloquant, ce qui le rend particulièrement performant pour les applications web qui, comme la nôtre, effectuent de nombreuses opérations d'attente (appels à la base de données, lectures de fichiers).

Express.js est le framework qui structure notre serveur, facilitant la création de routes d'API claires et la gestion d'une chaîne de middlewares pour traiter les requêtes.

Prisma ORM :

C'est un des choix technologiques les plus structurants du projet. Prisma, en tant qu'ORM (Object-Relational Mapper) de nouvelle génération, agit comme une couche d'abstraction sécurisée et typée entre notre application et la base de données PostgreSQL. Ses atouts majeurs sont :

→ **Schéma Déclaratif :** Le fichier `schema.prisma` est la source unique de vérité pour notre modèle de données.

→ **Sécurité de Type de bout en bout :** La combinaison de Prisma avec TypeScript garantit que les requêtes et leurs résultats sont entièrement typés, éliminant une classe entière d'erreurs d'exécution.

→ **Gestion des Migrations :** Prisma gère automatiquement la génération et l'application des migrations SQL, assurant une synchronisation parfaite entre le modèle de données et la base de données réelle.

JWT (JSON Web Token) et sécurité :

L'authentification est gérée via des JWT. Ce standard permet de créer des sessions sans état (stateless), où le serveur n'a pas besoin de stocker d'information de session. Chaque jeton, signé numériquement, contient les informations de l'utilisateur (ID, rôle) et est validé à chaque requête par un middleware dédié. La sécurité est renforcée par une suite de middlewares essentiels : Helmet (protection des en-têtes HTTP), CORS (gestion des requêtes inter-domaines), Morgan (journalisation) et express-rate-limit (prévention des attaques par force brute).

Utilitaires (PDFKit & Nodemailer)

Pour répondre à des besoins fonctionnels spécifiques, nous avons intégré PDFKit pour la génération dynamique côté serveur des documents administratifs (offres de prix, ordres de service), numérisant ainsi les formulaires F.47 et F.84. Nodemailer gère quant à lui l'envoi d'e-mails transactionnels (notifications de changement de statut, alertes) via un serveur SMTP.

2.2.3 Base de données et persistance

PostgreSQL :

Ce système de gestion de base de données relationnelle a été choisi pour sa réputation de fiabilité, sa robustesse face à de larges volumes de données et sa riche conformité au standard

SQL. Il garantit l'intégrité ACID (Atomicité, Cohérence, Isolation, Durabilité) des transactions, un critère indispensable pour notre application.

Système de fichiers local :

Pour les opérations de génération de rapports et d'export de données, le serveur utilise un répertoire de stockage local dédié (/uploads) pour créer temporairement les fichiers PDF et CSV avant de les transmettre à l'utilisateur.

2.2.4 Justification des choix

L'ensemble de ces technologies forme un écosystème cohérent qui répond à nos objectifs principaux :

Performance et Scalabilité :

La nature asynchrone de Node.js, couplée à la performance reconnue de PostgreSQL, assure des temps de réponse rapides et la capacité de gérer un nombre croissant d'utilisateurs et de données.

Sécurité :

Une approche de sécurité à plusieurs niveaux a été mise en place : validation des entrées (express-validator), hachage des mots de passe (bcrypt), authentification par JWT, protection des routes par rôles (RBAC), et protection contre les injections SQL grâce à Prisma.

Évolutivité et Maintenabilité :

L'architecture modulaire (composants React, routes Express), le typage fort de TypeScript et le schéma déclaratif de Prisma rendent le système facile à comprendre, à maintenir et à faire évoluer. L'ajout de nouvelles fonctionnalités ou la modification des entités de données sont des processus structurés et sécurisés.

Communauté et documentation :

Tous les outils sélectionnés bénéficient d'une large communauté active et d'une documentation abondante et de grande qualité. Cet aspect est crucial car il garantit un support durable, facilite la résolution de problèmes et assure la pérennité du projet.

2.3 Modélisation technique

Après avoir défini les besoins fonctionnels, la phase de modélisation technique a pour objectif de traduire ces besoins en une architecture logicielle et une structure de données concrètes. Cette section présente les diagrammes UML qui ont servi de plan directeur pour le développement de l'application.

Language UML

UML (Unified Modeling Language), ou Langage de Modélisation Unifié, est une notation graphique normalisée qui a pour but de modéliser des systèmes d'information, des logiciels ou des processus métier. Il offre un vocabulaire visuel commun pour représenter les différentes facettes d'un système, de sa conception fonctionnelle à son architecture technique. En utilisant une série de diagrammes structurels (comme le diagramme de classes) et comportementaux (comme les diagrammes de cas d'utilisation et de séquence), UML permet de clarifier les exigences, de faciliter la communication entre les parties prenantes et de fournir un plan directeur robuste pour le développement.

2.3.1 Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation a pour objectif de définir les fonctionnalités du système du point de vue des utilisateurs, appelés "acteurs". Il permet de délimiter le périmètre fonctionnel de l'application et de visualiser les interactions principales.

→ Administrateur : Utilisateur possédant les privilèges les plus élevés, responsable de la configuration et de la gestion globale du système.

→ Superviseur : Utilisateur interne (personnel de la ferme) en charge du suivi opérationnel des projets qui lui sont attribués.

→ Client : Utilisateur final (chercheur, partenaire) qui soumet des demandes de réservation et consulte l'état de ses projets.

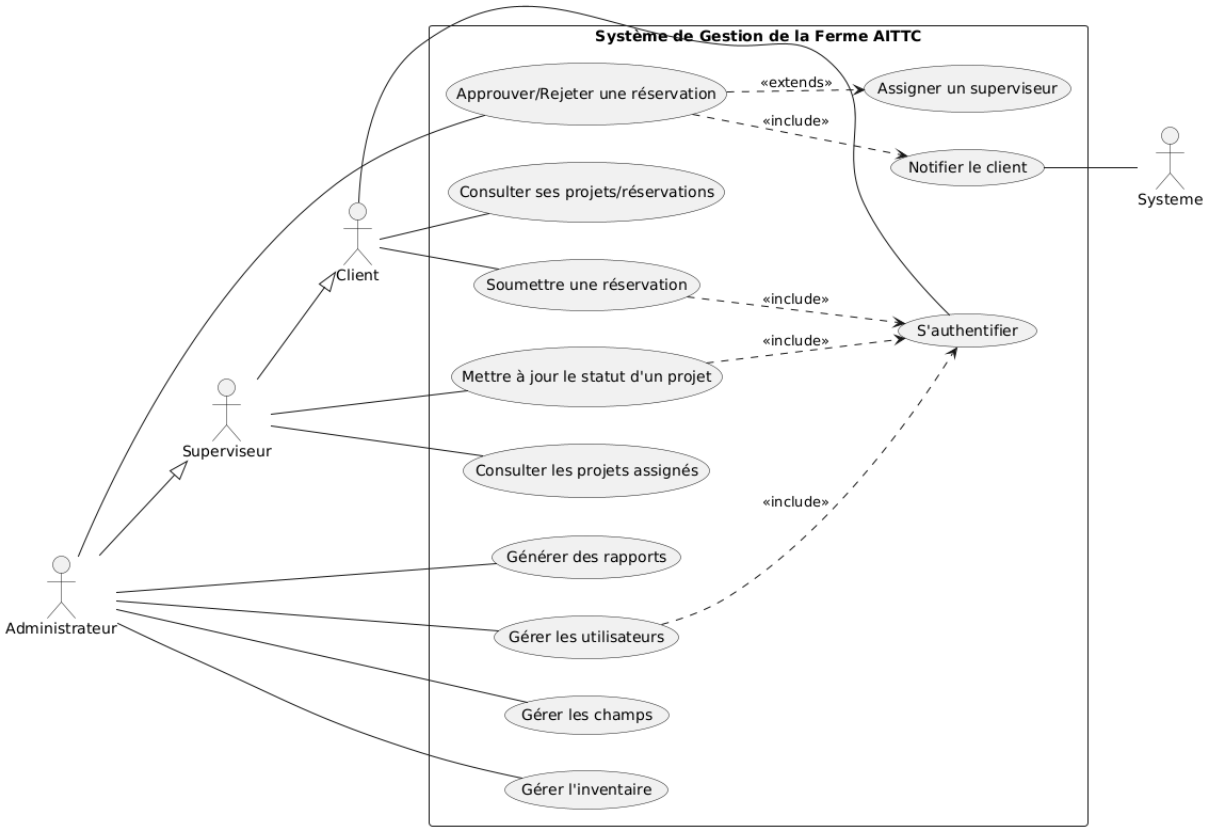

FIGURE 2.2 – Diagramme de cas d'utilisation des acteurs du système.

Description des Fonctionnalités

Ce diagramme de cas d'utilisation illustre les interactions entre les différents acteurs et les fonctionnalités principales du système. Trois types d'acteurs sont définis, chacun avec des permissions spécifiques : le Client, le Superviseur et l'Administrateur.

Le Client peut s'inscrire, soumettre des demandes de réservation de parcelles, et consulter l'état de ses propres projets et réservations. Le Superviseur est responsable du suivi opérationnel : il consulte les projets qui lui sont assignés et met à jour leur statut d'avancement.

L'Administrateur, quant à lui, dispose d'un accès complet à toutes les fonctionnalités de gestion. Il est le seul à pouvoir approuver ou rejeter les réservations, une action qui peut être étendue («extend») par l'assignation d'un superviseur et la notification du client. Il gère également l'ensemble des ressources du système : utilisateurs, champs, projets, inventaire, offres de prix et ordres de service.

Toutes les actions, à l'exception de l'inscription, requièrent une authentification préalable, comme le montre la relation d'inclusion («include») systématique vers le cas d'utilisation "S'authentifier". Enfin, l'acteur Système intervient pour des actions automatisées, telles que l'envoi de notifications. Ce modèle garantit une séparation claire des responsabilités et un contrôle d'accès

sécurisé à chaque fonctionnalité.

2.3.2 Diagramme de Classes

Le diagramme de classes modélise la structure statique du système. Il représente les entités de données, leurs attributs, ainsi que les relations et les contraintes qui les lient. Ce diagramme constitue le plan conceptuel de notre base de données PostgreSQL, implémenté via l'ORM Prisma.

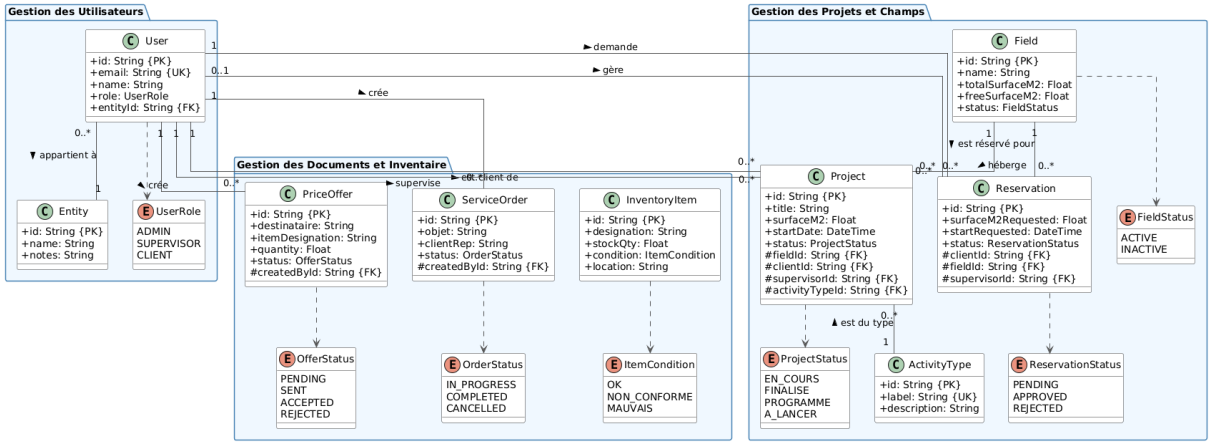

FIGURE 2.3 – Diagramme de Classes.

Description Fonctionelle

Ce diagramme de classes UML illustre la structure statique et les relations entre les principales entités de notre application, organisées en paquets logiques pour plus de clarté. Chaque classe, représentée par un rectangle, modélise un concept clé comme User ou Project, avec ses attributs listés à l'intérieur. La visibilité de ces attributs est indiquée par des symboles (+ pour public), et des stéréotypes comme PK ou FK précisent leur rôle en tant que clé primaire ou étrangère. Les associations entre les classes, matérialisées par des lignes pleines, sont définies par une cardinalité (ex : 1, 0..*) qui spécifie le nombre d'instances pouvant être liées. Par exemple, un User peut être lié à "zéro ou plusieurs" (0..*) Project. Les états et types prédéfinis, tels que les rôles ou les statuts, sont modélisés par des énumérations (ex : UserRole, ProjectStatus), auxquelles les classes sont liées par une relation de dépendance (ligne pointillée), garantissant ainsi l'intégrité et la cohérence des données à travers tout le système.

2.3.3 Diagramme de Séquence :

Ce diagramme illustre le comportement dynamique d'une interaction spécifique en montrant la chronologie des messages échangés entre les différents objets ou composants du système. Nous présentons ici le workflow d'approbation d'une réservation, un processus critique qui met en jeu plusieurs composants de notre architecture.

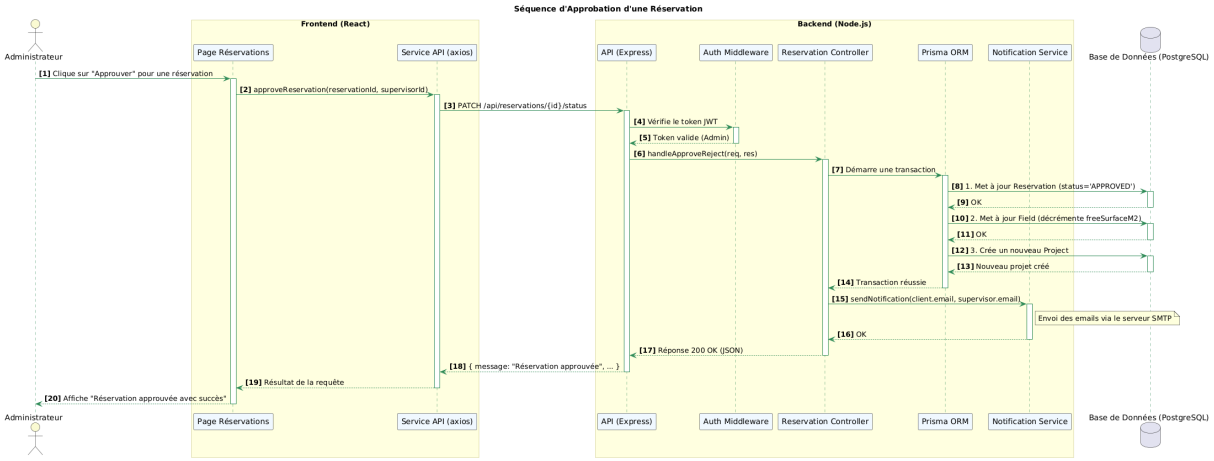

FIGURE 2.4 – Diagramme de Séquence.

Analyse des Interactions :

Le diagramme de séquence ci-contre décrit le processus d'approbation d'une réservation dans le système. En UML, ce type de diagramme permet de visualiser l'enchaînement chronologique des messages entre les différents acteurs et composants logiciels. Ici, l'administrateur initie l'action depuis l'interface React : un appel API est envoyé au backend Express, qui vérifie l'authentification (JWT) et le rôle. Le contrôleur met alors à jour la réservation, ajuste la surface du champ, crée un projet associé et envoie une notification email via le service SMTP. Une réponse de succès est ensuite renvoyée au frontend. Ce diagramme met en avant la séparation des responsabilités, la sécurité (vérification du JWT), la gestion transactionnelle et l'automatisation des notifications, tout en illustrant la rigueur de la modélisation UML (lignes de vie, messages, services externes).

2.4 Sécurité et gestion des accès :

La sécurité constitue un aspect critique de toute application web, particulièrement dans un contexte professionnel où la confidentialité et l'intégrité des données sont essentielles. La plateforme de gestion de la ferme expérimentale intègre plusieurs mécanismes de sécurité pour garantir la protection des informations et le contrôle d'accès approprié.

2.4.1 Gestion de l'authentification :

L'authentification des utilisateurs repose sur le standard JWT (JSON Web Token), une solution moderne et sécurisée pour la gestion des sessions. Le processus d'authentification s'articule autour de plusieurs étapes.

Lors de la connexion initiale, l'utilisateur fournit ses identifiants (email et mot de passe) via le formulaire de connexion. Le backend vérifie ces informations en comparant le hash du mot de

passe (généré avec bcrypt) avec celui stocké en base de données.

En cas de succès, le serveur génère un token JWT contenant les informations essentielles de l'utilisateur (ID, email, rôle) et signé avec une clé secrète. Ce token a une durée de vie limitée pour réduire les risques de compromission.

Le token JWT est stocké côté client dans le localStorage du navigateur et est automatiquement inclus dans l'en-tête Authorization de chaque requête API via un intercepteur Axios. Chaque requête vers une route protégée est interceptée par un middleware d'authentification qui vérifie la validité et la signature du token JWT avant d'autoriser l'accès.

L'accès à la plateforme est sécurisé par une page de connexion dédiée, où chaque utilisateur doit renseigner son identifiant et son mot de passe. Cette étape garantit l'authentification des utilisateurs et conditionne l'accès aux différentes fonctionnalités selon leur rôle.

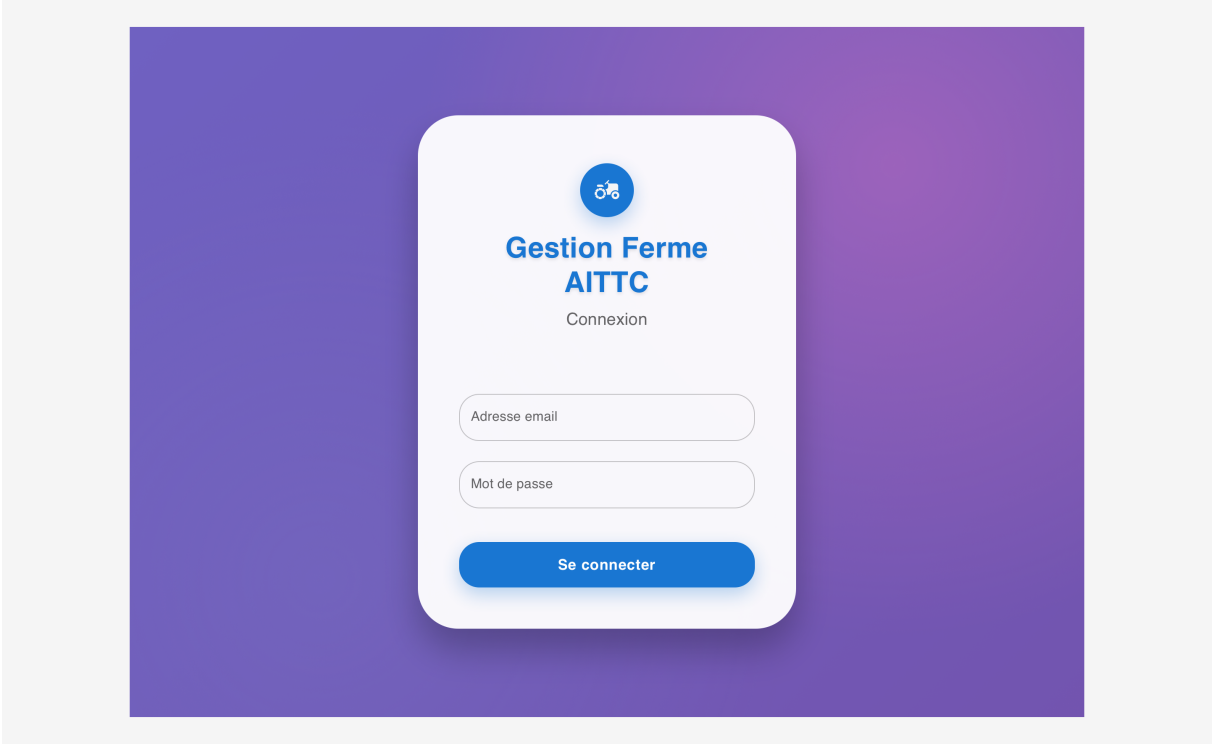

FIGURE 2.5 – Page de connexion de l'application

2.4.2 Gestion des rôles et permissions (RBAC) :

Le système de contrôle d'accès repose sur le modèle RBAC (Role-Based Access Control), permettant une gestion granulaire des permissions selon le rôle de l'utilisateur.

L'application définit trois niveaux de rôles distincts. Le rôle ADMIN dispose d'un accès complet à toutes les fonctionnalités et données du système. Le rôle SUPERVISOR bénéficie d'un accès limité aux projets et réservations qui lui sont attribués. Le rôle CLIENT se voit restreint à l'accès de ses propres données et projets.

Un middleware de contrôle des rôles, ‘requireRole()‘, vérifie que l’utilisateur authentifié possède le rôle requis pour accéder à une ressource spécifique. Ce middleware est appliqué aux routes sensibles telles que la gestion des utilisateurs ou la validation des demandes.

Le système intègre également un contrôle d’ownership via le middleware ‘requireOwnershipOrAdmin()‘. Ce mécanisme permet aux utilisateurs d’accéder uniquement à leurs propres ressources (un client ne peut voir que ses propres réservations), tout en autorisant les administrateurs à accéder à toutes les données.

2.4.3 Protection des routes et des données :

La protection des routes et des données s’appuie sur plusieurs mécanismes complémentaires pour assurer la sécurité globale du système.

Toutes les routes sensibles sont protégées par des middlewares d’authentification et d’autorisation. Les routes publiques (connexion, inscription) sont séparées des routes privées nécessitant une authentification, garantissant une séparation claire des responsabilités.

Chaque requête est validée côté serveur à l’aide de middlewares de validation (express-validator) pour prévenir les injections et les attaques par manipulation de données. Cette validation s’applique à tous les paramètres d’entrée avant leur traitement par la logique métier.

L’application intègre plusieurs middlewares de sécurité pour se protéger contre les attaques courantes. Helmet sécurise les en-têtes HTTP pour prévenir les attaques XSS et autres vulnérabilités. CORS contrôle l’accès cross-origin pour éviter les requêtes non autorisées. Un système de rate limiting limite le nombre de requêtes par utilisateur pour prévenir les attaques par déni de service. Morgan assure le logging des requêtes pour la surveillance et le débogage.

Les mots de passe sont hashés avec bcrypt (algorithme de hachage sécurisé) avant d’être stockés en base de données, garantissant qu’ils ne peuvent pas être récupérés même en cas de compromission de la base. Les tokens JWT ont une durée de vie limitée et peuvent être révoqués si nécessaire. Le système gère également la déconnexion en supprimant le token côté client.

Cette architecture de sécurité multi-couches assure la protection des données sensibles tout en maintenant une expérience utilisateur fluide et adaptée au rôle de chaque utilisateur.

2.5 Conception de l’interface utilisateur

L’interface utilisateur constitue le point de contact principal entre les utilisateurs et la plateforme. Sa conception a été guidée par des principes d’ergonomie, d’accessibilité et d’expérience utilisateur optimale, tout en s’adaptant aux différents profils d’utilisateurs identifiés.

2.5.1 Fonctionnalités principales accessibles à l'administrateur

L'administrateur dispose d'un accès complet à l'ensemble des modules de gestion de la plateforme : gestion des projets, des champs, des offres de prix, de l'inventaire, des utilisateurs et des réservations. Chaque module est accessible via la barre de navigation principale et propose des interfaces adaptées pour la consultation, la création, la modification et la suppression des données.

Gestion des projets et des champs

L'administrateur dispose d'interfaces dédiées pour la gestion centralisée des champs et des projets. Il peut ainsi consulter, ajouter, modifier ou supprimer les différentes parcelles agricoles (champs), en renseignant leur nom, localisation, surface et statut. De la même manière, il gère l'ensemble des projets associés à la ferme, en suivant leur avancement, le champ concerné, le client ou l'entité impliquée, ainsi que le superviseur responsable. Ces modules, organisés sous forme de tableaux interactifs, facilitent le suivi des activités et l'actualisation des données, tout en offrant des actions rapides et un accès simplifié à l'information.

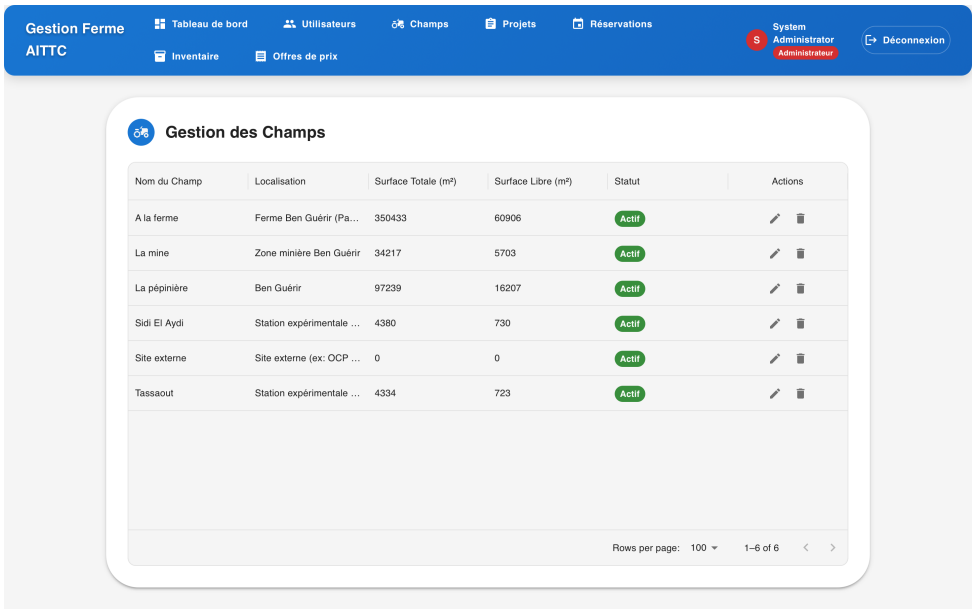

FIGURE 2.6 – Interface de gestion des champs

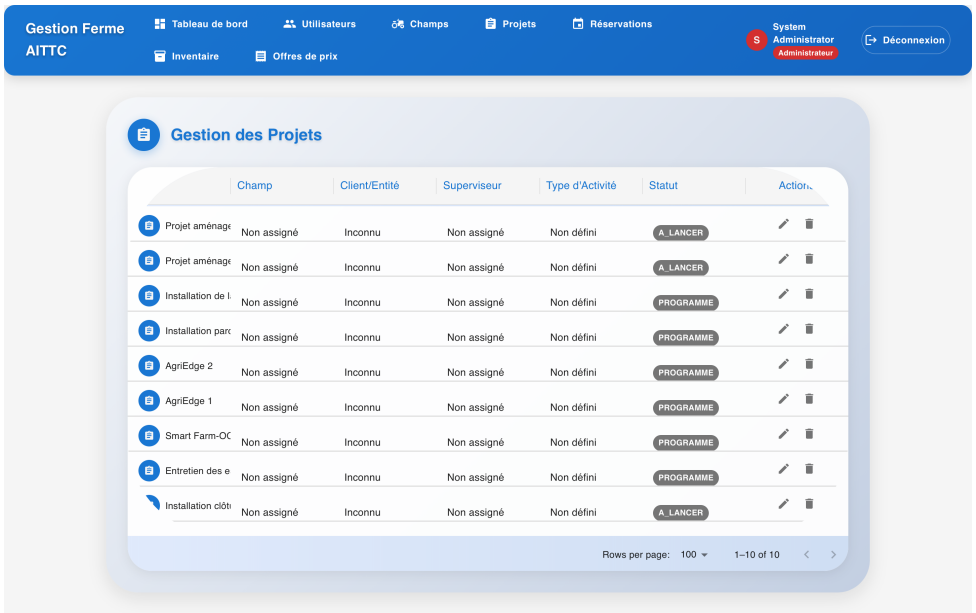

FIGURE 2.7 – Interface de gestion des projets

Gestion des Utilisateurs et des résevations

Les pages de gestion (utilisateurs, champs, projets, réservations, inventaire) adoptent une structure tabulaire avec des fonctionnalités de recherche, de filtrage et de pagination. Chaque page intègre des boutons d'action contextuels (créer, modifier, supprimer) selon les permissions de l'utilisateur. Les formulaires de saisie sont organisés de manière logique avec validation en temps réel et messages d'erreur explicites.

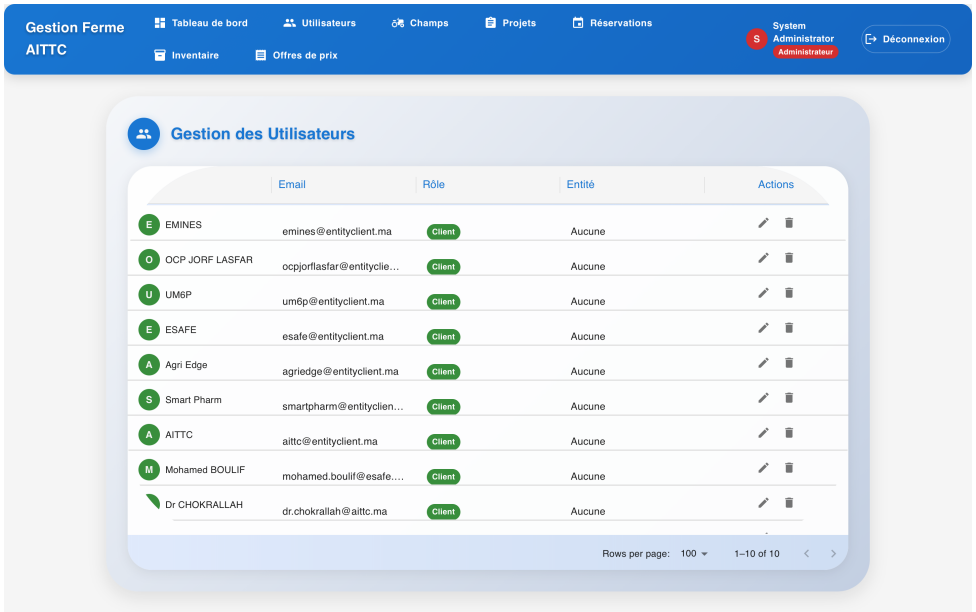

FIGURE 2.8 – Interface de gestion des utilisateurs avec tableau de données et actions contextuelles

L'administrateur gère l'ensemble des utilisateurs de la plateforme (création, modification, sup-

pression, attribution de rôles) ainsi que les réservations de champs, avec la possibilité d'approuver ou de rejeter les demandes selon la disponibilité et la politique de la ferme.

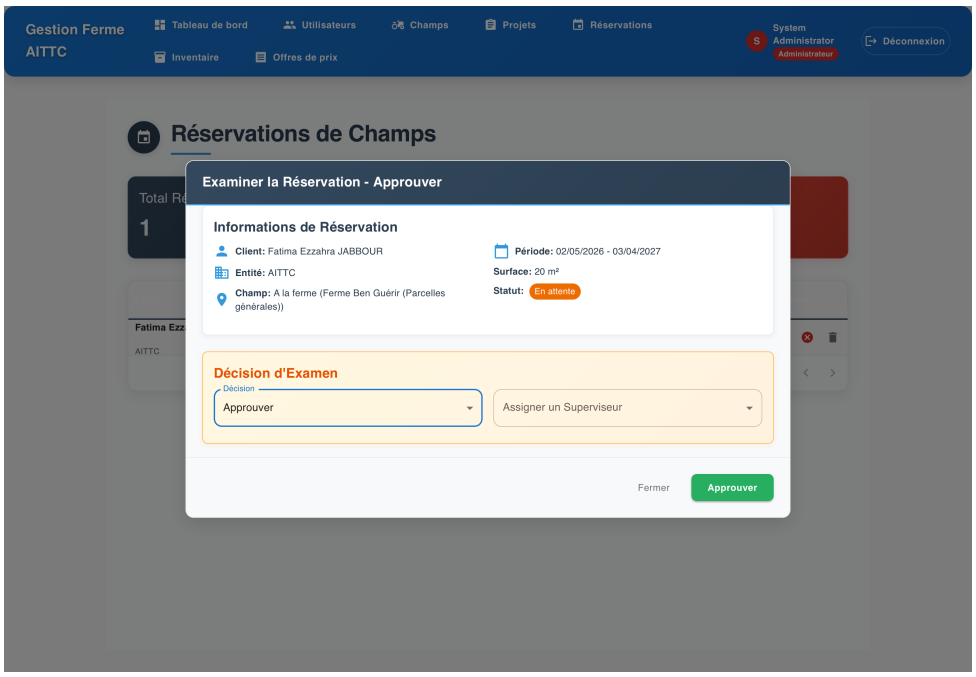

FIGURE 2.9 – Fenêtre d'examen et de validation d'une réservation par l'administrateur

Gestion des Inventaires

Ce module centralise la gestion des stocks de matériel, d'intrants ou de produits récoltés. L'administrateur peut, en théorie, suivre les quantités en stock, l'état des articles, leur emplacement, et effectuer des opérations d'ajout, de modification ou de suppression. Cependant, il est à noter que la fiche d'inventaire hebdomadaire fournie (voir annexe) ne comporte pas suffisamment de détails pour alimenter de manière pertinente et exhaustive ce module. Faute de données exploitables et structurées, l'inventaire reste donc vide dans la version actuelle de l'application. L'architecture du module a néanmoins été prévue pour permettre une intégration future dès que des données plus complètes seront disponibles.

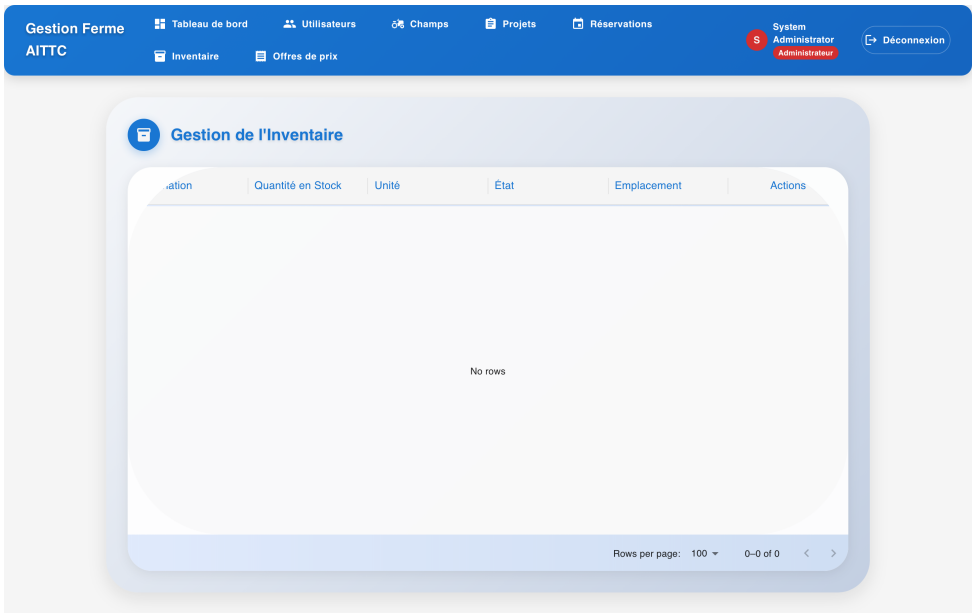

FIGURE 2.10 – Interface de gestion des inventaires

2.6 Fonctionnalités principales accessibles à l'utilisateur

Effectuer une réservation

L'utilisateur client a la possibilité d'effectuer une nouvelle réservation de champ via un formulaire dédié. Il peut ensuite suivre en temps réel l'état de ses réservations (en attente, approuvée ou rejetée) directement depuis son espace personnel, ce qui lui offre une visibilité immédiate sur l'avancement de ses demandes.

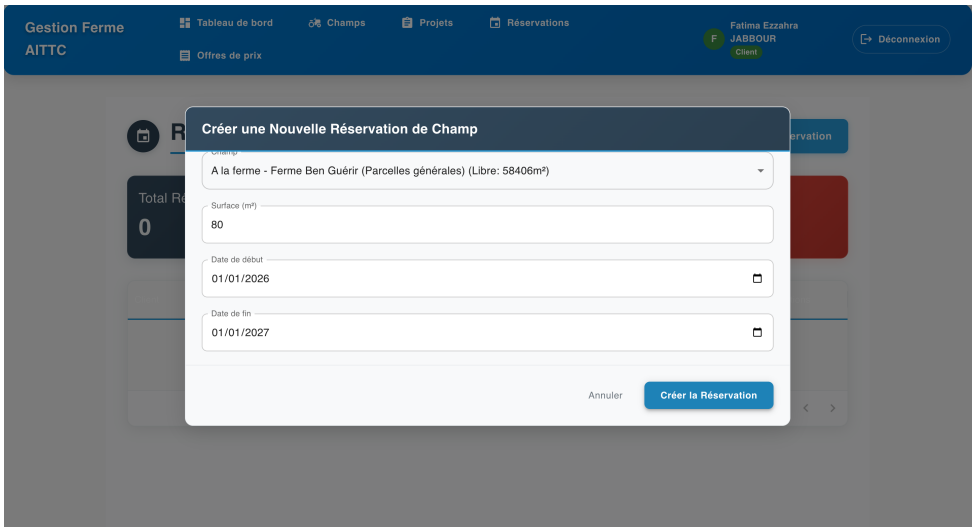

FIGURE 2.11 – Interface de réservation d'un champ

Suivi des Projets

L'utilisateur client a accès à une vue personnalisée des projets, limitée uniquement à ceux auxquels il est associé. Cette restriction garantit la confidentialité des données et permet au client de suivre l'avancement de ses propres projets sans accéder aux informations des autres entités. L'interface affiche la liste des projets concernés, leur statut et les principales informations utiles, offrant ainsi un suivi clair et sécurisé.

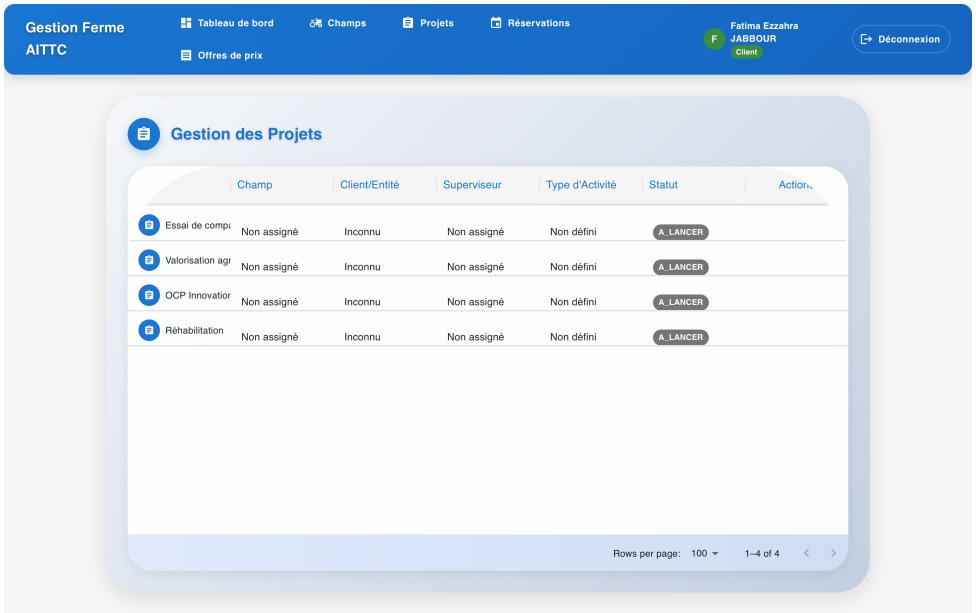

FIGURE 2.12 – Interface de la gestion des projets (clients)

Gestion des champs

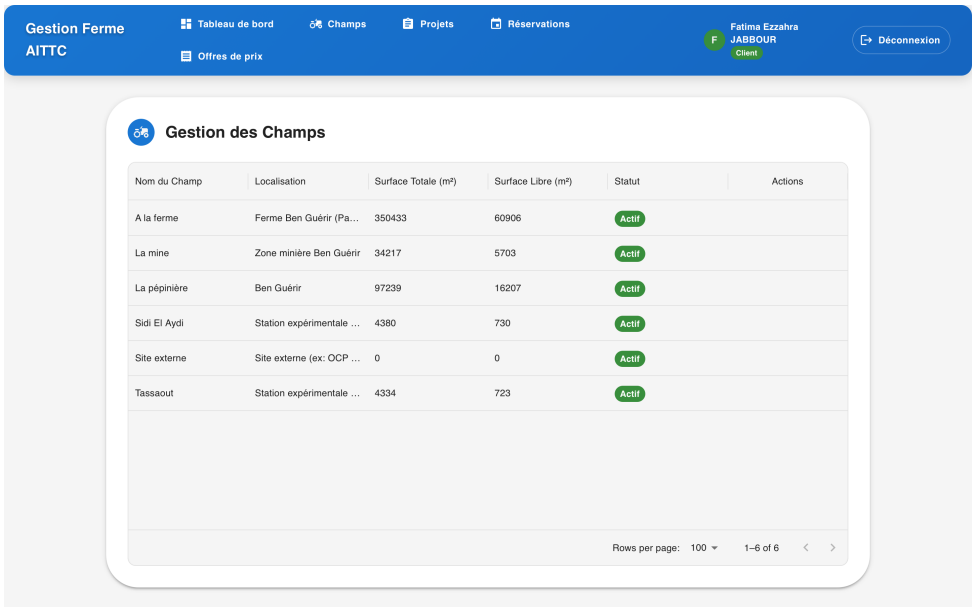

FIGURE 2.13 – Interface de la gestion des champs (clients)

L'interface "Gestion des Champs" permet à l'utilisateur client de consulter la liste des différentes parcelles agricoles disponibles, avec leurs principales caractéristiques : nom, localisation, surface totale et surface libre, ainsi que le statut d'activité. Cette vue offre une transparence sur l'état des champs et facilite la planification des réservations ou des projets selon la disponibilité des parcelles.

Gestion Offre Prix

Tant les administrateurs que les clients disposent d'un accès au module Offre de prix, leur permettant de créer, consulter et exporter des offres personnalisées. À travers un formulaire dédié, l'utilisateur saisit les informations du client, détaille les articles (désignation, quantité, prix unitaire, etc.) et obtient automatiquement le calcul des totaux avec la TVA. L'interface, identique pour les deux profils, a été conçue pour être intuitive et accessible, garantissant une expérience fluide quel que soit le rôle de l'utilisateur. Une fois l'offre complétée, elle peut être exportée au format PDF pour un partage simplifié.

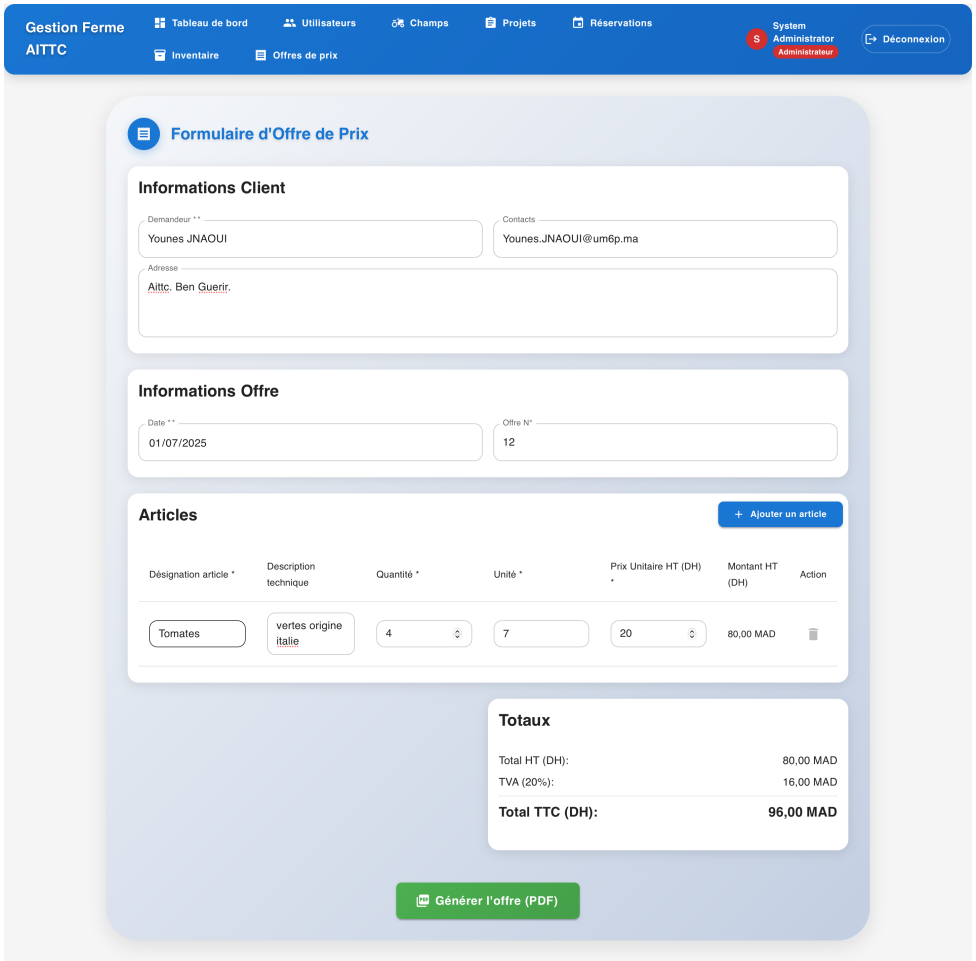

FIGURE 2.14 – Interface de gestion des offres prix

Remarque Générale

L'administrateur dispose de droits complets sur l'ensemble de la plateforme, tandis que le superviseur n'a accès qu'aux projets qui lui sont assignés et ne peut ni gérer les utilisateurs ni modifier la configuration globale.

2.6.1 Maquettes et wireframes

L'interface de la plateforme a été conçue en s'appuyant sur des principes de design moderne et professionnel. L'utilisation de Material-UI (MUI) comme bibliothèque de composants a permis d'assurer une cohérence visuelle et une expérience utilisateur uniforme.

Le tableau de bord principal offre une vue d'ensemble des informations essentielles selon le rôle de l'utilisateur. Pour les administrateurs, il présente des statistiques globales (nombre de réservations en attente, projets actifs, alertes d'inventaire) sous forme de cartes colorées et de graphiques. Les superviseurs accèdent à un tableau de bord focalisé sur leurs projets assignés et les échéances à venir. Les clients visualisent leurs propres réservations et projets en cours.

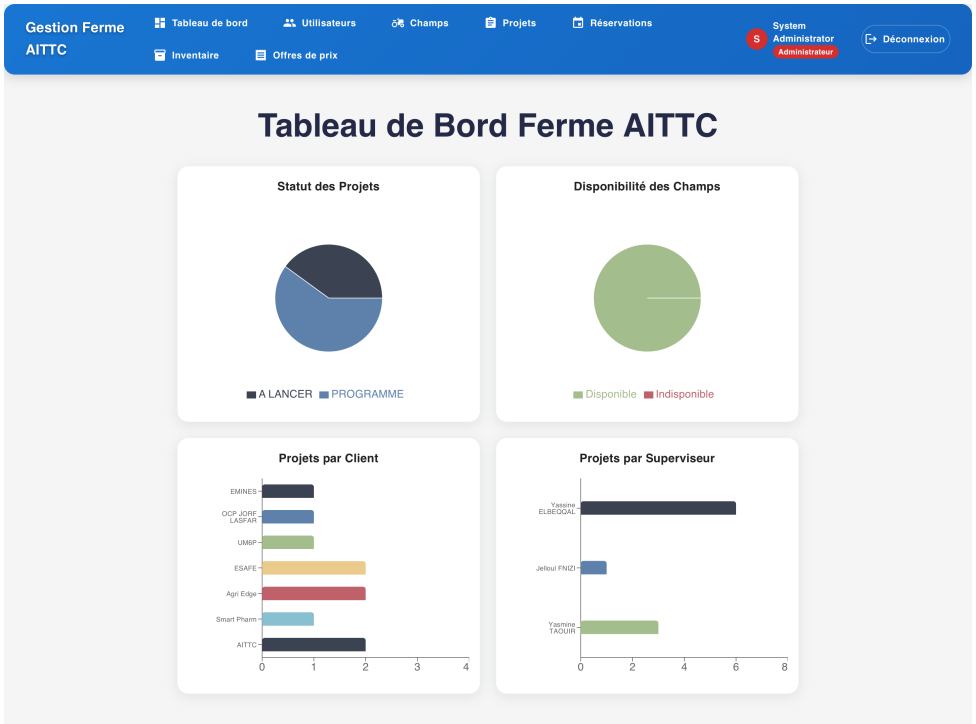

FIGURE 2.15 – Vue d'ensemble du tableau de bord administrateur avec statistiques et métriques clés

2.6.2 Principes d'ergonomie et d'accessibilité

L'ergonomie de l'interface a été pensée pour minimiser la charge cognitive des utilisateurs et optimiser leur efficacité. La navigation est intuitive avec une barre de navigation latérale ou supérieure claire, permettant un accès rapide aux différentes sections de l'application.

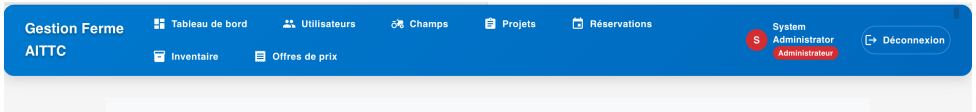

FIGURE 2.16 – Navigation adaptative selon le rôle utilisateur (ADMIN)

L'utilisation de couleurs cohérentes et contrastées améliore la lisibilité et facilite l'identification des éléments importants. Les statuts des projets et réservations sont représentés par des codes couleur standardisés (vert pour "approuvé", orange pour "en attente", rouge pour "rejeté"). Les icônes Material-UI sont utilisées de manière cohérente pour représenter les actions et les types de contenu.

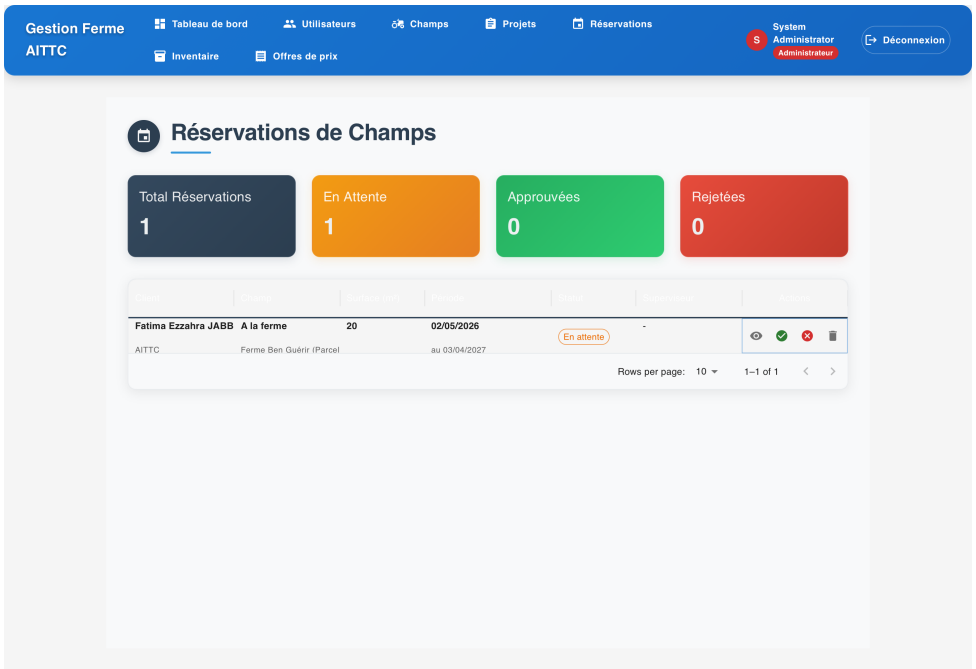

FIGURE 2.17 – Représentation visuelle des statuts par codes couleur (PENDING, APPROVED, REJECTED)

L'accessibilité a été prise en compte dès la conception. L'interface respecte les standards WCAG (Web Content Accessibility Guidelines) avec des contrastes de couleurs appropriés, des textes alternatifs pour les images, et une navigation au clavier possible. Les formulaires incluent des labels explicites et des messages d'erreur clairs pour faciliter l'utilisation par tous les utilisateurs.

2.6.3 Navigation et notifications

La navigation de l'application est organisée de manière hiérarchique et adaptative selon le rôle de l'utilisateur. La barre de navigation principale affiche uniquement les sections accessibles à l'utilisateur connecté, évitant la confusion et améliorant la sécurité.

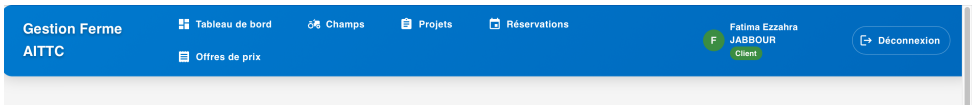

FIGURE 2.18 – Barre de navigation principale affichant les sections accessibles au client

FIGURE 2.19 – Message d'erreur affiché à l'utilisateur

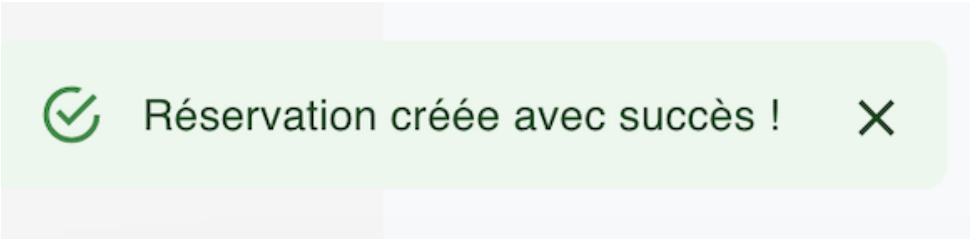

FIGURE 2.20 – Message de succès affiché à l'utilisateur

Le système de breadcrumbs (fil d'Ariane) aide les utilisateurs à se repérer dans la hiérarchie des pages. Les messages de confirmation et les notifications informent l'utilisateur du succès ou de l'échec de ses actions. La gestion des erreurs est proactive avec des messages explicites et des suggestions de résolution.

Cette conception centrée sur l'utilisateur assure une prise en main rapide de la plateforme tout en optimisant l'efficacité des tâches quotidiennes pour chaque profil d'utilisateur.

Conclusion

Ce chapitre a présenté la conception technique complète de la plateforme de gestion de la ferme expérimentale. L'architecture en couches, fondée sur le modèle SPA (Single Page Application) associé à une API REST, a été définie pour garantir la séparation des responsabilités, la maintenabilité et l'évolutivité du système. Les choix technologiques retenus (React et TypeScript pour le frontend, Node.js/Express pour le backend, PostgreSQL avec Prisma pour la persistance des données) offrent une base solide, moderne et évolutive pour le développement de la solution. Une attention particulière a été portée à la sécurité, avec la mise en place d'une architecture multi-couches intégrant l'authentification JWT, le contrôle d'accès par rôles (RBAC) et la protection contre les attaques courantes. La conception de l'interface utilisateur, centrée sur l'ergonomie et l'accessibilité, vise à optimiser l'expérience pour chaque profil d'utilisateur

(administrateur, superviseur, client), tout en respectant les standards web actuels. L'ensemble de cette conception technique détaillée constitue la fondation sur laquelle repose l'implémentation du système. Le chapitre suivant présentera la réalisation concrète de cette architecture, en détaillant l'organisation du code, l'implémentation des principales fonctionnalités, ainsi que les défis techniques rencontrés et les solutions apportées lors du développement.

Chapitre 3

Implémentation

Introduction

Ce chapitre présente l'implémentation concrète de la plateforme de gestion de la ferme expérimentale, basée sur la conception technique détaillée dans le chapitre précédent. L'objectif est de démontrer la réalisation effective des choix architecturaux et technologiques, en mettant l'accent sur l'organisation du code, l'implémentation des fonctionnalités principales et les défis techniques rencontrés lors du développement. L'implémentation s'est déroulée en parallèle pour le frontend et le backend, avec une approche itérative permettant de valider progressivement les fonctionnalités. La structure du projet, les standards de codage adoptés et l'organisation des composants constituent la base de cette implémentation. Les sections suivantes détaillent l'architecture du code, le développement des différentes couches de l'application et les solutions techniques mises en œuvre pour répondre aux exigences fonctionnelles et non fonctionnelles identifiées.

3.1 Structure et organisation du projet

3.1.1 Architecture des dossiers

L'organisation du code source du projet reflète concrètement les principes d'architecture modulaire et de séparation des responsabilités définis lors de la conception. Le projet adopte une structure monorepo, regroupant à la racine l'ensemble des éléments nécessaires au fonctionnement de la plateforme. On distingue principalement deux sous-dossiers : le dossier `client`, qui contient l'application frontend développée avec React et TypeScript, et le dossier `src` dédié au backend Node.js/Express.

La structure du backend s'articule autour de plusieurs dossiers spécialisés. Le dossier `routes` centralise la gestion des routes API, chaque fichier correspondant à une entité métier telle que

les utilisateurs, les projets, les réservations ou l'inventaire. Le dossier `middleware` regroupe les fonctions d'authentification, d'autorisation et de validation des données, tandis que le dossier `utils` contient les services utilitaires comme la génération de PDF et l'envoi d'emails. La configuration de la base de données et les scripts de migration sont quant à eux rassemblés dans le dossier `prisma`.

Le frontend, situé dans `client/src`, suit les conventions modernes de développement React. Les composants réutilisables sont organisés dans le dossier `components`, les pages principales dans `pages`, la gestion de l'état global dans `context`, et les services d'API dans `services`. Les types TypeScript sont centralisés dans le dossier `types` afin d'assurer la cohérence du typage à travers l'application.

Cette organisation claire et structurée facilite la maintenance, la collaboration et l'évolution du projet, en assurant une séparation nette entre les différentes couches et responsabilités.

3.1.2 Standards de codage et conventions

L'implémentation respecte des standards de codage stricts pour garantir la lisibilité et la maintenabilité du code. Le backend utilise JavaScript avec des conventions ES6+, incluant l'utilisation systématique de `const` et `let`, les fonctions fléchées, et la déstructuration des objets. Les noms de variables et de fonctions suivent la convention camelCase, tandis que les constantes utilisent UPPER_SNAKE_CASE.

Le frontend, développé en TypeScript, applique des règles de typage strictes avec des interfaces définies pour tous les objets métier. Les composants React suivent la convention PascalCase et sont organisés en composants fonctionnels avec hooks. Les props sont typées explicitement, et les états locaux utilisent le hook `useState` avec des types appropriés.

La gestion des erreurs est standardisée avec des try-catch blocks et des messages d'erreur explicites. Les logs de débogage utilisent des niveaux appropriés (info, warn, error) pour faciliter le diagnostic des problèmes en production.

3.1.3 Gestion des dépendances et configuration

La gestion des dépendances s'appuie sur npm avec des fichiers`package.json` séparés pour le backend et le frontend. Les versions des dépendances sont figées grâce aux fichiers `package-lock.json` pour garantir la reproductibilité des builds. Le backend et le frontend se lancent séparément, chacun avec ses propres commandes (npm start pour le backend, npm start dans le dossier client pour le frontend). La configuration environnementale utilise des variables d'environnement via le package `dotenv`. Les configurations sensibles (clés JWT, connexion à la base de données, paramètres SMTP) sont externalisées pour faciliter le déploiement dans différents environnements. L'architecture du projet permet l'intégration de scripts de validation automatique (vérification

de la syntaxe, validation du typage TypeScript, etc.), afin de garantir la qualité du code à chaque étape du développement.

3.2 Développement du backend

L'implémentation du backend constitue le cœur de la plateforme, centralisant la logique métier, la gestion des données et la sécurité. Le développement s'est articulé autour de la configuration du serveur Express, l'implémentation des routes API, et l'intégration des services utilitaires.

3.2.1 Configuration du serveur Express et middlewares

Le serveur Express a été configuré avec une approche modulaire pour faciliter la maintenance et l'évolutivité. L'initialisation du serveur dans `server.js` configure les middlewares essentiels dans un ordre spécifique pour garantir la sécurité et la performance.

Le middleware Helmet sécurise les en-têtes HTTP pour prévenir les attaques XSS et autres vulnérabilités web. CORS est configuré pour autoriser les requêtes cross-origin depuis le frontend, avec des options restrictives pour la production. Morgan assure le logging des requêtes HTTP pour le monitoring et le débogage. Le middleware de limitation de débit (express-rate-limit) protège contre les attaques par déni de service en limitant le nombre de requêtes par utilisateur.

La gestion des erreurs est centralisée avec un middleware d'erreur global qui capture les exceptions non gérées et renvoie des réponses d'erreur standardisées. Cette approche garantit une gestion cohérente des erreurs à travers toute l'application.

3.2.2 Implémentation des routes API principales

Les routes API sont organisées par domaine métier, chaque fichier dans le dossier `routes` correspondant à une entité principale. La route `/auth` gère l'authentification avec les endpoints de connexion, inscription et vérification de token. Les routes `/users`, `/fields`, `/projects`, `/reservations` et `/inventory` implémentent les opérations CRUD pour chaque entité.

Chaque route intègre la validation des données d'entrée via express-validator, garantissant l'intégrité et la sécurité des données. Les middlewares d'authentification et d'autorisation sont appliqués selon les besoins de chaque endpoint. La pagination et la recherche sont implémentées de manière cohérente pour optimiser les performances sur les grandes collections de données.

Les réponses API suivent un format standardisé avec des codes de statut HTTP appropriés, des messages d'erreur explicites et une structure de données cohérente. Cette standardisation facilite l'intégration côté frontend et améliore l'expérience développeur.

3.2.3 Gestion de l'authentification et des autorisations

L'authentification JWT est implémentée via un middleware personnalisé qui vérifie la validité et la signature des tokens. Le middleware `authenticateToken` extrait le token de l'en-tête Authorization, le valide avec la clé secrète, et enrichit l'objet request avec les informations de l'utilisateur.

Le contrôle d'accès basé sur les rôles (RBAC) est assuré par le middleware `requireRole` qui vérifie que l'utilisateur possède le rôle requis pour accéder à une ressource. Le middleware `requireOwnershipOrAdmin` permet aux utilisateurs d'accéder uniquement à leurs propres données, tout en autorisant les administrateurs à accéder à toutes les ressources.

La gestion des sessions inclut la génération de tokens avec une durée de vie limitée, la possibilité de révocation des tokens, et la gestion de la déconnexion. Les tokens sont signés avec une clé secrète robuste et incluent les informations essentielles de l'utilisateur (ID, email, rôle).

3.2.4 Intégration de Prisma et gestion de la base de données

L'ORM Prisma est configuré pour simplifier l'accès à la base de données PostgreSQL. Le schéma Prisma définit toutes les entités, relations et contraintes de la base de données. Les migrations automatiques permettent d'évoluer le schéma de manière contrôlée et reproductible.

Les opérations de base de données sont encapsulées dans des transactions pour garantir la cohérence des données, particulièrement pour les opérations complexes comme la gestion des surfaces de parcelles lors de la création de projets. Les requêtes sont optimisées avec des includes appropriés pour éviter le problème N+1 et améliorer les performances.

La gestion des erreurs de base de données inclut la capture des erreurs de contrainte, la validation des données avant insertion, et la gestion des cas d'erreur spécifiques (ex : surface insuffisante pour une réservation). Les logs de base de données facilitent le diagnostic des problèmes de performance et d'intégrité.

3.2.5 Services utilitaires

Les services de génération de PDF et d'envoi d'emails sont implémentés comme des modules indépendants dans le dossier `utils`. Le service PDF utilise PDFKit pour générer des documents administratifs (ordres de service, offres de prix) avec un formatage professionnel et des informations dynamiques.

Le service email, basé sur Nodemailer, gère l'envoi de notifications et d'alertes aux utilisateurs. Les templates d'email sont externalisés pour faciliter la maintenance et la personnalisation. La gestion des erreurs d'envoi inclut la retry automatique et le logging des échecs.

Ces services sont intégrés dans les routes API appropriées, permettant la génération automatique

de documents et l'envoi de notifications lors d'événements métier importants (validation de réservation, création de projet, etc.).

3.3 Défis techniques et solutions apportées

Le développement de la plateforme a présenté plusieurs défis techniques qui ont nécessité des solutions innovantes et des compromis réfléchis. Cette section présente les principaux obstacles rencontrés et les approches adoptées pour les résoudre.

3.3.1 Gestion de l'état global et synchronisation

Un défi majeur a été la gestion cohérente de l'état de l'application entre les différents composants React. L'utilisation du Context API pour l'authentification et la gestion des rôles a permis de centraliser l'état global tout en évitant la complexité d'une solution comme Redux pour ce projet.

La synchronisation entre les données du frontend et du backend a été gérée par des appels API optimisés et une gestion intelligente du cache. Les tokens JWT stockés dans le localStorage permettent de maintenir la session utilisateur même après rechargement de la page, améliorant l'expérience utilisateur.

3.3.2 Sécurité et validation des données

La sécurité de la plateforme s'appuie sur une authentification par JWT, avec gestion des rôles (RBAC) pour restreindre l'accès aux différentes fonctionnalités selon le profil utilisateur (administrateur, superviseur, client). Les routes sensibles du backend sont protégées par des middlewares d'authentification et de contrôle d'accès.

La validation des données côté backend est réalisée via des vérifications manuelles dans les routes et les contrôleurs, tandis que le frontend assure une première validation des champs obligatoires dans les formulaires (ex : champs requis, format d'email). Les mots de passe sont stockés de manière sécurisée grâce au hachage (bcrypt). Les variables d'environnement sont utilisées pour protéger les informations sensibles (clés JWT, accès base de données, SMTP)

3.3.3 Performance et optimisation

L'optimisation des performances a été abordée à plusieurs niveaux. Côté frontend, la pagination des tableaux de données (utilisateurs, projets, réservations, etc.) permet de limiter le volume d'informations affiché et d'améliorer la réactivité de l'interface. L'utilisation de composants fonctionnels et de hooks React (useState, useEffect) garantit une gestion efficace du cycle de vie des données. Côté backend, les requêtes vers la base de données sont optimisées grâce à Prisma ORM, et l'utilisation d'index sur les champs critiques améliore les temps de réponse. La gestion

des erreurs et la validation en temps réel contribuent également à une expérience utilisateur fluide tout en réduisant la charge serveur.

3.3.4 Gestion des erreurs et robustesse

La gestion des erreurs a été implémentée de manière cohérente à travers l'application. Le middleware d'erreur Express capture et traite les exceptions de manière appropriée, retournant des réponses d'erreur structurées au frontend.

Côté client, les intercepteurs Axios gèrent automatiquement les erreurs de réseau et les réponses d'erreur du serveur. Les messages d'erreur sont présentés de manière claire à l'utilisateur, facilitant la résolution des problèmes et améliorant l'expérience utilisateur globale.

Conclusion

Ce chapitre a présenté l'implémentation concrète de la plateforme de gestion de la ferme expérimentale. L'organisation modulaire du code, l'implémentation des fonctionnalités principales et la résolution des défis techniques ont permis de créer une solution robuste et évolutive. L'architecture en couches, avec une séparation claire entre frontend et backend, a facilité le développement et la maintenance. Les fonctionnalités implémentées couvrent l'ensemble des besoins identifiés, avec un accent particulier sur la sécurité, l'ergonomie et la gestion des rôles. Les défis techniques rencontrés, notamment la gestion de l'état global, la sécurité et l'optimisation des performances, ont été résolus par des solutions appropriées. La plateforme développée constitue une base solide pour la gestion numérique de la ferme expérimentale, offrant les fonctionnalités essentielles tout en maintenant la flexibilité nécessaire pour les évolutions futures.

Chapitre 4

Résultats et évaluation

Introduction

Ce chapitre présente les résultats obtenus lors du développement de la plateforme de gestion de la ferme expérimentale et évalue la solution développée par rapport aux objectifs fixés initialement. L'évaluation porte sur les aspects techniques, fonctionnels et qualitatifs de l'application.

La plateforme développée répond aux besoins de digitalisation identifiés lors de l'analyse préalable, offrant une solution complète pour la gestion des utilisateurs, des projets, des réservations et de l'inventaire. L'évaluation permet de mesurer l'adéquation de la solution aux attentes exprimées et d'identifier les points d'amélioration pour les évolutions futures.

L'analyse des résultats s'appuie sur les tests réalisés, les retours d'utilisation et l'évaluation des performances techniques. Cette évaluation globale permet de dresser un bilan objectif du projet et de proposer des perspectives d'amélioration.

4.1 Fonctionnalités réalisées

L'ensemble des fonctionnalités essentielles identifiées lors de l'analyse des besoins a été implémenté dans la plateforme. Le système d'authentification et de gestion des rôles permet un contrôle d'accès sécurisé et adapté à chaque profil utilisateur (administrateur, superviseur, client). Les modules de gestion des utilisateurs, des champs, des projets et des réservations sont opérationnels et offrent toutes les fonctionnalités prévues, telles que la création, la modification, la consultation et la suppression des données, ainsi que la validation des réservations selon le workflow défini. L'interface utilisateur, développée avec React et Material-UI, assure une navigation fluide et une expérience adaptée à chaque rôle. Le tableau de bord présente les principales statistiques et indicateurs d'activité. Le module d'inventaire a été conçu mais reste partiellement exploité en raison du manque de données structurées. L'ensemble de ces fonctionnalités a été

testé et validé, garantissant la conformité de la solution avec les besoins exprimés en phase de conception. Les éventuelles limitations ou axes d'amélioration sont détaillés dans les sections suivantes.

4.2 Évaluation technique

L'évaluation technique de la plateforme développée porte sur les aspects de performance, de sécurité, de robustesse et de qualité du code. Cette analyse s'appuie sur les tests réalisés et l'observation du comportement de l'application.

4.2.1 Performance et stabilité

L'application présente des performances satisfaisantes pour les opérations courantes. Les temps de réponse des requêtes API restent acceptables, avec des délais généralement inférieurs à 500ms pour les opérations de lecture et de moins de 1 seconde pour les opérations d'écriture. L'utilisation de Prisma ORM optimise les requêtes de base de données et réduit les temps de traitement.

La stabilité de l'application est bonne, avec une gestion appropriée des erreurs côté serveur. Le middleware d'erreur Express capture les exceptions et retourne des réponses d'erreur structurées, évitant les plantages de l'application. Les validations côté client et serveur contribuent à la robustesse du système.

4.2.2 Sécurité et robustesse

La sécurité de l'application a été implémentée selon les bonnes pratiques. L'authentification JWT assure la protection des sessions utilisateur, avec des tokens stockés de manière sécurisée dans le localStorage. Le hashage des mots de passe avec bcrypt garantit la protection des données sensibles.

La gestion des rôles (RBAC) fonctionne correctement, avec des vérifications de permissions à chaque niveau d'accès. Les routes protégées rejettent les requêtes non authentifiées et les tentatives d'accès non autorisées. La validation des données d'entrée avec Joi prévient les injections et les attaques par manipulation de données.

4.2.3 Qualité du code et maintenabilité

Le code de l'application suit des standards de qualité acceptables. L'architecture en couches facilite la maintenance et l'évolution du système. La séparation claire entre frontend et backend permet un développement et une maintenance indépendants.

L'utilisation de TypeScript côté frontend améliore la qualité du code en apportant un typage strict et en détectant les erreurs à la compilation. Les composants React sont bien structurés avec une séparation des responsabilités. L'utilisation de Material-UI assure une cohérence visuelle et réduit le temps de développement.

Côté backend, l'organisation modulaire des routes et des middlewares facilite la maintenance. L'utilisation de Prisma simplifie la gestion de la base de données et réduit les risques d'erreurs SQL. Les validations et la gestion d'erreurs sont cohérentes à travers l'application.

4.2.4 Points d'amélioration identifiés

Plusieurs points d'amélioration ont été identifiés pour optimiser les performances et la robustesse de l'application. L'implémentation de tests automatisés complets permettrait d'améliorer la fiabilité du système. L'ajout de logs détaillés faciliterait le débogage et le monitoring en production.

L'optimisation des requêtes de base de données, notamment pour les opérations de pagination et de filtrage, pourrait améliorer les performances sur de gros volumes de données. L'implémentation d'un système de cache réduirait les temps de réponse pour les données fréquemment consultées.

4.3 Évaluation fonctionnelle

La plateforme répond globalement aux besoins de digitalisation identifiés lors de l'analyse préalable. Le système d'authentification et de gestion des rôles fonctionne correctement, permettant à chaque type d'utilisateur d'accéder aux fonctionnalités qui lui sont destinées. Les modules de gestion des utilisateurs, des champs, des projets et des réservations couvrent l'essentiel des besoins fonctionnels, avec des opérations de création, modification, consultation et suppression disponibles selon les droits de chaque profil. Le workflow de validation des réservations est opérationnel et adapté aux différents rôles.

4.3.1 Expérience utilisateur

L'interface utilisateur, développée avec React et Material-UI, offre une expérience fluide et cohérente. La navigation est intuitive, le rôle de l'utilisateur est clairement affiché, et les fonctionnalités accessibles sont adaptées à chaque profil. Les formulaires intègrent des validations en temps réel et des messages d'erreur explicites, facilitant la saisie et la correction des données. L'application est responsive et s'adapte aux différents supports. Le tableau de bord présente les informations essentielles de manière synthétique et accessible.

4.3.2 Gestion des cas d'erreur

La gestion des erreurs est assurée à la fois côté serveur et côté client. Les erreurs serveur sont capturées et retournées avec des messages structurés, tandis que le frontend affiche des notifications claires à l'utilisateur en cas d'échec d'une opération ou de problème réseau. Les états de chargement sont gérés pour informer l'utilisateur lors des opérations longues.

4.3.3 Fonctionnalités manquantes ou limitées

Certaines fonctionnalités prévues initialement n'ont pas pu être finalisées dans l'interface utilisateur. La génération de PDF pour les ordres de service et les offres de prix, bien que disponible côté backend, n'est pas accessible via l'interface. Le module d'inventaire reste partiellement exploité en raison du manque de données structurées. De plus, les processus métier réels de la ferme expérimentale s'avèrent plus complexes que les modèles fournis, ce qui limite l'automatisation complète de certains workflows. L'export de données, le reporting avancé et la gestion des notifications par email pourraient être enrichis dans de futures évolutions.

4.3.4 Satisfaction des objectifs

La plateforme atteint une grande partie des objectifs fixés, en digitalisant les processus de gestion des utilisateurs, des champs, des projets et des réservations, et en améliorant l'accessibilité et la traçabilité des informations. L'automatisation des workflows de validation contribue à l'efficacité opérationnelle. Toutefois, certains objectifs restent partiellement atteints, notamment la génération automatisée de documents et les fonctionnalités avancées de reporting, qui nécessiteraient un développement complémentaire et une analyse approfondie des besoins métier.

4.4 Limites et difficultés rencontrées

Le développement de la plateforme a été confrontée à plusieurs contraintes et difficultés qui ont influencé l'étendue et la qualité de la solution finale. Cette section présente les principales limitations rencontrées et les défis techniques auxquels le projet a été confronté.

4.4.1 Contraintes techniques

Le développement de la plateforme a été marqué par une évolution technologique significative. Initialement basé sur Python pour sa simplicité et la richesse de ses bibliothèques, le projet a rapidement rencontré des obstacles, notamment des difficultés dans la gestion des dépendances et la complexité de configuration des environnements. Face à ces défis, une transition stratégique vers l'écosystème Node.js avec Express a été opérée. Ce choix s'est avéré plus adapté, offrant une intégration native avec les technologies frontend comme React, une gestion des dépendances

simplifiée grâce à npm, et une mise en œuvre plus fluide des fonctionnalités clés comme l'authentification JWT et les API REST.

Par ailleurs, la génération du PDF de l'offre de prix n'a pas pu être totalement finalisée, certaines informations et validations nécessaires n'ayant pas encore pu être consolidées ; des améliorations restent donc à prévoir pour aboutir à une version pleinement opérationnelle.

Enfin, le projet a été fortement contraint par le cadre temporel du PFE, ce qui a nécessité de prioriser les fonctionnalités essentielles au détriment d'aspects importants comme l'optimisation des performances ou la mise en place d'une suite de tests automatisés complète.

4.4.2 Limitations fonctionnelles

L'implémentation complète de certaines fonctionnalités s'est heurtée à la complexité des processus métier réels. Les fiches fournies (F.10, F.47, F.75, F.84) constituent des modèles de base, mais les processus opérationnels de la ferme expérimentale impliquent des workflows plus complexes et des données supplémentaires qui n'étaient pas disponibles dans le cadre de ce projet.

La génération de documents PDF (ordres de service, offres de prix) a été partiellement implémentée côté backend mais n'a pas pu être intégrée dans l'interface utilisateur faute de temps et de spécifications détaillées sur les formats exacts attendus. L'export de données et les fonctionnalités de reporting avancées nécessiteraient une analyse plus approfondie des besoins réels.

4.4.3 Défis de développement

La gestion de l'état global de l'application côté frontend a présenté des défis techniques, notamment pour la synchronisation des données entre les différents composants. L'utilisation du Context API a permis de résoudre ces problèmes, mais une solution plus robuste comme Redux pourrait être envisagée pour des applications plus complexes.

La sécurité de l'application a nécessité une attention particulière, notamment pour la gestion des tokens JWT et la protection des routes sensibles. L'implémentation du système RBAC a été complexe mais nécessaire pour garantir un accès contrôlé aux fonctionnalités selon les rôles des utilisateurs.

4.4.4 Contraintes de ressources

Le développement s'est déroulé avec des ressources limitées, notamment en termes de temps et d'accès aux utilisateurs finaux. L'absence de tests utilisateurs réguliers a limité la possibilité d'optimiser l'expérience utilisateur selon les besoins réels.

L'implémentation de fonctionnalités avancées comme la gestion des notifications par email, l'export de données et les rapports personnalisés aurait nécessité plus de temps et de ressources.

Ces fonctionnalités pourraient être développées dans une phase ultérieure du projet.

4.5 Conclusion de l'évaluation

L'évaluation de la plateforme de gestion de la ferme expérimentale révèle une solution fonctionnelle qui répond partiellement aux objectifs initiaux du projet. Cette conclusion synthétise les résultats obtenus et mesure la satisfaction des objectifs fixés.

4.5.1 Synthèse des résultats

La plateforme développée constitue une base solide pour la digitalisation de la gestion de la ferme expérimentale. Les principales fonctionnalités implémentées (authentification, gestion des utilisateurs, des champs, des projets, des réservations, de l'inventaire et des offres de prix) sont opérationnelles et s'appuient sur une interface utilisateur moderne, claire et intuitive. L'architecture technique, reposant sur un backend Node.js/Express et une interface frontend en React/TypeScript, s'est révélée adaptée aux besoins du projet. Cette stack a permis une séparation claire des responsabilités, une meilleure maintenabilité du code et une intégration fluide avec la base de données PostgreSQL via Prisma. La sécurité de l'application est assurée par une authentification basée sur JWT et une gestion fine des droits d'accès selon les rôles (RBAC). Les validations côté client et serveur contribuent à l'intégrité des données et à la fiabilité des opérations. L'expérience utilisateur a été soignée, avec une navigation fluide et des retours d'erreur explicites. Certaines fonctionnalités avancées, telles que la génération complète de PDF pour les offres de prix ou l'automatisation de rapports, restent à finaliser pour atteindre une version pleinement aboutie.

4.5.2 Satisfaction des objectifs

Les objectifs de digitalisation et d'automatisation ont été en grande partie atteints pour les fonctionnalités principales. La plateforme permet de centraliser et de simplifier la gestion des utilisateurs, des réservations, de l'inventaire et des projets, réduisant ainsi la dépendance aux processus papier. L'accès aux informations est facilité grâce à une interface web unique, accessible depuis n'importe quel navigateur. Néanmoins, certains objectifs initiaux restent à finaliser, notamment la génération automatisée et complète de documents (ordres de service, offres de prix) ainsi que la mise en place de fonctionnalités de reporting avancées. Ces aspects nécessitent un développement complémentaire, en raison de la complexité des processus métier et des contraintes de temps rencontrées lors du projet.

4.5.3 Points forts et acquis

Les points forts de la solution développée incluent une architecture modulaire et évolutive, une interface utilisateur moderne et responsive, et une gestion robuste de la sécurité. L'utilisation de technologies modernes (React, TypeScript, Prisma) assure la maintenabilité et l'évolutivité du code.

L'expérience acquise lors du développement, notamment la gestion des défis techniques et l'adaptation des choix technologiques, constitue un acquis précieux pour de futurs projets. La compréhension des contraintes réelles du développement d'applications web professionnelles enrichit significativement le bagage technique.

4.5.4 Perspectives d'amélioration

La plateforme développée constitue une base solide pour des évolutions futures. Les fonctionnalités manquantes pourraient être implémentées dans une phase ultérieure, en s'appuyant sur l'architecture existante et en bénéficiant d'une collaboration plus étroite avec les utilisateurs finaux.

L'amélioration des tests automatisés, l'optimisation des performances et l'ajout de fonctionnalités avancées comme l'export de données et les rapports personnalisés permettraient d'aboutir à une solution complète répondant à l'ensemble des besoins identifiés.

4.6 Bilan du projet

4.6.1 Objectifs atteints

Le projet a permis de développer une plateforme fonctionnelle pour la gestion de la ferme expérimentale, répondant aux besoins essentiels de digitalisation identifiés. L'authentification sécurisée, la gestion des rôles et les fonctionnalités CRUD pour les entités principales ont été implémentées avec succès.

L'architecture technique adoptée s'est avérée robuste et maintenable, offrant une base solide pour les évolutions futures. L'interface utilisateur moderne et responsive améliore significativement l'expérience utilisateur par rapport aux processus papier existants.

4.6.2 Compétences développées

Ce projet a permis de développer des compétences techniques approfondies dans le développement d'applications web modernes. L'utilisation de technologies comme React, TypeScript, Node.js et Prisma a enrichi le bagage technique et permis de comprendre les défis du développement d'applications professionnelles.

La gestion des contraintes de temps, l'adaptation des choix technologiques et la résolution de problèmes techniques ont développé des compétences de gestion de projet et de prise de décision technique.

4.6.3 Limites et apprentissages

Les limites rencontrées, notamment la complexité des processus métier réels et les contraintes de temps, ont permis de comprendre l'importance d'une analyse approfondie des besoins et d'une collaboration étroite avec les utilisateurs finaux. L'expérience acquise sera précieuse pour de futurs projets professionnels.

Conclusion générale

Le projet de développement de la plateforme de gestion de la ferme expérimentale s'inscrit dans une démarche de modernisation et d'optimisation des processus agricoles. Dès l'analyse des besoins, l'accent a été mis sur la centralisation des données, la sécurisation des accès et la simplification des tâches quotidiennes pour les différents utilisateurs. La solution mise en place, reposant sur une architecture moderne (React, Node.js, PostgreSQL), a permis de répondre efficacement à ces enjeux.

Tout au long du projet, plusieurs modules essentiels ont été conçus et intégrés : gestion des utilisateurs et des rôles, administration des parcelles et des projets, gestion des réservations, suivi de l'inventaire et traitement des offres de prix. Chaque fonctionnalité a été pensée pour offrir une expérience utilisateur fluide, tout en garantissant la sécurité et la fiabilité des données. L'implémentation d'un système d'authentification robuste et d'une gestion fine des droits d'accès a constitué un atout majeur pour la protection des informations sensibles.

Ce travail a également été l'occasion de mettre en pratique des compétences variées, allant de la modélisation de bases de données à la conception d'interfaces ergonomiques, en passant par la gestion de projet et la collaboration avec les parties prenantes. Les difficultés rencontrées, notamment en matière de coordination et de collecte d'informations, ont été surmontées grâce à une adaptation continue et à une communication régulière avec les utilisateurs finaux.

Au-delà des aspects techniques, ce projet a permis de mesurer l'importance de l'écoute des besoins du terrain et de la capacité à proposer des solutions évolutives. Les perspectives d'amélioration identifiées, telles que l'optimisation des performances, l'ajout de fonctionnalités de reporting, de monitoring ou de notifications, témoignent du potentiel d'évolution de la plateforme.

En conclusion, cette expérience a été particulièrement enrichissante sur le plan professionnel et personnel. Elle a permis de concrétiser un projet à fort impact, tout en développant des compétences essentielles pour la conduite de projets informatiques complexes. La plateforme constitue aujourd'hui une base solide pour accompagner la digitalisation des activités de la ferme expérimentale et pourra servir de fondation à de futurs développements.

Bibliographie

[1] React Documentation, Meta Platforms, Inc., 2024.
https://react.dev/

[2] Node.js Documentation, Node.js Foundation, 2024.
https://nodejs.org/docs/

[3] TypeScript Handbook, Microsoft Corporation, 2024.
https://www.typescriptlang.org/docs/

[4] Prisma Documentation, Prisma Data Platform, 2024.
https://www.prisma.io/docs/

[5] Material-UI Documentation, MUI, 2024.
https://mui.com/material-ui/

[6] Express.js Documentation, Express.js Team, 2024.
https://expressjs.com/

[7] JSON Web Token (JWT), RFC 7519, IETF, 2015.
https://tools.ietf.org/html/rfc7519

[8] Web Content Accessibility Guidelines (WCAG) 2.1, W3C, 2018.
https://www.w3.org/WAI/WCAG21/quickref/

[9] MDN Web Docs, Mozilla Developer Network, 2024.
https://developer.mozilla.org/

[10] Stack Overflow, Stack Exchange, 2024.
https://stackoverflow.com/

[11] GitHub Documentation, GitHub, Inc., 2024.
https://docs.github.com/

[12] npm Documentation, npm, Inc., 2024.
https://docs.npmjs.com/

[13] Jest Documentation, Meta Platforms, Inc., 2024.
https://jestjs.io/docs/getting-started

[14] bcrypt Documentation, npm, 2024.
https://www.npmjs.com/package/bcryptjs

[15] Helmet.js Documentation, 2024.
https://helmetjs.github.io/

[16] CORS Documentation, MDN Web Docs, 2024.
https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

[17] PostgreSQL Documentation, PostgreSQL Global Development Group, 2024.
https://www.postgresql.org/docs/

[18] Axios Documentation, 2024.
https://axios-http.com/docs/intro

[19] React Router Documentation, 2024.
https://reactrouter.com/docs/en/v6

[20] React Hook Form Documentation, 2024.
https://react-hook-form.com/docs

[21] Yup Validation Documentation, 2024.
https://github.com/jquense/yup

[22] Joi Validation Documentation, 2024.
https://joi.dev/api/

[23] Morgan Logger Documentation, 2024.
https://github.com/expressjs/morgan

[24] Multer File Upload Documentation, 2024.
https://github.com/expressjs/multer

[25] Nodemailer Documentation, 2024.
https://nodemailer.com/

[26] PDFKit Documentation, 2024.
https://pdfkit.org/docs/getting_started.html

[27] Express Rate Limit Documentation, 2024.
https://github.com/express-rate-limit/express-rate-limit

[28] UUID Documentation, 2024.
https://github.com/uuidjs/uuid

[29] Web Vitals Documentation, Google, 2024.
https://web.dev/vitals/

[30] ESLint Documentation, 2024.
https://eslint.org/docs/latest/

[31] "Modern Web Development Practices", *Web Development Journal*, 2023.

[32] "Database Design Principles", *Database Systems Journal*, 2022.

[33] "Security in Web Applications", *Cybersecurity Review*, 2023.

[34] "REST API Design Best Practices", *API Development Journal*, 2023.

[35] "Frontend Architecture Patterns", *Software Engineering Review*, 2023.

[36] "Testing Strategies for Web Applications", *Quality Assurance Journal*, 2023.

[37] "Agile Development Methodologies", *Project Management Review*, 2023.

[38] "User Experience Design in Web Applications", *UX Design Journal*, 2023.

[39] "Performance Optimization Techniques", *Web Performance Review*, 2023.

[40] "Code Quality and Best Practices", *Software Quality Journal*, 2023.

Annexe A

Annexes

A.1 Annexe A : Documents de référence

A.1.1 Fiche F.10 - Inventaire hebdomadaire

MOHAMMED VI POLYTECHNIC UNIVERSITY / Agricultural Innovation and Technology Transfer Center	Centre : AITTC / UM6P Division : Ferme expérimentale site Ben Guérir Service : Gestion administrative et support /Gestion de stock et patrimoine **Fiche : Inventaire hebdomadaire**	Code : F.10 Version : 02 Date : 29/04/2021 page 1 sur 1

Propriétaire d'article	famille d'article	Sous famille	Désignation	Stock	Unité	Etat d'article	Emplacement article

F.EXP - V.02 - 04/21 - F.10

A.1.2 Fiche F.47 - Fiche offre de prix

MOHAMMED VI POLYTECHNIC UNIVERSITY / Agricultural Innovation and Technology Transfer Center	Centre : AITTC / UM6P Division : Ferme expérimentale site Ben Guérir Service : Gestion administrative et support /Gestion commerciale, partenariats et communication **Fiche : offre de prix**	Code : F.47 Version : 01 Date : 09/06/2021 Page : 1/1

Date :

Demande N° :

Destinataire :

Centre AITTC de l'Université Mohamed 6 Polytechniques (UM6P)

Demandeur :

Contacts :

Adresse :

Numéro d'ordre	Désignation article	Description technique	Unité	Quantité

Centre AITTC de l'Université Mohamed 6 Polytechniques (UM6P)
Adresse : Lot 660,Hay Moulay Rachid 43150,
Ben Guérir -www.um6p.ma
R.C.n° 1037-Patente n° 45408944-I.F. 14437938-CNSS n° 9515919-ICE n° 000189568000063
Morrocco

FIGURE A.2 – Fiche F.47 - Fiche offre de prix (document fourni)

A.1.3 Fiche F.75 - Fiche management du comité de pilotage

MOHAMMED VI UNIVERSITY	**Centre : AITTC / UM6P** **Division : Ferme expérimentale site Ben Guérir** **Service : Management de la ferme expérimentale de l'AITTC** **Fiche de répartition des responsabilités et suivi des essais et projets à la ferme expérimentale**	**Code : F.75** **Version : 01** **Date : 22/9/2021** **page 1 sur 1**

Membre du Comité de pilotage	Younes JNAOUI	Responsable ferme expérimentale
	Yasmine TAOUIR	Technicienne agricole
	Lahoucine AIT YAHIA	Technicien agricole
	Jelloul FNIZI	Technicien agricole
	Yassine ELBEQQAL	Technicien agricole
	Aziz JMEL	Chargé de la gestion administrative

Répartition des responsabilités du comité de pilotage		
Activités	**Superviseur**	**Remplaçant**
Supervision de la fertilisation, de l'irrigation et de la fertigation	Jelloul FNIZI	Yasmine TAOUIR
Supervision de la protection phytosanitaire	Yasmine TAOUIR	Jelloul FNIZI
Installation et suivi des projets de recherche et de production à la ferme	Techniciens agricoles	
Assistant gestion administrative et contrôle de gestion interne	Aziz JMEL	
Assistant gestion administrative et Commerciale	Brahim EL KAJAF	
Gestion de la ferme	Younes JNAOUI	

Axe	Service	Cellule	Coordinateur	Remplaçant
Gestion administrative et support ferme	**Gestion administrative et support**	Gestion des ressources humaines ferme	Aziz JMEL	Yasmine TAOUIR
		Gestion des achats et approvisionnements	Aziz JMEL	Brahim EL KAJAF
		Gestion de stock et patrimoine	Brahim EL KAJAF	Aziz JMEL
		Gestion QHSSE	Brahim EL KAJAF	Yassine ELBEQQAL
		Gestion atelier et parc matériel ferme	Bouchaib ESSAIH	Jelloul FNIZI
		Gestion commerciale, partenariats et communication	Brahim EL KAJAF	Younes JNAOUI
		Gestion finance et comptabilité	A développer	
Gestion technique et production à la ferme	**Gestion technique et production**	Recherche scientifique	Technicien (superviseur du projet)	Technicien (superviseur)
		Partie commune (recherche scientifique et production agricole)	Technicien (superviseur du projet)	Technicien (superviseur)
Gestion et management globale de la ferme	**Gestion et management de la ferme**	Gestion et management de la ferme	Comité de pilotage	

Date dernière mise à jour : 26/03/2022

Cellule gestion technique	Intitulé du projet ou essai	Superficier	Unité	Emplacement	Du coté ferme assuré par		Du coté client assuré par	Département	Date prévue du début et de la fin du projet ou essai	Situation projet
					Superviseur	Remplaçant	Responsable projet			
Recherche Scientifique	OCP Innovation	30,00	m^2	La pépinière	Yasmine TAOUIR	Yassine ELBEQQAL	Fatima Ezzahra JABBOUR	AITTC	09-2021 au 04-2021	Finalisé
	OCP Strategy (ferme)	30,00	m^2	La pépinière	Yasmine TAOUIR	Yassine ELBEQQAL	Adnan BENAICH	AITTC	02-2021 au 12-2021	Finalisé
	OCP Strategy (Station expérimentale Sidi El Aydi)	3 650,00	m^2	Sidi El Aydi	Yasmine TAOUIR	Yassine ELBEQQAL	Adnan BENAICH	AITTC	02-2021 au 12-2021	Finalisé
	OCP Strategy (Station expérimentale Tassaout)	3 611,00	m^2	Tassaout	Yasmine TAOUIR	Yassine ELBEQQAL	Adnan BENAICH	AITTC	02-2021 au 12-2021	Finalisé
	Multiplication des semences Vigna et Lentille	30,00	m^2	La pépinière	Yasmine TAOUIR	Yassine ELBEQQAL	Adil EL BAOUCHI	AGBS	05-2021 au 08-2021	Finalisé
	Valorisation agronomique de la boue	60,00	m^2	La pépinière	Yasmine TAOUIR	Yassine ELBEQQAL	Fatima Ezzahra JABBOUR	AITTC	11-2021 au 02-2022	Finalisé
	Le potentiel biostimulant des extraits des plantes-1	20,00	m^2	La pépinière	Yasmine TAOUIR	Yassine ELBEQQAL	Hassan ANNAZ	AGBS	11-2021 au 01-2022	Finalisé
	Collection Cactus AgBS	100,00	m^2	La pépinière	Yasmine TAOUIR	Yassine ELBEQQAL	Manal MHADA	AGBS	Depuis 08-2021	En cours
	Réhabilitation des sols minier par les PAM	3,00	m^2	La pépinière	Yasmine TAOUIR	Yassine ELBEQQAL	Khadija AIT ELALLAM	AGBS	11-2021 au 03-2022	En cours
	Test d'inoculation de la fève sous stress	44,00	m^2	La pépinière	Yasmine TAOUIR	Yassine ELBEQQAL	Bouchra BENMRID	AGBS	11-2021 au 04-2022	En cours
	Le potentiel biostimulant des extraits des plantes-2	20,00	m^2	La pépinière	Yasmine TAOUIR	Yassine ELBEQQAL	Hassan ANNAZ	AGBS	04-04-22 au 15-06-22	En cours
	Collecte et tri des déchets ménagers	-	-	La pépinière	Yasmine TAOUIR	Yassine ELBEQQAL	Khadija NASRAOUI Hamid MOUTAOUAKIL	AITTC		En cours
	Agroforesterie	29 160,00	m^2	La pépinière	Yasmine TAOUIR	Yassine ELBEQQAL	Massamba DIOP	AITTC	12-2020 au 12-2024	En cours
	Collection cactus AITTC	130,00	m^2	La pépinière	Yasmine TAOUIR	Yassine ELBEQQAL	Aziza TANGI	AITTC	Depuis 11-2018	En cours
	Développement de formulation de fertilisant(AS17)	50,00	m^2	La pépinière	Yasmine TAOUIR	Yassine ELBEQQAL	Rachid GHANI	AGBS	13-12-2021 au 13-05-2022	En cours
	Essai de triticale	300,00	m^2	La pépinière	Yasmine TAOUIR	Yassine ELBEQQAL	Dr EL HAFIDI	AGBS	depuis 20-12-2021	En cours
	Essai ComSilageCorn (Etablir des connaissances scientifiques sur le comportement du compost dans le sol de Benguerir)	25,00	m^2	La pépinière	Yasmine TAOUIR	Yassine ELBEQQAL	Rachida Naciri, Mustapha El Janati	AGBS	25-03-22 au 23-06-2022	En cours
	Effets d'engrais phosphatés sur les cultures légumineuses(SoilPhorUfie)	20,00	m^2	La pépinière	Yasmine TAOUIR	Yassine ELBEQQAL	Chtouki Mohamed	AGBS	04-22 au 06-22	En cours
	Effets d'engrais phosphatés enrichis en micronutriments sur la croissance du maïs. (ESAFE)	50,00	m^2	La pépinière	Yasmine TAOUIR	Yassine ELBEQQAL	Rachida Nacri Safi Uwase	AGBS	28-03-22 au 30-6-22	En cours
	Fertialgues 1(Essai préliminaire pour la détermination de la dose d'application des extraits d'algues)	2,00	m^2	La pépinière	Yassine ELBEQQAL	Yasmine TAOUIR	Mohamed El mahdi EL BOUKHARI	AGBS	09-2021 au 10-2021	Finalisé
	Fertialgues 2(Effet des extraits d'algues sur l'atténuation du stress hydrique chez la tomate)	30,00	m^2	La pépinière	Yassine ELBEQQAL	Yasmine TAOUIR	Mohamed El mahdi EL BOUKHARI	AGBS	10-2021 au 11-2022	Finalisé
	Croissance des tomates sous stress	35,00	m^2	La pépinière	Yassine ELBEQQAL	Yasmine TAOUIR	Mohamed El mahdi EL BOUKHARI	AGBS	10-2021 au 12-2021	Finalisé
	Fertialgues3 (Effet des extraits d'algues et des acides humiques sur la croissance et le développement des plantes)	30,00	m^2	La pépinière	Yassine ELBEQQAL	Yasmine TAOUIR	El Boukhari et Bouhia Younes	AGBS	28-12-2021 au 28-02-2022	Finalisé
	Evaluer les dates de semis du quinoa	1 699,00	m^2	La pépinière	Yassine ELBEQQAL	Yasmine TAOUIR	Nawal TAAIME	AITTC	11-2021 au 09-2022	En cours
	Evaluer l'effet de la fertilisation NPK	5 809,00	m^2	La pépinière	Yassine ELBEQQAL	Yasmine TAOUIR	Nawal TAAIME	AITTC	12-2021 au 09-2022	En cours
	Effet résiduel de l'amendement organique sur quinoa	3 584,00	m^2	La pépinière	Yassine ELBEQQAL	Yasmine TAOUIR	Nawal TAAIME	AITTC	12-2021 au 09-2022	En cours
	Evaluation de la collection du quinoa	2 268,00	m^2	La pépinière	Yassine ELBEQQAL	Yasmine TAOUIR	Manal MHADA	BPS-AGBS	10-2021 au 07-2022	En cours
	Carbon Farming (ferme)	20 000,00	m^2	La pépinière	Yassine ELBEQQAL	Jelloul FNIZI	Mohamed Louay MAATOUGUI	AITTC	10-2018 au 04-2022	En cours
	Effet d'applications de plusieurs produits organiques sur l'amélioration de la croissance et du développement des plantes	10,00	m^2	La pépinière	Yassine ELBEQQAL	Jelloul FNIZI	Younes BOUHIYA	AGBS	11-2021 au 03-2022	En cours
	Plateforme collection acacia	4 607,50	m^2	La pépinière	Yassine ELBEQQAL	Yasmine TAOUIR	Mohamed Louay MAATOUGUI	AITTC	Programme	En cours
	Essai de comparaison de fertilisants azotés	14,00	m^2	La pépinière	Yassine ELBEQQAL	Yasmine TAOUIR	Manal MHADA	AGBS	11-2021 au 04-2022	En cours
	Test efficacité du produit Biostimulant(Projet Mycorhize)	*******	m^2	UM6p	Yassine ELBEQQAL	Yasmine TAOUIR	Imad KHATOUR	Africane genome-center(AGC)		En cours au laboratoire de l'UM6P
	Essai de comparaison de compostes	14,00	m^2	La pépinière	Yassine ELBEQQAL	Yasmine TAOUIR	Khadija El HAZZAM Manal Mhada	BPS-AGBS	15/02/2022 au 06/2022	En cours
	Réhabilitation	8 500,00	m^2	La mine	Yassine ELBEQQAL	Jelloul FNIZI	Haitam MOULAY Dontien	AITTC	02-2020 au 02-2024	En cours
	Evaluation du phosphogypse en tant que fertilisant low-cost pour les cultures en plein champ-**Phosphogypse**	3 500,00	m^2	La pépinière	Yassine ELBEQQAL	Yasmine TAOUIR	Mbarka OUTBAKAT	AITTC	09-2020 au 12-2022	En cours
	Evaluation de l'effet de la dose et de la fréquence d'amendement en phosphogypse sur les cultures et la qualité des sols affectés par la salinité -**Phosphogypse**	150,00	m^2	La pépinière	Yassine ELBEQQAL	Yasmine TAOUIR	Mbarka OUTBAKAT	AITTC	09-2020 au 12-2022	En cours
	Effet de la qualité du phosphogypse et des eaux d'irrigation sur les cultures et la qualité des différents sols affectés par la salinité-**Phosphogypse**	150,00	m^2	La pépinière	Yassine ELBEQQAL	Yasmine TAOUIR	Mbarka OUTBAKAT	AITTC	09-2020 au 12-2023	En cours
	Test de croissance des plantes-test amendement organique	50,00	m^2	La pépinière	Yassine ELBEQQAL	Yasmine TAOUIR	Saida TAYIBI Chaimaa HAKIM	AGBS	08/02/2022 au 08/06/2022	En cours
	Installation d'une plateforme des cultures biosaline(3 espaces du Gazon et cultures foragers)	20,00	m^2	La pépinière	Lahoucine AIT YAHIA	Yasmine TAOUIR	Dr CHOKRALLAH	AITTC	Depuis 05-2021	En cours
	Oliveraie Agro-écologique	14 600,00	m^2	La pépinière	Jelloul FNIZI	Yasmine TAOUIR	Mohamed BOULIF	ESAFE/AGBS	06/01/2021 à l'infini	En cours
	Carbon Farming (mine)	20 000,00	m^2	La mine	Jelloul FNIZI	Yassine ELBEQQAL	Mohamed Louay MAATOUGUI	AITTC	10-2018 au 04-2022	En cours
	Projet introduction de la culture du goji à la région de BHAMNA	5 000,00	m^2	La pépinière	Jelloul FNIZI	Yassine ELBEQQAL	Khadija NASRAOUI	AITTC	mois 10-2021	En cours
	Résistance des bacs	2,00	m^2	La pépinière	Jelloul FNIZI	Yasmine TAOUIR	Adil EL BAOUCHI	AGBS	02-2021 au 03-2021	En cours
Total		**127 407,50**	**m^2**							

Cellule gestion technique	Type d'activité	Catégorie des activités		Emplacement	Nature et intitulé du projet	Superficier	Unité	Superviseur	Remplaçant	Département	Durée du projet	situation projet
Production agricole	Production animale	Projet avicole		A la ferme	Poulet fermier			Yamine TAOUIR	Yassine ELBEQQAL	AITTC	Programme ferme	En cours
	Production végétale	Reproduction et multiplication des espèces		A la pépinière	Multiplication des plantes arboricoles, ornementales, maraichères et P.A.M	4 000,00	m^2	Lahoucine AIT YAHIA	Yasmine TAOUIR	AITTC	Programme ferme	En cours
		Production des semences		A la pépinière	Production des semences ancestrales du cumin	26 700,00	m^2	Yasmine TAOUIR	Lahoucine AIT YAHIA	AITTC	Programme	En cours
					Production des semences des céréales INRA	1 000,00	m^2	Yasmine TAOUIR	Lahoucine AIT YAHIA	AITTC	Programme	En cours
					Production des semences quinoa	*****	m^2	Yasmine TAOUIR	Lahoucine AIT YAHIA	AITTC	Programme	En cours
		Production des cultures	Arboriculture	A la pépinière	Plateforme arganier	2 600,00	m^2	Jelloul FNIZI	Lahoucine AIT YAHIA	AITTC	Programme	En cours
					Plateforme cactus	1 0 000	m^2	Yasmine TAOUIR	Yassine ELBEQQAL	AITTC	Programme	En cours
				A la ferme	Verger Olivier (jeune plantation et olviers âgés)		m^2	Jelloul FNIZI	Yasmine TAOUIR	AITTC	Programme	En cours
			Maraichage	A la pépinière	Projet panier (Tomate, carotte, courgette ...)	******	m^2	Jelloul FNIZI	Yassine ELBEQQAL	AITTC	Programme	A lancer
			Céréaliculture	A la ferme	Semis d'orge	150 000,00	m^2	Jelloul FNIZI	Lahoucine AIT YAHIA	AITTC	Programme	A lancer
			Cultures fourragères	A la pépinière	Production atriplex	100,00	m^2	Lahoucine AIT YAHIA	Yasmine TAOUIR	AITTC	programme	En cours
					Production panicum	50,00	m^2	Lahoucine AIT YAHIA	Yasmine TAOUIR	AITTC	programme	En cours
			P.A.M	A la pépinière	Production safran, menthe, verveine ...	794,00	m^2	Yasmine TAOUIR	Jelloul FNIZI	AITTC	Programme	En cours
	Projets et activités de développement	Au sein de la ferme	Ornementale et espaces verts	Ferme AITTC	Installation clôture végétales de la ferme		m^2	Yassine ELBEQQAL	Jelloul FNIZI	AITTC	Programme	A lancer
					Entretien des espaces verts de l'AITTC	10 000,00	m^2	Yassine ELBEQQAL	Yasmine TAOUIR	AITTC	Programme	En cours
			Biostimulants et fertilisants		Smart Farm-OCP Agribiotech	40 000,00	m^2	Yasmine TAOUIR	Jelloul FNIZI	Smart Pharm	Programme	En cours
			Agriculture de précision		AgriEdge 1	1 727,00	m^2	Yasmine TAOUIR	Yassine ELBEQQAL	Agri Edge	Programme	En cours
					AgriEdge 2	50 000,00	m^2	Yasmine TAOUIR	Yassine ELBEQQAL	Agri Edge	Programme	En cours
			Learning by farming		Installation parcelles pour étudiants-**ESSAFE**	2 200,00	m^2	Yassine ELBEQQAL	Jelloul FNIZI	ESSAFE	Programme	En cours
					Installation de la parcelle master exécutive (Agriculture de précision)-**ESSAFE**	10 000,00	m^2	Yassine ELBEQQAL	Jelloul FNIZI	ESSAFE	Programme	En cours

FIGURE A.3 – Fiche F.75 - Fiche management du comité de pilotage (document fourni)

A.1.4 Fiche F.84 - Fiche ordre de service

MOHAMMED VI POLYTECHNIC UNIVERSITY / Agricultural Innovation and Technology Transfer Center	Centre : AITTC / UM6P	Code : F.84
	Division : Ferme expérimentale site Ben Guérir	Version : 01
	Service : Management de la ferme expérimentale de l'AITTC	Date :01/12/2021
	Fiche : ordre de service	page 1 sur 1

ORDRE DE SERVICE

Objet : Commencement des travaux du marché N° : …… ayant pour objet : …… à la ferme expérimentale de Ben Guérir / Université Mohamed 6 Polytechnique de ben Guérir.

Monsieur …… , est invité à commencer les travaux objet du marché relatif au BC N° …… à compter du ……

Le présent ordre de service certifié conforme à la minute inscrite au registre sera notifié à Monsieur M. ……, le représentant de la …….. par Monsieur le responsable de la ferme expérimentale de l'UM6P.

Client	Fournisseur
Université Mohamed 6 Polytechnique	Société ……..

FIGURE A.4 – Fiche F.84 - Fiche ordre de service (document fourni)

A.2 Annexe B : Structure de la base de données

A.2.1 Schéma de base de données (Prisma)

```
[language=prisma, caption={Sch ma principal de la base de donn es Prism
model User {
  id            String    @id @default(uuid())
  name          String
  email         String    @unique
  password      String
  role          UserRole  @default(CLIENT)
  isVerified    Boolean   @default(false)
  entityId      String?
  entity        Entity?   @relation(fields: [entityId], references: [id])
  projects      Project[]
  reservations  Reservation[]
  createdAt     DateTime  @default(now())
  updatedAt     DateTime  @updatedAt
}

model Entity {
  id            String    @id @default(uuid())
  name          String
  users         User[]
  projects      Project[]
  createdAt     DateTime  @default(now())
  updatedAt     DateTime  @updatedAt
}

model Field {
  id              String    @id @default(uuid())
  name            String
  location        String
  totalSurfaceM2  Float
  freeSurfaceM2   Float
  status          String    @default("ACTIVE")
  notes           String?
  projects        Project[]
  reservations    Reservation[]
  createdAt       DateTime  @default(now())
```

```
  updatedAt       DateTime  @updatedAt
}

model Project {
  id            String    @id  @default(uuid())
  title         String
  description   String?
  status        String    @default("PENDING")
  entityId      String
  entity        Entity    @relation(fields: [entityId], references: [id])
  fieldId       String
  field         Field     @relation(fields: [fieldId], references: [id])
  supervisorId  String
  supervisor    User      @relation(fields: [supervisorId], references: [id
  reservations  Reservation[]
  createdAt     DateTime  @default(now())
  updatedAt     DateTime  @updatedAt
}

model Reservation {
  id                  String    @id  @default(uuid())
  clientId            String
  client              User      @relation(fields: [clientId], references:
  fieldId             String
  field               Field     @relation(fields: [fieldId], references: [
  projectId           String?
  project             Project?  @relation(fields: [projectId], references:
  surfaceM2Requested  Float
  startRequested      DateTime
  endRequested        DateTime
  status              ReservationStatus  @default(PENDING)
  decisionDate        DateTime?
  supervisorId        String?
  supervisor          User?     @relation(fields: [supervisorId], referenc
  createdAt           DateTime  @default(now())
  updatedAt           DateTime  @updatedAt
}

model InventoryItem {
```

```
  id            String    @id  @default(uuid())
  designation   String
  stockQty      Int
  unit          String
  condition     String
  location      String
  createdAt     DateTime  @default(now())
  updatedAt     DateTime  @updatedAt
}

enum UserRole {
  ADMIN
  SUPERVISOR
  CLIENT
}

enum ReservationStatus {
  PENDING
  APPROVED
  REJECTED
}
```

A.3 Annexe C : Extraits de code source représentatif

A.3.1 Configuration du serveur Express (extrait principal)

Listing A.1 – Configuration principale du serveur Express (server.js)

```
const express = require('express');
const cors = require('cors');
const helmet = require('helmet');
const morgan = require('morgan');
const rateLimit = require('express-rate-limit');
require('dotenv').config();

const { PrismaClient } = require('@prisma/client');
const authRoutes = require('./routes/auth');
const userRoutes = require('./routes/users');
const fieldRoutes = require('./routes/fields');
const projectRoutes = require('./routes/projects');
const reservationRoutes = require('./routes/reservations');
const inventoryRoutes = require('./routes/inventory');
```

```
const app = express();
const prisma = new PrismaClient();

// Middleware
app.use(helmet());
app.use(cors({
  origin: process.env.FRONTEND_URL || 'http://localhost:3000',
  credentials: true
}));
app.use(morgan('combined'));
app.use(express.json({ limit: '10mb' }));
app.use(express.urlencoded({ extended: true }));

// Rate limiting
const limiter = rateLimit({
  windowMs: 15 * 60 * 1000, // 15 minutes
  max: 100 // limit each IP to 100 requests per windowMs
});
app.use(limiter);

// Routes
app.use('/auth', authRoutes);
app.use('/users', userRoutes);
app.use('/fields', fieldRoutes);
app.use('/projects', projectRoutes);
app.use('/reservations', reservationRoutes);
app.use('/inventory', inventoryRoutes);

// Error handling middleware
app.use((err, req, res, next) => {
  console.error(err.stack);
  res.status(500).json({
    message: 'Something went wrong!',
    error: process.env.NODE_ENV === 'development' ? err.message : {}
  });
});

const PORT = process.env.PORT || 5000;
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

A.3.2 Composant React principal (extrait)

Listing A.2 – Composant principal de l'application (App.tsx)

```
import React from 'react';
import { BrowserRouter as Router, Routes, Route, Navigate } from 'react-
   router-dom';
import { ThemeProvider, createTheme } from '@mui/material/styles';
import { CssBaseline, Container, Box, CircularProgress } from '@mui/
   material';
import { AuthProvider, useAuth } from './context/AuthContext';
import Login from './pages/Login';
import NavBar from './components/NavBar';
import Users from './pages/Users';
import Fields from './pages/Fields';
import Dashboard from './pages/Dashboard';
import Projects from './pages/Projects';
import Inventory from './pages/Inventory';
import Clients from './pages/Clients';
import Reservations from './pages/Reservations';

const theme = createTheme({
  palette: {
    primary: { main: '#1976d2', light: '#42a5f5', dark: '#1565c0' },
    secondary: { main: '#dc004e', light: '#ff5983', dark: '#9a0036' },
    background: { default: '#f5f5f5', paper: '#ffffff' },
  },
  typography: { fontFamily: '"Roboto", "Helvetica", "Arial", sans-serif'
      },
  shape: { borderRadius: 12 },
});

const AppRoutes = () => {
  const { user, loading } = useAuth();
  if (loading) {
    return (
      <Box display="flex" justifyContent="center" alignItems="center"
         minHeight="100vh">
        <CircularProgress size={60} />
      </Box>
    );
  }
  return (
    <>
      {user && <NavBar />}
      <Container sx={{ mt: 4, mb: 4, px: { xs: 2, sm: 3, md: 4 },
         maxWidth: '1400px' }}>
        <Routes>
          <Route path="/login" element={user ? <Navigate to="/dashboard"
              /> : <Login />} />
```

```
          <Route path="/dashboard" element={user ? <Dashboard /> : <
             Navigate to="/login" />} />
          <Route path="/users" element={user ? <Users /> : <Navigate to
             ="/login" />} />
          <Route path="/fields" element={user ? <Fields /> : <Navigate
             to="/login" />} />
          <Route path="/projects" element={user ? <Projects /> : <
             Navigate to="/login" />} />
          <Route path="/inventory" element={user ? <Inventory /> : <
             Navigate to="/login" />} />
          <Route path="/clients" element={user ? <Clients /> : <Navigate
              to="/login" />} />
          <Route path="/reservations" element={user ? <Reservations /> :
              <Navigate to="/login" />} />
          <Route path="/" element={<Navigate to={user ? "/dashboard" :
             "/login"} />} />
        </Routes>
      </Container>
    </>
  );
};

function App() {
  return (
    <ThemeProvider theme={theme}>
      <CssBaseline />
      <AuthProvider>
        <Router>
          <AppRoutes />
        </Router>
      </AuthProvider>
    </ThemeProvider>
  );
}

export default App;
```