République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Saad Dahlab Blida 1

Département d'Informatique

Spécialité : Ingénierie **L**ogicielle

Mémoire de Fin d'Etudes

Pour l'obtention du diplôme de Master en Informatique

Conception et Réalisation d'une application web pour la gestion de l'archive administrative et électorale de la cour constitutionnelle

Présenté par : **El HEIT Mohamed Wassim**

Présenté devant la commission d'examination composée de :

Mme. Meskaldji K (Promotrice)

Mme. Neddar S (Encadrante)

Mr Kamech H (Président)

Mme Lahiani N (Examinateur)

Remerciements

Je tiens à témoigner ma reconnaissance à DIEU tout puissant, qui ma a aidé et bénit par sa volonté durant toute cette période.

Je tiens à remercier en premier lieu Mr Hamdad Amine pour sa rigueur, sa disponibilité et ses conseils.

Mes remerciements s'adressent également aux membres du jury qui m'ont fait le grand honneur d'évaluer ce travail.

Finalement, je tiens à remercier Mme Meskaldji Khouloud, Mr Ziane Ibrahim et tous ceux qui, d'une manière ou d'une autre aidé et encouragé à la réalisation de ce modeste travail

Dédicaces

Je dédie ce travail :

A mes très chers parents qui se sacrifient pour moi, que dieu les protège.

A Mes Sœur et Mon frère

A tous mes amis(es).

A tous ceux qui m'aiment et que j'aime.

Résumé

La cour constitutionnelle est l'institution chargée de veiller au respect de la constitution et des principes fondamentaux de la démocratie dans un pays. Elle est également responsable de la régulation du processus électoral et du contrôle de la validité des élections.

Afin de remplir ces missions, la cour constitutionnelle produit et reçoit de nombreux documents administratifs et électoraux qui constituent des sources d'information précieuses pour l'histoire politique et juridique du pays.

Ces documents doivent être conservés, classés, indexés et accessibles aux citoyens, aux chercheurs et aux autorités compétentes. Cependant, la gestion de ces archives pose des défis importants en termes de volume, de diversité, de sécurité et de pérennité.

Le but de ce projet est de concevoir et de réaliser une application web pour la gestion de l'archive administrative et électorale de la cour constitutionnelle. Cette application devra permettre aux utilisateurs autorisés de consulter, rechercher, ajouter, modifier et supprimer des documents archivés dans une base de données centralisée.

Mots clés: Archive, Gestion électronique des documents, Micro services, Sécurité.

Abstract

The constitutional court is the institution responsible for ensuring compliance with the constitution and the fundamental principles of democracy in a country. It is also responsible for regulating the electoral process and verifying the validity of elections.

In order to fulfill these tasks, the constitutional court produces and receives numerous administrative and electoral documents that constitute valuable sources of information for the political and legal history of the country.

These documents must be preserved, classified, indexed and accessible to citizens, researchers and competent authorities. However, the management of these archives poses significant challenges in terms of volume, diversity, security and durability.

The goal of this project is to design and implement a web application for the management of the administrative and electoral archives of the Constitutional Court. This application will allow authorized users to consult, search, add, modify and delete archived documents in a centralized database.

Keywords: Archive, Electronic document management, Micro services, Security.

ملخص

المحكمة الدستورية هي المؤسسة المسؤولة عن ضمان الالتزام بالدستور والمبادئ الأساسية للديمقراطية في بلد ما. كما أنها مسؤولة عن تنظيم العملية الانتخابية والتحقق من صحة الانتخابات.

ولتنفيذ هذه المهام، تنتج وتتلقى المحكمة الدستورية عددًا كبيرًا من الوثائق الإدارية والانتخابية التي تشكل مصادر قيمة .للمعلومات عن التاريخ السياسي والقانوني للبلد

يجب الحفاظ على هذه الوثائق وتصنيفها وفهرستها وجعلها متاحة للمواطنين والباحثين والسلطات المختصة. ومع ذلك، تواجه إدارة هذه الأرشيفات تحديات كبيرة من حيث الحجم والتنوع والأمن والاستدامة

الهدف من هذا المشروع هو تصميم وتنفيذ تطبيق ويب لإدارة الأرشيفات الإدارية والانتخابية للمحكمة الدستورية. سيسمح هذا التطبيق للمستخدمين المصرح لهم بالاطلاع على الوثائق المؤرشفة والبحث فيها وإضافتها وتعديلها وحذفها في قاعدة بيانات مركزية

الكلمات المفتاحية: الأرشيف ، إدارة المستندات الإلكترونية ، الخدمات المصغرة ، الأمن

Table des matières

Listes des figures

Liste des Tables

Liste des Abréviations

GED : Gestion électronique des Documents.

APN : Assemblée Populaire Nationale.

ANIE : Autorité Nationale Indépendante des élections.

API: Application Programing Interface

REST: Representation State Transfer.

RPC: Remote Procedure Call.

SOA: Architecture Orientée Services.

OAuth2: Open Authorization version 2.0.

JWT: JSON Web Token.

LDAP: Lightweight Directory Access Protocol.

CSRF: Cross Site Request Forgery.

MVVM: Modèle Vue Modèle.

MVC: Modèle Vue Contrôleur

RAID: Redundant Array of Independent Disks.

HTTP: Hypertext Transfer Protocol.

HQL: Hibernate Query Language.

SQL: Structured Query Language.

Introduction générale

Ces dernières années, le développement du web a connu des changements importants. De nouvelles technologies sont apparues, offrant plusieurs choix pour la création d'applications Web.

L'augmentation constante du volume documentaire dans la cour constitutionnelle basée sur le recours massif au papier, rend la vie de l'archiviste si difficile à cause d'une gestion traditionnelle des documents.

Dans ce contexte, la gestion électronique des documents (GED) est la solution idéale. Elle permet la numérisation de documents papiers et la gestion de toutes les étapes du cycle de vie d'un document numérique.

Les applications sont généralement créées de manière monolithique. Autrement dit, tous les éléments de l'application sont dans cette seule application. Donc, il est difficile d'ajouter des fonctionnalités et de résoudre rapidement les problèmes qui surviennent.

Avec une approche basée sur des **Micros-Services**, il est possible de résoudre ces problèmes, d'améliorer le développement et de gagner en réactivité.

L'objectif de notre travail est de développer une application web pour la gestion d'archivage administrative et électoral de la cour constitutionnelle basée sur les Micros-Services afin de résoudre de nombreux problèmes rencontrés par l'archiviste lorsqu'il gère les documents. Ce qui lui permet de gagner du temps et d'obtenir les meilleurs résultats.

Ce travail est structuré de la façon suivante :

- Le premier chapitre Étude de l'existant **et** Analyse des Besoins, est dédié à la présentation de La cour constitutionnelle ainsi que le cahier des charges de l'application.
- Le deuxième chapitre est consacré **à** la **conception** de l'application.
- Le dernier chapitre concerne la **Réalisation** de l'application en présentant les outils utilisés ainsi que quelques interfaces utilisateurs de l'application.

Chapitre 1 : Étude de l'existant et Analyse des Besoins

1. La Cour Constitutionnelle :

La **Cour constitutionnelle** créée par le constituant dans la Constitution du 1er Novembre 2020, à la place du **Conseil constitutionnel, [1]** est l'institution chargée de veiller au respect de la constitution et des principes fondamentaux de la démocratie dans le pays. Elle est également chargée de réglementer le processus électoral et de vérifier la validité des élections.

La Constitution a attribué à la Cour constitutionnelle un chapitre indépendant dans le titre IV, intitulé « Institutions de contrôle ». La Cour Constitutionnelle se voit doter de larges prérogatives lui conférant le rôle d'épine dorsale de l'État de droit, à la faveur des nouvelles dispositions contenues dans la Constitution du 1er Novembre 2020. **[1]**

Occupant ainsi une place de choix dans la Constitution de 2020, la Cour constitutionnelle a pour mission de contrôler la constitutionnalité des lois et la protection des droits et libertés.

Dans le même sens, l'article 190 de la Loi fondamentale du pays énonce que la Cour constitutionnelle se prononce par une décision sur la constitutionnalité des traités, des lois et des règlements, et peut être saisie sur la constitutionnalité des traités avant leur ratification et sur les lois avant leur promulgation, comme elle peut être saisie sur la constitutionnalité des règlements. Elle se prononce aussi sur « la conventionalité des lois et des règlements dans les mêmes conditions. **[1]**

2. Organisation de la cour constitutionnelle

Vu le décret présidentiel n° 21-453 du 11 Rabie Ethani 1443 correspondant au 16 novembre 2021 portant désignation du président de la Cour constitutionnelle.

Et le décret présidentiel n° 22-93 du 5 Chaâbane 1443 correspondant au 8 mars 2022 relatif aux règles se rapportant à l'organisation de la Cour constitutionnelle, notamment ses articles 9, 11 et 21, les structures et les organes de la Cour constitutionnelle comprennent :

La figure ci-dessus représente l'Organigramme de la Cour Constitutionnelle :

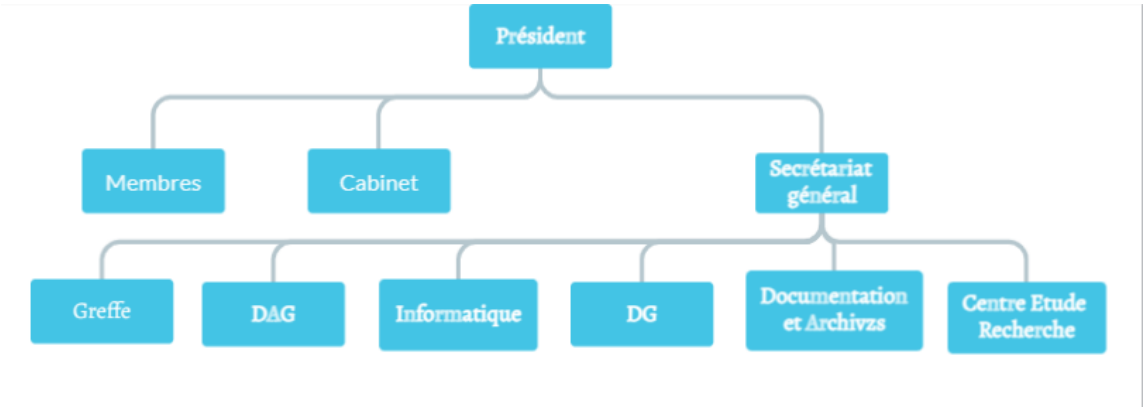


***Figure 1**: Organigramme de la cour constitutionnel*

La figure ci-dessus représente l'Organigramme du Conseil Constitutionnelle :

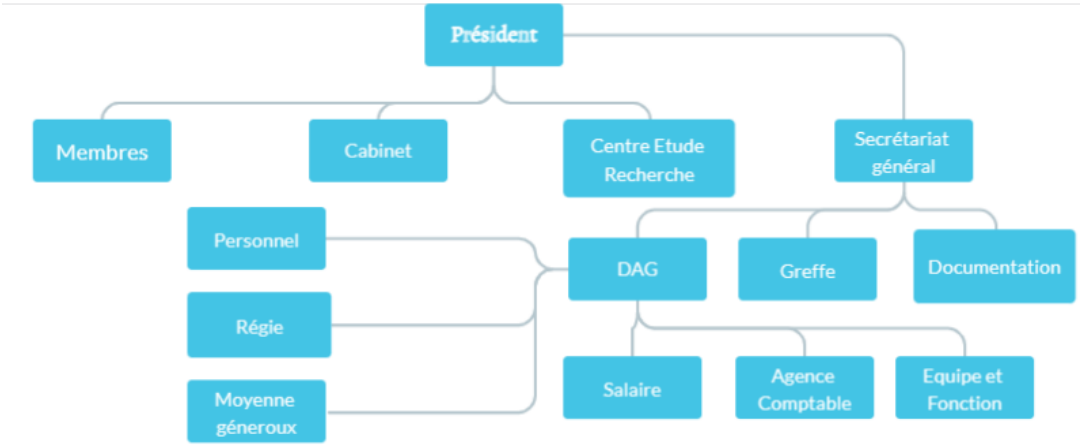


***Figure 2**: Organigramme du Conseil constitutionnelle*

3. élections

Les élections sont un élément essentiel de tout système démocratique. Elles permettent aux citoyens de participer activement à la vie politique de leur pays en choisissant leurs représentants et en exprimant leur volonté sur des questions importantes. Les élections offrent un moyen pacifique et légitime de sélectionner les dirigeants et de prendre des décisions collectives.

Dans de nombreux pays, y compris l'Algérie, les élections sont organisées régulièrement selon des procédures établies pour assurer leur transparence, leur impartialité et leur légitimité.

Ces élections offrent aux citoyens algériens l'opportunité de participer activement à la prise de décisions politiques et de contribuer au développement de leur pays. Elles reflètent également l'importance accordée à la démocratie et aux principes de gouvernance participative en Algérie.

3.1 Types élections en Algérie

3.2.1 Élections législatives

Les élections législatives en Algérie sont un processus essentiel pour choisir les membres de l'Assemblée populaire nationale (APN), qui est la chambre basse du parlement algérien. Ces élections permettent aux citoyens de participer activement à la sélection de leurs représentants politiques et de contribuer à la prise de décisions importantes pour le pays.

Les élections législatives en Algérie se déroulent généralement tous les cinq ans, conformément à la Constitution du pays. Les électeurs algériens ont la possibilité de voter pour des listes de partis politiques ou de coalitions de partis. Les sièges à l'Assemblée populaire nationale sont répartis en fonction des résultats électoraux, avec un système de représentation proportionnelle. **[2]**

3.2.2 Élection présidentielle

Les élections présidentielles en Algérie sont un processus important permettant aux citoyens de choisir leur président de la République. Ces élections se déroulent au suffrage universel direct tous les cinq ans, conformément à la Constitution algérienne.

Pour participer aux élections présidentielles en Algérie, les candidats doivent répondre à certaines conditions et se soumettre à un processus d'enregistrement et de validation par l'ANIE (Autorité nationale indépendante des élections). Les candidats doivent obtenir un certain nombre de parrainages d'électeurs pour pouvoir concourir à la présidence**.**

3.2.3 Élections sénatoriales

Les élections sénatoriales en Algérie ont pour objectif de choisir les membres du Conseil de la Nation, qui est la chambre haute du parlement algérien. Les sénateurs sont élus par un collège électoral composé de députés des assemblées locales (conseils communaux, conseils de wilaya, etc.) et de délégués des organisations professionnelles et socio-économiques.

Le Conseil de la Nation est renouvelé par tiers tous les trois ans. Les élections sénatoriales visent à garantir une représentation équitable des différentes régions et secteurs de la société au sein du parlement algérien.

3.2.4 Référendum

Les référendums en Algérie sont des consultations populaires directes sur des questions spécifiques, telles que des réformes constitutionnelles ou d'autres sujets importants. Les citoyens algériens ont l'opportunité de voter « oui » ou « non » pour exprimer leur opinion sur la proposition soumise au référendum.

3.3 Rôle de la cour constitutionnelle dans les élections

- Le Contrôle des opérations de révision des listes électorales par l'administration.
- L'Elaboration de recommandations pour l'amélioration du dispositif législatif et de régulation des opérations électorales.
- L'organisation de cycles de formation civique au bénéfice des formations politiques, sur la surveillance des scrutins et la formulation des recours.

4. Architecture Logicielle :

L'architecture logicielle décrit de manière symbolique et schématique les différents éléments d'un ou de plusieurs systèmes informatiques, leurs interrelations et leurs interactions **[9].**

L'architecture logicielle est produite lors de la phase de conception et constitue le plus gros livrable d'un processus logiciel après le produit (le logiciel lui-même). Les deux objectifs principaux de toute architecture logicielle sont la réduction des coûts et l'augmentation de la qualité du logiciel.

4.1 Architecture Monolithique :

Dans une architecture monolithique, toutes les fonctionnalités sont encapsulées dans une seule application, de sorte que ses modules ne peuvent pas être exécutés indépendamment.

Ce type d'architecture est étroitement couplé, et toute la logique de traitement d'une demande s'exécute dans un seul processus. Cela vous permet d'utiliser les fonctionnalités de base de votre langage pour diviser l'application en classes, fonctions et espaces de noms. Avec un peu d'attention, vous pouvez exécuter et tester l'application sur l'ordinateur portable d'un développeur et utiliser un pipeline de déploiement pour vous assurer que les changements sont correctement testés et déployés en production**. [11]**

Vous pouvez faire évoluer horizontalement le monolithe en exécutant de nombreuses instances derrière un équilibreur de charge.

C'est une bonne idée de commencer un projet en utilisant ce type d'architecture, car cela permet d'explorer à la fois la complexité d'un système et les limites de ses composants. Une fois que l'application devient importante et que l'équipe s'agrandit, cette architecture présente des **inconvénients** qui deviennent de plus en plus significatifs **[15] :**

- L'application peut être difficile à comprendre et à modifier. L'application peut être difficile à comprendre et à modifier, ce qui ralentit généralement le développement.
- Le déploiement continu est difficile ; une modification apportée à une petite partie de l'application nécessite de reconstruire et de déployer l'ensemble du monolithe.
- La mise à l'échelle de l'application peut être difficile (une architecture monolithique ne peut être mise à l'échelle qu'horizontalement).
- Nécessite un engagement à long terme vis-à-vis d'une pile technologique.

4.2 Architecture Micros-service

Inspirés de service-oriented computing, les micro-services sont de petites applications à responsabilité unique qui peuvent être déployées, mises à l'échelle et testées de manière indépendante.

Cette décomposition du monolithe (dont les modules ne peuvent être exécutés indépendamment) en un système granulaire interagissant par messages (par le biais d'API (Application Programing Interface) basées sur RPC (Remote Procedure Call) ou de services web RESTful (Représentation State Transfer) par exemple) permet aux organisations d'atteindre un meilleur délai de commercialisation grâce à des livraisons plus rapides et plus continues.

Les micros services sont développés de manière autonome. Cela permet également aux équipes agiles de structurer leur travail autour de ces services, étant donné que les micro-services sont, par définition, développés de manière autonome. **[18]**

Cependant, les micros services présentent également des défis et des inconvénients.

Les défis incluent :

- Décomposition du monolithe en micro services.
- La surveillance et le déploiement continus de l'architecture.
- Des tests plus complexes, des versions et des dépréciations, et la gestion des états.

Un **inconvénient** particulier est que les micros services peuvent impliquer des facteurs immatériels, tels que le besoin de personnel expérimenté et la difficulté d'apprendre la technologie**. [18]**

Des entreprises telles qu'Amazon, Deutsche Telekom, LinkedIn, Netflix, Sound Cloud, The Guardian, Uber et Verizon adoptent rapidement des approches basées sur les micros services. Souvent, les micros services sont utilisés pour moderniser les applications existantes.

L'objectif est de diviser ces systèmes monolithiques en micro services par le biais d'un remaniement. Cela permet de moderniser progressivement les logiciels existants et est peut-être moins risqué que de redévelopper complètement l'ensemble du système en micro services.

4.2.1 Technologies Pour Micros-Services

Les logiciels Micro-service décomposent les systèmes et les applications à un niveau plus granulaire et modulaire. Le concept a été créé dans le prolongement de l'architecture orientée services (SOA) il y a une dizaine d'années.

Il s’agit de fragmenter les applications complexes en petits éléments et de mettre en place un modèle de livraison fluide qui fournit des services à la demande, améliorant ainsi les performances. Diverses.

Les Technologies de micro services ont évolué au cours des deux dernières années. Le **tableau 1** donne un aperçu pour des technologies actuelles :

Tableau 1:Technologies de micro services de qualité industrielle.**[19]**

	Technology			
	Azure Service Fabric	Lagom	MicroProfile	Spring Suite (Boot)
Authentication	Active Directory	Basic	JavaScript Object Notation (JSON) Web Token	Spring Security
Security	Security Center	Basic	JSON Web Token	Spring Security
Tracing	Event Tracing Windows	Basic	OpenTracing	Spring Cloud Sleuth
Deployment	Built-in	—	Java Platform, Enterprise Edition (J2EE)	Yes, especially for the Spring framework
Reliability	Reliable Collections	Via others	MicroProfile Fault Tolerance 1.0	Built-in
Cost	Paid	Free	Free	Free
Orchestration	Azure Container Service	ConductR	Via others	Spring Suite
Monitoring (health check)	Application, cluster, or service based	Not available	MicroProfile Health Check 1.0	Hystrix
Usability	High	Low	Low	Medium
Containers	Azure Container Service	Via others	Via others	Via others
Language	C#, .NET, Java	Java, Scala	Java	Java
URL	azure.microsoft.com /en-us/services /service-fabric	www.lagomframework .com	microprofile.io	projects.spring.io /spring-boot

5. Sécurité des applications web

La sécurité des applications web désigne la pratique consistant à protéger les sites web, les applications et les API contre les attaques. Il s'agit d'une vaste discipline, mais ses objectifs ultimes sont de maintenir le bon fonctionnement des applications web et de protéger les entreprises du cyber vandalisme, du vol de données, de la concurrence déloyale et d'autres conséquences négatives. **[22]**

5.1 Principales types d’attaques

Les applications web peuvent être confrontées à plusieurs types d'attaques en fonction des objectifs de l'attaquant, de la nature du travail de l'organisation ciblée et des lacunes de sécurité particulières de l'application. **[22]**

Il existe plusieurs types d’attaques, les plus courants sont les suivants:

- **Vulnérabilités de type "zero day"** : Il s'agit de vulnérabilités inconnues des fabricants d'une application, et qui n'ont donc pas de correctif disponible. Les attaques cherchent à exploiter rapidement ces vulnérabilités.
- **Cross-Site Scripting (XSS) :** Cette faille de sécurité permet à un pirate d'injecter des scripts au sein d'une page web, côté client, dans le but d'accéder directement à des informations importantes, d'usurper l'identité des utilisateurs ou de piéger ces derniers afin de les amener à révéler des informations importantes.
- **Cross-Site Request Forgery (CSRF) :** Un navigateur web renvoie automatiquement tous les cookies qu'il détient pour un site web chaque fois qu'il y effectue une requête. Cela inclut tous les cookies d'identification de session active ou de jeton d'authentification. Pour cela, il est très facile pour un pirate de tromper votre navigateur et de lui faire envoyer des requêtes à n'importe quel site sur Internet. **[33]**
- **Injection SQL (SQi) :** Cette technique permet aux acteurs malveillants d'exploiter les vulnérabilités inhérentes à la manière dont une base de données exécute les requêtes de recherche. L'injection SQL permet ainsi d'accéder à des informations sans autorisation, de modifier ou de créer de nouvelles autorisations utilisateur, voire de manipuler ou détruire des données sensibles.
- **Attaques par déni de service (DoS) et déni de service distribué (DDoS) :** Les pirates peuvent surcharger un serveur ciblé ou son infrastructure environnante sous différents types de trafic hostile. Un serveur incapable de traiter efficacement les requêtes entrantes réagira de plus en plus lentement et finira par refuser de répondre aux requêtes entrantes des utilisateurs légitimes.

5.2 Principales stratégies de sécurité des applications web

La sécurité des applications web est une discipline vaste et en constante évolution. Les meilleures pratiques de la discipline évoluent au fur et à mesure que de nouvelles attaques et vulnérabilités apparaissent. **[22]**

Il existe plusieurs stratégies de sécurité, les plus courantes sont les suivantes:

- **Atténuation des attaques DDoS**: Les services d'atténuation des attaques DDoS s'interposent entre un serveur et l'internet public, en utilisant un filtrage spécialisé et une capacité de bande passante extrêmement élevée pour empêcher les pics de trafic malveillant de submerger le serveur**. [22]**
- **Pare-feu d'application Web (WAF):** Filtre le trafic connu pour ou suspecté de tirer parti des vulnérabilités des applications Web. Les WAF sont importants, Ils aident les organisations pour détecter les nouvelles vulnérabilités trop rapidement et trop discrètement.
- **Passerelles API:** Elles aident à gérer et à surveiller le trafic des API., Elles permettent d'identifier les "API fantômes" et de bloquer ou suspecté de cibler les vulnérabilités des API.
- **DNSSEC:** Protocole qui garantit que le trafic DNS d'une application web est acheminé en toute sécurité vers les bons serveurs, afin que les utilisateurs ne soient pas interceptés par un attaquant sur le chemin.
- **Gestion des certificats de chiffrement :** Dans lequel un tiers gère les éléments clés du processus de chiffrement SSL/TLS, tels que la génération de clés privées, le renouvellement des certificats et la révocation des certificats en raison de vulnérabilités. Cela élimine le risque que ces éléments passent inaperçus et exposent le trafic privé**. [22]**
- **Gestion des robots :** Qui utilise l'apprentissage automatique et des méthodes de détection pour distinguer le trafic automatisé des utilisateurs humains, et l’empêcher d'accéder à une application web.

- **Sécurité côté client :** permet aux organisations de détecter plus rapidement les activités malveillantes en vérifiant les nouvelles dépendances JavaScript et les modifications de code.

6. Conclusion :

La Cour Constitutionnelle dispose d'une grande base de données qui s'accroît de plus en plus chaque année. De plus, elle est capable de changer à tout moment (ex : la transformation du Conseil Constitutionnel en Cour Constitutionnelle)

Nous mettons en œuvre une application web utilisant les micro-services. Cette application doit permettre aux utilisateurs autorisés de visualiser, rechercher, ajouter, modifier et supprimer des documents stockés dans une base de données centralisée. Le chapitre suivant est consacré à la présentation de l'analyse et la conception.

Chapitre 2 : Analyse et conception

1. Introduction

Pour une meilleure organisation. Nous devons utiliser le processus de conception **orienté objet**, qui est une approche de développement souvent utilisée en conjonction avec le langage **UML**, qui nous permet de comprendre et de décrire les exigences de définition et de documentation du système.

2. Processus de développement

Un modèle est une représentation simplifiée, conventionnelle et pertinente du réel. Il permet de comprendre, concevoir, communiquer et documenter la solution informatique apportée à un système. Pour ce faire nous avons décidé d'utiliser UML (Unified Modeling Language) qui est un ensemble de modèles permettant de représenter un système informatique et son utilisation prévue dans l'entreprise. Il est nécessaire de préciser le processus d'élaboration des modèles, pour cela nous avons choisi le Modèle en cascade qui se décompose en un certain nombre de phase effectuées les unes après les autres. Chaque phase doit être approuvée avant de pouvoir commencer l'autre **[23].**

La figure ci-dessus représente le Modèle en Cascade :

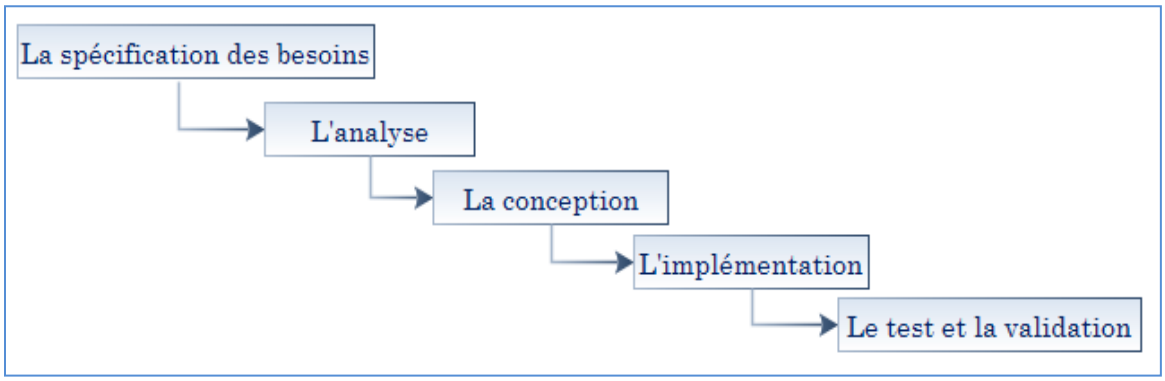


***Figure 3**: Modèle en cascade*

3. Acteurs :

Un acteur est un utilisateur type qui se comporte selon ses privilèges. Les acteurs de notre champ d'étude sont :

- **ADMINISTRATEUR**
- **ARCHIVISTE**

4. Diagramme de contexte :

Un diagramme de contexte est un diagramme conceptuel qui fournit une vue générale de l'interaction entre le système et la communication avec l'environnement externe, sachant que notre objectif est de concevoir une application Web pour les administrateurs et les archivistes.

Le diagramme ci-dessus représente le diagramme de contexte global de notre système :

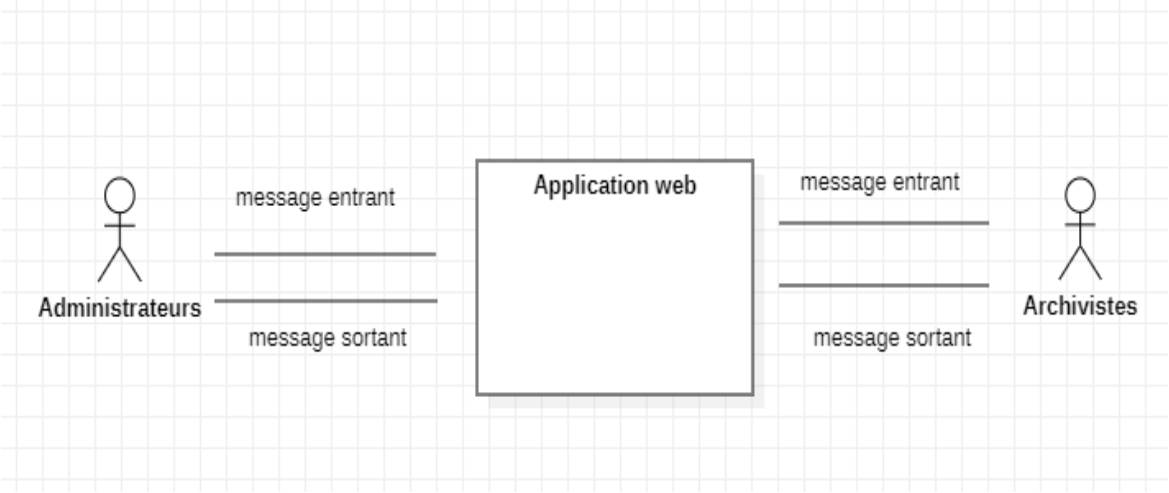


***Figure 4**: Diagramme de contexte.*

5. Les fonctionnalités :

Le tableau ci-dessus représente les Fonctionnalités des Acteurs de notre système :

***Tableau 2**:Fonctionnalités des Acteurs*

Acteur	Fonctionnalités
Archiviste	**Gestion des consultations :** à tout moment, un Archiviste doit avoir le droit pour consulter l'archive. **Gestion de l'archivage :** à tout moment, l'Archiviste peut ajouter un document, dossier ou une boite à la base des données **Gestion des mises à jour :** à tout moment, l'Archiviste peut modifier ou supprimer un document, dossier ou une boite à la base des données. **Gestion des Emprunts :** tout personne désirant emprunter un document, dossier ou une boite (version originale) doit avoir le droit et peut le/la garder pour un délai donné.
Admin	**Gestion des consultations :** à tout moment, un Admin doit avoir le droit pour consulter l'archive. **Gestion de l'archivage :** à tout moment, l'Admin peut ajouter un document, dossier ou une boite à la base des données **Gestion des mises à jour :** à tout moment, l'Admin peut modifier ou supprimer un document, dossier ou une boite à la base des données. **Gestion des Emprunts :** tout personne désirant emprunter un document, dossier ou une boite (version originale) doit avoir le droit et peut le/la garder pour un délai donné.

	Gestion des utilisateurs : un Admin peut ajouter des utilisateurs, et leur attribuer des rôles. **Gérer les modifications architecturales :** un Admin ajouter des nouveaux Structures (Divisions),

6. Les Scénarios :

Un Scénario est une séquence et une description des scènes qui composent un événement et représente un cas d'utilisation qui peut avoir plusieurs événements.

Les tâches de chaque acteur et les scénarios conduisant à leur mise en œuvre sont définis dans le tableau Spécification des scénarios.

Le tableau ci-dessus représente la Spécification des scénarios de notre système :

Tableau 3*:Spécification des scénarios.*

Acteurs / Taches	Scénarios	
Archiviste	T1 : s'authentifier	S0 : accède à l'application web. S1 écrire username S2 écrire password
	T2 : gérer les Documents.	S3 : ajouter un Document. S4 : affecter un Document a un Dossier. S5 : modifier un Document. S6 : supprimer un Document. S7 : rechercher un Document.
	T3 : gérer les Dossiers.	S8 : ajouter un Dossier. S9 : affecter un Dossier a une Boite. S10 : modifier un Dossier. S11 : supprimer un Dossier. S12 : rechercher un Dossier.
	T4 : gérer les Boites.	S13 : ajouter une Boite. S14 : affecter une Boite a une Division. S15 : modifier une Boite. S16 : supprimer une Boite. S17 : rechercher une Boite.
	T5 : gérer les empruntes.	S18 ajouter une emprunte. S19 confirmer la remise d'une emprunte S20 annuler une emprunte.
	T6 : déconnexion	S21 : déconnexion
Administrateur	T1 : s'authentifier	S0 : accède à l'application web. S1 : écrire username S2 : écrire password

	T2 : gérer les utilisateurs	S3 : Ajouter utilisateur. S4 : Affecter rôle a un utilisateur S5 : Modifier Rôle a un utilisateur. S6 : Modifier un Utilisateur. S7 : Supprimer un utilisateur. S8 : chercher un Utilisateur.
	T3 : gérer les Documents.	S9 : ajouter un Document. S10 : affecter un Document a un Dossier. S11 : modifier un Document. S12 : supprimer un Document. S13 : rechercher un Document.
	T4 : gérer les Dossiers.	S14 : ajouter un Dossier. S15 : affecter un Dossier a une Boite. S16 : modifier un Dossier. S17 : supprimer un Dossier. S18 : rechercher un Dossier.
	T5 : gérer les Boites.	S19 : ajouter une Boite. S20 : affecter une Boite a une Division. S21 : modifier une Boite. S22 : supprimer une Boite. S23 : rechercher une Boite.
	T6 : gérer les Divisions.	S24 : ajouter une Division. S25 : modifier une Division. S26 : supprimer une Division. S27 : rechercher une Division.
	T7 : gérer les empruntes.	S28 ajouter une emprunte. S29 confirmer la remise d'une emprunte S30 annuler une emprunte.
	T8 : déconnexion	S31 : déconnexion

7. Diagramme de cas d'utilisation :

Diagrammes de cas d'utilisation et un système qui implique un ensemble de cas d'utilisation et un ensemble d'acteurs. Chaque cas d'utilisation représente une partie de la fonctionnalité que le système fournit. L'ensemble de cas d'utilisation montre la fonctionnalité complète du système à un certain niveau de détail. De même, chaque acteur représente un type d'objet pour lequel le système peut exécuter un comportement. L'ensemble d'acteurs représente l'ensemble complet d'objets que le système peut servir. Les objets accumulent le comportement de tous les systèmes avec lesquels ils interagissent en tant qu'acteurs**. [24]**

Le diagramme ci-dessus représente le Diagramme de cas d'utilisation global de notre système :

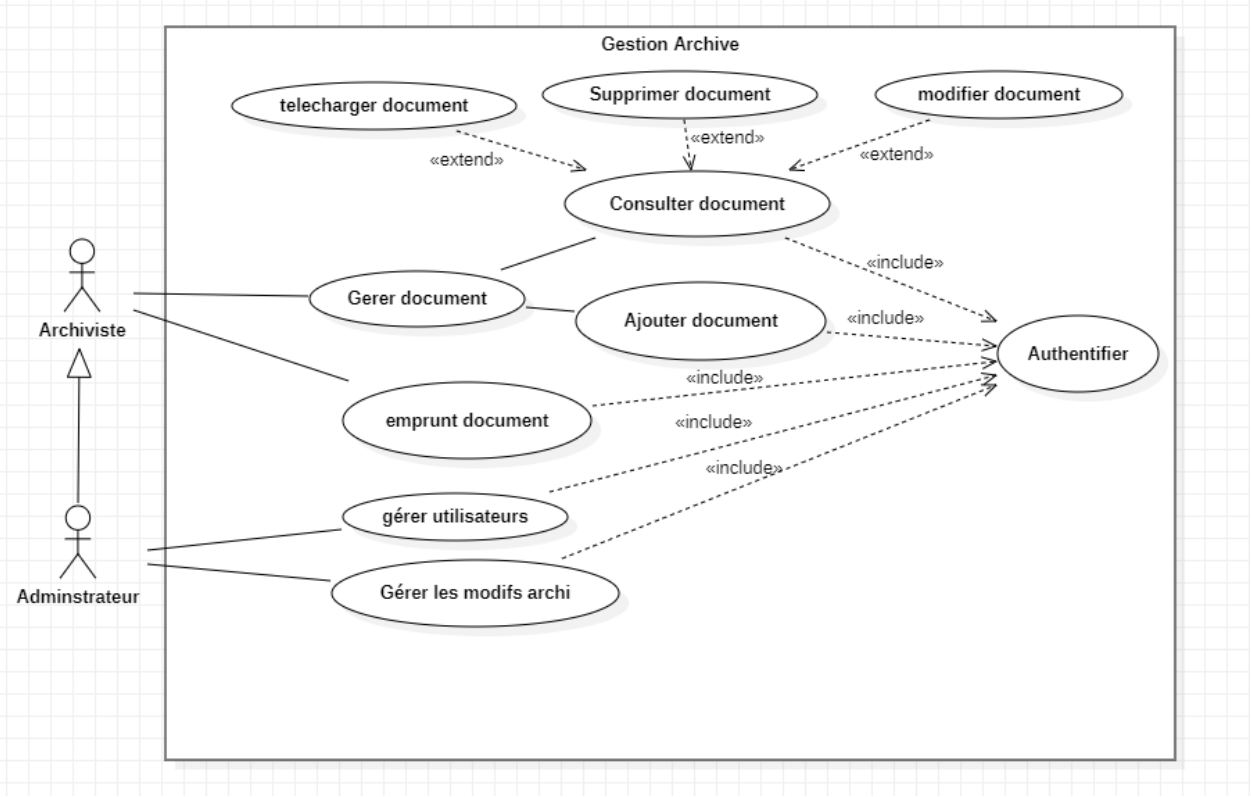


Figure 5*: Diagramme de cas d'utilisation.*

8. Diagrammes de séquence :

Un diagramme de séquence est un diagramme interactif qui décrit en détail comment les actions sont effectuées : quels messages sont envoyés et quand.

Les diagrammes de séquence sont tracés en fonction du temps. Le temps passe au fur et à mesure que vous faites défiler la page. Les objets participant à l'opération sont répertoriés de gauche à droite en fonction du moment où ils participent à la séquence de messages. **[25]**

- Diagramme de séquence du cas d'utilisation « Authentification » :

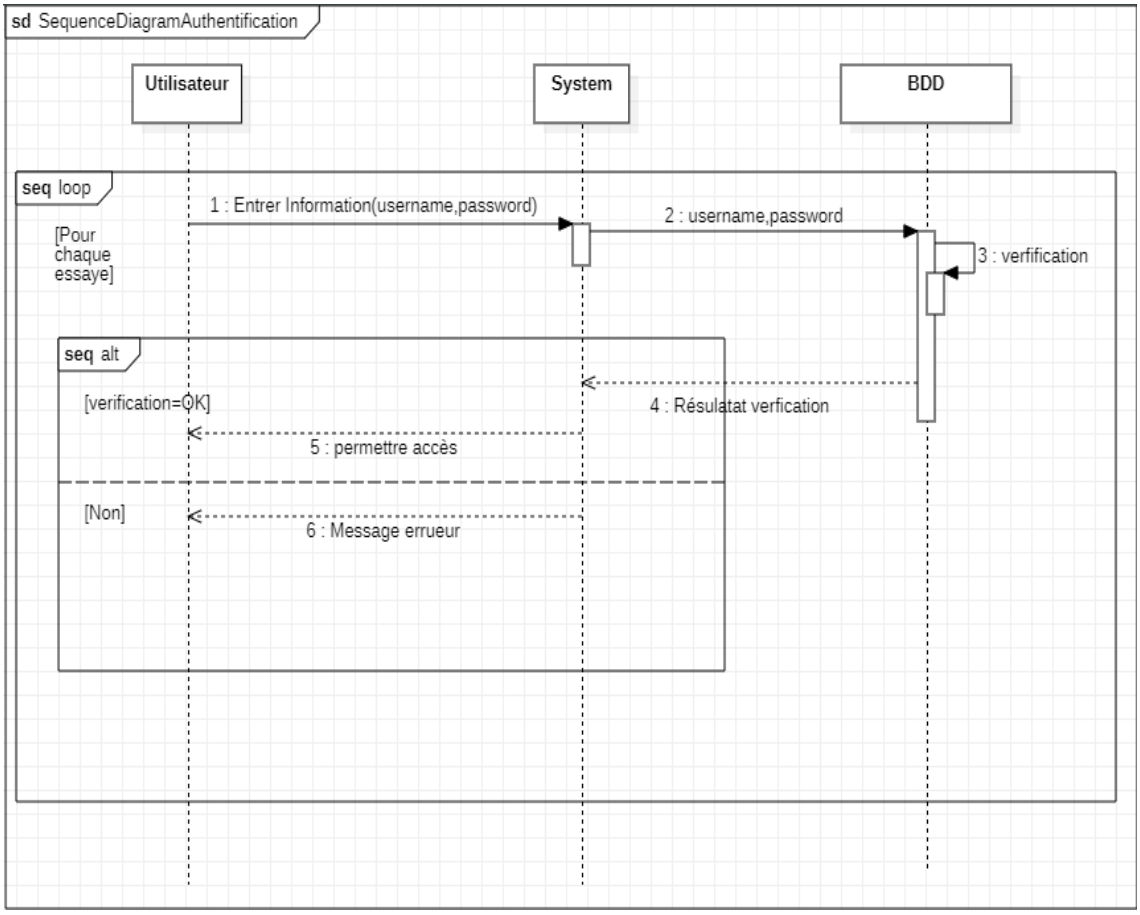


***Figure 6**: Diagramme de séquence du cas d'utilisation «Authentification »*

- Diagramme de séquence du cas d'utilisation « L'ajout » :

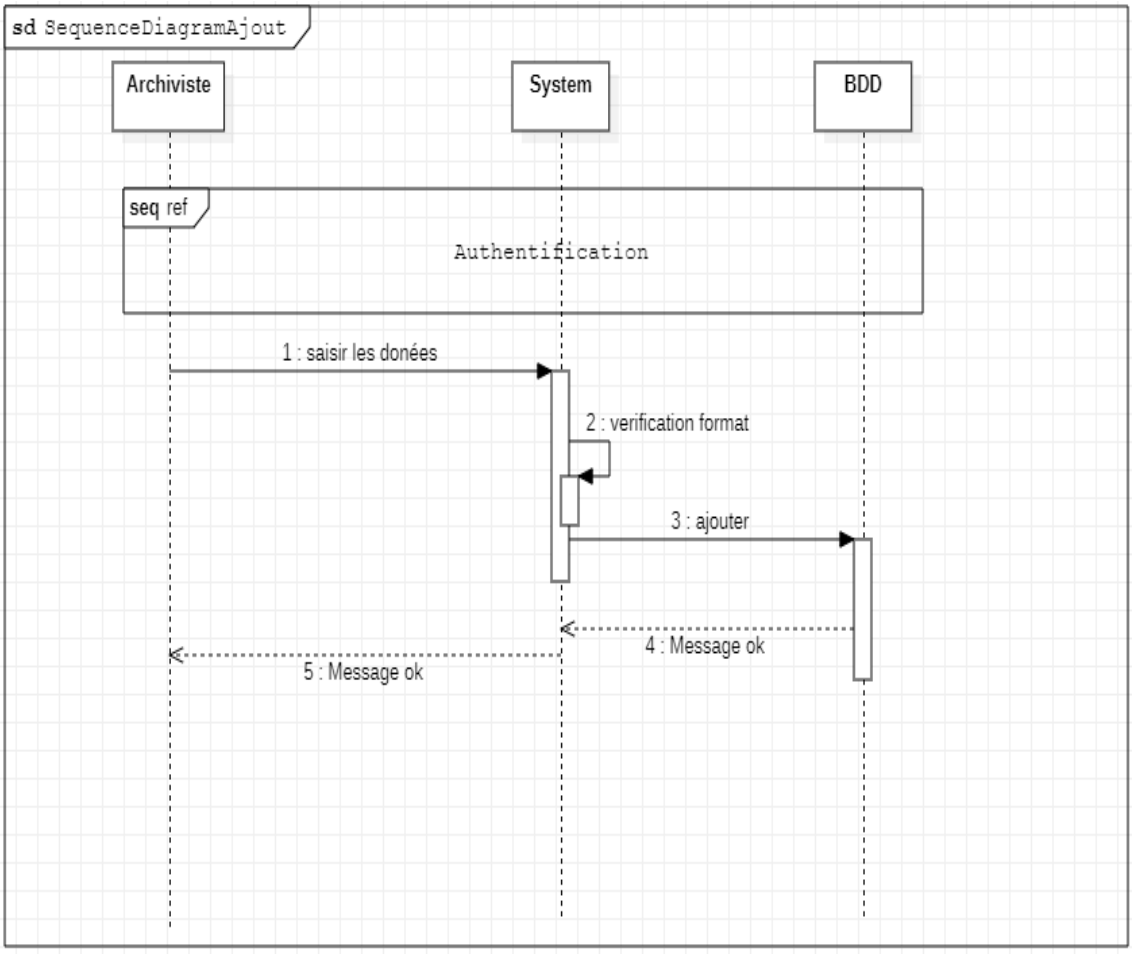


***Figure 7**: Diagramme de séquence du cas d'utilisation « L'ajout »*

- Diagramme de séquence du cas d'utilisation «Consultation » :

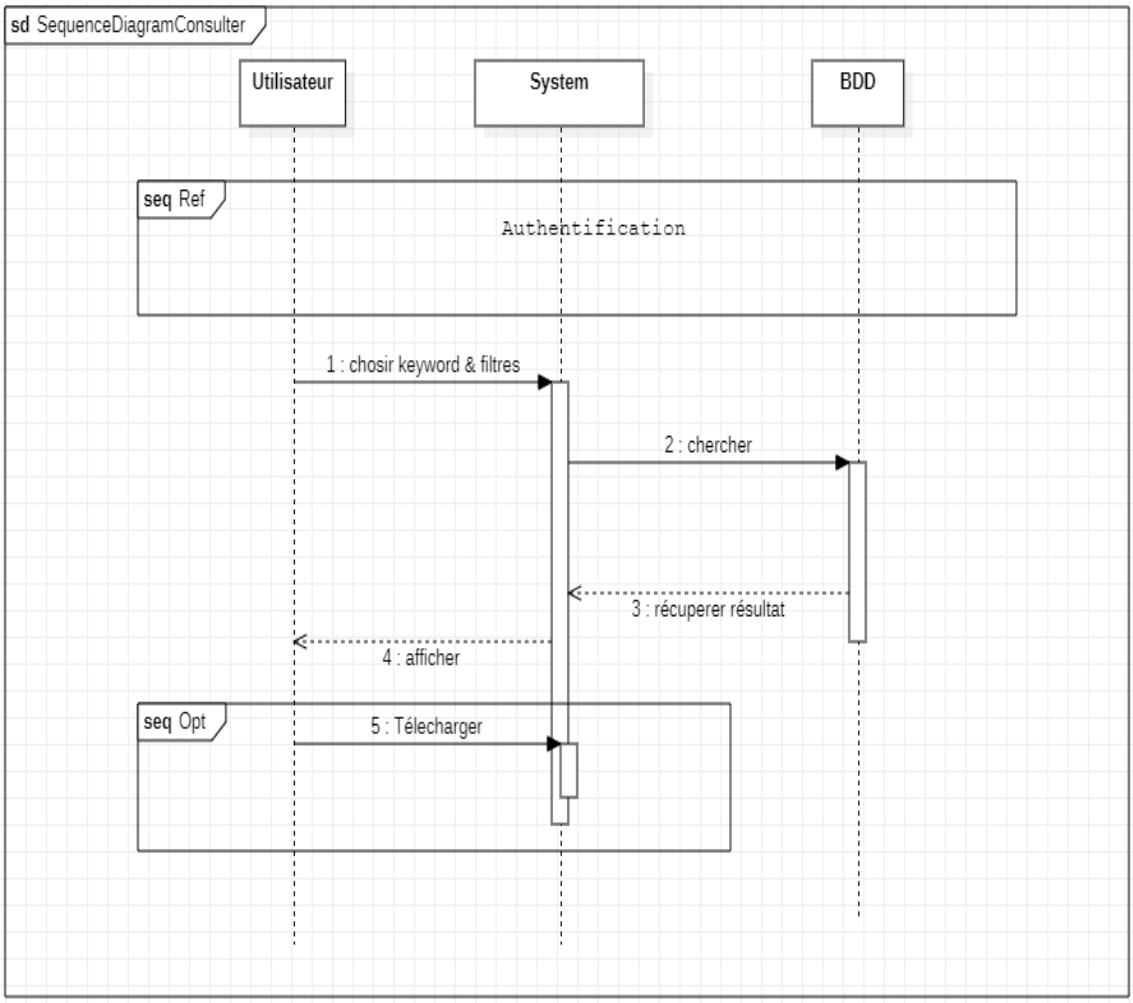


***Figure 8**: Diagramme de séquence du cas d'utilisation «Consultation»*

- Diagramme de séquence du cas d'utilisation « Suppression » :

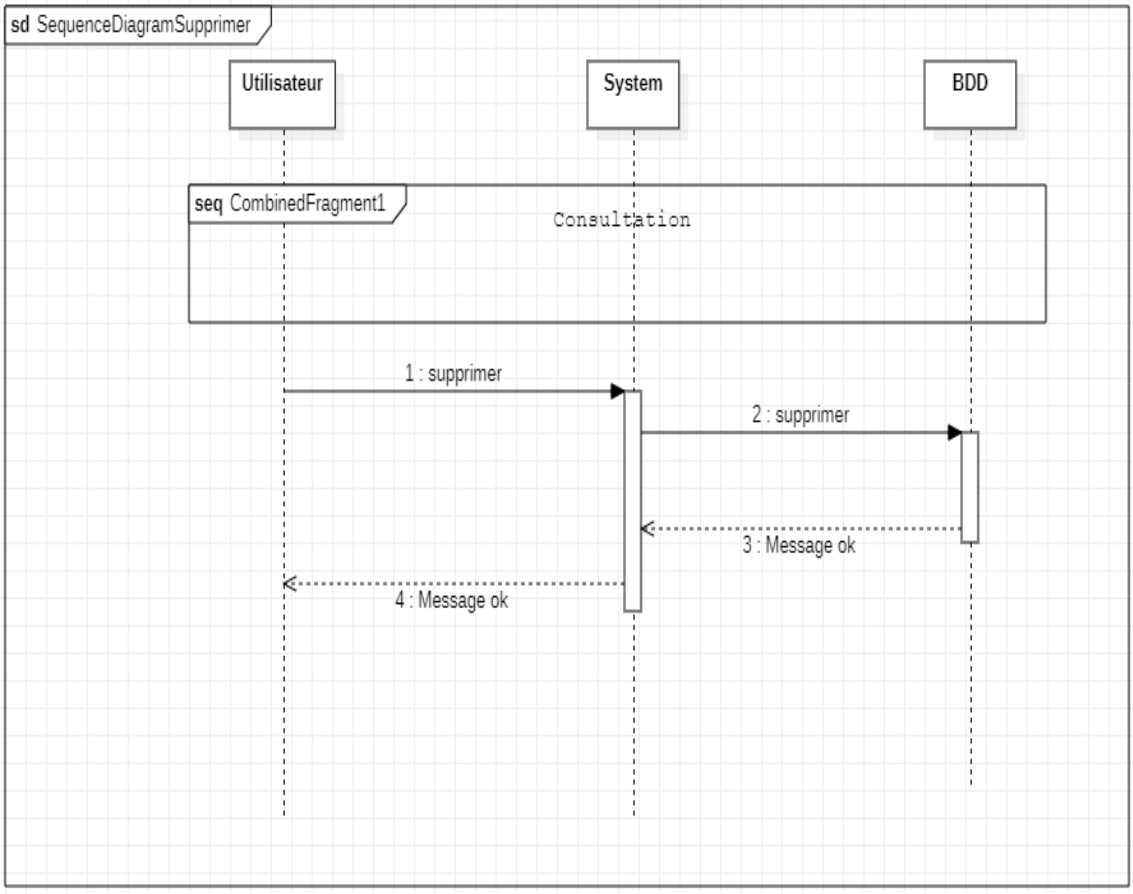


***Figure 9**: Diagramme de séquence du cas d'utilisation « Suppression »*

➢ Diagramme de séquence du cas d'utilisation « Modification » :

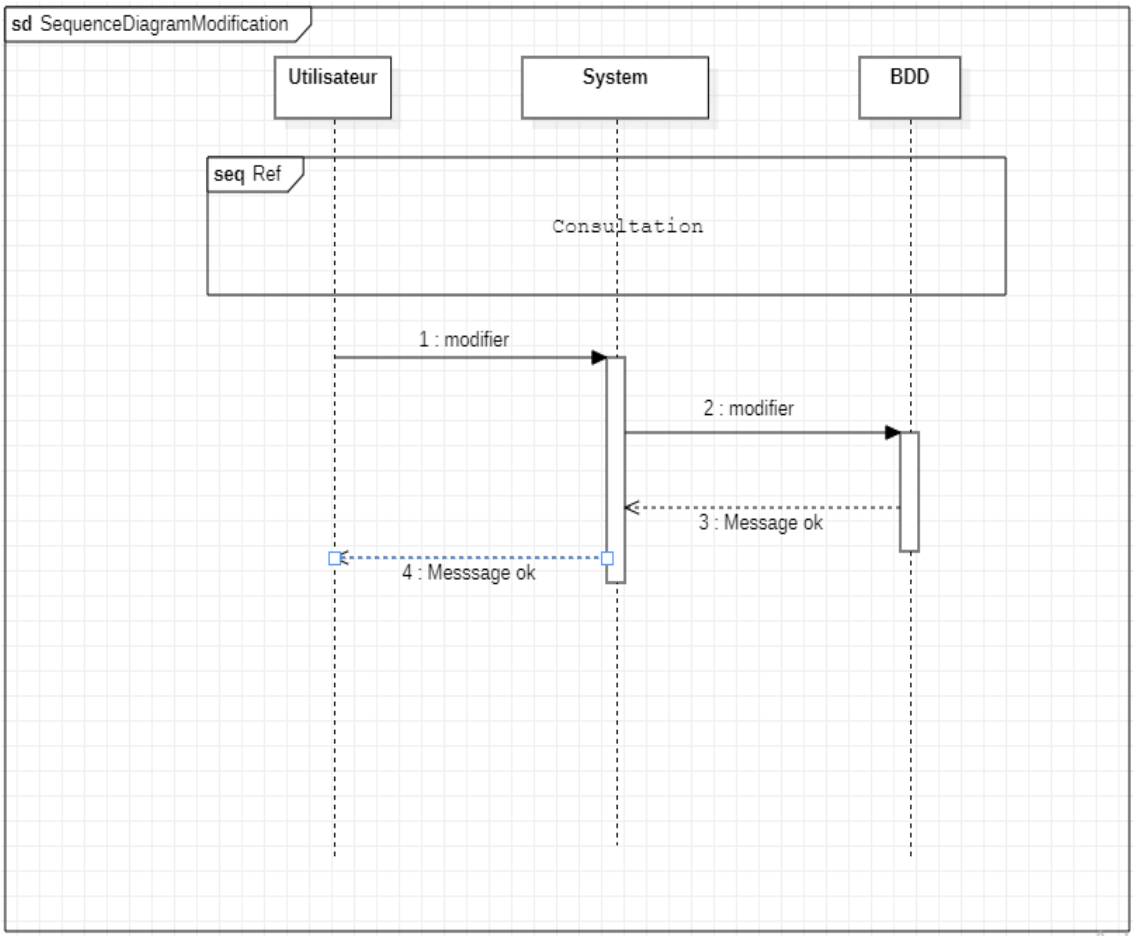


***Figure 10**: Diagramme de séquence du cas d'utilisation « Modification »*

9. Diagramme de communication :

Les diagrammes de communication ressemblent aux diagrammes d'objets dans lesquels une ligne de vie représente les objets de l'interaction et les flèches représentent les messages transmis entre les lignes de vie. Les pointes de flèches indiquent la direction des messages et les numéro de séquence indiquent l'ordre de transmission des messages**. [26]**

Le diagramme ci-dessus représente la communication entre deux micro services dans notre système : le "Service Data" et le "Service de Sécurité".

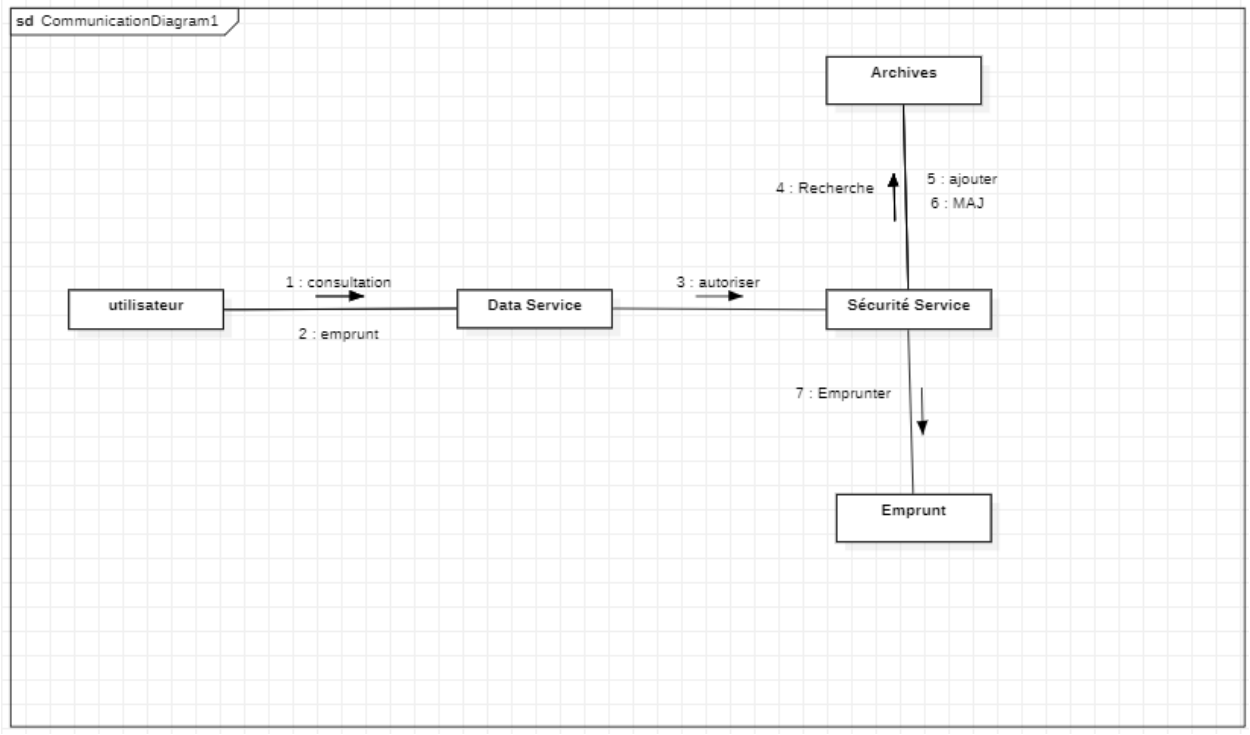


***Figure 11**: Diagramme de communication*

10. Sécurité et Gestion des Erreurs :

Le service de sécurité est responsable de garantir l'authentification et l'autorisation appropriées pour les opérations CRUD (Create, Read, Update, Delete) effectuées par le Service Data. Pour ce faire, il utilise JSON Web Tokens (JWT) pour sécuriser les requêtes entrantes.

- **Création d'un JWT** : Lorsqu'un utilisateur authentifié souhaite effectuer une opération via le Service Data, le service de sécurité génère un JWT contenant des

informations d'authentification et d'autorisation pertinentes. Ce JWT est signé numériquement pour garantir son intégrité.

- **Inclusion du JWT dans les requêtes :** L'utilisateur doit inclure ce JWT dans les requêtes envoyées au Service Data en tant que partie des en-têtes HTTP. Le JWT est généralement inclus dans l'en-tête "Authorization" de la requête.
- **Vérification du JWT :** Lorsqu'une requête atteint le Service Data, ce dernier extrait le JWT de l'en-tête de la requête. Le service de sécurité vérifie ensuite la validité du JWT, en s'assurant qu'il n'a pas été modifié et qu'il n'a pas expiré. Il valide également la signature numérique pour garantir l'authenticité du JWT.
- **Attribution des autorisations :** Après avoir vérifié le JWT avec succès, le service de sécurité identifie l'utilisateur et extrait les informations d'autorisation appropriées du JWT. Ces informations déterminent les opérations CRUD que l'utilisateur est autorisé à effectuer sur les ressources d'archivage. Le service de sécurité accorde alors les autorisations nécessaires au Service Data pour exécuter l'opération demandée.
- **Gestion des Erreurs :** En cas de JWT invalide, modifié ou expiré, le service de sécurité renvoie un code d'erreur approprié, généralement un code d'état HTTP 401 (Non autorisé), indiquant que l'authentification a échoué. De plus, il peut fournir un message d'erreur explicatif pour aider à diagnostiquer le problème.

11. Documentation :

11.1 Micro service Data

Le Micro service Data est responsable de la gestion des archives administratives et électorales de la Cour Constitutionnelle. Il expose une API REST sécurisée qui permet aux utilisateurs autorisés d'effectuer des opérations CRUD (Create, Read, Update, Delete) sur les archives.

- **Point d'Accès :** URL de base: https:// localhost:9191/api/Data.

- **Opérations Supportées** : Création d'une nouvelle archive : POST /archive, Lecture d'une archive : GET /archive/{id}, Mise à jour d'une archive : PUT /archive/{id} et Suppression d'une archive : DELETE /archive/{`id}.`
- **Authentification et Autorisation :** L'authentification est basée sur JSON Web Tokens (JWT). Les utilisateurs doivent inclure leur JWT valide dans l'en-tête "Authorization" de la requête.

  Les autorisations sont attribuées en fonction des informations du JWT, notamment les rôles de l'utilisateur.
- **Formats de Données :** Les données sont échangées au format JSON.

Exemple de modèle de Document JSON :

Figure 12 : *Exemple de Formats de Données Microservice Data*

- **Gestion des Erreurs** : Les erreurs sont signalées avec des codes d'état HTTP appropriés (par exemple, 400 pour une demande incorrecte, 401 pour une authentification échouée, 403 pour une autorisation refusée, etc.).

  Des réponses d'erreur JSON sont fournies avec des détails explicites pour aider à diagnostiquer les problèmes.

11.2 Micro service de Sécurité

Le micro service de sécurité est chargé de gérer l'authentification et l'autorisation des utilisateurs pour accéder au micro service d'archivage. Il expose une API REST pour la gestion des comptes utilisateur et la délivrance des JWT.

- **Point d'accès :** URL de base : https:// localhost:9292/api/securite.

- **Opérations Supportées :** Création d’un compte utilisateur : POST /utilisateur et Authentification d’un utilisateur : POST /authentification**.**
- **Authentification et Autorisation** : L'authentification est effectuée en vérifiant les informations de l'utilisateur (nom d'utilisateur et mot de passe) lors de l'opération d'authentification.

  Les autorisations pour l'accès aux différentes opérations sont gérées par le micro service d'archivage en fonction des rôles définis dans le JWT.
- **Formats de Données** : Les données sont échangées au format JSON.

Exemple de modèle d'utilisateur JSON :

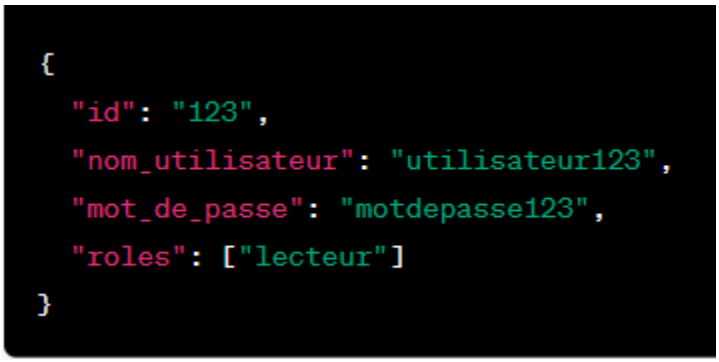


Figure 13 : *Exemple de Formats de Données Micro service Sécurité*

- **Gestion des Erreurs :** Les erreurs sont retournées avec des codes d'état HTTP appropriés (par exemple, 400 pour une demande incorrecte, 401 pour une authentification échouée, 403 pour une autorisation refusée, etc.).

  Des messages d'erreur JSON explicatifs sont inclus pour faciliter le débogage et la résolution des problèmes.

12. Diagramme de Classe :

Un diagramme de classes représente les blocs de construction de tout système orienté objet.

Les diagrammes de classe représentent une vue statique du modèle. Ou une partie du modèle qui décrit ses propriétés et son comportement, plutôt que de détailler les méthodes pour réaliser l’opération.

Il illustre les relations entre les classes et les interfaces. Les généralisations, les résumés et les associations sont tous précieux et reflètent l'héritage, la composition ou l'utilisation ou les connexions. **[26]**

Le diagramme ci-dessus représente le diagramme de classe global de notre système :

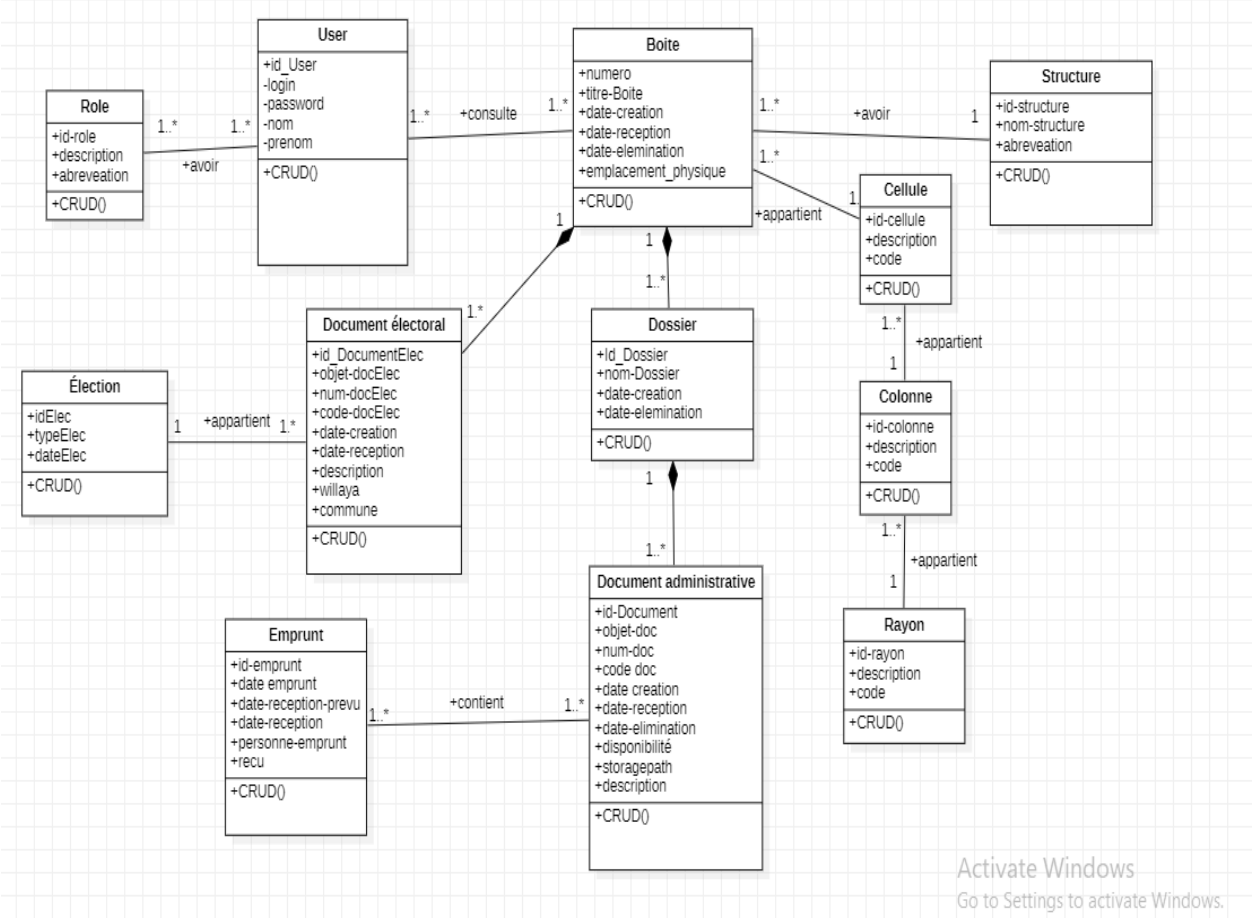


***Figure 14**: Diagramme de classe*

- **Classe "Utilisateur" :** La classe "Utilisateur" représente les individus qui interagissent avec notre système. Chaque utilisateur peut être associé à un ou plusieurs "Rôles" pour déterminer ses autorisations.
- **Classe "Rôle"** : La classe "Rôle" définit les différentes autorisations et privilèges qu'un utilisateur peut avoir.

- **Classe "Boite"** : La classe "Boite" représente les boîtes d'archives administratives et électorales. Chaque boîte est associée à une "Structure" et à une "Cellule, Colonne, Rayon" qui définissent son emplacement.
- **Classe "Structure":** La classe "Structure" représente les structures organisationnelles auxquelles les boîtes peuvent être attribuées.
- **Classe "Cellule, Colonne, Rayon"** : Cette classe représente la hiérarchie de localisation des boîtes dans notre système. Une boîte peut être située dans une "Cellule", une "Colonne" et un "Rayon" spécifiques.
- **Classe "Document Électoral"** : La classe "Document Électoral" représente les documents liés aux élections. Chaque document électoral est associé à une "Élection" spécifique.
- **Classe "Élection"** : La classe "Élection" représente les élections auxquelles les documents électoraux sont liés. Chaque document électoral appartient à une élection spécifique.
- **Classe "Dossier"** : La classe "Dossier" représente les dossiers d'archives qui peuvent être stockés dans une boîte. Chaque dossier est composé de "Documents Administratifs" qui peuvent être empruntés.
- **Classe "Document Administratif"** : La classe "Document Administratif" représente les documents administratifs stockés dans les dossiers. Ils peuvent être empruntés par les utilisateurs pour consultation.

13. Passage au modèle relationnel :

Lors de la transformation d'un diagramme de classe en modèle relationnel, certaines règles clés sont suivies :

- Les classes sont représentées par des tables, permettant de structurer les données de manière organisée.
- Les attributs des classes deviennent des colonnes dans les tables correspondantes.
- Les relations entre les classes sont traduites en clés étrangères, assurant l'intégrité référentielle.

- Les associations de type plusieurs-à-plusieurs nécessitent l'ajout d'une table de jonction pour maintenir les relations.

Le tableau ci-dessus représente le Modéle Relationnel de notre système :

***Tableau 4**: Modéle Relationnel.*

Nom Table	Attributs
User	**id_User**, login, password, nom, prenom
Role	**id_Role**, description, abreveation
User_Role	**#id_User, #id_Role**
Structure	**id-structure**, nom-structure, abreveation
Boite	**numero,# id-structure**, titre-Boite, date-creation, date-reception, date-elimination, emplacement-physique
User_Boite	**#id_User,# numero**
Dossier	**id_Dossier,# numero,** nom-dossier, date-creation, date-elimination
Document administrative	**id_Document**,**#id_Dossier,** objet-doc, num-doc, code-doc, date-creation, date-reception, date-elimination, disponibilité, storagepath, description
Document électoral	**id_DocumentElec**,#numero,#idElec, objet-docElec, num-docElec,Code -docElec, date-creation, date-reception, date-elimination, description, willaya, commune
Élection	**idElec,** typeElec, dateElec
Cellule	**ID_Cellule,#id-Colonne, description, code**
Colonne	**ID_Colonne,#id-rayon, description, code**
Rayon	**id-rayon,** description, code
Emprunt	**id-emprunt,** date emprunt, date reception-prevu, date-reception, personne-emprunt, recu
Emprunt_ Document administrative	**#id-emprunt,# id_Document**

14. Conclusion :

Ce chapitre présente les tâches de notre programme. Pour réaliser, nous avons proposé une solution basée sur l'analyse et la conception du modèle en utilisant le langage de modélisation unifié UML.

Pour ce faire, nous avons défini les acteurs de notre programme, les tâches qu'ils effectuent et les scénarios associés à chaque tâche.

Nous avons créé diagramme de cas d'utilisation pour les acteurs, des diagrammes de séquence et enfin un diagramme de classe. Dans le chapitre suivant, on va traiter l'implémentation de notre propre application.

Chapitre 3 : Implémentation et Réalisation

1. Choix de Technologie Pour Micros-Service

Spring Boot, Spring Security, REST APIs et Angular sont des technologies très utilisées dans le développement d'applications web basées sur une architecture micro services. Ces technologies offrent de nombreux avantages pour la création, le déploiement et la sécurisation de services web modulaires et évolutifs.

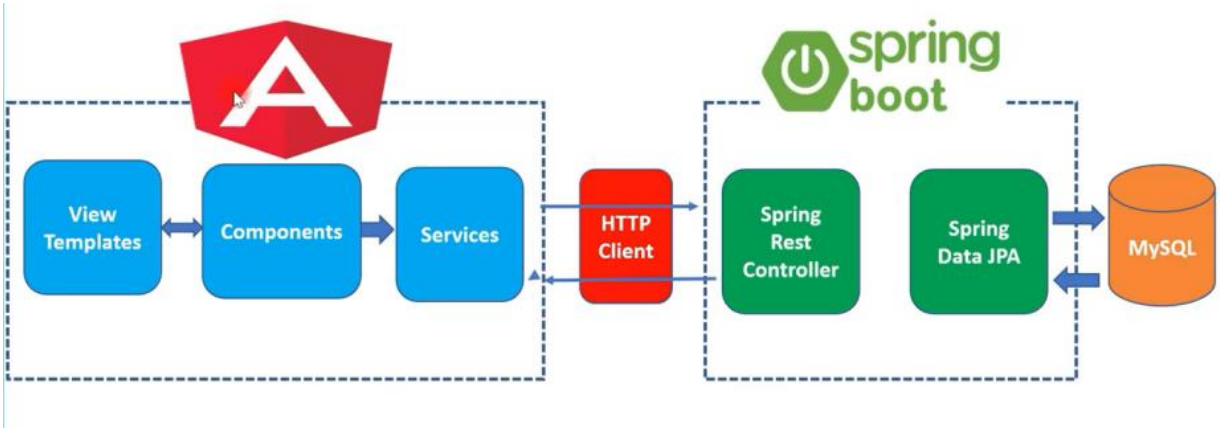


***Figure 15**; Spring Boot, Spring Security, REST APIs, and Angular.*

.

1.1 Spring Boot :

Un Framework qui simplifie la configuration et le démarrage d'applications Spring. Il permet de créer des applications autonomes et prêtes à l'exécution, sans avoir besoin de dépendre d'un serveur d'applications externe. Spring Boot fournit également des fonctionnalités pour tester, surveiller et gérer les applications.

Dans un projet SpringBoot doit trouver :

- **Couche Contrôleur :** qui contient les EndPoints des API REST.
- **Couche Service :** qui contient la logique de l'application.
- **Couche DAO (Data Access Object):** responsable de la communication avec la base de données**.**

On utilise Le Serveur **TomCat** pour interpréter les requêtes http arrivant au port associé au protocole http et fournir la réponse avec le même protocole.

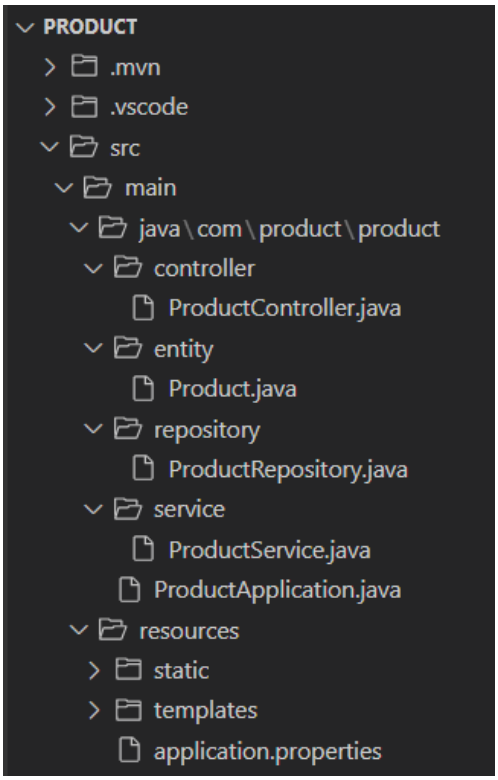


***Figure 16**: Structure Projet Spring Boot.*

1.2 Spring Security:

Un module de Spring qui offre des solutions pour l'authentification et l'autorisation des utilisateurs et des ressources. Il intègre facilement avec les standards et les protocoles de sécurité tels qu'OAuth2, JWT, LDAP, etc. Spring Security permet de protéger les Endpoints des API REST contre les accès non autorisés ou malveillants.

1.2.1 Systèmes d'authentification :

Deux Types de modèles d'authentification :

- **Stateful:** Les données de la session sont enregistrées coté serveur d'authentification comme des objets sessions. Les sessions ont un time out (exemple : 20 minutes de rien utilisé, la session est éliminée).

Tanque la session est active, chaque requête HTTP envoyée par le client avec l'**ID** de session sera acceptée par le serveur**.**

- **Stateless:** les données de la session sont enregistrées coté comme des JWT (Jason web Token). Chaque requête HTTP envoyée par le client avec le JWT sera acceptée par le serveur.

Ce JWT contient des infos comme : Username, Rôles, … et une **signature.** Si le JWT change (exemple : rôle modifié), la signature aussi change et l'authentification sera bloquée par le serveur.

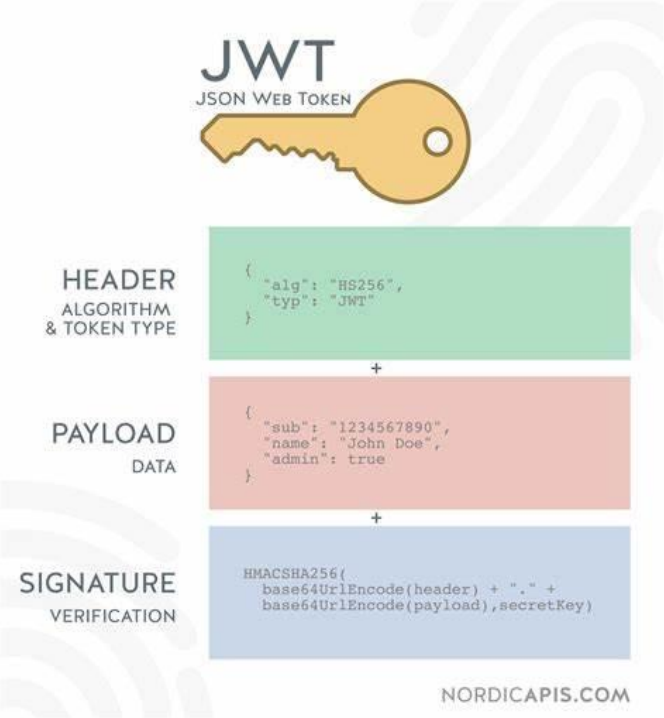


***Figure 17**: JWT (Jason Web Token)*

Pourquoi choisir l'option « sateless » plutôt que l'option « stateful » ?

Dans l'Authentification Stateful, il existe une faille de sécurité **csrf** (cross site request forgery), Les IDs des sessions sont stockées comme des cookies et le navigateur envoi les cookies avec chaque requête HTTP.

Des requêtes HTTP peuvent être forcées avec des e-mails (phishing emails) et le serveur les acceptera tant que la session est disponible.

1.3 REST APIs :

Des interfaces de programmation qui exposent les données et les fonctionnalités d'une application web à travers le protocole HTTP. Elles utilisent un style architectural qui favorise l'échange de données au format JSON ou XML, en respectant les principes de l'architecture REST (Representational State Transfer). Les API REST facilitent l'intégration et la communication entre les différents services web. **[28]**

1.4 Angular :

Un Framework front-end qui permet de créer des applications web dynamiques et réactives. Il utilise le langage **TypeScript**, qui est une extension du JavaScript, et offre des fonctionnalités telles que le data binding, les composants, les directives, les services, les pipes, etc. Angular se connecte aux API REST pour consommer et afficher les données dans le navigateur. **[29]**

Angular utilise une architecture MVVM (Modèle Vue Modèle) qui est similaire à MVC (Modèle Vue Contrôleur) sauf que le modèle et le contrôleur sont groupés.

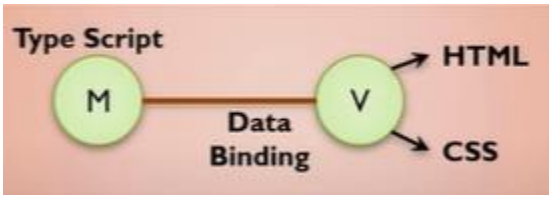
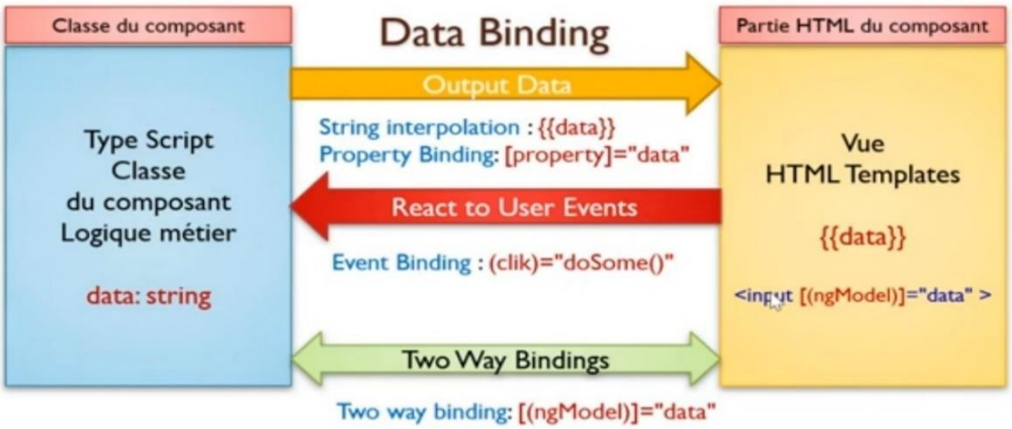


***Figure 18**: Architecture MVVM.*



La communication entre le modèle et vue ce fait via **Data Binding**.

***Figure 19**: Data Binding.*

- **Output Data** : Utiliser une variable ou un traitement de la vue dans le modèle. En utilisant le design pattern observé. Elle fait subscribe à chaque changement.
- **React to User event:** Data Binding vers l'autre sense. Faire un traitement dans le modèle au moment où un évènement dans la vue se fait.
- **Two way Binding:** Les deux côtés font subscribe et donc un changement d'une côté affecte l'autre.

On utilise une architecture **client-serveur** pour communiquer avec la partie backend (serveur) via des requêtes HTTP ou des websockets.

- **Le client** est composé de plusieurs composants, qui sont des éléments réutilisables de l'interface utilisateur. Chaque composant a une logique interne, une vue et un style.
- **Le serveur** est responsable de fournir les données et le logique métier aux composants du client. (Node js pour Angular).
- **Requête** : Est un message envoyé du client a un serveur comportant la tâche à exécuter.
- **Réponse** : Est un message émis par un serveur à un client comporte les éléments les résultats de la Requête de client.

Les Avantages de l'architecture client/serveur sont :

- **Des ressources centralisées** : Étant donné que le serveur et au centre du réseau peut gérer des ressources communes à tous les utilisateurs comme par exemple une base de données centralisés afin d'éviter des problèmes de redondance et de coordination.
- **Une meilleure sécurité :** Car le nombre de points d'entrée permanentent l'accès aux donnes est moins important.
- **Une administration au niveau serveur :** Les clients ayant peu d'importance dans ce modèle, ils ont besoin d'être administrés.
- **Un réseau évolutif :** Il est possible de supprimer ou rajouter des clients sans perturber le fonctionnement du réseau et sans modification majeur.

L'inconvénient de l'architecture client/serveur :

L'architecture client/serveur a tout même quelques lacunes. Le serveur est le seul maillon faible du réseau client-serveur étant donné que tout le réseau est architecturé autour de lui, heureusement le serveur à une grande tolérance aux pannes (notamment grâce à son système RAID). **[16]**

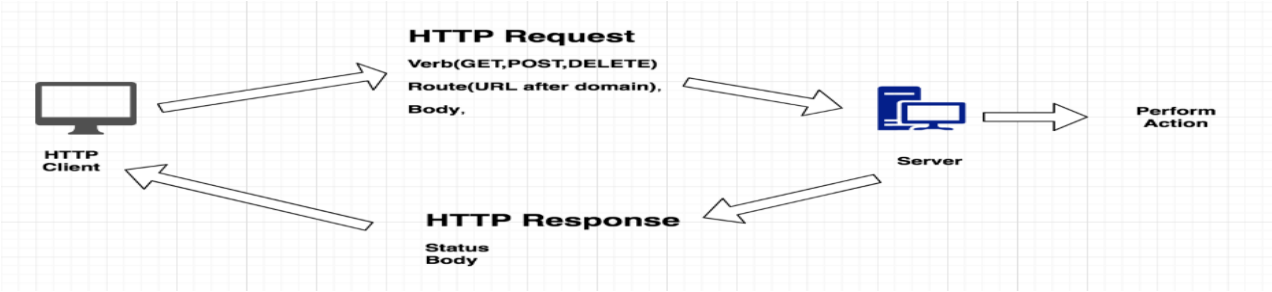


***Figure 20**: Architecture Client-Server with Rest API.*

Angular utilise les **components** qui sont introduites depuis HTML5, chaque component a :

- Une partie HTML
- Une partie CSS.
- Une partie TypeScript.

En gros, au lancement de l'application angulaire, une page index.html et afficher, et nous manipulons le(s) composant(s) qui s'affiche(nt)

***Figure 21**: Exemple de Components.*

2. Outils de développement :

2.1 Vs Code :

Vs Code est un IDE puissant pour le développement web, mobiles,……Il fournit un support complet pour des centaines des langages de programmation tel que : Java, JavaScript, Python, C++…. il suffit juste de choisir les bonnes extensions à ajouter.

L'un des avantages de cet IDE est qu'il ne consomme pas trop de mémoire et de processeur.et il donne des résultats similaires que d'autre comme Intellij et apacheNetbeans qui sont un peu lord pour quelque PC.

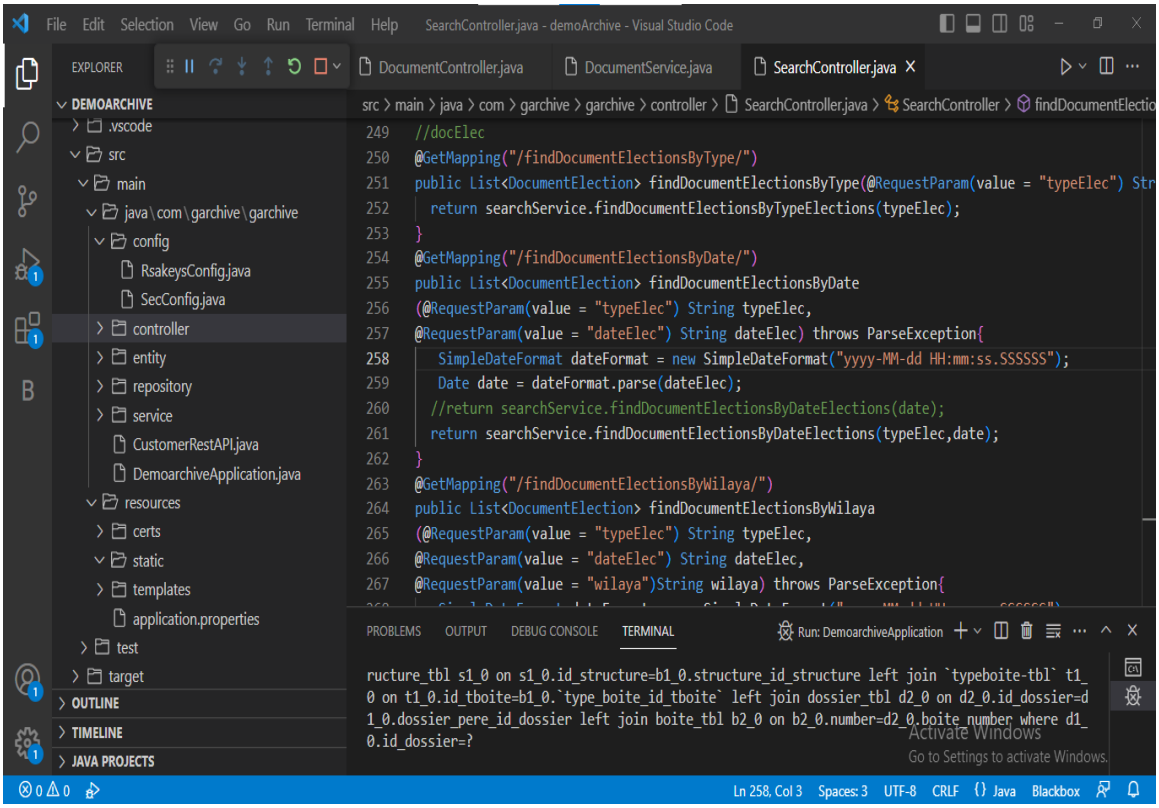


***Figure 22**: Vs Code.*

2.2 MYSQL Workbench

MySQL Workbench est un logiciel de gestion de base de données MySQL qui permet de gérer des tables (ajout, modification, suppression) à travers une interface graphique simple d'usage, une connexion peut être réalisée avec une instance MySQL afin d'importer un modèle sur un environnement de développement ou de production.

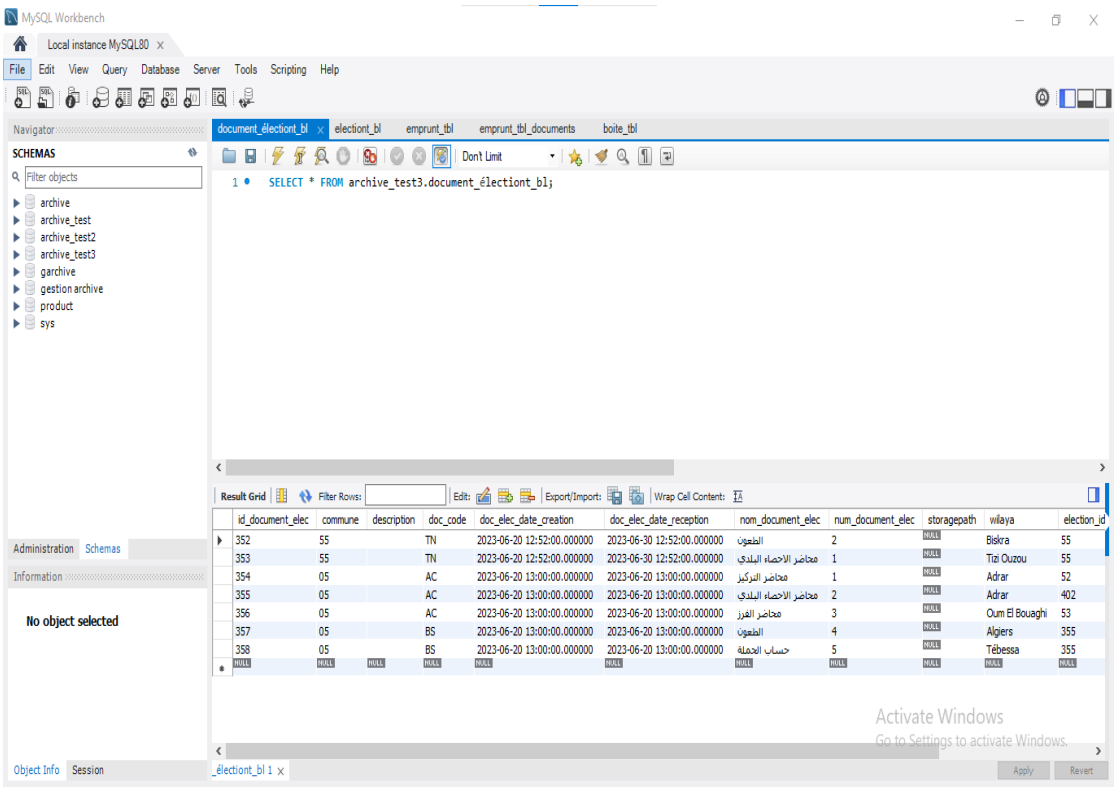


***Figure 23**: MYSQL Workbench*

2.3 Postman

Postman est un logiciel qui permet d'appeler et tester une API (Dans notre cas via des requêtes HTTP).Il simplifier chaque étape du cycle de vie des API et de rationaliser la collaboration, afin de créer, plus facilement et plus rapidement, de meilleures API. **[34]**

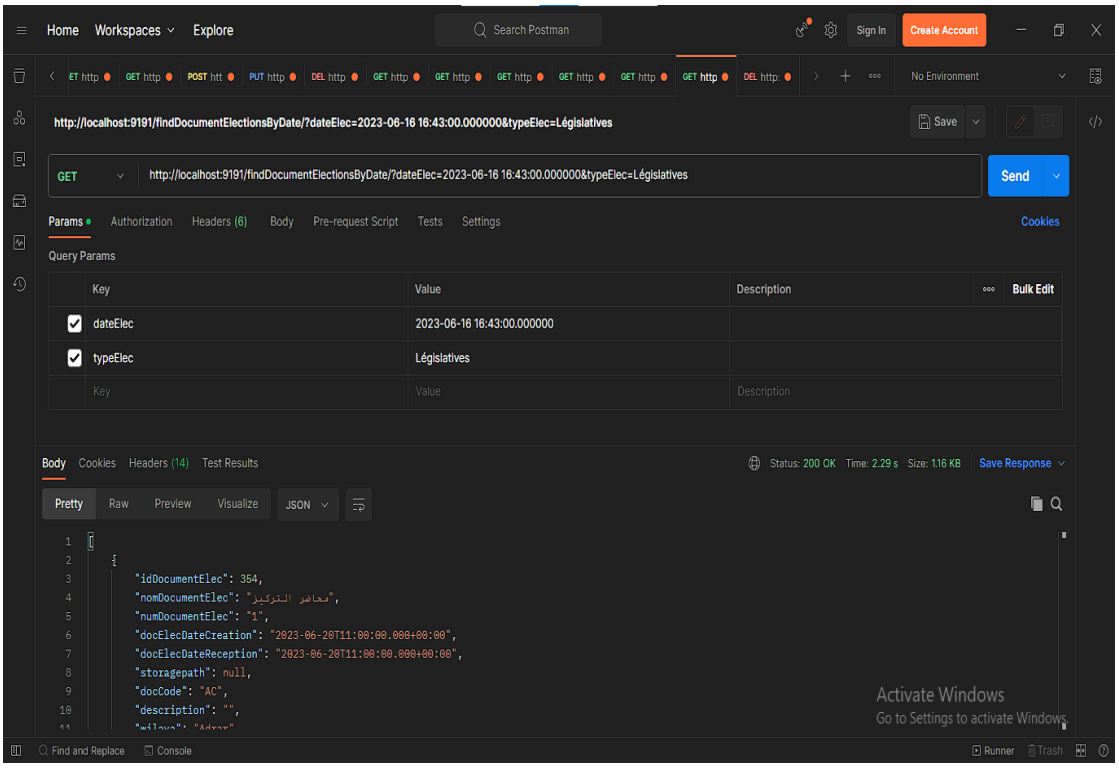


***Figure 24**: Postman.*

3. Interfaces Graphiques :

En représente ici quelque interface graphique de notre application :

- Un utilisateur doit saisir son username et mot de passe pour avoir l'accès à l'application et les privilèges.

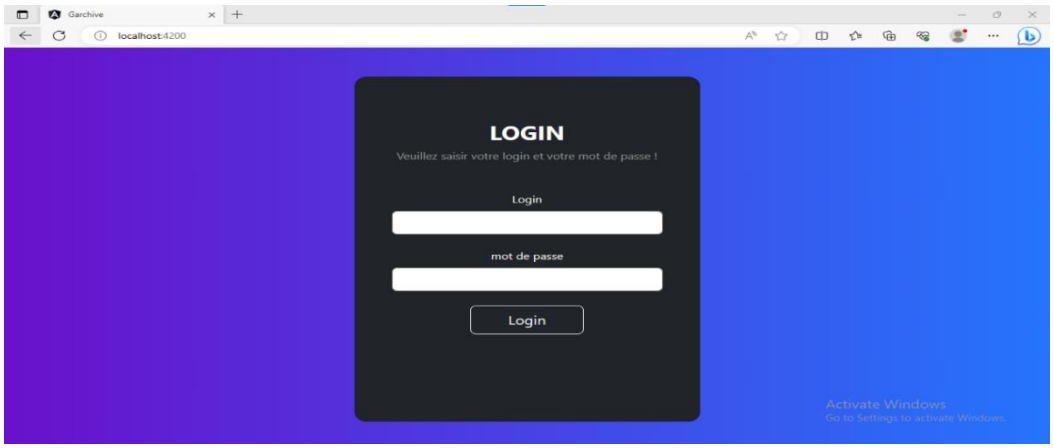


***Figure 25**: Login.*

- Un utilisateur doit choisir si on veut l'archive de la cour ou de conseil.

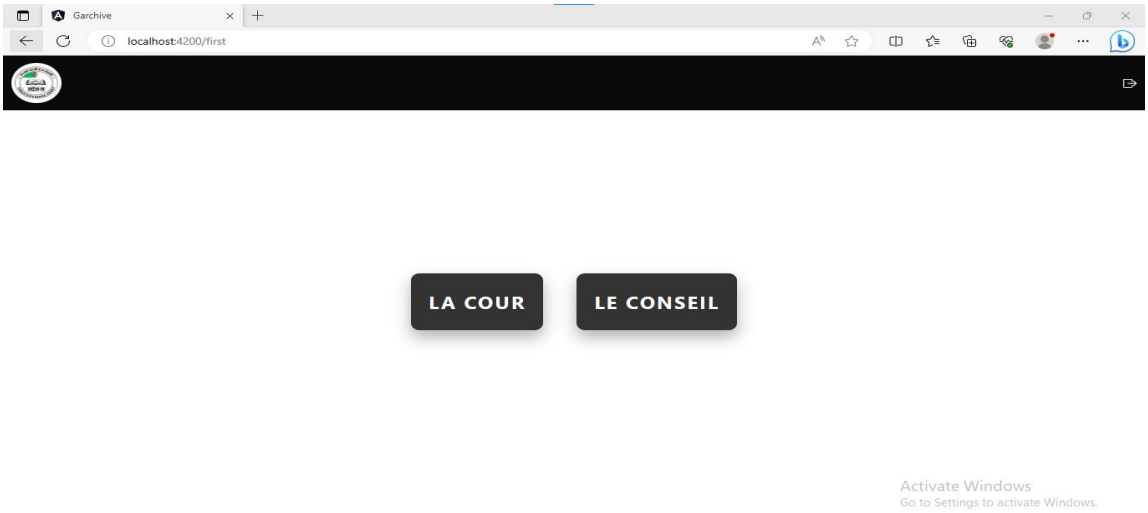


***Figure 26**: Interface choisir Archive.*

- Cette Interface Permet à l'utilisateur d'afficher, chercher, modifier, supprimer, visionner et télécharger un document administratif.

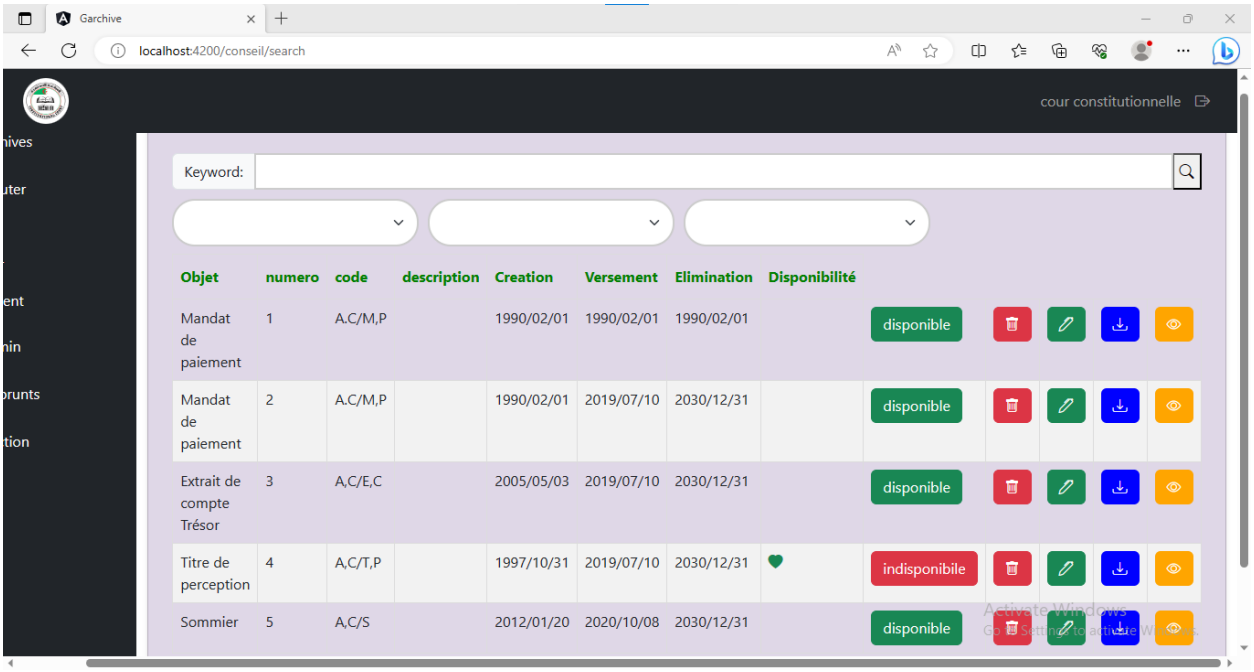


***Figure 27**: Interface gérer Documents administratifs.*

- Cette Interface Permettre à l'utilisateur d'afficher, chercher, modifier, supprimer, visionner et télécharger un document électoral.

***Figure 28**: Interface gérer Documents électorales..*

4. Conclusion :

Ce chapitre présente les technologies utilisées pour la réalisation de notre application : Spring Boot, Spring Security, REST APIs et Angular, ainsi que les outils utilisés : Vs code, MySQL workBench et Postman et quelques interfaces graphiques de l'interface.

Conclusion Générale

La réalisation de ce projet a été une expérience extrêmement enrichissante au cours de laquelle nous avons pu acquérir de nouvelles connaissances approfondies sur le développement d'applications web, ainsi que sur divers outils, langages et technologies web tels que Java, TypeScript, SQL, HQL (Hibernate Query Language), Postman, VsCode, etc. Cette expérience nous a également permis d'avoir une vue d'ensemble sur les microservices et leur utilité dans le contexte spécifique de notre projet.

De plus, nous avons eu l'opportunité de comprendre en profondeur le fonctionnement du département des archives et d'observer comment le cahier des charges est discuté. Nous avons également acquis une compréhension pratique de la façon dont les serveurs fonctionnent et de la procédure de déploiement de notre application sur ces serveurs.

En conclusion, nous sommes confiants que notre projet répond aux besoins établis et qu'il satisfera pleinement les utilisateurs. Dans les perspectives futures, nous envisageons d'enrichir davantage cette application en y ajoutant de nouvelles fonctionnalités et en la rendant accessible à un plus grand nombre d'utilisateurs. Nous espérons sincèrement que notre contribution aidera la Cour constitutionnelle dans la gestion efficace de son archive administrative et électorale.

Références

1. Site officielle de la cour constitutionnelle : Accueil - Cour constitutionnelle (cour-constitutionnelle.dz).
2. Autorité nationale indépendante des élections (ANIE) en Algérie : www.anie.dz
3. JOURNAL OFFICIEL DE LA REPUBLIQUE ALGERIENNE N° 25 11 Ramadhan 1443 12 avril 2022 : www.joradp.dz
4. *Présentation du conseil constitutionnel algérien*. Available at: https://cdn.accf-francophonie.org/2019/04/Algerie-presentation-generale.pdf (Accessed: 05 June 2023).
5. Article 11 et 12 du décret présidentiel n° 22-93 du 5 Chaâbane 1443 correspondant au 8 mars 2022
6. article 20 du décret présidentiel n° 22-93 du 5 Chaâbane 1443 correspondant au 8 mars 2022
7. Mounia, B. (2021) *Cour Constitutionnelle : De larges attributions Lui conférant Le rôle d'épine dorsale de l'etat de droit*, *APS*. Available at: https://www.aps.dz/algerie/130907-cour-constitutionnelle-de-larges-attributions-lui-conferant-le-role-d-epine-dorsale-de-l-etat-de-droit.
8. *Base de Données juridiques en ligne: Legal doctrine Base de données juridiques en ligne | Legal Doctrine*. Available at: https://legal-doctrine.com/edition/controle-constitutionnel-et-surveillance-des-elections-en-algerie/ .
9. Techno-Science.net *architecture logicielle - définition et explications*, *Techno*. Available at: https://www.techno-science.net/glossaire-definition/Architecture-logicielle.html.
10. Jacques PRINTZ, « **LOGICIELS** », *Encyclopædia Universalis* [en ligne], consulté le 4 mai 2023. URL : https://www.universalis.fr/encyclopedie/logiciels/3-une-breve-histoire-du-genie-logiciel/
11. IEEE, 2019 38th International Conference of the Chilean Computer Science Society (SCCC), 04 May 2018.
12. J. Lewis and M. Fowler, *Microservices. A definition of this new architectural term*, [online] Available: https://martinfowler.com/articles/microservices.html.
13. M. Fowler, *Monolithfirst*, [online] Available: https://martinfowler.com/bliki/MonolithFirst.html
14. C. Richardson, *Pattern: Monolithic architecture*, [online] Available: https://microservices.io/pattems/monolithic.html.

15. *Architecture Monolithique Guide #1 • blog osmova agence digitale Montpellier* (2021) *Blog Osmova Agence Digitale Montpellier*. Available at: https://blog.osmova.com/architecture-monolithique/ (Accessed: 01 July 2023).
16. A. Balalaie, A. Heydarnoori and P. Jamshidi, "Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture", *IEEE Software*, vol. 33, no. 3, pp. 42-52, 20
17. C. Pautasso et al., "Microservices in Practice Part 1: Reality Check and Service Design", *IEEE Software*, vol. 34, no. 1, pp. 91-98, 2017.
18. . J. Thönes, "Microservices", *IEEE Software*, vol. 32, no. 1, pp. 116,113-115, 2015.
19. IEEE, IEEE Software (Volume: 35, Issue: 3, May/June 2018).
20. *Supports de transmission - fad.umi.ac.ma*. Available at: https://fad.umi.ac.ma/pluginfile.php/178969/mod_folder/content/0/Support%20%20de%20Cours_r%C3%A9seau_Fst_Chp4.pdf.
21. *MySQL Workbench - Logiciel de Gestion de MySQL SQL*. Available at: https://sql.sh/logiciel/mysql-workbench).
22. Affirmer la sécurité des applications web | Cloudflare. https://www.cloudflare.com/fr-fr/learning/security/what-is-web-application-security/.
23. A Brief Guide to the Standard Object Modeling Language" by Martin Fowler.
24. Object-Oriented Modeling and Design with UML by James Rumbaugh, Michael. Blaha.
25. *Définition des diagrammes de séquence UML 1.5 - RAD Studio*. Available at: https://docwiki.embarcadero.com/RADStudio/Sydney/fr/D%C3%A9finition_des_diagrammes_de_s%C3%A9quence_UML_1.5.
26. *Diagrammes de communication* (no date) *IBM*. Available at: https://www.ibm.com/docs/fr/rsas/7.5.0?topic=uml-communication-diagrams (Accessed: 05 September 2023).
27. Badji, S. & Zamoum, N. (2018). Conception et réalisation d'une application web jee et mobile pour la gestion de la bibliothèque [Mémoire de Master, Université Mouloud Mammeri - Tizi Ouzou].
28. Affirmer la sécurité des applications web | Cloudflare. https://www.cloudflare.com/fr-fr/learning/security/what-is-web-application-security/.
29. *Elodie : Introduction à angular pour débutants: Apprendre Les Fondamentaux du développement web avec le framework angular de google.* (no date) *Superprof*. Available at: https://www.superprof.lu/introduction-angular-debutants-apprendre-fondamentaux-developpement-web-framework-angular-google.html (Accessed: 01 July 2023).

30. Introduction au dossier : Sécurité des applications web. https://connect.ed-diamond.com/MISC/misc-101/introduction-au-dossier-securite-des-applications-web.
31. Comment apprendre la sécurité des applications Web? - Geekflare. https://geekflare.com/fr/learn-web-application-security/.
32. Sullivan, B. and Liu, V. (2012) *Web application security: A beginner's guide*. New York: McGraw-Hill.
33. La Rédaction JDN (2023) *Postman : Comment Utiliser La plateforme d'api no code, E*. Available at: https://www.journaldunet.fr/web-tech/guide-de-l-entreprise-digitale/1511313-postman-comment-utiliser-la-plateforme-d-api-no-code/.
34. *UML 2 tutoriel - diagrammes de classe* (no date) *Diagramme de Classe - UML 2 Tutoriel | Sparx Systems*. Available at: https://www.sparxsystems.fr/resources/tutorials/uml2/class-diagram.html.