

RÉPUBLIQUE DU SÉNÉGAL
UNIVERSITÉ GASTON BERGER DE SAINT-LOUIS
U.F.R DES SCIENCES APPLIQUÉES ET TECHNOLOGIES

Département Informatique



Mémoire présenté pour l'obtention du diplôme de Master en Informatique
Option : Gestion de Données et Ingénierie logicielle (GDIL)

Analyse et conception d'une plateforme numérique personnalisée
pour la gestion des projets et activités scientifiques du Centre de
Recherche Agricole de Saint-Louis ISRA/CRA-SL

Présenté par
Mouhamadou Lamine Bathily

Lieu de stage :
ISRA/CRA Saint-Louis



Encadré et supervisé par :
Dr Mamadou Ndiaye (ISRA/CRA Saint-Louis)
Pr Cheikh Mouhamadou Bamba Dione (UGB)

Soutenu le 03 Août 2023 devant le jury composé de :
Daouda SANGARÉ Président Professeur titulaire
Nom2 NOM2 Rapporteur Maître de conférences
Nom3 NOM3 Examinateur Maître-assistant
Nom4 NOM4 Membre Docteur

Dédicace

À la mémoire de,

Remerciements

[Votre contenu ici]

Résumé

[Votre résumé ici]

Abstract

[Your abstract content here]

Notations et Abréviations

[Votre contenu ici - liste des notations]

Table des matières

Dédicace	i
Remerciements	ii
Résumé	iii
Abstract	iv
Notations et Abréviations	v
Introduction	1
1 Présentation Générale	2
1.1 Présentation de la structure d'accueil	2
1.2 Présentation du sujet	3
1.3 Démarche Méthodologique	4
2 Spécifications et Analyse des besoins	5
2.1 Spécifications	5
2.2 Analyse des besoins	8
2.3 Diagrammes UML	9
3 Conception de la solution	26
3.1 Architecture logique	26
3.2 Choix technologiques	30
3.3 Architecture technique	33
4 Mise en oeuvre et résultats	36
4.1 Environnement de développement	36
4.2 Présentation et analyse des résultats	38
4.3 Etude de performances et de coûts	38
Conclusion et Perspectives	39
Références	39

Table des figures

2.1	Diagramme des cas d'utilisation — Vue Globale Modulaire	10
2.2	Diagramme des cas d'utilisation — Module Authentification et Gestion Utilisateurs	10
2.3	Diagramme des cas d'utilisation — Module Gestion des projets	11
2.4	Diagramme des cas d'utilisation — Module Gestion des activités	11
2.5	Diagramme des cas d'utilisation — Module Publication	12
2.6	Diagramme de classes — Modèle Utilisateurs et Authentification	13
2.7	Diagramme de classes — Modèle Projets et Activités	14
2.8	Diagramme de classes — Modèle Convention et Partenariat	15
2.9	Diagramme de classes — Modèle Documents et Données	16
2.10	Diagramme de classes — Modèle Collaboration et Séminaires	17
2.11	Diagramme de classes — Module Administration et audit	18
2.12	Diagramme de séquence — Processus de Connexion	19
2.13	Diagramme de séquence — Processus de Connexion WebSocket	20
2.14	Diagramme de séquence — Processus de création de projet	21
2.15	Diagramme de séquence — Collete de données	22
2.16	Diagramme de composants — Architecture Logicielle de la Plateforme	23
2.17	Diagramme de déploiement — Architecture Physique du Système	24
3.1	Architecture de la couche de présentation de l'application CRA	27
3.2	Architecture de la couche métier et données – API Node.js / Express/Prisma/PostgreSQL	29
3.3	Architecture technique 3-tiers de la solution (Frontend React – Backend Node.js/Express – Données PostgreSQL)	34
4.1	Environnement de développement Visual Studio Code utilisé pour le projet	36
4.2	Dépôt GitHub du projet et historique des versions	37
4.3	Conteneurisation de l'application à l'aide de Docker	37
4.4	Documentation interactive des API REST via Swagger	38

Liste des tableaux

3.1 Analyse des technologies Backend	31
3.2 Analyse des technologies Frontend	32

Introduction

L'agriculture occupe une place centrale dans le développement économique et social du Sénégal. Elle constitue non seulement une source essentielle de revenus pour les populations rurales, mais aussi un levier stratégique pour atteindre la sécurité alimentaire et la souveraineté nationale. Cependant, les transformations rapides observées dans ce secteur exigent aujourd'hui une adaptation constante des pratiques, des méthodes scientifiques et des outils de gestion utilisés par les institutions de recherche.

Le Centre de Recherches Agricoles (CRA) de Saint-Louis, structure de l'Institut Sénégalaïs de Recherches Agricoles (ISRA), joue un rôle majeur dans la production et la diffusion des connaissances scientifiques destinées à améliorer la productivité et la durabilité des systèmes de production dans la vallée du fleuve Sénégal. Malgré son importance, le CRA fait face à plusieurs contraintes dans la gestion de ses données et activités scientifiques. Les rapports, fiches d'activités et données expérimentales sont souvent dispersés, conservés localement, et difficilement accessibles aux équipes de recherche. Cette situation limite la capitalisation des résultats, la collaboration entre chercheurs et la valorisation des productions scientifiques.

Dans un contexte marqué par la transition numérique et la recherche d'efficacité organisationnelle, l'utilisation des technologies de l'information s'impose désormais comme un levier incontournable. La mise en place d'une plateforme numérique collaborative apparaît ainsi comme une solution pertinente pour moderniser la gestion des activités scientifiques, faciliter la centralisation des données et renforcer la communication au sein du centre.

C'est dans cette optique que s'inscrit le présent mémoire, réalisé dans le cadre de mon stage de fin d'études au CRA de Saint-Louis. Ce travail vise à concevoir une application web personnalisée dédiée à la gestion des activités scientifiques, des projets, des documents et des données des chercheurs. L'objectif est d'apporter une solution simple, efficace et adaptée au contexte local, tout en contribuant à la valorisation des résultats de la recherche et à la visibilité du centre.

L'ensemble de cette étude s'articule autour d'une question centrale : comment concevoir une solution numérique durable et adaptée permettant aux acteurs de la recherche agricole de mieux organiser, valoriser et partager leurs productions scientifiques au sein du CRA de Saint-Louis ?

Chapitre 1

Présentation Générale

1.1 Présentation de la structure d'accueil

Le Centre de Recherches Agricoles (CRA) de Saint-Louis est une structure de l'Institut Sénégalaïs de Recherches Agricoles (ISRA). Il a pour mission principale de contribuer à l'amélioration de la productivité agricole, à la sécurité alimentaire et à la résilience des systèmes de production dans la vallée du fleuve Sénégal.

Le CRA joue un rôle central dans la recherche appliquée et le développement de solutions innovantes adaptées aux conditions agroécologiques locales. Ses travaux s'articulent autour de plusieurs thématiques majeures telles que la sélection variétale (notamment le riz et le blé), la gestion durable des ressources naturelles (sols et eau), la protection des cultures, la transformation agroalimentaire et l'agroforesterie.

Le centre mène également des programmes structurants sur la production de semences, l'intensification rizicole, la culture du blé et l'agroécologie. Par ailleurs, il collabore avec de nombreux partenaires nationaux et internationaux afin de renforcer la production scientifique et technique au service de l'agriculture sénégalaise.

Ses principales missions consistent à :

- Générer des connaissances scientifiques et techniques adaptées aux écosystèmes de la vallée ;
- Développer des innovations agricoles favorisant une production durable et compétitive ;
- Encadrer les producteurs dans l'adoption des résultats de la recherche ;
- Participer à la formation des étudiants, chercheurs et agents de développement ;
- Promouvoir une gestion durable des ressources naturelles et des systèmes de production.

1.2 Présentation du sujet

1.2.1 Contexte

Le CRA de Saint-Louis produit chaque année une grande quantité de données scientifiques, de rapports, de fiches d'activités et de documents techniques issus des travaux de recherche menés par les chercheurs et techniciens. Cependant, ces informations sont souvent dispersées, conservées localement et difficilement accessibles. Cette situation limite la capitalisation, le partage et la valorisation des résultats scientifiques, surtout lors du départ de certains agents ou du renouvellement du personnel.

Dans un contexte où la digitalisation des processus scientifiques devient un levier essentiel d'efficacité, il est nécessaire de concevoir une plateforme numérique collaborative, moderne et sécurisée. Cette solution doit permettre à chaque agent (chercheur, cadre, technicien supérieur, ingénieur ou assistant de recherche) de disposer d'un espace personnel de travail pour organiser, suivre et partager ses activités professionnelles.

1.2.2 Problématique

Malgré l'importance stratégique du CRA dans le dispositif national de recherche agricole, il ne dispose pas encore d'un système d'information intégré permettant la gestion centralisée des activités scientifiques et techniques. Les difficultés rencontrées se traduisent notamment par :

- Une dispersion et une faible accessibilité des données et documents de recherche ;
- L'absence d'un système collaboratif favorisant le travail en équipe et le partage d'informations ;
- Une gestion manuelle des projets et des activités rendant le suivi difficile ;
- Un manque d'outils d'analyse et de visualisation des indicateurs de recherche.

Ces constats soulignent la nécessité de mettre en place une plateforme numérique adaptée aux besoins spécifiques du CRA, afin d'assurer une meilleure gestion des projets de recherche, des données scientifiques et des documents produits.

1.2.3 Objectifs

L'objectif principal de ce mémoire est de concevoir, modéliser et développer une plateforme web de gestion des données, projets, documents et activités des chercheurs du CRA de Saint-Louis.

De manière spécifique, il s'agit de :

- Permettre à chaque chercheur de disposer d'un espace personnel pour gérer ses projets, ses tâches et ses documents ;
- Centraliser les données scientifiques et assurer leur traçabilité et leur sécurité ;
- Faciliter la collaboration entre chercheurs à travers le partage de fichiers, la planification de séminaires et le suivi des activités collectives ;
- Offrir à l'administration une vue globale des activités et indicateurs de performance scientifique ;

- Favoriser la valorisation et la diffusion des résultats de recherche produits au CRA.

1.3 Démarche Méthodologique

Pour atteindre les objectifs fixés, une démarche méthodologique rigoureuse a été adoptée, articulée autour de deux axes : la gestion de projet et l'analyse des besoins.

1.3.1 Méthode de gestion de projet

Le projet a été conduit suivant une approche agile inspirée de la méthode *Scrum*, qui favorise la flexibilité et l'adaptation progressive du produit aux besoins des utilisateurs. Cette approche se justifie par le caractère évolutif du projet et la nécessité de tester rapidement les fonctionnalités développées avec les futurs utilisateurs (chercheurs et encadrants).

Les principales étapes de cette approche sont :

- L'analyse des besoins et la rédaction du cahier des charges ;
- La modélisation conceptuelle et logique du système d'information ;
- Le développement incrémental des modules de la plateforme ;
- Les tests, validations et ajustements selon les retours des utilisateurs ;
- La mise en production et la documentation technique.

1.3.2 Méthode d'analyse

L'analyse du système s'est appuyée sur des entretiens et des observations au sein du CRA de Saint-Louis afin de bien comprendre le fonctionnement des processus scientifiques et les attentes des utilisateurs. Les méthodes suivantes ont été mobilisées :

- L'analyse fonctionnelle, pour identifier les besoins métiers et les fonctionnalités clés de la plate-forme ;
- L'analyse des utilisateurs, pour définir les profils et leurs rôles respectifs (chercheur, cadre, technicien, administrateur) ;
- La modélisation UML, pour représenter les cas d'utilisation, la structure des entités et leurs relations ;
- L'analyse technologique, pour sélectionner les outils et langages adaptés au développement (React, Node.js, PostgreSQL, etc.).

Cette démarche intégrée a permis d'assurer la cohérence entre les besoins fonctionnels exprimés et les choix technologiques retenus pour la mise en œuvre du projet.

Chapitre 2

Spécifications et Analyse des besoins

2.1 Spécifications

Cette section présente les spécifications générales du système à développer. L'objectif est de décrire de manière précise et formelle l'ensemble des fonctionnalités attendues, ainsi que les contraintes techniques, organisationnelles et qualitatives qui encadrent la mise en œuvre de la plateforme numérique destinée au CRA de Saint-Louis.

La plateforme doit permettre une gestion centralisée, structurée et sécurisée des activités scientifiques, tout en offrant à chaque type d'utilisateur (chercheur, cadre, technicien, assistant de recherche, administrateur) un espace de travail adapté à ses responsabilités. Les spécifications qui suivent constituent la base du cahier des charges fonctionnel.

2.1.1 Spécifications fonctionnelles

Les spécifications fonctionnelles décrivent l'ensemble des services que le système doit fournir aux utilisateurs. Elles ont été identifiées à partir d'entretiens, d'observations de terrain ainsi que du cadrage établi par l'équipe du CRA.

1. Gestion des utilisateurs

Le système devra :

- Permettre à l'administrateur de créer, modifier et supprimer des comptes utilisateurs ;
- Affecter à chaque utilisateur un rôle spécifique (chercheur, coordonnateur, cadre, technicien supérieur, assistant de recherche, administrateur) ;
- Garantir à chaque rôle un ensemble de droits d'accès et de permissions personnalisées ;
- Permettre la connexion sécurisée via email et mot de passe.

2. Espace personnel de travail

Chaque utilisateur devra disposer d'un espace dédié permettant :

- L'accès à un tableau de bord personnalisé ;
- La consultation des projets, activités, tâches, documents et données auxquels il est associé ;
- La gestion de son profil et de ses informations personnelles.

3. Gestion des projets et activités de recherche

Le système devra permettre :

- La création, modification et archivage de projets scientifiques ;
- L'ajout et la mise à jour des fiches d'activités liées à chaque projet ;
- L'assignation de tâches à des assistants de recherche ou techniciens ;
- Le suivi de l'avancement des projets et des activités.

4. Gestion des documents et données scientifiques

La plateforme devra :

- Autoriser le téléversement, le classement, la consultation et la suppression de documents (PDF, Word, Excel, CSV, images) ;
- Permettre l'importation et l'exportation de données au format CSV ou XLSX ;
- Garantir la traçabilité des documents publiés (rapports, fiches d'activités, fiches techniques, données expérimentales).

5. Outils de collaboration

La plateforme doit inclure :

- Un module de planification et gestion des séminaires (calendrier, invitations, participation) ;
- Le partage de documents entre utilisateurs ;
- Des fonctionnalités de commentaires sur les projets, activités ou tâches.

6. Tableaux de bord et statistiques

Le système devra :

- Offrir aux chercheurs une visualisation des données collectées (courbes, diagrammes, indicateurs) ;
- Fournir à l'administrateur une vue globale de l'ensemble des ressources du système (projets, activités, documents, données, utilisateurs).

7. Génération de rapports

La plateforme doit permettre :

- La génération automatisée de rapports PDF ;
- L'exportation de documents et données pour les besoins administratifs ou scientifiques.

2.1.2 Spécifications non fonctionnelles

Les spécifications non fonctionnelles définissent les exigences de qualité et les contraintes techniques auxquelles le système doit répondre pour assurer sa robustesse et sa pérennité.

1. Performance

- Le système doit répondre rapidement aux requêtes (moins de 3 secondes pour les opérations courantes) ;
- Les pages contenant des visualisations doivent s'afficher sans latence notable.

2. Sécurité

- Authentification sécurisée via JWT ;
- Chiffrement des mots de passe ;
- Gestion stricte des rôles et permissions ;
- Protection contre les attaques web courantes (XSS, CSRF, injections).

3. Scalabilité

- L'architecture doit permettre l'ajout futur de nouveaux modules ;
- La base de données doit supporter une croissance significative du volume de données.

4. Disponibilité

- Le système doit être accessible à distance 24h/24 ;
- Les sauvegardes doivent être régulières et automatisées.

5. Ergonomie et accessibilité

- Interface simple, intuitive et responsive ;
- Navigation claire, adaptée aux niveaux variés de maîtrise numérique.

6. Maintenabilité

- Le code doit être structuré et documenté selon les bonnes pratiques ;
- Le système doit être facilement extensible par les équipes techniques du CRA.

Les spécifications fonctionnelles et non fonctionnelles présentées précédemment définissent le cadre général du système à développer, tant du point de vue des fonctionnalités attendues que des exigences de qualité, de sécurité et de performance.

Cependant, ces spécifications ne peuvent être pleinement comprises sans une analyse approfondie des besoins réels des utilisateurs et du contexte organisationnel du CRA de Saint-Louis. C'est dans

cette optique que la section suivante est consacrée à l’analyse détaillée des besoins métiers, utilisateurs et institutionnels.

2.2 Analyse des besoins

L’analyse des besoins vise à identifier les attentes des utilisateurs, les processus métiers existants, les contraintes institutionnelles et les fonctionnalités indispensables pour répondre aux objectifs définis dans le chapitre précédent.

Cette analyse se fonde sur plusieurs séances d’observation au CRA, des entretiens avec des chercheurs, techniciens, cadres et administrateurs, ainsi que sur l’étude des documents produits.

1. Besoins métiers

Les acteurs du CRA ont exprimé les besoins suivants :

- Centraliser l’ensemble des documents scientifiques pour réduire la dispersion ;
- Organiser les projets et activités au sein d’un système unique et structuré ;
- Faciliter le partage de données et la collaboration entre équipes ;
- Suivre l’évolution des travaux de recherche via des tableaux de bord ;
- Garantir la traçabilité des documents et données collectées ;
- Disposer d’un espace personnel structuré.

2. Besoins des utilisateurs

Chercheurs / Coordonnateurs

- Gérer leurs projets, activités, données et documents ;
- Collaborer avec assistants et techniciens ;
- Générer et archiver des rapports ;
- Visualiser les statistiques liées à leurs travaux.

Assistants de recherche / Techniciens

- Accéder aux tâches assignées ;
- Télécharger, téléverser et consulter des documents ;
- Renseigner des données via des formulaires dynamiques.

Administrateur

- Gérer utilisateurs, rôles, documents, données, séminaires ;
- Superviser l’activité globale du CRA ;
- Accéder à toutes les statistiques.

3. Contraintes organisationnelles

- Diversité des profils utilisateurs ;
- Forte production de documents hétérogènes ;
- Besoin d'un accès à distance pour les missions ;
- Nécessité d'une solution modulaire et évolutive.

L'analyse des besoins a permis de mettre en évidence les attentes fonctionnelles des différents acteurs, les processus métiers existants ainsi que les contraintes organisationnelles propres au CRA de Saint-Louis.

Afin de traduire ces besoins de manière formelle, structurée et compréhensible, une modélisation du système s'avère nécessaire. Cette modélisation repose sur l'utilisation des diagrammes UML, qui constituent un outil de référence pour représenter les fonctionnalités, la structure et le comportement du système avant sa phase de conception technique.

2.3 Diagrammes UML

Les diagrammes UML (Unified Modeling Language) permettent de représenter de manière graphique et formelle les différents aspects du système : structure, comportement, interactions et architecture. Dans le cadre de la plateforme du CRA de Saint-Louis, plusieurs types de diagrammes ont été utilisés pour modéliser les besoins fonctionnels et l'architecture logicielle. Chaque type de diagramme UML utilisé apporte une vue complémentaire du système : les diagrammes de cas d'utilisation décrivent les interactions fonctionnelles, les diagrammes de classes modélisent la structure statique des données, les diagrammes de séquences illustrent les scénarios dynamiques, tandis que les diagrammes de composants et de déploiement permettent de visualiser l'architecture logicielle et physique de la solution.

2.3.1 Diagramme des cas d'utilisations

Le diagramme des cas d'utilisations est un diagramme comportemental permettant de représenter les interactions entre les utilisateurs du système (acteurs) et les différentes fonctionnalités offertes par la plateforme. Il permet d'identifier clairement le périmètre fonctionnel du système ainsi que les rôles et responsabilités de chaque utilisateur : chercheur, technicien, assistant de recherche, coordonnateur et administrateur.

Les cas d'utilisation ont été organisés en cinq modules principaux en raison de la complexité fonctionnelle du système : (i) Vue Globale modulaire, (ii) gestion des utilisateurs, (iii) gestion des projets, (iv) gestion des activités, (v) gestion publication et formation

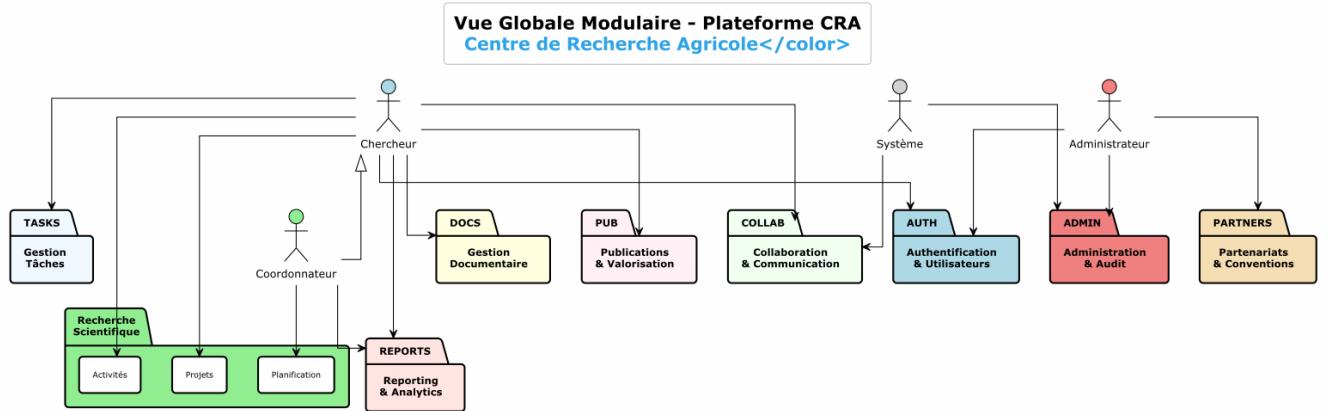


FIGURE 2.1 – Diagramme des cas d'utilisation — Vue Globale Modulaire

Description : Ce diagramme représente les différents module permettant aux chercheurs , coordonnateurs et administrateur de gérer les projets de recherche, les fiches d'activités, les tâches et le suivi scientifique.

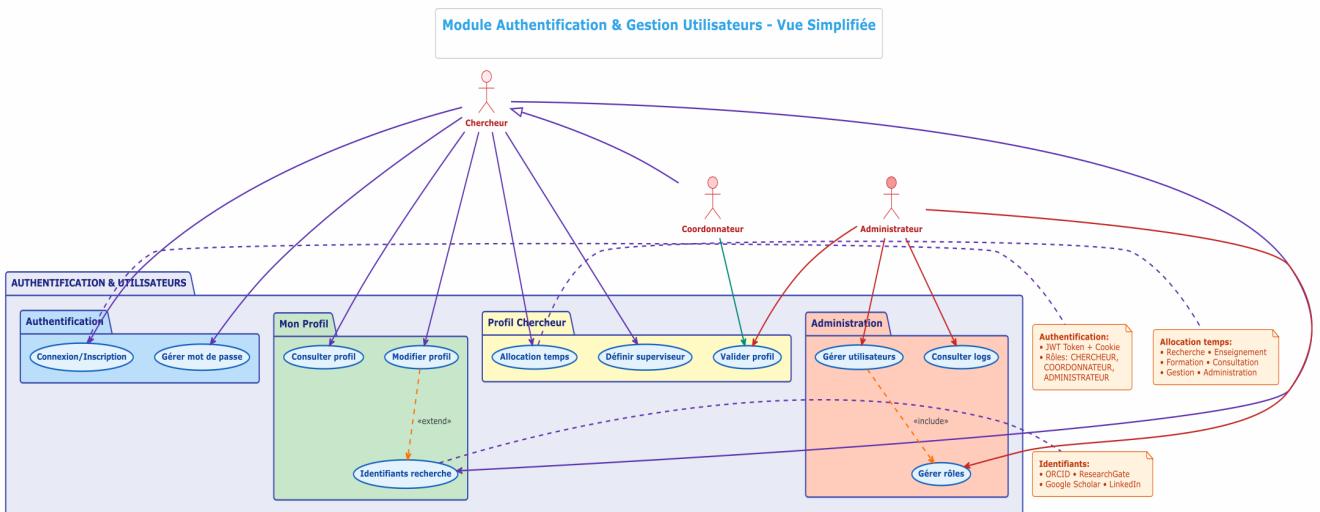


FIGURE 2.2 – Diagramme des cas d'utilisation — Module Authentification et Gestion Utilisateurs

Description : Ce diagramme illustre les interactions entre l'administrateur et la plateforme pour la gestion des comptes utilisateurs : création, modification, suppression et assignation des rôles. Il constitue un module essentiel pour la sécurité et la gestion des accès.

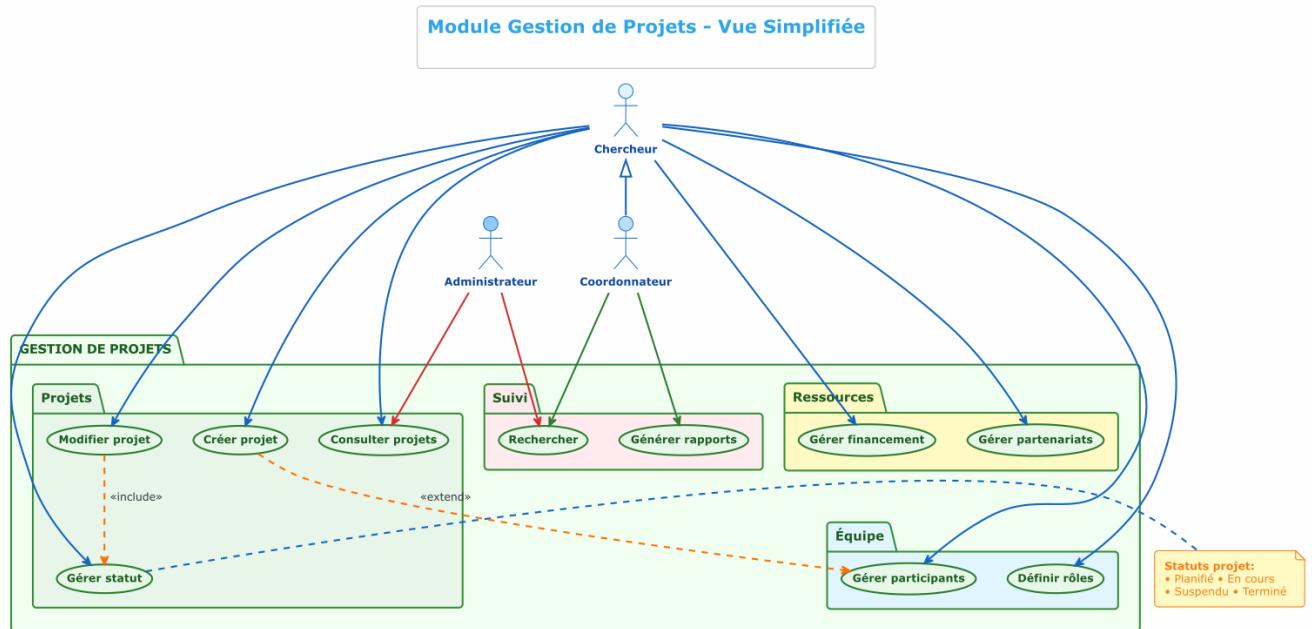


FIGURE 2.3 – Diagramme des cas d'utilisation — Module Gestion des projets

Description : Ce diagramme représente les fonctionnalités permettant aux chercheurs et coordonnateurs de gérer les projets de recherche.

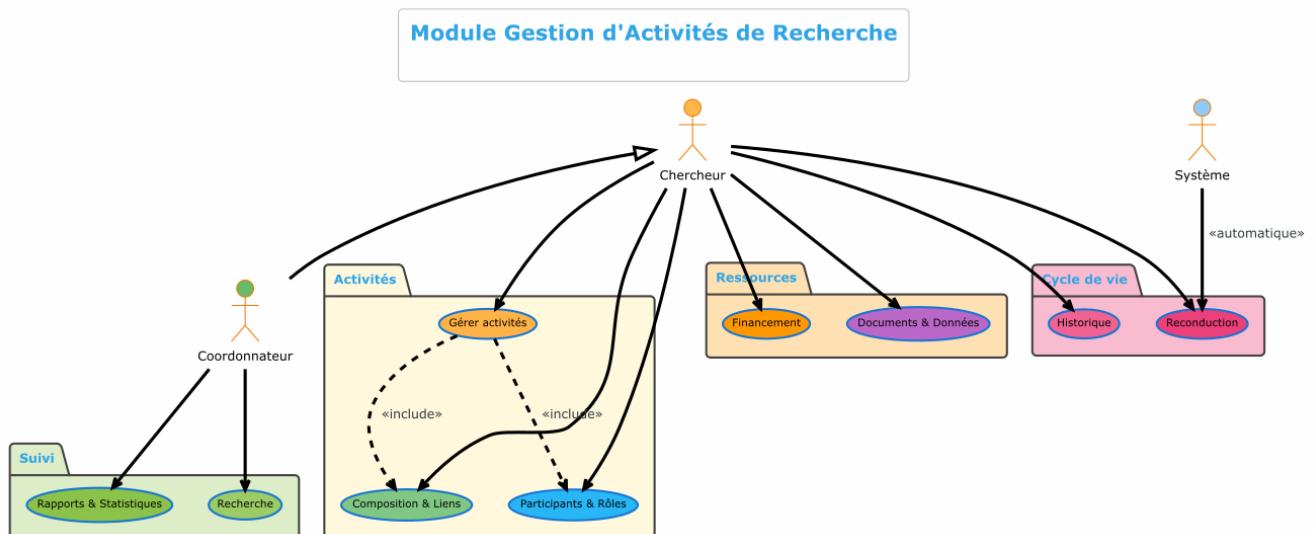


FIGURE 2.4 – Diagramme des cas d'utilisation — Module Gestion des activités

Description : Ce diagramme représente les fonctionnalités permettant aux chercheurs et coordonnateurs de gérer les fiches d'activités, les tâches et le suivi scientifique. Il met en évidence les liens entre les acteurs impliqués dans les activités de production scientifique.

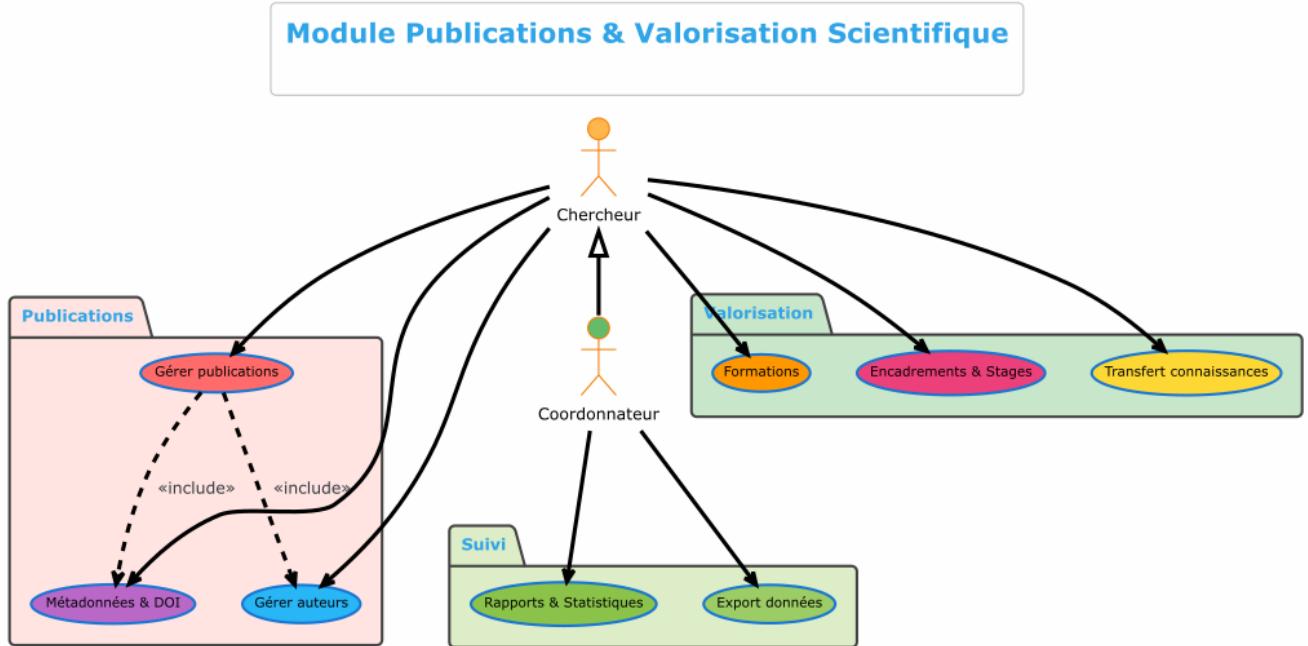


FIGURE 2.5 – Diagramme des cas d'utilisation — Module Publication

Description : Le diagramme de cas d'utilisation présenté illustre le fonctionnement du module Publications et Valorisation Scientifique de la plateforme développée pour le Centre de Recherches Agricoles (CRA) de Saint-Louis. Ce module a pour objectif principal de soutenir la production scientifique, la capitalisation des connaissances et la diffusion des résultats de recherche.

2.3.2 Diagramme de classes

Le diagramme de classes est un diagramme structurel décrivant les entités du système, leurs attributs, leurs méthodes et les relations entre elles. Il constitue la base de la modélisation du système, notamment pour la conception de la base de données et l'architecture backend.

En raison de la richesse du modèle de données de la plateforme du CRA, le diagramme de classes a été découpé en six sous-modèles : (i) utilisateurs et authentification, (ii) projets et activités, (iii) documents et données scientifiques, (iv) Gestion Documentaire (v) collaboration et séminaires, (vi) Administration et audit.

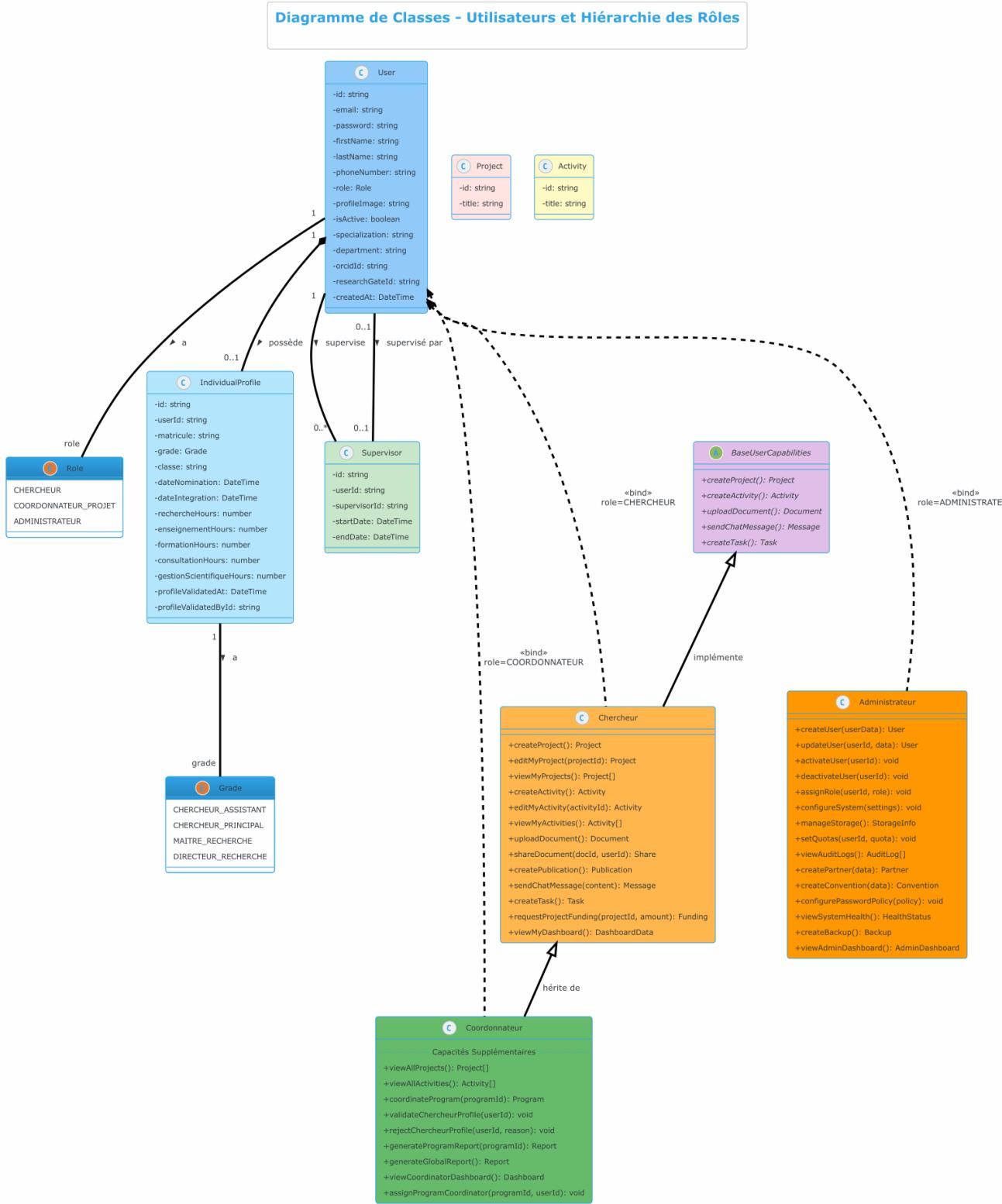


FIGURE 2.6 – Diagramme de classes — Modèle Utilisateurs et Authentification

Description : Ce diagramme présente les classes liées à la gestion des utilisateurs : **Utilisateur**, **Role**, **Compte** ainsi que les relations définissant les rôles, permissions et mécanismes d'authentification.

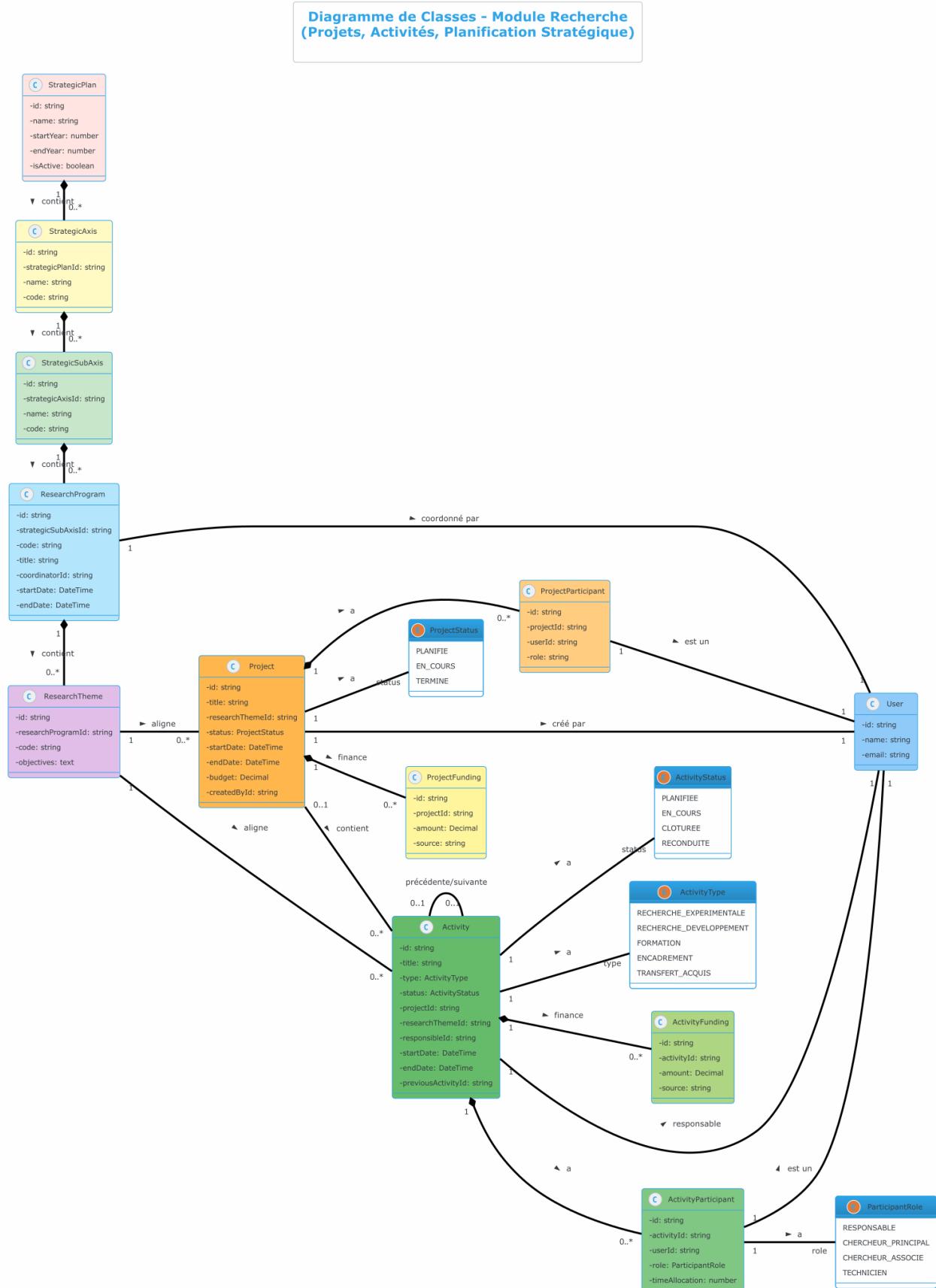


FIGURE 2.7 – Diagramme de classes — Modèle Projets et Activités

Description : Ce schéma représente la structure des entités liées aux projets de recherche : ProjetRecherche, Activite, Tache, et leurs relations. Il constitue le cœur du modèle scientifique de la plateforme.

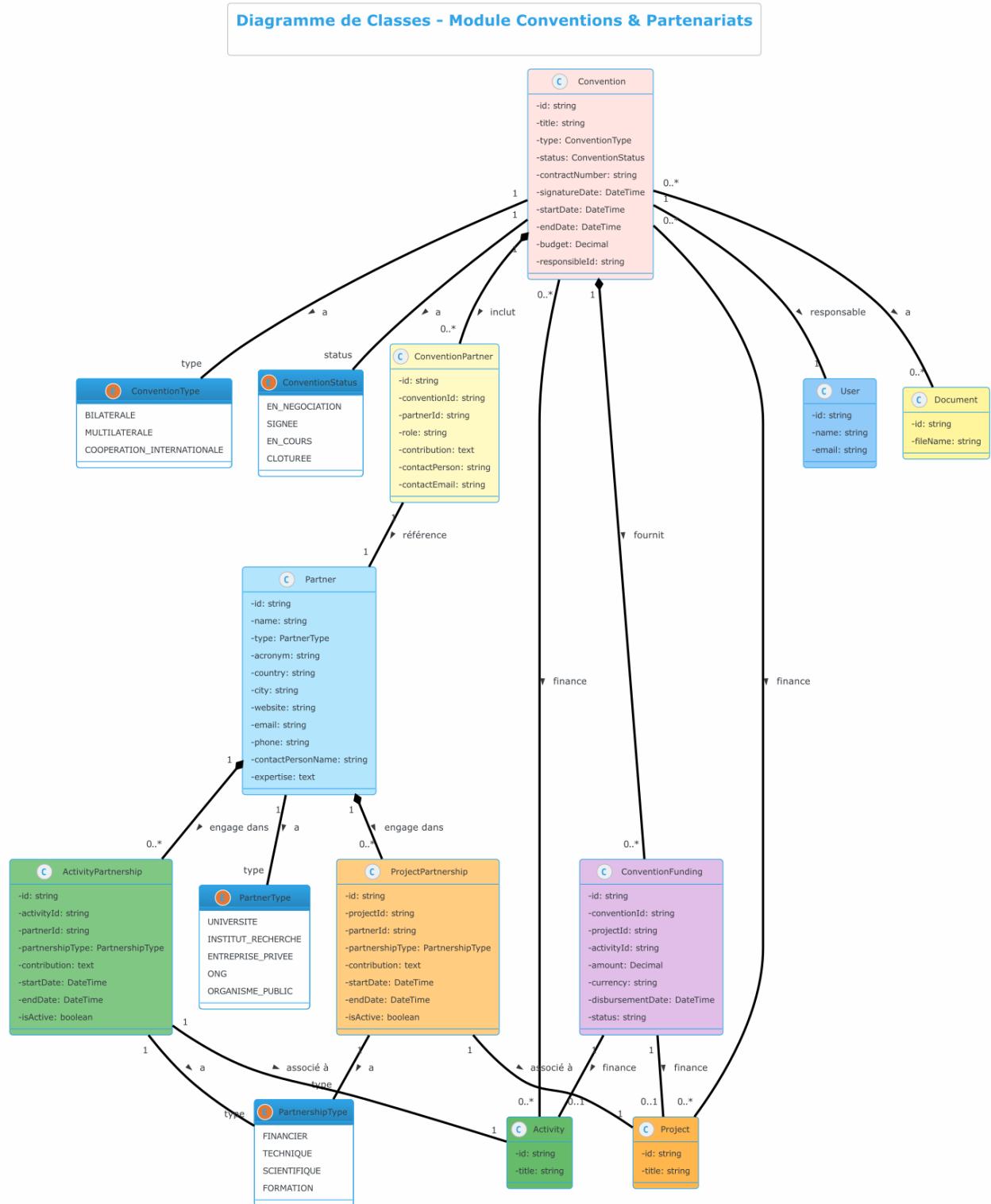


FIGURE 2.8 – Diagramme de classes — Modèle Convention et Partenariat

Description : Le diagramme de classes présenté modélise la structure statique du module Conventions et Partenariats de la plateforme du Centre de Recherches Agricoles (CRA) de Saint-Louis. Ce module a pour objectif de gérer les relations institutionnelles, contractuelles et financières entre le CRA et ses partenaires nationaux et internationaux.

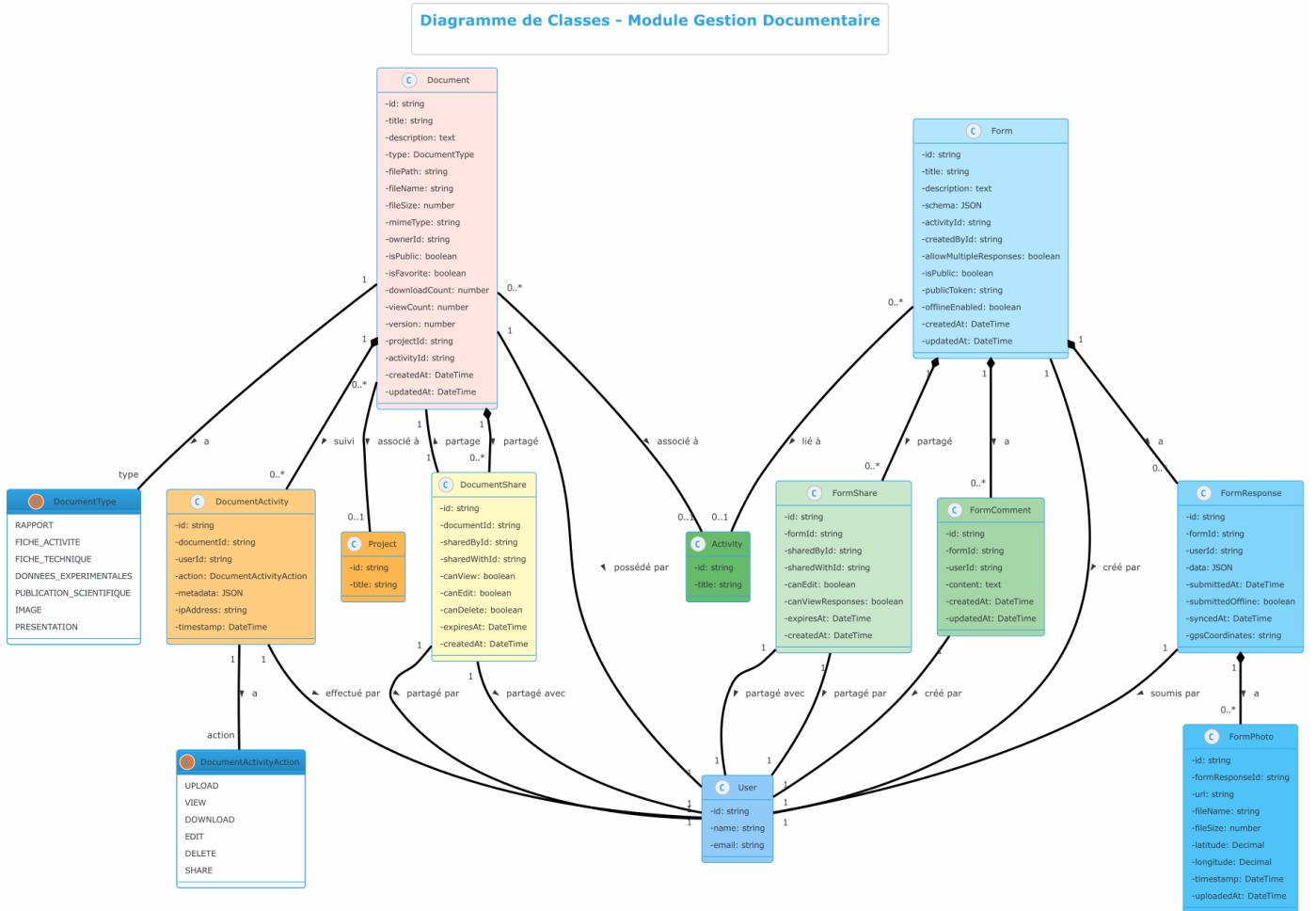


FIGURE 2.9 – Diagramme de classes — Modèle Documents et Données

Description : Ce diagramme modélise la gestion des documents, fichiers, données expérimentales et métadonnées scientifiques. Il montre les relations entre documents, projets, activités et utilisateurs.

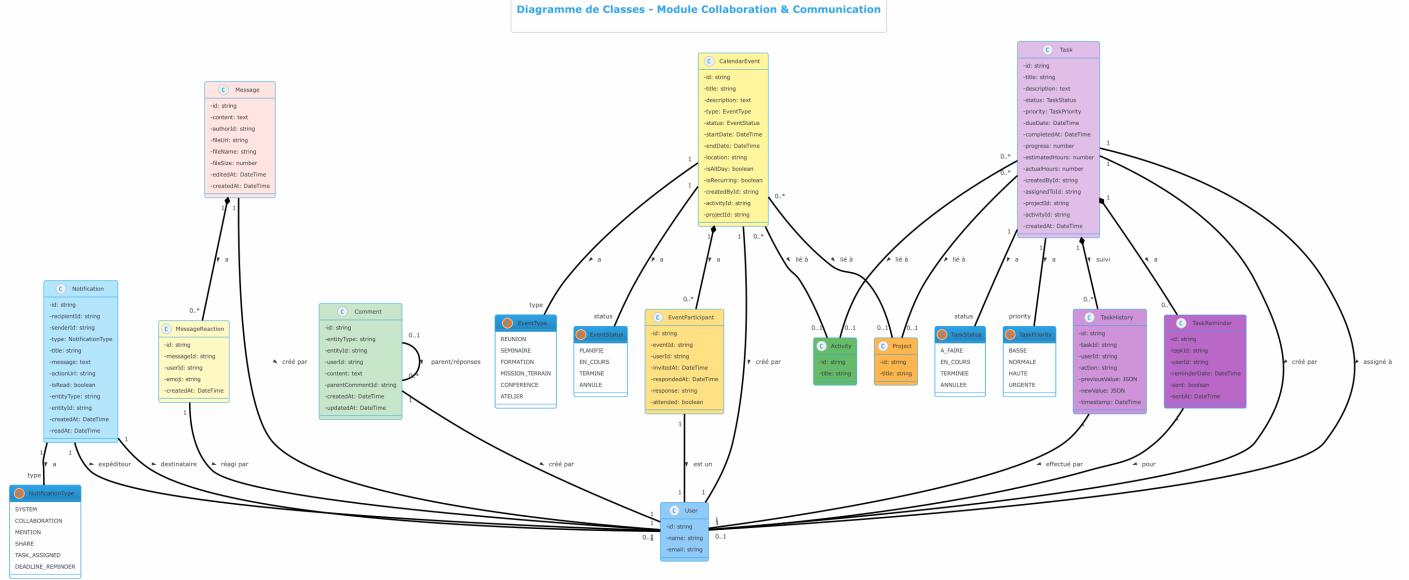


FIGURE 2.10 – Diagramme de classes — Modèle Collaboration et Séminaires

Description : Ce diagramme détaille les classes impliquées dans la planification de séminaires, la participation, les commentaires et autres interactions collaboratives entre chercheurs.

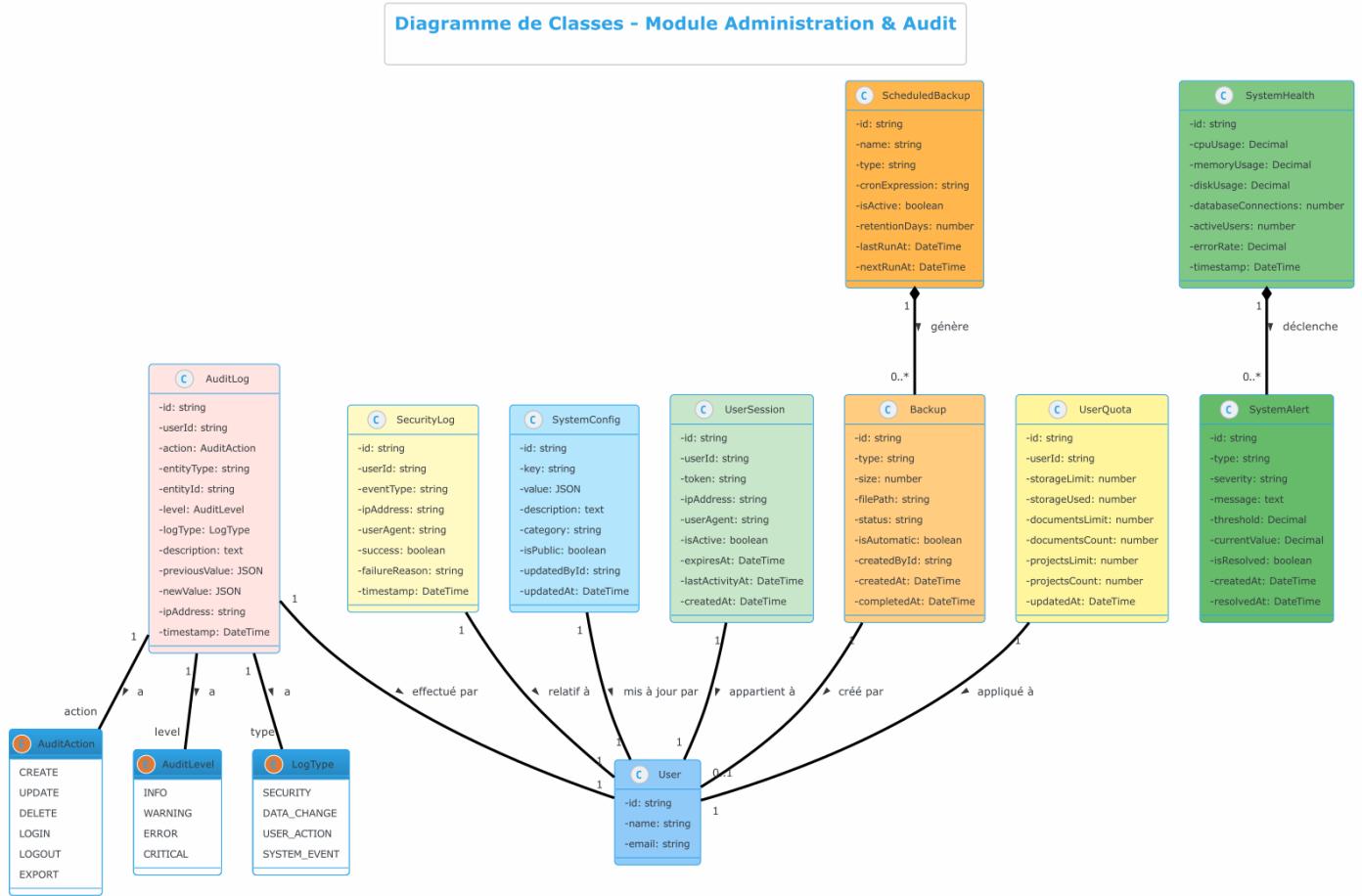


FIGURE 2.11 – Diagramme de classes — Module Administration et audit

Description : Le diagramme de classes présenté modélise l'architecture statique du module Administration et Audit de la plateforme du Centre de Recherches Agricoles (CRA) de Saint-Louis. Ce module joue un rôle central dans la gouvernance du système en assurant la sécurité, la traçabilité, la configuration, la supervision et la continuité de service.

2.3.3 Diagramme de séquences

Le diagramme de séquences est un diagramme dynamique montrant l'ordre chronologique des messages échangés entre les composants du système lors d'un scénario donné. Il permet d'illustrer le comportement opérationnel du système.

Quatre scénarios clés ont été représentés : (i) Processus de Connexion., (ii) Connection WebSocket (iii) création de projet (iv) collecte de données scientifiques

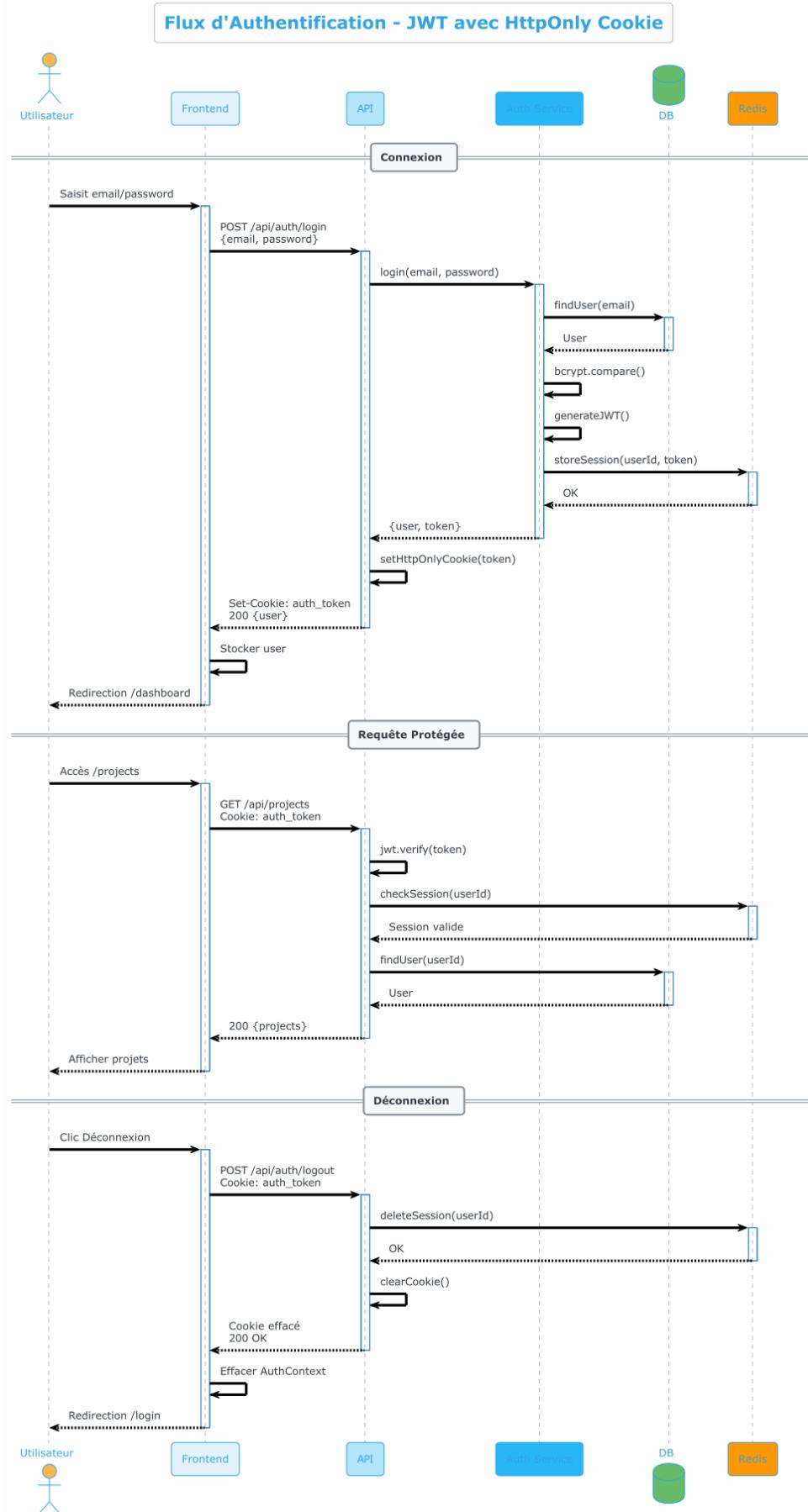


FIGURE 2.12 – Diagramme de séquence — Processus de Connexion

Description : Ce diagramme illustre les interactions entre l'utilisateur, le frontend, le backend et le module d'authentification lors de la connexion. Il met en évidence la génération et la vérification du jeton JWT.

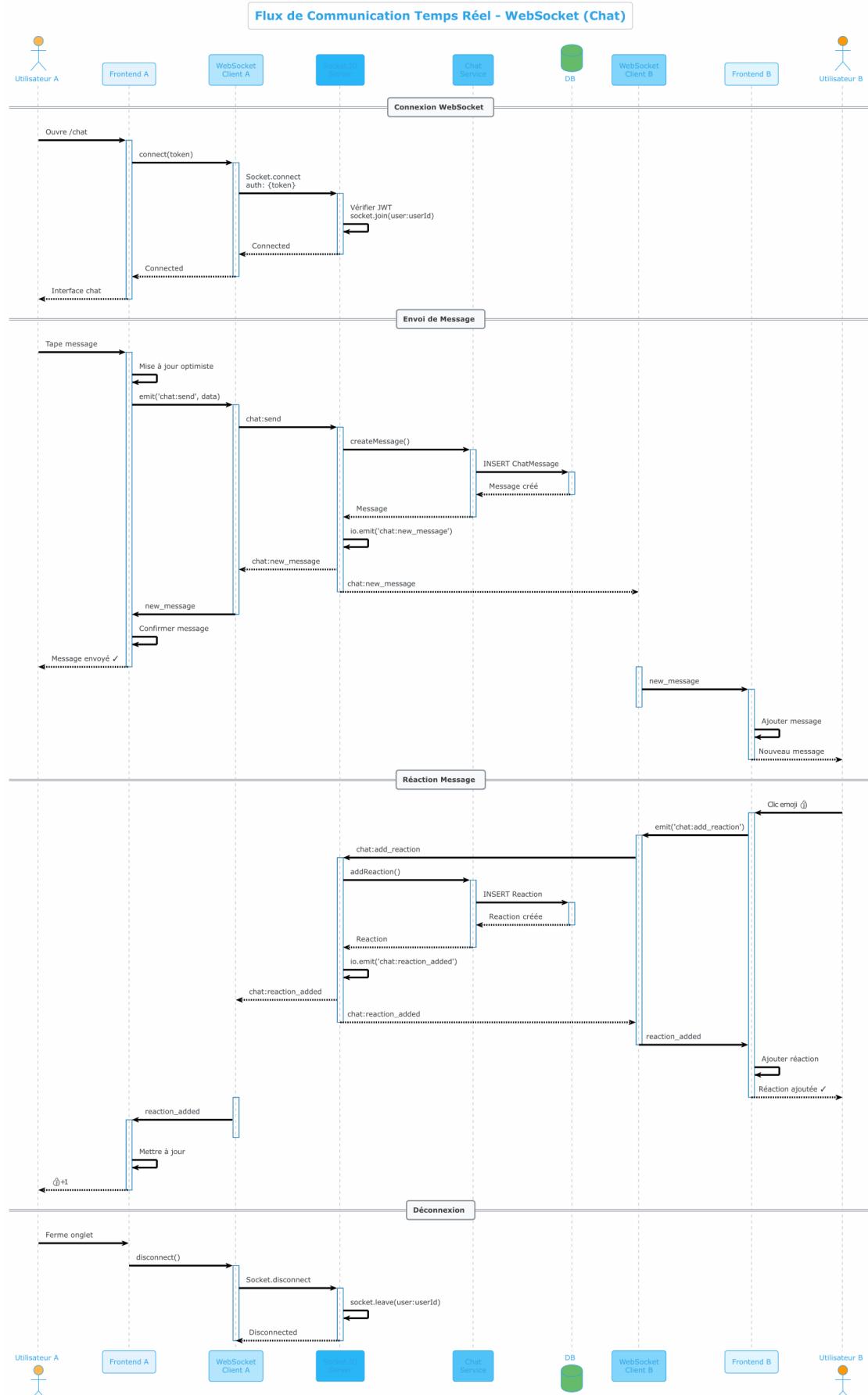


FIGURE 2.13 – Diagramme de séquence — Processus de Connexion WebSocket

Description : Ce diagramme illustre les interactions entre l'utilisateur, le frontend, le backend et le module d'authentification lors de la connexion WebSocket. Il met en évidence le processus de connexion des websockets et la vérification du jeton JWT.

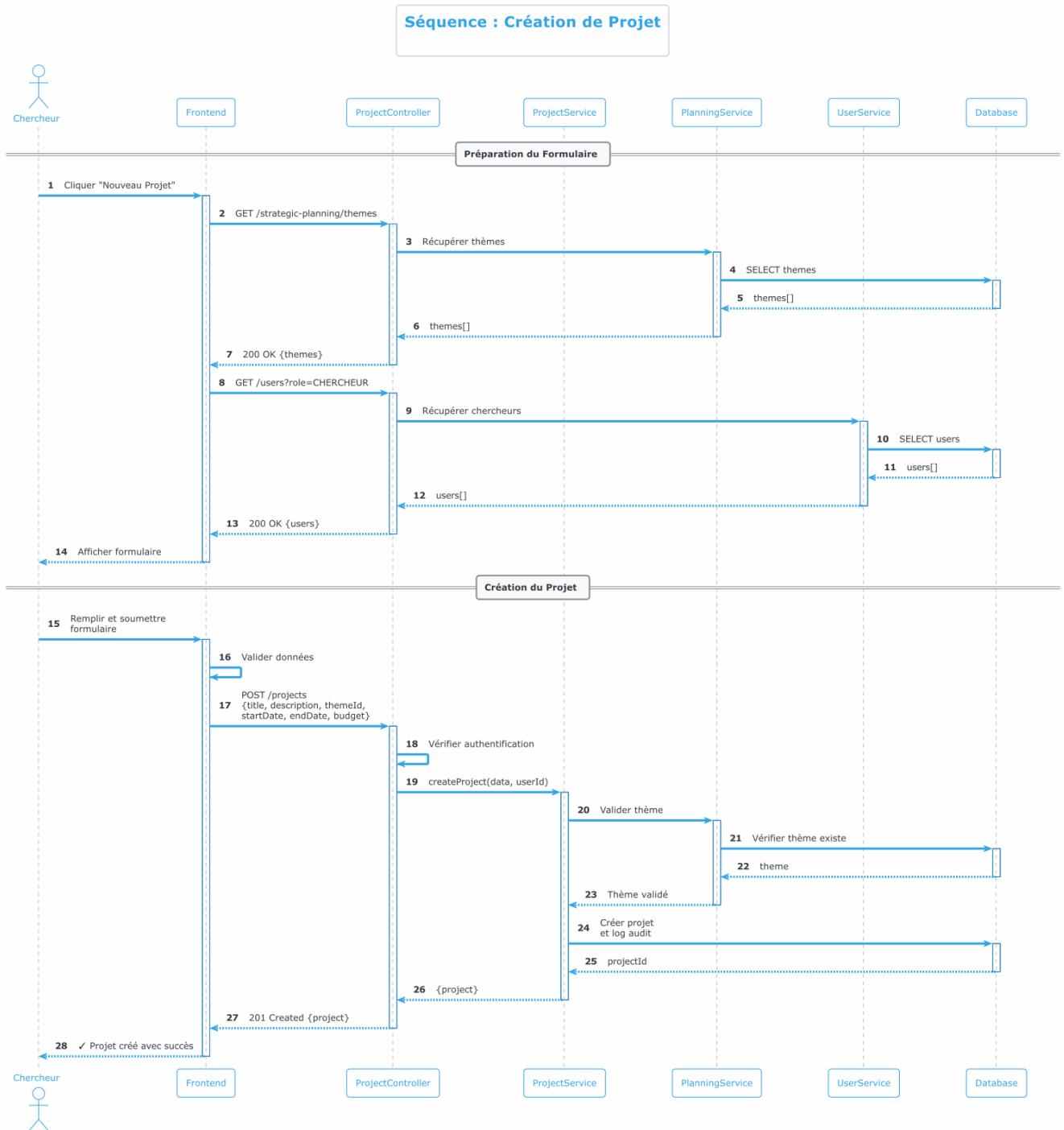


FIGURE 2.14 – Diagramme de séquence — Processus de création de projet

Description : Ce schéma montre les différentes étapes permettant à un chercheur de créer un projet : saisie, validation, enregistrement en base de données et confirmation dans l'interface.

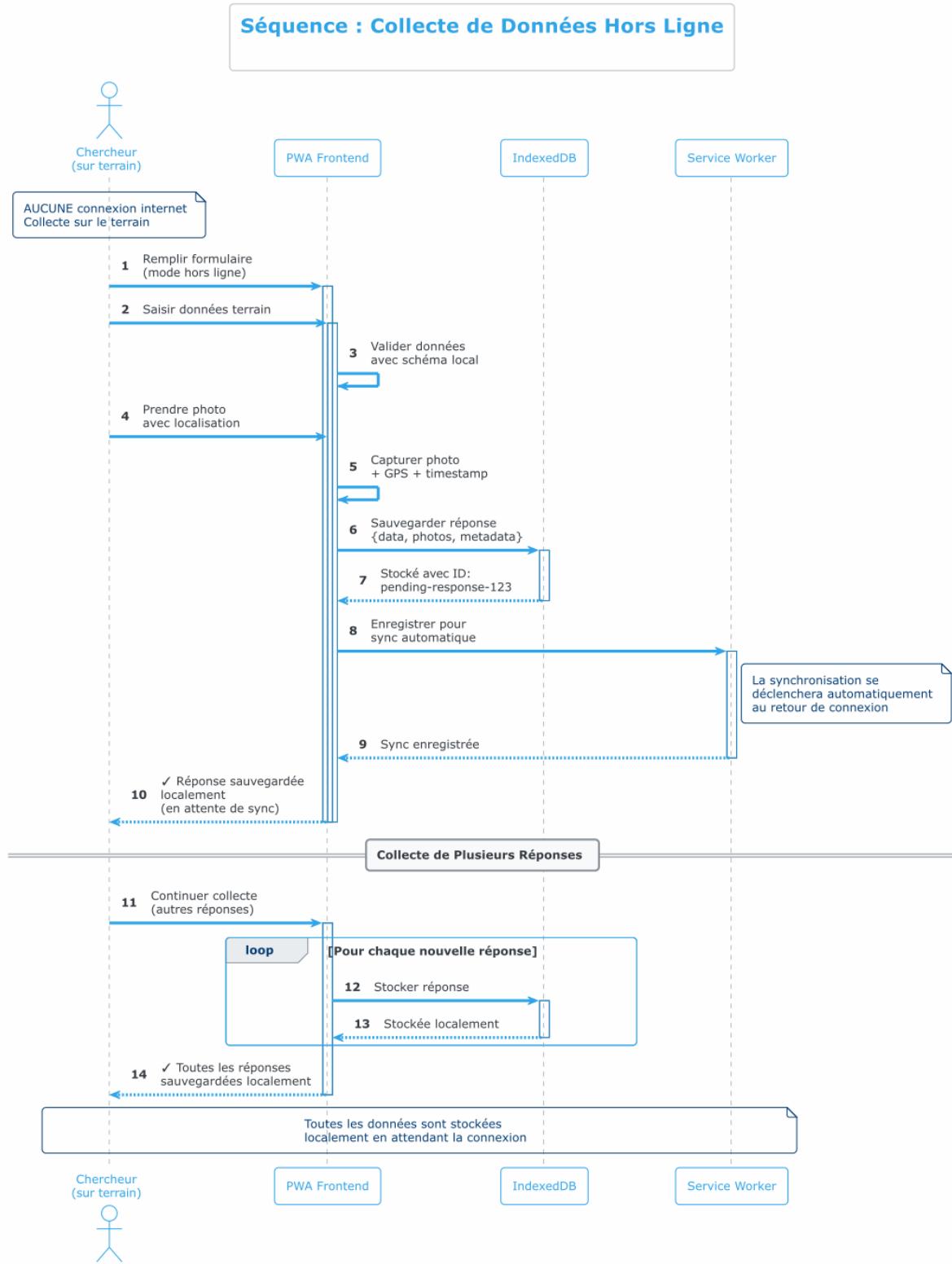


FIGURE 2.15 – Diagramme de séquence — Collete de données

Description : Ce schéma présente les interactions lors de la collecte de données via un formulaire, la validation des champs, l'enregistrement en base et l'affichage des statistiques dans le tableau de bord du chercheur.

2.3.4 Diagramme de composants

Le diagramme de composants représente l'architecture logicielle du système en illustrant les différents modules logiciels et les interactions entre eux.

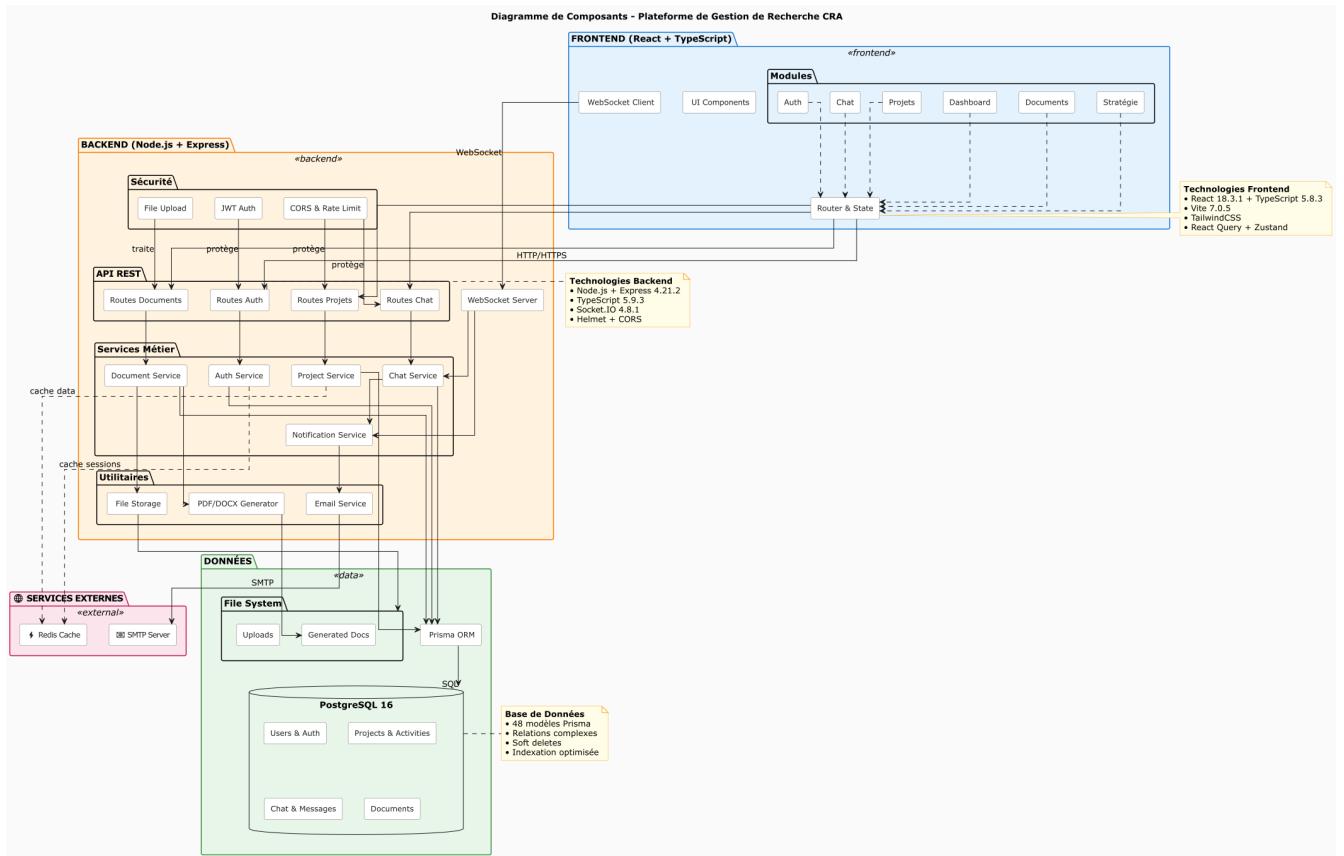


FIGURE 2.16 – Diagramme de composants — Architecture Logicielle de la Plateforme

Description : Ce diagramme montre les composants principaux de l'architecture : le frontend, l'API backend, le module JWT, le service de gestion des fichiers, les modules de statistiques et la base de données PostgreSQL.

2.3.5 Diagramme de déploiement

Le diagramme de déploiement décrit l'architecture physique du système, incluant les serveurs, les environnements et les connexions réseau.

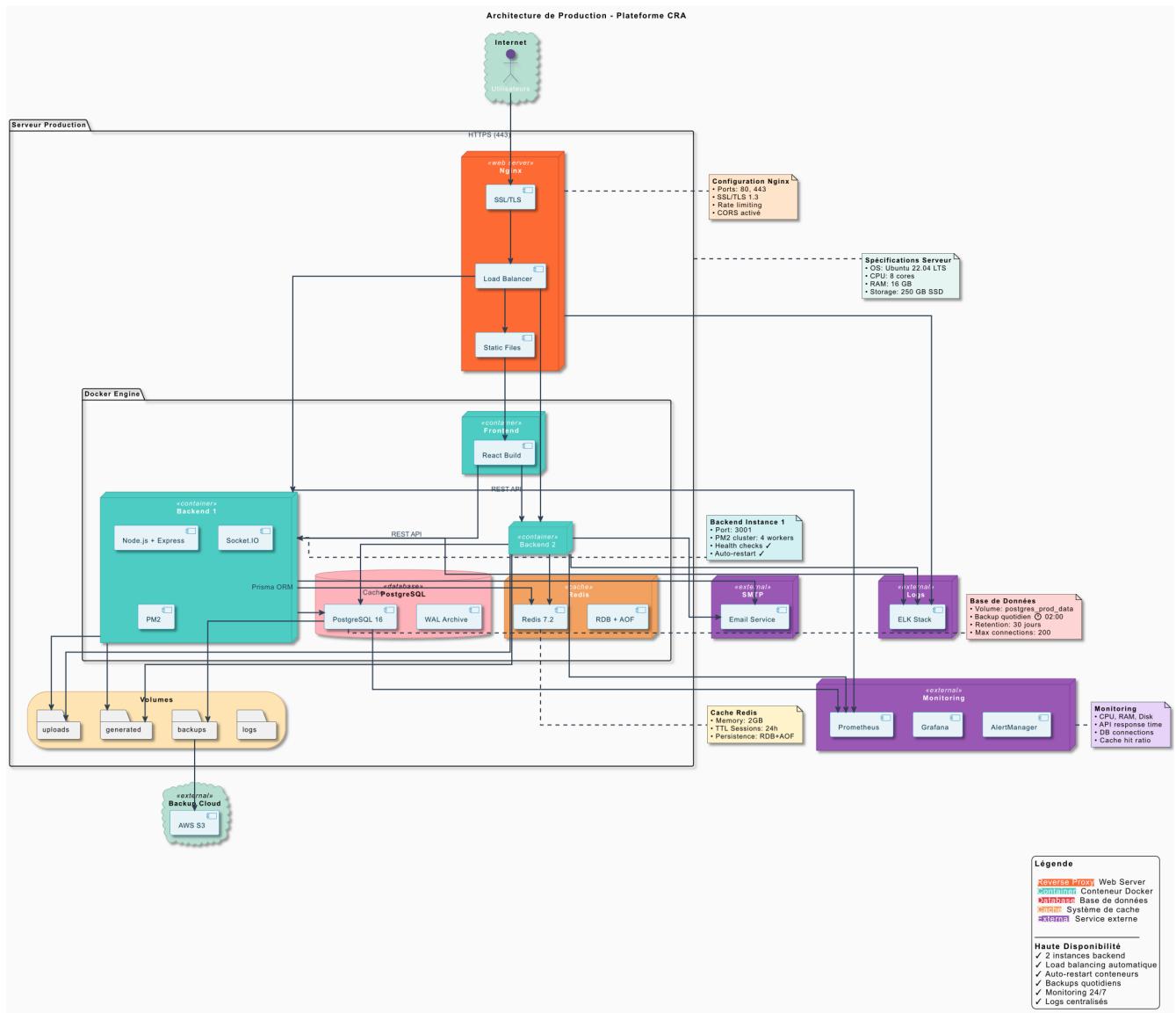


FIGURE 2.17 – Diagramme de déploiement — Architecture Physique du Système

Description : Ce diagramme illustre les nœuds physiques de déploiement : serveur web, serveur applicatif, base de données, ainsi que les environnements de développement, test et production. Il montre également le flux d'échange des données et la communication entre services.

Conclusion partielle

Ce chapitre a permis d'identifier et de formaliser l'ensemble des besoins fonctionnels et non fonctionnels de la plateforme numérique destinée au Centre de Recherches Agricoles (CRA) de Saint-Louis. À travers l'analyse des attentes des utilisateurs, des contraintes organisationnelles et des processus métiers existants, une vision claire et structurée du système cible a été établie.

La modélisation UML réalisée a joué un rôle central dans cette phase d'analyse, en offrant une représentation formelle et cohérente des fonctionnalités, des entités du système, des interactions entre les acteurs et des scénarios clés d'utilisation. Ces modèles constituent un socle essentiel pour garantir une conception maîtrisée et conforme aux exigences exprimées.

Sur la base de ces éléments, le chapitre suivant est consacré à la conception de la solution. Il

présentera l'architecture globale du système, les choix technologiques retenus ainsi que la conception logique et technique de la plateforme, en s'appuyant directement sur les besoins et les modèles définis dans ce chapitre.

Chapitre 3

Conception de la solution

3.1 Architecture logique

L'architecture logique de la solution proposée repose sur une organisation modulaire et orientée services, visant à assurer la séparation des responsabilités, la maintenabilité et l'évolutivité de la plateforme. Elle définit les différentes couches fonctionnelles du système ainsi que leurs interactions, indépendamment des choix techniques et matériels.

La plateforme est structurée autour de trois grandes couches logiques :

- la couche de présentation ;
- la couche métier (ou applicative) ;
- la couche d'accès aux données.

3.1.1 Couche de présentation

La couche de présentation correspond à l'interface utilisateur de la plateforme. Elle permet aux différents profils d'utilisateurs (administrateur, chercheur, cadre, technicien, stagiaire) d'interagir avec le système via un navigateur web.

Cette couche assure :

- l'affichage des interfaces graphiques ;
- la saisie et la validation des données utilisateur ;
- la gestion de la navigation et des rôles ;
- la communication avec la couche métier à travers des API REST.

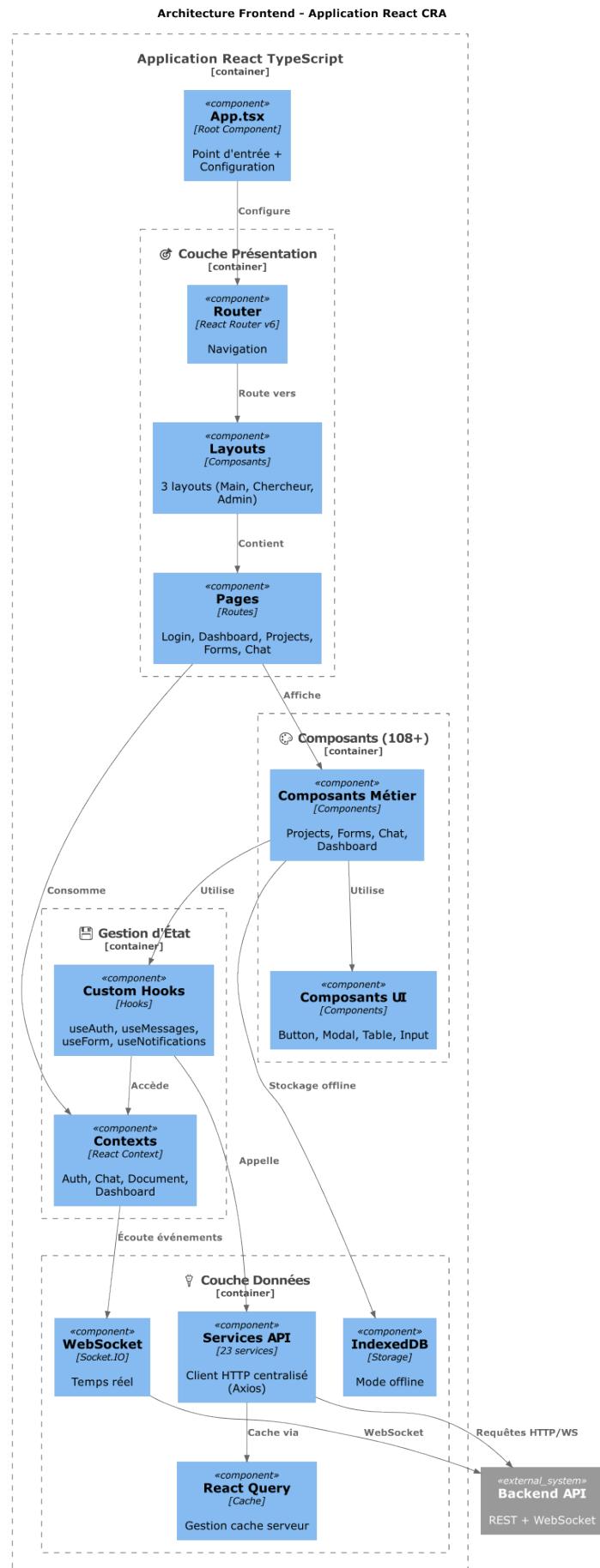


FIGURE 3.1 – Architecture de la couche de présentation de l’application CRA

Cette architecture repose sur une organisation modulaire de l'application React. Le composant `App.tsx` constitue le point d'entrée principal et assure la configuration globale de l'application. La navigation est gérée à l'aide de React Router, permettant l'accès aux différentes pages selon les routes définies.

Les interfaces sont structurées autour de plusieurs layouts adaptés aux profils des utilisateurs, tandis que les pages métiers encapsulent les fonctionnalités principales telles que la gestion des projets, des documents et des tableaux de bord. Les composants métiers s'appuient sur des composants d'interface réutilisables afin de garantir la cohérence visuelle et fonctionnelle.

La gestion de l'état applicatif est assurée par des contextes React et des hooks personnalisés, facilitant le partage des données et la communication entre composants. La couche de données s'appuie sur des services API centralisés, des mécanismes de cache via React Query et des technologies de communication temps réel telles que WebSocket, permettant ainsi une expérience utilisateur fluide et réactive.

3.1.2 Couche métier

La couche métier constitue le cœur fonctionnel de la plateforme. Elle implémente l'ensemble des règles de gestion issues de l'analyse des besoins et des spécifications fonctionnelles.

Elle est organisée en modules correspondant aux domaines fonctionnels du système, notamment :

- gestion des utilisateurs et des rôles ;
- gestion des projets et activités de recherche ;
- gestion des données scientifiques et documents ;
- publications et valorisation scientifique ;
- conventions, partenariats et financements ;
- administration, audit et sécurité.

Cette couche assure également la validation des données, le contrôle des accès et la cohérence des traitements.

3.1.3 Couche d'accès aux données

La couche d'accès aux données est responsable de la persistance et de la récupération des informations. Elle fournit une abstraction entre la couche métier et les systèmes de stockage.

Elle permet :

- la gestion des bases de données relationnelles ;
- l'accès sécurisé aux données ;
- l'optimisation des requêtes ;
- la garantie de l'intégrité et de la cohérence des données.

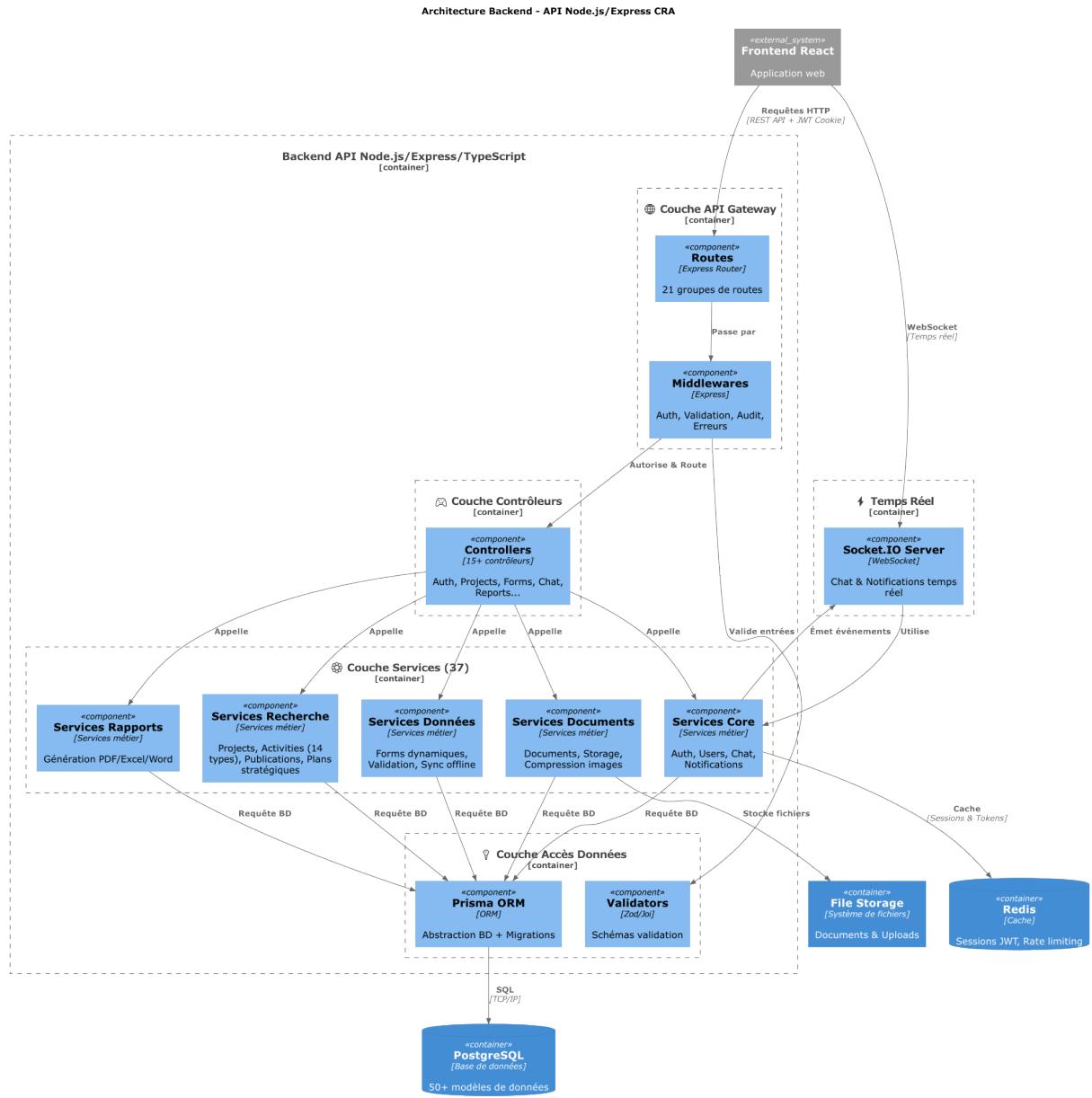


FIGURE 3.2 – Architecture de la couche métier et données – API Node.js / Express/Prisma/PostgreSQL

Comme illustré dans la figure 3.2, la couche métier est organisée selon une architecture en couches clairement séparées. Les requêtes HTTP émises par le frontend sont d'abord reçues par la couche API Gateway, composée des routes Express et des middlewares. Cette couche assure l'authentification, la validation des données, la journalisation des actions et la gestion centralisée des erreurs.

Les contrôleurs constituent le point d'entrée logique des fonctionnalités métier. Ils reçoivent les requêtes validées et déléguent le traitement aux services correspondants. Cette séparation permet de maintenir des contrôleurs légers et focalisés sur la gestion des flux de requêtes.

La couche des services regroupe l'ensemble des règles métier de l'application. Elle est subdivisée en services spécialisés, notamment les services cœur (authentification, gestion des utilisateurs, notifications), les services documents, les services données, les services recherche ainsi que les services de génération de rapports. Cette organisation favorise la modularité, la réutilisabilité du code et l'évolutivité du système.

L'accès aux données est assuré par Prisma ORM, qui fournit une abstraction typée de la base de données PostgreSQL et gère les migrations de schéma. Des validateurs assurent la cohérence et l'intégrité des données échangées entre les différentes couches. Les mécanismes de cache et de gestion de sessions s'appuient sur Redis afin d'améliorer les performances et la scalabilité.

Enfin, la communication en temps réel est prise en charge par un serveur WebSocket basé sur Socket.IO, permettant la gestion du chat et des notifications instantanées. Les fichiers et documents sont stockés dans un système de fichiers dédié, assurant la persistance et la gestion efficace des contenus volumineux. L'architecture retenue pour cette solution repose sur un modèle 3-tiers, combinant une séparation claire entre la couche de présentation, la couche métier et la couche d'accès aux données. Par ailleurs, la couche métier s'inspire du patron de conception MVC, où les contrôleurs Express jouent le rôle de contrôleurs, les modèles Prisma représentent la couche modèle, tandis que la vue est externalisée au niveau du frontend React.

Cette architecture hybride permet de bénéficier à la fois de la rigueur du modèle MVC et de la flexibilité d'une architecture 3-tiers, tout en facilitant la maintenabilité, l'évolutivité et le déploiement indépendant des différentes couches du système.

3.2 Choix technologiques

Les choix technologiques effectués pour la conception et le développement de la plateforme ont été guidés par des critères de performance, de flexibilité, de maintenabilité, de sécurité et d'adéquation avec les besoins du Centre de Recherches Agricoles (CRA) de Saint-Louis. L'architecture retenue repose sur une séparation claire entre le frontend et le backend, communiquant via des API REST.

3.2.1 Technologies Backend

Le backend de la plateforme est développé en utilisant **Node.js** avec le framework **Express.js**, combiné au langage **TypeScript**. Ce choix permet de concevoir des API REST performantes, modulaires et facilement maintenables.

L'accès aux données est assuré par l'ORM **Prisma**, qui facilite la gestion des modèles, des migrations et des requêtes vers la base de données relationnelle **PostgreSQL**.

TABLE 3.1 – Analyse des technologies Backend

Technologie	Points forts	Limites
Node.js	<ul style="list-style-type: none"> — Haute performance en E/S asynchrones — Large écosystème de packages — Adapté aux API REST 	<ul style="list-style-type: none"> — Moins adapté aux traitements très lourds en CPU
Express.js	<ul style="list-style-type: none"> — Framework léger et flexible — Facilité de création des routes — Large adoption communautaire 	<ul style="list-style-type: none"> — Peu d'opinions natives (structure à définir)
TypeScript	<ul style="list-style-type: none"> — Typage statique — Réduction des erreurs à l'exécution — Meilleure maintenabilité du code 	<ul style="list-style-type: none"> — Courbe d'apprentissage initiale
Prisma ORM	<ul style="list-style-type: none"> — Modélisation claire des données — Migrations automatiques — Sécurité des requêtes 	<ul style="list-style-type: none"> — Moins flexible que le SQL brut
PostgreSQL	<ul style="list-style-type: none"> — Robustesse et fiabilité — Support avancé des transactions — Conformité aux standards SQL 	<ul style="list-style-type: none"> — Administration plus exigeante

3.2.2 Technologies Frontend

La couche frontend de la plateforme est développée à l'aide de la bibliothèque **React.js**, associée à l'outil de build **Vite** et au langage **TypeScript**. Cette combinaison permet de concevoir des interfaces utilisateurs dynamiques, réactives et performantes.

TABLE 3.2 – Analyse des technologies Frontend

Technologie	Points forts	Limites
React.js	<ul style="list-style-type: none"> — Architecture basée sur les composants — Large communauté — Excellente gestion de l'état 	<ul style="list-style-type: none"> — Nécessite des bibliothèques supplémentaires
Vite	<ul style="list-style-type: none"> — Démarrage rapide — Hot Module Replacement efficace — Configuration simplifiée 	<ul style="list-style-type: none"> — Moins mature que Webpack
TypeScript	<ul style="list-style-type: none"> — Typage fort — Meilleure lisibilité du code — Facilite le travail en équipe 	<ul style="list-style-type: none"> — Temps de configuration initial

3.2.3 Justification des choix technologiques

Dans le cadre du développement de la plateforme du CRA de Saint-Louis, un accent particulier a été mis sur la performance, la réactivité et la capacité du système à gérer un nombre croissant d'utilisateurs et de requêtes simultanées. Ces exigences ont conduit à privilégier une architecture basée sur des traitements asynchrones et non bloquants. Le modèle de programmation asynchrone constitue aujourd'hui un pilier fondamental des architectures web modernes. Dans une approche synchrone classique, chaque requête traitée par le serveur bloque le fil d'exécution jusqu'à la fin de l'opération, notamment lors des accès à la base de données ou des appels à des services externes. Cette approche entraîne une sous-utilisation des ressources système et limite la capacité de montée en charge de l'application.

À l'inverse, le modèle asynchrone repose sur un mécanisme non bloquant (*non-blocking I/O*), dans lequel le serveur déclenche une opération et poursuit immédiatement le traitement d'autres requêtes. Une fois l'opération terminée, un événement notifie le système afin de finaliser la réponse. Cette approche permet une meilleure exploitation des ressources matérielles et une forte amélioration de la réactivité globale du système. Le choix de **Node.js** s'inscrit pleinement dans cette logique asynchrone. Node.js repose sur un modèle à *thread unique* basé sur une boucle événementielle (*event loop*), capable de gérer simultanément un grand nombre de connexions sans créer de multiples threads système. Cette architecture permet de réduire significativement la consommation mémoire tout en offrant une excellente capacité de montée en charge.

Dans le contexte de la plateforme du CRA, où les opérations incluent de nombreux accès à la base de données, des échanges de fichiers, ainsi que des requêtes concurrentes de plusieurs utilisateurs, ce modèle se révèle particulièrement adapté. Le framework **Express.js** a été retenu pour la conception des API REST en raison de sa légèreté et de sa flexibilité. Contrairement à des frameworks plus contraignants, Express.js adopte une approche minimaliste qui laisse une grande liberté dans l'organisation du code et la structuration des services.

Cette souplesse permet de concevoir des API performantes, facilement extensibles et adaptées à

des architectures modernes telles que les micro-services ou les applications temps réel. Express.js facilite également l'intégration de mécanismes de sécurité, de journalisation et de validation des données. L'utilisation de **TypeScript** aussi bien côté backend que frontend constitue un choix stratégique visant à améliorer la qualité et la maintenabilité du code. Le typage statique permet de détecter un grand nombre d'erreurs dès la phase de développement, de renforcer la lisibilité du code et de faciliter le travail collaboratif au sein de l'équipe.

Dans un projet de taille conséquente comme celui du CRA, TypeScript contribue à la robustesse de l'application et à la pérennité de son évolution. L'ORM **Prisma** a été retenu pour assurer l'interaction avec la base de données **PostgreSQL**. Prisma offre une modélisation claire des schémas de données, une gestion automatisée des migrations et une sécurisation des requêtes, réduisant ainsi les risques d'erreurs et d'injections SQL.

Le choix de PostgreSQL s'explique par sa robustesse, sa fiabilité et son support avancé des transactions, garantissant l'intégrité des données scientifiques et administratives stockées par la plateforme.

3.3 Architecture technique

L'architecture technique décrit l'organisation physique et logicielle des composants du système ainsi que leur mode de déploiement. Elle repose sur une architecture client–serveur distribuée.

La plateforme est constituée des éléments suivants :

- un client web accessible via un navigateur ;
- un serveur applicatif hébergeant les API REST ;
- un serveur de base de données ;
- des services de sécurité et de sauvegarde.

Le client web communique avec le serveur applicatif via le protocole HTTP sécurisé (HTTPS). Le serveur applicatif interagit avec la base de données pour le stockage et la récupération des informations.

Cette architecture permet :

- une séparation claire entre les composants ;
- une meilleure scalabilité ;
- une maintenance facilitée ;
- une sécurisation des échanges.

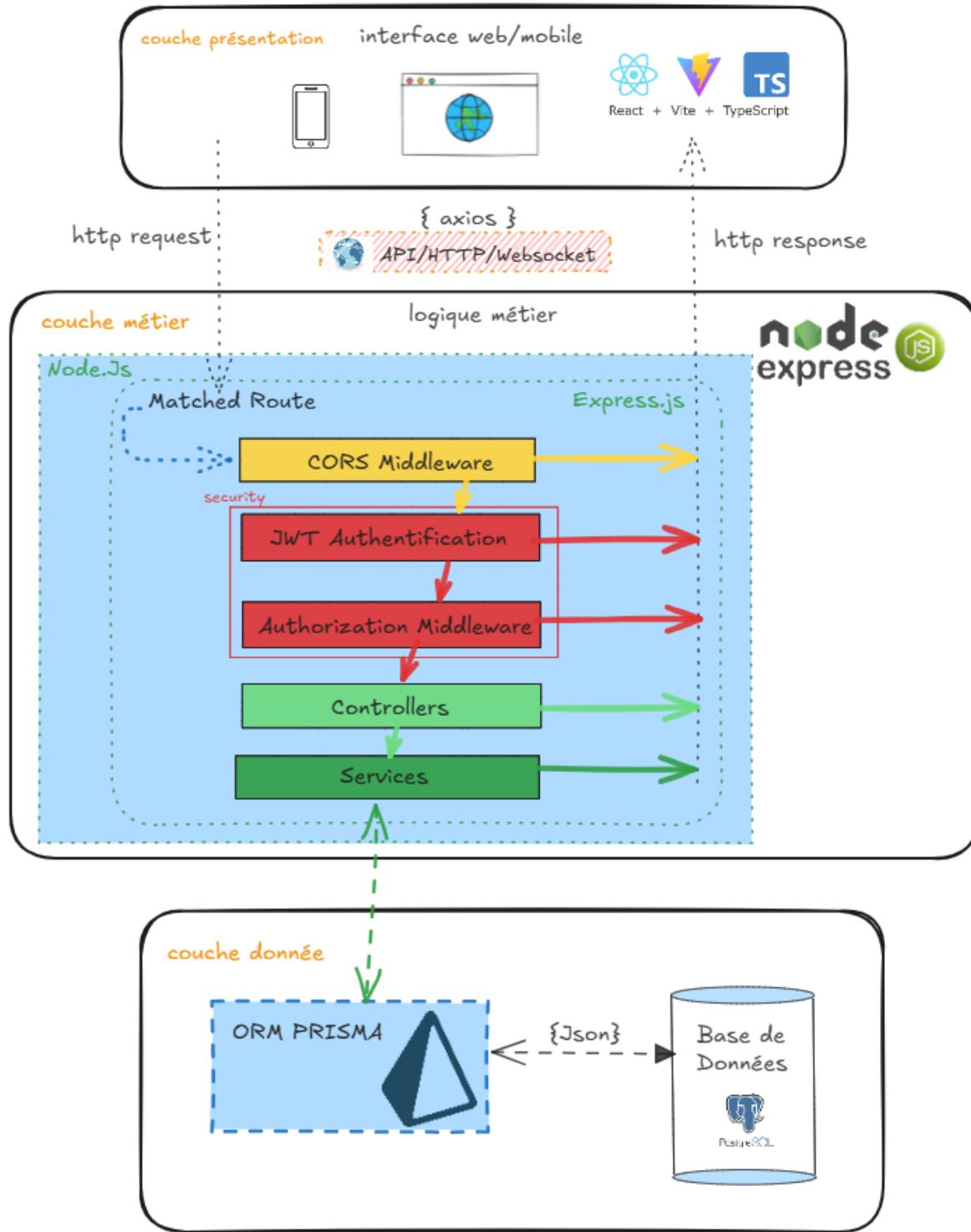


FIGURE 3.3 – Architecture technique 3-tiers de la solution (Frontend React – Backend Node.js/Express – Données PostgreSQL)

La figure 3.3 illustre l'architecture technique globale de la solution, basée sur une approche 3-tiers intégrant une communication asynchrone entre les différentes couches du système.

La couche de présentation repose sur une application web développée avec React, Vite et TypeScript. Elle constitue l'interface utilisateur accessible depuis des navigateurs web ou des terminaux mobiles. Cette couche communique avec le backend via des requêtes HTTP et WebSocket, en utilisant la bibliothèque Axios pour les appels API.

La couche métier est implémentée à l'aide de Node.js et du framework Express.js. Elle est responsable

du traitement de la logique applicative et de la gestion des règles métier. Les requêtes entrantes sont d'abord interceptées par des middlewares transverses, notamment le middleware CORS pour la sécurité, le mécanisme d'authentification basé sur JWT ainsi que les middlewares d'autorisation. Une fois validées, les requêtes sont traitées par les contrôleurs Express, lesquels délèguent les traitements aux services métier.

La couche d'accès aux données repose sur l'ORM Prisma, qui assure l'abstraction entre la logique métier et la base de données PostgreSQL. Prisma facilite la gestion des modèles de données, la réalisation des requêtes SQL et l'application des migrations, tout en garantissant la cohérence et l'intégrité des données persistées.

Cette architecture technique favorise une séparation claire des responsabilités, améliore la maintenabilité du système et permet une évolutivité facilitée grâce à l'indépendance des différentes couches.

Conclusion partielle

Ce chapitre a permis de définir la conception globale de la solution proposée pour le Centre de Recherches Agricoles (CRA) de Saint-Louis. À travers la présentation de l'architecture logique, des choix technologiques et de l'architecture technique, une vision claire et structurée du système a été établie.

L'architecture retenue, fondée sur un modèle 3-tiers et inspirée du patron de conception MVC, garantit une séparation nette des responsabilités entre les différentes couches du système. Les choix technologiques opérés, tant au niveau du frontend que du backend, répondent aux exigences de performance, de sécurité, de maintenabilité et d'évolutivité identifiées lors de l'analyse des besoins.

Sur la base de cette conception, le chapitre suivant est consacré à la mise en œuvre concrète de la solution. Il détaillera l'environnement de développement adopté, les étapes de réalisation de la plateforme ainsi que l'analyse des résultats obtenus, incluant une étude des performances associées à la solution développée.

Chapitre 4

Mise en oeuvre et résultats

4.1 Environnement de développement

La mise en œuvre de la plateforme du Centre de Recherches Agricoles (CRA) de Saint-Louis a été réalisée dans un environnement de développement moderne, collaboratif et orienté qualité. Les outils sélectionnés visent à faciliter le développement, la collaboration entre les membres de l'équipe, la reproductibilité des environnements et la documentation des services exposés.

Visual Studio Code

L'environnement de développement intégré (IDE) **Visual Studio Code (VS Code)** a été utilisé pour l'ensemble du développement frontend et backend. Cet outil offre un excellent support pour les technologies employées dans le projet, notamment TypeScript, React, Node.js et Prisma.

VS Code fournit des fonctionnalités avancées telles que la complétion automatique, le débogage intégré, la gestion des extensions et la navigation efficace dans le code. Ces fonctionnalités contribuent à améliorer la productivité, la lisibilité du code et la détection précoce des erreurs.

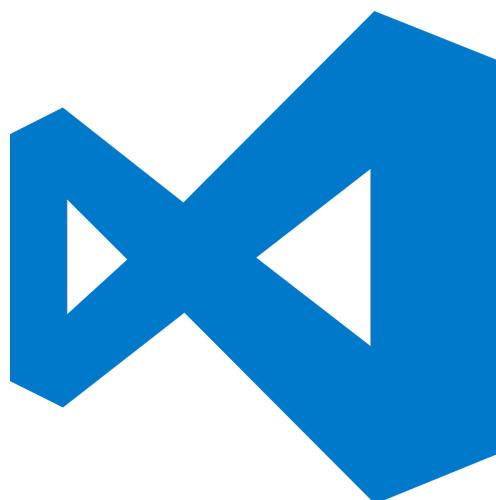


FIGURE 4.1 – Environnement de développement Visual Studio Code utilisé pour le projet

Git et GitHub

Le système de gestion de versions **Git** a été utilisé pour suivre l'évolution du code source et gérer les différentes versions du projet. Il permet de conserver un historique complet des modifications, de faciliter le travail collaboratif et de sécuriser le code contre les pertes accidentnelles.

La plateforme **GitHub** a servi de dépôt centralisé pour l'hébergement du code source. Elle a également permis la gestion des branches, le suivi des issues et la revue de code, favorisant ainsi une collaboration structurée et transparente.



FIGURE 4.2 – Dépôt GitHub du projet et historique des versions

Docker

La technologie **Docker** a été utilisée pour la conteneurisation des différents composants de la plateforme, notamment le backend, la base de données PostgreSQL et les services associés. Docker permet de garantir la cohérence des environnements de développement, de test et de déploiement.

Grâce à Docker, l'application peut être exécutée de manière identique sur différentes machines, réduisant ainsi les problèmes liés aux dépendances et aux configurations spécifiques. Cette approche facilite également le déploiement futur et la scalabilité du système.



FIGURE 4.3 – Conteneurisation de l'application à l'aide de Docker

Swagger (OpenAPI)

La documentation des API REST a été réalisée à l'aide de **Swagger**, basé sur la spécification **OpenAPI**. Swagger permet de décrire de manière formelle les endpoints exposés par le backend, les paramètres attendus, les réponses retournées et les mécanismes de sécurité.

L'interface Swagger offre également la possibilité de tester les API directement depuis un navigateur, facilitant ainsi la validation fonctionnelle, le débogage et la communication entre le frontend et backend.



FIGURE 4.4 – Documentation interactive des API REST via Swagger

4.2 Présentation et analyse des résultats

[Contenu...]

4.3 Etude de performances et de coûts

Conclusion et Perspectives

[Votre conclusion ici]