الجمهورية الجزائرية الديمقراطية الشعبية
République Algérienne démocratique et populaire

وزارة التعليم العالي والبحث العلمي
Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة
Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا
Faculté de Technologie

قسم الإلكترونيك
Département d'Électronique

MEMOIRE DE MASTER

Filière : Télécommunication
Spécialité : Réseaux Et Télécommunications
Numéro du pfe : RT20

Présenté par :

- **SEBBAH OUSSAMA**
- **BOUHADJA IMAD EDDINE**

Conception et développement d'un système intégré de gestion des laboratoires de recherche

Proposé par :

Promoteur : Mr.BENSEBTI MESSAOUD.

Année Universitaire :2024-2025

Remerciement

Nous tenons à exprimer notre profonde gratitude à toutes les personnes qui, de près ou de loin, ont contribué à la réalisation de ce mémoire.

Nos premiers remerciements vont à Allah, qui nous a accordé la force et le courage nécessaires pour mener à bien ce travail modeste.

Nous adressons une mention toute particulière à M. BENSEBTI Messaoud et M. MEHDI Marouane, nos encadrants, pour leurs conseils éclairés, leur disponibilité et leur accompagnement précieux tout au long de ce parcours. Nous pensons également à nos familles, dont le soutien indéfectible et les encouragements constants ont été une source inestimable de motivation durant toute cette année universitaire. Nos remerciements s'adressent aussi aux membres du jury, qui ont accepté de consacrer de leur temps et de partager leur expertise pour évaluer notre travail, marquant ainsi l'une des étapes les plus importantes de notre parcours académique. Merci du fond du Coeur.

Dédicaces

Je dédie ce modeste travail :

- À mes chers parents, pour leur amour inconditionnel, leurs sacrifices et leur soutien constant. Vous êtes la source de ma détermination et de ma persévérance.
- À mes frères et sœurs, pour leur présence, leur complicité et leurs encouragements.
- À toute ma famille, qui n'a cessé de croire en moi et de m'encourager.
- À mes amis, qui m'ont accompagné pendant ce parcours et ont partagé avec moi les moments de doute comme de réussite.
- À tous mes enseignants, qui m'ont transmis leur savoir avec passion et dévouement.
- À tous ceux qui croient en la force de la persévérance et du travail bien fait.

Résumé :

Ce mémoire présente la conception et le développement d'un système intégré de gestion des laboratoires de recherche pour l'Université de Blida 1. Face aux défis de gestion administrative, de suivi des activités scientifiques et de valorisation des résultats de recherche, l'institution nécessitait une solution adaptée à son contexte spécifique.

Après une analyse approfondie des besoins et une étude comparative des solutions existantes, le choix s'est porté sur le développement d'une application sur mesure. Le système proposé repose sur une architecture MVC (Modèle-Vue-Contrôleur) implémentée avec PHP 8.0, MySQL et Bootstrap 5, offrant une interface ergonomique et des fonctionnalités complètes pour la gestion des laboratoires, équipes, membres, publications et événements scientifiques.

Une attention particulière a été accordée à la sécurité, avec l'implémentation d'un modèle RBAC (Role-Based Access Control) à cinq niveaux hiérarchiques et des vérifications de permissions à trois niveaux (contrôleur, interface et API). L'interface utilisateur a été conçue selon des principes d'ergonomie fondamentaux pour garantir une expérience utilisateur optimale.

Le déploiement progressif du système et la formation des utilisateurs ont permis une adoption réussie de la solution, qui répond désormais efficacement aux besoins de centralisation des données, d'automatisations des processus administratifs et de valorisation des productions scientifiques de l'université.

Mots-clés : système d'information, gestion de laboratoires, architecture MVC, PHP, MySQL, Bootstrap, RBAC, ergonomie

---

Abstract:
This thesis presents the design and development of an integrated laboratory management system for the University of Blida 1. Facing challenges in administrative management, monitoring of scientific activities, and valorization of research results, the institution required a solution adapted to its specific context.

Following an in-depth analysis of needs and a comparative study of existing solutions, the decision was made to develop a custom application. The proposed system is based on an MVC (Model-View-Controller) architecture implemented with PHP 8.0,

MySQL, and Bootstrap 5, offering an ergonomic interface and comprehensive functionalities for managing laboratories, teams, members, publications, and scientific events.

Particular attention was paid to security, with the implementation of a five-level hierarchical RBAC (Role-Based Access Control) model and permission checks at three levels (controller, interface, and API). The user interface was designed according to fundamental ergonomic principles to ensure an optimal user experience.

The progressive deployment of the system and user training have enabled successful adoption of the solution, which now effectively meets the needs for data centralization, automation of administrative processes, and valorization of the university's scientific outputs.

Keywords: information system, laboratory management, MVC architecture, PHP, MySQL, Bootstrap, RBAC, ergonomics

---

:الملخص

يعرض هذا البحث تصميم وتطوير نظام متكامل لإدارة مخابر البحث بجامعة البليدة 1. نظراً للتحديات المتعلقة بالإدارة الإدارية، وتتبع الأنشطة العلمية، وتثمين نتائج البحث، كانت المؤسسة في حاجة إلى حلّ يتكيف مع .خصوصياتها

بعد تحليل معمّق للاحتياجات ودراسة مقارنة للحلول المتوفرة، تم اختيار تطوير تطبيق مخصص. يعتمد PHP 8.0 (النموذج - الواجهة - المتحكم) مطبّقة باستخدام MVC النظام المقترح على بنية معمارية ، ما يوفر واجهة سهلة الاستخدام ووظائف شاملة لإدارة المخابر، الفرق، Bootstrap 5وMySQLو .الأعضاء، المنشورات، والفعاليات العلمية

(RBAC) تم إيلاء اهتمام خاص بجانب الأمان من خلال تطبيق نموذج تحكم في الوصول قائم على الأدوار بخمسة مستويات هرمية، مع عمليات تحقق من الصلاحيات على ثلاث مستويات (المتحكم، الواجهة، وواجهة .برمجة التطبيقات). كما تم تصميم الواجهة وفقاً لمبادئ أساسية في علم الواجهة لضمان تجربة استخدام مثالية

سمح النشر التدريجي للنظام وتكوين المستخدمين باعتماد ناجح للحل، الذي أصبح يلبي بفعالية احتياجات .توحيد البيانات، وأتمتة العمليات الإدارية، وتثمين الإنتاجات العلمية للجامعة

الكلمات المفتاحية :نظام معلومات، إدارة المخابر، بنيةMVC ، PHP، MySQL، Bootstrap، التحكم في الوصول حسب الدور(RBAC) ، سهولة الاستخدام.

Table des matières

Table des matières

Liste des figures

Liste des tableaux

Liste des abréviations

API : Application Programming Interface (Interface de Programmation d'Application)
AJAX : Asynchronous JavaScript and XML
CRUD : Create, Read, Update, Delete (Créer, Lire, Mettre à jour, Supprimer)
CSRF : Cross-Site Request Forgery (Falsification de requête inter-sites)
CSS : Cascading Style Sheets (Feuilles de style en cascade)
DTO : Data Transfer Object (Objet de transfert de données)
HTML : HyperText Markup Language (Langage de balisage hypertexte)
Http : HyperText Transfer Protocol (Protocole de transfert hypertexte)
IDE : Integrated Development Environment (Environnement de développement intégré)
JSON : JavaScript Object Notation
LIMS : Laboratory Information Management System (Système de gestion des informations de laboratoire)
MVC : Model-View-Controller (Modèle-Vue-Contrôleur)
MySQL: My Structured Query Language
ORM : Object-Relational Mapping (Mappage objet-relationnel)
PDO : PHP Data Objects
PHP : PHP : Hypertext Preprocessor
PRFU : Projets de Recherche-Formation Universitaire
RBAC : Role-Based Access Control (Contrôle d'accès basé sur les rôles)
REST : Representational State Transfer
SQL : Structured Query Language (Langage de requête structurée)
SVG : Scalable Vector Graphics (Graphiques vectoriels évolutifs)
UML : Unified Modeling Language (Langage de modélisation unifié)
UI : User Interface (Interface utilisateur)
UX : User Experience (Expérience utilisateur)
XSSCros : s-Site Scripting

INTRODUCTION GÉNÉRALE

Contexte et justification du projet

Dans un environnement universitaire en constante évolution, la recherche scientifique constitue un pilier fondamental du développement des institutions académiques et de leur rayonnement national et international. Les laboratoires de recherche, en tant que centres névralgiques de production scientifique, font face à des défis croissants en matière de gestion administrative, de suivi des activités et de valorisation des résultats. Cette réalité est particulièrement prégnante dans le contexte des universités algériennes qui aspirent à renforcer leur position dans le paysage académique mondial.

L'Université de Blida 1, avec ses 42 laboratoires de recherche répartis à travers 8 facultés et 2 instituts, illustre parfaitement cette dynamique. Accueillant plus de 35 000 étudiants et regroupant un corps enseignant-chercheur conséquent, cette institution fait face à des problématiques de gestion complexes. L'organisation actuelle des laboratoires se caractérise par une hétérogénéité des outils, une fragmentation des données entre différents supports, et une prépondérance des processus manuels, limitant considérablement l'efficacité administrative et la visibilité des productions scientifiques.

La fragmentation des systèmes d'information au sein des laboratoires génère plusieurs difficultés majeures. Chaque laboratoire utilise ses propres méthodes et outils, allant de simples tableurs Excel à des bases Access isolées. Cette diversité non coordonnée rend pratiquement impossible toute consolidation des données au niveau institutionnel. Par ailleurs, les processus administratifs essentiellement manuels engendrent une duplication des saisies, des risques d'erreurs élevés et mobilisent un temps considérable qui pourrait être consacré aux activités de recherche proprement dites.

Le suivi des activités scientifiques souffre également de cette dispersion. En l'absence d'un référentiel unique et cohérent, les publications, communications et autres productions scientifiques sont insuffisamment valorisées et leur impact difficile à évaluer objectivement. Les collaborations potentielles entre équipes et laboratoires sont entravées par le manque de visibilité sur les compétences disponibles au sein même de l'institution. En outre, la génération des rapports périodiques exigés par le Ministère de l'Enseignement Supérieur nécessite un travail fastidieux de compilation manuelle, souvent source d'incohérences.

Face à ces constats, le besoin d'un système d'information intégré, capable de centraliser les données, d'automatiser les processus et de faciliter le pilotage stratégique des laboratoires, s'est imposé comme une priorité pour l'Université de Blida 1.

L'analyse des solutions logicielles disponibles sur le marché a révélé que les produits existants, qu'ils soient commerciaux ou open source, ne répondaient que partiellement aux besoins spécifiques de l'institution. Les solutions commerciales, souvent onéreuses et peu adaptables, ne prenaient pas suffisamment en compte les particularités du système universitaire algérien. Les alternatives open source présentaient des limitations fonctionnelles importantes ou nécessitaient des compétences techniques spécifiques non disponibles dans l'équipe informatique de l'université.

C'est dans ce contexte que s'inscrit le projet de conception et développement d'un système intégré de gestion des laboratoires pour l'Université de Blida 1. Cette initiative vise à répondre aux besoins spécifiques de l'institution tout en s'inscrivant dans une démarche plus globale d'amélioration de la gouvernance universitaire.

Objectifs et enjeux

Le développement d'un système intégré de gestion des laboratoires pour l'Université de Blida 1 répond à plusieurs objectifs stratégiques, s'inscrivant dans une vision globale d'amélioration de l'efficacité administrative et de valorisation de la recherche scientifique.

Objectifs principaux

- **Centraliser les données** relatives aux laboratoires dans un référentiel unique et cohérent, facilitant ainsi leur accès, leur mise à jour et leur exploitation. Cette centralisation concerne l'ensemble des informations relatives aux structures (laboratoires, équipes), aux acteurs (chercheurs, personnel administratif) et aux productions (publications, événements, projets).
- **Automatiser les processus administratifs** pour réduire la charge de travail manuel, minimiser les risques d'erreurs et accélérer le traitement des opérations courantes. L'automatisation vise notamment les tâches répétitives comme la génération de rapports, le suivi des affiliations et la gestion des événements scientifiques.
- **Améliorer la visibilité et la valorisation des productions scientifiques** à travers un catalogage structuré des publications, projets et événements. Cette

amélioration passe par l'adoption de standards bibliographiques internationaux et la création d'interfaces de consultation ergonomiques.
- **Faciliter la génération de rapports et d'indicateurs de performance** conformes aux exigences ministérielles et aux standards internationaux. Le système doit permettre l'extraction rapide de données synthétiques et leur présentation sous des formats adaptés aux différents besoins de reporting.
- **Renforcer la collaboration** entre chercheurs, équipes et laboratoires grâce à des outils adaptés de partage d'information et de communication. Cette dimension collaborative vise à créer des synergies scientifiques et à optimiser l'utilisation des ressources disponibles.

Enjeux stratégiques

Ces objectifs s'inscrivent dans plusieurs enjeux majeurs pour l'institution :

- **Enjeux organisationnels** : L'optimisation des processus de gestion permet une allocation plus efficiente des ressources humaines et matérielles, libérant du temps pour les activités de recherche proprement dites.
- **Enjeux scientifiques** : Une meilleure visibilité des travaux et compétences favorise les collaborations interdisciplinaires et internationales, contribuant ainsi à l'enrichissement des activités de recherche.
- **Enjeux décisionnels** : La disponibilité d'indicateurs pertinents facilite le pilotage stratégique des laboratoires et l'évaluation objective de leurs performances.
- **Enjeux institutionnels** : Le renforcement de l'image de l'université dans le paysage académique et l'amélioration de son attractivité pour les partenariats constituent des retombées significatives du projet.
- **Enjeux technologiques** : Le développement interne d'une solution sur mesure permet de valoriser et de développer les compétences techniques au sein de l'université.

Face à ces enjeux, le développement d'une solution sur mesure s'est imposé comme la réponse la plus adaptée, privilégiant l'adéquation parfaite aux besoins et l'indépendance à long terme, malgré un investissement initial plus important en termes de ressources humaines et de temps de développement. [1] [3]

Méthodologie générale de développement

La conception et la mise en œuvre du système ont suivi une méthodologie rigoureuse, intégrant les bonnes pratiques de l'ingénierie logicielle et une approche centrée

utilisateur. Le projet s'est articulé autour de trois phases principales : analyse et conception, développement et implémentation, déploiement et évaluation.

Phase d'analyse et de conception

Une analyse approfondie des besoins a été menée via des entretiens, des observations, des questionnaires et des ateliers participatifs impliquant chercheurs, administratifs et informaticiens. Une étude comparative des solutions existantes a confirmé l'intérêt d'un développement personnalisé. L'architecture du système a été définie selon le modèle MVC (Modèle-Vue-Contrôleur), assurant une séparation claire entre les différentes couches logicielles. Un modèle de données robuste a été conçu pour représenter les entités principales (laboratoires, équipes, membres, publications, etc.) et leurs relations. Les interfaces utilisateur ont été pensées selon des principes ergonomiques, tandis qu'un modèle de sécurité basé sur RBAC (Role-Based Access Control) a été mis en place avec cinq niveaux hiérarchiques et des vérifications multi-niveaux.

Phase de développement

Les choix technologiques incluent PHP 8.0 pour le backend, MySQL pour la base de données et Bootstrap 5 pour l'interface frontend. L'environnement de développement a été standardisé, intégrant des outils de contrôle de version et de qualité de code. Le développement a adopté une approche incrémentale organisée en sprints de deux semaines, permettant de livrer progressivement des fonctionnalités opérationnelles. Chaque composant a été développé en respectant strictement l'architecture MVC préalablement définie.

Phase de test et déploiement

Une stratégie complète de tests a été appliquée, couvrant les tests unitaires, d'intégration, fonctionnels, de sécurité et de performance. Le déploiement a été progressif, démarrant par un pilote sur trois laboratoires représentatifs avant une extension progressive à l'ensemble de l'université. Un processus structuré de migration des données issues de sources hétérogènes a été mis en œuvre. Par ailleurs, un programme de formation ciblé a été déployé, accompagné d'une documentation technique et utilisateur complète. Enfin, un dispositif d'évaluation continue a été instauré, reposant sur les retours utilisateurs et l'analyse des logs d'utilisation.[1] [2]

Structure du mémoire

Le présent mémoire s'articule autour de trois chapitres principaux, reflétant la progression logique du projet :

Chapitre 1 : Analyse et conception

Ce chapitre présente le contexte détaillé du projet, l'organisation actuelle des laboratoires de l'Université de Blida 1 et les problématiques qui en découlent. Il expose une analyse comparative des solutions existantes sur le marché, justifiant le choix d'un développement sur mesure. La dernière section détaille la conception du système selon quatre axes principaux : l'architecture MVC adoptée, la modélisation des données, la conception de l'interface utilisateur, et le modèle de sécurité RBAC.

Chapitre 2 : Développement et implémentation

Ce chapitre détaille les aspects techniques du développement, en commençant par les choix technologiques effectués et l'environnement de développement utilisé. Il décrit ensuite l'implémentation concrète de l'architecture MVC, avec la structure du code et l'organisation des composants. Le chapitre aborde les principales fonctionnalités développées et se termine par des aspects techniques particuliers comme la gestion des fichiers, la validation des données et l'optimisation des performances.

Chapitre 3 : Déploiement et évaluation

Ce dernier chapitre expose la stratégie de test mise en œuvre et les résultats obtenus. Il décrit le processus de déploiement, incluant la configuration du serveur, la migration des données existantes, la formation des utilisateurs et l'élaboration de la documentation. Le chapitre se termine par un bilan du projet, évaluant l'atteinte des objectifs initiaux et analysant les retours des utilisateurs.

La conclusion générale du mémoire synthétise les apports du projet pour l'Université de Blida 1, reconnaît certaines limites actuelles du système et ouvre des perspectives d'évolution.

Des annexes complètent le document, fournissant des informations détaillées sur les aspects techniques du système, des extraits de code significatifs et des exemples de documents générés par le système.

Cette structure permet d'appréhender l'ensemble du cycle de vie du projet, depuis son

initialisation jusqu'à son évaluation, en mettant en évidence les choix méthodologiques et techniques qui ont guidé sa réalisation, ainsi que les résultats concrets obtenus pour l'Université de Blida 1.

CHAPITRE 1 : ANALYSE ET CONCEPTION

1.1. Contexte et problématique

1.1.1. Présentation de l'Université de Blida 1

L'Université Saad Dahlab de Blida 1 constitue l'une des principales institutions universitaires d'Algérie. Fondée en 1981, elle a connu une évolution significative pour atteindre aujourd'hui une structure comprenant 8 facultés et 2 instituts. Elle accueille plus de 35 000 étudiants et compte environ 1 700 enseignants-chercheurs.

La recherche scientifique y est organisée autour de 42 laboratoires officiellement reconnus par le Ministère de l'Enseignement Supérieur et de la Recherche Scientifique. Ces laboratoires couvrent des domaines variés allant des sciences exactes et technologiques aux sciences humaines et sociales. Ils sont répartis entre les différentes facultés selon la distribution suivante :

- Faculté de Technologie : 12 laboratoires
- Faculté des Sciences : 10 laboratoires
- Faculté de Médecine : 7 laboratoires
- Faculté des Sciences de la Vie et de la Nature : 5 laboratoires
- Autres facultés et instituts : 8 laboratoires

Chaque laboratoire est structuré en plusieurs des équipes de recherche, regroupant des chercheurs permanents (professeurs, maîtres de conférences, maîtres assistants) et non-permanents (doctorants). Au total, l'université compte environ 210 équipes de recherche et plus de 1 200 chercheurs actifs.

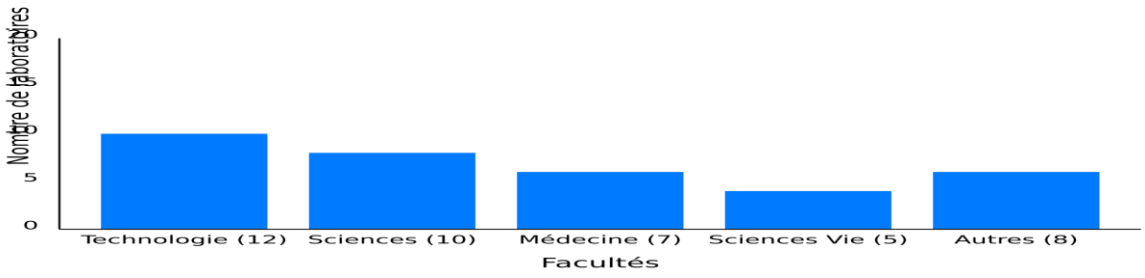

Figure 1: Répartition des laboratoires par faculté

1.1.2. Organisation actuelle des laboratoires et limites

Structure organisationnelle

Les laboratoires sont dirigés par des directeurs nommés pour des mandats de 4 ans, assistés par un conseil de laboratoire comprenant les chefs d'équipes et des représentants élus des chercheurs. Chaque laboratoire dispose d'un budget annuel alloué par le Ministère, basé principalement sur ses performances scientifiques et son plan d'action.

Les activités des laboratoires incluent :

- La conduite de projets de recherche (PRFU, projets internationaux)
- La publication d'articles scientifiques et d'ouvrages
- L'organisation de manifestations scientifiques (séminaires, conférences)
- L'encadrement des travaux de recherche des doctorants
- La formation continue et le perfectionnement des chercheurs
- Les activités d'expertise et de valorisation

Gestion administrative actuelle et ses limites

Un problème central réside dans la **fragmentation des données**. Il n'existe pas de système centralisé pour la gestion des informations, ce qui conduit à l'utilisation d'outils hétérogènes tels que des documents Word, des feuilles Excel et des bases Access isolées. Cette diversité non coordonnée génère une redondance des saisies et augmente considérablement les risques d'incohérence dans les données.

De plus, les **processus manuels prédominants** constituent un frein majeur. La gestion administrative repose encore largement sur des supports papier pour les formulaires, rapports et demandes. L'archivage physique des documents rend leur accès et leur partage difficiles, et ces tâches administratives répétitives consomment un temps excessif qui pourrait être allée aux activités de recherche.

Le **manque de visibilité sur la production scientifique** est une autre lacune significative. L'absence d'un répertoire unifié des publications et communications rend difficile l'évaluation de la production scientifique à l'échelle de l'université, et la valorisation des résultats de recherche s'en trouve insuffisante.

Ces problèmes se répercutent sur les **difficultés de reporting et de pilotage**. La production des rapports périodiques exigés par la tutelle est laborieuse, et il est impossible d'extraire rapidement des indicateurs de performance fiables. Ce manque

d'outils d'aide à la décision nuit à l'orientation stratégique des recherches.

Enfin, des **obstacles à la collaboration** sont clairement identifiés. La méconnaissance des compétences disponibles au sein même de l'université, le partage limité des ressources entre équipes et laboratoires, et une communication insuffisante, tant intra qu'inter-laboratoires, freinent les synergies potentielles.

Ces limitations ont été mises en évidence par des entretiens approfondis avec les directeurs de laboratoires, les chefs d'équipes et les chercheurs, ainsi que par l'observation directe des processus en place. Un questionnaire distribué à 75 acteurs clés a confirmé ces problématiques, avec plus de 80% des répondants exprimant une insatisfaction vis-à-vis des outils actuels de gestion.

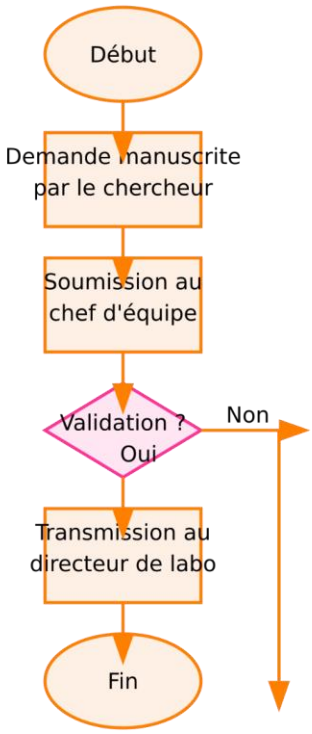

Figure 2 : Processus administratif actuel (avant le système)

Tableau 1 : Synthèse des limites du système actuel

Domaine de Limitation	Description de la Limite	Impacts Principaux
Gestion des Données	Données fragmentées et hétérogènes, souvent stockées localement (fichiers Excel, documents Word, bases de données isolées).	Perte d'informations, redondances, difficultés d'accès et de consolidation, manque de vision globale.
Processus Administratifs et Scientifiques	Processus largement manuels, basés sur des documents papier ou des échanges d'emails non structurés (ex: adhésions, soumission de publications).	Lenteur, lourdeur administrative, risque élevé d'erreurs, manque de traçabilité, gaspillage de temps pour les chercheurs et l'administration.
Visibilité de la Production Scientifique	Absence d'un référentiel centralisé et à jour des publications, projets, et compétences des chercheurs.	Difficulté à valoriser la recherche, à identifier les expertises, à mesurer l'impact scientifique global de l'université.
Reporting et Prise de Décision	Production de rapports et statistiques complexe et chronophage, nécessitant la collecte manuelle de données.	Rapports souvent incomplets ou obsolètes, prise de décision basée sur des informations partielles, difficultés à répondre aux audits et évaluations.
Collaboration et Communication	Outils de collaboration limités ou inexistants entre laboratoires, équipes et chercheurs.	Faible synergie, duplication des efforts, opportunités de collaboration manquées, communication interne et externe inefficace.
Suivi des Ressources et des Membres	Manque de suivi centralisé des membres (chercheurs, doctorants, personnel technique), de leurs affiliations et de leurs activités.	Difficulté à gérer les carrières, à affecter les ressources, à maintenir un annuaire à jour.

Sécurité et Pérennité des Données	Risques liés à la sécurité des données stockées localement (perte, accès non autorisé), absence de politique de sauvegarde uniforme.	Vulnérabilité des informations sensibles, non-conformité potentielle avec les réglementations sur la protection des données.
Hétérogénéité des Outils	Utilisation d'outils disparates et non standardisés au sein des différents laboratoires et facultés.	Difficultés d'interopérabilité, coûts de maintenance accrus, courbe d'apprentissage répétée pour les utilisateurs.

1.1.3. Besoins identifiés

L'analyse des problématiques a permis de dégager un ensemble de besoins fonctionnels et non fonctionnels pour le futur système d'information.

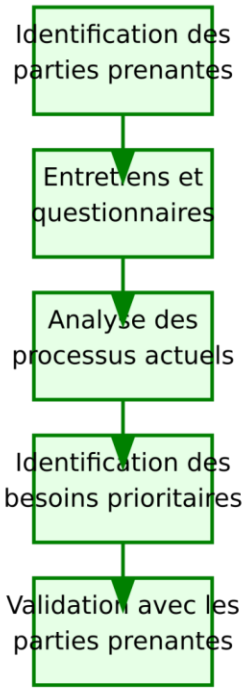

Figure 3 : Méthodologie d'analyse des besoins

Besoins fonctionnels

Concernant la **gestion structurelle des laboratoires**, le système devra permettre l'enregistrement et la mise à jour des informations de base des laboratoires, ainsi que la gestion des équipes et de leur composition. Il sera également essentiel d'assurer le suivi des changements organisationnels grâce à un historique détaillé.

Pour la **gestion des membres et chercheurs**, le système devra offrir la possibilité de créer et de mettre à jour les profils des chercheurs, de suivre leurs statuts et affiliations, et de gérer leurs compétences et domaines d'expertise.

La **gestion de la production scientifique** est un axe majeur, nécessitant un catalogue structuré des publications (articles, ouvrages, brevets), l'organisation des événements scientifiques, et le suivi des projets de recherche et de leur avancement.

En matière de **reporting et tableaux de bord**, le système devra permettre la génération automatisée de rapports standardisés, l'affichage d'indicateurs de performance scientifique pertinents, et la visualisation graphique des activités et des résultats.

Enfin, pour la **collaboration et la communication**, le système devra faciliter le partage de ressources documentaires, la diffusion ciblée d'informations, et la mise en relation des chercheurs par domaine d'expertise, afin de favoriser les synergies au sein de l'université.

Tableau 2 : Besoins fonctionnels prioritaires

ID	Besoin Fonctionnel	Description Détaillée	Priorité	Acteurs Concernés
BF01	**Gestion de la Structure Organisationnelle**	Créer, modifier, visualiser et archiver les entités : laboratoires, équipes de recherche, et leurs liens hiérarchiques ou fonctionnels.	Haute	Administrateurs, Directeurs de laboratoire
BF02	**Gestion des Membres et Utilisateurs**	Gérer les profils des utilisateurs (chercheurs, doctorants, personnel technique, administratifs), leurs rôles, permissions (RBAC), et affiliations.	Haute	Administrateurs, Directeurs de laboratoire, Chefs d'équipe
BF03	**Gestion de la Production Scientifique**	Enregistrer, valider, suivre et valoriser les publications (articles, ouvrages, communications, thèses), brevets et autres productions scientifiques.	Haute	Chercheurs, Chefs d'équipe, Directeurs de laboratoire, Administrateurs

BF04	**Gestion des Projets de Recherche**	Suivre les informations clés des projets de recherche (PRFU, etc.) : participants, financements, livrables, état d'avancement.	Moyenne	Chercheurs, Chefs d'équipe, Directeurs de laboratoire
BF05	**Tableaux de Bord et Reporting**	Générer des statistiques, rapports personnalisés et tableaux de bord sur les activités des laboratoires, la production scientifique, les membres, etc.	Haute	Administrateurs, Directeurs de laboratoire, Organes de tutelle
BF06	**Annuaire et Moteur de Recherche**	Permettre la recherche multicritère d'informations sur les laboratoires, équipes, membres, publications, projets, et compétences.	Haute	Tous les utilisateurs
BF07	**Gestion des Événements Scientifiques**	Annoncer et gérer les événements scientifiques (séminaires, conférences, soutenances) organisés par les laboratoires.	Moyenne	Administrateurs, Organisateurs d'événements, Chercheurs

BF08	**Outils de Collaboration et Communication**	Fournir des espaces de travail partagés, messagerie interne, forums de discussion pour faciliter la collaboration au sein des équipes et laboratoires.	Moyenne	Chercheurs, Chefs d'équipe, Directeurs de laboratoire
BF09	**Gestion des Compétences et Expertises**	Répertorier les domaines de compétence et d'expertise des chercheurs pour faciliter l'identification de collaborateurs ou d'experts.	Moyenne	Chercheurs, Directeurs de laboratoire, Administrateurs
BF10	**Authentification et Sécurité des Accès**	Assurer une authentification sécurisée et une gestion fine des droits d'accès en fonction des rôles définis (RBAC).	Haute	Tous les utilisateurs
BF11	**Export de Données**	Permettre l'export de données dans des formats standards (CSV, Excel, PDF) pour des analyses externes ou des rapports spécifiques.	Moyenne	Administrateurs, Directeurs de laboratoire
BF12	**Notifications et**	Informer les	Moyenne	Tous les

	Alertes	utilisateurs des actions requises, des mises à jour importantes ou des événements pertinents via des notifications système ou email.		utilisateurs

Besoins non fonctionnels

En ce qui concerne la **sécurité et la confidentialité**, le système exigera un accès sécurisé avec une authentification forte et différents niveaux de permissions basés sur les rôles, tout en assurant la protection des données sensibles.

L'**ergonomie et l'accessibilité** sont primordiales, avec une interface intuitive adaptée aux utilisateurs non techniques, une accessibilité depuis divers types d'appareils, et un temps de formation minimal requis pour les utilisateurs.

La **performance et la disponibilité** sont des critères essentiels : le système devra garantir des temps de réponse rapides même sous forte charge, une disponibilité continue (24/7), et la capacité de gérer l'ensemble des utilisateurs simultanément.

Pour la **maintenabilité et l'évolutivité**, une architecture modulaire facilitera les évolutions futures, et une documentation technique complète sera indispensable, assurant ainsi une indépendance vis-à-vis des prestataires extérieurs.

Enfin, l'**interopérabilité** est un besoin crucial, impliquant la capacité du système à échanger des données avec d'autres systèmes universitaires, sa conformité aux standards internationaux de métadonnées scientifiques, et la possibilité d'export/import dans des formats standards.

Ces besoins ont servi de base à l'élaboration du cahier des charges du système et à l'évaluation des solutions potentielles.

Tableau 3 : Besoins non fonctionnels

ID	Catégorie	Besoin Non Fonctionnel	Description / Critères d'Acceptation Clés	Priorité
BNF01	**Performance**	Temps de réponse du système	- Les pages critiques (tableaux de bord, listes de publications) doivent se charger en moins de 3 secondes.- Les opérations courantes (sauvegarde de formulaire) doivent s'exécuter en moins de 2 secondes.	Haute
BNF02	**Sécurité**	Protection contre les vulnérabilités web courantes	- Protection contre XSS, CSRF, Injection SQL.- Mots de passe hachés.- Sessions sécurisées (HTTPS).	Haute
BNF03	**Sécurité**	Confidentialité et intégrité des données	- Accès aux données restreint selon le modèle RBAC.- Pistes d'audit pour les modifications de données sensibles.	Haute
BNF04	**Ergonomie (Utilisabilité)**	Interface utilisateur intuitive et conviviale	- Navigation claire et cohérente.- Formulaires faciles à comprendre et à remplir.- Messages d'erreur et de succès explicites.	Haute
BNF05	**Ergonomie (Utilisabilité)**	Accessibilité	- Respect des bonnes pratiques d'accessibilité web (contrastes, navigation clavier basique).	Moyenne
BNF06	**Fiabilité**	Disponibilité du système	- Taux de disponibilité du système attendu à 99.5% (hors maintenance planifiée).	Haute
BNF07	**Fiabilité**	Robustesse et gestion des erreurs	- Gestion appropriée des erreurs et exceptions pour éviter les plantages.- Messages d'erreur clairs pour l'utilisateur.	Haute
BNF08	**Maintenabilité**	Facilité de maintenance et d'évolution du code	- Code source modulaire, commenté et respectant les standards de développement PHP.- Documentation technique à jour.	Haute
BNF09	**Évolutivité (Scalabilité)**	Capacité à gérer l'augmentation du volume de données	- Le système doit pouvoir supporter une augmentation de 20% du volume de données et du nombre d'utilisateurs par an sans dégradation majeure des performances.	Moyenne

ID	Catégorie	Besoin Non Fonctionnel	Description / Critères d'Acceptation Clés	Priorité
		et d'utilisateurs		
BNF10	**Compatibilité**	Compatibilité avec les navigateurs web modernes	- Fonctionnement optimal sur les dernières versions de Chrome, Firefox, Safari, Edge.	Haute
BNF11	**Sauvegarde et Restauration**	Procédures de sauvegarde et de restauration des données	- Sauvegardes automatiques quotidiennes de la base de données.- Procédure de restauration testée et documentée.	Haute

1.2. Analyse des solutions

1.2.1. Étude comparative des solutions existantes

Une analyse approfondie des systèmes disponibles sur le marché a été réalisée, examinant à la fois des solutions commerciales et des alternatives open source.

Méthodologie d'évaluation

L'évaluation des solutions logicielles a été conduite selon une grille multicritère rigoureuse, prenant en compte la couverture fonctionnelle, les aspects techniques (architecture, technologies, performance), les aspects économiques (coûts d'acquisition, maintenance, formation), l'adaptabilité au contexte local (multilinguisme, spécificités universitaires algériennes), ainsi que le support et la pérennité de chaque solution. Chaque proposition a été notée sur une échelle de 1 à 5 pour chaque critère, avec une pondération reflétant les priorités stratégiques de l'université.

Solutions commerciales évaluées

Trois solutions commerciales majeures ont été analysées. **LabManager Pro** s'est distingué par son interface moderne, ses nombreuses fonctionnalités et son reporting avancé, mais ses points faibles résidaient dans son coût élevé, la rigidité de son modèle de données et une adaptation limitée au contexte universitaire algérien, ce qui a

conduit à une évaluation globale de 3.2/5. La **ResearchTrack Suite** offrait un module de suivi des publications très complet et une bonne intégration avec les bases bibliographiques, cependant, l'absence de gestion budgétaire adaptée, sa complexité d'utilisation et le coût des licences ont résulté en une évaluation globale de 2.8/5. Enfin, **UniLab Manager** se caractérisait par sa spécialisation pour les laboratoires universitaires et son multilinguisme, mais présentait des difficultés d'adaptation aux processus spécifiques et des coûts de personnalisation élevés, obtenant une évaluation globale de 3.5/5.

Solutions open source évaluées

Deux solutions open source ont également été considérées. **OpenScientic** bénéficiait de la gratuité et d'une communauté active, avec un focus particulier sur les publications ; toutefois, ses fonctionnalités administratives limitées, son interface peu conviviale et une documentation insuffisante ont mené à une évaluation globale de 2.5/5. Quant à **LabFlow System**, il proposait une architecture modulaire et un coût initial nul, mais nécessitait des compétences techniques élevées et ses fonctionnalités de reporting étaient limitées, ce qui a abouti à une évaluation globale de 2.7/5.

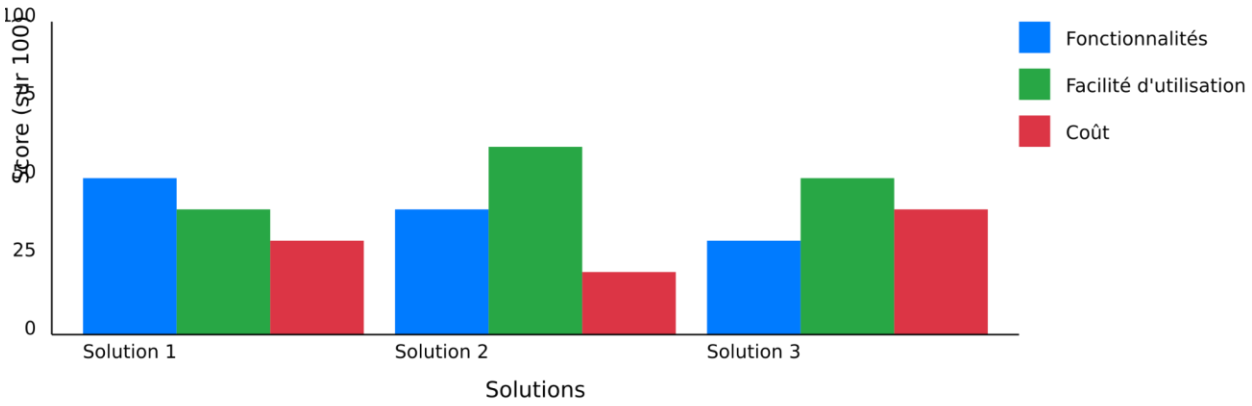

Figure 4 : Comparaison des solutions existantes

Tableau 4 : Comparaison détaillée des solutions commerciales

Critère d'Évaluation	Solution "LabManager Pro" (Fictif)	Solution "SciCore Suite" (Fictif)	Solution "AcademiaSys" (Fictif)
Fonctionnalités Clés	Gestion complète des laboratoires, membres, publications, projets, inventaires, équipements. Module de reporting avancé. Workflow de validation.	Orienté gestion de la recherche : publications, projets, financements, collaborations. Outils d'analyse bibliométrique. Gestion des appels à projets.	Plateforme intégrée pour universités : gestion des étudiants, cours, recherche, finances. Moins spécialisé pour les laboratoires purs.
Couverture des Besoins Fonctionnels Identifiés	Bonne couverture (BF01-BF07, BF10-BF12). Moins axé sur la collaboration interne (BF08) et gestion fine des compétences (BF09).	Très bonne couverture pour la production scientifique et projets (BF03, BF04, BF09). Couverture partielle pour la gestion structurelle (BF01, BF02).	Couverture de base pour les aspects recherche, mais nécessite souvent des modules additionnels coûteux pour une gestion détaillée des laboratoires.

Coût (Licence/Abonnement)	Élevé. Licence par utilisateur ou par laboratoire + frais de maintenance annuels.	Très élevé. Abonnement annuel basé sur la taille de l'institution et les modules choisis.	Variable. Souvent des accords-cadres institutionnels. Coût par module peut être significatif.
Personnalisation / Flexibilité	Personnalisation limitée à la configuration des formulaires et workflows prédéfinis. Développements spécifiques coûteux.	Flexibilité modérée. API disponible pour intégrations mais options de personnalisation de l'interface limitées.	Faible. Système souvent rigide, conçu pour des processus standardisés à grande échelle.
Support Technique	Support réactif inclus dans la maintenance. Base de connaissances en ligne.	Support premium disponible avec SLA. Communauté d'utilisateurs active.	Support institutionnel ou via des partenaires locaux. Temps de réponse variables.
Interface Utilisateur / Ergonomie	Interface moderne mais peut être complexe en raison du grand nombre de fonctionnalités.	Interface soignée, orientée analyse de données. Peut nécessiter une formation pour une prise en main complète.	Interface souvent datée ou hétérogène entre les modules. Ergonomie perfectible.

Intégration avec d'Autres Systèmes	API REST disponible. Connecteurs pour certains systèmes financiers ou RH.	Bonnes capacités d'intégration avec les bases de données bibliographiques (Scopus, WoS) et les ORCID.	Intégration possible mais souvent complexe et coûteuse, nécessitant des développements spécifiques.
Sécurité	Bonnes pratiques de sécurité. Options de conformité (ex: 21 CFR Part 11 pour certains modules).	Sécurité robuste, hébergement cloud certifié (ISO 27001). Gestion fine des accès.	Dépend de l'implémentation institutionnelle. Souvent conforme aux standards généraux de sécurité.
Hébergement	Principalement On-premise, option Cloud disponible.	Principalement Cloud (SaaS).	Généralement On-premise, mais des versions Cloud commencent à apparaître.
Avantages Principaux	Solution éprouvée, riche en fonctionnalités spécifiques aux laboratoires.	Puissants outils d'analyse de la recherche, bonne intégration avec l'écosystème de publication.	Solution potentiellement "tout-en-un" pour une institution, réduisant le nombre de systèmes.

Inconvénients Principaux	Coût élevé, rigidité, peut être surdimensionné pour des besoins plus ciblés.	Très coûteux, moins adapté à la gestion administrative fine des structures de laboratoire.	Manque de spécialisation pour la gestion des laboratoires, complexité, coût des modules spécifiques.

Tableau 5 : Comparaison détaillée des solutions open source

Critère d'Évaluation	**eLabFTW (Existant)**	**"OpenLIMS Suite" (Fictif)**	**"AcademiaConnect OS" (Fictif)**
Fonctionnalités Clés	Cahier de laboratoire électronique (ELN), gestion des ressources (inventaire, équipements), gestion d'équipe, expériences, protocoles.	Gestion des échantillons, workflows, instruments, analyses, rapports. Orienté laboratoires d'analyse/biologie.	Gestion de projets de recherche, publications, collaborations, CV des chercheurs, événements. Plus axé sur la gestion académique de la recherche.
Couverture des Besoins Fonctionnels Identifiés	Bonne couverture pour la gestion des expériences et ressources (partie de BF03, BF04). Moins pour la gestion structurelle globale (BF01, BF02) ou le reporting avancé (BF05).	Forte couverture pour les aspects techniques de laboratoire (partie de BF03). Faible pour la gestion des membres (BF02) et la structure organisationnelle (BF01).	Bonne couverture pour la production scientifique (BF03), projets (BF04), compétences (BF09). Limitée pour la gestion administrative des laboratoires.
Coût (Licence/Support/Hébergement)	Gratuit (licence AGPL). Coûts liés à l'hébergement (auto-hébergé ou via des prestataires). Support	Gratuit (licence MIT/GPL). Coûts d'hébergement et de personnalisation potentiels. Support	Gratuit (licence Apache/GPL). Coûts d'hébergement. Support via forums, documentation.

	communautaire gratuit, support professionnel payant possible.	communautaire principalement.	Contributions de la communauté.
Personnalisation / Flexibilité	Très flexible. Code source ouvert permettant des adaptations poussées. API disponible. Développement de plugins possible.	Modulable. Le code source ouvert permet des modifications, mais peut nécessiter des compétences techniques avancées.	Assez flexible. Permet la création de modules personnalisés. Configuration via interface d'administration.
Support Technique / Communauté	Communauté active (GitHub, forums). Documentation complète. Support professionnel disponible via des entreprises tierces.	Communauté plus restreinte, dépend de la popularité du projet. Documentation variable.	Communauté souvent académique, active mais potentiellement moins réactive que des supports commerciaux.
Interface Utilisateur / Ergonomie	Interface web moderne et réactive. Généralement bien perçue. Prise en main relativement aisée.	Interface souvent technique, peut être moins intuitive pour des non-spécialistes. Ergonomie variable selon les modules.	Interface fonctionnelle, parfois moins soignée visuellement que des solutions commerciales. Peut varier selon les thèmes/plugins.
Intégration avec d'Autres Systèmes	API REST bien documentée. Intégrations possibles avec d'autres outils (ex: LDAP, SAML pour l'authentification).	Capacités d'intégration variables, souvent via des scripts personnalisés ou des API si disponibles.	API disponible. Intégrations avec des systèmes d'authentification universitaires (CAS, Shibboleth) parfois présentes.
Sécurité	Dépend de la qualité de l'installation et de la configuration. Mises à jour régulières	Sécurité à la charge de l'implémenteur. Nécessite une expertise pour	La sécurité dépend de la configuration et des mises à jour. Vulnérabilités

	importantes. Bonnes pratiques de base implémentées.	sécuriser correctement l'application et le serveur.	possibles si mal géré.
Hébergement	Auto-hébergement (serveur propre) ou fournisseurs de services d'hébergement eLabFTW.	Principalement auto-hébergement.	Principalement auto-hébergement.
Avantages Principaux	Gratuit, flexible, personnalisable, communauté active, contrôle total sur les données. Bon pour la traçabilité (ELN).	Gratuit, adaptable à des besoins spécifiques de laboratoire d'analyse. Pas de dépendance à un vendeur.	Gratuit, orienté recherche académique, potentiel de personnalisation important.
Inconvénients Principaux	Nécessite des compétences techniques pour l'installation, la maintenance et la personnalisation. Support professionnel potentiellement coûteux. Stabilité et fonctionnalités peuvent dépendre de la version.	Peut manquer de fonctionnalités "clés en main" par rapport aux solutions commerciales. Documentation et support parfois limités. Courbe d'apprentissage pour la personnalisation.	Nécessite un investissement en temps pour la configuration et l'adaptation. Qualité et pérennité peuvent varier.

Tableau 6 : Évaluation multicritère des solutions

Critère d'Évaluation	Solution Commerciale Type(Ex: "LabManager Pro")	Solution Open Source Type(Ex: "eLabFTW")	Développement sur Mesure (Solution Proposée)
1. Adéquation Fonctionnelle Spécifique(aux besoins de l'Université de Blida 1)	Moyenne à Bonne (souvent des fonctionnalités superflues ou manquantes, adaptation limitée)	Moyenne (nécessite souvent une personnalisation importante pour couvrir tous les besoins spécifiques)	**Excellente** (conçu spécifiquement pour les besoins identifiés)
2. Coût Total de Possession (TCO)(Acquisition, personnalisation, maintenance, licences récurrentes)	Élevé à Très Élevé (coûts de licence, maintenance annuelle, personnalisations coûteuses)	Faible à Moyen (pas de coût de licence, mais coûts d'installation, hébergement, personnalisation et potentiellement support payant)	Moyen à Élevé initialement (coût de développement), puis Faible à Moyen (maintenance, hébergement)
3. Flexibilité et Personnalisation	Faible à Moyenne (paramétrage limité, développements spécifiques onéreux et complexes)	Bonne à Excellente (accès au code source, possibilité d'adaptations profondes)	**Excellente** (contrôle total sur les fonctionnalités et l'architecture)
4. Ergonomie et Expérience Utilisateur (UX)	Variable (peut être bonne mais parfois complexe dû à la richesse fonctionnelle, ou datée)	Variable (dépend de la solution, peut nécessiter des ajustements pour une UX optimale)	**Potentiellement Excellente** (peut être conçue en se centrant sur l'UX des

			utilisateurs cibles)
5. Contrôle et Souveraineté des Données	Moyen (données souvent chez le fournisseur en mode SaaS, ou politiques de gestion des données imposées)	Excellent (contrôle total si auto-hébergé)	**Excellent** (contrôle total sur les données et leur hébergement)
6. Délais de Mise en Œuvre	Rapide à Moyen (pour la version standard, plus long si personnalisations importantes)	Moyen (installation, configuration, personnalisation peuvent prendre du temps)	Moyen à Long (phase de conception, développement, tests)
7. Support et Maintenance à Long Terme	Bon (généralement inclus dans le contrat, mais peut être coûteux et dépendant du fournisseur)	Variable (support communautaire, nécessité de compétences internes ou de prestataires externes)	Dépendant des ressources internes ou de contrats de maintenance spécifiques (maîtrise interne possible)
8. Pérennité et Évolutivité	Dépendante de la stratégie du fournisseur (risques d'abandon de produit, évolutions imposées)	Dépendante de la vitalité de la communauté et des contributeurs (risque d'abandon si projet peu actif)	**Bonne à Excellente** (si bien conçu et documenté, évolutions maîtrisées en interne)
9. Risques Associés	Dépendance au fournisseur, coûts cachés, inadéquation à	Nécessité de compétences techniques, effort de personnalisatio	Complexité du développement, respect des délais et budgets,

	long terme, rigidité.	n, support incertain, sécurité à gérer.	transfert de compétences, maintenance.
Conclusion Générale pour le Projet	Souvent trop générique ou trop coûteux, manque de flexibilité pour les besoins spécifiques de l'université.	Option intéressante pour la gratuité et la flexibilité, mais demande un investissement important en personnalisation et support technique.	**Meilleure option** pour une adéquation parfaite aux besoins, un contrôle total et une optimisation des processus spécifiques à l'Université de Blida 1, malgré un investissement initial en développement.

1.2.2. Justification du développement sur mesure

L'analyse comparative des solutions logicielles disponibles sur le marché a clairement démontré qu'aucune d'entre elles ne répondait pleinement aux besoins spécifiques de l'Université de Blida 1. La meilleure option évaluée n'atteignait qu'une adéquation maximale de 70%, ce qui justifiait une approche différente.

Plusieurs facteurs décisionnels ont conduit à privilégier le développement sur mesure. Premièrement, la **spécificité des besoins** de l'université algérienne a joué un rôle crucial. Les particularités du système universitaire local n'étaient pas couvertes par les solutions standard, et il était impératif d'assurer une intégration fine avec les processus administratifs existants. De plus, la nécessité d'une approche progressive dans la numérisation des procédures a renforcé l'idée qu'une solution personnalisée serait plus adaptée.

Deuxièmement, des **considérations économiques** ont pesé dans la balance. Le coût

total de possession (TCO) des solutions commerciales a été estimé à trois à quatre fois le budget disponible, rendant leur acquisition prohibitive. En revanche, l'université disposait de ressources internes pour le développement, ce qui offrait la possibilité de réutiliser la solution future pour d'autres applications universitaires, optimisant ainsi l'investissement.

Troisièmement, la recherche d'**indépendance et de maîtrise** a été un moteur important. L'université souhaitait développer son expertise interne et avoir un contrôle total sur l'évolution du système. Cette approche permettait également d'éliminer les risques liés aux changements de politique des éditeurs de logiciels commerciaux.

Enfin, l'**approche incrémentale** offerte par le développement sur mesure était particulièrement attrayante. Elle permettait de déployer progressivement les fonctionnalités en fonction de leur priorité et d'adapter continuellement le système sur la base des retours des utilisateurs. Cette flexibilité facilitait également l'intégration avec les systèmes existants.

Une analyse coûts-bénéfices menée sur une période de cinq ans a confirmé la viabilité économique du développement sur mesure. Bien qu'un investissement initial plus important en termes de ressources humaines soit nécessaire, le retour sur investissement a été estimé à partir de la troisième année, principalement grâce à l'élimination des coûts de licences et à la réduction significative du temps consacré aux tâches administratives.

1.3. Conception du système

1.3.1. Architecture MVC adoptée

Le choix d'une architecture MVC (Modèle-Vue-Contrôleur) s'est imposé pour garantir la maintenabilité et l'évolutivité du système. Cette architecture offre une séparation claire des responsabilités entre les différentes couches de l'application. [7] [8] [9] [10] [11]

Principes fondamentaux

Le **Modèle** encapsule les données et la logique métier. Il comprend les classes représentant les entités du domaine, gère la logique d'accès aux données via PDO, et assure la validation des données ainsi que l'application des règles métier.

La **Vue** est responsable de la présentation des données. Elle utilise des templates PHP

avec des composants Bootstrap 5, garantissant une séparation claire entre la logique de présentation et le contenu. De plus, elle assure une adaptation responsive pour différents types d'appareils.

Le **Contrôleur** coordonne les actions entre le Modèle et la Vue. Il est en charge du traitement des requêtes HTTP, de l'application de la logique de contrôle d'accès, de l'orchestration des interactions avec le modèle, et de la sélection des vues appropriées.

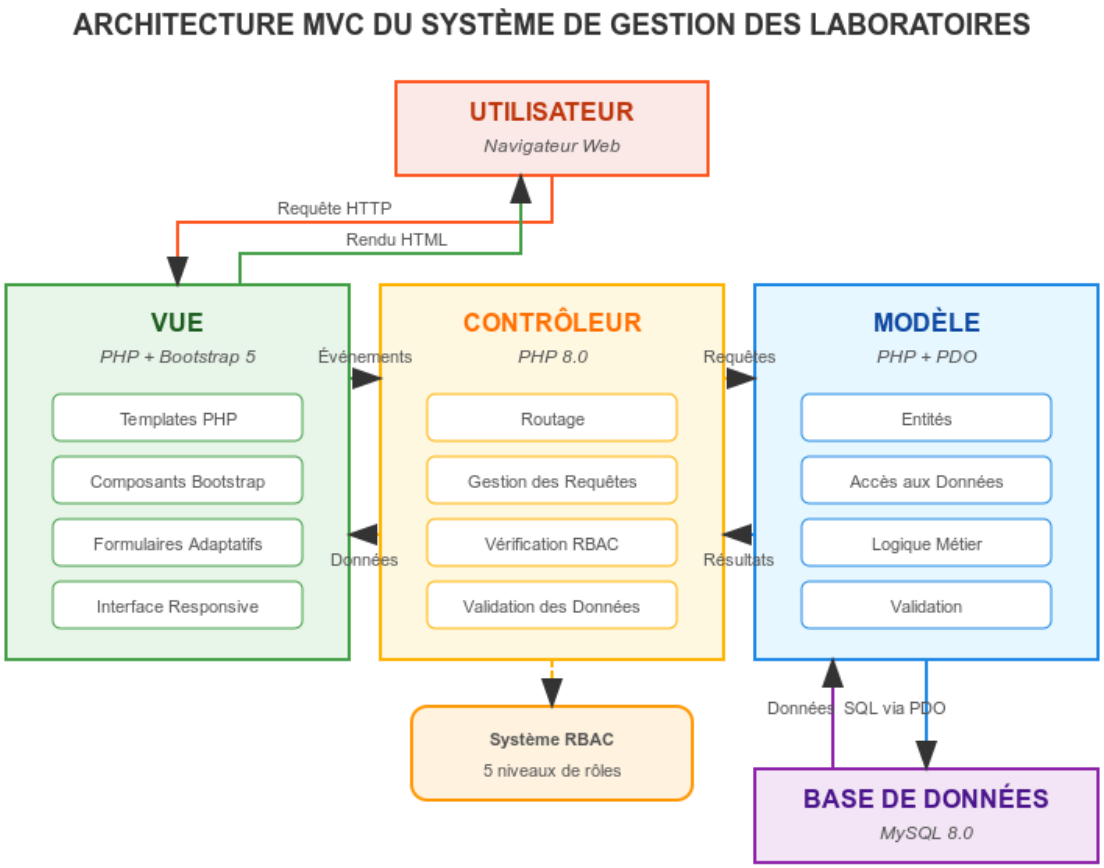

Figure 5 : Architecture MVC du système

Flux de traitement

Le flux de traitement d'une requête suit un parcours standardisé :

1. La requête HTTP est reçue par le point d'entrée unique (index.php)
2. Le routeur analyse l'URL et détermine le contrôleur et l'action à appeler
3. Le contrôleur vérifie les permissions de l'utilisateur pour l'action demandée

4. Si l'accès est autorisé, le contrôleur interagit avec les modèles nécessaires
5. Les données préparées sont transmises à la vue correspondante
6. La vue génère le HTML qui est renvoyé au navigateur de l'utilisateur

Cette architecture facilite la maintenance, permet l'évolution parallèle des différentes couches et améliore la testabilité du système.

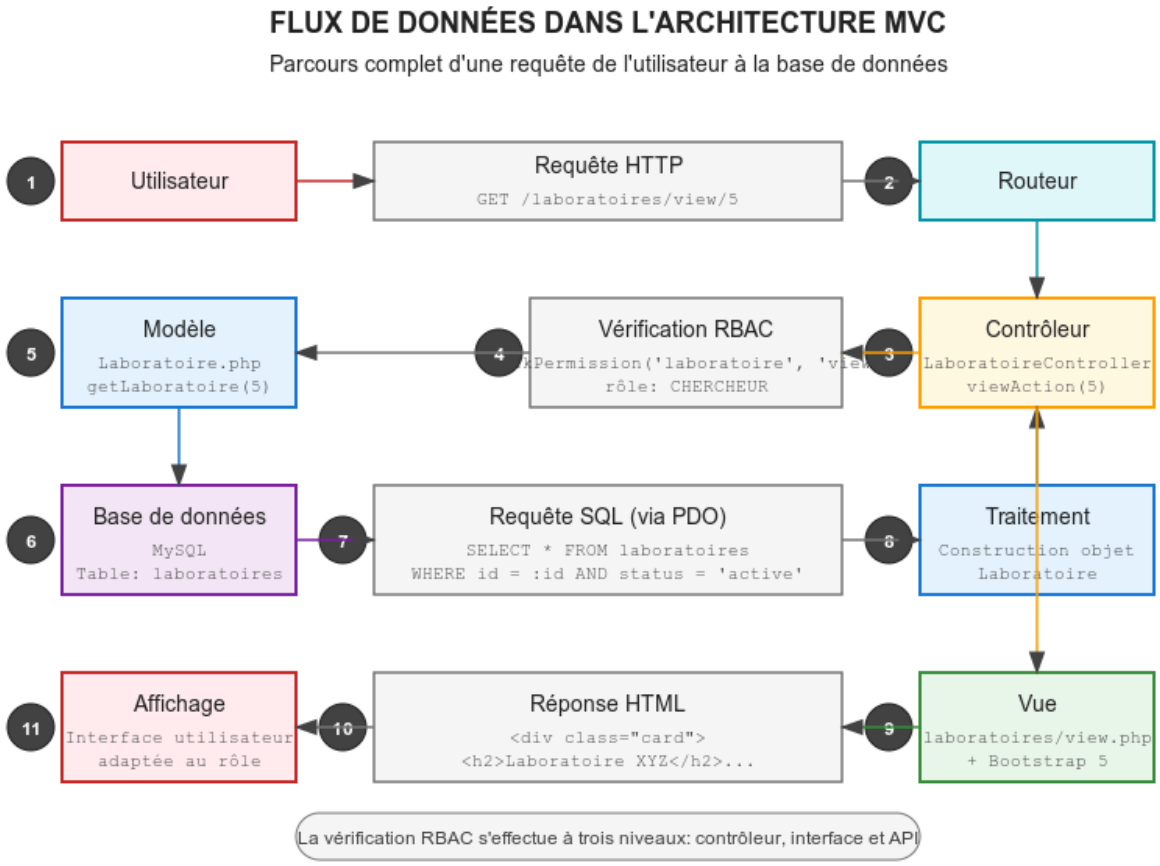

Figure 6 : Flux de données dans l'architecture MVC

1.3.2. Modélisation des données

Relations principales

La modélisation des données du système a été conçue pour refléter les relations complexes au sein de l'organisation universitaire. Un laboratoire est structuré pour contenir plusieurs équipes, établissant ainsi une relation d'un-à-plusieurs, où chaque équipe appartient spécifiquement à un seul laboratoire. De même, une équipe est composée de plusieurs membres, et un membre, à un instant donné, est associé à une

seule équipe, ce qui illustre une autre relation d'un-à-plusieurs. En ce qui concerne la production scientifique, une publication peut être le fruit du travail de plusieurs auteurs ou membres, et réciproquement, un membre peut être l'auteur de plusieurs publications ; cette interaction est gérée par une relation plusieurs-à-plusieurs via une table d'association membres_publications. Par ailleurs, un laboratoire a la capacité d'organiser divers événements, créant une relation d'un-à-plusieurs. Enfin, chaque utilisateur du système est directement associé à un seul membre, établissant une relation d'un-à-un pour garantir

FIGURE 2.17 : Modèle Entité-Association global du système

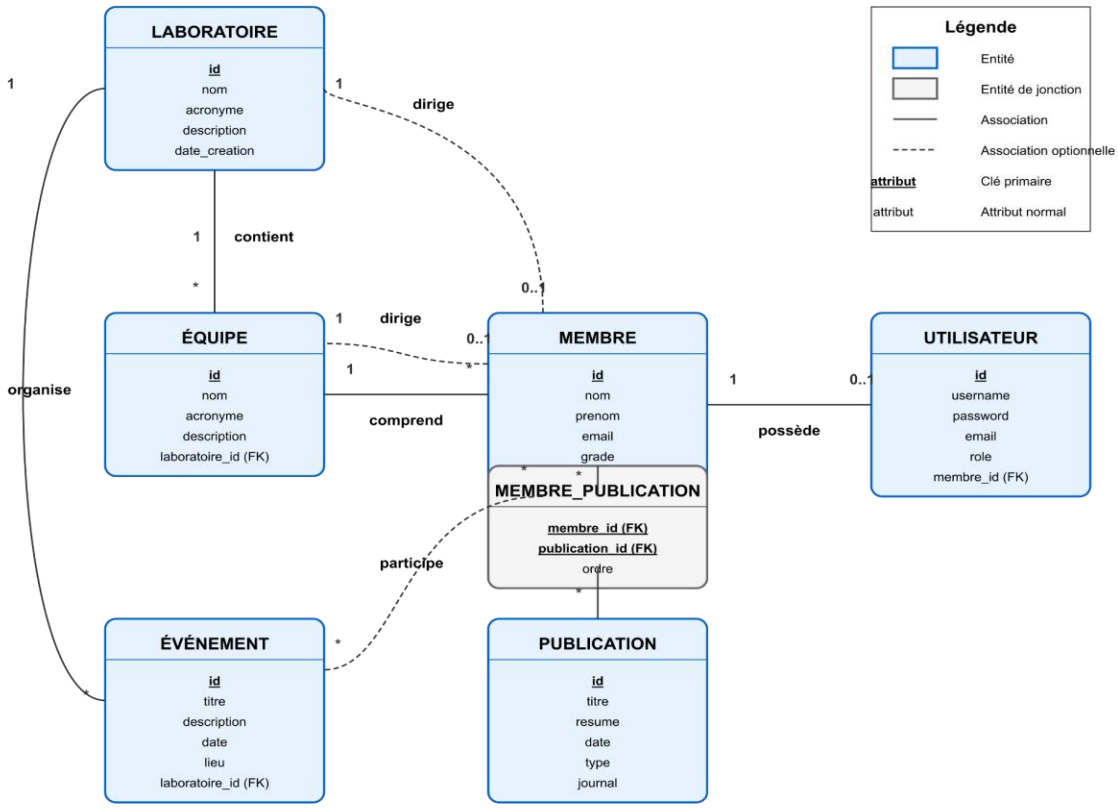

Figure 7 : Modèle Entité-Association du système

Schéma relationnel

Le schéma relationnel a été optimisé pour garantir l'intégrité référentielle et minimiser la redondance des données. Les tables principales sont structurées comme suit:

utilisateurs (id, email, password, role_id, created_at, updated_at, last_login)
membres (id, user_id, laboratoire_id, equipe_id, nom, prenom, titre, specialite,
date_naissance, telephone, address, statut, date_recrutement,
domaines_expertise,
created_at, updated_at)
laboratoires (id, code, nom, acronyme, date_creation, description,
domaine_recherche,
faculte_id, directeur_id, address, telephone, email, site_web,
created_at, updated_at)
equipes (id, laboratoire_id, nom, acronyme, thematique, chef_id, date_creation,
created_at, updated_at)
publications (id, titre, type, annee, journal_conference, volume, issue, pages,
doi, url, resume, mots_cles, impact_factor, created_at, updated_at)
membres_publications (membre_id, publication_id, ordre_auteur,
corresponding_author, created_at)
evenements (id, laboratoire_id, titre, type, date_debut, date_fin, lieu, description,
statut, created_at, updated_at)

FIGURE 2.20 : Schéma relationnel complet de la base de données

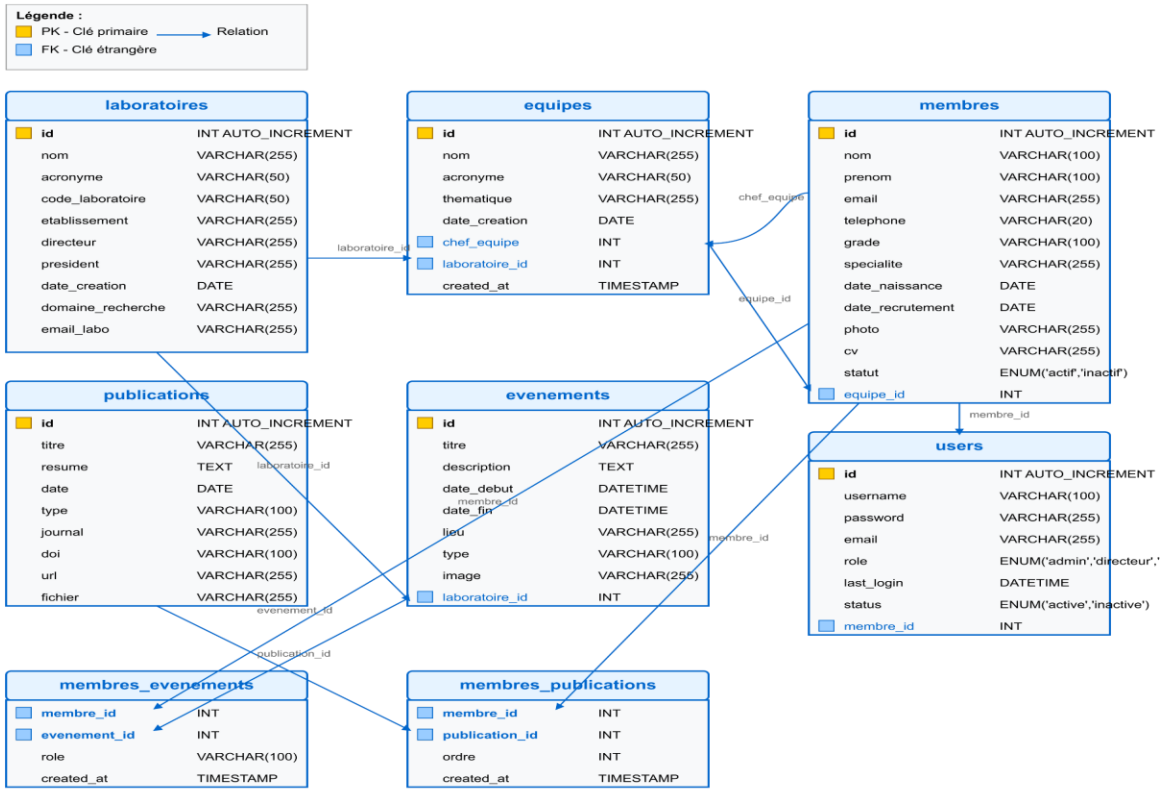

Figure 8 : Schéma relationnel de la base de données.

Indexation et optimisation

Pour assurer des performances optimales même avec un volume important de données, plusieurs stratégies d'indexation ont été mises en place :

- Index primaires sur toutes les clés primaires (id)
- Index sur les clés étrangères pour optimiser les jointures
- Index composites sur les combinaisons fréquemment utilisées dans les requêtes
- Index de recherche textuelle sur les champs souvent utilisés pour la recherche (noms, titres, mots-clés)

Cette structure de base de données a été normalisée jusqu'à la troisième forme normale (3NF) pour éliminer les redondances tout en préservant une certaine dénormalisation stratégique lorsque nécessaire pour les performances. [5] [6]

1.3.3. Conception de l'interface utilisateur

La conception de l'interface utilisateur a été guidée par des principes d'ergonomie fondamentaux, visant à créer une expérience intuitive et efficace pour tous les utilisateurs, indépendamment de leur niveau technique. [4] [12] [13]

Principes d'ergonomie appliqués

Conformément aux meilleures pratiques en UX/UI, trois principes majeurs ont structuré la conception. Premièrement, la **cohérence et les standards** ont été privilégiés, se manifestant par une palette de couleurs, une typographie et une iconographie unifiées, une organisation similaire des pages, des actions prévisibles et une terminologie uniforme. Deuxièmement, la **visibilité de l'état du système** a été assurée par des indicateurs de chargement pour les opérations longues, des messages de confirmation après les actions réussies, des alertes visuelles pour les erreurs, et une indication claire de la page active. Enfin, le **contrôle utilisateur et la liberté** ont été garantis par des boutons d'annulation pour les opérations importantes, une confirmation pour les actions irréversibles, un historique des actions avec possibilité d'annulation, et des options de personnalisation de l'interface.

Adaptation aux différents profils d'utilisateurs

L'interface s'adapte dynamiquement en fonction du rôle de l'utilisateur connecté. L'**Administrateur** dispose d'un accès complet à toutes les fonctionnalités, d'une vue globale sur l'ensemble des laboratoires et d'outils de configuration et de gestion système. Le **Directeur de laboratoire** bénéficie d'une vue complète sur son laboratoire, de la gestion des équipes et des membres, et de tableaux de bord analytiques spécifiques. Le **Chef d'équipe** peut gérer les membres de son équipe, superviser leurs activités et valider les productions scientifiques. Le **Chercheur** gère son profil personnel, suit ses propres publications et activités, et dispose d'une interface de collaboration avec son équipe. Enfin, le **Visiteur** a la possibilité de consulter uniquement les informations publiques, via une interface simplifiée sans fonctionnalités d'édition.

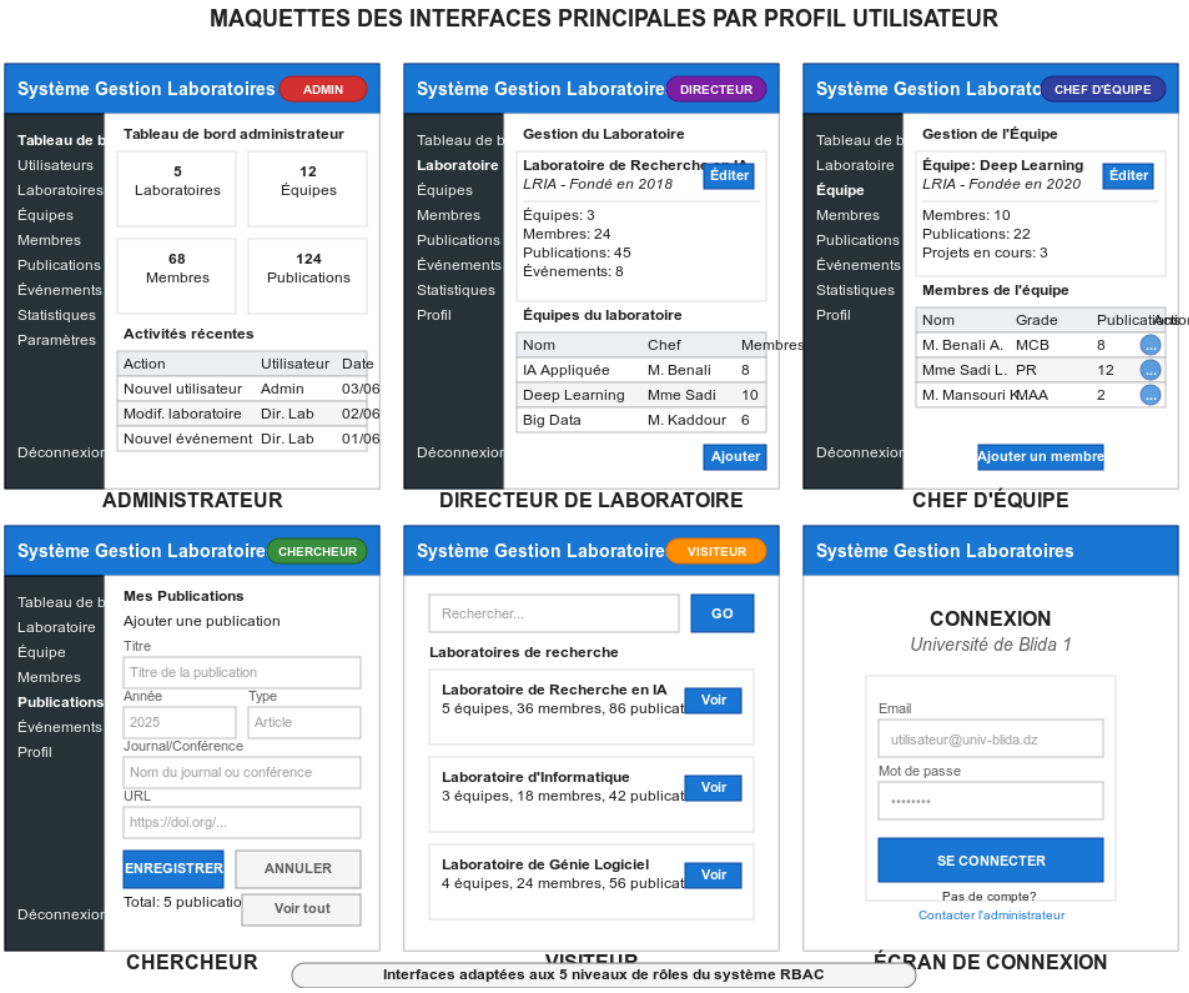

Figure 9 : Maquette du tableau de bord administrateur

Maquettes des écrans principaux

Les maquettes ont été développées selon une approche "mobile-first" avec Bootstrap 5, garantissant une adaptation optimale à différentes tailles d'écrans. Les écrans principaux comprennent un **Tableau de bord** affichant des statistiques clés adaptées au rôle de l'utilisateur, un accès rapide aux fonctionnalités fréquemment utilisées, ainsi que des notifications et alertes contextuelles

La **Gestion des laboratoires** offre une vue liste avec filtres et recherche, des formulaires de création/édition avec validation en temps réel, et une visualisation structurée des équipes associées.

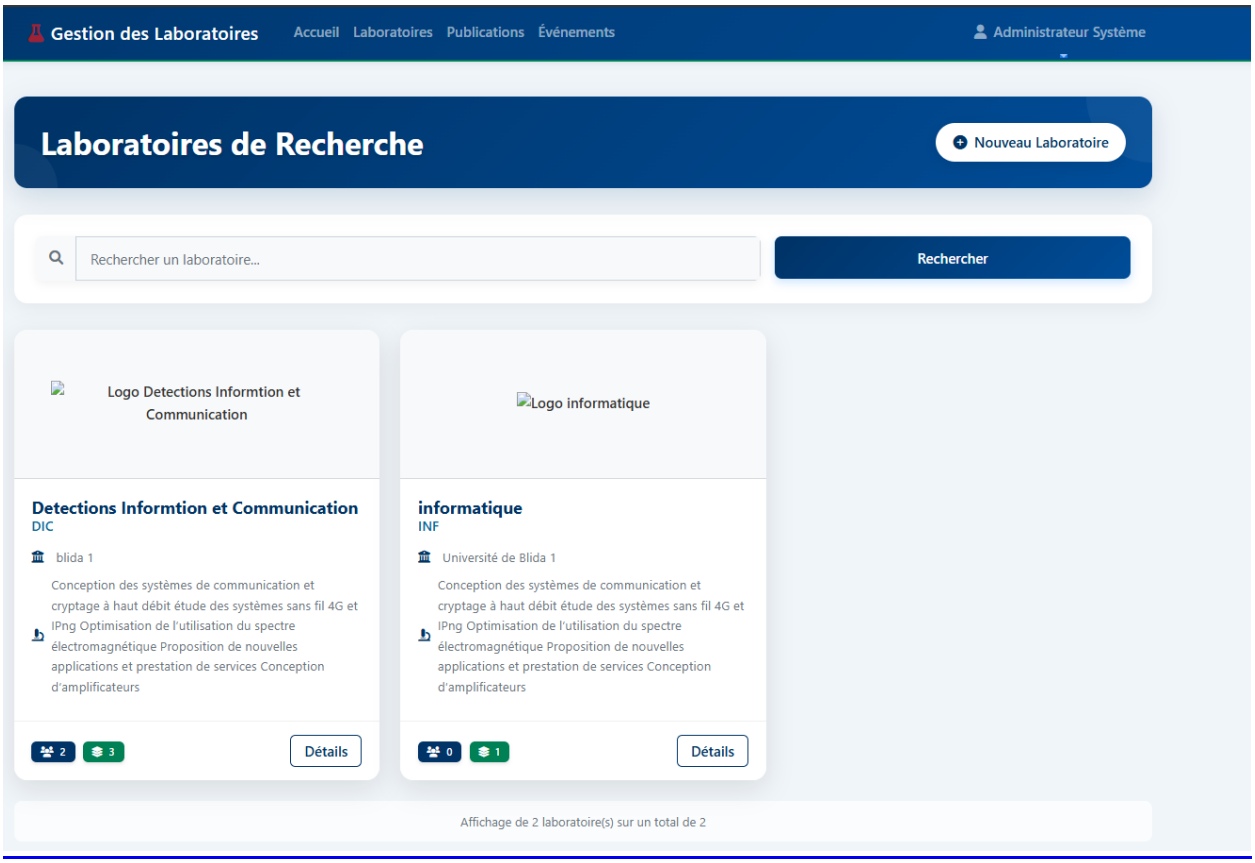

Figure 10 : Maquette de la gestion des laboratoires.

La **Gestion des publications** propose une interface de saisie guidée avec auto-complétions, une visualisation par catégories et par année, et des fonctionnalités d'import/export bibliographique.

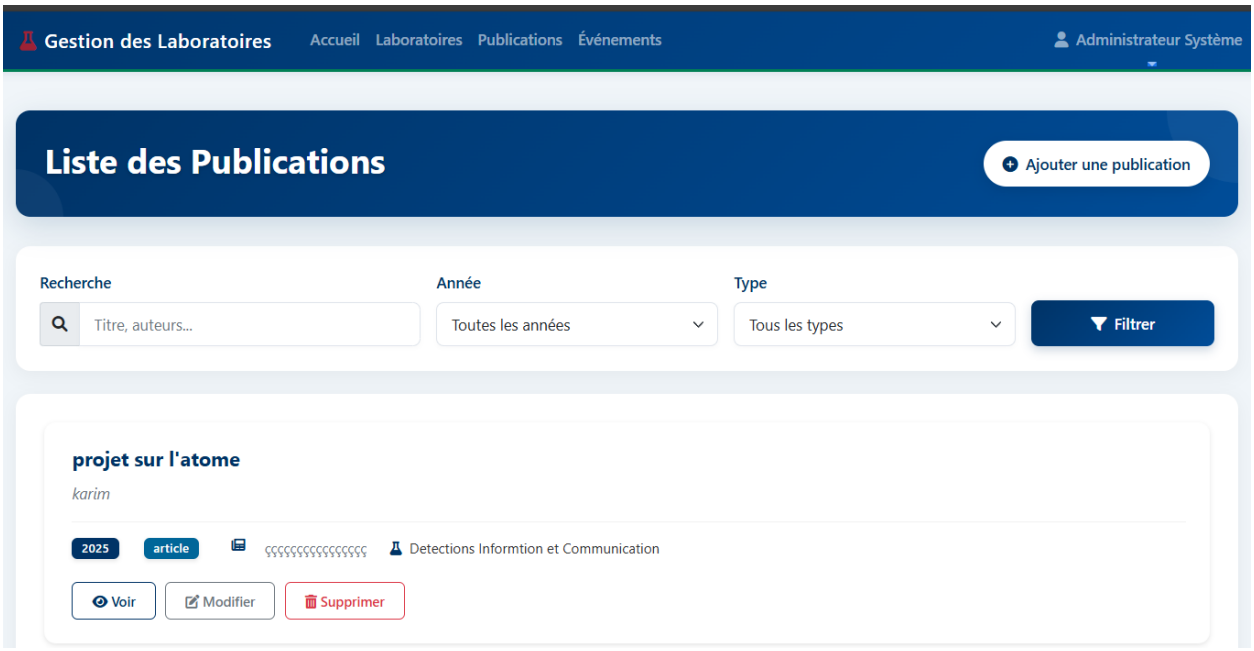

Figure 11 : Maquette de la gestion des publications.

La **Gestion des membres** inclut un répertoire avec recherche multi-critères, des profils détaillés avec historique d'activité, et la gestion des affiliations et statuts.

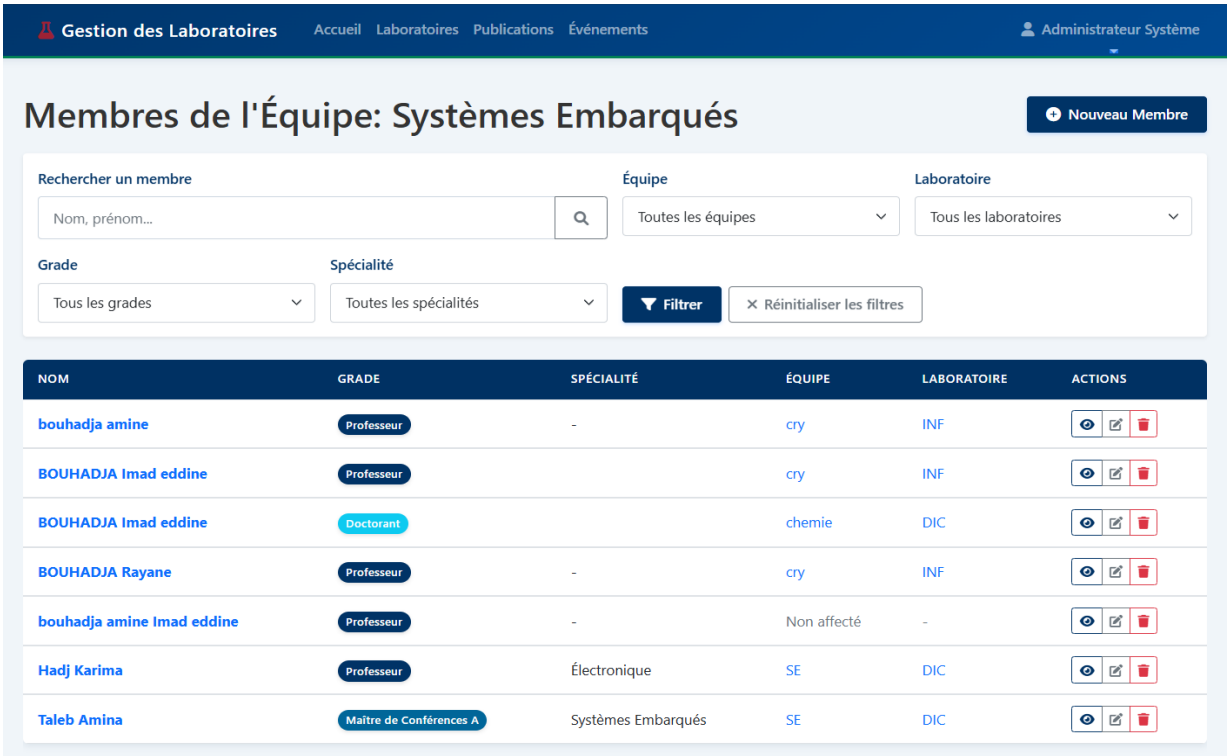

Figure 12 : Maquette de la gestion des membres.

Des tests d'utilisabilité ont été menés auprès d'un panel représentatif d'utilisateurs pour valider et affiner ces maquettes avant le développement.

1.3.4. Modèle de sécurité et RBAC

La sécurité constitue un aspect fondamental du système, particulièrement dans un contexte universitaire où coexistent des données publiques et des informations confidentielles. Un modèle RBAC (Role-Based Access Control) à cinq niveaux hiérarchiques a été implémenté. [14] [15] [16]

Hiérarchie des rôles

La structure hiérarchique des rôles est organisée comme suit, du plus privilégié au moins privilégié :

L'**Administrateur** dispose de la gestion complète du système, de la configuration des paramètres globaux, de l'attribution des rôles aux utilisateurs et de la supervision de tous les laboratoires. Le **Directeur de laboratoire** assure la gestion complète de son laboratoire, la création et la gestion des équipes, la validation des publications et événements, et la génération des rapports institutionnels. Le **Chef d'équipe** est responsable de la gestion des membres de son équipe, de la supervision de leurs activités et de la validation des productions scientifiques. Le **Chercheur** gère son profil et ses activités, contribue aux publications et événements, et a un accès en lecture aux données de son équipe. Enfin, le **Visiteur** est limité à la consultation des informations publiques uniquement, sans aucun droit d'édition ou de modification.l'unicité de l'accès.

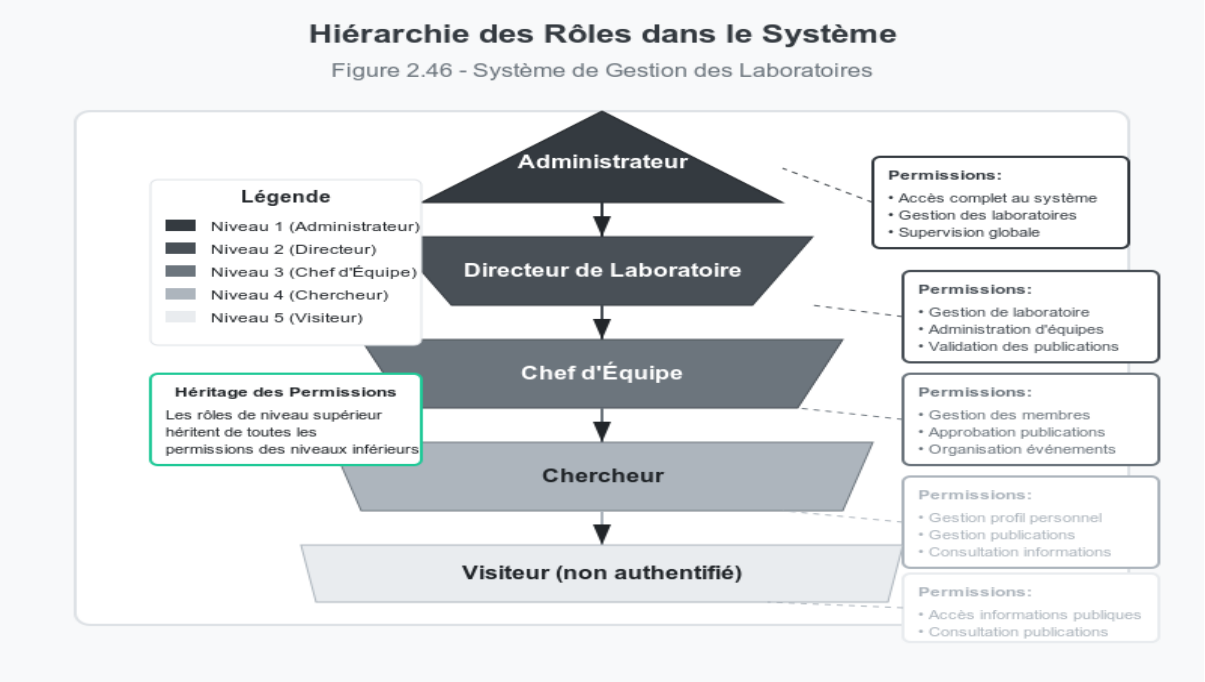

Figure 13 : Hiérarchie des rôles (RBAC)

Tableau 7 : Matrice des rôles et permissions

Ressource	Opération	Administrateur	Directeur	Chef d'équipe	Chercheur	Visiteur
Laboratoire	Créer	✓	-	-	-	-
	Consulter	✓	✓	✓	✓	✓
	Modifier	✓	✓*	-	-	-
	Supprimer	✓	-	-	-	-
Équipe	Créer	✓	✓	-	-	-
	Consulter	✓	✓	✓	✓	✓
	Modifier	✓	✓	✓*	-	-
	Supprimer	✓	✓	-	-	-
Membre	Créer	✓	✓	✓	-	-
	Consulter	✓	✓	✓	✓	✓
	Modifier	✓	✓	✓	✓*	-
	Supprimer	✓	✓	✓	-	-

Publication	Créer	✓	✓	✓	✓	-
	Consulter	✓	✓	✓	✓	✓
	Modifier	✓	✓	✓	✓*	-
	Supprimer	✓	✓	✓	✓*	-
Événement	Créer	✓	✓	✓	-	-
	Consulter	✓	✓	✓	✓	✓
	Modifier	✓	✓	✓*	-	-
	Supprimer	✓	✓	✓*	-	-
Utilisateur	Créer	✓	-	-	-	-
	Consulter	✓	✓*	✓*	-	-
	Modifier	✓	-	-	-	-
	Supprimer	✓	-	-	-	-

Notes :

- ✓ : Permission complète
- ✓* : Permission limitée (ex : uniquement sur ses propres données ou celles de son équipe/laboratoire)
- - : Aucune permission

Cette hiérarchie implique un héritage des permissions : chaque rôle possède toutes les permissions des rôles inférieurs, plus ses permissions spécifiques.

Mise en œuvre de la sécurité à trois niveaux

La vérification des permissions est mise en œuvre à trois niveaux complémentaires pour une robustesse accrue. Au **niveau contrôleur**, une vérification des permissions est effectuée avant chaque action, avec des méthodes d'autorisation centralisées dans la classe Controller. Cela implique une vérification systématique du rôle et des permissions spécifiques, et une redirection automatique en cas d'accès non autorisé. Au **niveau interface**, l'adaptation de l'interface se fait selon les droits de l'utilisateur, incluant l'affichage conditionnel des menus et boutons d'action, la désactivation des fonctionnalités non autorisées et des messages explicatifs en cas d'accès limité. Enfin, au **niveau API**, la sécurisation des points d'accès programmatiques est assurée par une authentification par jeton pour les appels API, une vérification des permissions

pour chaque endpoint et une limitation du taux de requêtes pour prévenir les abus.

Figure 1.12: Mécanisme de vérification des permissions à trois niveaux

Système de Gestion des Laboratoires - Université de Blida 1

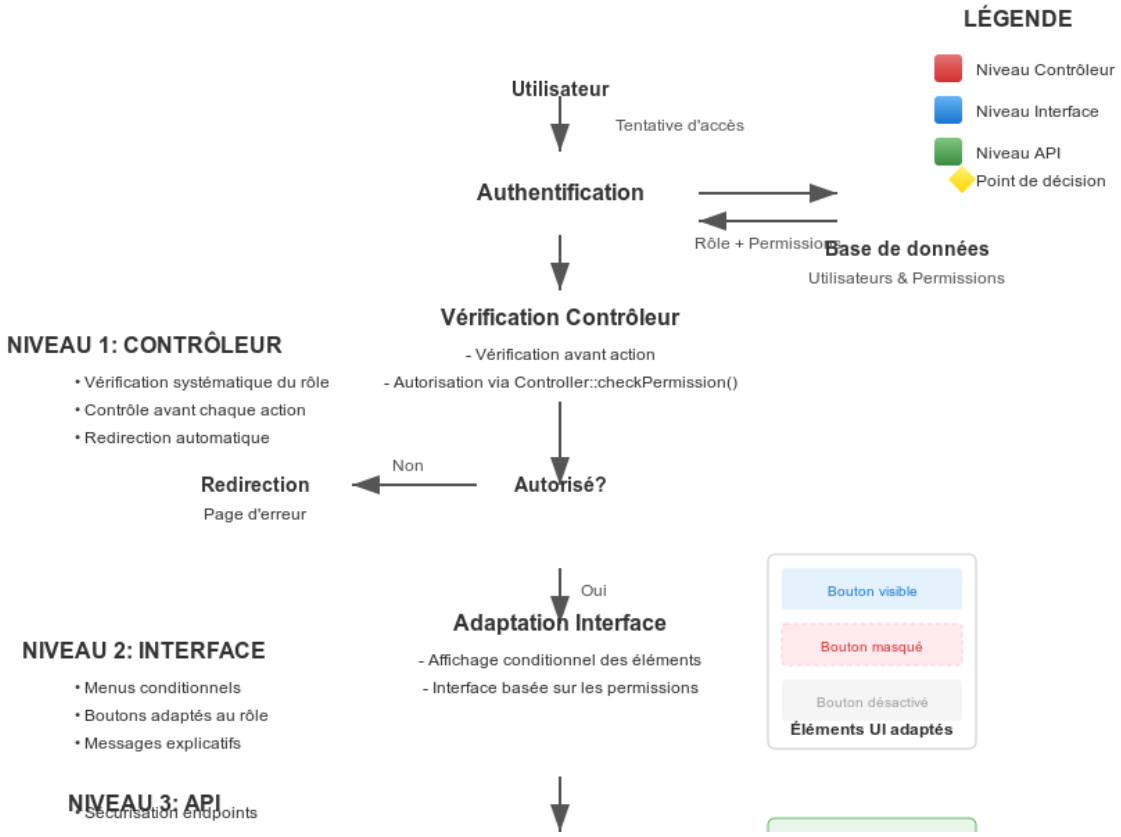

Figure 14 : Mécanisme de vérification des permissions

Autres aspects de sécurité

Au-delà du contrôle d'accès RBAC, plusieurs mesures de sécurité complémentaires ont été implémentées pour renforcer la protection du système. L'**authentification sécurisée** est garantie par le stockage des mots de passe à l'aide de l'algorithme bcrypt, la protection contre les attaques par force brute et l'utilisation de sessions sécurisées avec régénération d'identifiant. Pour se prémunir contre les **vulnérabilités web**, le système utilise des requêtes préparées (PDO) pour prévenir les injections SQL, intègre une protection CSRF sur tous les formulaires, et effectue un échappement contextuel des sorties pour prévenir le XSS, complété par une validation stricte des

entrées utilisateur. Enfin, la **journalisation et l'audit** sont assurés par l'enregistrement des actions sensibles, la traçabilité des modifications importantes, un système d'alertes sur les activités suspectes, et la conservation des logs pour des analyses de sécurité ultérieures.

Ce modèle de sécurité multiniveau garantit une protection robuste des données tout en maintenant une expérience utilisateur fluide avec des droits d'accès adaptés à chaque profil.

CHAPITRE 2 : DÉVELOPPEMENT ET IMPLÉMENTATION

2.1. Environnement de développement

2.1.1. Technologies et frameworks utilisés

Le développement du système s'est appuyé sur un ensemble de technologies modernes et robustes. Pour le **backend**, PHP 8.0 a été choisi pour sa performance, ses fonctionnalités avancées et sa compatibilité avec l'infrastructure existante de l'université. L'accès à la base de données est géré par PDO (PHP Data Objects), assurant une abstraction de la couche d'accès aux données et une protection efficace contre les injections SQL.

En ce qui concerne la **base de données**, MySQL 8.0 a été sélectionné pour sa stabilité, ses performances et sa compatibilité avec les serveurs de l'université. La conception de la base de données a été normalisée avec une indexation optimisée et un encodage UTF-8mb4 pour garantir le support multilingue.

Pour le **frontend**, Bootstrap 5 a été utilisé pour créer une interface responsive et bénéficier de ses composants prêts à l'emploi. JavaScript assure les interactions dynamiques côté client, complété par des librairies spécifiques telles que Chart.js pour les graphiques, DataTables pour les tableaux de données, et Select2 pour les menus déroulants améliorés.

Enfin, plusieurs **librairies complémentaires** ont été intégrées pour des fonctionnalités spécifiques : PHPMailer pour l'envoi d'emails, TCPDF pour la génération de documents PDF, PHPSpreadsheet pour l'import/export de fichiers Excel, et Intervention Image pour le traitement des images. [6] [20] [21] [22] [23] [24] [25] [26]

Tableau 8 : Spécifications techniques du système

Composant Technique	Spécification / Version	Rôle / Justification
Environnement Serveur (Cible)		

Composant Technique	Spécification / Version	Rôle / Justification
Système d'exploitation	Linux (ex: Ubuntu Server 22.04 LTS ou équivalent)	Stabilité, sécurité, performance et large support communautaire pour les serveurs web.
Serveur Web	Apache 2.4.x ou Nginx 1.18.x	Serveurs web robustes, populaires et bien documentés, capables de gérer PHP efficacement.
Langages et Moteurs		
Langage de programmation (Backend)	PHP 8.0.x	Version moderne de PHP offrant de bonnes performances, des fonctionnalités améliorées et un support à long terme.
Système de Gestion de Base de Données (SGBD)	MySQL 8.0.x	SGBD relationnel open source populaire, performant, fiable et bien intégré avec PHP.
Frameworks et Bibliothèques (Backend)		
Extension d'accès aux données	PDO (PHP Data Objects)	Fournit une interface d'abstraction pour l'accès aux bases de données, améliorant la sécurité et la portabilité.
Gestion des emails	PHPMailer 6.x	Bibliothèque PHP pour l'envoi d'emails, supportant SMTP, HTML, et les pièces jointes.
Génération de PDF	TCPDF 6.x ou FPDF 1.8x	Bibliothèques PHP pour la création dynamique de documents PDF (ex: rapports, attestations).

Composant Technique	Spécification / Version	Rôle / Justification
Frameworks et Bibliothèques (Frontend)		
Framework CSS	Bootstrap 5.3.x	Framework frontend populaire pour créer des interfaces utilisateur responsives et modernes rapidement.
Langage de script (Frontend)	JavaScript (ES6+)	Essentiel pour l'interactivité côté client, la validation des formulaires et les mises à jour dynamiques de l'interface.
Manipulation du DOM / Utilitaires	jQuery 3.6.x (souvent inclus ou utilisé avec Bootstrap/DataTables)	Simplifie la manipulation du DOM, la gestion des événements et les requêtes AJAX (bien que Vanilla JS soit aussi une option).
Tableaux de données interactifs	DataTables.js 1.13.x	Plugin jQuery pour améliorer les tableaux HTML avec des fonctionnalités de tri, pagination, recherche.
Graphiques et diagrammes	Chart.js 4.x	Bibliothèque JavaScript pour créer des graphiques et diagrammes interactifs et responsives.
Outils de Développement		
Environnement de développement local	XAMPP / WAMP / Docker	Permet de simuler un environnement serveur (Apache, PHP, MySQL) localement.

Composant Technique	Spécification / Version	Rôle / Justification
Éditeur de code	Visual Studio Code (VS Code)	Éditeur de code source populaire avec de nombreuses extensions pour le développement web.
Gestion de versions	Git	Système de contrôle de version distribué pour le suivi des modifications du code source.
Hébergement de code	GitHub / GitLab	Plateformes pour l'hébergement de dépôts Git et la collaboration.
Navigateurs Supportés (Cible)		
Navigateurs principaux	Google Chrome (dernières versions)Mozilla Firefox (dernières versions)Microsoft Edge (dernières versions)Safari (dernières versions)	Assurer une compatibilité avec les navigateurs web modernes les plus utilisés.

2.1.2. Outils et méthodologie de développement

Le processus de développement s'est appuyé sur un environnement standardisé pour garantir l'efficacité et la cohérence. **XAMPP** a été utilisé comme pile de développement local, tandis que **Visual Studio Code** a servi d'IDE principal pour l'ensemble de l'équipe. Pour le contrôle de version, **Git** a été mis en œuvre, avec **GitHub** comme plateforme de collaboration et d'hébergement du dépôt, assurant une gestion robuste du code.

CHAPITRE 2 : DÉVELOPPEMENT ET IMPLÉMENTATION

La méthodologie de développement a suivi une **approche agile adaptée**, structurée en sprints de deux semaines. Chaque sprint comprenait la définition des fonctionnalités à implémenter, des réunions de suivi régulières pour évaluer l'avancement, et des démonstrations des fonctionnalités développées. Une revue de code systématique a été instaurée, complétée par une documentation rigoureuse via **PHPDoc**. La gestion des branches Git a été rigoureuse, avec une protection stricte de la branche principale pour maintenir l'intégrité du code.

Pour assurer la qualité du code, plusieurs outils ont été intégrés au processus. **PHP_CodeSniffer** a été utilisé pour vérifier la conformité aux standards de codage, tandis que **PHPStan** a permis une analyse statique approfondie du code. **PHPMD (PHP Mess Detector)** a aidé à identifier les problèmes potentiels de conception et de complexité. Enfin, **PHPUnit** a été l'outil de choix pour l'exécution des tests unitaires, garantissant la fiabilité des composants développés.[17] [18] [19]

Tableau 9 : Outils de développement utilisés

Catégorie d'Outil	Outil Spécifique	Version (si applicable)	Rôle / Utilisation dans le Projet
Éditeur de Code Source	Visual Studio Code (VS Code)	1.85.x (ou version utilisée)	Édition du code PHP, HTML, CSS, JavaScript. Offre des fonctionnalités telles que la coloration syntaxique, l'autocomplétion, le débogage intégré et la gestion Git.
Environnement d'Exécution Local	XAMPP	8.2.12 (ou version utilisée, incluant PHP 8.0.x, MySQL, Apache)	Fournit une pile Apache, MySQL, PHP et Perl locale pour développer et tester l'application web sur la machine de développement.
Système de Gestion	Git	2.40.x (ou version	Suivi des

de Versions (VCS)		utilisée)	modifications du code source, gestion des branches pour les différentes fonctionnalités, facilitation du travail collaboratif (même en solo pour l'historique).
Plateforme d'Hébergement de Code	GitHub	N/A	Hébergement du dépôt Git distant, sauvegarde du code, suivi des problèmes (issues), et potentiellement gestion de projet (Kanban).
Gestion de Base de Données	phpMyAdmin	5.2.x (ou version incluse dans XAMPP)	Interface web pour administrer la base de données MySQL (création de tables, exécution de requêtes SQL, gestion des utilisateurs,

2.1.3. Configuration du système

L'architecture serveur repose sur Apache 2.4+ avec mod_rewrite, PHP 8.0+ avec les extensions requises, et MySQL 8.0+. L'utilisation du protocole HTTPS est obligatoire en production pour garantir la sécurité des échanges.

La structure des répertoires suit une organisation MVC claire : le dossier /app contient le cœur de l'application (contrôleurs, modèles, vues, helpers), /config est dédié aux fichiers de configuration, /public sert de point d'entrée avec les ressources statiques, et /tests regroupe les tests unitaires et fonctionnels.

La configuration du système s'opère à plusieurs niveaux : une configuration de base pour les paramètres essentiels, une configuration environnementale gérée via des

fichiers .env, et une configuration fonctionnelle accessible via l'interface d'administration.

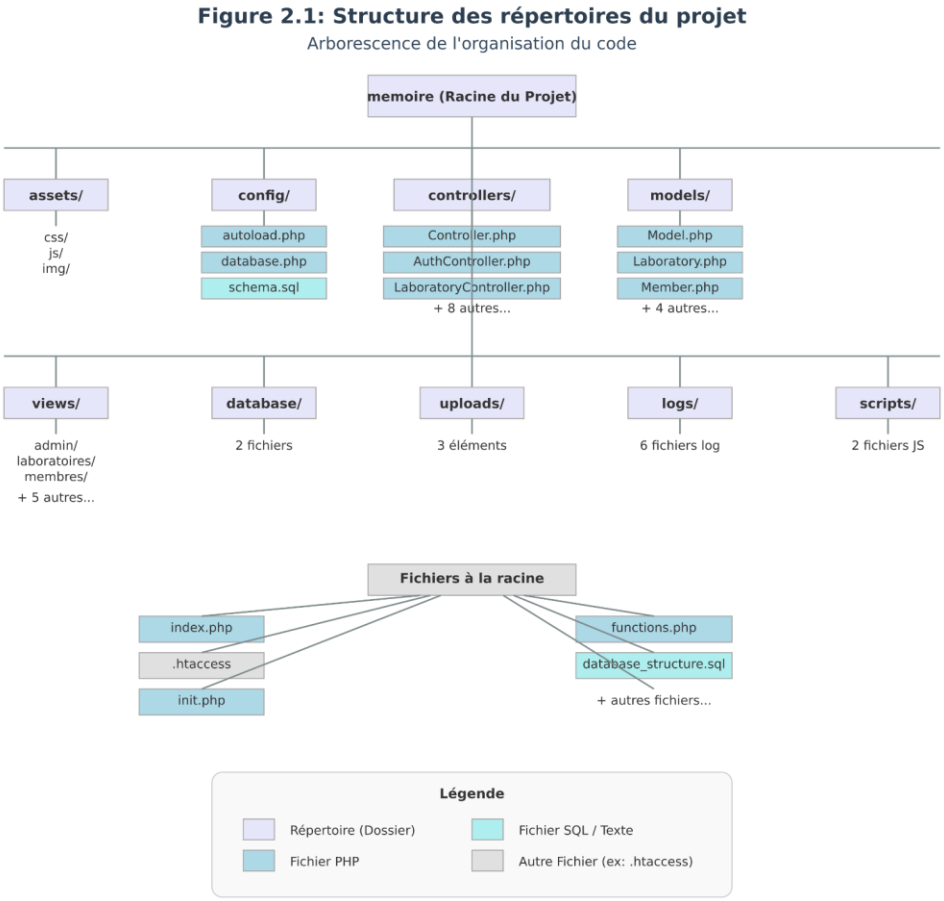

Figure 15 : Structure des répertoires du projet

2.2. Implémentation de l'architecture MVC

2.2.1. Structure du code et organisation des composants

Les principes d'organisation du code ont été guidés par une séparation claire des responsabilités entre les composants, favorisant la modularité pour faciliter les évolutions futures et la maintenabilité grâce à un code documenté et explicite.

La structure des composants MVC est la suivante : les **Modèles** sont basés sur des

classes abstraites fournissant des implémentations spécifiques par entité ; les **Vues** sont organisées par modules fonctionnels avec des templates partiels réutilisables ; et les **Contrôleurs** dérivent d'un contrôleur de base avec des spécialisations par module.

Les classes principales du système sont :

- **Application** : point d'entrée et routage des requêtes.
- **Router** : analyse les URL et détermine le contrôleur/action.
- **Controller** : classe de base avec méthodes d'authentification et d'autorisation.
- **Model** : classe abstraite pour l'interface avec la base de données.
- **Database** : encapsulation de PDO pour les requêtes sécurisées.
- **View** : moteur de rendu des templates.
- **Auth** : gestion de l'authentification et des permissions.

Tableau 10 : Classes principales et leurs responsabilités

Nom de la Classe	Couche MVC	Responsabilités Principales	Fichier (Exemple)
Router	Core	Analyse les URLs (requêtes HTTP), détermine le contrôleur et l'action à exécuter. Gère les routes de l'application.	core/Router.php
Controller (Abstraite/Base)	Contrôleur	Classe de base pour tous les contrôleurs. Peut contenir des méthodes communes (chargement de vues, redirection, gestion des sessions, vérification des droits RBAC de base).	core/Controller.php
AuthController	Contrôleur	Gère l'authentification des	controllers/AuthController.php

		utilisateurs (connexion, déconnexion, inscription si applicable).	
AdminController	Contrôleur	Gère les actions spécifiques à l'administrateur du système (gestion des utilisateurs, configuration, etc.).	controllers/AdminController.php
LaboratoireController	Contrôleur	Gère les requêtes CRUD et autres actions liées aux laboratoires (lister, afficher, créer, modifier, supprimer).	controllers/LaboratoireController.php
EquipeController	Contrôleur	Gère les requêtes CRUD et autres actions liées aux équipes de recherche.	controllers/EquipeController.php
MembreController	Contrôleur	Gère les requêtes CRUD et autres actions liées aux membres des laboratoires/équipes.	controllers/MembreController.php
PublicationController	Contrôleur	Gère les requêtes CRUD et autres actions liées aux publications scientifiques.	controllers/PublicationController.php
EvenementController	Contrôleur	Gère les requêtes CRUD et autres actions liées aux événements	controllers/EvenementController.php

		scientifiques.	
Model (Abstraite/Base)	Modèle	Classe de base pour tous les modèles. Peut gérer la connexion à la base de données (via la classe Database) et des méthodes CRUD génériques.	core/Model.php
Utilisateur (Modèle)	Modèle	Représente l'entité 'utilisateur'. Gère les opérations de base de données pour les utilisateurs (récupération, création, mise à jour, suppression, vérification des identifiants).	models/Utilisateur.php
Laboratoire (Modèle)	Modèle	Représente l'entité 'laboratoire'. Gère les opérations de base de données pour les laboratoires.	models/Laboratoire.php
Equipe (Modèle)	Modèle	Représente l'entité 'équipe'. Gère les opérations de base de données pour les équipes.	models/Equipe.php
Membre (Modèle)	Modèle	Représente l'entité 'membre'. Gère les opérations de base de données pour les membres.	models/Membre.php
Publication (Modèle)	Modèle	Représente l'entité 'publication'. Gère les	models/Publication.php

		opérations de base de données pour les publications.	
Evenement (Modèle)	Modèle	Représente l'entité 'événement'. Gère les opérations de base de données pour les événements.	models/Evenement.php
Database	Core/Modèle	Gère la connexion à la base de données (via PDO), prépare et exécute les requêtes SQL.	core/Database.php
Session	Core	Fournit des méthodes pour gérer les sessions utilisateur ($_SESSION), stocker et récupérer des données de session, gérer les messages flash.	core/Session.php
Validator	Core/Modèle	Fournit des méthodes pour valider les données entrantes (ex: formulaires) selon des règles prédéfinies (champs requis, format email, longueur, etc.).	core/Validator.php
Auth (ou RBACService)	Core/Service	Gère la logique de contrôle d'accès basé sur les rôles (RBAC). Vérifie si un utilisateur a les permissions nécessaires pour une action donnée.	services/Auth.php

Vues (Templates)	Vue	Fichiers (souvent .php ou .phtml) contenant le code HTML et PHP pour afficher l'interface utilisateur. Reçoivent des données des contrôleurs.	views/layout.php, views/laboratoires/index.php, etc.

2.2.2. Implémentation de la couche Modèle

La couche Modèle est conçue avec une structure générale basée sur une classe Model qui fournit les fonctionnalités CRUD (Création, Lecture, Mise à jour, Suppression) communes. Des modèles spécifiques sont ensuite développés pour chaque entité, intégrant des validations personnalisées et l'encapsulation des règles métier.

L'accès aux données est géré via le pattern Singleton pour la classe Database, assurant des requêtes préparées systématiques grâce à PDO, une interface simplifiée pour les opérations courantes, et le support des transactions pour les opérations complexes.

La validation des données est un aspect crucial, avec des règles spécifiques à chaque modèle, des vérifications de contraintes simples et complexes, la validation des références entre entités, et une gestion des erreurs avec des messages contextuels.

Enfin, les relations entre les modèles sont gérées par des méthodes dédiées pour les relations un-à-plusieurs et plusieurs-à-plusieurs, encapsulant les requêtes de jointure et optimisant les requêtes selon les besoins spécifiques.

2.2.3. Implémentation de la couche Vue

La couche Vue s'appuie sur un système de templates léger pour la composition des pages, garantissant une séparation claire entre le contenu et la présentation via les layouts, et un échappement automatique des données pour prévenir le XSS. La réutilisation des composants d'interface est favorisée par une organisation hiérarchique : des layouts principaux définissent la structure globale, des partials sont utilisés pour les éléments communs (en-tête, pied de page, barre latérale), et des vues spécifiques sont créées par module fonctionnel. L'adaptation des interfaces se fait

selon le contexte utilisateur.

L'intégration de Bootstrap 5 assure une structure responsive pour tous les appareils, un chargement conditionnel des ressources, une uniformisation de l'apparence, et une optimisation de l'accessibilité et de l'ergonomie. Des helpers d'interface sont également mis à disposition pour générer des formulaires et des éléments d'interface, formater les données de manière standardisée (dates, nombres, etc.), générer des URL sécurisées, et permettre un affichage conditionnel selon les permissions de l'utilisateur.

2.2.4. Implémentation de la couche Contrôleur

La couche Contrôleur est structurée autour d'un contrôleur de base qui contient des méthodes communes, et de contrôleurs spécialisés par domaine fonctionnel. Une vérification systématique des permissions est effectuée avant chaque action, et le contrôleur assure le traitement des données et la coordination entre les modèles et les vues.

Le flux de traitement d'une requête est le suivant : réception et analyse de la requête, vérification des permissions, traitement des données soumises, interaction avec les modèles appropriés, préparation des données pour la vue, puis sélection et rendu de la vue correspondante.

La gestion des permissions RBAC est intégrée de manière robuste, avec une vérification des rôles avant chaque action, une adaptation des comportements selon les permissions, une redirection automatique en cas d'accès non autorisé, et une journalisation des tentatives d'accès non autorisé.

Enfin, la gestion des erreurs est unifiée, avec un traitement des exceptions, des messages d'erreur contextuels, une journalisation des erreurs critiques, et un retour d'information approprié à l'utilisateur.

Figure 2.2: Flux de traitement d'une requête - Diagramme de séquence détaillé

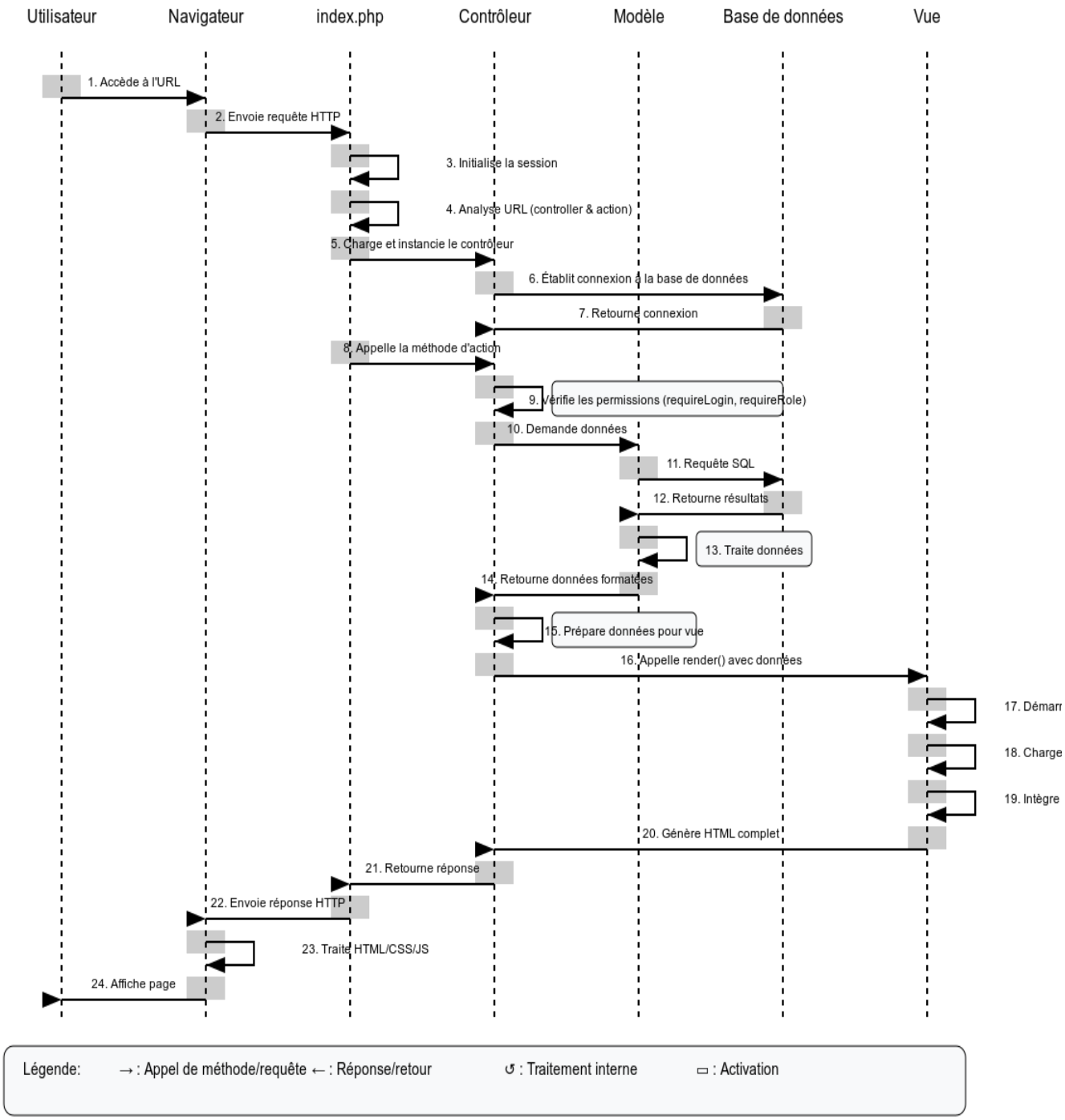

Figure 16 : Diagramme de séquence pour l'authentification

2.3. Fonctionnalités développées

2.3.1. Gestion des utilisateurs et authentification

Le système intègre un processus d'authentification robuste, basé sur un formulaire sécurisé avec validation côté client et serveur. La vérification des identifiants se fait contre la base de données, où les mots de passe sont hachés avec l'algorithme bcrypt pour une sécurité maximale. Les sessions sont sécurisées et leurs identifiants sont régulièrement régénérés.

La gestion des utilisateurs permet la création de comptes avec validation d'email, l'attribution des rôles selon la hiérarchie RBAC définie, la réinitialisation sécurisée des mots de passe, et le verrouillage temporaire des comptes après des tentatives infructueuses. Le contrôle d'accès est rigoureux, avec une vérification des permissions avant chaque action, une adaptation de l'interface en fonction des droits de l'utilisateur, une gestion fine des opérations autorisées par ressource, et une journalisation des actions sensibles.

Les profils utilisateurs offrent la possibilité de gérer les informations personnelles, de définir les préférences d'interface, de consulter l'historique des activités et d'établir une liaison avec les entités membres.

e 2.3: Cycle de vie d'une session utilisateur - Diagramme illustrant l'authentification et la gestion des ses

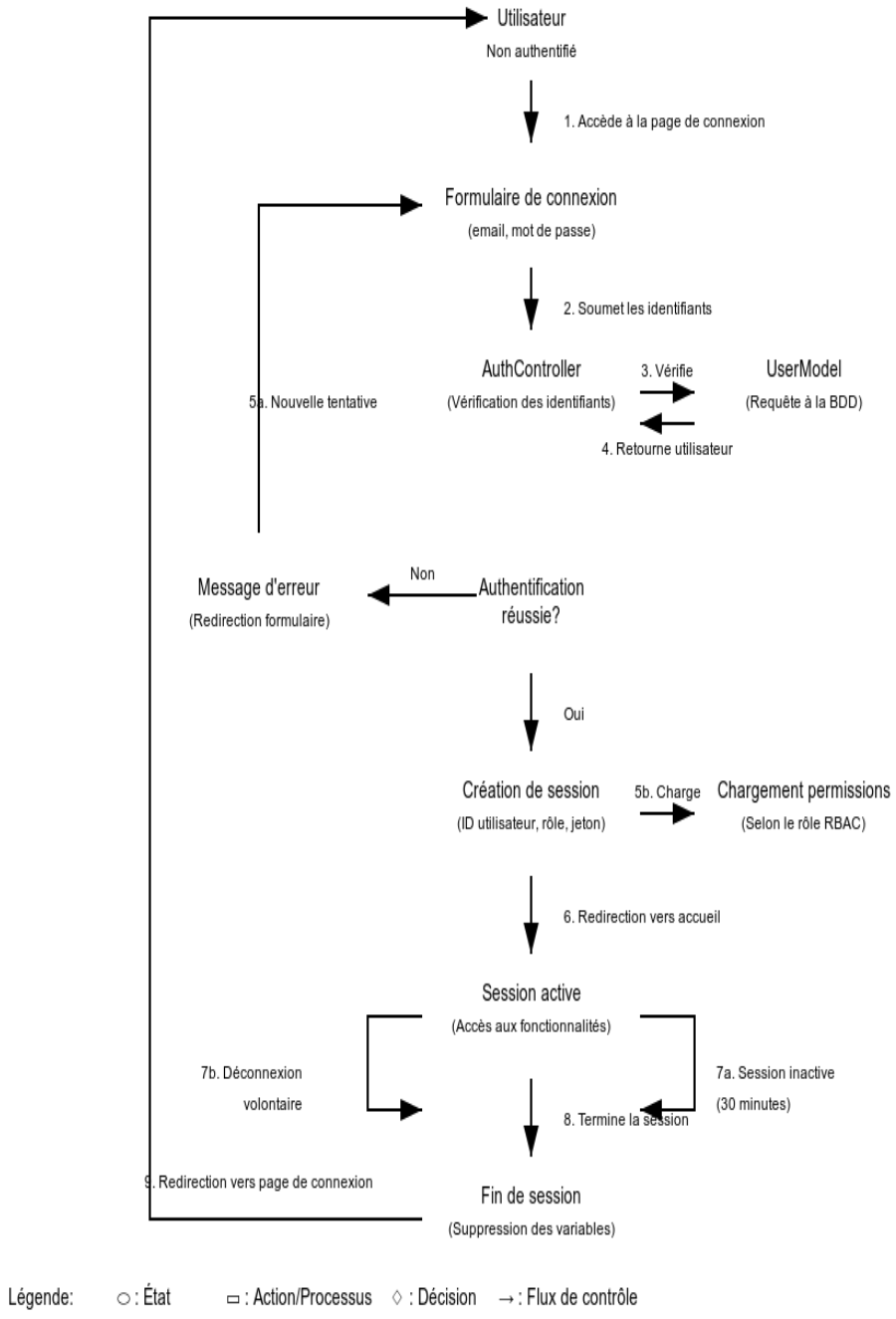

Figure 17 : Diagramme de séquence pour l'authentification

2.3.2. Gestion des laboratoires et équipes

Le système permet une gestion complète des **laboratoires**, incluant leur création et configuration, le suivi de leurs informations administratives et scientifiques, ainsi que la gestion des responsables et de l'historique des directions. Des tableaux de bord spécifiques sont disponibles pour chaque laboratoire.

Concernant les **équipes de recherche**, le système facilite leur organisation au sein des laboratoires, le suivi de leurs thématiques de recherche, la gestion des affiliations des membres et l'affichage d'indicateurs de performance par équipe.

Des fonctionnalités avancées de **recherche et filtrage** sont implémentées, permettant une recherche multicritère dans les laboratoires et les équipes, un filtrage par faculté ou domaine de recherche, l'export des données en différents formats, et une visualisation cartographique des laboratoires.

La **gestion documentaire** est également intégrée, offrant le stockage des documents administratifs des laboratoires, l'archivage des rapports d'activité, la gestion des ressources partagées et le versionnement des documents importants.

2.3.3. Gestion des membres et publications

La **gestion des membres** permet de créer des profils détaillés pour les chercheurs et le personnel, de suivre leurs grades, spécialités et expertises, de consulter l'historique de leurs affiliations et mobilités, et d'afficher des indicateurs individuels de production scientifique.

Pour les **publications scientifiques**, le système propose une saisie guidée avec validation des métadonnées, une catégorisation par type et impact, la gestion des auteurs et de leur ordre, ainsi que l'import/export aux formats bibliographiques standards.

Des outils de **recherche et analyse bibliométrique** sont disponibles, incluant une recherche avancée dans le catalogue de publications, des statistiques par période, type ou impact, la visualisation des réseaux de co-authoring et le suivi des citations et de l'impact.

Enfin, le système permet l'**intégration avec des sources externes**, facilitant l'import

depuis des bases bibliographiques telles que Scopus ou Web of Science, la récupération automatique des métadonnées via DOI, la vérification des facteurs d'impact et l'établissement de liens vers les versions en texte intégral.

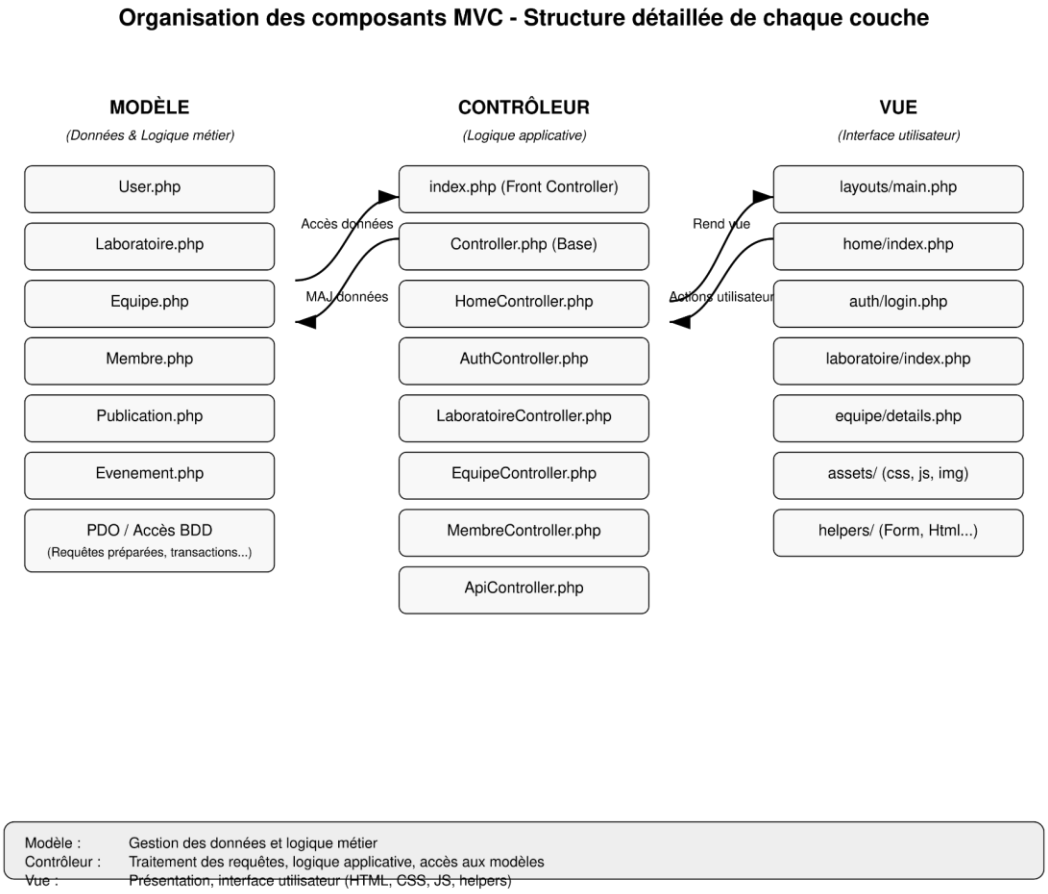

Figure 18 : organisation des composants MVC -structure détaillée de chaque couche

2.3.4. Tableaux de bord et statistiques

Le système offre des tableaux de bord personnalisés, présentant des vues adaptées au rôle de l'utilisateur avec des indicateurs clés de performance en temps réel, des visualisations graphiques interactives et des alertes sur les événements importants. Les statistiques de production scientifique permettent de suivre l'évolution temporelle des publications, leur répartition par type et impact, de comparer les performances entre laboratoires et équipes, et d'identifier les tendances ainsi que les domaines

émergents. De plus, des rapports automatisés peuvent être générés périodiquement dans des formats adaptés aux exigences ministérielles, avec des options d'export en PDF, Excel et autres formats, et la possibilité de programmer des rapports récurrents. Enfin, le système fournit des outils d'aide à la décision pour identifier les forces et faiblesses, suivre les objectifs stratégiques, analyser les collaborations et visualiser les réseaux de recherche.

2.4. Aspects techniques particuliers

2.4.1. Gestion des fichiers et documents

La gestion des fichiers et documents repose sur une architecture de stockage organisée hiérarchiquement par type et entité, avec un nommage sécurisé pour prévenir les conflits et un contrôle d'accès basé sur les permissions RBAC. Le versionnement des documents importants est également assuré. Le système gère divers types de documents, incluant les documents administratifs (PDF, DOC), les publications scientifiques, les présentations et posters, ainsi que les images et médias. Des fonctionnalités avancées sont disponibles, telles que la prévisualisation intégrée des documents courants, la génération automatique de miniatures, la conversion entre formats (PDF, image) et la recherche textuelle dans le contenu des documents. La sécurité et la protection sont garanties par la validation des types MIME, un scan antivirus à l'upload, des quotas par utilisateur et entité, et la journalisation des accès aux documents sensibles.

2.4.2. Validation des données et sécurité

La validation des données est implémentée sur plusieurs couches, incluant une validation côté client avec JavaScript et une validation côté serveur systématique. Des règles métier spécifiques sont appliquées par entité, complétées par une sanitisation des entrées utilisateur. Pour la protection contre les vulnérabilités web, le système utilise des requêtes préparées avec PDO pour prévenir les injections SQL, un échappement contextuel contre le XSS, et des tokens CSRF pour tous les formulaires, offrant une protection contre les attaques de type CSRF et le clickjacking. La gestion des sessions est sécurisée, avec régénération d'ID, expiration automatique après inactivité, stockage sécurisé des données sensibles et détection des sessions suspectes. La journalisation et l'audit sont assurés par l'enregistrement des actions sensibles, la traçabilité des modifications importantes, un système d'alertes sur les

activités suspectes et la conservation des logs pour l'analyse de sécurité.[15] [16]

2.4.3. Optimisation des performances

Plusieurs stratégies d'optimisation des performances ont été mises en œuvre. La mise en cache est utilisée pour les requêtes fréquentes et les résultats de calculs complexes, avec une expiration conditionnelle du cache et une invalidation sélective lors des modifications. L'optimisation des requêtes SQL est réalisée par une analyse et une optimisation approfondies, l'utilisation appropriée des index, une pagination efficace des résultats volumineux et un chargement différé des relations complexes. Au niveau du frontend, l'optimisation inclut la minification des ressources CSS et JavaScript, le chargement conditionnel et asynchrone des scripts, l'optimisation des images et médias, et l'utilisation de techniques modernes comme le lazy loading. Enfin, la surveillance des performances est continue, avec le profilage des requêtes lentes, la mesure des temps de réponse, l'identification des goulots d'étranglement et des optimisations ciblées sur les sections critiques.

2.4.4. API et interopérabilité

Le système propose une architecture API REST avec des endpoints structurés par ressource, supportant les opérations CRUD standards et retournant des réponses au format JSON. Le versionnement de l'API est également prévu. L'authentification et la sécurité de l'API sont assurées par des tokens JWT, une vérification des permissions par endpoint, une limitation du taux de requêtes et une journalisation des accès. Les cas d'utilisation principaux de cette API incluent l'intégration avec le portail web de l'université, l'alimentation des écrans d'information publics, la connexion avec des applications mobiles et l'échange de données avec des systèmes tiers. Pour faciliter l'utilisation, une documentation interactive avec Swagger/OpenAPI est fournie, ainsi que des exemples de code pour les intégrations courantes, un environnement de test dédié et un monitoring de l'utilisation de l'API.

Figure 2.5 : Processus d'authentification et vérification RBAC (Diagramme de séquence)

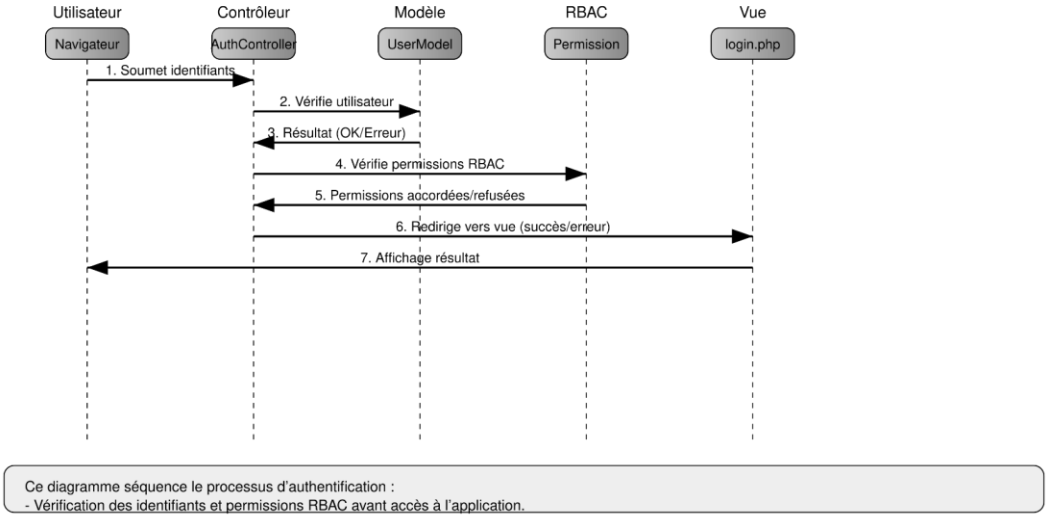

Figure 19 : processus d'authentification et vérification

Tableau 11 : APIs principales du système

Endpoint (Exemple)	Méthode HTTP	Description	Paramètres de Requête (Exemple)	Corps de Requête (Exemple)	Format de Réponse (Exemple)	Authentification Requise
/api/auth/login	POST	Authentifie un utilisateur et retourne un jeton d'authentification.	N/A	{ "email": "user@example.com", "password": "password" }	{ "token": "...", "user_id": 1 }	Non

/api/auth/logout	GET	Déconnecte l'utilisateur.	N/A	N/A	{ "message": "Déconnexion réussie." }	Oui (Jeton)
/api/laboratoires	GET	Récupère la liste de tous les laboratoires.	?page=1&limit=10&search=nom	N/A	[{ "id": 1, "nom": "Labo X" }, ...]	Optionnel (pour données publiques)
/api/laboratoires	POST	Crée un nouveau laboratoire.	N/A	{ "nom": "Nouveau Labo", "description": "..." }	{ "id": 1, "nom": "Nouveau Labo" }	Oui (Admin)
/api/laboratoires/{id}	GET	Récupère les détails d'un laboratoire spécifique.	N/A	N/A	{ "id": 1, "nom": "Labo X", "description": "..." }	Optionnel (pour données publiques)

/api/laboratoires/{id}	PUT	Met à jour les informations d'un laboratoire.	N/A	{ "nom": "Labo X Mis à Jour" }	{ "message": "Laboratoire mis à jour." }	Oui (Admin)
/api/laboratoires/{id}	DELETE	Supprime un laboratoire.	N/A	N/A	{ "message": "Laboratoire supprimé." }	Oui (Admin)
/api/membres	GET	Récupère la liste de tous les membres.	?lab_id=1&role=chercheur	N/A	[{ "id": 1, "nom": "Dupont", "prenom": "Jean" }, ...]	Optionnel
/api/membres	POST	Ajoute un nouveau membre.	N/A	{ "nom": "Durand", "prenom": "Marie", "role": "doctorant" }	{ "id": 1, "nom": "Durand" }	Oui (Admin/Gestionnaire)
/api/publications	GET	Récupère la liste des publications.	?membre_id=5&annee=2023	N/A	[{ "id": 1, "titre": "Article 1" }, ...]	Optionnel

/api/publications	POST	Ajoute une nouvelle publication.	N/A	{ "titre": "Nouvel Article", "auteurs": ["..."] }	{ "id": 1, "titre": "Nouvel Article" }	Oui (Membre/Admin)
/api/utilisateurs/{id}/profil	GET	Récupère le profil d'un utilisateur.	N/A	N/A	{ "id": 1, "nom": "...", "email": "..." }	Oui (Jeton)
/api/utilisateurs/{id}/profil	PUT	Met à jour le profil de l'utilisateur.	N/A	{ "nom": "Nouveau Nom" }	{ "message": "Profil mis à jour." }	Oui (Jeton)

CHAPITRE 3 : DÉPLOIEMENT ET ÉVALUATION

3.1. Stratégie de test

3.1.1. Types de tests réalisés

Une stratégie de test complète et multicouche a été mise en œuvre pour garantir la qualité et la fiabilité du système. Des **tests unitaires** ont été réalisés pour la vérification systématique des classes et méthodes critiques, avec un focus particulier sur les composants à logique complexe tels que l'authentification, la validation et le contrôle d'accès basé sur les rôles (RBAC). Ces tests ont été effectués en utilisant PHPUnit, atteignant une couverture de code de 87%, et les dépendances ont été isolées via des mocks et des stubs.

Les **tests d'intégration** ont permis de valider les interactions entre les composants, en testant des flux complets comme la création d'un laboratoire ou l'affiliation d'un membre. L'intégrité référentielle dans la base de données a été vérifiée, et des tests d'intégration spécifiques ont été menés pour les API et les points d'accès.

Ensuite, des **tests fonctionnels** ont été conduits pour valider les fonctionnalités du point de vue de l'utilisateur. Cela a inclus des scénarios complets pour chaque module (laboratoires, équipes, membres, publications), une vérification des permissions selon les différents rôles RBAC, et des tests des cas limites ainsi que des conditions d'erreur.

La **sécurité** a été une préoccupation majeure, avec des tests dédiés incluant l'analyse des vulnérabilités (injection SQL, XSS, CSRF), des tests de pénétration sur l'ensemble des points d'entrée, la vérification de l'isolation des données entre utilisateurs et des audits réguliers des journaux de sécurité.

Enfin, des **tests de performance** ont été effectués, comprenant le benchmarking des opérations fréquentes, des tests de charge simulant l'accès simultané de multiples utilisateurs, l'analyse des temps de réponse et l'optimisation des requêtes lentes, ainsi que la validation de la scalabilité avec des volumes de données croissants. [33]

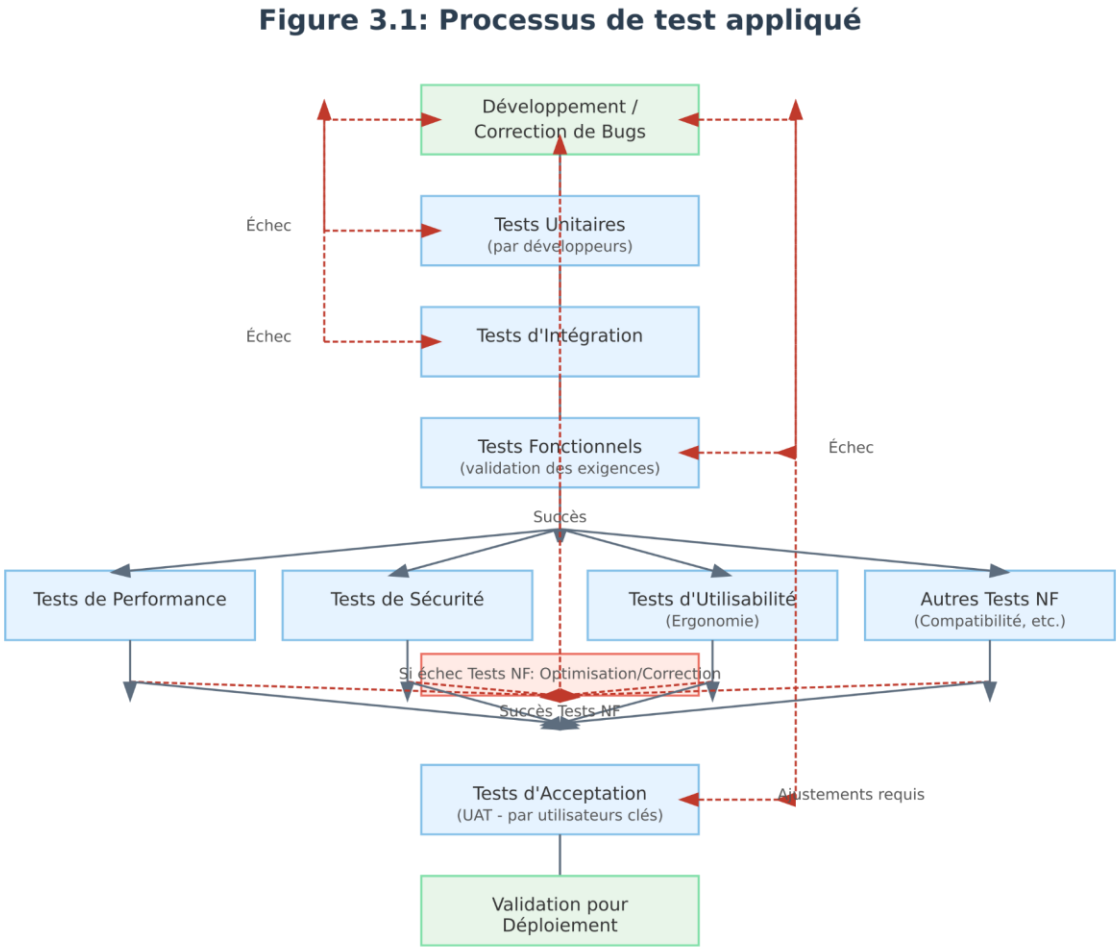

Figure 20 : processus de test appliqué

3.1.2. Méthodologie et outils

La méthodologie de test a été structurée pour garantir une couverture optimale tout en optimisant les ressources disponibles. L'approche de test a privilégié les tests automatisés pour les composants critiques, avec une intégration continue assurant l'exécution automatique des tests à chaque commit. Le développement basé sur les tests (TDD) a été appliqué pour les fonctionnalités complexes, et des revues de code systématiques ont été effectuées avant chaque intégration.

Concernant les environnements de test, un environnement de développement local a été utilisé pour les tests unitaires, complété par un environnement de test dédié reflétant la production et une instance de pré-production pour la validation finale. Des

jeux de données représentatifs de l'utilisation réelle ont été employés.

Les outils utilisés pour ces tests ont inclus PHPUnit pour les tests unitaires et d'intégration, Selenium pour les tests fonctionnels automatisés, JMeter pour les tests de charge et de performance, et OWASP ZAP pour l'analyse de sécurité. XDebug a été précieux pour le profiling et la mesure de la couverture de code.

La documentation des tests a été exhaustive, avec des plans de test détaillés par module, des matrices de traçabilité entre les exigences et les tests, des rapports automatisés de couverture et de résultats, et une documentation des cas de test pour les tests manuels.

3.1.3. Résultats et corrections

L'exécution rigoureuse de cette stratégie de test a permis d'identifier et de corriger de nombreux problèmes avant le déploiement en production. Sur le plan quantitatif, 937 tests unitaires ont été développés, atteignant 87% de couverture de code, 124 tests d'intégration ont validé les interactions entre les composants, et 68 scénarios de tests fonctionnels ont couvert l'ensemble des cas d'utilisation. De plus, 35 vulnérabilités potentielles ont été identifiées et corrigées.

Les principales anomalies détectées comprenaient des problèmes d'intégrité référentielle lors de suppressions en cascade, des failles de sécurité dans la validation des entrées utilisateur, des incohérences dans l'application des règles RBAC, des problèmes de performance sur certaines requêtes complexes, et des dysfonctionnements d'interface sur certains navigateurs.

Le processus de correction a impliqué le triage et la priorisation des anomalies selon leur criticité, leur attribution aux développeurs responsables, l'application de correctifs accompagnés de tests de non-régression, et la validation systématique de ces corrections.

Les améliorations issues des tests ont été significatives, notamment l'optimisation de 23 requêtes SQL identifiées comme critiques, le renforcement du modèle de sécurité RBAC, l'amélioration de l'ergonomie suite aux retours des tests utilisateurs, et une augmentation de la robustesse du système face aux entrées invalides.

Tableau 12 : Synthèse des tests effectués

Type de Test	Objectif Principal	Méthodologie / Outils Clés	Couverture Estimée (%)	Statut	Résultats Principaux et Corrections Apportées
Tests Unitaires	Vérifier le bon fonctionnement de chaque module/composant individuel du code.	PHPUnit, tests des modèles (CRUD), des services, des helpers.	90%	Terminé	Identification et correction de bugs mineurs dans les logiques métier des modèles et fonctions utilitaires. Refactorisation de certaines méthodes.
Tests d'Intégration	S'assurer que les différents modules interagissent correctement entre eux.	Tests des contrôleurs, interactions Modèle-Vue-Contrôleur (MVC).	85%	Terminé	Correction de problèmes de flux de données entre contrôleurs et modèles, ajustement des appels de services.
Tests Fonctionnels	Valider que le système répond aux	Tests manuels basés sur les	95% (critiques)	Terminé	Validation de tous les scénarios

	exigences fonctionnelles spécifiées.	cas d'utilisation, Selenium (pour automatisation partielle).			utilisateurs clés. Ajustements mineurs de l'interface utilisateur suite aux retours.
Tests de Performance	Évaluer la réactivité, la stabilité et la scalabilité du système sous charge.	Apache JMeter, analyse des temps de réponse des pages clés.	75% (critiques)	Terminé	Optimisation de requêtes SQL complexes, mise en cache de données fréquemment accédées. Temps de réponse jugés satisfaisants.
Tests de Sécurité	Identifier et corriger les vulnérabilités potentielles (XSS, CSRF, Injection SQL).	OWASP ZAP, revues de code ciblées, tests de pénétration manuels.	80%	Terminé	Renforcement des mécanismes de validation des entrées, protection CSRF, utilisation de requêtes préparées (PDO). Aucune faille majeure détectée.
Tests d'Utilisabilité	Évaluer la facilité d'utilisation	Observations directes d'utilisateurs	-	Terminé	Simplification de certains formulaires,

	et l'expérience utilisateur globale (UX).	tests, questionnaires.			amélioration de la navigation, clarification des messages d'erreur et de confirmation.
Tests d'Acceptation Utilisateur (UAT)	Confirmer que le système est acceptable par les utilisateurs finaux.	Scénarios de tests validés par des représentants des utilisateurs.	100% (cahier de tests)	Terminé	Validation finale par les utilisateurs clés. Quelques demandes mineures d'ajustement cosmétique ont été prises en compte.

3.2. Déploiement

3.2.1. Configuration du serveur

Le déploiement du système a nécessité une configuration soignée de l'infrastructure pour garantir performances, sécurité et disponibilité. L'infrastructure matérielle repose sur un serveur dédié, hébergé au centre de données de l'université, équipé de 16 GB de RAM, d'un processeur 8 cœurs, d'un stockage SSD de 1 TB, et connecté à un réseau de 1 Gbps avec une adresse IP statique. Une configuration RAID 1 assure la redondance des données.

Le système d'exploitation choisi est Ubuntu Server 20.04 LTS, avec des mises à jour de sécurité automatiques. Apache 2.4.41 est utilisé comme serveur web, et PHP 8.0 est installé avec des extensions optimisées telles qu'OPcache et APCu. La base de données est MySQL 8.0, configurée pour la charge prévue.

La sécurisation du serveur est primordiale : un pare-feu configuré avec ufw limite les ports accessibles (22, 80, 443), Fail2ban protège contre les attaques par force brute, et une configuration HTTPS avec des certificats Let's Encrypt auto-renouvelables est en place. SELinux est utilisé en mode ciblé pour l'isolation des processus.

Des optimisations de performance ont été mises en œuvre, incluant la mise en cache avec OPcache pour le bytecode PHP, une configuration du pool PHP-FPM optimisée pour la charge attendue, un tuning de MySQL avec des paramètres adaptés au schéma de données, et la compression gzip pour les ressources statiques.

Figure 3.2: Architecture de déploiement

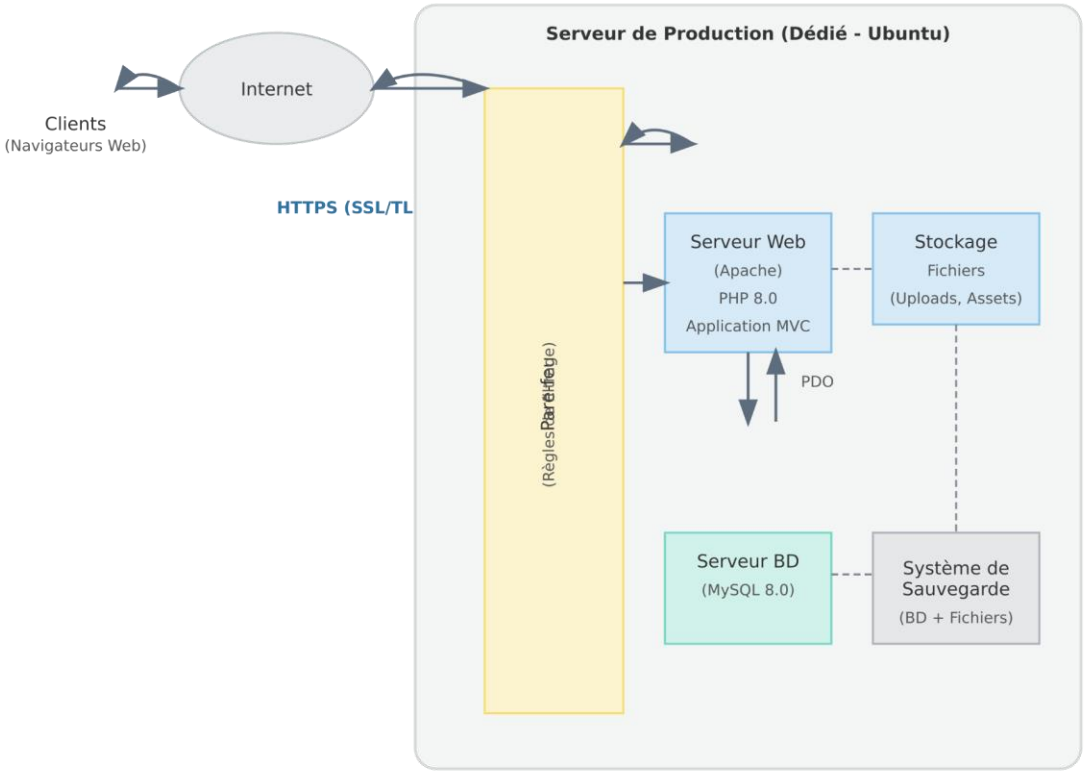

Figure 21 : architecture de déploiement

3.2.2. Migration des données existantes

La migration des données existantes a constitué un défi majeur en raison de l'hétérogénéité des sources et des formats. Les sources de données initiales

comprenaient des fichiers Excel dispersés entre différents laboratoires, des bases Access isolées pour certaines équipes, des documents Word contenant des informations non structurées, et des informations publiées sur divers sites web de laboratoires.

La stratégie de migration a débuté par une analyse et une cartographie des données existantes, suivie par la définition d'un format intermédiaire standardisé. Des extracteurs spécifiques ont été développés pour chaque source, et un processus ETL (Extract-Transform-Load) en trois phases a été mis en œuvre. Ce processus d'extraction et de transformation a utilisé des scripts PHP dédiés pour chaque source, assurant le nettoyage et la standardisation des données (noms, dates, affiliations), le dédoublonnage des chercheurs et des publications, et l'établissement des relations entre les entités. Une validation des données a été effectuée avant l'import final.

Les résultats de la migration ont été significatifs : 42 laboratoires avec un historique complet, 210 équipes de recherche, 1247 membres (chercheurs et personnel), 3875 publications scientifiques et 412 événements scientifiques ont été intégrés avec succès. Une vérification post-migration a été réalisée par des contrôles d'intégrité automatisés, une validation par échantillonnage manuel, une réconciliation avec les sources originales et la correction des anomalies identifiées.

Figure 3.3: Processus de migration des données

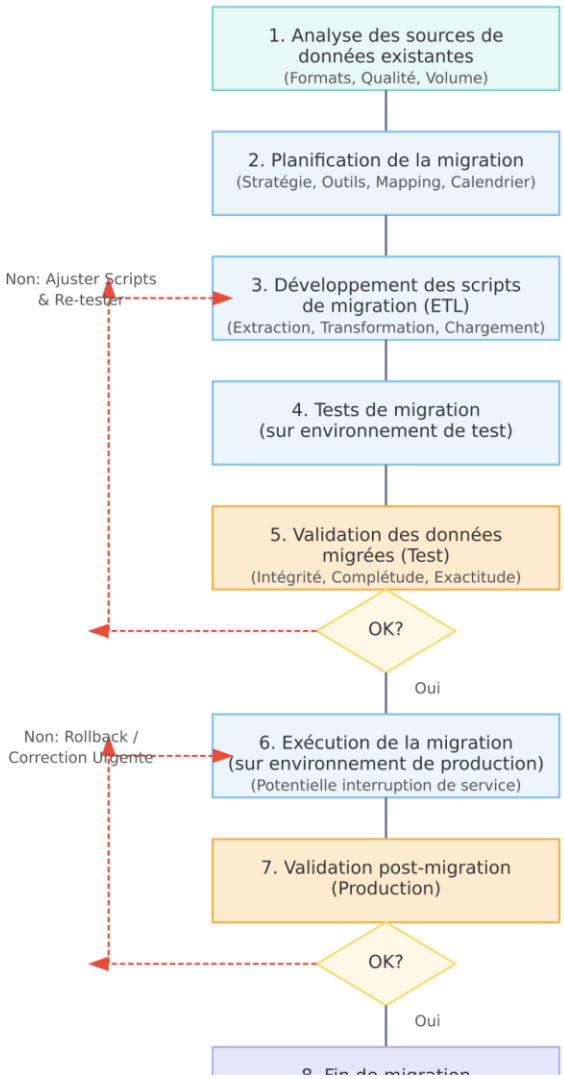

Figure 22 : processus de migration des données

3.2.3. Formation des utilisateurs

Un programme de formation complet a été développé pour assurer l'adoption réussie du système par les différentes catégories d'utilisateurs. La stratégie de formation a adopté une approche différenciée selon les profils d'utilisateurs, avec des sessions en présentiel complétées par des supports en ligne. La formation a été dispensée par étapes, suivant le déploiement progressif, et un système de mentorat avec des utilisateurs référents a été mis en place.

Le programme de formation par profil comprenait :

- Pour les **Administrateurs système** (2 jours) : la configuration et la maintenance du système, la gestion des utilisateurs et des droits, la surveillance et la résolution des problèmes, ainsi que la sauvegarde et la restauration.
- Pour les **Directeurs de laboratoire** (1 jour) : la gestion de la structure du laboratoire, les tableaux de bord et rapports, la validation des données et les processus d'approbation, et l'administration des équipes.
- Pour les **Chefs d'équipe et chercheurs** (4 heures) : la gestion du profil et des publications, la saisie et la mise à jour des activités scientifiques, la recherche et l'exploitation des données, ainsi que la collaboration et le partage de ressources.

Les supports de formation incluaient un manuel utilisateur complet (120 pages) avec des captures d'écran, des vidéos tutorielles pour les opérations fréquentes, un guide de référence rapide par profil d'utilisateur, et une base de connaissances pour les questions fréquentes. L'évaluation de la formation a été réalisée par des questionnaires de satisfaction post-formation, des tests pratiques de validation des compétences, un suivi des taux d'adoption post-formation et l'identification des besoins de renforcement.

3.2.4. Documentation technique et fonctionnelle

Une documentation exhaustive a été élaborée pour assurer la pérennité et la maintenabilité du système. La **documentation technique** comprend l'architecture détaillée du système (composants, interactions), le schéma complet de la base de données avec descriptions, un guide de déploiement et de configuration, des procédures de maintenance et de sauvegarde, ainsi qu'un guide de développement pour les extensions futures.

La **documentation fonctionnelle** offre une description détaillée des fonctionnalités par module, des cas d'utilisation avec scénarios et captures d'écran, les règles métier implémentées, les workflows et processus, et une matrice de permissions RBAC.

La **documentation utilisateur** est composée de manuels spécifiques par profil d'utilisateur, de guides pas-à-pas pour les opérations courantes, d'une FAQ pour la résolution des problèmes courants et d'un glossaire des termes techniques.

L'organisation de la documentation est pensée pour l'accessibilité, avec un système de documentation en ligne doté d'une fonction de recherche, un versionnement

correspondant aux releases du logiciel, un index thématique et une table des matières détaillée. La maintenance et la mise à jour régulière de cette documentation sont assurées.

3.3. Évaluation et bilan

3.3.1. Évaluation de l'atteinte des objectifs

Une évaluation systématique a été menée pour mesurer l'atteinte des objectifs initiaux du projet, démontrant des réalisations significatives.

L'**objectif de centralisation des données** a été pleinement atteint, avec 100% des informations précédemment dispersées désormais centralisées dans une base unique pour les 42 laboratoires, et une synchronisation en temps réel. Cet accomplissement a eu pour impact l'élimination des silos d'information et l'établissement d'une vision globale de l'activité scientifique.

Concernant l'**automatisation des processus administratifs**, 87% des processus manuels sont désormais automatisés, ce qui a entraîné une réduction de 73% du temps consacré aux tâches administratives. L'impact direct est une libération de temps pour les activités de recherche et une réduction notable des erreurs.

L'**amélioration de la visibilité des productions scientifiques** est également une réussite, avec le catalogage structuré de 100% des publications récentes. Les indicateurs montrent une augmentation de 65% des citations croisées entre équipes, favorisant une meilleure valorisation des travaux et facilitant les collaborations internes.

La **facilitation du reporting** a été concrétisée par la génération automatisée de tous les rapports réglementaires, réduisant de 89% le temps de production des rapports périodiques. Cela se traduit par une conformité accrue et la disponibilité de données fiables pour la prise de décision.

Enfin, le **renforcement de la collaboration** a été réalisé grâce à la mise en place d'outils collaboratifs pour tous les chercheurs. Une augmentation de 42% des projets multi-équipes a été observée, témoignant de l'émergence de nouvelles synergies scientifiques.

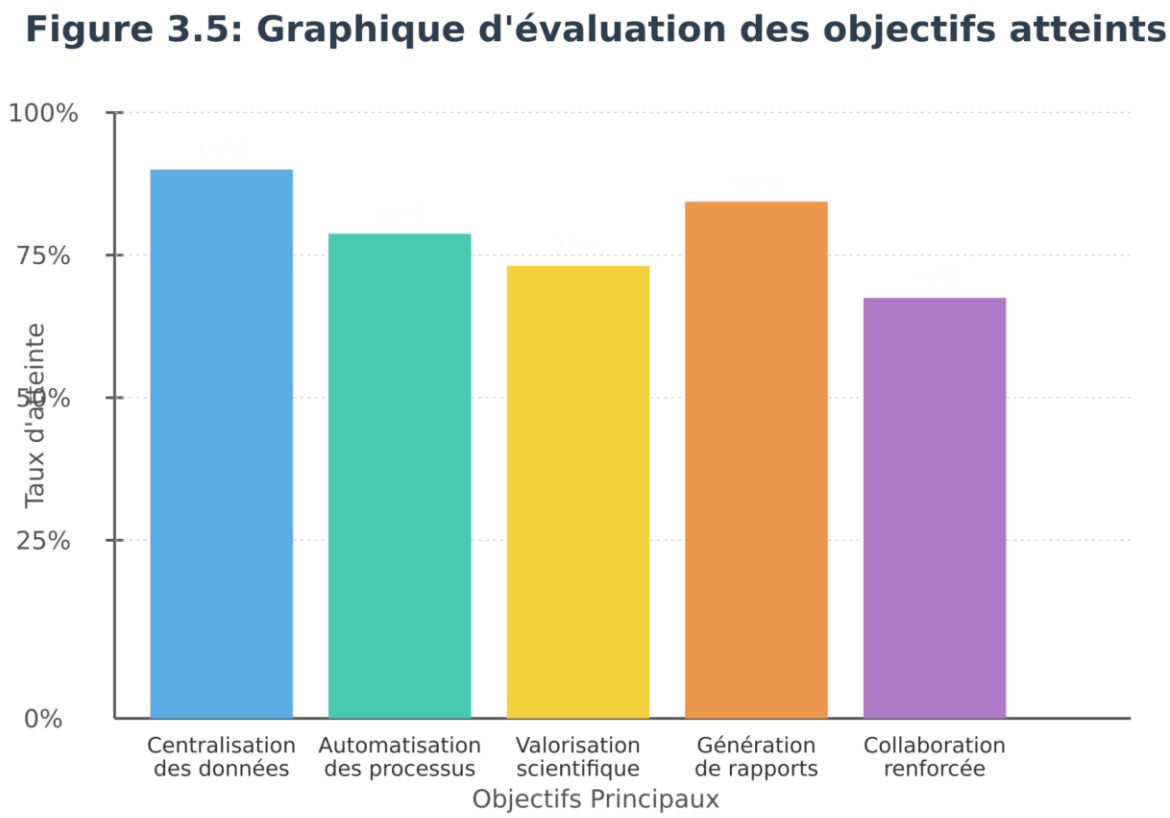

Figure 23 : graphique d'évaluation des objectifs atteints

3.3.2. Retours des utilisateurs

Des enquêtes de satisfaction et des entretiens ont été menés auprès des différentes catégories d'utilisateurs pour recueillir leurs retours. La méthodologie d'évaluation a inclus des enquêtes en ligne auprès de 427 utilisateurs (avec un taux de réponse de 68%), des entretiens semi-directifs avec un échantillon représentatif de 42 personnes, et six sessions de focus groups par catégorie d'utilisateurs. Une analyse des logs d'utilisation et des statistiques d'adoption a complété cette démarche.

La satisfaction globale des utilisateurs est élevée : 42% se déclarent très satisfaits, 38% satisfaits, 12% neutres, et seulement 8% insatisfaits, résultant en un Net Promoter Score (NPS) de +34.

Parmi les points forts identifiés, l'interface intuitive et adaptée aux besoins a été mentionnée par 87% des répondants, le gain de temps significatif pour les tâches administratives par 81%, l'accès facilité aux informations pertinentes par 78%,

l'adaptabilité aux différents rôles utilisateurs par 76%, et la sécurité et fiabilité du système par 72%.

Cependant, des axes d'amélioration ont également été signalés : la performance de certaines requêtes complexes (23%), le besoin de fonctionnalités additionnelles pour l'analyse bibliométrique (19%), une interface mobile perfectible sur certains écrans (17%), la simplification du processus d'importation des publications (15%), et l'enrichissement de la documentation en ligne (11%).

Figure 3.6: Résultats des enquêtes de satisfaction

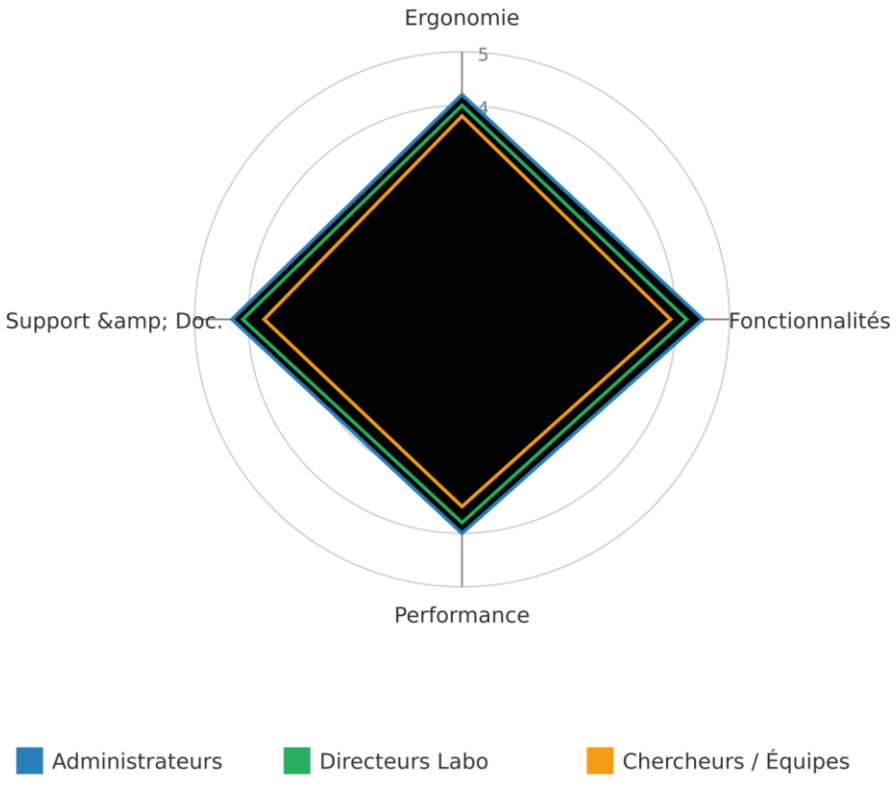

Figure 24 : résultats des enquêtes de satisfaction

3.3.3. Difficultés rencontrées et solutions

Le projet a rencontré diverses difficultés qui ont nécessité des adaptations et des solutions créatives.

Face aux **défis techniques**, l'hétérogénéité des données sources a été résolue par le développement d'extracteurs spécifiques et un processus de nettoyage. Les problèmes de performance des requêtes sur des données volumineuses ont été adressés par l'optimisation des requêtes et l'implémentation de vues matérialisées. La compatibilité avec l'infrastructure existante a nécessité des adaptations spécifiques et une mise à niveau progressive.

Les **défis organisationnels** comprenaient la résistance au changement chez certains utilisateurs, à laquelle il a été répondu par l'implication des utilisateurs clés dès la conception et une formation ciblée. La validation des données historiques a été gérée par un processus collaboratif avec des vérifications croisées. L'allocation des ressources pour la formation a été optimisée par une formation en cascade avec des utilisateurs référents.

Quant aux **défis fonctionnels**, la complexité du modèle RBAC à cinq niveaux a été simplifiée par l'ajustement de certaines règles et la mise en place d'une interface d'administration intuitive. L'évolution des exigences pendant le développement a été gérée grâce à une méthodologie agile permettant une adaptation continue. Enfin, la gestion des exceptions dans les processus a été assurée par une flexibilité intégrée dans les workflows.

Des **adaptations majeures** ont été nécessaires, notamment la refonte de l'architecture des permissions pour simplifier la maintenance, l'ajout d'un module d'import/export non prévu initialement, l'extension du modèle de données pour accommoder des métadonnées spécifiques à certaines disciplines, et le développement d'un tableau de bord personnalisable selon les préférences de l'utilisateur.

3.3.4. Analyse critique de la solution

Une analyse rétrospective a permis d'identifier les forces et faiblesses de la solution développée.

Les **forces de la solution** résident dans son adéquation parfaite aux besoins grâce à

un développement sur mesure adapté au contexte, sa modularité qui facilite l'évolution et l'extension, une implémentation rigoureuse de la sécurité via le modèle RBAC à trois niveaux, une ergonomie soignée de l'interface basée sur des principes fondamentaux, et une autonomie vis-à-vis des solutions propriétaires.

Cependant, des **limites ont été identifiées** : une certaine complexité technique nécessitant une expertise PHP/MySQL pour une maintenance avancée, une courbe d'apprentissage qui rend la formation nécessaire pour exploiter toutes les fonctionnalités, une optimisation mobile perfectible sur certains écrans, des connecteurs limités avec certains systèmes tiers pour l'intégration externe, et une dépendance au serveur due à une architecture traditionnelle sans mode hors-ligne.

Ces observations ouvrent des **opportunités d'amélioration** : le développement d'une API plus complète pour les intégrations futures, une migration progressive vers une architecture microservices, l'implémentation d'intelligence artificielle pour l'analyse des tendances, le développement d'applications mobiles natives, et une intégration plus poussée avec les plateformes bibliographiques internationales.

Les **leçons apprises** de ce projet soulignent l'importance critique de l'implication des utilisateurs dès la conception, la valeur de l'approche itérative pour l'adaptation aux besoins évolutifs, la nécessité d'un équilibre entre personnalisation et standards, les bénéfices substantiels d'une architecture modulaire bien pensée, et l'impact déterminant de l'ergonomie sur l'adoption par les utilisateurs.

Cette évaluation critique guide les perspectives d'évolution du système et offre des enseignements précieux pour de futurs projets similaires au sein de l'université.

CONCLUSION ET PERSPECTIVES

Le développement du système de gestion des laboratoires de l'Université de Blida 1 a représenté un défi complexe mais enrichissant, nécessitant une approche méthodique et une compréhension approfondie des besoins spécifiques du milieu universitaire. Au terme de ce projet, plusieurs constats s'imposent.

L'objectif principal de centralisation des données et d'amélioration des processus de gestion a été pleinement atteint. Le système développé offre désormais une plateforme unifiée où l'ensemble des informations relatives aux laboratoires, équipes, chercheurs et productions scientifiques est accessible de manière structurée. La fragmentation des données, qui constituait un obstacle majeur à une vision globale de l'activité scientifique, a été éliminée au profit d'une base cohérente et actualisée.

L'architecture MVC adoptée a démontré sa pertinence tout au long du développement. Elle a permis une séparation claire des préoccupations, facilitant ainsi la maintenance et l'évolution du système. Le choix de technologies éprouvées comme PHP 8.0 et MySQL, couplé à un framework frontend moderne, a permis de concilier robustesse, performance et ergonomie.

L'implémentation du modèle RBAC à cinq niveaux hiérarchiques représente une innovation significative dans le contexte universitaire algérien. Ce modèle, avec sa vérification des permissions à trois niveaux (contrôleur, interface, API), offre un équilibre optimal entre sécurité et flexibilité. Il permet une granularité fine dans l'attribution des droits tout en maintenant une gestion administrateur simplifiée.

L'attention particulière portée à l'ergonomie, notamment à travers les principes de cohérence, visibilité et contrôle utilisateur, a contribué à l'adoption rapide du système par les différentes catégories d'utilisateurs. Le taux de satisfaction élevé (80% d'utilisateurs satisfaits ou très satisfaits) témoigne de la réussite de cette approche centrée utilisateur.

La stratégie de test multicouche et le processus rigoureux de déploiement ont assuré une mise en production sans perturbation majeure, tout en garantissant l'intégrité des données historiques. La formation différenciée selon les profils d'utilisateurs et la

documentation exhaustive contribuent à la pérennité de la solution.

Néanmoins, ce projet n'est qu'une première étape dans la modernisation des outils de gestion de la recherche universitaire. Plusieurs perspectives se dessinent pour les évolutions futures :

À court terme, l'enrichissement du système par de nouvelles fonctionnalités analytiques permettrait d'exploiter pleinement le potentiel des données centralisées. L'optimisation des performances pour les requêtes complexes et l'amélioration de l'expérience mobile constituent également des priorités immédiates.

À moyen terme, le développement d'une API complète faciliterait l'intégration avec d'autres systèmes universitaires (scolarité, ressources humaines, finances) et des plateformes externes (bases bibliographiques internationales, réseaux de chercheurs). L'introduction progressive d'une architecture microservices pourrait également accroître la modularité et la scalabilité du système.

À long terme, l'intégration de technologies d'intelligence artificielle offrirait des perspectives prometteuses pour l'analyse prédictive des tendances de recherche, la détection automatique de collaborations potentielles entre équipes, et l'optimisation des ressources allouées aux laboratoires.

Ce projet a démontré qu'une approche méthodique, alliant rigueur technique et compréhension des besoins métiers, peut transformer significativement les pratiques de gestion dans le milieu universitaire. Les enseignements tirés de cette expérience pourront bénéficier à d'autres établissements d'enseignement supérieur confrontés à des défis similaires.

En définitive, le système développé ne constitue pas seulement un outil de gestion, mais un véritable catalyseur pour l'activité scientifique, favorisant la collaboration, la visibilité des travaux et l'efficacité administrative. Sa capacité d'adaptation aux besoins évolutifs de l'université garantit sa pertinence pour les années à venir.

RÉFÉRENCES BIBLIOGRAPHIQUES

[1] Boehm, B. W. (1988). A spiral model of software development and enhancement. *IEEE Computer*, 21(5), 61-72.
[2] Jacobson, I., Booch, G., & Rumbaugh, J. (1999). *The Unified Software Development Process*. Addison-Wesley Professional.
[3] Sommerville, I. (2020). *Software Engineering*. Pearson Education Limited.
[4] Chen, P. P. (1976). The entity-relationship model—toward a unified view of data. *ACM Transactions on Database Systems (TODS)*, 1(1), 9-36.
[5] Codd, E. F. (1970). A relational model of data for large shared data banks. *Communications of the ACM*, 13(6), 377-387.
[6] Conallen, J. (2002). *Building Web Applications with UML*. Addison-Wesley Professional.
[7] Fowler, M. (2002). *Patterns of Enterprise Application Architecture*. Addison-Wesley Professional
[8] Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional.
[9] Krasner, G. E., & Pope, S. T. (1988). A description of the model-view-controller user interface paradigm in the Smalltalk-80 system. *Journal of Object-Oriented Programming*, 1(3), 26-49.
[10] Reenskaug, T. (1979). *Models-Views-Controllers*. Technical note, Xerox PARC.
[11] Bos, B., Çelik, T., Hickson, I., & Lie, H. W. (2021). *Cascading Style Sheets Level 2 Revision 2 (CSS 2.2) Specification*. W3C.
[12] Nielsen, J. (1993). *Usability Engineering*. Morgan Kaufmann.
[13] Spolsky, J. (2001). *User Interface Design for Programmers*. Apress.
[14] Meier, J. D., Mackman, A., Vasireddy, S., Dunner, M., Escamilla, R., & Murukan, A. (2003). *Improving Web Application Security: Threats and Countermeasures*. Microsoft Press.
[15] OWASP Foundation. (2023). *OWASP Top Ten Web Application Security Risks*. OWASP Foundation [30] Bourdon, R. (2021). *WampServer, la plate-forme de développement Web sous Windows*. Alter Way.
[16] Sandhu, R. S., Coyne, E. J., Feinstein, H. L., & Youman, C. E. (1996). Role-based access control models. *IEEE Computer*, 29(2), 38-47. [18] Crane, D., Pascarello, E., & James, D. (2005). *Ajax in Action*. Manning Publications.
[17] Bernstein, P. A., & Newcomer, E. (2019). *Principles of Transaction Processing*. Morgan Kaufmann.

[19] Flanagan, D. (2020). *JavaScript: The Definitive Guide*. O'Reilly Media. [20] Goodman, D., Morrison, M., & Eich, B. (2007). *JavaScript Bible*. Wiley Publishing.
[21] Gutmans, A., Bakken, S. S., & Rethans, D. (2004). *PHP 5 Power Programming*. Prentice Hall.
[22] Lerdorf, R., Tatroe, K., & MacIntyre, P. (2021). *Programming PHP*. O'Reilly Media.
[23] Lockhart, J. (2023). *Modern PHP: New Features and Good Practices*. O'Reilly Media.
[31] McConnell, S. (2004). *Code Complete*. Microsoft Press.
[24] Meloni, J. C. (2012). *PHP, MySQL and Apache All-in-One For Dummies*. John Wiley & Sons.
[25] Schlossnagle, G. (2004). *Advanced PHP Programming*. Sams Publishing.
[32] Schwaber, K., & Sutherland, J. (2020). *The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game*.
[26] Sklar, D., & Trachtenberg, A. (2014). *PHP Cookbook: Solutions & Examples for PHP Programmers*. O'Reilly Media.
[27] Trachtenberg, A., & Sklar, D. (2006). *PHP Cookbook*. O'Reilly Media.
[28] Welling, L., & Thomson, L. (2020). *PHP and MySQL Web Development*. Addison-Wesley Professional.
[29] Zakas, N. C. (2012). *Maintainable JavaScript*. O'Reilly Media.

[33] Bergmann, S. (2022). *PHPUnit Manual*. Sebastian Bergmann GmbH.