
RAPPORT DE PROJET DE FIN D'ÉTUDES

En vue de l'obtention du
Diplôme National de la Licence en Sciences Informatiques

Présenté par
**Farah Saidi
Ranim Chedli**

Sujet :

**Conception et développement d'une plateforme
de gestion d'archive**

Soutenu le 24 juin 2026, devant le jury composé de :

M Zied Kechaou

Président(e)

Mme Rania Chakroun

Rapporteur

Mme Samar Bouazizi

Encadrante académique

M Mohamed Belgharek

Encadrant industriel

A.U. : 2025 - 2026

Dédicace

Avant tout, je rends grâce à Dieu Tout-Puissant pour la force, le courage et la persévérance qu'Il m'a accordés tout au long de ce parcours et qui m'ont permis d'achever ce travail.

Je dédie ce mémoire à mes très chers parents, **Rim** et **Kamel**, qui ont toujours cru en moi et m'ont soutenue sans relâche. Merci pour votre amour inconditionnel, vos sacrifices, vos encouragements et votre présence à chaque étape de ma vie. Aucun mot ne saurait exprimer toute ma gratitude envers vous.

À mes frères, **Aziz**, **Bayrem** et **Ahmed**, pour leur affection, leur soutien et tous les moments de bonheur partagés.

À mes chères amies, **Takwa** et **Houyem**, pour leur amitié sincère, leur présence constante et leurs encouragements tout au long de cette aventure.

À ma binôme, **Ranim**, pour son sérieux, son implication et l'excellente collaboration qui nous a permis de mener ce projet à bien.

J'adresse également mes plus sincères remerciements à mon encadrante, **Madame Samar Bouazizi**, pour son accompagnement, ses précieux conseils, sa disponibilité et surtout pour son soutien moral permanent. Merci d'avoir toujours cru en mes capacités, de m'avoir encouragée dans les moments difficiles et de m'avoir motivée à donner le meilleur de moi-même. Votre bienveillance et votre confiance ont été une véritable source d'inspiration tout au long de ce parcours.

Mes remerciements vont également à mes encadrants en entreprise, **Monsieur Mohamed Belgharak** et **Monsieur Khalid Said**, pour leur accompagnement, leurs conseils et leur confiance.

Enfin, je dédie ce travail à toutes les personnes qui m'ont soutenue, encouragée et accompagnée de près ou de loin durant cette expérience.

Farah SAIDI

Dédicace

C'est par la grâce de Dieu que j'ai pu accomplir ce travail.

Je dédie ce mémoire à ma chère famille, en particulier à ma mère et à mon père, pour leur amour inconditionnel, leur soutien et tous les sacrifices qu'ils ont consentis pour mon éducation et ma réussite.

À mes tantes, à mon fiancé, ainsi qu'à ma sœur, mon frère et mes cousines, pour leur présence, leur affection et leurs encouragements tout au long de mon parcours.

Je tiens également à exprimer ma profonde gratitude à mon encadrante à la faculté, **Madame Samar Bouazizi**, pour son accompagnement, ses conseils précieux, sa disponibilité et son soutien tout au long de ce projet.

Enfin, je remercie mon encadrant en entreprise, **Monsieur Khaled**, pour son aide, son suivi et ses orientations tout au long de mon stage.

Ranim CHEDLI

Remerciements

Au terme de notre projet de fin d'études, nous tenons à exprimer notre profonde gratitude à Monsieur le Doyen de la Faculté des Sciences et Techniques de Sidi Bouzid ainsi qu'à l'ensemble du corps administratif et professoral pour leurs efforts considérables et leur engagement constant, notamment au sein du département Génie Logiciel.

Nous adressons nos sincères remerciements à Mme BOUAZIZI Samar pour son encadrement rigoureux, son suivi attentif, ainsi que pour ses conseils pertinents et ses orientations précieuses qui ont été déterminants dans la réussite de ce projet.

Nous exprimons également notre reconnaissance à M. BELGHARAK Mohamed Anass, notre encadrant industriel, pour son accompagnement, sa disponibilité et ses conseils avisés tout au long de notre stage.

Nos remerciements s'étendent aussi à M. KHALED pour les ressources logicielles et matérielles mises à notre disposition, qui ont grandement facilité la réalisation de ce travail au sein de l'équipe EL MAZRAA.

Nous remercions les membres du jury d'avoir accepté d'évaluer notre travail et de consacrer leur temps à l'examen de ce projet.

Enfin, il nous est particulièrement agréable d'exprimer notre profonde reconnaissance envers toutes les personnes qui nous ont soutenus, de près ou de loin, durant notre parcours universitaire, pour leur encouragement, leur confiance et leur bienveillance.

Table des matières

Dédicace — Farah Saidi	i
Dédicace — Ranim Chedli	ii
Remerciements	iii
Liste des figures	viii
Liste des tableaux	ix
Liste des abréviations	x
Introduction générale	1
1 Cadre du projet	2
1.1 Introduction	3
1.2 Présentation de l'organisme d'accueil	3
1.2.1 Le Groupe Poulina Group Holding (PGH)	3
1.2.2 L'entreprise EL MAZRAA	3
1.2.3 Aperçu historique	4
1.2.4 Structure et organigramme	4
1.3 Étude préalable	5
1.3.1 Étude de l'existant	5
1.3.2 Critique de l'existant	6
1.3.3 Solution proposée	7
1.3.4 Analyse des causes racines (Diagramme d'Ishikawa)	7
1.3.5 Analyse SWOT	9
1.3.6 Problématique	10
1.3.7 Objectifs du projet	11
1.4 Méthodologie et conduite du projet	11
1.4.1 Méthodologie adoptée	11
1.4.2 Rôles, événements et artefacts Scrum	12
1.4.3 Organisation des Sprints	13
1.5 Conclusion	13
2 Analyse et Conception du Système	14
2.1 Introduction	15
2.1.1 Spécification des besoins	15
2.1.2 Présentation des acteurs	15
2.1.3 Besoins fonctionnels	15

2.2	Modélisation du système	18
2.2.1	Diagramme de cas d'utilisation global	18
2.2.2	Gestion des rôles et permissions (RBAC)	19
2.2.3	Diagramme de classes global	21
2.2.4	Présentation des rôles Scrum	23
2.2.5	Présentation des parties prenantes	23
2.2.6	Product Backlog	23
2.3	Architecture de l'application	25
2.3.1	Choix architectural	25
2.3.2	Pourquoi ce choix ? (Justification)	26
2.4	Environnement de travail	27
2.4.1	Configuration matérielle	27
2.4.2	Outils et technologies utilisés	27
2.4.3	Conclusion	30
3	Réalisation et déroulement des sprints	31
3.1	Introduction	32
3.2	Planification des sprints	32
3.3	Exemple détaillé du Sprint 3	33
3.3.1	Objectif du sprint	33
3.3.2	Tâches réalisées	33
3.3.3	Suivi du sprint avec Taiga	34
3.3.4	Diagramme de cas d'utilisation du Sprint 3	35
3.3.5	Diagrammes de séquence	37
3.3.6	Résultats obtenus à la fin du Sprint 3	42
3.3.7	Bilan du Sprint 3 (Sprint Review)	45
3.3.8	Difficultés rencontrées et solutions apportées	45
	Conclusion	45
4	Résultats finaux et validation	46
4.1	Introduction	47
4.2	Validation de l'API	47
4.2.1	Environnement de test	47
4.2.2	Principales API développées	47
4.2.3	Scénarios de test de l'API	47
4.2.4	Détail des tests avec captures d'écran	48
4.2.5	Bilan des tests	53
4.3	Ce que l'utilisateur voit concrètement	53
4.3.1	Le backoffice web (gestion centralisée)	53
4.3.2	L'application mobile (terrain)	54
4.4	Performances constatées	54
4.5	Types de tests et améliorations apportées	54
4.6	Conclusion	54
	Conclusion générale	55
A	Description détaillée des sprints	56
A.1	Sprint 1 : Développement Backend	56

A.2 Sprint 2 : Développement Backoffice Web	58
Conclusion de l'annexe	67
Bibliographie	67
Webographie	68

Table des figures

Figure 1.1 Logo du groupe Poulina Group Holding (PGH)	3
Figure 1.2 Organigramme fonctionnel de EL MAZRAA	5
Figure 1.3 Workflow du processus actuel de gestion des archives	6
Figure 1.4 Diagramme d'Ishikawa des causes de l'inefficacité du système d'archivage	9
Figure 1.5 Framework Scrum et cycle de développement Agile [1]	12
Figure 1.6 Diagramme de Gantt prévisionnel	13
Figure 2.1 Diagramme de cas d'utilisation globale du système	18
Figure 2.2 Diagramme de classes global du système	21
Figure 2.3 Architecture globale du système	26
Figure 3.1 Tableau de bord du Sprint 3 (Taiga)	34
Figure 3.2 Courbe Burndown du Sprint 3	35
Figure 3.3 Diagramme de cas d'utilisation – Sprint 3	35
Figure 3.4 Diagramme de séquence : Scan de code-barres et affectation	38
Figure 3.5 Diagramme de séquence : Ajout d'une archive et impression du ticket	39
Figure 3.6 Diagramme de séquence : Authentification avec récupération du rôle	40
Figure 3.7 Diagramme de séquence : Recherche d'archive (scan, saisie, galerie)	41
Figure 3.8 Diagramme de séquence : Déplacement d'archive zone et emplacement	42
Figure 3.9 Écran de connexion	42
Figure 3.10 Formulaire d'ajout d'une archive	43
Figure 3.11 Code-barres généré pour l'archive	43
Figure 3.12 Affichage du code-barres généré	43
Figure 3.13 Scan de code-barres par caméra	44
Figure 3.14 Recherche d'archive par code-barres	44
Figure 3.15 Déplacement d'archive (zone et emplacement)	45
Figure 4.1 Authentification réussie – obtention du jeton JWT	48
Figure 4.2 Ajout réussi – code-barres généré	49
Figure 4.3 Erreur 400 – titre ou type manquant	49
Figure 4.4 Refus car zone inactive	50
Figure 4.5 Résultats de recherche avec pagination	51
Figure 4.6 Déplacement réussi d'une archive	52
Figure 4.7 Accès refusé pour un utilisateur non administrateur	53
Figure 4.8 Interface de gestion des archives – backoffice web	54

Figure A.1 Cas d'utilisation – Sprint 1	56
Figure A.2 Cas d'utilisation – Sprint 2	58
Figure A.3 Diagramme de séquence – Ajout d'un utilisateur	61
Figure A.4 Diagramme de séquence – Modification d'un utilisateur	62
Figure A.5 Diagramme de séquence – Ajout d'une zone	62
Figure A.6 Diagramme de séquence – Ajout d'un emplacement	63
Figure A.7 Diagramme de séquence – Ajout d'un type d'archive	63
Figure A.8 Diagramme de séquence – Ajout d'un rôle	64
Figure A.9 Écran de connexion au backoffice	64
Figure A.10 Tableau de bord (Dashboard)	65
Figure A.11 Gestion des utilisateurs	65
Figure A.12 Gestion du profil utilisateur	66
Figure A.13 Gestion des types d'archive	66
Figure A.14 Gestion des emplacements	67
Figure A.15 Gestion des zones de stockage	67

Liste des tableaux

Tableau 1.1 Analyse SWOT du système de gestion des archives	10
Tableau 1.2 Comparatif des technologies d'identification	10
Tableau 2.1 Définition des rôles et leurs permissions	20
Tableau 2.2 Exemples de contrôle d'accès par endpoint	20
Tableau 2.3 Rôles Scrum et responsabilités	23
Tableau 2.4 Product Backlog – Backoffice Web	24
Tableau 2.5 Product Backlog – Application Mobile	25
Tableau 2.6 Configuration matérielle utilisée pour le développement et les tests	27
Tableau 2.7 Technologies Frontend et Mobile	28
Tableau 2.8 Technologies Backend et Base de données	28
Tableau 2.9 Outils de développement et de gestion	29
Tableau 2.10 Outils de modélisation et de documentation	29
Tableau 2.11 Langages de programmation utilisés	30
Tableau 3.1 Planification des sprints avec priorisation des tâches et estimation de l'effort . .	32
Tableau 3.2 Répartition des tâches – Sprint 3	34
Tableau 3.3 Cas d'utilisation – S'authentifier	36
Tableau 3.4 Cas d'utilisation – Ajouter une archive	36
Tableau 3.5 Cas d'utilisation – Scanner un code-barres	36
Tableau 3.6 Cas d'utilisation – Rechercher une archive (mot-clé ou code-barres)	37
Tableau 3.7 Cas d'utilisation – Déplacer une archive	37
Tableau 4.1 Principales API REST développées	47
Tableau 4.2 Scénarios de test de l'API	47
Tableau A.1 Cas d'utilisation – S'authentifier (Sprint 1)	57
Tableau A.2 Cas d'utilisation – Gérer les utilisateurs (Sprint 1)	57
Tableau A.3 Répartition des tâches – Sprint 1	57
Tableau A.4 Cas d'utilisation – Gérer les utilisateurs	59
Tableau A.5 Cas d'utilisation – Gérer les rôles	59
Tableau A.6 Cas d'utilisation – Gérer les emplacements	60
Tableau A.7 Cas d'utilisation – Gérer les zones	60
Tableau A.8 Cas d'utilisation – Gérer les types d'archives	60
Tableau A.9 Cas d'utilisation – Consulter le tableau de bord	61

Liste des abréviations

AIDC	Automatic Identification and Data Capture
API	Application Programming Interface
CRUD	Create, Read, Update, Delete
CORS	Cross-Origin Resource Sharing
CSS	Cascading Style Sheets
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
JWT	JSON Web Token
MVC	Model-View-Controller
ODBC	Open Database Connectivity
PGH	Poulina Group Holding
RBAC	Role-Based Access Control
REST	Representational State Transfer
RFID	Radio Frequency Identification
SQL	Structured Query Language
UML	Unified Modeling Language
UI	User Interface
UX	User Experience

Introduction générale

Avec l'évolution des activités professionnelles et l'augmentation continue du volume des informations, les entreprises sont amenées à gérer quotidiennement une grande quantité de documents administratifs, techniques et financiers. Ces documents représentent une ressource essentielle pour assurer le suivi des activités, la traçabilité des opérations ainsi que la continuité du travail au sein des organisations.

Cependant, dans plusieurs entreprises, la gestion des archives repose encore sur des méthodes manuelles basées sur le classement physique des documents. Ce fonctionnement entraîne souvent des difficultés de recherche, des pertes de temps, des erreurs de classement ainsi qu'un manque de traçabilité des opérations effectuées sur les archives. Avec l'augmentation du volume documentaire, ces limites deviennent un véritable obstacle à l'efficacité et à l'organisation du travail.

Dans ce contexte, la transformation numérique constitue une solution adaptée pour améliorer la gestion documentaire et faciliter l'accès aux informations. C'est dans cette perspective que s'inscrit notre projet de fin d'études, réalisé au sein de l'entreprise EL MAZRAA, et portant sur la conception et le développement d'un système informatisé de gestion des archives basé sur l'identification par codes-barres.

La solution proposée comprend une application mobile Android destinée aux opérations de scan, de recherche et de consultation des archives, ainsi qu'un backoffice web permettant l'administration centralisée du système. L'utilisation des codes-barres permet d'assurer une identification unique des archives, de simplifier les opérations de recherche et d'améliorer la traçabilité des mouvements des documents.

Ce système vise principalement à améliorer l'efficacité opérationnelle, à réduire les erreurs humaines et à assurer une meilleure organisation des archives au sein de l'entreprise.

Ce rapport est structuré en quatre chapitres principaux :

- **Chapitre 1 Cadre du projet** : présente l'organisme d'accueil, le contexte général du projet, l'étude de l'existant ainsi que la solution proposée.
- **Chapitre 2 Analyse et conception** : décrit les besoins fonctionnels et non fonctionnels, les acteurs du système, les diagrammes UML ainsi que l'architecture adoptée.
- **Chapitre 3 Réalisation et déroulement des sprints** : présente l'organisation des sprints, les technologies utilisées ainsi que les différentes étapes de développement du système.
- **Chapitre 4 Tests et validation** : présente les scénarios de test, les résultats obtenus, l'évaluation des performances ainsi que la validation des fonctionnalités développées.

Enfin, une conclusion générale mettra en lumière les bilans de ce travail, les objectifs atteints ainsi que les perspectives d'évolution envisagées pour la plateforme.

Chapitre 1

Cadre du projet

Table des matières

1.1	Introduction	3
1.2	Présentation de l'organisme d'accueil	3
1.2.1	Le Groupe Poulina Group Holding (PGH)	3
1.2.2	L'entreprise EL MAZRAA	3
1.2.3	Aperçu historique	4
1.2.4	Structure et organigramme	4
1.3	Étude préalable	5
1.3.1	Étude de l'existant	5
1.3.2	Critique de l'existant	6
1.3.3	Solution proposée	7
1.3.4	Analyse des causes racines (Diagramme d'Ishikawa)	7
1.3.5	Analyse SWOT	9
1.3.6	Problématique	10
1.3.7	Objectifs du projet	11
1.4	Méthodologie et conduite du projet	11
1.4.1	Méthodologie adoptée	11
1.4.2	Rôles, événements et artefacts Scrum	12
1.4.3	Organisation des Sprints	13
1.5	Conclusion	13

1.1 Introduction

Ce chapitre présente le contexte général du projet ainsi que l'environnement dans lequel il a été réalisé. Il introduit l'entreprise el mazraa et décrit le fonctionnement actuel de la gestion des archives. Une analyse du système existant est également présentée afin d'identifier les principales limites et les besoins auxquels la solution proposée devra répondre. Enfin, ce chapitre présente la méthodologie Agile Scrum adoptée ainsi que l'organisation des sprints ayant structuré le développement du système.

1.2 Présentation de l'organisme d'accueil

Cette section présente le contexte professionnel du projet à travers une brève description du Groupe Poulina (PGH), de l'entreprise EL MAZRAA, ainsi que de sa structure organisationnelle et de son évolution historique.

1.2.1 Le Groupe Poulina Group Holding (PGH)

EL MAZRAA est une filiale du groupe Poulina Group Holding (PGH), considéré comme le premier groupe privé en Tunisie. Fondé en 1968, le groupe exerce ses activités dans plusieurs secteurs, notamment l'agroalimentaire, l'immobilier, les travaux publics ainsi que l'emballage.

La diversité de ces activités ainsi que l'importance des flux administratifs et documentaires nécessitent une gestion efficace de l'information afin d'assurer une meilleure organisation et de maintenir la compétitivité du groupe.



FIGURE 1.1 – Logo du groupe Poulina Group Holding (PGH)

1.2.2 L'entreprise EL MAZRAA

Fondée en 1968, EL MAZRAA est l'une des entreprises les plus connues dans le secteur avicole tunisien. L'entreprise intervient dans plusieurs activités liées à la production et à la transformation des viandes blanches, tout en assurant la maîtrise des différentes étapes de la chaîne de production, depuis l'élevage jusqu'à la distribution des produits finis.

- **Activité** : Production et transformation de viandes blanches.
- **Réseau** : Plus de 1500 points de vente et 600 magasins sous enseigne.
- **Effectif** : Entre 1200 et 1300 collaborateurs.

Grâce à son développement continu, EL MAZRAA traite quotidiennement un volume important de documents administratifs, techniques et logistiques liés aux activités de production, de distribution et de gestion interne.

Cette importante circulation d'informations nécessite une organisation rigoureuse des archives afin d'assurer une meilleure traçabilité des documents et un accès rapide aux informations lors des opérations de contrôle ou d'audit.

Compte tenu de la diversité de ses services et du flux documentaire important, l'entreprise a besoin d'un système d'archivage fiable, structuré et facilement exploitable. Une mauvaise gestion des archives peut entraîner des retards dans l'accès à l'information, compliquer les opérations de contrôle et réduire l'efficacité du suivi administratif.

1.2.3 Aperçu historique

Depuis sa création, EL MAZRAA a connu une évolution progressive dans le secteur avicole tunisien. L'entreprise a développé ses activités à travers plusieurs étapes importantes liées à la production, à la distribution ainsi qu'à l'amélioration de la qualité et du bien-être animal.

Parmi les principales étapes de développement de l'entreprise, nous pouvons citer :

- **1968** : Création de l'entreprise EL MAZRAA.
- **1972** : Ouverture du premier magasin commercial de la société.
- **1979** : Début de la production de charcuterie à base de dinde.
- **2011** : Introduction de la gamme d'œufs commercialisée sous la marque *Ovita*.
- **2012** : Développement du réseau de distribution *Mazraa Market*.
- **2017** : Adoption d'un nouveau système de ramassage de dinde favorisant le bien-être animal et lancement d'une unité spécialisée dans les produits surgelés.
- **2018** : Mise en place d'un système moderne de ramassage de poulet destiné à améliorer les conditions de transport des volailles.
- **2019** : Commercialisation du produit *Poulet Dhabihah Fermier*.

Cette évolution reflète la volonté de l'entreprise de moderniser ses méthodes de production et de renforcer sa position sur le marché tunisien.

1.2.4 Structure et organigramme

L'entreprise est organisée autour d'un directeur général et d'un responsable administratif qui supervise quatre services principaux : les ressources humaines, les finances, la logistique et l'informatique. La gestion des archives est un processus transversal qui concerne plusieurs de ces services. Une bonne organisation documentaire est donc essentielle pour assurer une coordination efficace.

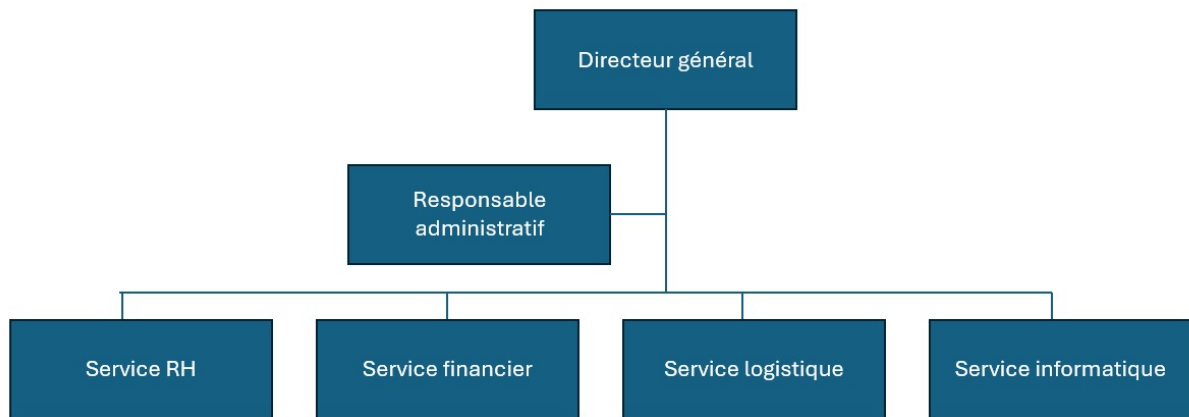


FIGURE 1.2 – Organigramme fonctionnel de EL MAZRAA

1.3 Étude préalable

Cette section analyse le fonctionnement actuel du système d’archivage manuel chez EL MAZRAA, identifie ses limites et quantifie son impact sur la productivité de l’entreprise.

1.3.1 Étude de l’existant

1.3.1.1 Description du système actuel

Le système de gestion des archives chez EL MAZRAA repose actuellement sur un fonctionnement manuel. Les documents sont classés physiquement dans des boîtes d’archives puis stockés dans des espaces dédiés.

Chaque service applique sa propre méthode de classement, sans système centralisé permettant de suivre les documents ou de faciliter leur recherche.

La consultation d’une archive peut prendre entre 15 et 30 minutes selon le volume des documents et leur emplacement. Cette situation augmente les risques d’erreurs, de perte de temps et de mauvaise organisation.

Les archives concernent principalement des factures, des bons de livraison, des documents administratifs et des pièces comptables.

1.3.1.2 Workflow du processus d’archivage manuel

Le processus actuel de gestion des archives suit généralement les étapes suivantes :

1. **Réception** : Collecte des documents papier par les différents services.
2. **Indexation manuelle** : Identification et annotation des références sur les boîtes ou classeurs.
3. **Stockage** : Classement physique des documents dans les espaces d’archivage.
4. **Recherche et consultation** : Déplacement d’un agent afin de localiser manuellement les documents demandés.

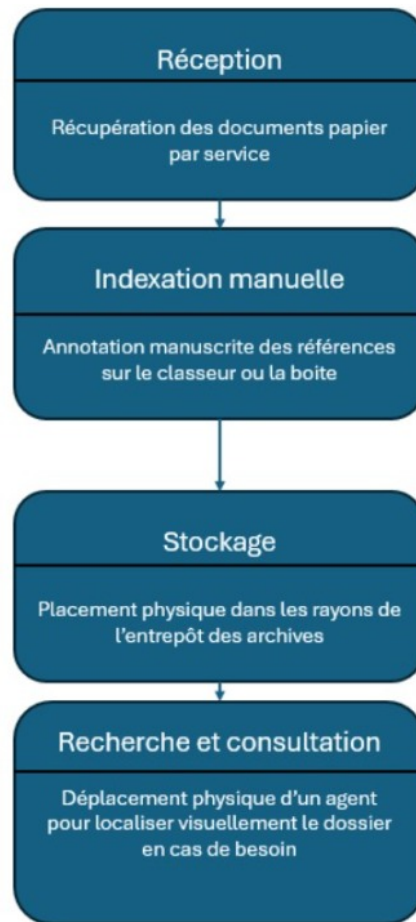


FIGURE 1.3 – Workflow du processus actuel de gestion des archives

1.3.1.3 Analyse quantitative de l'impact

Afin d'évaluer l'impact du système manuel, nous avons estimé le temps moyen consacré à la recherche documentaire.

En considérant une moyenne de 10 recherches par jour, avec une durée moyenne de 20 minutes par recherche, cela représente : $10 \times 20 = 200$ minutes par jour

Sur une base annuelle de 250 jours ouvrables, cela équivaut à plus de 40 000 minutes, soit environ 800 heures de travail consacrées uniquement à la recherche des documents.

Ces résultats montrent clairement l'impact du système manuel sur la productivité et l'efficacité organisationnelle de l'entreprise.

1.3.2 Critique de l'existant

L'analyse du système actuel de gestion des archives chez EL MAZRRAA a révélé plusieurs limites qui freinent l'efficacité du travail. Ces limites peuvent être classées en deux catégories : les problèmes de performance et les problèmes de sécurité.

Problèmes de performance et de fiabilité :

- **Manque de standardisation** : Chaque service classe ses documents à sa manière (par date, par type ou par ordre alphabétique). Du coup, tout dépend de l'expérience de l'employé. Si la personne qui connaît le système est absente, retrouver une archive devient un vrai casse-tête.
- **Recherche lente et coûteuse** : Trouver un document prend entre 15 et 30 minutes. Il faut se déplacer, ouvrir des boîtes et fouiller manuellement. Sur une année, cela représente environ 800 heures de travail perdues, l'équivalent d'un employé à plein temps pendant plusieurs mois.
- **Risques de perte et de détérioration** : Les documents papier peuvent être perdus, abîmés par l'humidité, déchirés ou même détruits accidentellement. Et souvent, il n'y a pas de copie de secours.

Problèmes de sécurité et de traçabilité :

- **Absence de traçabilité** : Personne ne sait qui a consulté un document, quand, ni où il a été déplacé. Pas d'historique, pas de suivi. En cas d'audit, c'est un vrai problème car l'entreprise ne peut pas prouver où se trouvent ses documents ni qui y a eu accès.
- **Sécurité insuffisante** : N'importe quel employé qui entre dans la zone d'archivage peut consulter des documents sensibles (contrats, données personnelles, infos financières). Aucun contrôle d'accès n'est en place.

1.3.3 Solution proposée

Afin de remédier aux limites de la gestion manuelle des archives, nous avons proposé la mise en place d'un système numérique basé sur l'utilisation des codes-barres.

La solution développée se compose d'une application mobile Android permettant le scan, la recherche, l'ajout et le déplacement des archives, ainsi qu'un backoffice web destiné à la gestion centralisée du système.

L'application mobile facilite les opérations d'archivage sur le terrain grâce à l'identification rapide des documents à l'aide des codes-barres. Elle permet également aux utilisateurs d'accéder rapidement aux informations des archives et d'assurer le suivi des mouvements en temps réel.

Le backoffice web permet quant à lui la gestion des utilisateurs, des rôles, des emplacements, des zones ainsi que des différents types d'archives. Il assure également l'administration globale du système et le contrôle des accès.

Cette solution permet d'améliorer la rapidité de recherche, la traçabilité des archives ainsi que la centralisation des données, tout en réduisant les erreurs liées à la gestion manuelle.

1.3.4 Analyse des causes racines (Diagramme d'Ishikawa)

Cette sous-section présente la méthode d'analyse des causes racines (diagramme causes-effets) et son application au contexte de la gestion manuelle des archives chez EL MAZRAA.

Objectif : L'objectif de cette analyse est d'identifier les causes profondes de l'inefficacité du système d'archivage actuel, plutôt que de se limiter aux symptômes observés (recherche lente, erreurs de classement). Cette démarche permet de s'assurer que la solution numérique proposée traite les causes réelles du problème et non uniquement ses effets visibles.

1.3.4.1 Présentation du diagramme d'Ishikawa

Le diagramme d'Ishikawa, appelé également diagramme causes-effets ou diagramme en arêtes de poisson, est une méthode utilisée pour analyser les différentes causes pouvant être à l'origine d'un problème [1].

Cette méthode, développée par Kaoru Ishikawa dans le domaine de la qualité, est souvent utilisée dans les démarches d'amélioration continue afin d'identifier les causes principales d'un dysfonctionnement.

Le principe du diagramme repose sur la classification des causes selon la méthode des 5M :

- **Main-d'œuvre** : les facteurs humains liés aux utilisateurs et aux employés ;
- **Méthode** : les procédures et méthodes de travail adoptées ;
- **Matériel** : les équipements et outils utilisés ;
- **Milieu** : l'environnement et les conditions de travail ;
- **Matière** : les documents, supports et ressources exploités.

Cette méthode facilite l'analyse des problèmes en organisant les différentes causes de manière structurée, ce qui aide à proposer des solutions adaptées.

1.3.4.2 Application du diagramme d'Ishikawa dans notre projet

Dans le cadre de notre projet, le diagramme d'Ishikawa a été utilisé dans le but de mener une analyse systémique des causes de l'inefficacité du système d'archivage. Plutôt que de traiter les symptômes (ex : "les recherches sont lentes"), il permet de remonter aux causes profondes (ex : "absence de système d'indexation"). Cette approche est cruciale pour garantir que la solution numérique proposée s'attaque aux véritables problèmes et ne se contente pas d'apporter un pansement sur une plaie plus profonde.

L'analyse a permis d'identifier plusieurs facteurs organisationnels, humains et techniques influençant l'efficacité du processus d'archivage. Les principales causes identifiées selon la méthode des 5M sont les suivantes :

- **Main-d'œuvre** : Temps important consacré à la recherche manuelle des documents.
- **Matériel** : Absence d'outils numériques tels que les scanners et lecteurs de codes-barres.
- **Méthode** : Manque de procédures standardisées pour le suivi des archives.
- **Milieu** : Difficulté d'organisation des espaces de stockage.
- **Matière** : Volume important de documents papier compliquant la traçabilité.

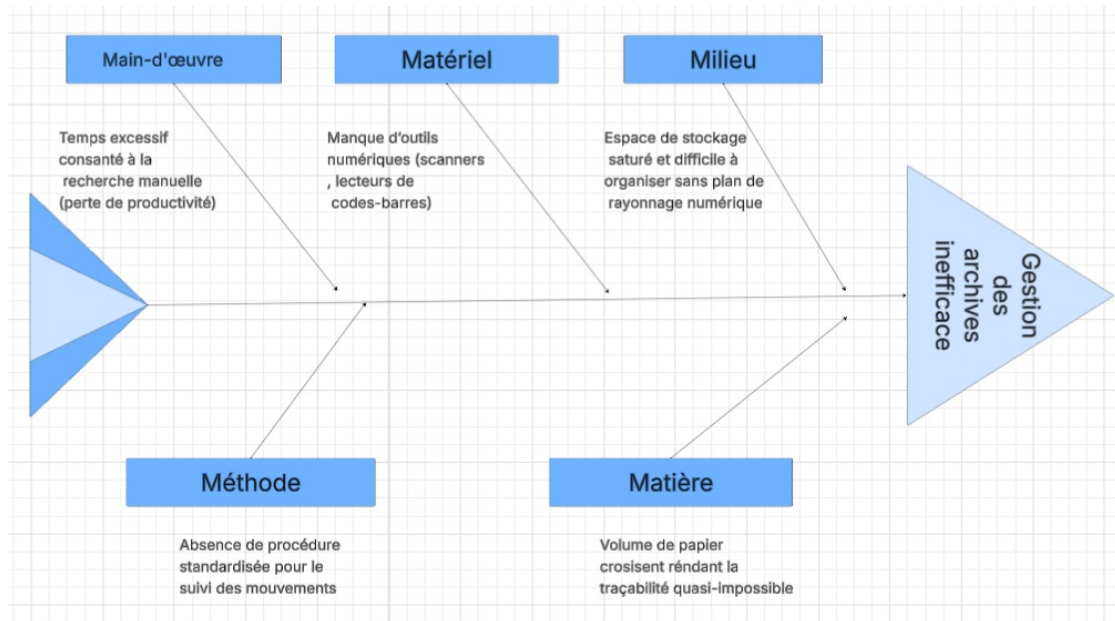


FIGURE 1.4 – Diagramme d'Ishikawa des causes de l'inefficacité du système d'archivage

Cette analyse met en évidence que les problèmes observés résultent de plusieurs facteurs interdépendants, ce qui justifie la mise en place d'une solution numérique centralisée basée sur les codes-barres.

1.3.5 Analyse SWOT

Cette sous-section présente l'analyse stratégique SWOT (Forces, Faiblesses, Opportunités, Menaces) appliquée au système de gestion des archives chez EL MAZRAA. **Objectif** : L'analyse SWOT vise ici à évaluer la pertinence stratégique de la solution envisagée avant son développement, en confrontant les capacités internes de l'entreprise (forces/faiblesses) à son environnement (opportunités/menaces), afin de s'assurer que le projet est viable et bien positionné avant d'investir du temps de développement.

1.3.5.1 Présentation de l'analyse SWOT

L'analyse SWOT est un outil stratégique qui examine un système sous quatre angles : les forces (Strengths), les faiblesses (Weaknesses), les opportunités (Opportunities) et les menaces (Threats) [2]. Elle permet d'identifier ce qui fonctionne bien, ce qui doit être amélioré, les possibilités à saisir et les risques à éviter. C'est une aide précieuse pour prendre des décisions éclairées.

1.3.5.2 Application de l'analyse SWOT dans notre projet

L'objectif de l'analyse SWOT dans ce projet est d'évaluer la pertinence stratégique de la solution numérique proposée. Elle permet de confronter les forces et faiblesses internes d'EL MAZRAA avec les opportunités et menaces de son environnement, afin de valider la viabilité du projet et d'orienter sa mise en œuvre.

Cette étude a mis en évidence plusieurs points forts, notamment l'existence d'un service informatique structuré et l'expérience du personnel. Elle a également révélé certaines limites du système actuel, telles que la lenteur des recherches, l'absence de traçabilité et les risques de perte des documents.

L'analyse a aussi montré les opportunités offertes par la transformation numérique, en particulier l'intégration d'une application mobile et des technologies d'identification automatique. En parallèle, certaines menaces ont été identifiées, comme la détérioration physique des archives et les risques liés aux audits réglementaires.

Points Forts	Points Faibles
- Existence d'un service informatique structuré. - Expérience métier du personnel.	- Lenteur des recherches documentaires. - Absence de traçabilité des consultations. - Risque de perte de documents.
Opportunités	Menaces
- Modernisation via les technologies mobiles. - Amélioration de l'efficacité opérationnelle.	- Détérioration physique des archives. - Risques lors des audits réglementaires.

TABLE 1.1 – Analyse SWOT du système de gestion des archives

Cette analyse confirme l'intérêt de mettre en place une solution numérique capable d'améliorer la rapidité d'accès aux documents, la traçabilité des opérations et la sécurité des archives.

1.3.5.3 Étude comparative des solutions d'identification

Avant de choisir la technologie d'identification pour nos archives, nous avons comparé trois solutions courantes : le code-barres, le QR code et la RFID. Le tableau 1.2 résume cette comparaison selon plusieurs critères.

Critère	Code-barres	QR Code	RFID
Coût de mise en œuvre	Faible	Faible	Très élevé
Facilité de lecture	Haute	Moyenne	Automatique
Durabilité	Moyenne	Moyenne	Haute
Volume de données	Limité	Moyen	Élevé
Choix retenu	X		

TABLE 1.2 – Comparatif des technologies d'identification

Notre choix : Nous avons retenu le **code-barres** pour trois raisons principales :

- Son coût est faible.
- Il est simple à utiliser (lecture avec un simple smartphone).
- Il s'intègre facilement dans l'environnement de l'entreprise.

Le QR code stocke plus d'informations, mais cela n'est pas nécessaire ici car toutes les données détaillées sont déjà dans notre base SQL Server. La RFID est plus rapide et automatique, mais son coût est beaucoup trop élevé pour notre projet.

Le code-barres offre donc le meilleur compromis entre coût, simplicité et efficacité.

1.3.6 Problématique

Aujourd'hui, à EL MAZRAA, la gestion des archives se fait encore entièrement à la main. Les documents sont rangés dans des boîtes, sans aucun système centralisé. Chaque service applique sa propre méthode de classement, ce qui rend la recherche encore plus compliquée.

En pratique, retrouver une archive peut prendre entre 15 et 30 minutes. Il faut se déplacer vers la zone de stockage, ouvrir plusieurs boîtes et fouiller manuellement. Ce fonctionnement entraîne une perte de temps considérable : environ 800 heures par an, rien que pour la recherche de documents.

À cela s'ajoute une absence totale de traçabilité. Personne ne sait qui a consulté un document, quand, ni où il a été déplacé. Les erreurs de classement sont fréquentes, et les risques de perte ou de détérioration des documents sont bien réels. En cas d'audit, l'entreprise ne peut pas prouver l'intégrité ni la localisation de ses archives.

Avec le volume de documents qui ne cesse d'augmenter, cette méthode manuelle atteint ses limites. L'entreprise a besoin d'un outil plus fiable, plus rapide et plus sécurisé. Il faut pouvoir retrouver une archive en quelques secondes, savoir qui l'a consultée ou déplacée, et éviter les pertes.

C'est dans ce contexte que nous proposons une solution informatique basée sur l'identification par codes-barres, permettant de répondre efficacement aux besoins d'organisation, de rapidité et de traçabilité.

1.3.7 Objectifs du projet

Objectif général : Remplacer le système d'archivage manuel par une solution numérique centralisée, afin de réduire drastiquement le temps de recherche documentaire et de garantir la traçabilité et la sécurité des archives de l'entreprise EL MAZRAA.

Objectifs spécifiques :

- Réduire le temps moyen de recherche d'une archive, actuellement estimé entre 15 et 30 minutes, à moins de 3 secondes.
- Assurer une traçabilité complète de chaque mouvement et consultation d'archive (qui, quand, où).
- Centraliser la gestion des archives, des utilisateurs et des espaces de stockage au sein d'un même système.
- Sécuriser l'accès aux documents sensibles via un contrôle d'accès basé sur les rôles (RBAC).
- Réduire les erreurs de classement liées au facteur humain.

Moyens mis en œuvre : Ces objectifs seront atteints via le développement d'une application mobile (Flutter), d'un backoffice web (React) et d'une API REST sécurisée (Node.js/Express), détaillés au chapitre 2.

1.4 Méthodologie et conduite du projet

Cette section présente la méthodologie Agile Scrum adoptée pour le développement du projet, ainsi que l'organisation des rôles, événements, artefacts et la planification des sprints.

1.4.1 Méthodologie adoptée

Nous avons choisi une méthode Agile appelée **Scrum**. Scrum fonctionne par cycles courts (les *Sprints*) : à la fin de chaque cycle, on livre une version fonctionnelle du système. Cela permet de s'adapter

facilement aux changements et de recevoir régulièrement les retours de l'entreprise [2]. Ce choix a été validé et recommandé par notre encadrant industriel, car Scrum est largement utilisée au sein du groupe Poulina pour les projets de développement logiciel.

Nous avons retenu Scrum parce que notre projet est évolutif et que nous avons besoin d'intégrer les remarques de notre encadrant industriel au fur et à mesure. Scrum facilite aussi la répartition du travail entre les deux membres de l'équipe et le suivi de l'avancement.

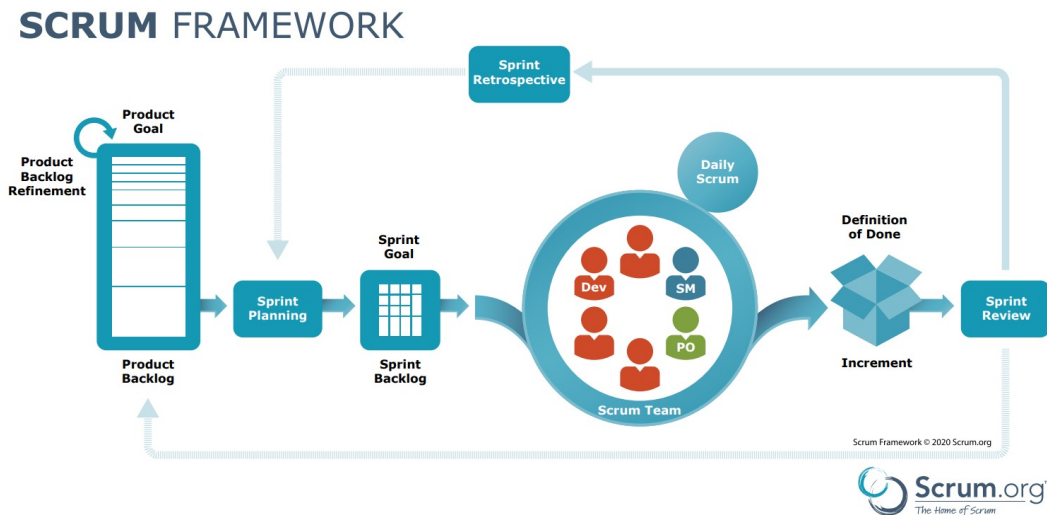


FIGURE 1.5 – Framework Scrum et cycle de développement Agile [1]

1.4.2 Rôles, événements et artefacts Scrum

Pour appliquer Scrum, nous avons défini les rôles suivants :

- **Product Owner** : représente l'entreprise EL MAZRAA, priorise les tâches et valide les livraisons. (Rôle tenu par Mohamed Belgharek)
- **Scrum Master** : veille au bon déroulement de la méthode et aide l'équipe à surmonter les obstacles. (Rôle tenu par Khaled Said)
- **Équipe de développement** : conçoit, développe et teste le backoffice web et l'application mobile. (Farah Saidi et Ranim Chedli)

1.4.2.1 Événements Scrum

Nous avons organisé quatre types de réunions :

- **Sprint Planning** : au début de chaque sprint, on choisit les tâches à faire.
- **Daily Scrum** : une courte réunion quotidienne pour voir l'avancement et les blocages.
- **Sprint Review** : à la fin du sprint, on présente ce qu'on a développé.
- **Sprint Retrospective** : on réfléchit ensemble à ce qu'on peut améliorer pour le sprint suivant.

1.4.2.2 Artefacts Scrum

Trois documents principaux nous ont aidés à organiser le travail :

- **Product Backlog** : la liste complète de tout ce qu'il faut faire, classée par priorité.

- **Sprint Backlog** : les tâches sélectionnées pour un sprint donné.
- **Incrément** : la version fonctionnelle du système livrée à la fin de chaque sprint.

Pour modéliser le système, nous avons utilisé **UML** (diagrammes de cas d'utilisation, de classes et de séquence).

1.4.3 Organisation des Sprints

Nous avons réparti le travail en trois sprints (le sprint 3 est détaillé dans le chapitre 3) :

- **Sprint 1** : Développement du backend (API REST, JWT, SQL Server).
- **Sprint 2** : Développement du backoffice web (React).
- **Sprint 3** : Développement de l'application mobile (Flutter) et validation finale.

Chaque sprint a été suivi d'une évaluation interne. La figure 1.6 montre le calendrier prévu.

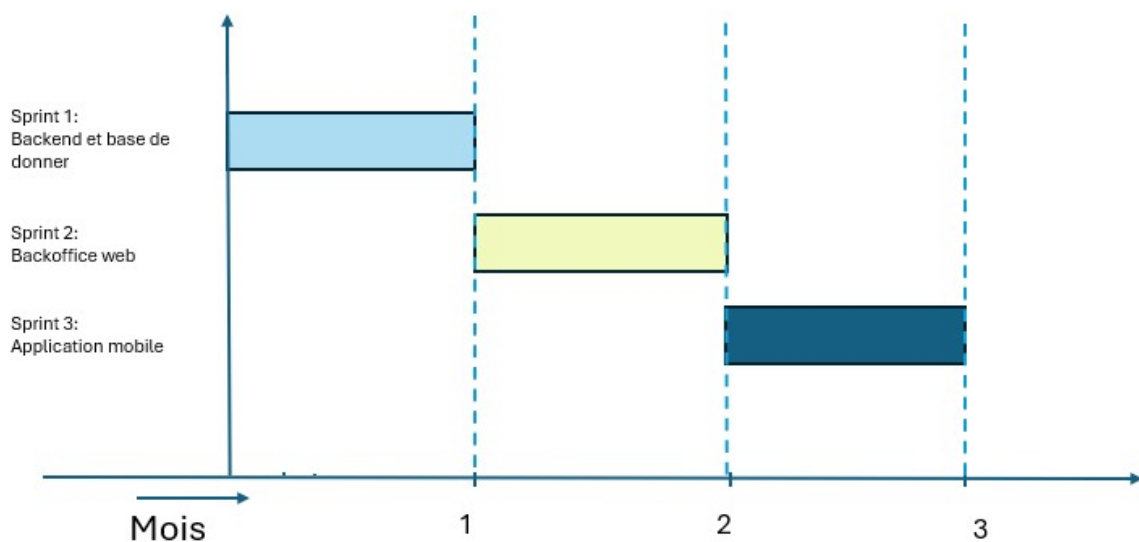


FIGURE 1.6 – Diagramme de Gantt prévisionnel

1.5 Conclusion

Dans ce chapitre, nous avons présenté le contexte du projet et analysé les limites du système actuel d'archivage d'EL MAZRAA. Cette étude nous a confirmé qu'il fallait passer à une solution informatisée, basée sur les codes-barres. Le prochain chapitre détaille l'analyse et la conception de cette solution.

Chapitre 2

Analyse et Conception du Système

Table des matières

2.1	Introduction	15
2.1.1	Spécification des besoins	15
2.1.2	Présentation des acteurs	15
2.1.3	Besoins fonctionnels	15
2.2	Modélisation du système	18
2.2.1	Diagramme de cas d'utilisation global	18
2.2.2	Gestion des rôles et permissions (RBAC)	19
2.2.3	Diagramme de classes global	21
2.2.4	Présentation des rôles Scrum	23
2.2.5	Présentation des parties prenantes	23
2.2.6	Product Backlog	23
2.3	Architecture de l'application	25
2.3.1	Choix architectural	25
2.3.2	Pourquoi ce choix ? (Justification)	26
2.4	Environnement de travail	27
2.4.1	Configuration matérielle	27
2.4.2	Outils et technologies utilisés	27
2.4.3	Conclusion	30

2.1 Introduction

Ce chapitre présente l'analyse et la conception de notre système de gestion d'archives pour EL MAZ-RAA. Nous y décrivons les besoins (fonctionnels et non fonctionnels), les différents utilisateurs, et la modélisation UML que nous avons réalisée. Nous présentons aussi l'organisation Scrum, l'architecture choisie et l'environnement technique utilisé.

2.1.1 Spécification des besoins

Cette section détaille ce que le système doit faire (besoins fonctionnels) et les contraintes qu'il doit respecter (besoins non fonctionnels). Nous identifions d'abord les acteurs qui interagiront avec le système.

2.1.2 Présentation des acteurs

Trois types d'utilisateurs interagissent avec notre système :

- **Administrateur** : il gère les utilisateurs, les rôles, les emplacements et les paramètres généraux (via le backoffice web).
- **Archiviste** : il utilise à la fois le backoffice web (avec des droits limités : consultation des archives, gestion des types, emplacements, zones et tableau de bord) et l'application mobile (pour scanner, ajouter, déplacer et rechercher des archives sur le terrain).
- **Consultant** : il peut seulement consulter et rechercher des archives, sans pouvoir les modifier.

2.1.3 Besoins fonctionnels

Cette section présente les principales fonctionnalités offertes par le système de gestion des archives selon les différents profils d'utilisateurs.

2.1.3.1 Fonctionnalités de l'administrateur (Backoffice web)

L'administrateur utilise le backoffice web (développé avec React) pour configurer et gérer l'ensemble du système. Ses principales fonctionnalités sont :

- **Gestion des utilisateurs** : L'administrateur peut créer de nouveaux comptes, modifier les informations d'un utilisateur (nom, CIN, etc.) ou désactiver un compte en cas de besoin. Il attribue également un rôle (Administrateur, Archiviste ou Consultant) à chaque utilisateur, ce qui détermine ses permissions dans le système.
- **Gestion des rôles et permissions** : L'administrateur peut définir ou modifier les droits associés à chaque rôle. Par exemple, seul l'administrateur a le droit de supprimer une archive ou de gérer les utilisateurs. Cette flexibilité permet d'adapter le système aux besoins spécifiques de l'entreprise.
- **Paramétrage des types d'archives** : L'administrateur peut ajouter, modifier ou supprimer les catégories d'archives (factures, contrats, dossiers RH, etc.). Cela permet de classer les documents de manière cohérente et facilite les recherches ultérieures.

- **Gestion des emplacements et des zones** : L’administrateur définit les espaces de stockage physiques (exemples : “Dépôt Central”, “Salle Archive 1”) ainsi que les zones à l’intérieur de ces espaces (exemples : “Armoire A”, “Étagère 3”). Il peut également activer ou désactiver un emplacement ou une zone, par exemple lors d’un réaménagement des locaux.
- **Consultation du tableau de bord** : L’administrateur accède à des indicateurs de performance (nombre total d’archives, mouvements récents, zones les plus utilisées, etc.). Ces statistiques l’aident à superviser l’activité du système et à détecter d’éventuels problèmes (exemple : une zone qui n’a jamais été utilisée).

Toutes ces actions sont bien entendu soumises à une authentification sécurisée (JWT) et ne sont accessibles qu’aux utilisateurs ayant le rôle “Administrateur”.

2.1.3.2 Fonctionnalités de l’archiviste (Backoffice web et Application mobile)

L’archiviste dispose d’un accès au backoffice web avec des permissions limitées, ainsi que d’un accès complet à l’application mobile sur le terrain. Ses principales fonctionnalités sont :

Dans le backoffice web (accès limité) :

- **Consultation des archives** : L’archiviste peut parcourir la liste des archives, consulter leurs détails (titre, description, type, emplacement, zone, code-barres) et rechercher un document spécifique à l’aide de mots-clés. Cette consultation lui permet de vérifier rapidement les informations sans se déplacer.
- **Gestion des types d’archives** : L’archiviste peut ajouter de nouvelles catégories (exemples : “Bon de livraison”, “Avenant”) ou modifier celles qui existent. Cela lui permet d’adapter le classement aux besoins changeants de son service.
- **Gestion des emplacements et des zones** : L’archiviste peut créer, modifier ou désactiver des emplacements (exemple : “Salle des archives 2”) ainsi que les zones à l’intérieur de ceux-ci (exemple : “Étagère haute”). Il ne peut toutefois pas supprimer un emplacement qui contient encore des archives.
- **Visualisation du tableau de bord** : L’archiviste accède aux indicateurs de performance (nombre d’archives ajoutées récemment, zones les plus utilisées, etc.). Cela l’aide à organiser son travail et à prioriser ses déplacements sur le terrain.

Sur l’application mobile (terrain) :

- **Ajout d’une archive** : L’archiviste remplit un formulaire (titre, description, type, emplacement, zone). Le système génère automatiquement un code-barres unique et le sauvegarde dans la base SQL Server. L’utilisateur peut ensuite télécharger ce code-barres en PDF (format A6) pour l’imprimer et le coller sur la boîte physique.
- **Scan d’un code-barres** : En utilisant la caméra du téléphone, l’archiviste scanne le code-barres imprimé sur une archive. L’application affiche immédiatement toutes les informations associées (titre, description, type, emplacement, zone). Cela permet d’identifier rapidement un document sans avoir à le chercher manuellement.
- **Recherche d’une archive** : L’archiviste peut saisir un mot-clé (titre ou description) ou un code-barres dans la barre de recherche. Les résultats s’affichent en temps réel avec pagination, ce qui facilite la navigation même lorsque le nombre d’archives est important.

- **Déplacement d'une archive** : Après avoir trouvé l'archive, l'archiviste peut modifier son emplacement et sa zone. Le backend vérifie que la nouvelle zone est active avant de valider le déplacement. Cette opération est enregistrée dans l'historique, assurant ainsi une traçabilité complète des mouvements.

Toutes les données sont synchronisées en temps réel avec le backend via des API REST sécurisées. Les actions de l'archiviste sont limitées par son rôle : il ne peut ni gérer les utilisateurs ni supprimer des archives.

2.1.3.3 Fonctionnalités du consultant

Le consultant possède un accès en lecture seule sur les deux interfaces (backoffice web et application mobile). Ses fonctionnalités sont volontairement restreintes :

- **Consultation des archives** : Le consultant peut parcourir la liste des archives et visualiser leurs détails (titre, type, emplacement, zone, code-barres). Il ne peut ni ajouter, ni modifier, ni supprimer aucun document.
- **Recherche d'archives** : Il peut effectuer des recherches par mot-clé ou par code-barres. Les résultats s'affichent avec pagination, comme pour l'archiviste. Cette fonctionnalité lui permet de retrouver rapidement un document sans avoir à contacter un archiviste.
- **Visualisation du tableau de bord** : Le consultant peut consulter les indicateurs de performance (nombre total d'archives, mouvements récents, etc.). Cela lui donne une vue d'ensemble de l'activité sans pour autant lui permettre d'intervenir sur les données.

Aucune action d'écriture (création, modification, suppression, déplacement) n'est autorisée pour le consultant. Toutes ces restrictions sont appliquées à la fois au niveau de l'interface (les boutons sont masqués) et au niveau du backend (vérification du rôle via JWT).

2.1.3.4 Besoins non fonctionnels

Nous avons veillé à ce que le système réponde à plusieurs exigences de qualité.

- **Sécurité** : L'authentification est obligatoire, les mots de passe sont chiffrés avec bcrypt, et les échanges sont protégés par des tokens JWT. La gestion des rôles (RBAC) garantit que chaque utilisateur ne fait que ce pour quoi il est autorisé. Par exemple, seul l'administrateur peut ajouter un utilisateur ou modifier son statut (actif / inactif). L'archiviste et le consultant ne peuvent effectuer aucune action d'administration.
- **Performances** : Le temps de réponse des API est inférieur à 500 ms. Grâce aux codes-barres, la recherche d'une archive ne prend que quelques secondes (contre 15 à 30 minutes dans l'ancien système). L'application reste fluide même avec des milliers d'archives.
- **Fiabilité** : SQL Server garantit l'intégrité des données. La synchronisation entre mobile, backend et backoffice se fait en temps réel. Le backend vérifie aussi qu'une zone est active avant d'autoriser l'ajout ou le déplacement d'une archive.
- **Traçabilité** : Chaque action sur une archive (ajout, modification, déplacement, consultation) est enregistrée. On sait toujours qui a fait quoi, quand et où. Essentiel pour les audits.
- **Ergonomie** : Les interfaces sont simples et intuitives. Chaque profil (admin, archiviste, consultant) voit uniquement les fonctionnalités qui le concernent. Les boutons non autorisés sont masqués.

- **Évolutivité** : L'architecture en trois couches (frontend, backend, base de données) permet d'ajouter de nouvelles fonctionnalités sans tout réécrire, et facilite la maintenance.

Ces exigences ont guidé nos choix techniques tout au long du développement.

2.2 Modélisation du système

Pour mieux concevoir notre système, nous avons utilisé UML (Unified Modeling Language). Ce langage nous a permis de représenter graphiquement les différentes parties du projet, leurs relations et leur comportement, ce qui facilite l'analyse et la compréhension [3].

2.2.1 Diagramme de cas d'utilisation global

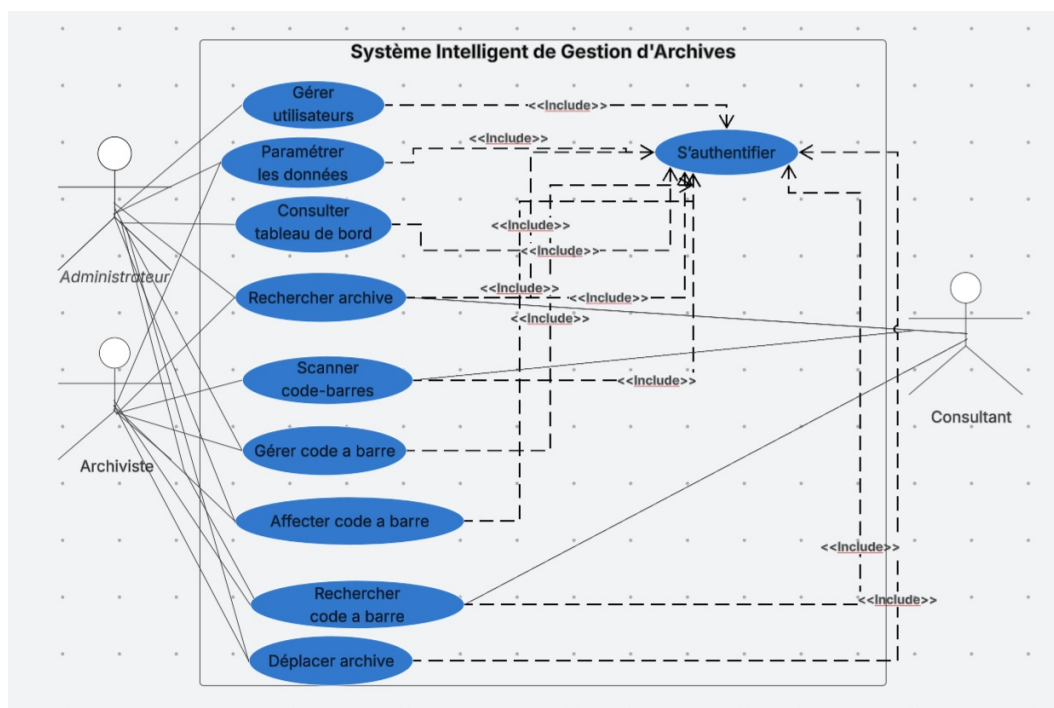


FIGURE 2.1 – Diagramme de cas d'utilisation globale du système

Ce diagramme (figure 2.1) donne une vue d'ensemble du système et montre qui peut faire quoi. On y retrouve trois types d'utilisateurs, chacun avec des droits différents. Les flèches d'héritage entre les acteurs indiquent que l'Administrateur hérite des droits de l'Archiviste, qui hérite lui-même des droits du Consultant. Ainsi, l'Administrateur possède toutes les permissions, l'Archiviste dispose de permissions élargies par rapport au Consultant, et le Consultant n'a qu'un accès en lecture seule.

Administrateur : L'Administrateur est l'acteur ayant le niveau de privilège le plus élevé. Il utilise le **backoffice web** pour administrer l'ensemble du système. Ses principales responsabilités sont :

- **Gérer les utilisateurs** : Créer, modifier ou désactiver des comptes.
- **Paramétrer les données** : Configurer les types d'archives, les emplacements et les zones de stockage.

- **Consulter le tableau de bord** : Visualiser les indicateurs de performance.
- **Rechercher des archives** : Effectuer des recherches par mot-clé ou code-barres.
- **Scanner des codes-barres** : Identifier rapidement une archive via la caméra.
- **Gérer les codes-barres** : Générer, consulter ou supprimer des codes-barres.
- **Affecter des codes-barres** : Associer un code-barres à une archive.
- **Rechercher des codes-barres** : Trouver un code-barres spécifique.
- **Déplacer des archives** : Changer l'emplacement et la zone d'une archive.

Archiviste : L'Archiviste est l'acteur principal sur le terrain. Il utilise à la fois le **backoffice web** (avec des droits limités) et l'**application mobile** (Flutter). Il possède les droits du Consultant, enrichis des fonctionnalités suivantes :

- **Paramétrer les données** : Peut ajouter ou modifier des types, emplacements et zones (sauf supprimer).
- **Consulter le tableau de bord** : Visualiser les statistiques pour organiser son travail.
- **Ajouter des archives** : Créer de nouvelles archives avec génération automatique de code-barres.
- **Scanner des codes-barres** : Identifier les archives sur le terrain via la caméra.
- **Rechercher des archives** : Retrouver des documents par mot-clé ou code-barres.
- **Gérer les codes-barres** : Générer des codes-barres pour les nouvelles archives.
- **Affecter des codes-barres** : Associer des codes-barres aux archives.
- **Rechercher des codes-barres** : Trouver un code-barres spécifique.
- **Déplacer des archives** : Mettre à jour l'emplacement et la zone d'une archive.

Consultant : Le Consultant a un accès en **lecture seule** sur les deux interfaces (backoffice web et application mobile). Ses fonctionnalités sont restreintes à :

- **Consulter le tableau de bord** : Visualiser les indicateurs sans pouvoir intervenir.
- **Rechercher des archives** : Effectuer des recherches par mot-clé ou code-barres.
- **Rechercher des codes-barres** : Trouver un code-barres spécifique.

Authentification : Toutes ces actions nécessitent une authentification préalable. C'est pourquoi on voit la relation «*include*» vers le cas "S'authentifier". Ce diagramme donne une vision claire et simple des responsabilités de chaque acteur. Les descriptions textuelles détaillées des cas d'utilisation liés à l'application mobile (S'authentifier, Ajouter une archive, Scanner un code-barres, Rechercher une archive, Déplacer une archive) sont présentées au chapitre 3 (section 3.3.4). Les cas d'utilisation liés au backoffice web (Gérer les utilisateurs, les rôles, les emplacements, les zones, les types d'archives, Consulter le tableau de bord) sont détaillés en Annexe A.

2.2.2 Gestion des rôles et permissions (RBAC)

Afin de garantir la sécurité et le contrôle d'accès au système, nous avons implémenté un mécanisme de contrôle d'accès basé sur les rôles (*Role-Based Access Control* – RBAC). Ce mécanisme permet de définir précisément ce que chaque utilisateur est autorisé à faire en fonction de son rôle.

2.2.2.1 Définition des rôles

Notre système comporte trois rôles distincts :

TABLE 2.1 – Définition des rôles et leurs permissions

Rôle	Permissions
Administrateur	Accès complet au backoffice web. Peut gérer les utilisateurs, les rôles, les archives, les types d'archives, les emplacements, les zones, et consulter le tableau de bord.
Archiviste	Accès au backoffice web (avec permissions limitées) et à l'application mobile. Dans le backoffice, peut consulter les archives, gérer les types d'archives, les emplacements et les zones, et visualiser le tableau de bord. Sur l'application mobile, peut ajouter, scanner, rechercher, consulter et déplacer des archives.
Consultant	Accès en lecture seule (via backoffice web ou mobile). Peut consulter et rechercher des archives, sans aucune action de modification.

2.2.2.2 Implémentation technique

1. **Côté backend (Node.js / Express)** : Chaque utilisateur possède un champ `roleId` dans la base SQL Server. Lors de l'authentification, le token JWT généré contient ce `roleId`. Un *middleware* vérifie à chaque requête si le rôle de l'utilisateur est autorisé à accéder à la ressource demandée. Par exemple, l'endpoint `DELETE /archives/:id` n'est accessible qu'aux administrateurs (code 403 pour les autres).
2. **Côté frontend web (React)** : L'interface adapte dynamiquement les éléments affichés en fonction du rôle stocké dans le `localStorage` (après connexion). Ainsi, un archiviste ne verra pas le menu "Gestion des utilisateurs", et un consultant n'aura pas de bouton "Ajouter une archive".
3. **Côté application mobile (Flutter)** : La même logique est appliquée : le rôle est récupéré depuis la réponse de l'API, stocké dans les *SharedPreferences*, et utilisé pour conditionner l'affichage des écrans et des boutons.

2.2.2.3 Exemple de vérification des droits

Le tableau suivant illustre quelques requêtes API et les rôles autorisés à les exécuter.

TABLE 2.2 – Exemples de contrôle d'accès par endpoint

Endpoint	Méthode	Rôles autorisés
<code>/auth/login</code>	POST	Tous (aucun token requis)
<code>/archives</code>	GET	Administrateur, Archiviste, Consultant
<code>/archives</code>	POST	Administrateur, Archiviste
<code>/archives/deplacer</code>	POST	Administrateur, Archiviste
<code>/utilisateurs</code>	GET	Administrateur uniquement
<code>/utilisateurs</code>	POST	Administrateur uniquement
<code>/zones</code>	GET	Administrateur, Archiviste, Consultant
<code>/zones</code>	POST	Administrateur uniquement

2.2.2.4 Avantages de cette approche

- **Sécurité renforcée** : Un utilisateur ne peut jamais effectuer une action pour laquelle il n'est pas autorisé, même s'il connaît l'URL de l'API.
- **Simplicité de gestion** : L'administrateur n'a qu'à attribuer un rôle à chaque utilisateur ; les permissions sont automatiquement appliquées.

- **Évolutivité** : Pour ajouter un nouveau rôle (ex : “Superviseur”), il suffit de le créer en base et d’ajouter ses permissions sans modifier tout le code.

Ce mécanisme RBAC a été implémenté dès le Sprint 1 (backend) et affiné lors des Sprints 2 (backoffice) et 3 (mobile). Il garantit une séparation claire des responsabilités conforme aux besoins de l’entreprise EL MAZRAA.

2.2.3 Diagramme de classes global

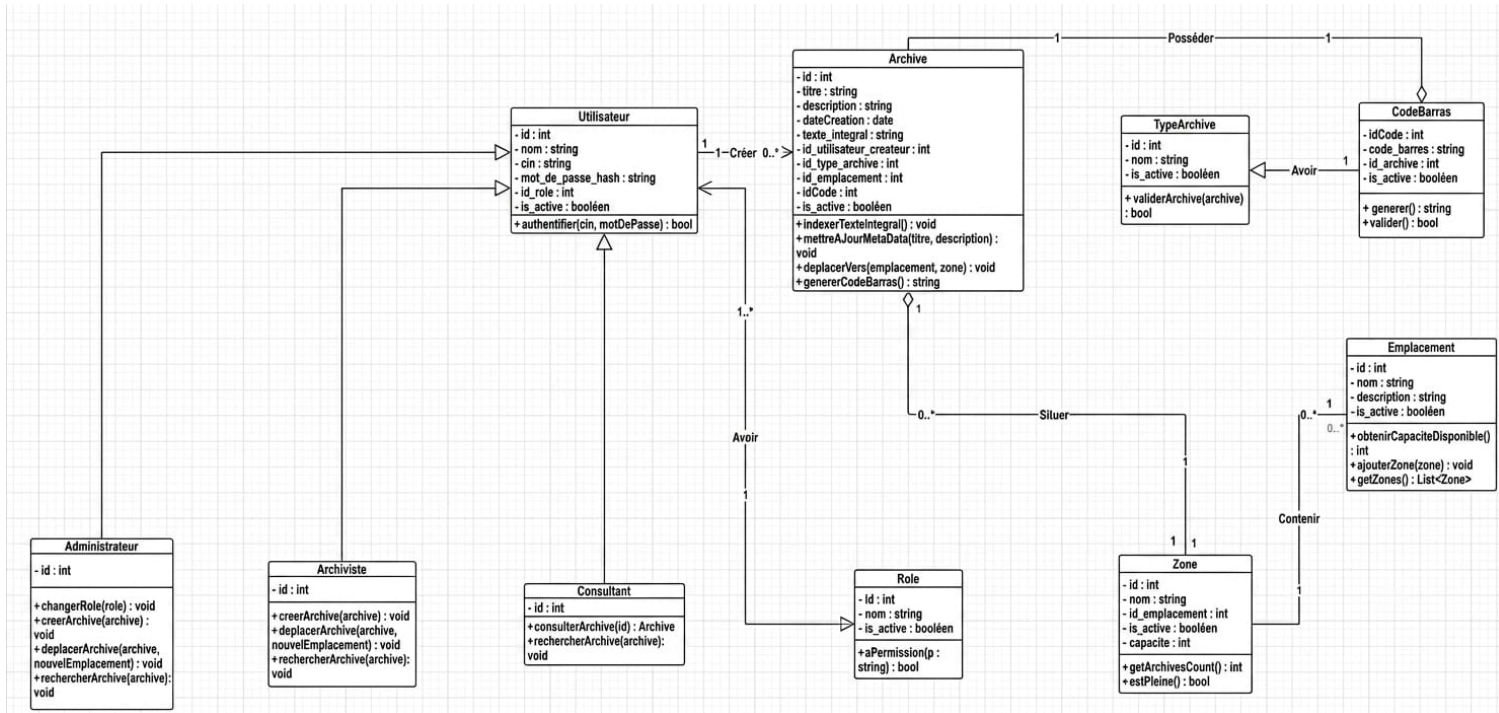


FIGURE 2.2 – Diagramme de classes global du système

Le diagramme de classes (figure 2.2) présente la structure statique du système. Il définit l’organisation des données à travers un ensemble de tables et leurs relations.

Tables principales du système Le modèle de données repose sur quatre tables fondamentales :

- **Archive** : Table centrale qui stocke les informations des documents (titre, description, date de création, texte intégral). C’est le cœur du système.
- **Utilisateur** : Table qui gère les comptes des personnes ayant accès au système (nom, CIN, mot de passe hashé, statut actif/inactif).
- **CodeBarres** : Table qui assure l’identification unique des archives. Chaque code-barres est généré automatiquement et lié à une seule archive.
- **TypeArchive** : Table de référence qui définit les catégories d’archives (ex : Contrat, Facture, Dossier RH).

Tables de gestion physique L'organisation spatiale des archives est assurée par deux tables hiérarchiques :

- **Emplacement** : Table qui représente les espaces de stockage principaux (ex : "Dépôt Central", "Salle Archive 1").
- **Zone** : Table qui représente les subdivisions à l'intérieur d'un emplacement (ex : "Armoire A", "Étagère 3"). Chaque zone est rattachée à un emplacement.

Hiérarchie des utilisateurs Le système implémente trois types d'utilisateurs organisés selon une hiérarchie de privilèges :

- **Administrateur** : Niveau de privilège le plus élevé. Peut gérer les utilisateurs, les paramètres du système et toutes les archives.
- **Archiviste** : Niveau intermédiaire. Travaille sur le terrain avec l'application mobile. Peut ajouter, scanner, rechercher et déplacer des archives.
- **Consultant** : Niveau le plus bas. Accès en lecture seule. Peut consulter et rechercher des archives sans pouvoir les modifier.

Relations entre les tables Les relations suivantes structurent le modèle de données :

- **Archive - CodeBarres** : Relation *un pour un* (1 :1). Chaque archive possède un code-barres unique et chaque code-barres est associé à une seule archive.
- **Archive - TypeArchive** : Relation *plusieurs pour un* (N :1). Plusieurs archives peuvent appartenir au même type.
- **Archive - Zone** : Relation *plusieurs pour un* (N :1). Plusieurs archives peuvent être stockées dans la même zone.
- **Zone - Emplacement** : Relation *plusieurs pour un* (N :1). Un emplacement peut contenir plusieurs zones.
- **Utilisateur - Rôle** : Relation d'héritage. Administrateur, Archiviste et Consultant héritent de la classe Utilisateur.

Règles de gestion principales Le diagramme de classes permet d'appliquer les règles métier suivantes :

- **Intégrité référentielle** : La suppression d'une archive entraîne la suppression automatique de son code-barres associé.
- **Contrôle des zones actives** : Une archive ne peut être ajoutée ou déplacée que dans une zone active (`is_active = true`).
- **Gestion des capacités** : Chaque zone dispose d'une capacité maximale. Le système vérifie qu'une zone n'est pas pleine avant d'y ajouter une archive.
- **Traçabilité des mouvements** : Les relations permettent de retracer le parcours complet d'une archive (Emplacement → Zone → Archive).

Synthèse Ce diagramme de classes constitue le modèle conceptuel de notre système. Il garantit une organisation claire des données, une séparation des responsabilités entre les acteurs et une traçabilité complète des archives. Il sert de base pour l'implémentation de la base de données SQL Server et des différentes couches de l'application.

2.2.4 Présentation des rôles Scrum

Le tableau 2.3 résume les trois rôles Scrum que nous avons mis en place, avec les noms des personnes qui les ont occupés.

TABLE 2.3 – Rôles Scrum et responsabilités

Rôle	Description	Membre
Product Owner	Représente l'entreprise EL MAZRAA, priorise les tâches et valide les livraisons.	Mohamed Belgharek
Scrum Master	Veille au bon déroulement de Scrum, organise les réunions et aide l'équipe à surmonter les obstacles.	Khaled Said
Équipe de développement	Conçoit, développe et teste l'application mobile et le backoffice web.	Farah Saidi, Ranim Chedli

Cette répartition nous a permis de bien organiser le travail, de communiquer facilement et de suivre l'avancement du projet.

2.2.5 Présentation des parties prenantes

Plusieurs personnes ou groupes ont participé à ce projet, que ce soit pour donner les besoins, suivre l'avancement ou utiliser le système final.

L'entreprise EL MAZRAA est la principale concernée : c'est elle qui a défini ce que le système doit faire et qui valide ce qu'on livre. Les administrateurs sont ceux qui gèrent le système (utilisateurs, rôles, paramètres). Les archivistes, eux, travaillent sur le terrain : ils ajoutent, scannent, recherchent et déplacent les archives. Enfin, les consultants ont un accès plus limité : ils ne peuvent que consulter et rechercher des documents.

Notre encadrante académique, Mme BOUAZIZI, a suivi notre travail sur le plan pédagogique et méthodologique. Quant à notre encadrant professionnel, M. BELGHARAK, il nous a guidés sur les besoins techniques et fonctionnels de l'entreprise.

Connaître toutes ces personnes impliquées nous a permis de mieux comprendre ce que le système devait vraiment offrir.

2.2.6 Product Backlog

Le Product Backlog est la liste complète de tout ce que notre système doit faire. Chaque fonctionnalité est écrite sous forme de « user story », c'est-à-dire une phrase simple du point de vue de l'utilisateur. Nous avons classé ces stories par priorité (Haute ou Moyenne) et nous leur avons attribué des points pour estimer leur complexité (en utilisant l'échelle de Fibonacci : 1,2,3,5,8).

Le backlog est divisé en deux parties : ce qui concerne le backoffice web (gestion et administration) et ce qui concerne l'application mobile (opérations sur le terrain). Les tableaux 2.4 et 2.5 présentent ces deux listes.

2.2.6.1 Product Backlog – Backoffice Web

Le tableau 2.4 résume les stories du backoffice. On y trouve l'authentification, la gestion des utilisateurs, des rôles, des types d'archives, des emplacements, des zones, ainsi que la consultation du tableau

de bord et la recherche d'archives. *Note : Les points sont basés sur la suite de Fibonacci (1,2,3,5,8). L'utilisation de cette échelle est détaillée dans le chapitre 3.*

TABLE 2.4 – Product Backlog – Backoffice Web

ID	User Story	Priorité	Points
US01	En tant qu'utilisateur (admin, archiviste, consultant), je veux m'authentifier pour accéder au backoffice.	Haute	3
US02	En tant qu'administrateur, je veux gérer les utilisateurs (ajout, modification, désactivation) pour contrôler les accès.	Haute	5
US03	En tant qu'administrateur, je veux gérer les rôles et permissions pour la sécurité.	Haute	5
US04	En tant qu'administrateur, je veux gérer les types d'archives (ajout, modification, suppression).	Moyenne	3
US05	En tant qu'administrateur et archiviste, je veux gérer les emplacements et zones (ajout, modification).	Haute	5
US06	En tant qu'administrateur et archiviste, je veux consulter le tableau de bord pour suivre l'activité.	Moyenne	3
US07	En tant qu'administrateur et archiviste, je veux consulter et rechercher les archives dans le backoffice.	Haute	4
US08	En tant qu'administrateur et archiviste, je veux modifier les archives (titre, description, type) pour garder les données à jour.	Moyenne	3
US09	En tant qu'administrateur, je veux activer/désactiver utilisateurs et zones pour mieux contrôler le système.	Moyenne	4
US10	En tant qu'archiviste, je veux gérer les types d'archives (ajout, modification) pour les organiser.	Moyenne	3
US11	En tant que consultant, je veux consulter les archives et le tableau de bord en lecture seule.	Moyenne	3

2.2.6.2 Product Backlog – Application Mobile

TABLE 2.5 – Product Backlog – Application Mobile

ID	User Story	Priorité	Points
US12	En tant qu’archiviste, je veux m’authentifier sur l’app mobile pour accéder aux fonctionnalités.	Haute	3
US13	En tant qu’archiviste, je veux ajouter une archive pour enregistrer ses informations.	Haute	5
US14	En tant qu’archiviste, je veux générer automatiquement un code-barres pour identifier l’archive.	Haute	5
US15	En tant qu’archiviste, je veux scanner un code-barres pour accéder rapidement à l’archive.	Haute	5
US16	En tant qu’archiviste, je veux rechercher une archive par mot-clé ou code-barres pour la retrouver vite.	Haute	4
US17	En tant qu’archiviste, je veux déplacer une archive pour mettre à jour son emplacement.	Haute	4
US18	En tant qu’archiviste, je veux télécharger le code-barres en PDF pour l’imprimer.	Moyenne	3
US19	En tant qu’archiviste, je veux consulter les détails d’une archive pour vérifier ses informations.	Moyenne	3
US20	En tant que consultant, je veux consulter les archives sur mobile pour accéder aux informations autorisées.	Moyenne	3
US21	En tant que consultant, je veux rechercher des archives sur mobile pour retrouver rapidement un document.	Moyenne	3

2.3 Architecture de l’application

2.3.1 Choix architectural

Définition de l’architecture client-serveur : L’architecture client-serveur est un modèle d’organisation des systèmes informatiques dans lequel un **client** (application web, mobile, etc.) envoie des requêtes à un **serveur** central qui les traite et renvoie les résultats. Ce modèle permet une séparation claire entre la présentation (interface utilisateur), la logique métier (traitements) et les données (stockage) [17]. Notre système repose sur une architecture client-serveur à plusieurs couches. Concrètement, nous avons séparé le code en trois grands blocs : ce que l’utilisateur voit (l’interface), la logique métier (le cerveau) et la base de données (la mémoire).

Nous avons développé deux clients différents. D’un côté, un backoffice web avec React pour les administrateurs, qui leur permet de gérer les utilisateurs, les statistiques et le paramétrage. De l’autre, une application mobile avec Flutter pour les archivistes sur le terrain, qui leur permet de scanner des codes-barres, d’ajouter et de déplacer des archives.

Ces deux clients communiquent avec un même backend central, développé avec Node.js et Express.js. Ce backend expose des API REST sécurisées. C’est lui qui gère l’authentification, les droits des utilisateurs (rôles), les règles métier et la communication avec la base de données SQL Server.

Cette architecture nous apporte plusieurs avantages : le code est mieux organisé, plus sûr, facile à faire évoluer et à maintenir.

La figure 2.3 illustre l’ensemble de ces échanges.

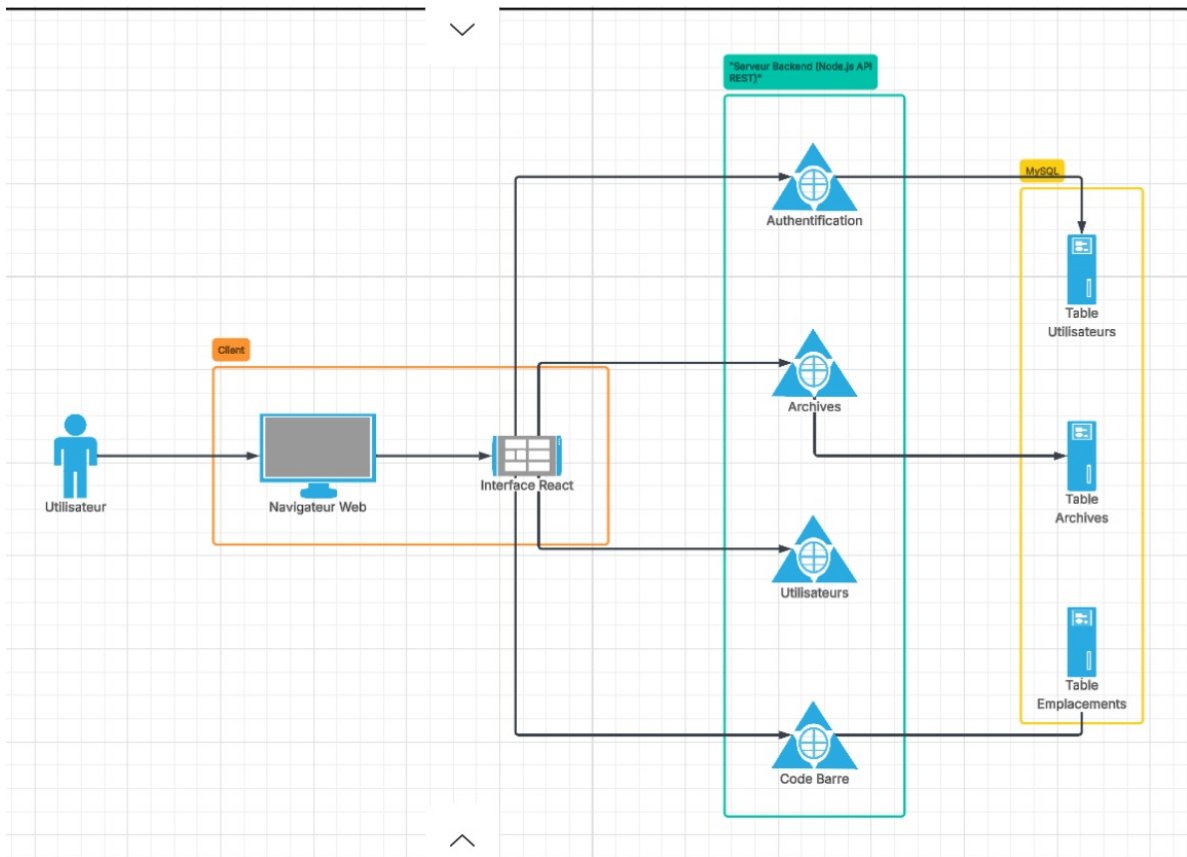


FIGURE 2.3 – Architecture globale du système

2.3.2 Pourquoi ce choix ? (Justification)

Nous avons séparé les couches pour plusieurs raisons. D'abord, les interfaces (web et mobile) ne contiennent aucune logique métier : tout le « cerveau » est centralisé dans le backend, ce qui rend le code plus facile à tester et à modifier. Ensuite, l'accès aux données passe obligatoirement par les API REST, qui sont protégées par des jetons JWT et par les rôles des utilisateurs. Il n'y a donc pas de risque de contournement. Par ailleurs, l'architecture est évolutive : on peut ajouter une nouvelle fonctionnalité (par exemple une tablette ou un scanner spécial) sans toucher aux autres couches. Enfin, la même API sert à la fois au site web et à l'application mobile, ce qui évite de dupliquer le code.

2.3.2.1 Ce que l'utilisateur voit (Couche présentation)

C'est la partie visible du système. Le backoffice React permet aux administrateurs de gérer les archives, les utilisateurs, les zones et les tableaux de bord. L'application Flutter permet aux archivistes sur le terrain de scanner, d'ajouter et de déplacer des archives. Ces deux interfaces envoient des requêtes HTTP au backend.

2.3.2.2 Le cerveau (Couche API backend)

C'est le cœur du système. Développé avec Node.js et Express, il fournit des API REST organisées par service : authentification, utilisateurs, archives, zones, emplacements, tableau de bord, codes-barres et export PDF. À chaque requête, il vérifie les droits de l'utilisateur, applique les règles métier (comme

la validation des zones actives), valide les données et dialogue avec SQL Server. Tout est centralisé et sécurisé.

2.3.2.3 La mémoire (Couche données)

Nous utilisons SQL Server pour stocker toutes les informations du système : utilisateurs, rôles, archives, types d'archives, emplacements, zones et codes-barres. Les relations entre les tables garantissent la cohérence des données et permettent au backend de retrouver facilement ce qu'il cherche.

2.3.2.4 Comment ça communique ? (Flux)

Le fonctionnement est simple. L'utilisateur clique ou tape sur une interface (web ou mobile). L'interface envoie une requête HTTPS au backend. Le backend analyse la requête, vérifie que l'utilisateur a le droit d'effectuer cette action, exécute la logique métier, va lire ou écrire dans SQL Server, puis renvoie la réponse à l'interface qui l'affiche à l'utilisateur. Le processus est fluide, propre et sécurisé.

2.4 Environnement de travail

2.4.1 Configuration matérielle

Le développement et les tests du système ont été réalisés sur les machines des membres de l'équipe, dont les spécifications sont présentées dans le tableau 2.6.

TABLE 2.6 – Configuration matérielle utilisée pour le développement et les tests

Composant	Farah Saidi	Ranim Chedli
Processeur	Intel Core i5-12450H @ 2.00 GHz	Intel Core i7-8650U @ 1.90 GHz
Mémoire RAM	8 Go	16 Go
Carte graphique	4 Go	Intel UHD Graphics 620
Système d'exploitation	Windows 11 Professionnel (64 bits)	Windows 11 (64 bits)
Stockage	SSD 477 Go	SSD 238 Go
Téléphone de test	Samsung Galaxy S24 Ultra / Android	Samsung A14

2.4.2 Outils et technologies utilisés

Dans le cadre du développement du système de gestion des archives, plusieurs outils et technologies ont été utilisés afin d'assurer la conception, le développement, les tests, la gestion du projet ainsi que la rédaction du rapport.

Le choix de ces outils s'est basé sur leur fiabilité, leur performance ainsi que leur compatibilité avec les besoins du projet.

2.4.2.1 Technologies utilisées

Nous avons choisi des technologies modernes, fiables et bien adaptées aux besoins du projet. Le tableau 2.7 résume celles utilisées pour le frontend (backoffice web) et le mobile.

TABLE 2.7 – Technologies Frontend et Mobile

Logo	Technologie	Définition	Utilisation dans le projet
	React	Bibliothèque JavaScript pour interfaces dynamiques [3].	Développement du backoffice web.
	Flutter	Framework multiplateforme de Google [4].	Développement de l'application mobile.
	Tailwind CSS	Framework CSS utilitaire [16].	Conception des interfaces du backoffice.
	Vite	Outil de développement rapide [18].	Initialisation et exécution du projet React.

Pour le backend et la base de données (tableau 2.8), nous avons utilisé Node.js et Express pour créer l'API REST, et SQL Server pour stocker toutes les données.

TABLE 2.8 – Technologies Backend et Base de données

Logo	Technologie	Définition	Utilisation dans le projet
	Node.js	Environnement JavaScript côté serveur [5].	Développement du backend.
	Express.js	Framework pour API REST [6].	Organisation des routes API.
	SQL Server	Base de données relationnelle Microsoft [8].	Stockage des données du système.

Côté gestion de projet et collaboration (tableau 2.9), nous avons utilisé GitHub pour le versionnement, Visual Studio Code comme éditeur, Postman pour tester les API, et Taiga pour le suivi agile (Scrum).

TABLE 2.9 – Outils de développement et de gestion

Logo	Outil	Définition	Utilisation dans le projet
	GitHub	Plateforme de gestion de versions [9].	Partage et suivi du code source.
	Visual Studio Code	Éditeur de code léger et extensible [10].	Développement backend, web et mobile.
	Postman	Outil de test d'API [14].	Tests des routes API.
	Taiga	Plateforme Agile Scrum [12].	Planification des sprints et suivi des tâches.
	ngrok	Outil qui crée un tunnel sécurisé pour exposer un serveur local (sur votre machine) sur Internet.[13].	Exposition du backend local (port 3000) à une URL publique fixe pour les tests mobiles sur différents réseaux.

Pour la modélisation et la rédaction (tableau 2.10), nous avons conçu les diagrammes UML avec Lucidchart et rédigé le rapport avec Overleaf (LaTeX).

TABLE 2.10 – Outils de modélisation et de documentation

Logo	Outil	Définition	Utilisation dans le projet
	Lucidchart	Application de diagrammes UML [11].	Conception des diagrammes UML.
	Overleaf (LaTeX)	Rédaction scientifique en ligne [19].	Rédaction du rapport de PFE.

Enfin, le tableau 2.11 liste les langages de programmation que nous avons manipulés : JavaScript (backend et web), Dart (mobile), SQL (requêtes SQL Server), et HTML/CSS (structuration des pages web).

TABLE 2.11 – Langages de programmation utilisés

Logo	Langage	Définition	Utilisation dans le projet
	JavaScript	Langage web dynamique [20].	Backend Node.js et frontend React.
	Dart	Langage de Google pour Flutter [21].	Application mobile.
	SQL	Langage des bases relationnelles [22].	Requêtes SQL Server.
	HTML / CSS	Structuration et mise en forme du web [23].	Interfaces du backoffice web.

L'ensemble de ces outils et technologies nous a permis de développer une solution cohérente, bien adaptée aux besoins de l'entreprise.

2.4.3 Conclusion

Dans ce chapitre, nous avons présenté l'analyse et la conception du système de gestion des archives, ainsi que les principaux besoins, les diagrammes UML et l'architecture adoptée.

Cette phase constitue une base essentielle pour la réalisation et l'implémentation du système présentées dans le chapitre suivant.

Chapitre 3

Réalisation et déroulement des sprints

Table des matières

3.1	Introduction	32
3.2	Planification des sprints	32
3.3	Exemple détaillé du Sprint 3	33
3.3.1	Objectif du sprint	33
3.3.2	Tâches réalisées	33
3.3.3	Suivi du sprint avec Taiga	34
3.3.4	Diagramme de cas d'utilisation du Sprint 3	35
3.3.5	Diagrammes de séquence	37
3.3.6	Résultats obtenus à la fin du Sprint 3	42
3.3.7	Bilan du Sprint 3 (Sprint Review)	45
3.3.8	Difficultés rencontrées et solutions apportées	45
	Conclusion	45

3.1 Introduction

Ce chapitre présente le déroulement du projet de gestion des archives à travers la planification des sprints et les différentes étapes de développement du système. Un focus particulier est consacré au Sprint 3, dédié au développement de l'application mobile, afin de présenter les objectifs, les tâches réalisées, les résultats obtenus ainsi que les difficultés rencontrées.

3.2 Planification des sprints

La planification des sprints consiste à définir les objectifs de chaque itération, à prioriser les tâches selon leur importance métier et à estimer l'effort nécessaire à leur réalisation.

Dans le cadre de notre projet, les sprints ont été organisés selon une progression logique allant du développement backend jusqu'à l'application mobile.

- **Sprint 1** : Développement Backend
- **Sprint 2** : Développement Backoffice Web
- **Sprint 3** : Développement Application Mobile

L'estimation de l'effort a été réalisée de manière approximative en se basant sur la complexité des tâches, leur durée ainsi que les contraintes techniques du projet. Cette approche a permis d'assurer une répartition équilibrée du travail et une meilleure gestion du temps.

Le tableau suivant présente la planification détaillée des sprints avec les priorités et les estimations.

TABLE 3.1 – Planification des sprints avec priorisation des tâches et estimation de l'effort

Sprint et tâches	Priorité	Points
Sprint 1 – Développement Backend		
Authentification (JWT)	Haute	8
Gestion des utilisateurs et rôles	Haute	8
Développement API archives	Haute	8
Connexion à la base de données SQL Server	Haute	5
Validation des zones actives	Moyenne	5
Sprint 2 – Développement Backoffice Web		
Interface gestion des archives	Haute	8
Gestion des emplacements et zones	Haute	5
Gestion des types d'archives	Moyenne	5
Tableau de bord	Moyenne	3
Sprint 3 – Application Mobile		
Authentification mobile	Haute	5
Ajout d'une archive	Haute	8
Scan de code-barres	Haute	8
Recherche d'archives	Moyenne	5
Synchronisation avec backend	Haute	5

Note : Une échelle inspirée de la suite de Fibonacci (1, 2, 3, 5, 8, etc.) est souvent utilisée dans la méthodologie Scrum afin d'estimer la complexité des tâches [7].

Cette planification met en évidence une progression structurée du projet, respectant une logique incrémentale allant de l'analyse jusqu'à l'implémentation finale. Elle a permis d'assurer une évolution cohérente du système ainsi qu'une meilleure organisation des différentes phases de développement.

Les Story Points dans notre projet : Les **Story Points** mesurent la **complexité** d'une tâche, pas le temps. Nous avons utilisé les chiffres 3, 5 et 8 (Fibonacci) pour noter nos fonctionnalités.

- **3 points** : tâche moyenne (exemple : se connecter, tableau de bord, générer un PDF).
- **5 points** : tâche complexe (exemple : gérer les utilisateurs, scanner un code-barres, gérer les emplacements).
- **8 points** : tâche très complexe (exemple : recherche avancée avec pagination).

Notre projet totalisait **35 points**. Notre équipe réalise environ **15 points par Sprint** (2 semaines). Nous avons donc besoin de **2 à 3 Sprints** pour finir le projet.

$$\text{Durée} = \frac{35}{15} \approx 2,33 \text{ Sprints}$$

À retenir : Fibonacci (3,5,8) évite la confusion entre notes proches (comme 4 et 5). Cela nous a aidés à mieux planifier notre travail.

3.3 Exemple détaillé du Sprint 3

Le Sprint 3 est présenté en détail car il concerne le développement de l'application mobile ainsi que son intégration avec le backend et le module de scan de codes-barres.

Ce sprint représente une étape importante du projet puisqu'il regroupe plusieurs fonctionnalités essentielles du système, notamment l'ajout d'archives, la recherche et la synchronisation des données.

Les autres sprints sont présentés de manière synthétique dans les sections précédentes.

3.3.1 Objectif du sprint

L'objectif principal du Sprint 3 était de développer l'application mobile du système de gestion des archives et d'assurer son intégration avec le backend.

Ce sprint visait également à implémenter les fonctionnalités essentielles, notamment l'ajout d'une archive, le scan de codes-barres pour une identification rapide, ainsi que la consultation des données en temps réel.

Par ailleurs, une attention particulière a été accordée à la synchronisation des données et à l'application des règles métier, afin de garantir la cohérence et la fiabilité du système.

3.3.2 Tâches réalisées

Le tableau suivant présente les principales tâches réalisées dans le cadre du Sprint 3. Cette répartition permet de prioriser les activités de développement, d'estimer l'effort à travers les story points et d'aligner les objectifs du sprint avec l'avancement global du projet.

La classification des tâches selon leur priorité et leur complexité permet d'assurer une meilleure organisation du travail et de garantir la livraison des fonctionnalités critiques telles que le scan de codes-barres et l'intégration avec le backend dans les délais prévus.

TABLE 3.2 – Répartition des tâches – Sprint 3

Tâche	Priorité	Points
Développement de l’authentification mobile	Haute	5
Implémentation de l’ajout d’une archive	Haute	8
Intégration du module de scan de code-barres	Haute	8
Connexion de l’application mobile au backend (API REST)	Haute	5
Implémentation de la recherche d’archives	Moyenne	5
Synchronisation des données en temps réel	Haute	5
Tests fonctionnels de l’application mobile	Moyenne	3
Correction des bugs et amélioration de l’interface	Moyenne	3

3.3.3 Suivi du sprint avec Taiga

Ce qu’on voit sur le tableau Taiga (Figure 3.1) Le tableau montre notre travail pendant le Sprint 3. On a plusieurs colonnes : “New”, “In Progress”, “Ready for Test”, “Closed”. Chaque tâche est une carte qu’on déplace au fur et à mesure. À la fin du sprint, toutes les cartes sont dans la colonne “Closed”. Cela veut dire que tout ce qu’on avait prévu (13 story points) a été terminé. Aucune tâche n’est restée bloquée.

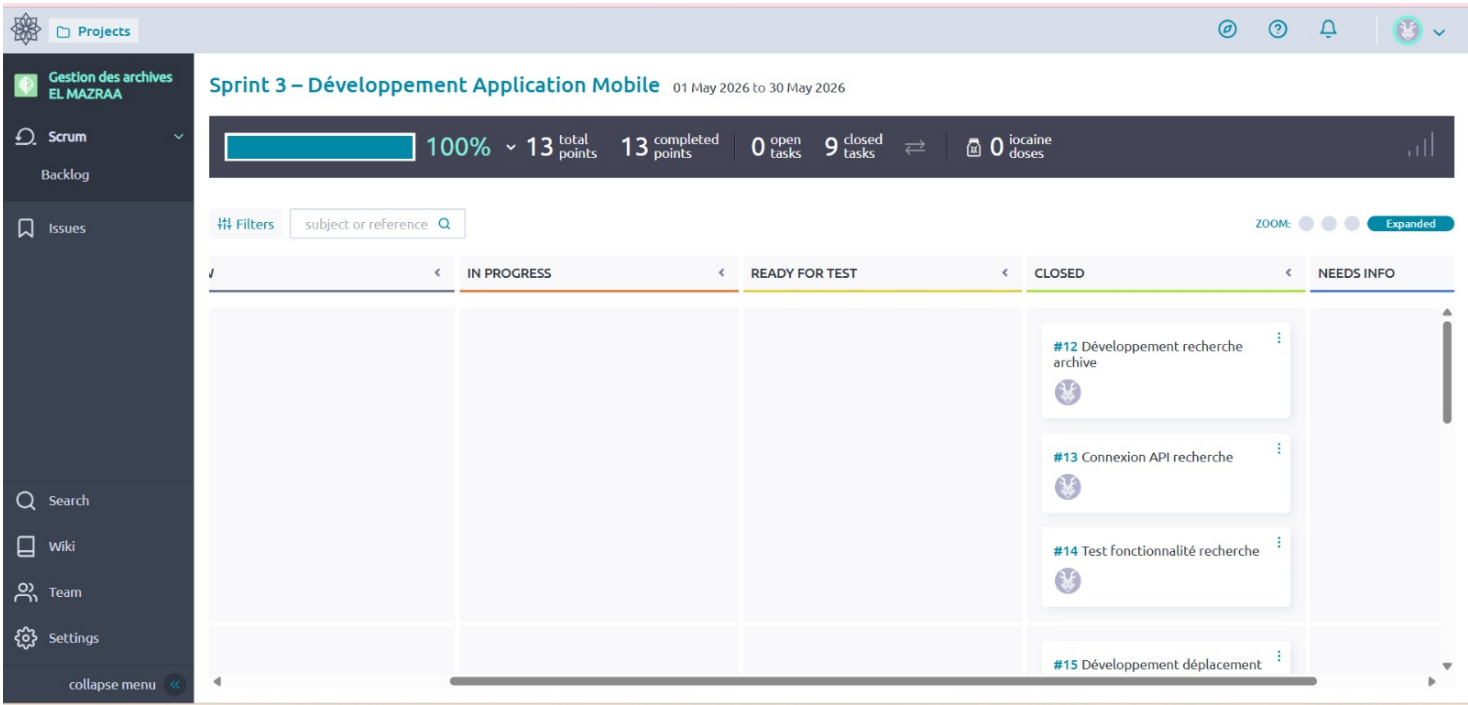


FIGURE 3.1 – Tableau de bord du Sprint 3 (Taiga)

Ce qu’on apprend de la courbe Burndown (Figure 3.2) La courbe Burndown montre la quantité de travail qui reste jour par jour. Au début du sprint, il y avait 13 points de travail. Les premiers jours, la courbe descend très vite : on a attaqué les tâches les plus grosses (scan, ajout d’archive). Ensuite, la pente devient plus douce : c’est la période des tests et des petites corrections. À la fin, la courbe arrive à zéro – plus aucun point restant. On a fini exactement dans les temps prévus (du 1er mai au 30 mai).

Sprint 3 – Développement Application Mobile 01 May 2026 to 30 May 2026

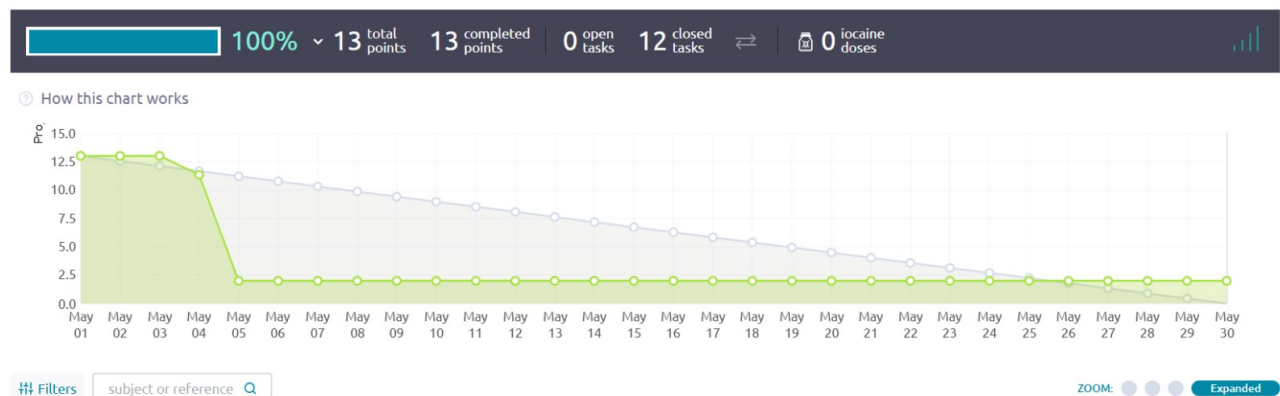


FIGURE 3.2 – Courbe Burndown du Sprint 3

Les bénéfices pour nous (binôme) Travailler avec Taiga nous a beaucoup aidés :

- **Voir l'avancement en temps réel** : on savait sans se poser de questions ce qui était fini et ce qui restait.
- **Partager le travail sans conflit** : chacun prenait une tâche différente et on ne faisait pas la même chose.
- **Repérer un retard vite** : si la courbe ne descendait pas assez, on réagissait immédiatement.
- **Se motiver** : fermer une tâche et la voir passer dans la colonne "Closed" donnait envie d'avancer.

Grâce à Taiga, on a terminé le Sprint 3 à 100%, sans stress inutile, et en restant toujours d'accord sur qui fait quoi.

3.3.4 Diagramme de cas d'utilisation du Sprint 3

includegraphics[width=0.7]figures/usecase_sprint4.png

FIGURE 3.3 – Diagramme de cas d'utilisation – Sprint 3

Ce diagramme concerne l'application mobile. Il montre ce que l'utilisateur (archiviste ou administrateur) peut faire pendant le Sprint 3.

Les actions principales sont :

- **Ajouter une archive** : l'utilisateur entre les informations, le système génère un code-barres unique.
- **Scanner un code-barres** : grâce à la caméra du téléphone, on lit le code et on retrouve l'archive.
- **Rechercher une archive** : on peut chercher par mot-clé ou par code-barres.
- **Déplacer une archive** : après avoir trouvé l'archive, on peut changer son emplacement et sa zone.

Toutes ces actions nécessitent d'être authentifié. C'est pourquoi on voit la relation «include» vers "S'authentifier". Ce diagramme résume bien le périmètre du Sprint 3 : l'essentiel des fonctionnalités terrain de l'application mobile.

TABLE 3.3 – Cas d'utilisation – S'authentifier

Acteur	Utilisateur
Objectif	Obtenir un token JWT
Précondition	Compte valide
Postcondition	Token stocké, accès Dashboard
Scénario nominal	1. Saisir CIN + mot de passe. 2. Envoi <code>POST /auth/login</code>. 3. Vérification CIN + bcrypt. 4. Génération token JWT. 5. Stockage token. 6. Redirection Dashboard.
Scénarios alternatifs	1a. (étape 1) CIN inexistant → “CIN non trouvé”. 1b. (étape 1) Mot de passe incorrect → “Mot de passe invalide”. 3a. (étape 3) Compte désactivé → “Compte désactivé”. 5a. (étape 5) Token expiré → retour à l'étape 1 (relogin).

TABLE 3.4 – Cas d'utilisation – Ajouter une archive

Acteur	Archiviste (ou Admin)
Objectif	Créer une archive et générer son code-barres
Précondition	Authentifié (token JWT) + rôle Archiviste/Admin
Postcondition	Archive dans SQL Server, code-barres unique, PDF téléchargeable
Scénario nominal	1. Ouvrir app mobile → “Ajouter archive”. 2. Remplir formulaire (titre, description, type). 3. Choisir emplacement (ex : Dépôt Central), puis zone. 4. Valider la création. 5. Envoi <code>POST /archives</code> au backend. 6. Backend vérifie zone active → insertion dans SQL Server. 7. Génération code-barres (format <code>ARCH_AAAAMMJJ_NNNNN</code>). 8. Affichage + téléchargement PDF (A6).
Scénarios alternatifs	1a. (étape 2) Titre vide → erreur 400 “Titre obligatoire”. 1b. (étape 6) Zone inactive → erreur 400 “Zone inactive”. 1c. (étape 2) Type invalide → message “Type invalide”. 1d. (étape 5) Pas de connexion réseau → “Serveur inaccessible”.

TABLE 3.5 – Cas d'utilisation – Scanner un code-barres

Acteur	Archiviste (ou Admin)
Objectif	Lire un code-barres avec la caméra et afficher l'archive
Précondition	Authentifié + permission caméra
Postcondition	Affichage des infos (titre, type, emplacement, zone)
Scénario nominal	1. Ouvrir app mobile → “Scanner”. 2. Cadrer le code-barres avec la caméra. 3. Lecture du code → envoi <code>GET /archives?barcode=...</code> 4. Backend cherche dans <code>codes_barres</code> puis <code>archives</code>. 5. Retourne les infos (titre, type, emplacement, zone). 6. Affichage des détails à l'écran.
Scénarios alternatifs	1a. (étape 3) Code illisible → message “Code non reconnu”. 1b. (étape 4) Code inexistant → erreur 404 “Archive non trouvée”. 1c. (étape 2) Caméra refusée → proposition de saisie manuelle.

TABLE 3.6 – Cas d’utilisation – Rechercher une archive (mot-clé ou code-barres)

Acteur	Administrateur, Archiviste, Consultant
Acteur	Archiviste / Admin
Objectif	Trouver une archive par mot-clé ou code-barres
Précondition	Authentifié (token JWT)
Postcondition	Liste paginée des archives correspondantes
Scénario nominal	1. Saisir mot-clé ou code-barres dans la barre de recherche. 2. Construction requête GET /archives?search=... & page=1 & limit=10. 3. Backend interroge SQL Server avec opérateurs LIKE. 4. Retourne liste paginée (page, limit, total). 5. Affichage des résultats (titre, type, emplacement, zone, code-barres).
Scénarios alternatifs	1a. (étape 4) Aucun résultat → message « Aucune archive correspondante ». 1b. (étape 2) Changement de page → requête avec page=2. 1c. (étape 1) Saisie code-barres exact → affichage direct (sans pagination).

TABLE 3.7 – Cas d’utilisation – Déplacer une archive

Acteur	Archiviste (ou Admin)
Objectif	Changer l’emplacement et la zone d’une archive
Précondition	Archive existe (code-barres valide) + authentifié
Postcondition	Mise à jour colonnes id_emplacement et id_zone
Scénario nominal	1. Scanner ou rechercher l’archive à déplacer. 2. Affichage emplacement et zone actuels. 3. Cliquer « Déplacer » → choisir nouvel emplacement. 4. Choisir nouvelle zone (liste filtrée). 5. Valider la modification. 6. Envoi POST /archives/deplacer (code-barres, nouvel emplacement, zone). 7. Backend vérifie zone active → exécute UPDATE. 8. Retourne message de succès. 9. Affichage des nouvelles coordonnées.
Scénarios alternatifs	1a. (étape 7) Zone inactive → erreur 400 « Zone inactive ». 1b. (étape 4) Zone n’appartient pas à l’emplacement → erreur 400 « Zone invalide ». 1c. (étape 1) Archive inexistante → erreur 404 « Archive non trouvée ».

Remarque : Les cas d’utilisation “Gérer archives”, “Rechercher archives” et “Déplacer archives” sont détaillés dans le diagramme de l’application mobile (chapitre 3) et dans le diagramme global (chapitre 2).

3.3.5 Diagrammes de séquence

Scan d’un code-barres et affectation Le diagramme de la figure 3.4 décrit le processus complet réalisé par un archiviste à l’aide de l’application mobile : scanner un code-barres puis affecter l’archive à une zone et un emplacement.

Phase 1 : Scan et consultation : L’archiviste accède à l’interface “Scanner Archive” depuis le tableau de bord mobile. Il active la caméra et cadre le code-barres imprimé sur l’archive physique. L’application transmet ce code au backend via une requête GET /archives?barcode=... Après vérification de l’authentification (token JWT), le backend interroge la base SQL Server (tables codes_barres puis archives). Si l’archive est trouvée, le backend renvoie ses informations (titre, type, emplacement, zone). L’application les affiche à l’écran.

Phase 2 : Affectation (zone et emplacement) : Une fois les détails de l’archive affichés, l’archiviste remplit

les champs obligatoires “Zone” et “Emplacement”. Il valide l’affectation. L’application envoie alors une requête `POST /archives/affecter` (ou `PUT`) au backend. Ce dernier met à jour les informations de l’archive dans SQL Server (colonnes `id_zone`, `id_emplacement`) et retourne une confirmation. L’application mobile affiche alors le message “Affectation enregistrée avec succès”.

Ce diagramme montre que le backend (Node.js) orchestre la sécurité (JWT), la recherche et la mise à jour des données, garantissant ainsi la cohérence et la traçabilité des mouvements des archives.

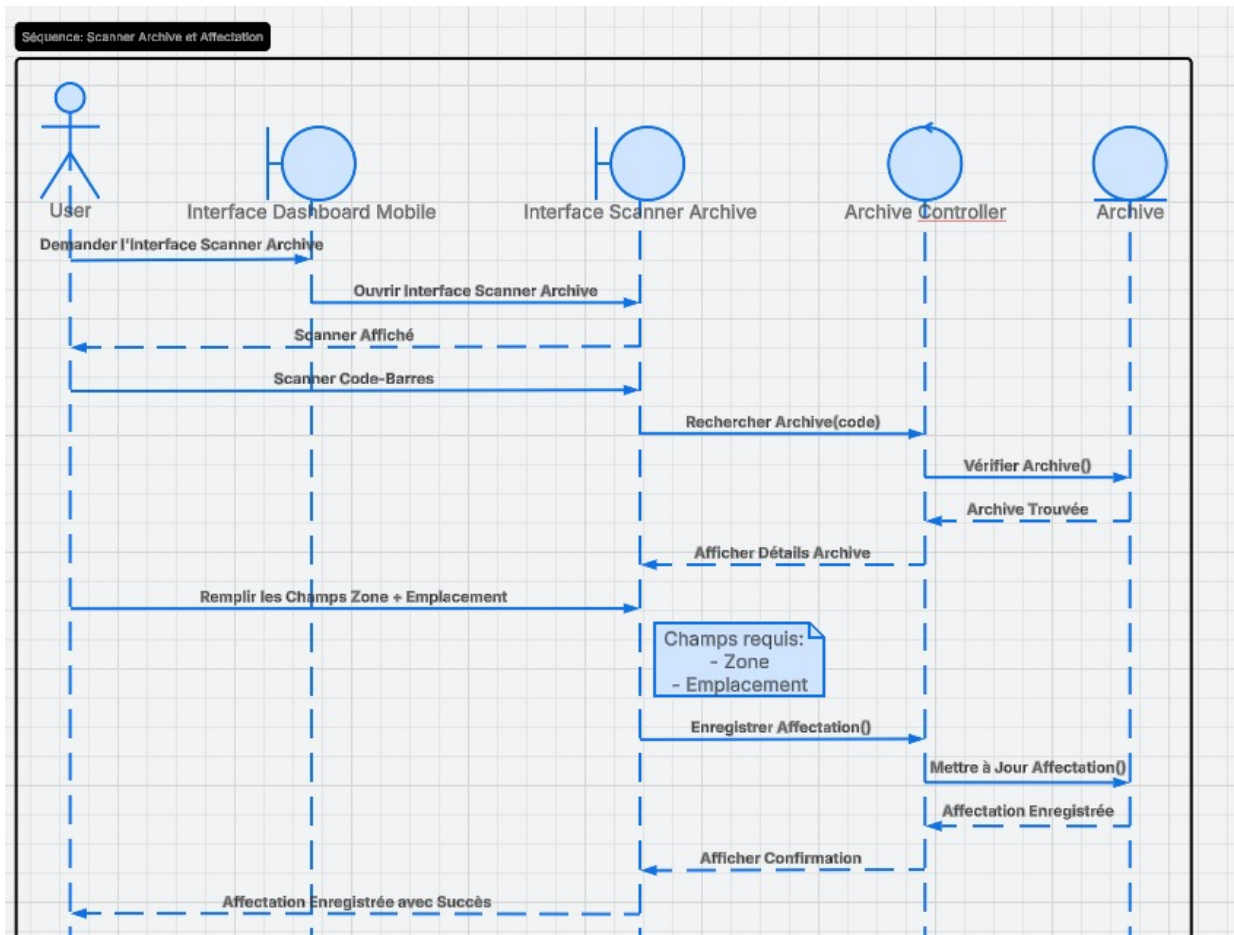


FIGURE 3.4 – Diagramme de séquence : Scan de code-barres et affectation

Ajout d’une archive La figure 3.5 illustre la création d’une nouvelle archive ainsi que l’impression de son ticket (code-barres).

Phase 1 : Saisie et envoi des données : L’archiviste, depuis le tableau de bord mobile, demande l’interface d’ajout d’archive. L’application affiche un formulaire qu’il remplit (titre, type, emplacement, zone). Une fois validé, l’application envoie une requête `POST /archives` au backend.

Phase 2 : Contrôles et création dans la base : Le backend vérifie l’authentification de l’utilisateur (token JWT) ainsi que son rôle (Archiviste ou Administrateur). Il s’assure également que la zone sélectionnée est active (colonne `is_active` à 1). Si toutes les conditions sont remplies, le backend insère la nouvelle archive dans la table `archives` de SQL Server. Il génère automatiquement un code-barres unique (format `ARCH_AAAAMMJJ_NNNNN`) et l’enregistre dans la table `codes_barres`.

Phase 3 : Affichage et impression du ticket : Le backend renvoie à l’application mobile les informations complètes de l’archive ainsi que le code-barres généré. L’application affiche alors un écran récapitulatif avec un bouton “Imprimer”. Lorsque l’archiviste clique sur ce bouton, l’application demande au backend de générer un ticket (fichier PDF au format A6) contenant le code-barres. Le backend retourne le PDF généré, et l’application l’affiche à l’écran, prêt à être imprimé.

Ce processus montre que le système ne se contente pas d'enregistrer des données : il applique des règles métier (vérification de la zone active) et automatise la génération des identifiants ainsi que l'impression des tickets.

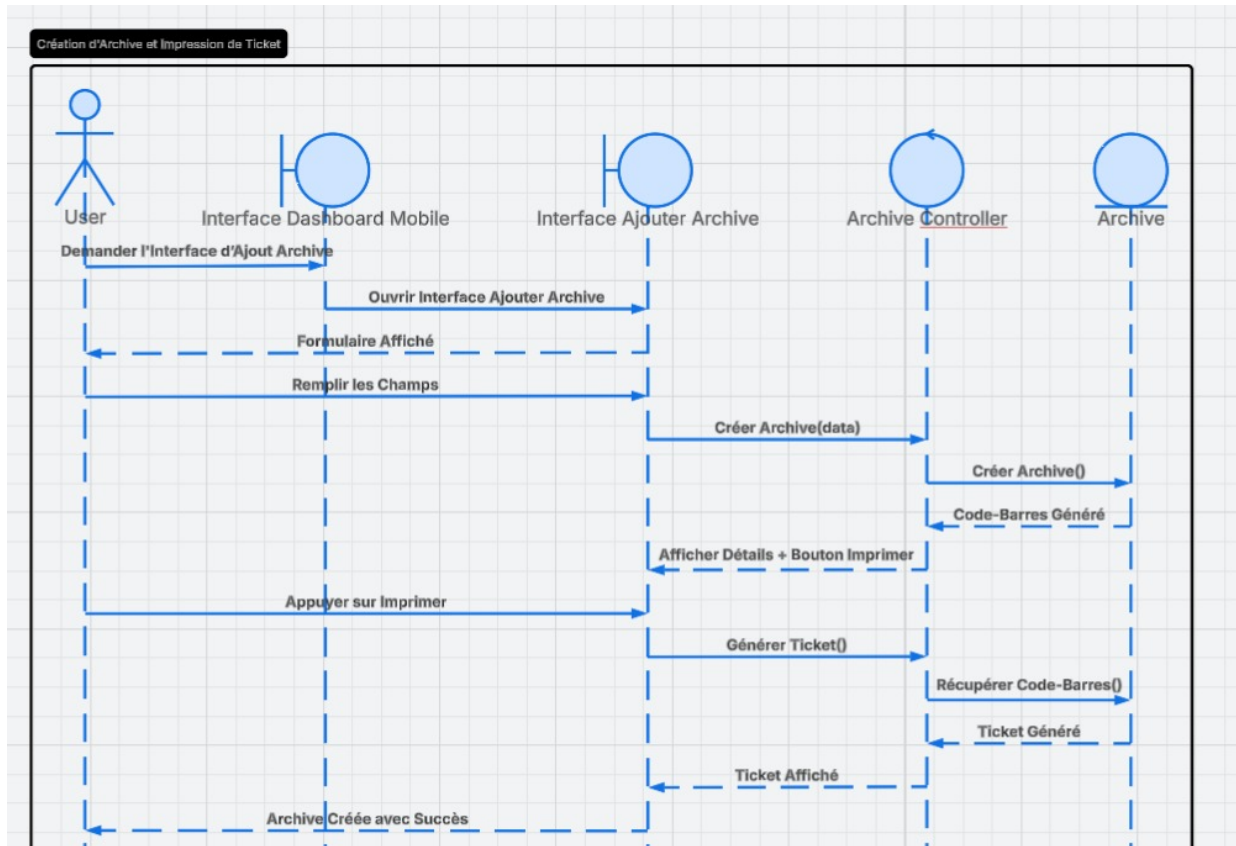


FIGURE 3.5 – Diagramme de séquence : Ajout d'une archive et impression du ticket

Authentification Le diagramme de la figure 3.6 détaille la procédure de connexion d'un utilisateur (administrateur, archiviste ou consultant).

Déroulement : L'utilisateur saisit son CIN (huit chiffres) et son mot de passe sur l'interface de connexion. L'application envoie ces identifiants au backend (Auth Controller) via une requête POST /auth/login.

Le backend interroge la base SQL Server pour vérifier l'existence du CIN dans la table Utilisateur. Si le CIN est trouvé, il compare le mot de passe fourni avec le hash stocké (bcrypt). Une fois l'utilisateur validé utilisateurValide(), le backend récupère le rôle associé (recupererRole(), roleUtilisateur()).

Enfin, le backend génère un jeton JWT (JSON Web Token) contenant les informations de l'utilisateur et de son rôle, puis le retourne à l'application. Ce jeton est stocké localement : dans les SharedPreferences pour l'application mobile, ou dans le localStorage pour le backoffice web.

L'utilisateur est alors redirigé vers son tableau de bord (accesDashboard()). Si les identifiants sont incorrects ou si le compte est désactivé, un message d'erreur approprié est affiché.

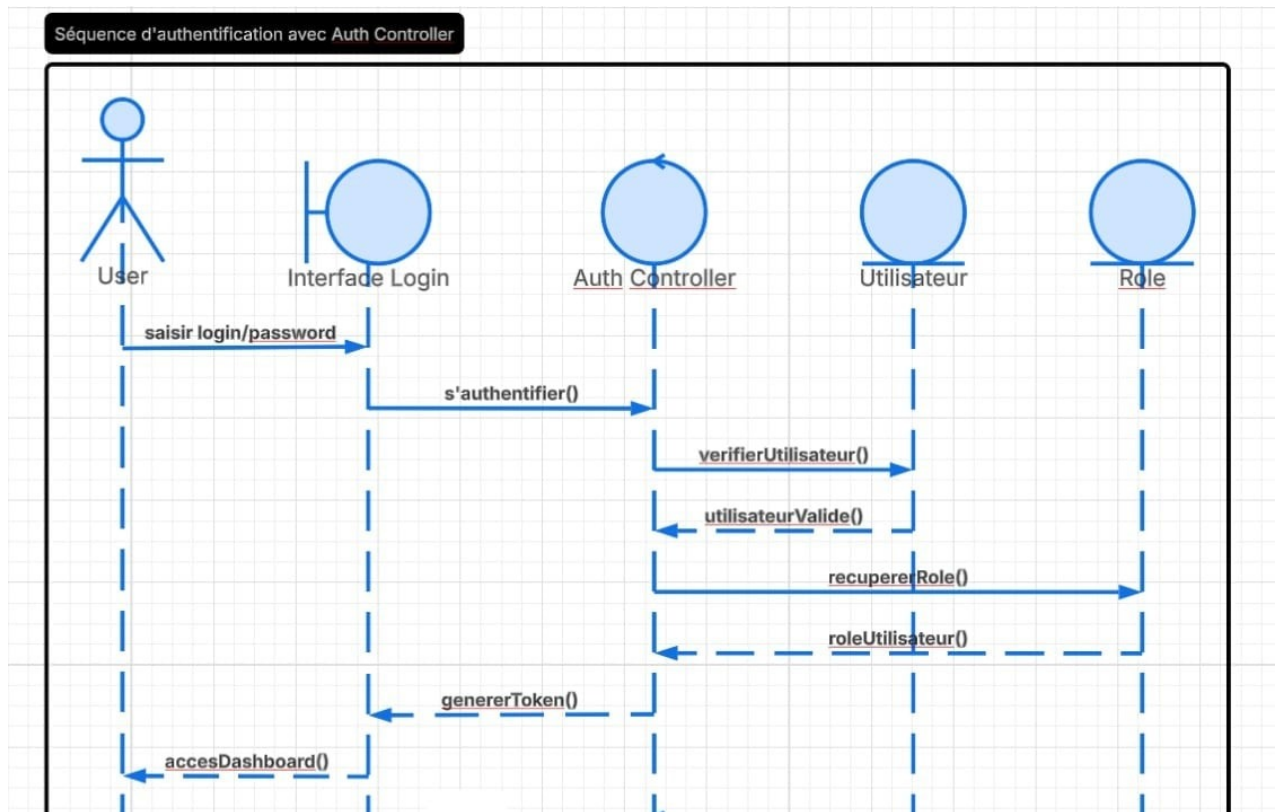


FIGURE 3.6 – Diagramme de séquence : Authentification avec récupération du rôle

Recherche d'archive La figure 3.7 présente les différentes méthodes de recherche d'une archive offertes à l'archiviste.

Accès et choix de la méthode : Depuis le tableau de bord mobile, l'utilisateur demande l'interface de recherche. L'application affiche trois options : scanner un code-barres (via la caméra), saisir le code manuellement, ou choisir une image de code-barres depuis la galerie.

Scan par caméra : L'utilisateur cadre le code-barres imprimé sur l'archive. L'application lit le code et l'envoie au backend.

Saisie manuelle : L'utilisateur tape directement le code-barres (ex : ARCH_20260420_00011) dans un champ de texte.

Recherche par image : L'utilisateur sélectionne une photo contenant un code-barres depuis la galerie. L'application extrait le code de l'image (via un module de reconnaissance optique) et le transmet au backend.

Traitement et résultat : Quelle que soit la méthode choisie, l'utilisateur clique sur "Rechercher". L'application envoie une requête GET `/archives?barcode=...` au backend. Ce dernier interroge la base SQL Server (tables `codes_barres` puis `archives`). Si l'archive est trouvée, le backend renvoie l'ensemble de ses informations (zone, emplacement, titre, type, date de création, créateur). L'application mobile affiche alors les détails complets. Si le code est invalide ou inexistant, un message d'erreur approprié est affiché.

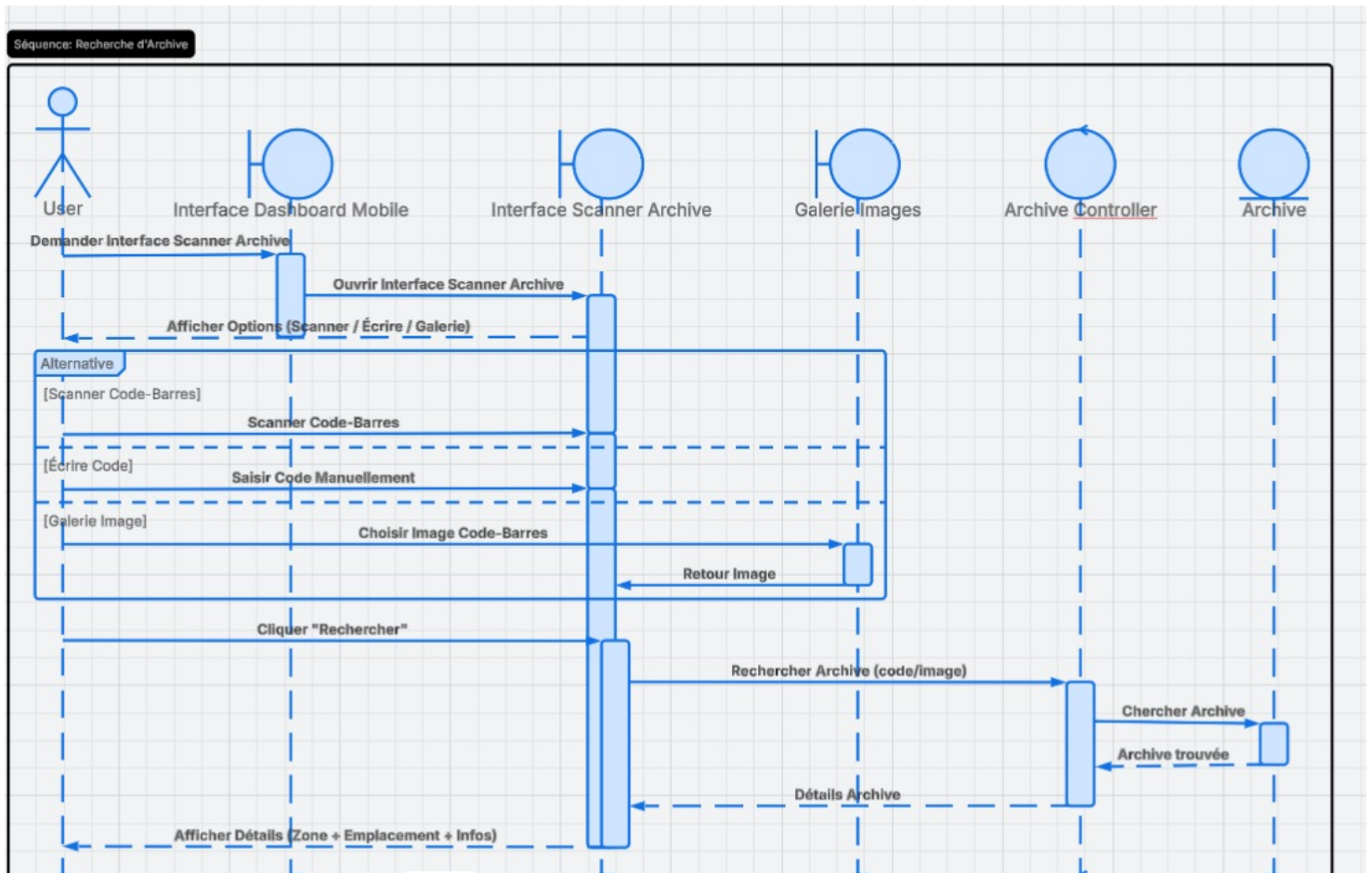


FIGURE 3.7 – Diagramme de séquence : Recherche d'archive (scan, saisie, galerie)

Déplacement d'une archive Le diagramme de la figure 3.8 décrit la procédure de mise à jour de l'emplacement et de la zone d'une archive existante.

Déroulement : L'archiviste ouvre l'interface de scan depuis l'application mobile et cadre le code-barres de l'archive à déplacer. L'application transmet le code au backend (`GET /archives?barcode=...`). Après vérification de l'authentification, le backend interroge la base SQL Server. Si l'archive est trouvée, l'application affiche ses informations actuelles : zone, emplacement, titre, type, etc.

L'archiviste modifie alors la zone et l'emplacement (par exemple, via des listes déroulantes). Il valide la modification. L'application envoie une requête `POST /archives/deplacer` (ou `PUT`) contenant le code-barres, le nouvel emplacement et la nouvelle zone.

Le backend procède à deux vérifications successives :

- Il s'assure d'abord que la nouvelle zone est active (colonne `is_active` à 1).
- Il vérifie ensuite que le nouvel emplacement est valide (existe et actif).

Si les deux vérifications sont positives, le backend exécute une requête `UPDATE` sur la table `archives` pour modifier les colonnes `id_zone` et `id_emplacement`. Une fois la mise à jour effectuée, le backend retourne un message de succès ("Modification enregistrée"). L'application mobile affiche alors une notification de réussite et redirige l'utilisateur vers l'écran précédent ou affiche les nouvelles coordonnées de l'archive.

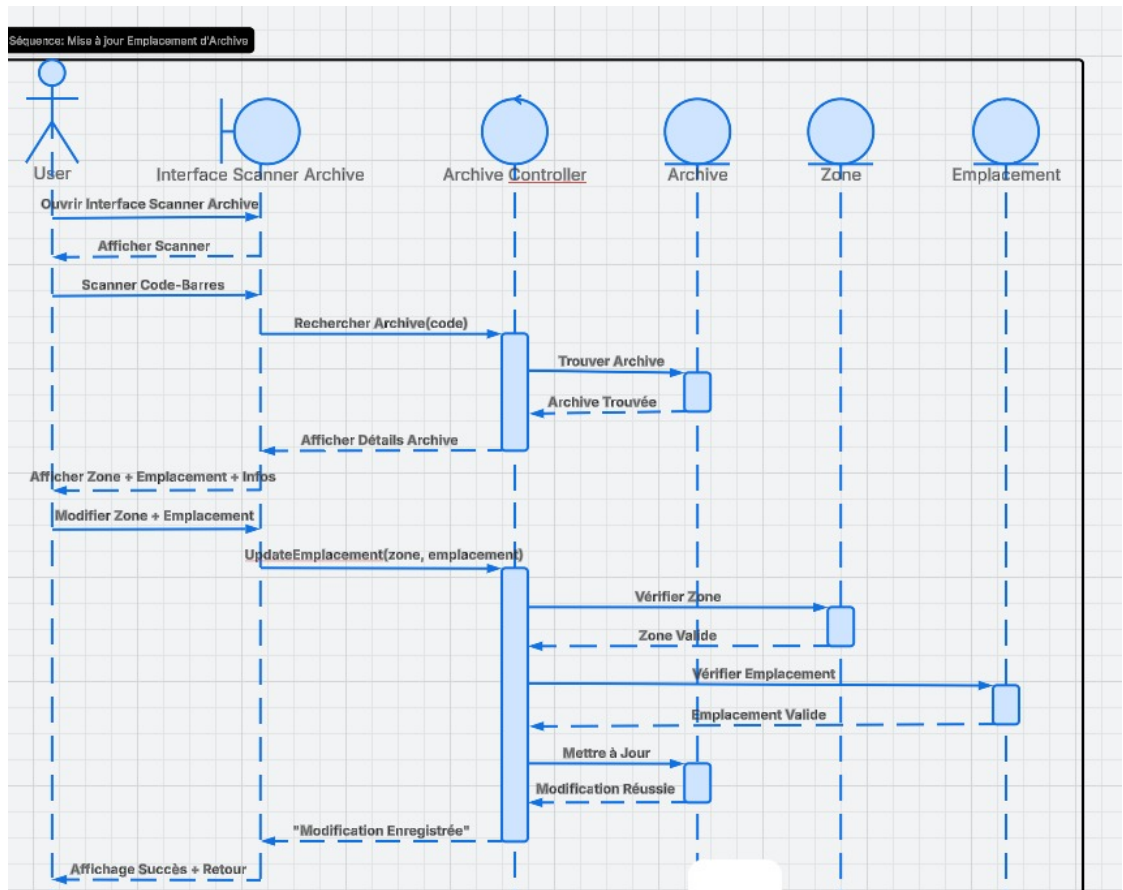


FIGURE 3.8 – Diagramme de séquence : Déplacement d'archive zone et emplacement

3.3.6 Résultats obtenus à la fin du Sprint 3

À la fin du Sprint 3, nous avons livré une application mobile fonctionnelle, intuitive et synchronisée en temps réel avec le backend via des API REST sécurisées.

1. Authentification L'utilisateur saisit son CIN et son mot de passe pour accéder à l'application (figure 3.9).

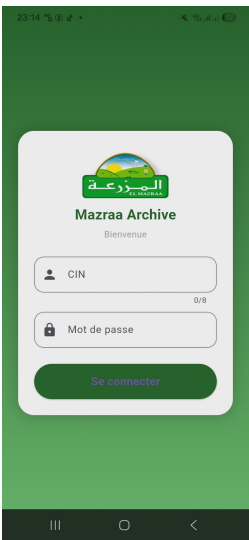


FIGURE 3.9 – Écran de connexion

2. Ajout d’une archive L’archiviste remplit le formulaire (titre, type, emplacement, zone). Le système génère un code-barres unique et le sauvegarde (figure 3.10).

FIGURE 3.10 – Formulaire d’ajout d’une archive

3. Code-barres généré Une fois l’archive créée, le système affiche le code-barres unique (format ARCH_AAAAMJJ_NNNNN) et permet son téléchargement en PDF (figure 3.11).

FIGURE 3.11 – Code-barres généré pour l’archive

FIGURE 3.12 – Affichage du code-barres généré

4. Scan par caméra L’utilisateur cadre le code-barres avec la caméra du téléphone. L’application lit le code et affiche immédiatement les informations de l’archive (figure 3.13).

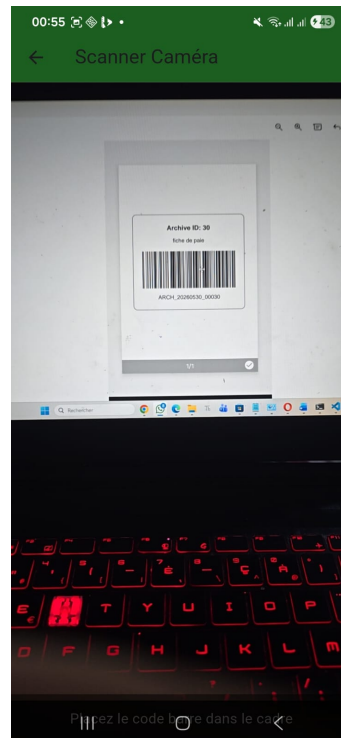


FIGURE 3.13 – Scan de code-barres par caméra

5. Recherche par code-barres L'archiviste peut également saisir manuellement un code-barres ou en sélectionner un depuis la galerie pour retrouver une archive (figure 3.14).



FIGURE 3.14 – Recherche d'archive par code-barres

6. Déplacement d'archive Après avoir trouvé l'archive, l'utilisateur modifie sa zone et son emplacement. Le backend vérifie que la nouvelle zone est active avant de valider (figure 3.15).



FIGURE 3.15 – Déplacement d’archive (zone et emplacement)

Tests et performances Les tests ont montré qu’une archive est retrouvée en 1 à 3 secondes, ce qui est acceptable pour un usage quotidien. Le livrable du Sprint 3 répond ainsi aux besoins de l’entreprise EL MAZRAA.

3.3.7 Bilan du Sprint 3 (Sprint Review)

À la fin du Sprint 3, toutes les fonctionnalités prévues ont été développées et testées avec succès. L’application mobile permet désormais de scanner, ajouter et consulter des archives. L’interaction avec le backend et la base de données fonctionne correctement. Ce sprint a donc validé l’intégration complète de l’application mobile.

3.3.8 Difficultés rencontrées et solutions apportées

Problèmes rencontrés Pendant le Sprint 3, nous avons rencontré plusieurs difficultés : la connexion entre l’application mobile et le backend était instable selon le réseau, l’ajout d’une archive dans une zone désactivée ne signalait pas d’erreur, le scan de codes-barres avait du mal avec les codes illisibles, et des retards de synchronisation sont apparus.

Solutions apportées Pour y remédier, nous avons figé l’adresse IP du serveur pour stabiliser la connexion, renforcé les contrôles backend (vérification des zones actives), amélioré le module de scan avec des messages d’erreur clairs, et effectué des tests supplémentaires pour optimiser la synchronisation.

Conclusion

Ce chapitre a détaillé l’organisation du projet en trois sprints, avec un focus sur le Sprint 3 (application mobile). Les fonctionnalités livrées ont été validées, confirmant la fiabilité du système. Le prochain chapitre présente les tests de validation et l’évaluation des performances.

Chapitre 4

Résultats finaux et validation

Table des matières

4.1	Introduction	47
4.2	Validation de l'API	47
4.2.1	Environnement de test	47
4.2.2	Principales API développées	47
4.2.3	Scénarios de test de l'API	47
4.2.4	Détail des tests avec captures d'écran	48
4.2.5	Bilan des tests	53
4.3	Ce que l'utilisateur voit concrètement	53
4.3.1	Le backoffice web (gestion centralisée)	53
4.3.2	L'application mobile (terrain)	54
4.4	Performances constatées	54
4.5	Types de tests et améliorations apportées	54
4.6	Conclusion	54

4.1 Introduction

Ce chapitre décrit les fonctionnalités finales de notre système de gestion d'archives. Il montre aussi les tests que nous avons réalisés pour vérifier que l'API, l'application mobile et le backoffice web fonctionnent correctement.

4.2 Validation de l'API

Nous avons testé les principales routes de notre backend (Node.js / Express) avec l'outil Thunder Client. Ces tests portent sur l'authentification, la création d'archive, la recherche, le déplacement, et la gestion des erreurs (exemple : zone inactive).

4.2.1 Environnement de test

Nous avons travaillé en local : le serveur tourne sur `http://localhost:3000`, la base est SQL Server, et nous utilisons des jetons JWT pour sécuriser les échanges.

4.2.2 Principales API développées

Le tableau 4.1 résume les points d'entrée essentiels de notre API.

Endpoint	Méthode	Description
/auth/login	POST	Authentification (CIN + mot de passe) → retourne un jeton JWT.
/archives	GET	Recherche d'archives par mot-clé ou code-barres.
/archives	POST	Ajout d'une archive → génération automatique du code-barres.
/archives/deplacer	POST	Déplacement d'une archive (vérifie que la zone est active).
/utilisateurs	GET	Liste des utilisateurs (admin uniquement).
/zones	GET	Liste des zones de stockage.

TABLE 4.1 – Principales API REST développées

4.2.3 Scénarios de test de l'API

Nous avons testé plusieurs cas importants. Le tableau 4.2 résume les résultats : chaque test a fonctionné comme attendu.

Test	Entrée	Résultat attendu	Résultat obtenu
Authentification	CIN + mot de passe valides	Token JWT	OK
Ajout archive valide	Données complètes	Succès (201 Created)	OK
Ajout archive invalide	Champs manquants	Erreur 400	OK
Zone inactive	zoneId = 2	Erreur 400	OK
Recherche archive	Mot-clé ou code-barres	Résultats retournés	OK
Déplacement archive	code-barres + nouvel emplacement + zone active	Succès (200 OK)	OK
Accès non autorisé	token non admin	Erreur 403	OK

TABLE 4.2 – Scénarios de test de l'API

4.2.4 Détail des tests avec captures d'écran

4.2.4.1 Authentification

Pour nous assurer que la connexion fonctionne, nous avons appelé l'endpoint `POST /auth/login` avec un CIN et un mot de passe corrects (figure 4.1). Le backend a répondu **200 OK** et a renvoyé un jeton JWT. Ce jeton est ensuite utilisé par l'application (mobile ou web) pour prouver son identité lors des appels suivants. L'authentification est donc opérationnelle et sécurisée.

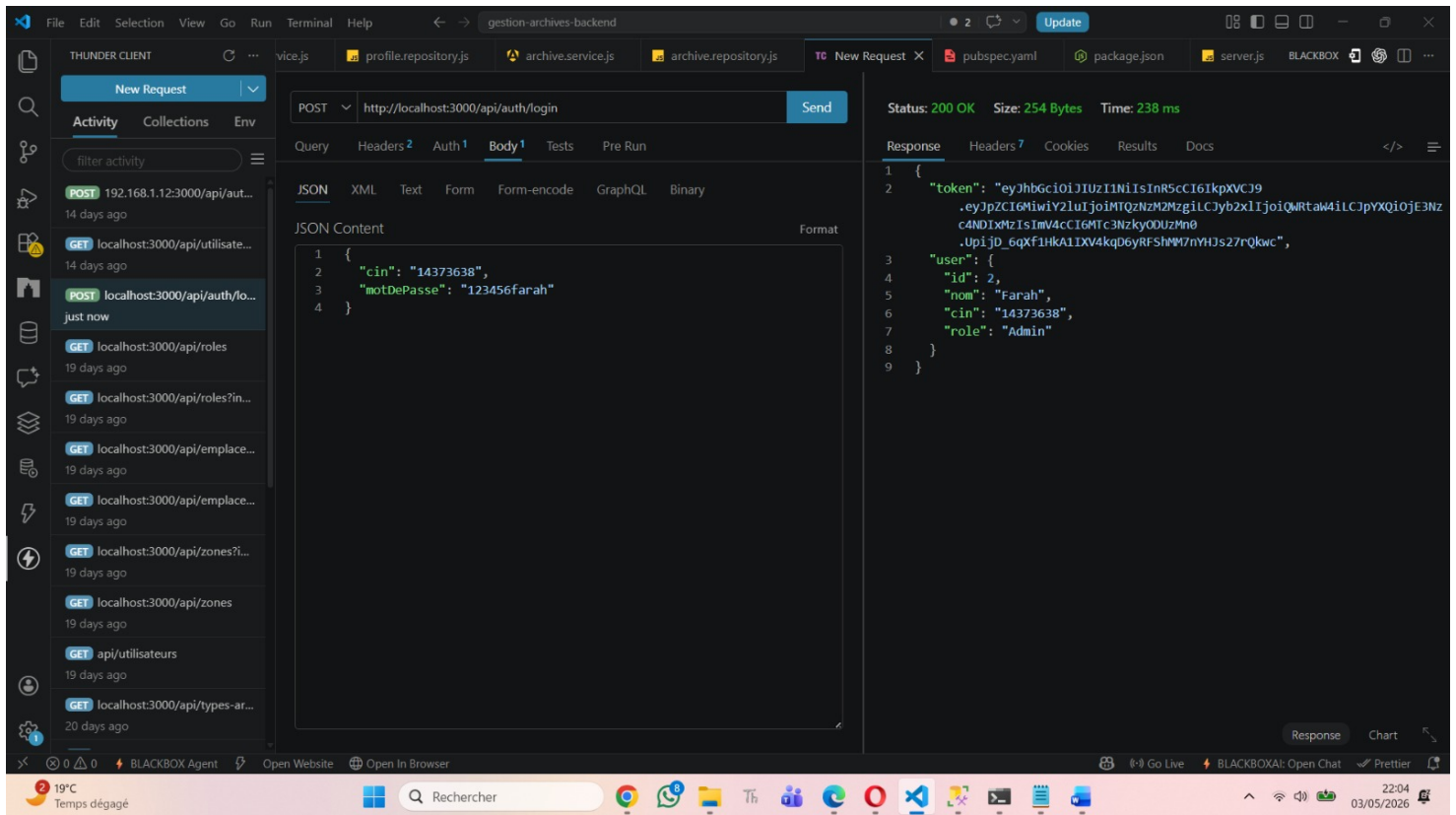


FIGURE 4.1 – Authentification réussie – obtention du jeton JWT

4.2.4.2 Ajout d'une archive

Nous avons testé l'ajout d'archive via l'endpoint `POST /archives`. Quand toutes les informations (titre, type, emplacement, zone) sont fournies, le backend répond avec un statut **201 Created** et retourne un code-barres unique (format `ARCH_AAAAMMJJ_NNNNN`), comme le montre la figure 4.2. À l'inverse, si le titre ou le type d'archive manque, le serveur rejette la requête avec une erreur **400 Bad Request** et le message "titre et typeId sont obligatoires" (figure 4.3). Cela prouve que nos validations fonctionnent correctement.

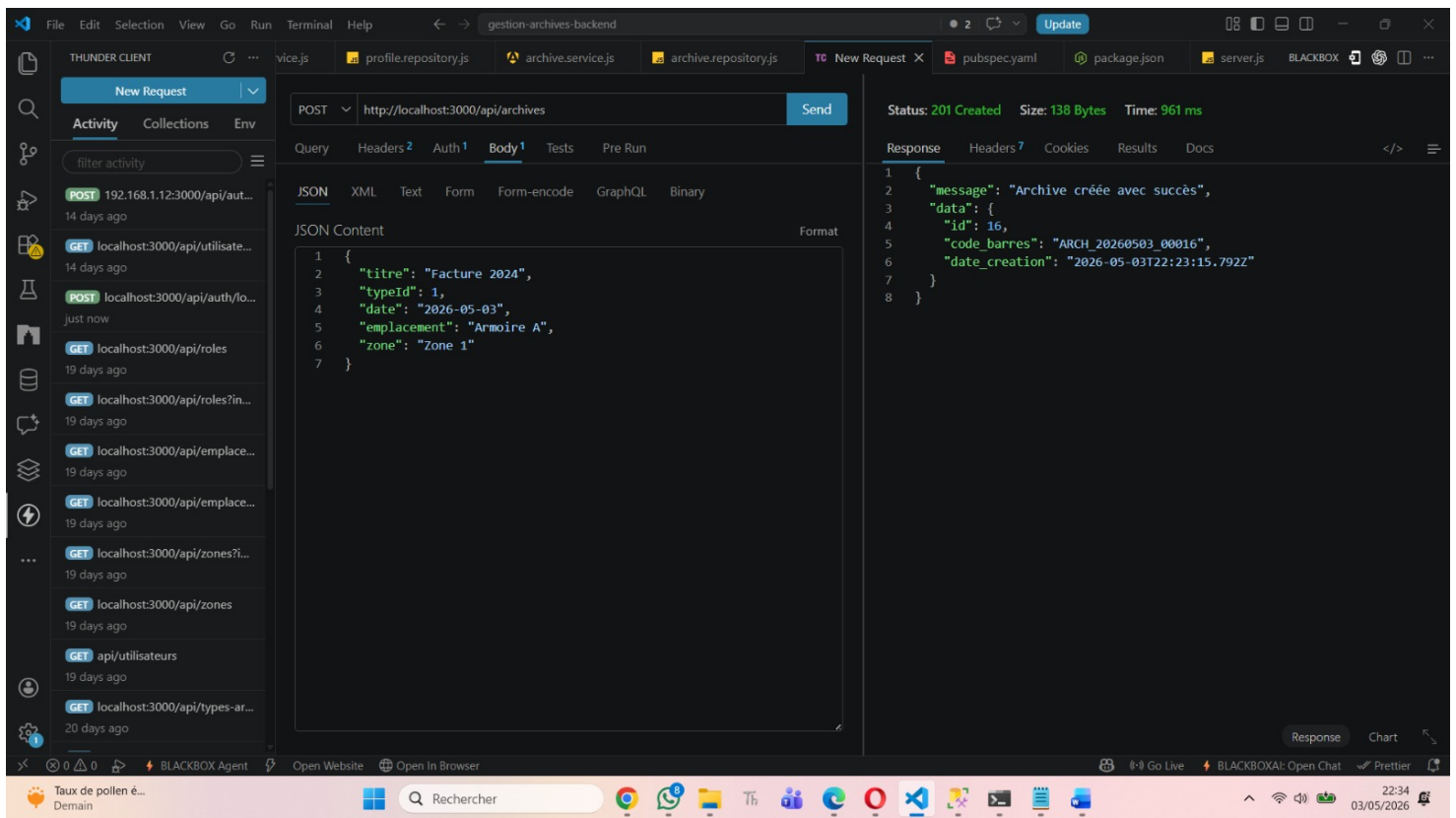


FIGURE 4.2 – Ajout réussi – code-barres généré

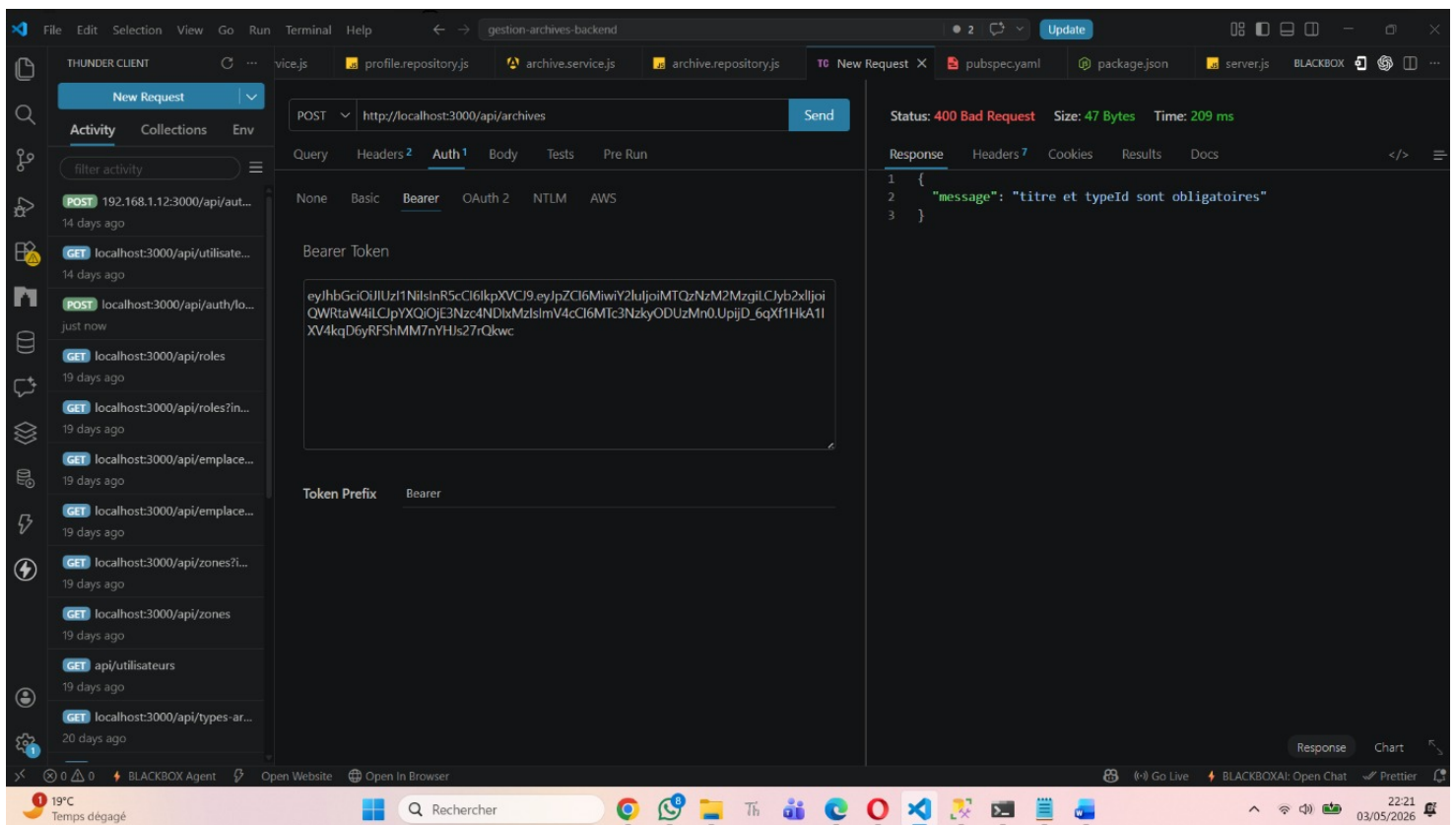


FIGURE 4.3 – Erreur 400 – titre ou type manquant

4.2.4.3 Validation des zones actives

Une règle importante de notre système est que l'on ne peut pas ranger une archive dans une zone désactivée. Nous avons donc essayé d'ajouter une archive avec `zoneId: 2` (zone inactive). Le backend a refusé la requête avec un statut **400 Bad Request** et le message "La zone sélectionnée est inactive" (figure 4.4). Le respect de cette règle évite les incohérences de stockage.

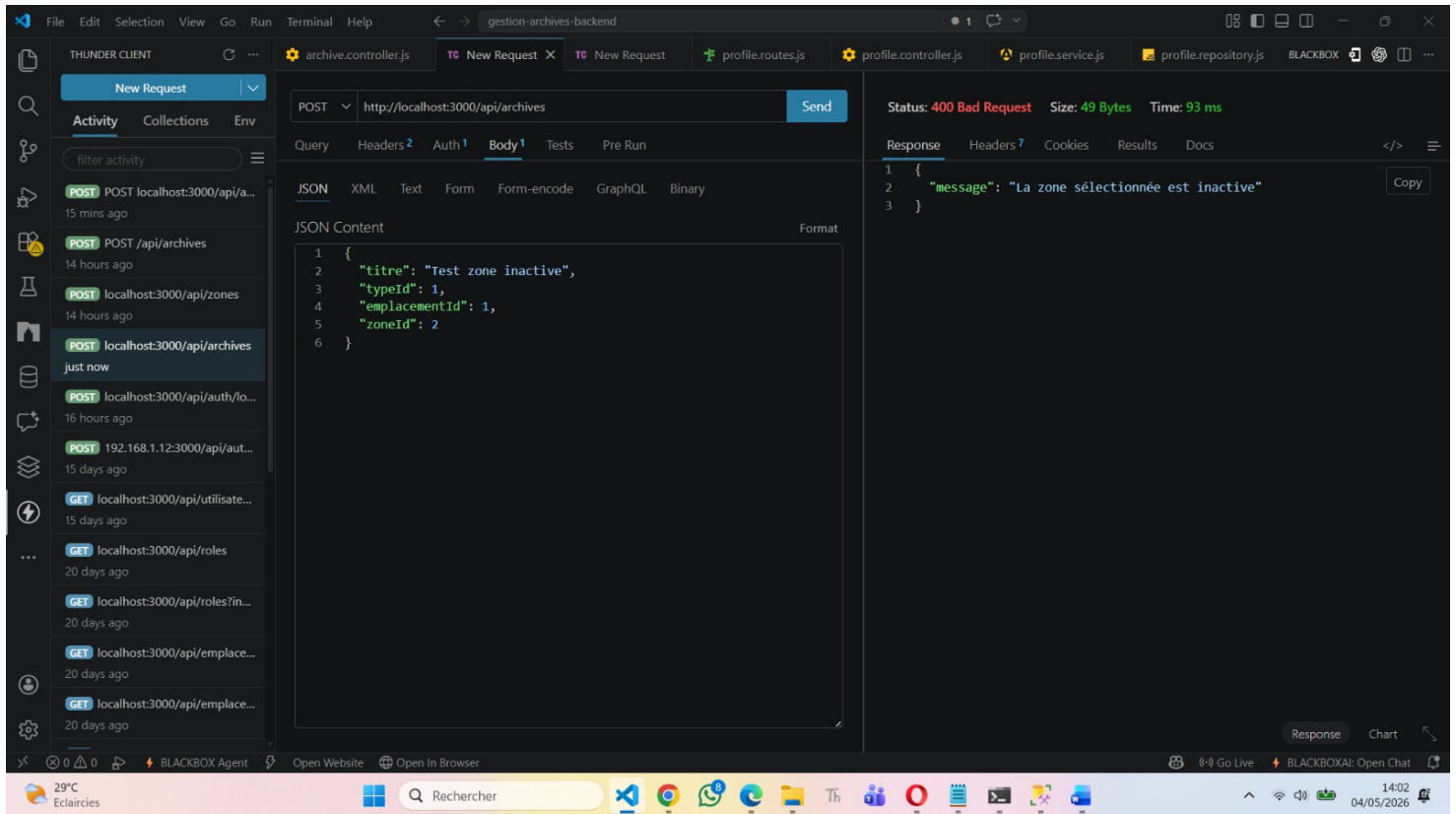


FIGURE 4.4 – Refus car zone inactive

4.2.4.4 Recherche d'archives

Pour tester la recherche, nous avons appelé `GET /archives?search=test` avec un jeton JWT valide. Le backend a renvoyé un statut **200 OK** et une liste d'archives dont le titre ou la description contient "test" (figure 4.5). Chaque résultat affiche le titre, le type, l'emplacement, la zone, le code-barres et le créateur. Le système utilise aussi la pagination pour éviter de charger trop de données à la fois – ce qui améliore les performances.

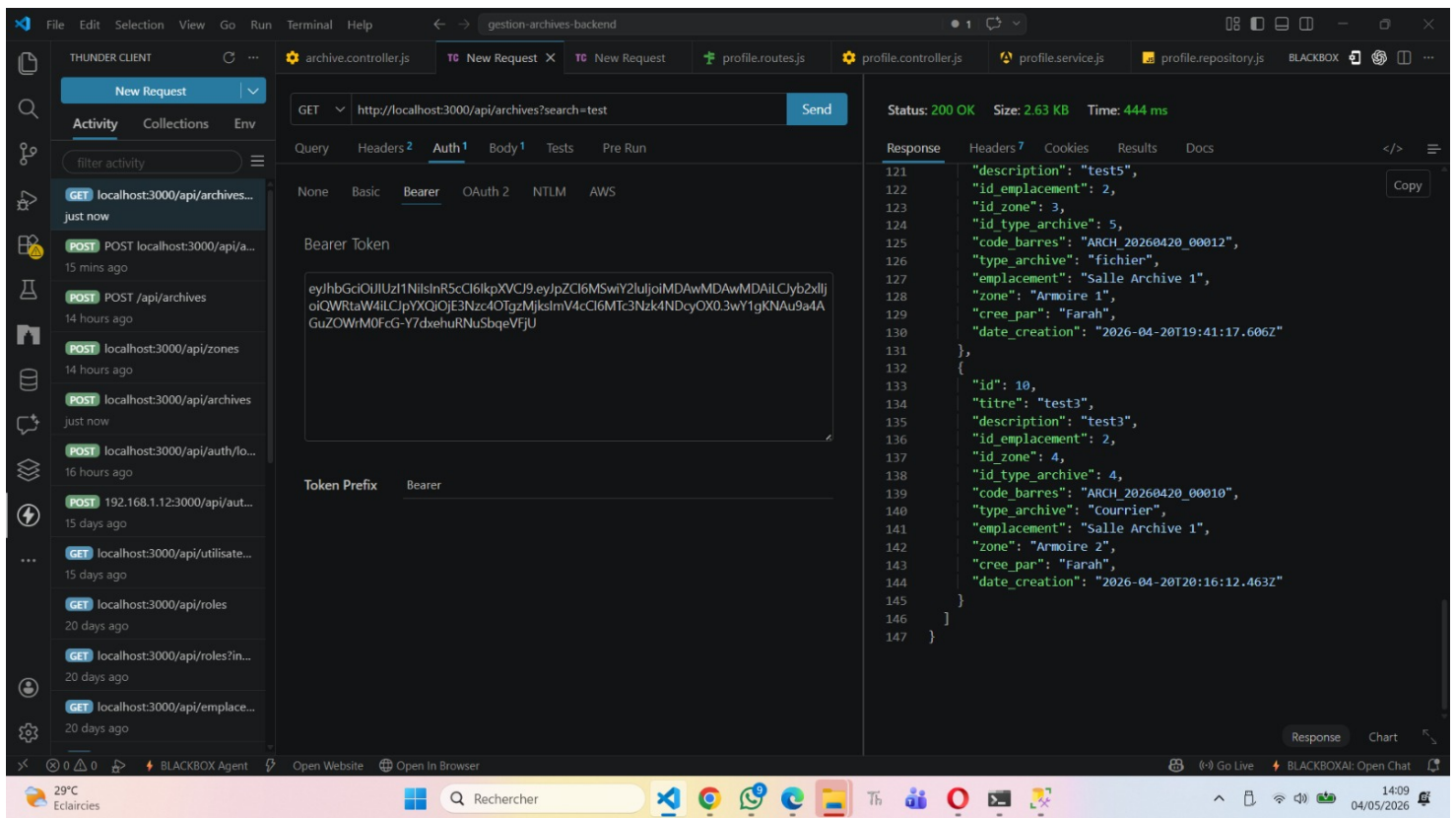


FIGURE 4.5 – Résultats de recherche avec pagination

4.2.4.5 Déplacement d'une archive

Nous avons également testé le déplacement d'une archive via l'endpoint `POST /archives/deplacer`. Pour cela, nous avons d'abord recherché une archive existante (par exemple `ARCH_20260420_00011`), puis nous avons envoyé une requête avec un nouvel emplacement et une nouvelle zone active. Le backend a retourné un statut **200 OK** et confirmé la mise à jour dans la base SQL Server. Ce test montre que la règle de validation des zones actives est également appliquée lors d'un déplacement.

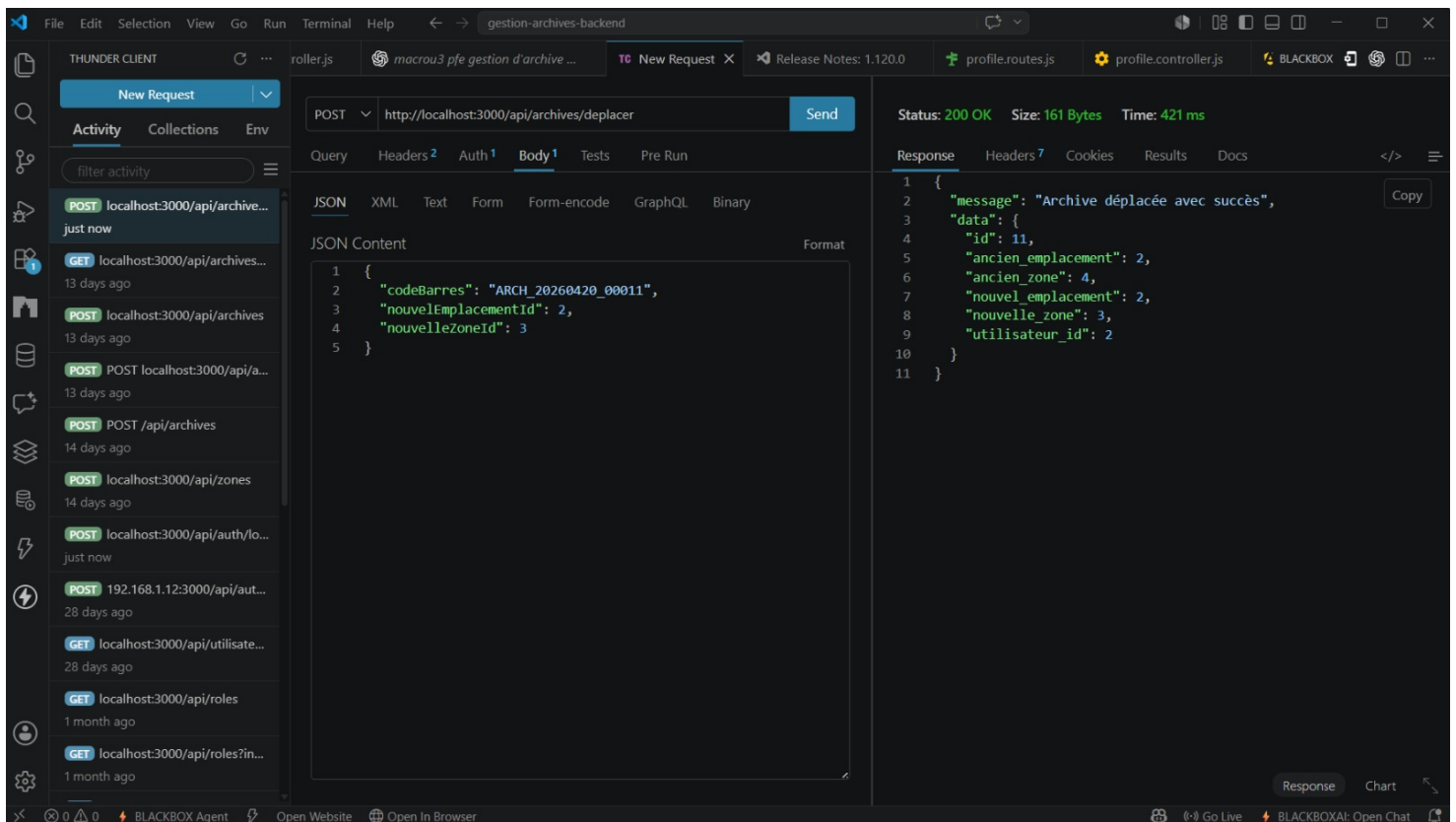


FIGURE 4.6 – Déplacement réussi d’une archive

4.2.4.6 Contrôle d’accès et token invalide

Pour vérifier que notre API est bien sécurisée, nous avons tenté d’appeler l’endpoint `GET /utilisateurs` (réservé aux administrateurs) avec un token JWT valide mais appartenant à un simple archiviste. Le backend a retourné une erreur **403 Forbidden** avec le message “Accès non autorisé”. De même, une requête sans token ou avec un token expiré renvoie une erreur **401 Unauthorized**. Ces tests confirment que notre système d’authentification et de gestion des rôles est fiable.

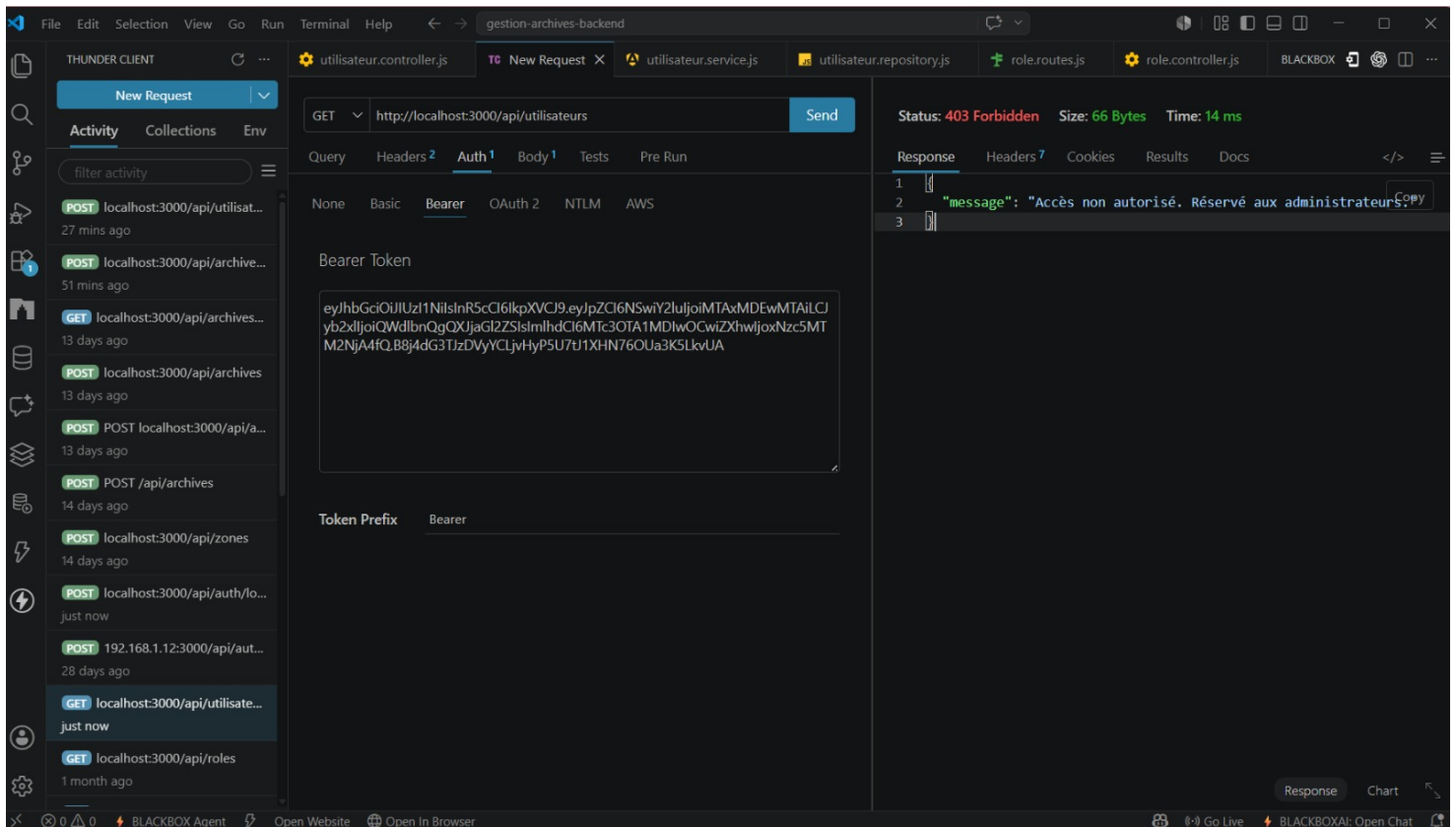


FIGURE 4.7 – Accès refusé pour un utilisateur non administrateur

4.2.5 Bilan des tests

Tous les tests que nous avons menés ont donné les résultats attendus. Les cas normaux (authentification, ajout, recherche, déplacement) fonctionnent parfaitement. Les erreurs volontaires (champs manquants, zone inactive, token absent) sont bien détectées et rejetées avec des messages clairs. L'API est donc fiable et sécurisée.

4.3 Ce que l'utilisateur voit concrètement

Notre système propose deux interfaces accessibles selon le rôle de l'utilisateur.

Backoffice web (React) : Il est utilisé par l'**administrateur** (accès complet) et par l'**archiviste** (accès limité). L'administrateur peut gérer les utilisateurs, les rôles, les archives, les types, les emplacements et les zones. L'archiviste peut consulter les archives, gérer les types, les emplacements et les zones, et visualiser le tableau de bord.

Application mobile (Flutter) : Elle est utilisée par l'**archiviste** sur le terrain pour ajouter, scanner, rechercher, consulter et déplacer des archives. Le **consultant** peut également y accéder en lecture seule.

Les deux interfaces communiquent avec la même API REST.

4.3.1 Le backoffice web (gestion centralisée)

Voici un aperçu de l'écran de gestion des archives dans le backoffice (figure 4.8). On y consulte, ajoute ou modifie les archives facilement.

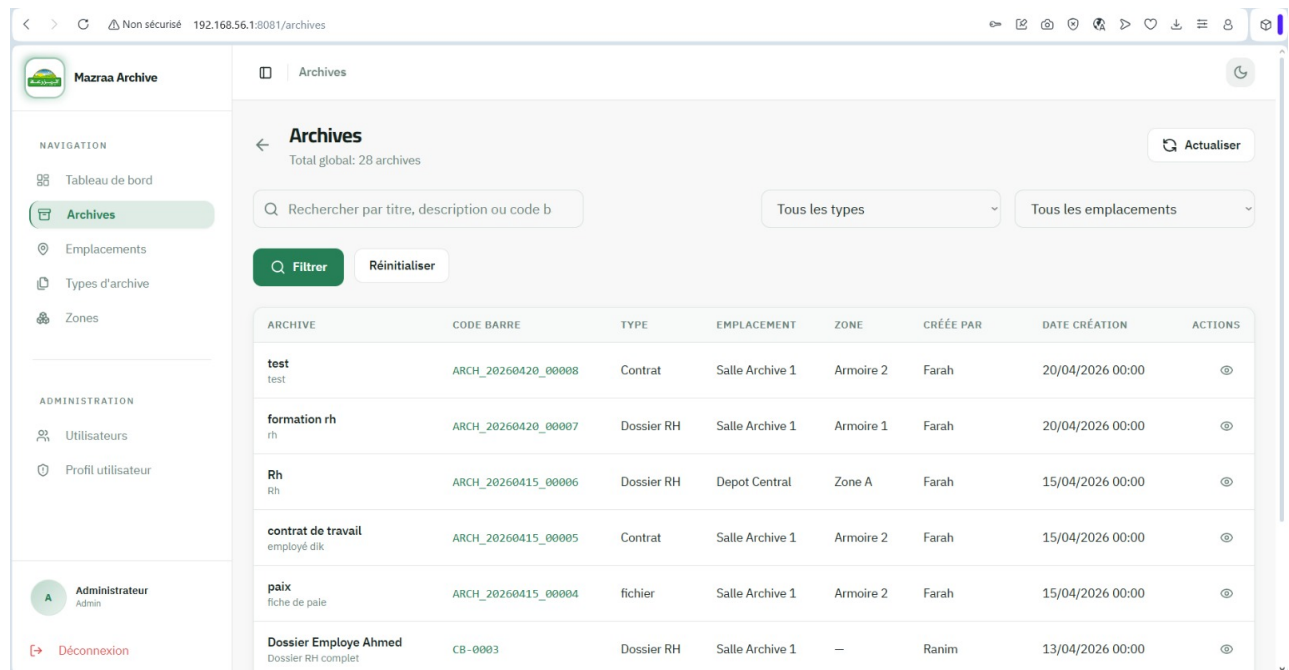


FIGURE 4.8 – Interface de gestion des archives – backoffice web

4.3.2 L'application mobile (terrain)

L'application mobile permet de s'authentifier, d'ajouter une archive, de scanner son code-barres, de rechercher et de déplacer des archives. Toutes ces opérations sont synchronisées en temps réel avec le backend via l'API REST. Les captures d'écran détaillées de l'application mobile sont présentées dans le chapitre 3 (section 3.13 et suivantes).

4.4 Performances constatées

Nous avons mesuré plusieurs choses. Le temps de réponse moyen des requêtes API est inférieur à 500 millisecondes, ce qui est confortable pour une utilisation quotidienne. Grâce aux codes-barres, la recherche d'une archive ne prend que quelques secondes. Les mécanismes de validation garantissent que les données restent propres. La pagination (page, limit) évite de surcharger le réseau. Enfin, l'architecture REST permet d'ajouter facilement de nouvelles fonctionnalités plus tard.

4.5 Types de tests et améliorations apportées

Nous avons fait des tests fonctionnels (vérifier que l'ajout, la recherche, le scan marchent), des tests d'intégration (communication entre frontend, backend et base de données), des tests de validation (règles métier comme la zone active) et des tests avec de vrais utilisateurs pour l'ergonomie. Ces essais nous ont aidés à améliorer la synchronisation des données, la clarté des messages d'erreur et la fluidité des interfaces. Globalement, la qualité du système s'en est trouvée renforcée.

4.6 Conclusion

Ce chapitre a présenté les fonctionnalités finales du système et les tests qui prouvent leur bon fonctionnement. Les résultats montrent que notre solution répond bien au besoin de l'entreprise EL MAZRAA : elle est rapide, fiable et sécurisée pour gérer les archives.

Conclusion générale

La gestion des archives constitue un pilier fondamental pour toute organisation souhaitant garantir la fiabilité de ses informations, assurer la continuité de ses activités et optimiser ses processus internes. Dans un contexte où le volume documentaire ne cesse de croître, la transformation numérique s'impose comme une nécessité stratégique plutôt qu'un simple choix technologique.

C'est dans cette perspective que s'est inscrit notre projet de fin d'études, réalisé au sein de l'entreprise EL MAZRAA. Partant d'un constat alarmant — un système d'archivage entièrement manuel, source de pertes de temps estimées à plus de 800 heures par an, d'absence de traçabilité et de risques documentaires importants — nous avons conçu et développé une solution numérique complète basée sur l'identification par codes-barres.

Sur le plan technique, la solution livrée repose sur une architecture client-serveur à trois couches. Une **application mobile Android** (Flutter) permet aux archivistes de scanner, ajouter, rechercher et déplacer des archives sur le terrain, avec génération automatique de codes-barres uniques et téléchargement d'étiquettes PDF. Un **backoffice web** (React / Tailwind CSS) offre à l'administrateur une gestion centralisée des utilisateurs, des rôles, des types d'archives, des emplacements et des zones de stockage. Enfin, une **API REST sécurisée** (Node.js / Express) assure la communication entre les deux interfaces et la base de données SQL Server, avec authentification JWT et contrôle d'accès basé sur les rôles (RBAC).

Sur le plan méthodologique, l'adoption de la méthode Agile Scrum a structuré notre travail en trois sprints progressifs, favorisant une livraison incrémentale, une meilleure communication avec les parties prenantes et une adaptation continue aux retours de l'encadrant industriel.

Sur le plan des résultats, les tests menés ont confirmé l'efficacité de la solution : le temps de recherche d'une archive est passé de 15 à 30 minutes à moins de 3 secondes, soit une réduction de plus de **99%**. La traçabilité complète des mouvements, la sécurisation des accès et la fiabilité des données constituent des apports majeurs pour l'entreprise EL MAZRAA.

Sur le plan personnel, ce projet nous a permis de mobiliser et d'approfondir les compétences acquises durant notre formation, notamment en développement web et mobile, en conception de bases de données relationnelles, en modélisation UML et en gestion de projet Agile. Il nous a également confrontés à des défis réels d'entreprise, renforçant ainsi notre capacité d'analyse, d'adaptation et de résolution de problèmes.

Concernant les perspectives d'amélioration, plusieurs axes d'évolution peuvent enrichir cette solution à l'avenir. L'intégration d'un module **OCR** permettrait de numériser et d'indexer automatiquement le contenu des documents papier. Le passage aux **QR Codes** ou à la technologie **RFID** offrirait une identification encore plus rapide avec une capacité de stockage d'informations plus importante. Le déploiement sur une infrastructure **Cloud** améliorerait l'accessibilité, la disponibilité et la sauvegarde automatique des données. L'ajout d'un système de **notifications en temps réel** alerterait les responsables lors de mouvements sensibles ou d'anomalies détectées. Enfin, le développement d'un module de **statistiques avancées** et de rapports exportables faciliterait les audits et la prise de décision.

En définitive, ce projet démontre qu'une solution numérique bien conçue, même avec des technologies accessibles, peut transformer profondément l'organisation d'une entreprise. Il constitue une base solide et évolutive sur laquelle EL MAZRAA pourra s'appuyer pour moderniser davantage sa gestion documentaire et renforcer sa compétitivité dans un environnement en constante mutation.

Annexe A

Description détaillée des sprints

Cette annexe présente les diagrammes et les captures d'écran des sprints 1 et 2 qui n'ont pas été détaillés dans les chapitres précédents. Chaque section est volontairement concise pour éviter les répétitions avec le chapitre 3.

A.1 Sprint 1 : Développement Backend

Objectif

Ce sprint a consisté à développer l'API REST (Node.js/Express), à connecter la base SQL Server, à implémenter l'authentification JWT et à créer les endpoints pour les archives, les utilisateurs, les emplacements et les zones.

Cas d'utilisation

La figure A.1 présente les fonctionnalités backend réalisées : authentification, gestion des utilisateurs, des archives, et validation des zones actives.

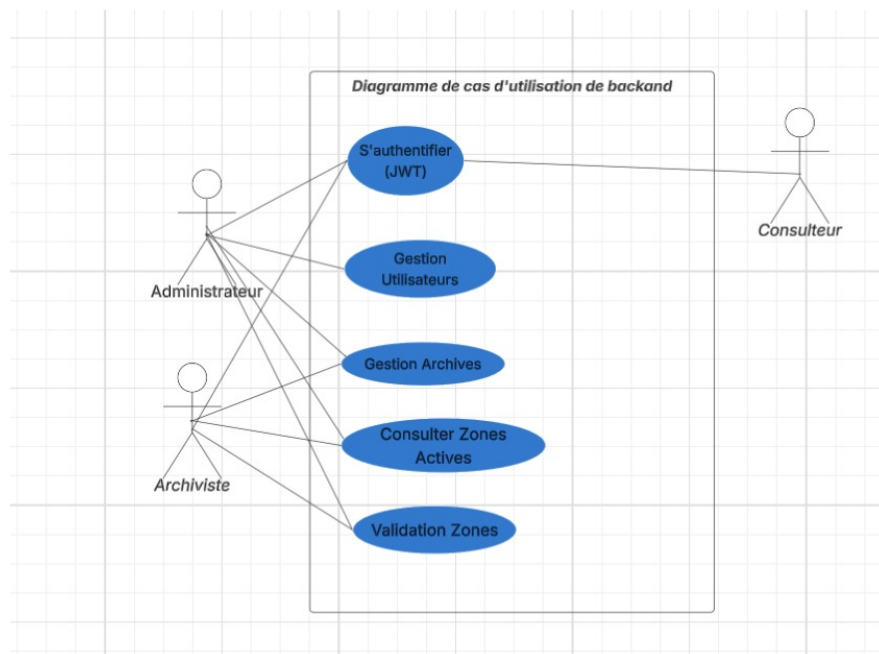


FIGURE A.1 – Cas d'utilisation – Sprint 1

Description textuelle des cas d'utilisation

Cas d'utilisation : S'authentifier

TABLE A.1 – Cas d'utilisation – S'authentifier (Sprint 1)

Acteur	Utilisateur (Admin, Archiviste, Consultant)
Objectif	Obtenir un token JWT pour accéder aux API sécurisées
Précondition	Compte valide (CIN + mot de passe hashé dans SQL Server)
Postcondition	Un token JWT est généré et retourné
Scénario nominal	1. Envoi des identifiants via <code>POST /auth/login</code> . 2. Vérification dans SQL Server. 3. Génération et retour du token.
Scénarios alternatifs	2a. CIN inexistant → message “CIN non trouvé”. 2b. Mot de passe incorrect → message “Mot de passe invalide”.

Cas d'utilisation : Gérer les utilisateurs

TABLE A.2 – Cas d'utilisation – Gérer les utilisateurs (Sprint 1)

Acteur	Administrateur
Objectif	Consulter, ajouter, modifier ou désactiver des comptes
Précondition	Administrateur authentifié (token JWT)
Postcondition	Modifications enregistrées dans SQL Server
Scénario nominal	1. Requête <code>GET /utilisateurs</code> (ou <code>POST / PATCH</code>). 2. Application des modifications. 3. Mise à jour de la base.
Scénarios alternatifs	1a. Token non admin → erreur 403 “Accès non autorisé”. 2a. CIN déjà existant → message “Utilisateur déjà enregistré”.

Remarque

Les cas d'utilisation décrits dans cette section concernent uniquement le backend (sprint 1). Ils complètent les diagrammes de cas d'utilisation déjà présentés dans le chapitre 2 (administration, archiviste, consultant) et ne reprennent pas les détails des sprints 2 et 3 (traités respectivement dans les chapitres 3 et 4).

Découpage des tâches

TABLE A.3 – Répartition des tâches – Sprint 1

Tâche	Priorité	Points
Serveur Node.js / Express	Haute	5
Configuration SQL Server	Haute	5
Authentification JWT	Haute	8
Gestion utilisateurs et rôles	Haute	8
API des archives	Haute	8
Validation des zones actives	Moyenne	5

Résultat

À la fin du sprint, toutes les routes API étaient opérationnelles et testées avec Postman. La communication avec SQL Server fonctionnait correctement.

A.2 Sprint 2 : Développement Backoffice Web

Objectif

Développer l'interface d'administration web (React) pour gérer l'ensemble du système.

Cas d'utilisation

La figure A.2 présente les fonctionnalités disponibles dans le backoffice (gestion des archives, emplacements, zones, types, consultation du tableau de bord, gestion des utilisateurs).

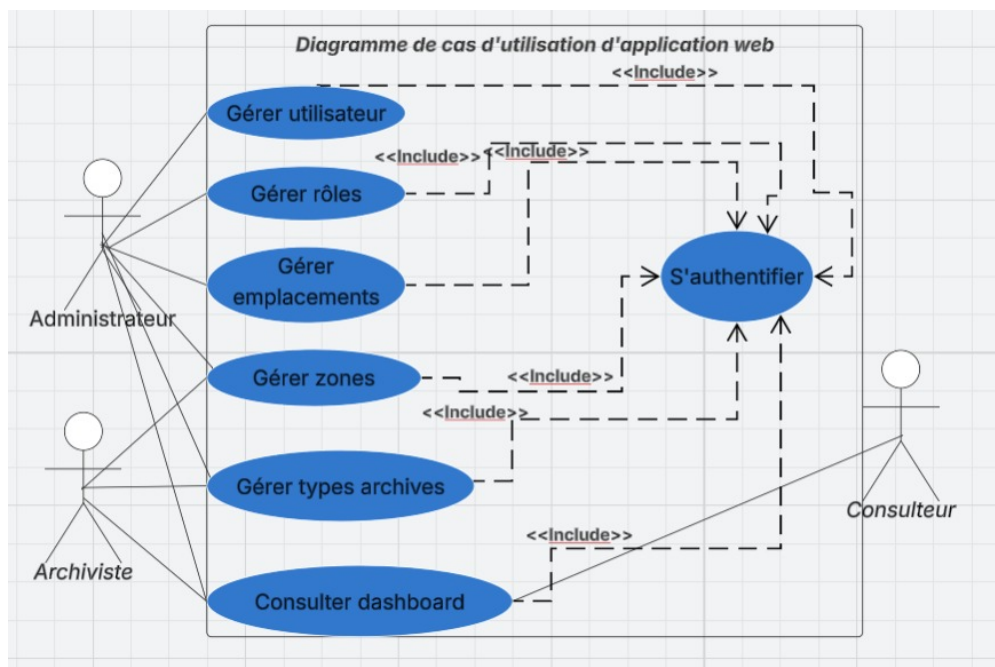


FIGURE A.2 – Cas d'utilisation – Sprint 2

Présentation du diagramme de cas d'utilisation

La figure A.2 illustre les fonctionnalités du backoffice web selon le rôle de l'utilisateur. Ce diagramme complète le diagramme global du chapitre 2 en se concentrant uniquement sur les actions réalisables depuis l'interface web.

Administrateur : Il doit d'abord s'authentifier («*include*» vers "S'authentifier"). Une fois connecté, il peut :

- **Gérer les utilisateurs :** ajouter, modifier ou désactiver des comptes.
- **Gérer les rôles :** attribuer ou modifier les permissions.
- **Gérer les emplacements :** créer, modifier ou désactiver des espaces de stockage.
- **Gérer les zones :** définir les zones et activer/désactiver leur utilisation.
- **Gérer les types d'archives :** ajouter ou modifier les catégories.

Archiviste : Il peut consulter le tableau de bord pour visualiser les indicateurs (nombre d’archives, mouvements récents). Il ne peut ni gérer les utilisateurs ni modifier les paramètres.

Consultant : Il a un accès en lecture seule (consulter et rechercher des archives). Ce rôle n’apparaît pas directement dans ce diagramme car il ne dispose d’aucune action spécifique dans le backoffice web en dehors de l’authentification.

Toutes les actions sont soumises au contrôle d’accès via token JWT. Les cas d’utilisation “Gérer archives”, “Rechercher archives” et “Déplacer archives” sont détaillés dans le diagramme de l’application mobile (chapitre 3) et dans le diagramme global (chapitre 2).

Description textuelle des cas d’utilisation

Le tableau suivant détaille le cas “Gérer les utilisateurs”. Les autres cas (“Gérer rôles”, “Gérer emplacements”, “Gérer zones”, “Gérer types archives”) suivent la même structure.

TABLE A.4 – Cas d’utilisation – Gérer les utilisateurs

Acteur	Administrateur
Objectif	Consulter, ajouter, modifier ou désactiver des comptes
Précondition	Administrateur authentifié (token JWT)
Postcondition	Modifications enregistrées dans SQL Server
Scénario nominal	1. Consulter la liste des utilisateurs (GET /utilisateurs). 2. Ajouter (POST), modifier (PATCH) ou désactiver (DELETE) un compte. 3. Backend applique les modifications. 4. Base de données mise à jour. 5. Message de confirmation retourné.
Scénarios alternatifs	1a. (étape 1) Token non admin → erreur 403 “Accès non autorisé”. 2a. (étape 2) CIN déjà existant → message “Utilisateur déjà enregistré”. 2b. (étape 2) Champ obligatoire manquant → erreur 400 “Données invalides”.

TABLE A.5 – Cas d’utilisation – Gérer les rôles

Acteur	Administrateur
Objectif	Attribuer ou modifier les permissions d’un rôle
Précondition	Administrateur authentifié (token JWT)
Postcondition	Permissions mises à jour dans SQL Server
Scénario nominal	1. Sélectionner un rôle existant. 2. Modifier ses permissions. 3. Backend applique les modifications. 4. Base de données mise à jour. 5. Message de confirmation retourné.
Scénarios alternatifs	1a. (étape 1) Rôle inexistant → erreur 404 “Rôle non trouvé”. 2a. (étape 2) Permission invalide → erreur 400 “Permission inconnue”. 1b. (étape 1) Token non admin → erreur 403 “Accès non autorisé”.

TABLE A.6 – Cas d'utilisation – Gérer les emplacements

Acteur	Administrateur, Archiviste
Objectif	Créer, modifier ou désactiver un emplacement
Précondition	Administrateur authentifié (token JWT)
Postcondition	Emplacement enregistré ou mis à jour dans SQL Server
Scénario nominal	1. Saisir le nom et la description d'un emplacement. 2. Valider la création (POST /emplacements). 3. Backend enregistre les données. 4. Message de confirmation retourné.
Scénarios alternatifs	1a. (étape 1) Nom d'emplacement déjà existant → erreur 409 "Emplacement déjà enregistré". 1b. (étape 1) Nom vide → erreur 400 "Nom obligatoire". 1c. (étape 1) Token non admin → erreur 403 "Accès non autorisé".

TABLE A.7 – Cas d'utilisation – Gérer les zones

Acteur	Administrateur, Archiviste
Objectif	Ajouter, modifier ou activer/désactiver une zone
Précondition	Administrateur authentifié (token JWT)
Postcondition	Zone mise à jour dans SQL Server
Scénario nominal	1. Sélectionner un emplacement parent. 2. Ajouter une nouvelle zone (POST /zones). 3. Backend vérifie que l'emplacement existe. 4. Zone enregistrée.
Scénarios alternatifs	1a. (étape 1) Emplacement parent inexistant → erreur 404 "Emplacement non trouvé". 2a. (étape 2) Nom de zone déjà existant dans le même emplacement → erreur 409 "Zone déjà enregistrée". 1b. (étape 1) Token non admin → erreur 403 "Accès non autorisé".

TABLE A.8 – Cas d'utilisation – Gérer les types d'archives

Acteur	Administrateur, Archiviste
Objectif	Ajouter ou modifier une catégorie d'archive
Précondition	Administrateur authentifié (token JWT)
Postcondition	Type d'archive enregistré dans SQL Server
Scénario nominal	1. Saisir le nom du type (ex : "Contrat"). 2. Valider la création (POST /types). 3. Backend enregistre les données. 4. Message de confirmation retourné.
Scénarios alternatifs	1a. (étape 1) Type d'archive déjà existant → erreur 409 "Type déjà enregistré". 1b. (étape 1) Nom vide → erreur 400 "Nom obligatoire". 1c. (étape 1) Token non admin → erreur 403 "Accès non autorisé".

TABLE A.9 – Cas d'utilisation – Consulter le tableau de bord

Acteur	Administrateur, Archiviste, Consultant
Objectif	Visualiser les indicateurs de performance
Précondition	Authentifié (token JWT)
Postcondition	Affichage des statistiques
Scénario nominal	1. Accéder à la page “Tableau de bord”. 2. Backend envoie les statistiques (GET /dashboard). 3. Interface affiche les données.
Scénarios alternatifs	1a. (étape 1) Token expiré ou invalide → erreur 401 → redirection vers login. 2a. (étape 2) Aucune donnée disponible → message “Aucune statistique pour le moment”.

Les diagrammes de séquence associés à ces cas d'utilisation sont présentés dans les sections suivantes.

Diagrammes de séquence du backoffice

Cette section présente les principaux diagrammes de séquence illustrant les actions d’administration dans le backoffice web.

1. Ajout d’un utilisateur La figure A.3 décrit le processus d’ajout d’un nouvel utilisateur par l’administrateur. L’administrateur remplit le formulaire (CIN, nom, rôle, etc.). Le backend vérifie que le rôle existe, puis enregistre le nouvel utilisateur dans SQL Server.

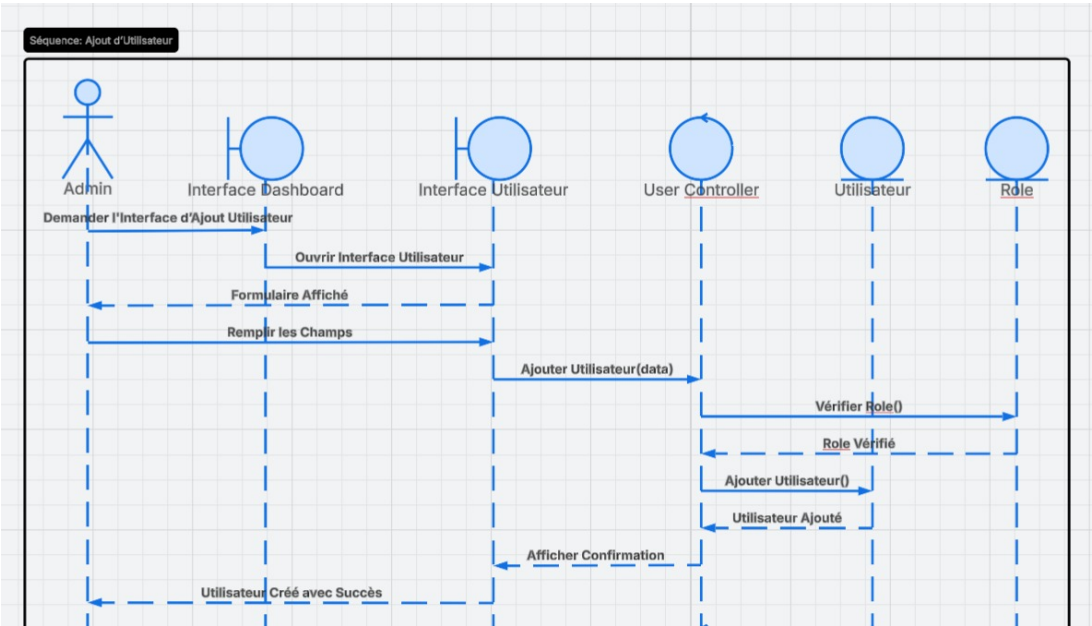


FIGURE A.3 – Diagramme de séquence – Ajout d’un utilisateur

2. Modification d’un utilisateur La figure A.4 illustre la modification des informations d’un utilisateur existant. L’administrateur sélectionne un utilisateur, modifie ses champs (nom, rôle, etc.), puis valide. Le backend met à jour la base de données.

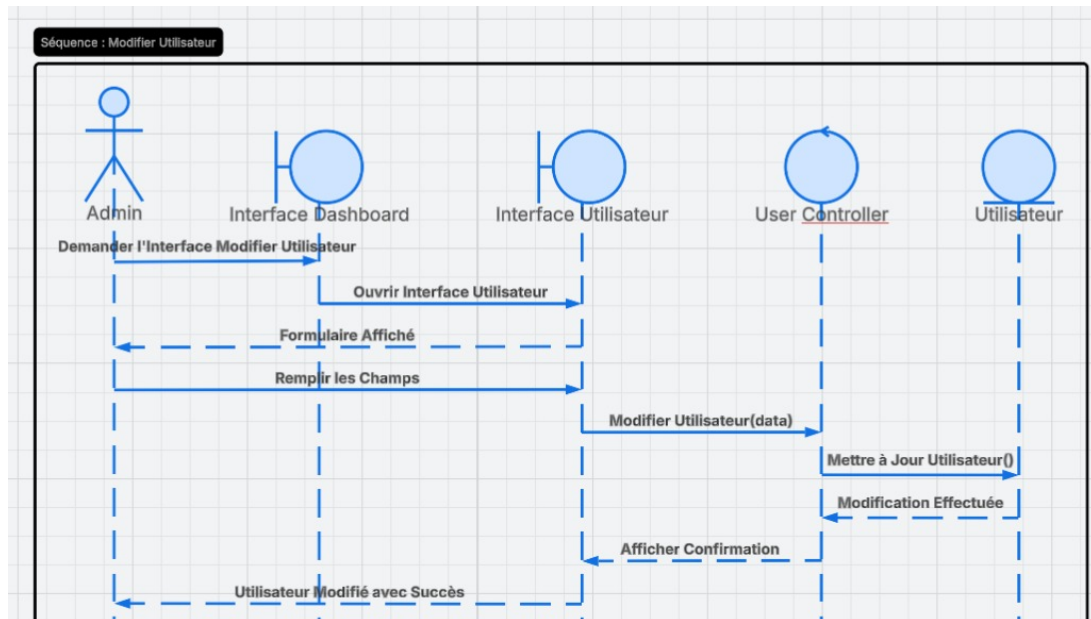


FIGURE A.4 – Diagramme de séquence – Modification d'un utilisateur

3. Ajout d'une zone La figure A.5 décrit la création d'une nouvelle zone de stockage. L'administrateur saisit le nom de la zone et l'emplacement parent. Le backend enregistre la zone et retourne une confirmation.

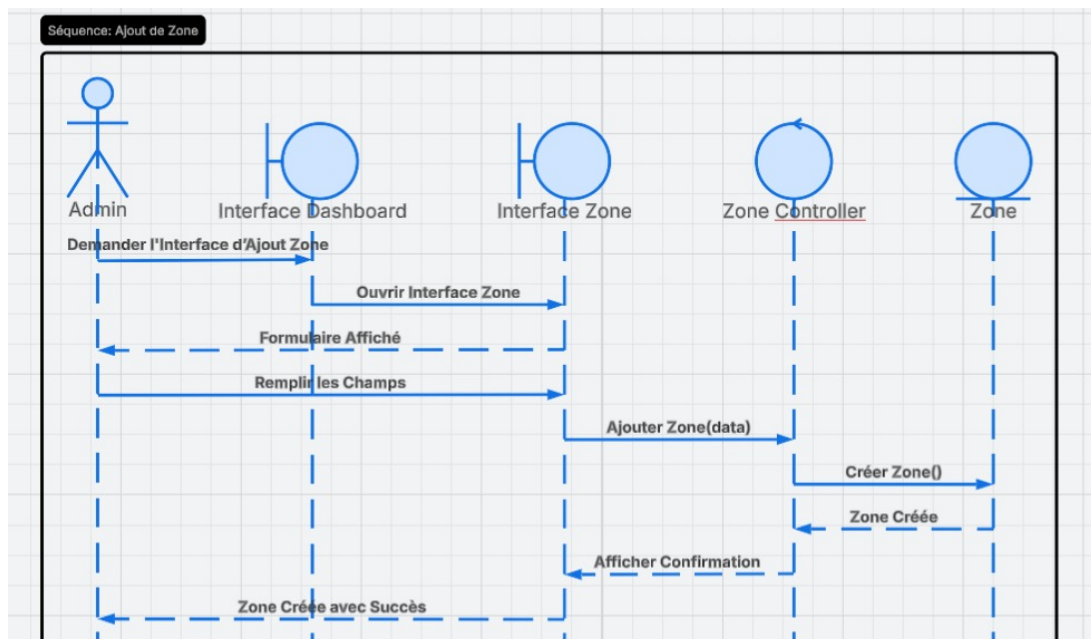


FIGURE A.5 – Diagramme de séquence – Ajout d'une zone

4. Ajout d'un emplacement La figure A.6 illustre la création d'un nouvel emplacement. L'administrateur remplit le formulaire (nom, description). Le backend crée l'emplacement et l'enregistre.

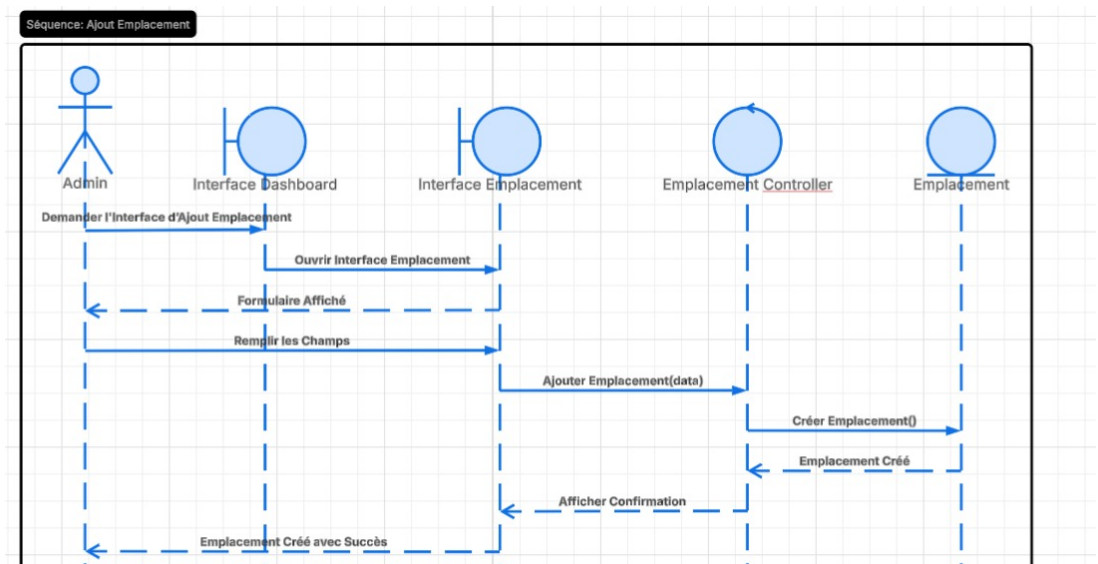


FIGURE A.6 – Diagramme de séquence – Ajout d’un emplacement

5. Ajout d’un type d’archive La figure A.7 décrit la création d’un nouveau type d’archive (ex : “Contrat”, “Facture”). L’administrateur saisit le nom du type. Le backend le sauvegarde et affiche une confirmation.

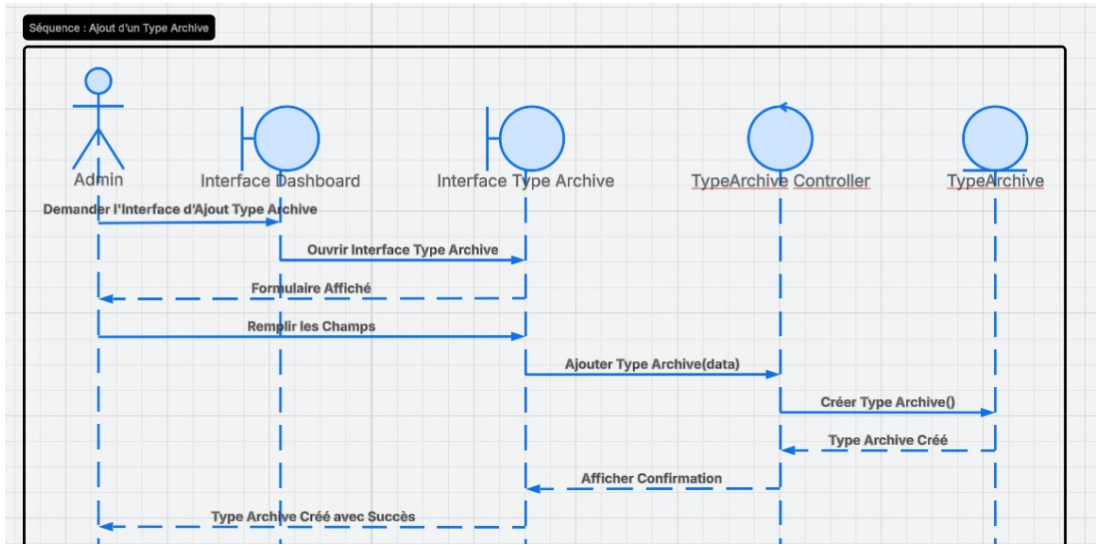


FIGURE A.7 – Diagramme de séquence – Ajout d’un type d’archive

6. Ajout d’un rôle La figure A.8 illustre la création d’un nouveau rôle avec ses permissions. L’administrateur définit le nom du rôle et les droits associés. Le backend l’enregistre.

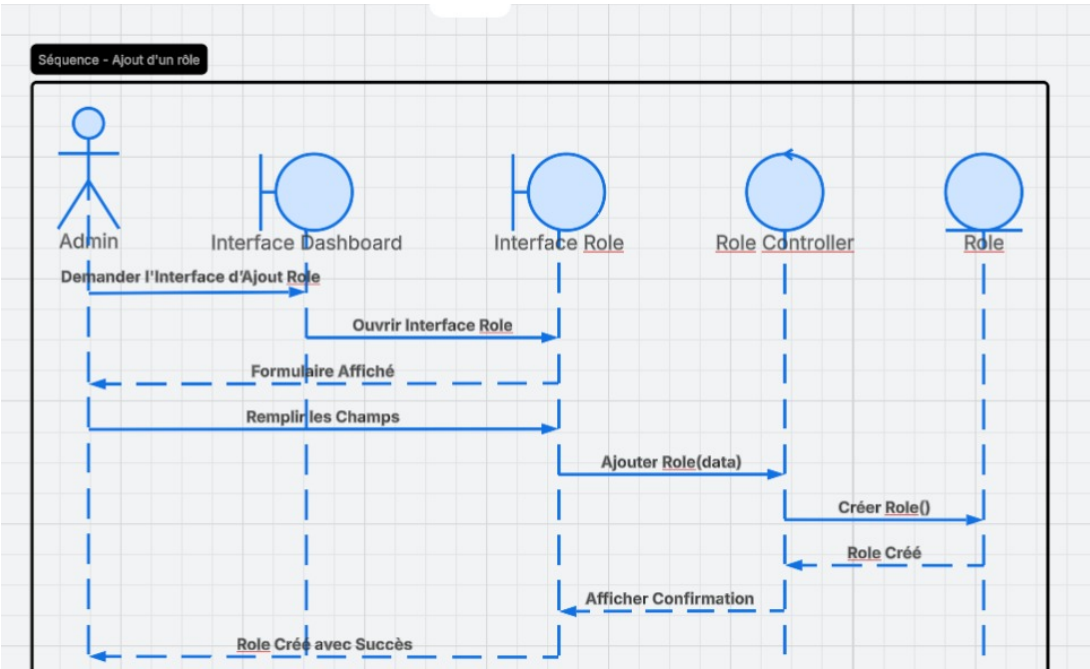


FIGURE A.8 – Diagramme de séquence – Ajout d’un rôle

Interfaces du backoffice

Les captures suivantes présentent l’ensemble des écrans réalisés (ces figures n’apparaissent pas dans les chapitres précédents).

Authentification et tableau de bord : L’administrateur commence par se connecter (figure A.9), puis accède au tableau de bord (figure A.10) pour visualiser les indicateurs clés.

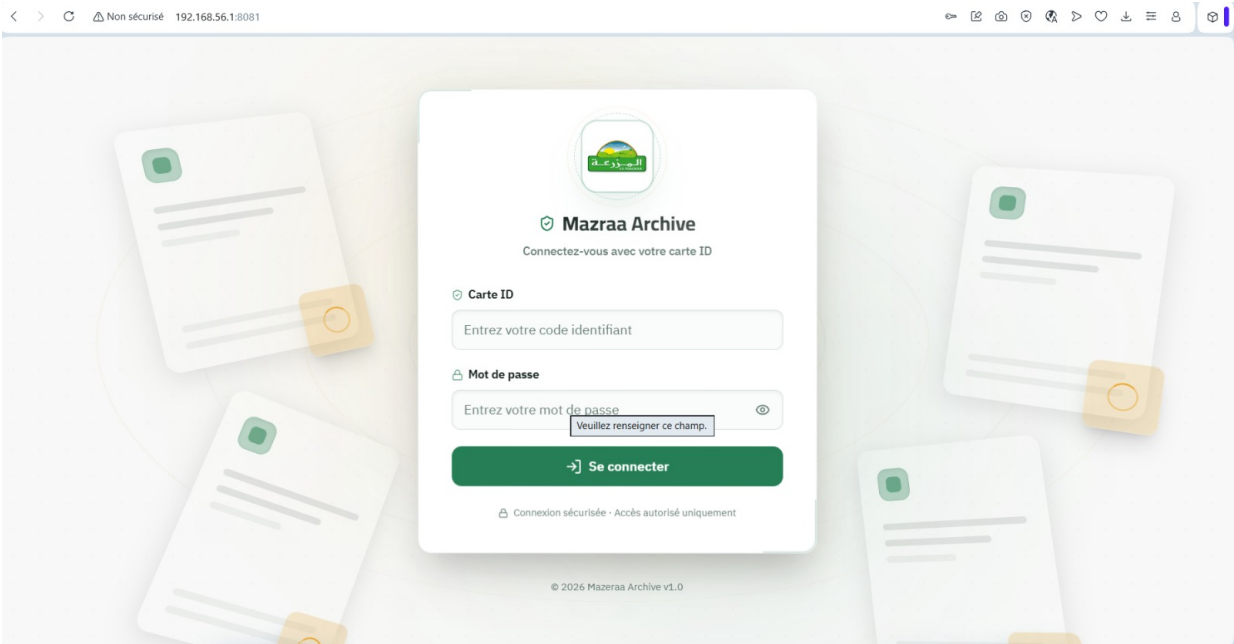


FIGURE A.9 – Écran de connexion au backoffice

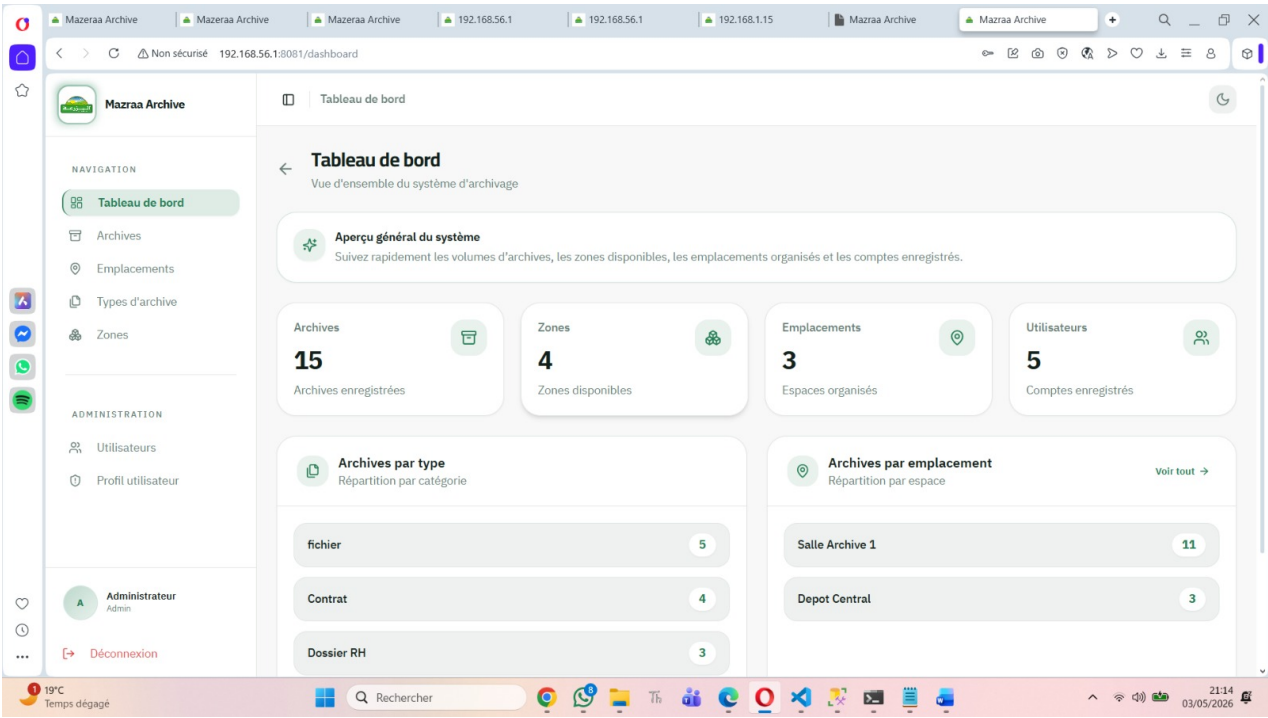


FIGURE A.10 – Tableau de bord (Dashboard)

Gestion des utilisateurs : L'administrateur peut gérer les comptes utilisateurs (figure A.11) et modifier son propre profil (figure A.12).

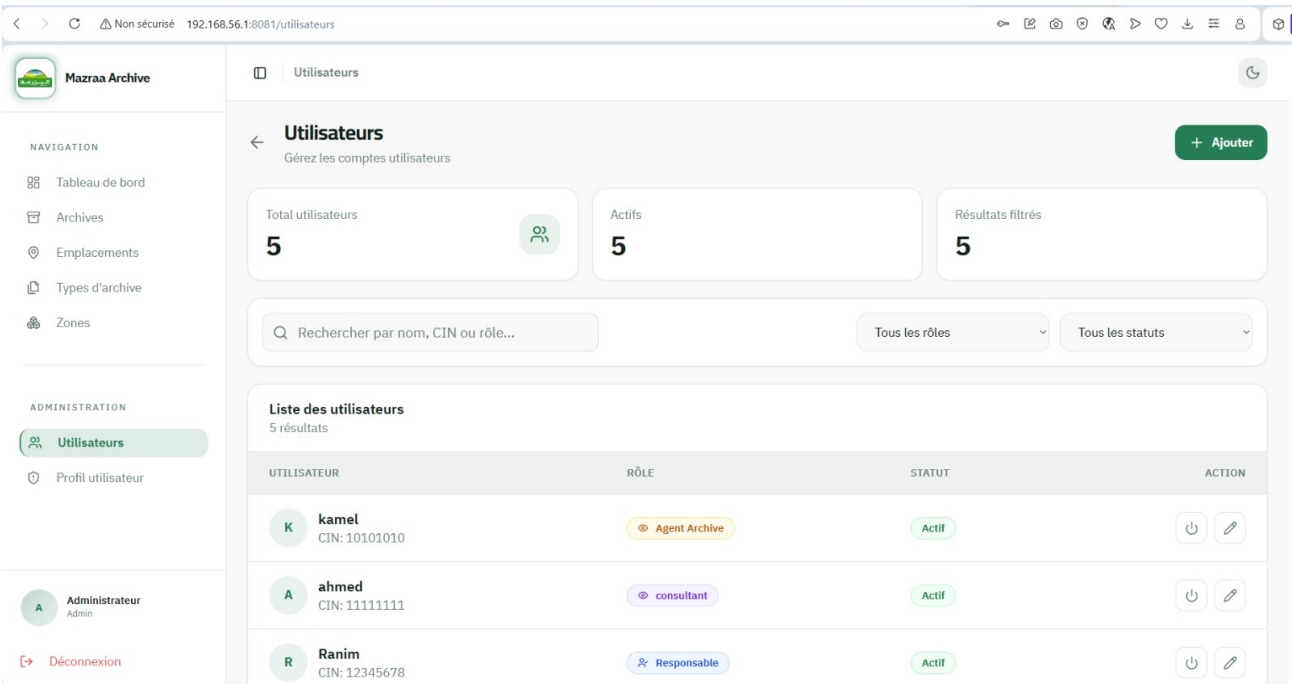


FIGURE A.11 – Gestion des utilisateurs

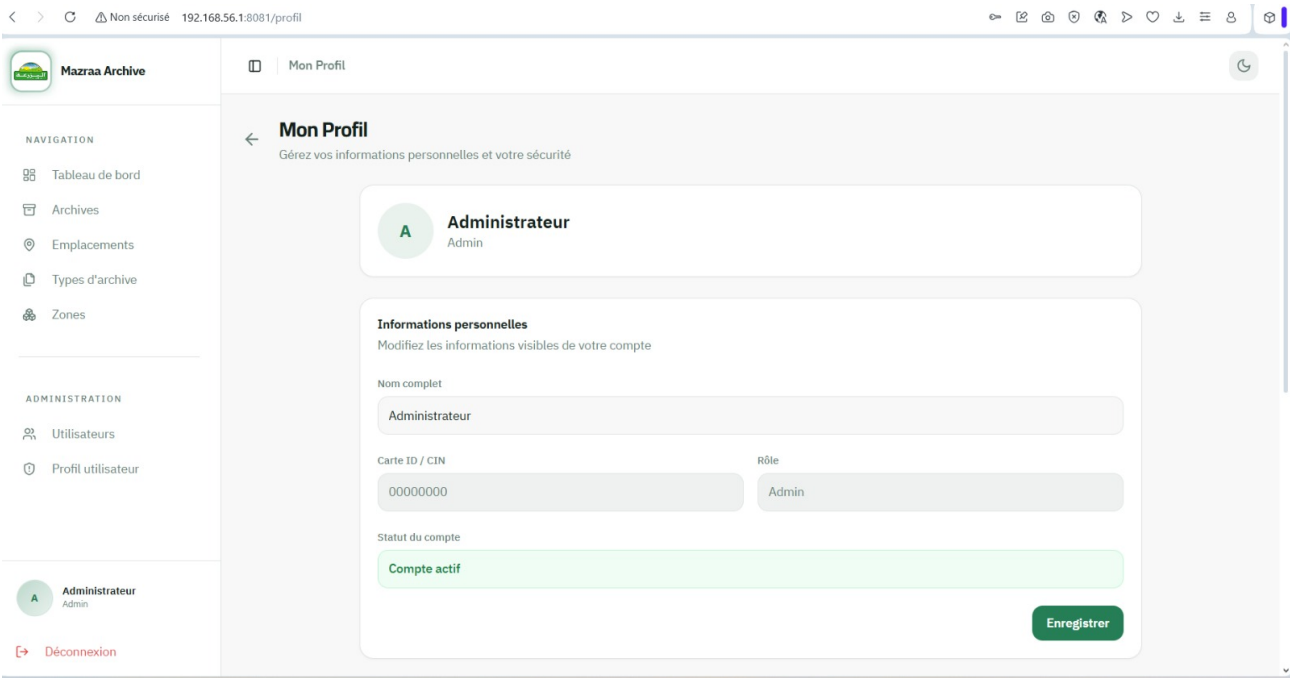


FIGURE A.12 – Gestion du profil utilisateur

Paramétrage du système : L'administrateur configure les types d'archives (figure A.13), les emplacements (figure A.14) et les zones de stockage (figure A.15).

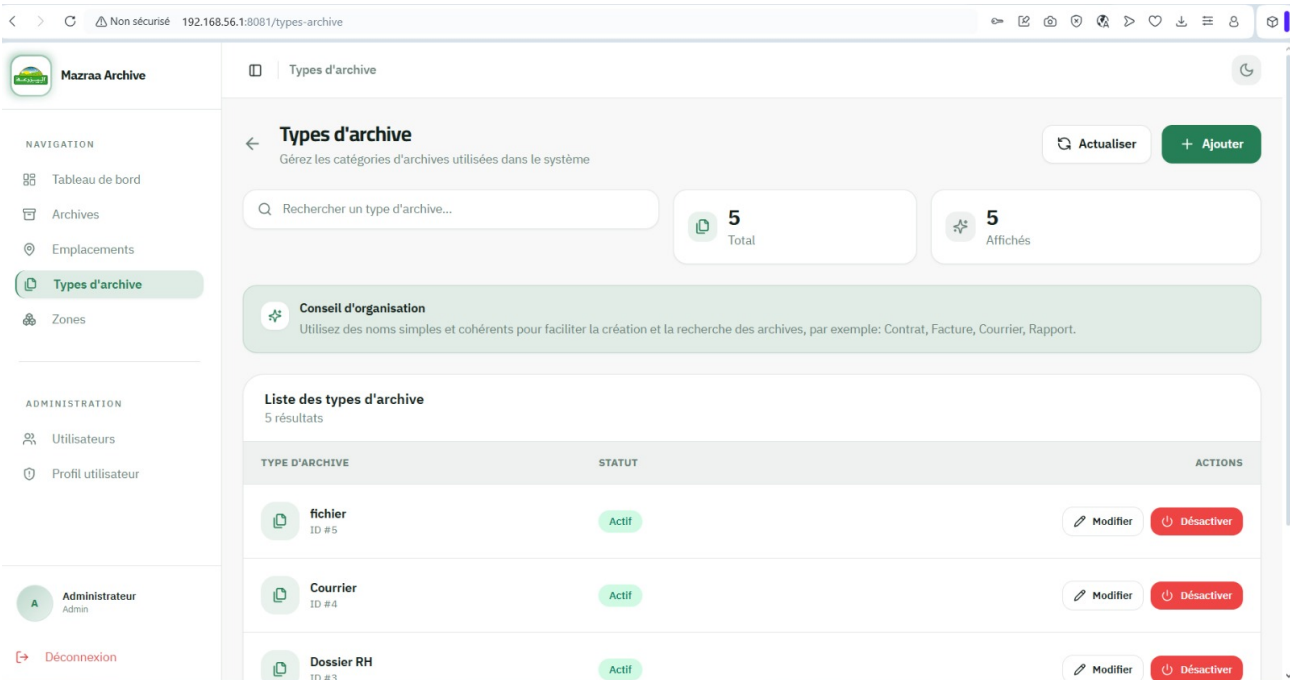


FIGURE A.13 – Gestion des types d'archive

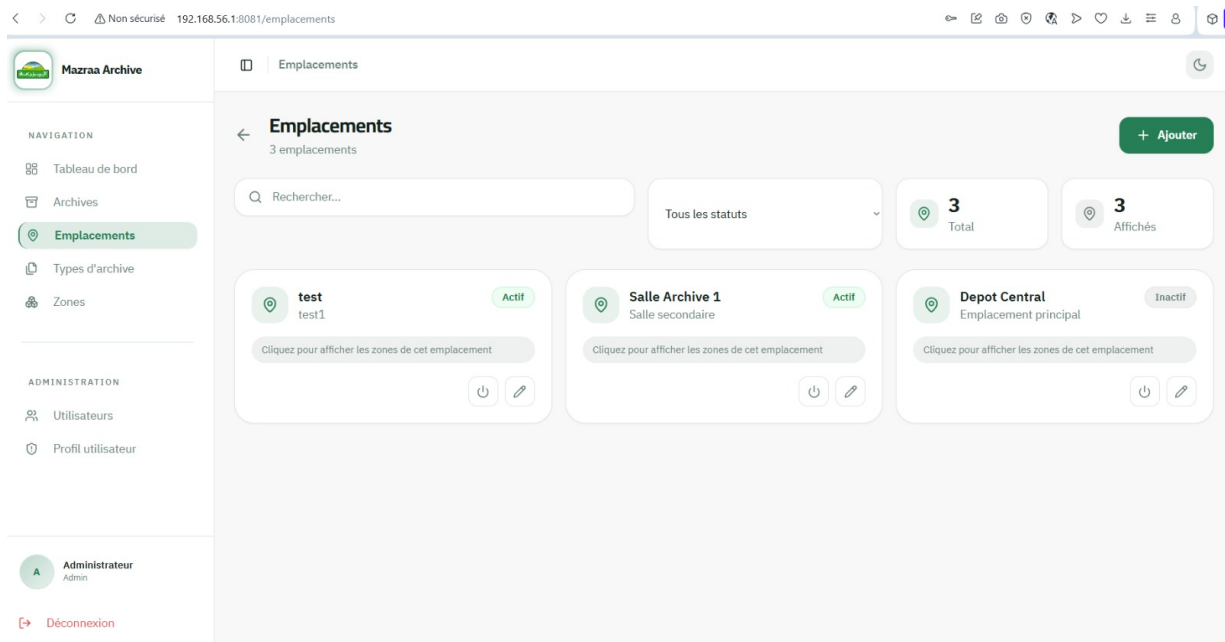


FIGURE A.14 – Gestion des emplacements

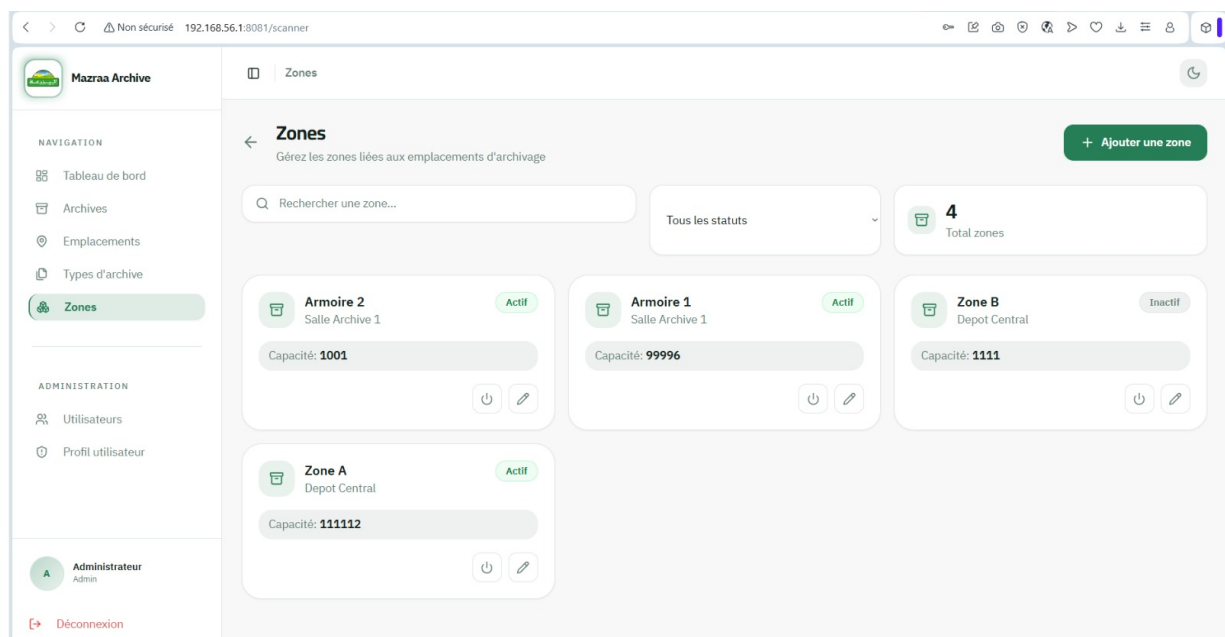


FIGURE A.15 – Gestion des zones de stockage

Résultat

À la fin du sprint, le backoffice web était pleinement opérationnel. Les administrateurs peuvent gérer l'ensemble des paramètres du système (utilisateurs, archives, zones, emplacements, types) via une interface moderne et responsive.

Conclusion de l'annexe

Cette annexe a rassemblé les diagrammes et les interfaces des trois premiers sprints sans reprendre les informations déjà présentes dans le chapitre 3. L'ensemble constitue une documentation complète pour le jury.

Bibliographie

- [1] Kaoru Ishikawa, *Guide to Quality Control*, Asian Productivity Organization, 1986.
- [2] Kotler, P., Keller, K. L. (2012). *Marketing Management*. Pearson Education.
- [3] Booch, G., Rumbaugh, J., Jacobson, I. (2005). *The Unified Modeling Language User Guide*. Addison-Wesley.
- [4] Sommerville, I. (2011). *Software Engineering*. Pearson Education.

Webographie

- [1] Scrum.org (2024). *Scrum Framework Poster*. Disponible sur : <https://www.scrum.org/resources/scrum-framework-poster> Consulté le 10 Février 2026.
- [2] Schwaber, K., Sutherland, J. (2020). *The Scrum Guide*. Disponible sur : <https://scrumguides.org> Consulté le 10 Février 2026.
- [3] Meta Open Source (2024). *React Documentation*. Disponible sur : <https://react.dev> Consulté le 20 Février 2026.
- [4] Google (2024). *Flutter Documentation*. Disponible sur : <https://flutter.dev> Consulté le 20 Février 2026.
- [5] Node.js Foundation (2024). *Node.js Documentation*. Disponible sur : <https://nodejs.org> Consulté le 25 Février 2026.
- [6] Express.js Contributors (2024). *Express Documentation*. Disponible sur : <https://expressjs.com> Consulté le 25 Février 2026.
- [7] Atlassian (2024). *Scrum Story Points and Fibonacci Scale*. Disponible sur : <https://www.atlassian.com/agile/project-management/estimation> Consulté le 13 Mai 2026.
- [8] Microsoft (2024). *SQL Server Documentation*. Disponible sur : <https://learn.microsoft.com/en-us/sql/> Consulté le 1 Mars 2026.
- [9] GitHub, Inc. (2024). *GitHub Documentation*. Disponible sur : <https://docs.github.com> Consulté le 5 Mars 2026.
- [10] Microsoft (2024). *Visual Studio Code Documentation*. Disponible sur : <https://code.visualstudio.com/docs> Consulté le 5 Mars 2026.
- [11] Lucid Software (2024). *Lucidchart Documentation*. Disponible sur : <https://www.lucidchart.com> Consulté le 15 Février 2026.
- [12] Taiga Agile, LLC (2024). *Taiga Documentation*. Disponible sur : <https://taiga.io> Consulté le 20 Février 2026.
- [13] ngrok Inc. (2025). *ngrok Documentation*. Disponible sur : <https://ngrok.com/docs> Consulté le 10 Avril 2026.
- [14] Postman, Inc. (2024). *Postman Documentation*. Disponible sur : <https://www.postman.com> Consulté le 10 Mars 2026.
- [15] Thunder Client (2024). *Thunder Client Documentation*. Disponible sur : <https://www.thunderclient.com/docs/intro> Consulté le 5 Juin 2026.
- [16] Tailwind Labs (2024). *Tailwind CSS Documentation*. Disponible sur : <https://tailwindcss.com> Consulté le 1 Mars 2026.
- [17] CNRTL (2024). *Architecture client-serveur - Définition*. Disponible sur : <https://www.cnrtl.fr/definition/client-serveur> Consulté le 25 Janvier 2026.
- [18] Vite Team (2024). *Vite Documentation*. Disponible sur : <https://vitejs.dev> Consulté le 1 Mars 2026.
- [19] Overleaf (2024). *Overleaf Documentation*. Disponible sur : <https://www.overleaf.com> Consulté le 1 Juin 2026.
- [20] MDN Web Docs (2025). *JavaScript Guide*. Disponible sur : <https://developer.mozilla.org/fr/docs/Web/JavaScript> Consulté le 1 Mars 2026.
- [21] Dart.dev (2025). *Dart Programming Language*. Disponible sur : <https://dart.dev> Consulté le 15 Mars 2026.

- [22] Microsoft (2025). *SQL Documentation*. Disponible sur : <https://learn.microsoft.com/en-us/sql/>
Consulté le 1 Mars 2026.
- [23] MDN Web Docs (2025). *HTML and CSS Documentation*. Disponible sur : <https://developer.mozilla.org/fr/docs/Web>
Consulté le 15 Février 2026.

Résumé

ملخص

تم إنجاز هذا المشروع داخل مؤسسة إل مزرعة بهدف تطوير نظام معلوماتي ذكي لإدارة الأرشيف بالاعتماد على تقنية الباركود. يتكوّن الحل من تطبيق هاتفي مطوّر باستخدام فلاتر يتيح لأمين الأرشيف مسح الباركود وإضافة الوثائق والبحث عنها ونقلها، ومن منصة ويب مطوّرة باستخدام ريباكت مخصّصة للإدارة المركزية وتسيير المستخدمين والصلاحيات وأنواع الأرشيف والمواقع والمناطق. يساهم هذا النظام في تحسين تنظيم الوثائق، وتسريع عمليات البحث والاسترجاع، وضمان تتبّع دقيق لحركة الأرشيف، مما يعزّز كفاءة إدارة الوثائق داخل المؤسسة.

الكلمات المفتاحية: إدارة الأرشيف، الرموز الشريطية (الباركود)، تطبيقات الهاتف الجوال، واجهة إدارة ويب، فلاتر، ريباكت، نود جي إس، إس كيو إل سيرفر، تتبع العمليات، منهجية سكرم

Résumé

Ce projet, réalisé au sein de l'entreprise EL MAZRAA, porte sur la conception et le développement d'un système informatisé de gestion des archives basé sur l'identification par codes-barres. La solution comprend une application mobile (Flutter) permettant aux archivistes de scanner, ajouter, rechercher et déplacer des archives, ainsi qu'un backoffice web (React) destiné à l'administration centralisée (gestion des utilisateurs, rôles, types, emplacements et zones). L'utilisation des codes-barres garantit l'unicité des documents, une recherche rapide et une traçabilité complète des mouvements.

Mots-clés : Gestion des archives, codes-barres, application mobile, backoffice web, Flutter, React, Node.js, SQL Server, traçabilité, Scrum.

Abstract

This project, carried out within the company EL MAZRAA, focuses on the design and development of a computerized archive management system based on barcode identification. The solution includes a mobile application (Flutter) allowing archivists to scan, add, search and move archives, as well as a web backoffice (React) for centralized administration (management of users, roles, types, locations and zones). The use of barcodes guarantees the uniqueness of documents, fast search and complete traceability of movements.

Keywords : Archive management, barcodes, mobile application, web backoffice, Flutter, React, Node.js, SQL Server, traceability, Scrum.