Encadrant FSB

J'autorise l'étudiant à faire le dépôt initial du mémoire ou de l'essai aux fins d'évaluation.

Signature

Encadrant Entreprise

J'autorise l'étudiant à faire le dépôt initial du mémoire ou de l'essai aux fins d'évaluation.

Signature et cachet

Dédicaces

Remerciements

Table des matières

Table des figures

Liste des tableaux

Introduction générale

Chapitre 1

Présentation du cadre du projet

Plan

Introduction

En premier lieu nous commençons par mettre le projet dans son cadre et son jargon en expliquant certaines notions. Tout d'abord, nous présentons l'organisme d'accueil. Ensuite, nous mettons notre projet intitulé "Conception et développement d'une plateforme d'Email Center" dans son contexte. Puis, nous enchaînons par la problématique générale. Et nous finissons par définir la méthodologie de travail adoptée.

1.1 Présentation de l'entreprise d'accueil

Dans cette partie, nous présentons l'organisme d'accueil, ses domaines d'expertise afin de donner une idée sur l'environnement professionnel dans lequel nous avons effectué notre projet de fin d'études.

1.1.1 Présentation générale

Progress Engineering est une entreprise d'ingénierie informatique, spécialisée dans la conception, le développement et l'intégration de solutions clés en main adaptées à des besoins spécifiques. Fondée en 2000 par une équipe d'ingénieurs hautement qualifiés travaillant dans le domaine du développement de logiciels et les télécommunications.

Cette société offre une variété de solutions de développement et de conseils en se basant sur l'e-service. La société a acquit un savoir faire, des compétences et de l'expertise lui permettant de mener à bien tout projet informatique avec tout ses acpects (spécification, conception, développement, test, validation, déploiement, assistance et maintenance).

Grâce à sa politique de croissance étudiée, Progress Engineering a su maintenir sa position d'acteur majeur dans le secteur de SSII (Société de Service en Ingénierie Informatique) en Tunisie.

1.2 Présentation du sujet

Dans cette partie, nous allons présenter une problématique puis nous allons proposer notre solution spécifiant ses apports techniques et fonctionnels.

1.2.1 Problématique

Chaque entreprise ou organisation possède plusieurs comptes email utilisés par son personnel. Les messages véhiculés par ces comptes email représentent une partie importante des interactions de l'organisation avec son milieu extérieur.

Ces messages renferment des informations importantes échangées avec les partenaires, les fournisseurs, les clients et les tiers d'une façon générale.

Souvent, les informations échangées dans ces emails ont une valeur contractuelle vis à vis des tiers. Il est important pour une société ou une organisation d'en garder trace.

1.2.2 Solution envisagée

Progress Engineering l'entreprise où se déroule notre stage de fin d'études, a proposé une solution afin de mieux répondre aux besoins des entreprises.

L'entreprise a décidé de mettre en place une plateforme Web administrable en ligne qui permet de définir l'ensemble des comptes emails à scanner périodiquement et de récupérer les emails échangés.

1.3 Contexte du projet

Ce projet porte sur l'étude, la conception et la réalisation d'une plateforme Web qui permettra de collecter, d'analyser, de stocker et d'archiver efficacement les emails échangés au sein de l'organisation.

Conclusion

Ce chapitre nous a permis de mettre le projet dans son cadre général et de le présenter. Dans le chapitre suivant nous présentons la partie étude de l'existant qui permet d'analyser des projets similaires existants.

Chapitre 2

Spécification des besoins

Plan

Introduction

Dans ce chapitre, nous présentons les besoins fonctionnels aussi bien que les besoins non fonctionnels de notre projet. Aussi nous présentons les acteurs et leurs rôles dans notre application.

2.1 Le langage UML

« UML (en anglais Unified Modeling Language ou « langage de modélisation unifié ») est un langage de modélisation graphique à base de pictogrammes. Il est apparu dans le monde du génie logiciel, dans le cadre de la « conception orientée objet ». Couramment utilisé dans les projets logiciels, il peut être appliqué à toutes sortes de systèmes ne se limitant pas au domaine informatique [1].

2.2 Les besoins fonctionnels

Notre projet est une application WEB, le logiciel doit être propriétaire et payant.
Notre projet définie par (voir figure 2.1) :

- L'application est composée de deux parties :
  - Une partie back office qui regroupe un ensemble des gestions administratives.
  - Une partie Front office qui désigne la partie visible par les utilisateurs du site.

Figure 2.1 : La composition de notre projet

2.2.1 Les acteurs

- Administrateur :
  - **Description :** Responsable de la gestion et de la configuration de la plateforme.
  - **Actions :**
    - o Définition des comptes emails à scanner périodiquement.
    - o Gestion des utilisateurs et des paramètres globaux du service, tels que la fréquence de scan et le nombre de processus par boîte.
    - o Gestion des droits d'accès des utilisateurs.
  - **Responsabilités :** Assurer le bon fonctionnement et la sécurité de la plateforme, ainsi que la configuration des paramètres globaux.
- Utilisateurs de l'organisation :
  - **Description :** Employés de l'organisation qui utilisent la plateforme pour accéder aux emails et aux contacts.
  - **Actions :**
    - o Recherche dans les emails et les contacts collectés.
    - o Spécification des comptes emails à surveiller et des règles de collecte et d'archivage pour leurs propres besoins.
    - o Consultation des emails archivés.
    - o Intéraction avec les fonctionnalités de la plateforme selon leurs droits d'accès.
  - **Responsabilités :** Utiliser la plateforme pour faciliter la communication et la gestion des emails, ainsi que la configuration des paramètres spécifiques à leurs besoins.

2.2.2 Les besoins fonctionnels du site

- **Collecte automatisée des emails provenant de comptes spécifiques :**
  - Mise en place d'un mécanisme automatisé permettant de récupérer périodiquement les emails provenant des comptes désignés par l'administrateur de la plateforme.
  - Configuration des paramètres de collecte, tels que la fréquence de récupération des emails et les comptes à scanner.
- **Analyse du contenu des emails pour identifier les contacts mentionnés :**
  - Développement d'un module d'analyse sémantique pour extraire les informations pertinentes des emails, notamment l'identification des contacts mentionnés dans les messages.

  - Utilisation d'algorithmes de traitement du langage naturel (NLP) pour comprendre le contexte et extraire les données utiles.

- **Classification automatique des emails en fonction des contacts identifiés :**
  - Création d'un système de classification automatique des emails, basé sur les contacts identifiés lors de l'analyse du contenu.
  - Attribution de catégories ou de tags aux emails en fonction des contacts impliqués, facilitant ainsi leur gestion et leur recherche ultérieure.
- **Configurer les règles de scan, de classification et d'archivage :**
  - Développement d'une interface d'administration conviviale permettant à l'utilisateur de configurer les règles de collecte, d'analyse, de classification et d'archivage des emails.
  - Paramétrage des règles de manière flexible, avec la possibilité d'ajuster les critères de classification et les périodes de scan en fonction des besoins de l'organisation.
- **Règles et périodes de scan des emails :**
  - **Définition des comptes emails à scanner :** L'administrateur peut spécifier les comptes emails à scanner, en fournissant les adresses emails et les identifiants associés.
  - **Période de scan :** L'utilisateur peut définir la fréquence à laquelle les comptes emails seront scannés pour collecter de nouveaux messages. Par exemple, les scans peuvent être programmés toutes les heures, tous les jours ou selon une autre périodicité définie.
  - **Critères de sélection des emails :** L'utilisateur peut définir des critères pour sélectionner les emails à collecter, tels que les expéditeurs, les destinataires, les mots-clés, les pièces jointes, etc.
- **Règles de classification des messages :**
  - **Classification automatique :** L'utilisateur peut définir des règles de classification automatique pour catégoriser les messages en fonction de différents critères, tels que l'expéditeur, le destinataire, le sujet, les mots-clés, etc.
  - **Attribution de tags ou de labels :** En fonction des règles définies, les messages seront automatiquement étiquetés ou classés dans des dossiers spécifiques pour une organisation efficace et une récupération ultérieure facilitée.
- **Règles d'archivage des messages :**

- **Durée de conservation :** L'utilisateur peut définir la durée pendant laquelle les messages seront conservés dans la plateforme avant d'être archivés ou supprimés.
- **Politiques d'archivage :** Des politiques d'archivage peuvent être établies pour déterminer comment les messages seront archivés, par exemple en les stockant dans des archives spécifiques ou en les exportant vers un système d'archivage externe.

- **Recherche et l'accès aux emails et contacts collectés :**
  - Conception et développement d'une interface utilisateur conviviale permettant aux utilisateurs de rechercher et d'accéder facilement aux emails et contacts collectés.
  - Mise en place de filtres de recherche avancés pour permettre une recherche efficace en fonction de différents critères tels que l'expéditeur, le destinataire, la date, etc.
- **Gestion des droits d'accès pour assurer la confidentialité et la sécurité des données :**
  - Implémentation d'un système de gestion des droits d'accès robuste pour garantir que seules les personnes autorisées puissent accéder aux emails et aux données de contact.
  - Attribution de niveaux d'autorisation différents en fonction des rôles des utilisateurs, avec la possibilité de définir des permissions spécifiques pour chaque utilisateur ou groupe d'utilisateurs.

2.3 Les besoins non fonctionnels

Les besoins non fonctionnels sont importants car ils agissent de façon indirecte sur le résultat et sur le rendement de l'utilisateur, ce qui fait qu'ils ne doivent pas être négligés, pour cela il faut répondre aux exigences suivantes :

- **Sécurité :**
  - **Confidentialité des données :** La plateforme doit mettre en œuvre des mécanismes de chiffrement robustes pour garantir que les données échangées via les emails sont protégées contre tout accès non autorisé.
  - **Intégrité des données :** Des mesures de sécurité doivent être mises en place pour s'assurer que les données des emails ne sont ni altérées ni modifiées lors de leur collecte, de leur analyse ou de leur stockage.
  - **Contrôle d'accès :** Un système de contrôle d'accès granulaire doit être implémenté pour garantir que seules les personnes autorisées ont accès aux fonctionnalités et aux

données appropriées de la plateforme.

- **Scalabilité :**
  - **Extensibilité :** La plateforme doit être conçue pour être extensible, permettant ainsi d'ajouter de nouvelles fonctionnalités et de supporter une augmentation du nombre d'utilisateurs et de messages emails sans compromettre les performances.
  - **Élasticité :** Des mécanismes d'élasticité doivent être mis en place pour permettre à la plateforme de s'adapter dynamiquement à des charges de travail variables, en allouant et en libérant automatiquement les ressources en fonction des besoins.
- **Compatibilité :**
  - **Interopérabilité :** La plateforme doit être compatible avec les systèmes et les technologies existants de l'organisation, permettant ainsi une intégration fluide avec les infrastructures informatiques existantes.
  - **Normes et protocoles :** Elle doit supporter les normes et les protocoles de l'industrie couramment utilisés, garantissant ainsi une compatibilité avec les différentes applications et plateformes utilisées au sein de l'organisation.

2.4 Les diagrammes de cas d'utilisation du projet

Le diagramme des cas d'utilisation est utilisé pour donner une vision globale du comportement fonctionnel du système. Dans cette section, nous présentons le diagramme de cas d'utilisation général du plateforme d'emails.

2.4.1 Le diagramme de cas d'utilisation général du plateforme d'emails

L'administrateur et les utilisateurs de ce plateforme doivent se connecter chacun à son compte pour prendre en charge la gestion du plateforme afin de le tenir à jour.

2.4.2 Diagramme de cas d'utilisation détaillé «Gérer comptes email»

Ce cas d'utilisation est destiné aux utilisateurs du rôle "Administrateur ou Utilisateur". C'est la généralisation des trois sous cas à savoir ajouter un nouveau compte, consulter, modifier et supprimer un compte. Le diagramme 2.3 dans la page 12 détaille ce cas d'utilisation.

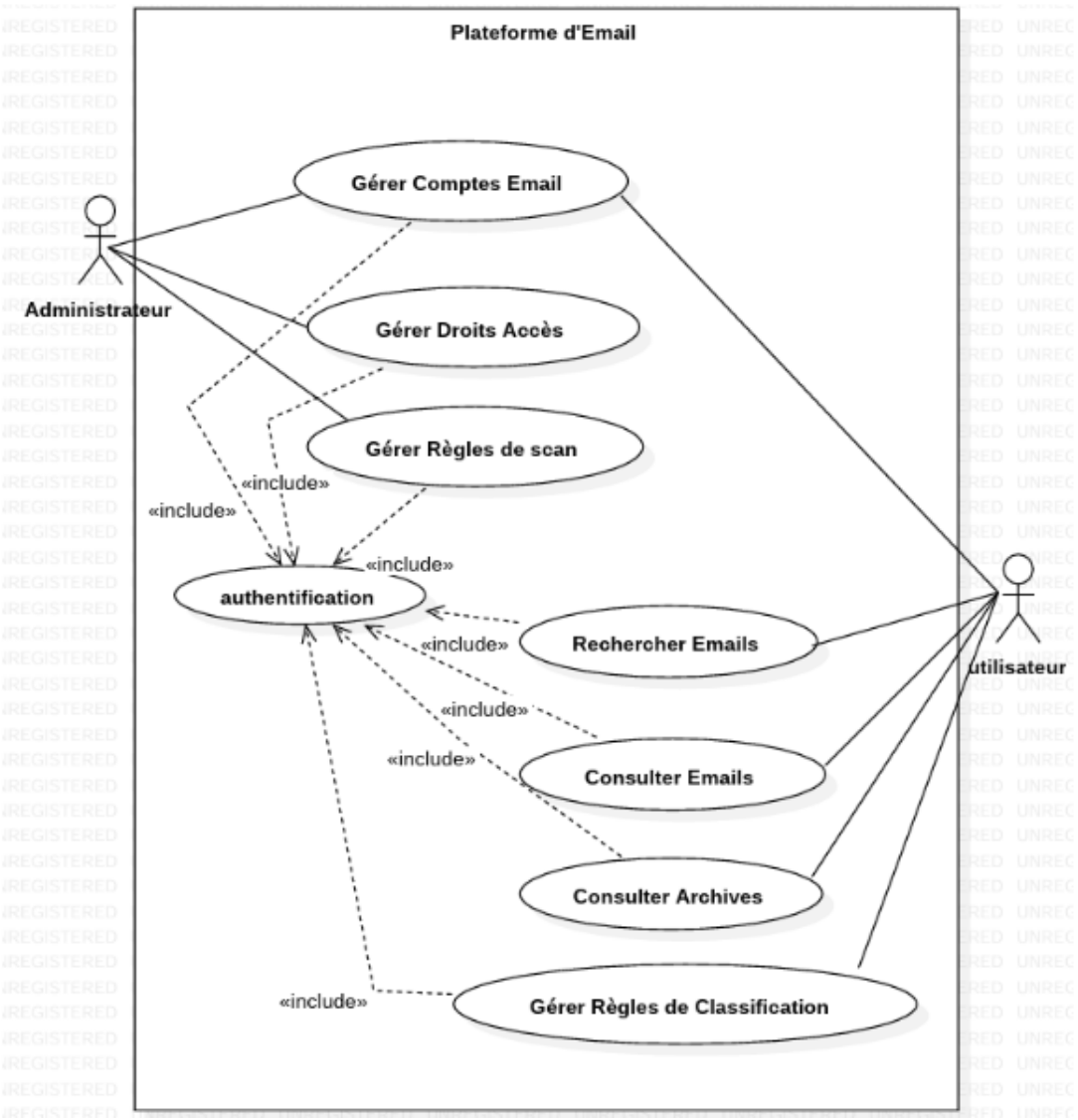

Figure 2.2 : Diagramme de cas d'utilisation général du plateforme d'emails

2.4.3 Diagramme de cas d'utilisation détaillé «Gérer droits d'accès»

Ce cas d'utilisation est destiné aux utilisateurs du rôle "Administrateur". L'objectif principal de la gestion des droits d'accès est d'ajouter des utilisateurs, consulter, modifier et supprimer un utilisateur. Le diagramme 2.4 dans la page 12 détaille ce cas d'utilisation.

2.4.4 Diagramme de cas d'utilisation détaillé «Gérer règles de scan»

Ce cas d'utilisation est destiné aux utilisateurs du rôle "Administrateur". L'objectif principal de la gestion des règles de scan est de définr les règles de scan par définir la période de scan et définir les comptes emails à scanner. Le diagramme 2.5 détaille ce cas d'utilisation.

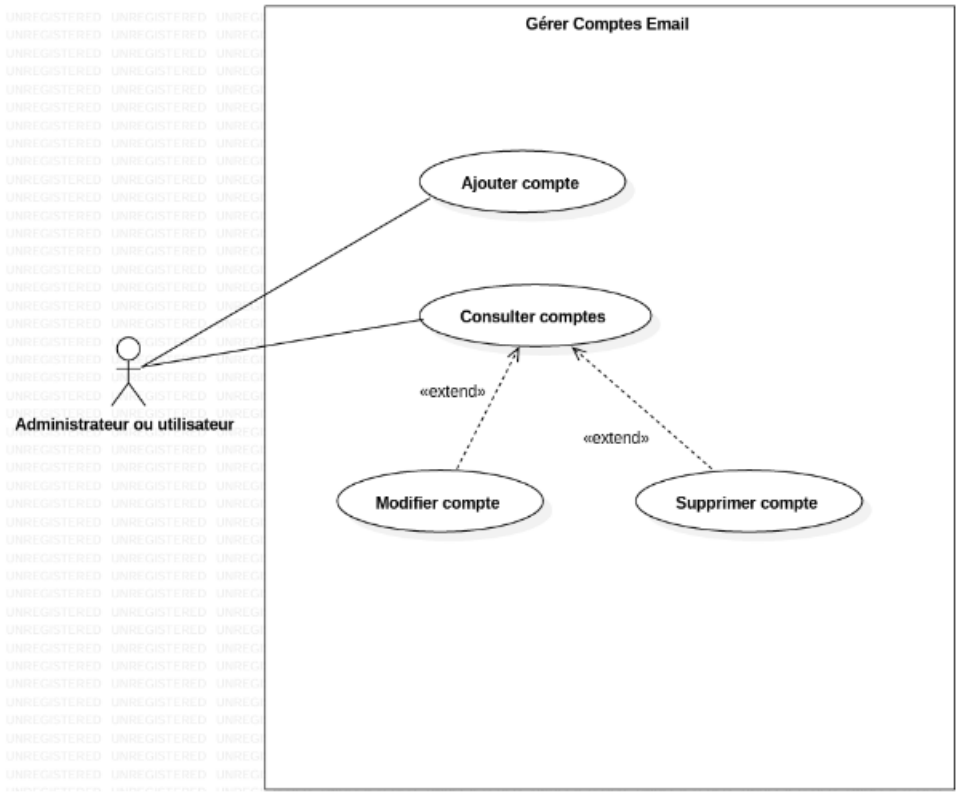

Figure 2.3 : Diagramme de cas d'utilisation détaillé « Gérer comptes email »

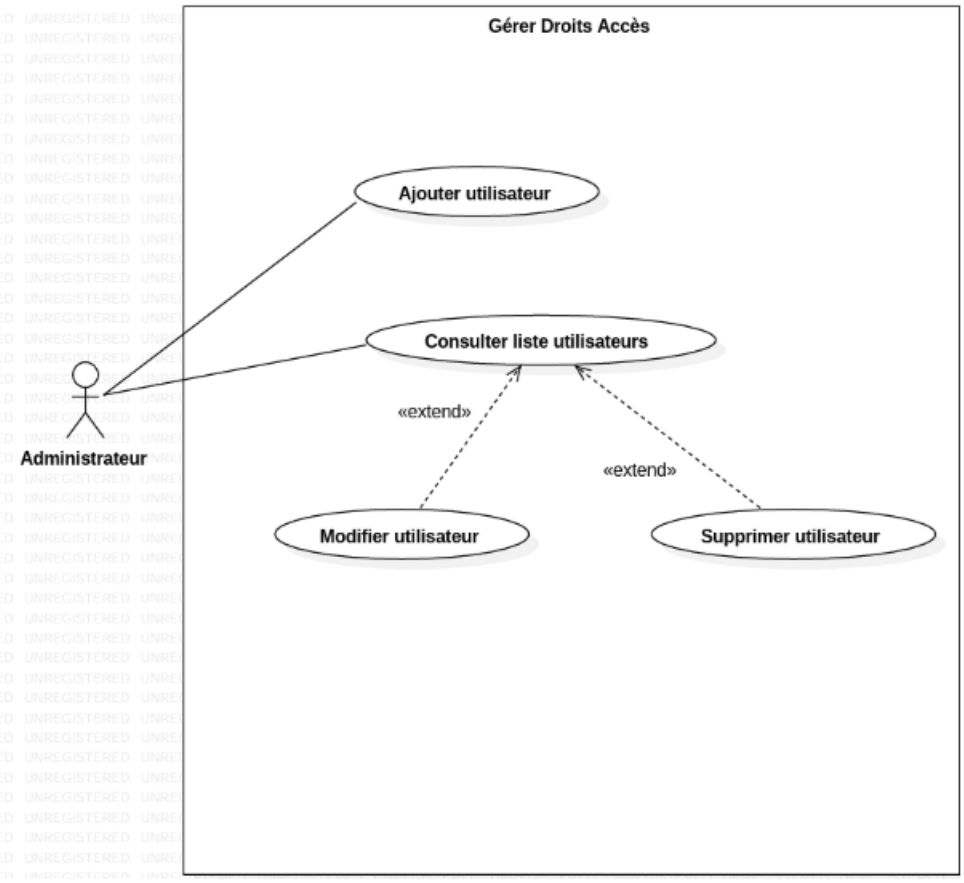

Figure 2.4 : Diagramme de cas d'utilisation détaillé « Gérer droits d'accès »

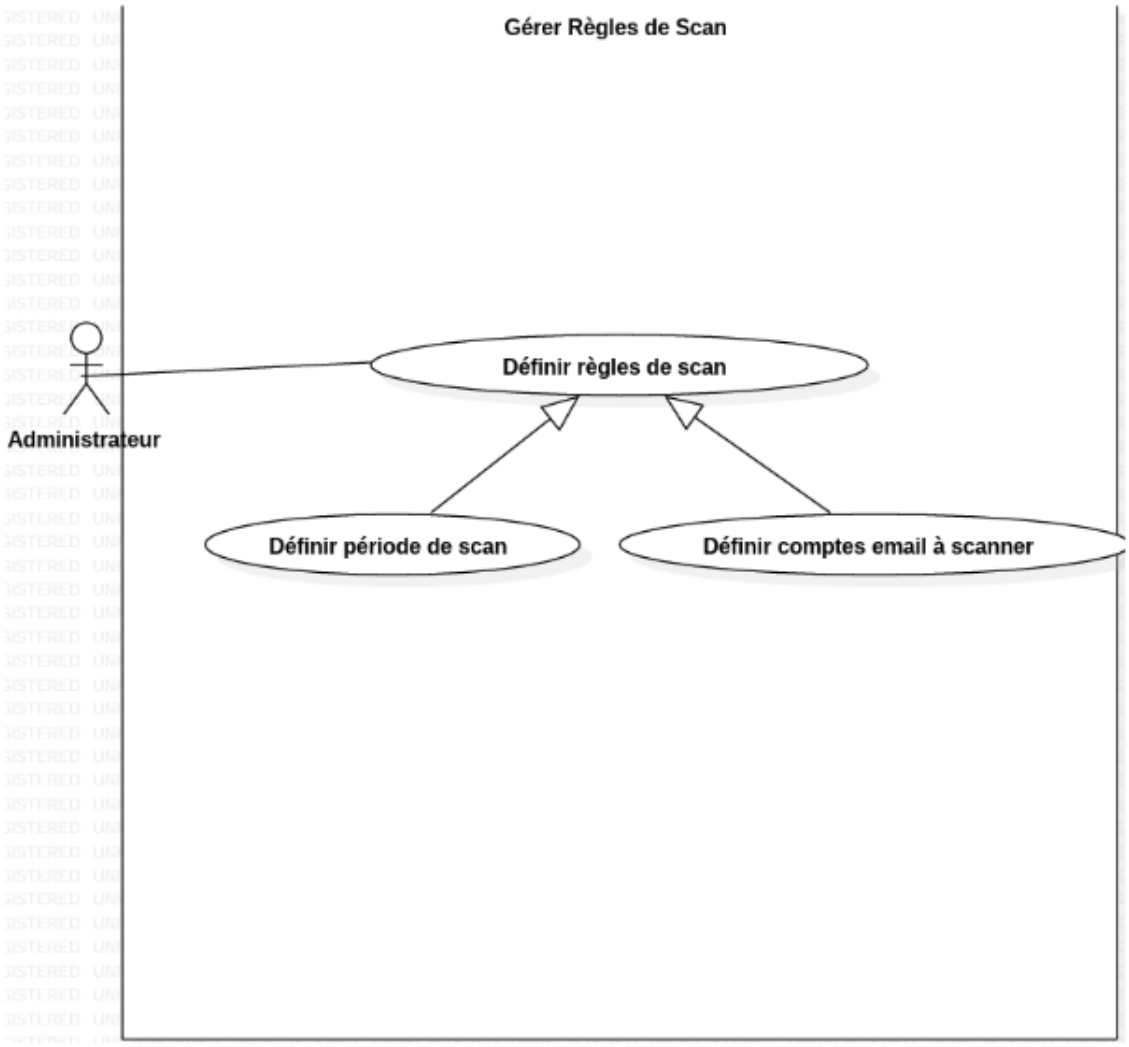

Figure 2.5 : Diagramme de cas d'utilisation détaillé « Gérer règles de scan »

2.4.5 Diagramme de cas d'utilisation détaillé «Gérer règles de classification»

Ce cas d'utilisation est destiné aux utilisateurs du rôle "Utilisateur". L'objectif principal de la gestion des règles de classification est de créer, consulter, modifier et supprimer les règles de classification. Le diagramme 2.7 dans la page 16 détaille ce cas d'utilisation.

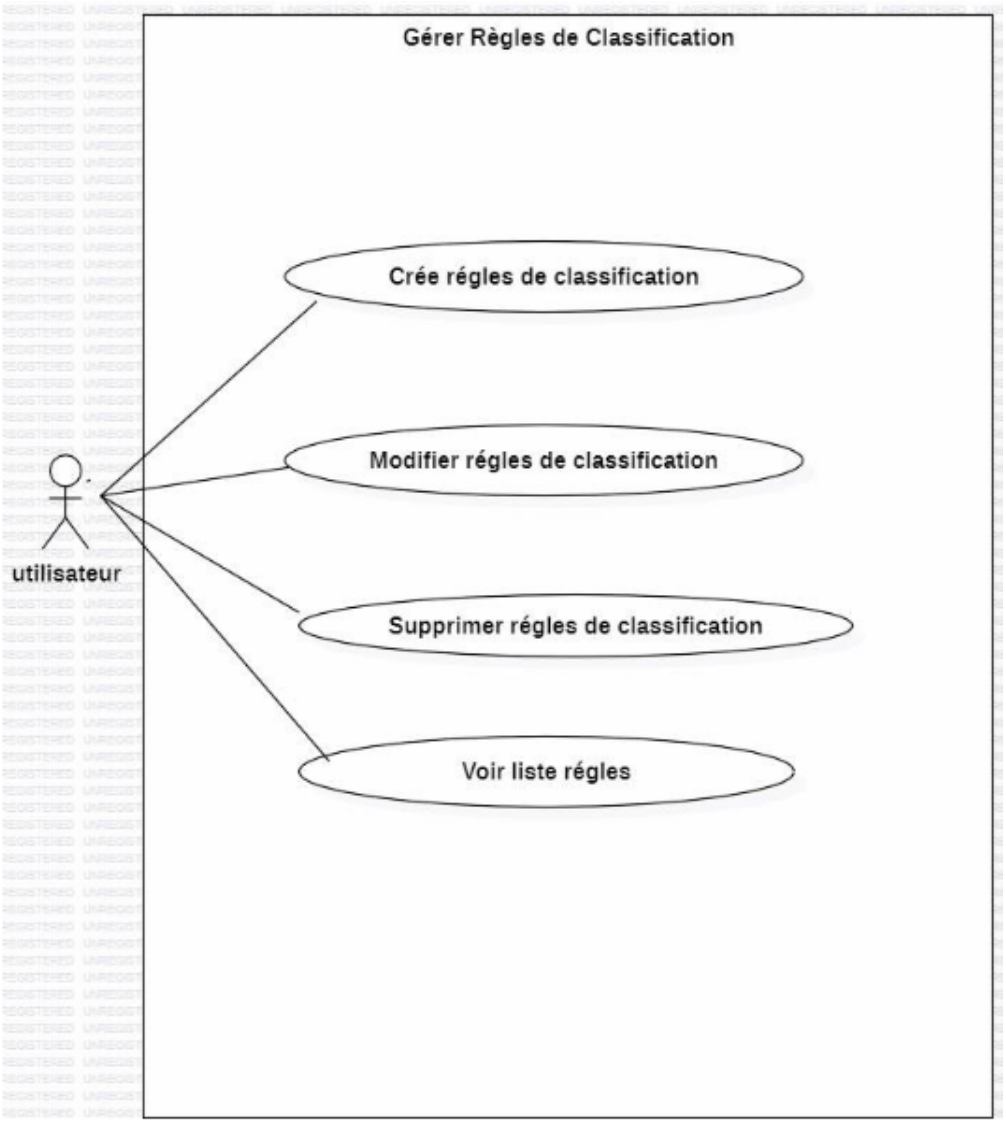

Figure 2.6 : Diagramme de cas d'utilisation détaillé « Gérer règles de classification »

2.5 Les diagrammes de séquence système du projet

Dans cette section, nous présentons les diagrammes de séquence système du plateforme emails.

2.5.1 Le diagramme de séquence de cas d'utilisation «Ajouter compte email»

Dans le tableau 2.1 de la page 15, nous présentons la description textuelle du cas d'utilisation « Ajouter compte email »

Cas d'utilisation	Ajouter compte email
Acteur	Administrateur ou Utilisateur
But	Ajouter les comptes emails qui seront scannés périodiquement
Pré-condition	L'utilisateur autorisé doit être authentifié dans le système
Post-condition	Le compte email est ajouté dans le système et prêt à être utilisé pour la collecte des emails
Scénario principal	1-L'utilisateur autorisé accède à la fonctionnalité de gestion des comptes emails. 2-Le système affiche la liste des comptes emails déjà configurés, s'il y en a. 3-L'utilisateur autorisé choisit l'option de créer un nouveau compte email. 4-Le système affiche un formulaire de création de compte email. 5-L'utilisateur autorisé saisit les informations du compte email à ajouter. 6-L'utilisateur autorisé confirme la création du nouveau compte email. 7-Le système vérifie les informations saisies et enregistre le nouveau compte email. 8-Si les champs sont valides : 8.1-Le système enregistre le nouveau compte email et affiche un message de confirmation d'ajout. 9-Si non le système affiche un message d'erreur.

Tableau 2.1 : La description textuelle de cas d'utilisation « Ajouter compte email »

Le diagramme de séquence système associé au scénario définit dans le tableau 2.1 est représenté dans la figure 2.7 de la page 16.

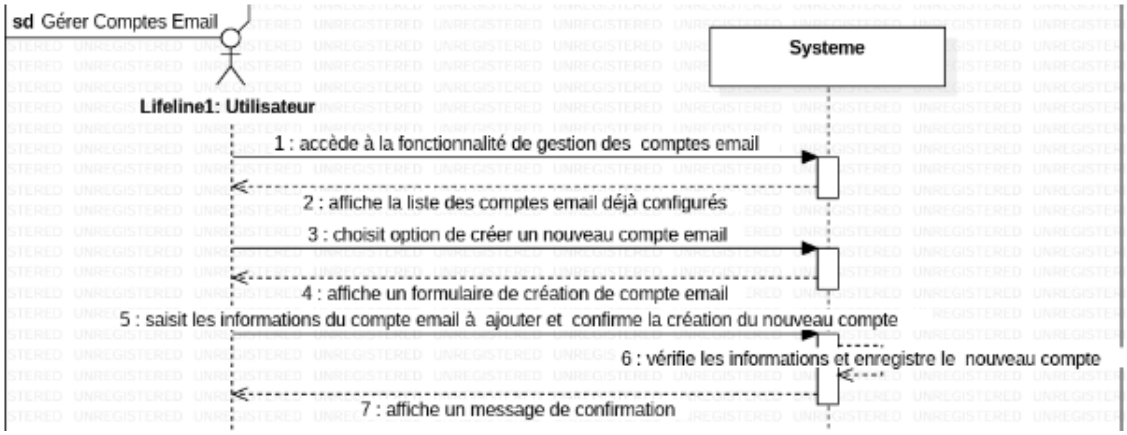

Figure 2.7 : Diagramme de séquences système « Ajouter compte email »

2.5.2 Le diagramme de séquence de cas d'utilisation «Modifier droits d'accès»

Dans le tableau 2.2 de la page 17, nous présentons la description textuelle du cas d'utilisation « Modifier droits d'accès »

Cas d'utilisation	Modifier droits d'accès
Acteur	Administrateur
But	Définir les droits d'accès d'un utilisateur
Pré-condition	L'administrateur doit être authentifié et avoir les autorisations nécessaires pour accéder à la fonctionnalité de gestion des droits d'accès
Post-condition	Les droits d'accès des utilisateurs sont mis à jour conformément aux modifications apportées par l'administrateur
Scénario principal	1-L'administrateur accède à l'interface de gestion des droits d'accès. 2-L'administrateur sélectionne l'utilisateur dont il souhaite modifier les droits d'accès. 3-L'administrateur modifie les droits d'accès de l'utilisateur en fonction des besoins. 4-L'administrateur enregistre les modifications apportées aux droits d'accès. 5-Si les informations sont valides : 5.1-Le système met à jour les droits d'accès de l'utilisateur dans la base de données. 6-Si non le système affiche un message d'erreur.

Tableau 2.2 : La description textuelle de cas d'utilisation « Modifier droits d'accès »

Le diagramme de séquence système associé au scénario définit dans le tableau 2.2 est représenté dans la figure 2.8 de la page 18.

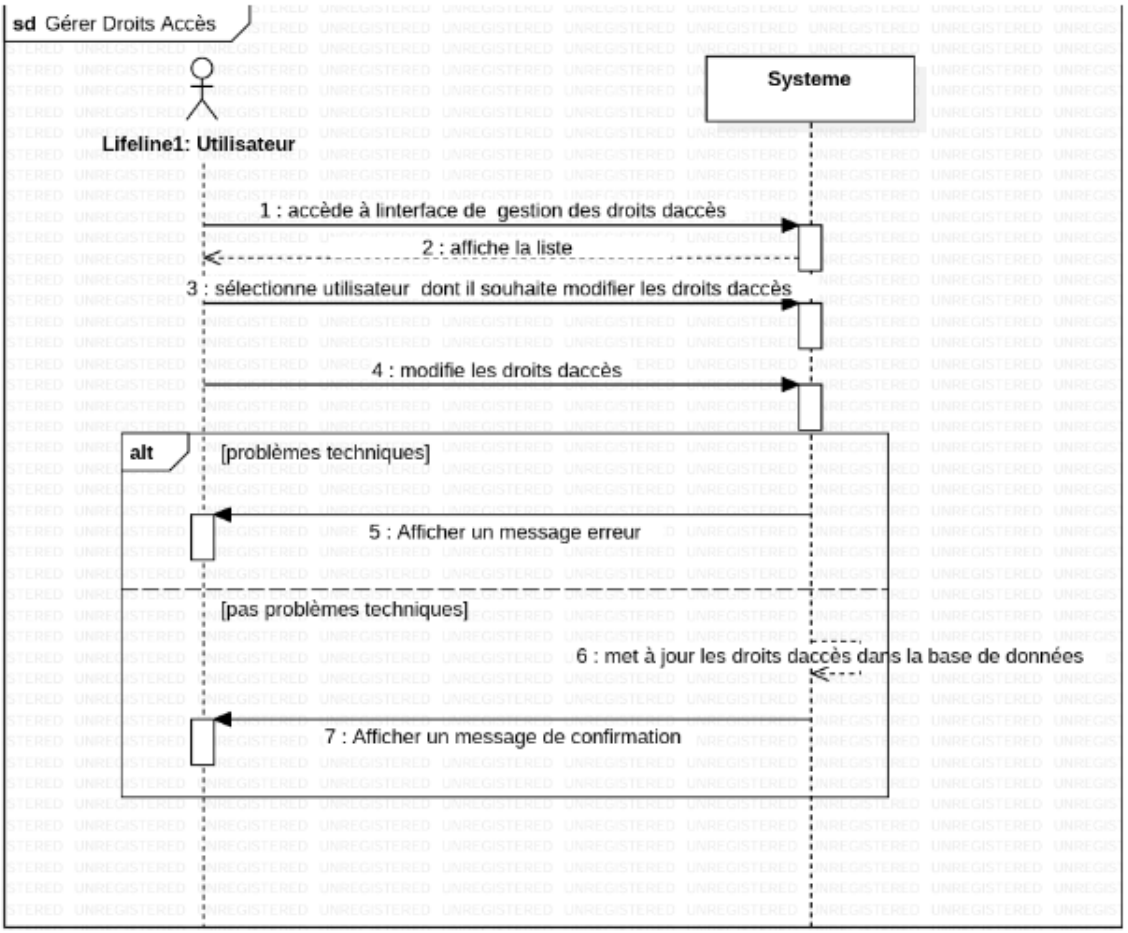

Figure 2.8 : Diagramme de séquences système « Modifier droits d'accès »

2.5.3 Le diagramme de séquence de cas d'utilisation «Ajouter règle de scan»

Dans le tableau 2.4 de la page 20, nous présentons la description textuelle du cas d'utilisation « Ajouter règle de scan »

Cas d'utilisation	Ajouter règle de scan
Acteur	Administrateur
But	Définir les règles de scan des emails
Pré-condition	L'administrateur est authentifié dans le système
Post-condition	Les règles de scan des emails sont créées
Scénario principal	1-L'administrateur accède à la section de gestion des règles de scan dans l'interface d'administration. 2-L'administrateur choisit l'option de création d'une nouvelle règle de scan. 3-Le système affiche un formulaire permettant à l'administrateur de spécifier les critères de la nouvelle règle de scan, tels que les comptes email à surveiller, les critères de sélection des emails, la périodicité du scan. 4-L'administrateur remplit le formulaire avec les informations nécessaires. 5-Si les champs sont valides : 5.1-Le système enregistre la nouvelle règle de scan dans la base de données. 6-Si non le système affiche un message d'erreur.

Tableau 2.3 : La description textuelle de cas d'utilisation « Ajouter règle de scan »

Le diagramme de séquence système associé au scénario définit dans le tableau 2.4 est représenté dans la figure 2.10 de la page 21.

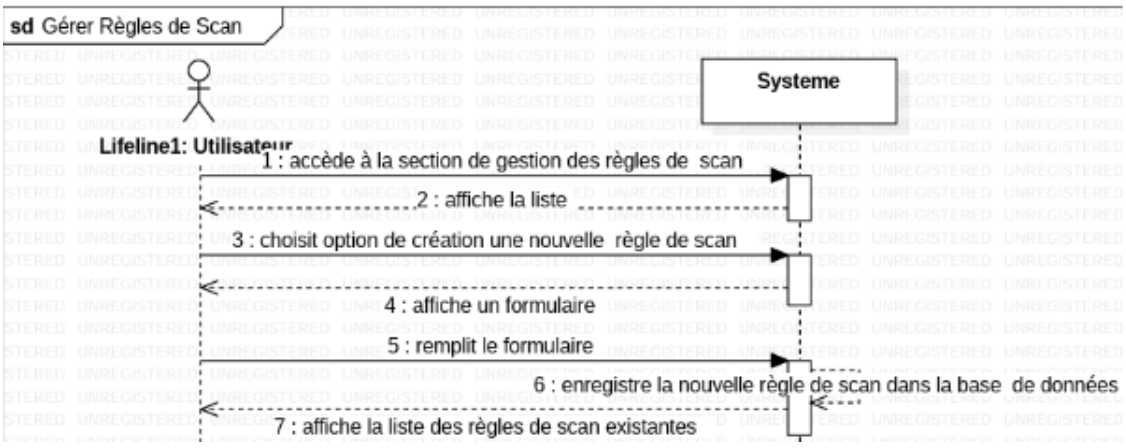

Figure 2.9 : Diagramme de séquences système « Ajouter règle de scan »

2.5.4 Le diagramme de séquence de cas d'utilisation «Créer règle de classification»

Dans le tableau 2.4 de la page 20, nous présentons la description textuelle du cas d'utilisation « Créer règle de classification »

Cas d'utilisation	Créer règle de classification
Acteur	Utilisateur
But	Créer des règles de classification
Pré-condition	L'utilisateur dispose des droits nécessaires pour administrer les règles de classification
Post-condition	Les règles de classification des emails sont créées
Scénario principal	1-L'utilisateur accède à la section de gestion des règles de classification dans l'interface de la plateforme d'Email Center. 2-L'utilisateur choisit l'option de création d'une nouvelle règle de classification. 3-Le système affiche un formulaire permettant à l'utilisateur de spécifier les critères de la nouvelle règle de classification, tels que les expéditeurs, les destinataires, les mots-clés, etc. 4-L'utilisateur remplit le formulaire avec les informations requises. 5-Si les champs sont valides : 5.1-Le système enregistre la nouvelle règle de classification dans la base de données. 6-Si non le système affiche un message d'erreur.

Tableau 2.4 : La description textuelle de cas d'utilisation « Créer règle de classification »

Le diagramme de séquence système associé au scénario définit dans le tableau 2.4 est représenté dans la figure 2.10 de la page 21.

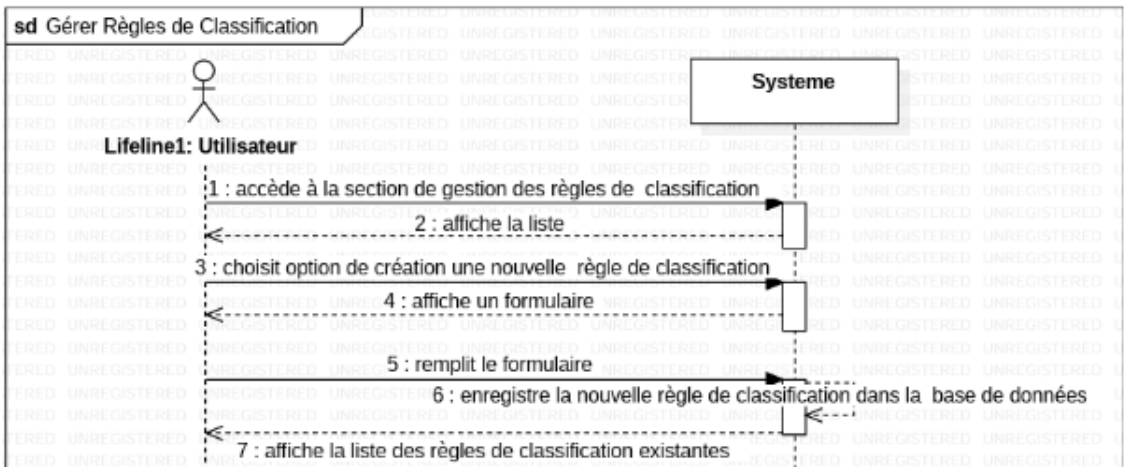

Figure 2.10 : Diagramme de séquences système « Créer règle de classification »

2.6 Le diagramme de classe d'analyse du projet

Ce diagramme de classe est utilisé pour dévoiler la structure interne du système. Il s'agit d'une description de la structure des objets et des informations utilisées par l'application à implémenter et ceci sans tenir compte de l'aspect temporel dans le comportement du système [2]. Le diagramme de classes est considéré comme étant le diagramme le plus important et le seul obligatoire lors d'une modélisation orientée objet. La figure 2.11 dans la page 22 illustre le diagramme de classes de notre projet.

Conclusion

Au cours de ce chapitre, nous avons présenté une étude du projet à réaliser en citant les besoins fonctionnels et non fonctionnels ainsi que les diagrammes d'analyse. Dans le chapitre suivant nous allons aborder l'étude conceptuelle de notre projet.

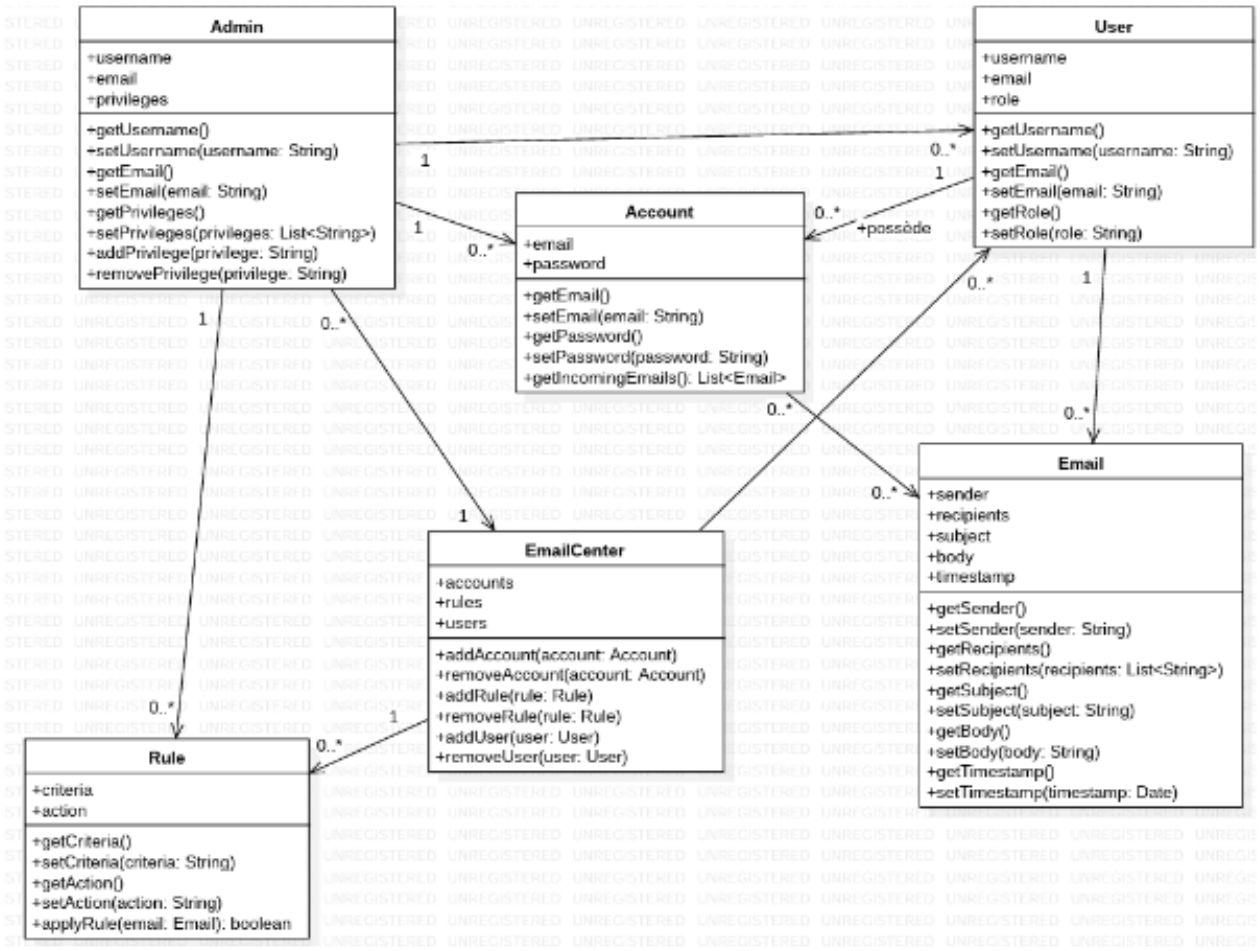

Figure 2.11 : Diagramme de classe d'analyse du projet

Chapitre 3

Conception et Sécurité

Introduction

Nous nous intéressons dans le cadre de ce chapitre, à l'architecture de notre projet et à la conception des cas d'utilisation que nous venons d'analyser dans le chapitre précédent. Nous débutons ce chapitre par la présentation de l'architecture opérationnelle et logicielle de notre projet suivi des diagrammes de conception, l'architecture du web services de notre projet et la sécurité.

3.1 L'architecture opérationnelle

Notre projet est basé sur l'architecture 3-tiers. Cette architecture avec ces trois niveaux ou bien trois couches, facilite le développement et la maintenance du projet.

- **La couche présentation** est associée aux utilisateurs pour interagir avec le système. C'est la partie interactive et visible de l'application.
- **La couche fonctionnelle** (ou bien couche métier) joue le rôle d'intermédiaire entre la présentation et les données. Elle interagit avec la couche d'accès aux données pour accéder à la base de données, et communique avec la couche de présentation pour traiter les requêtes des utilisateurs.
- **La couche de données** englobe toutes les interactions effectuées entre la base de données et l'application. Elle est chargée de l'accès aux données et de leur manipulation, indépendamment du SGBD déployé.
- Une couche "log" est ajoutée à notre architecture pour gérer l'historique des événements. Cette couche est utilisée par les trois couches présentation, logique métier et accès aux données.

  La figure 3.1 dans la page 25 représente un schéma descriptif du modèle 3-tiers.

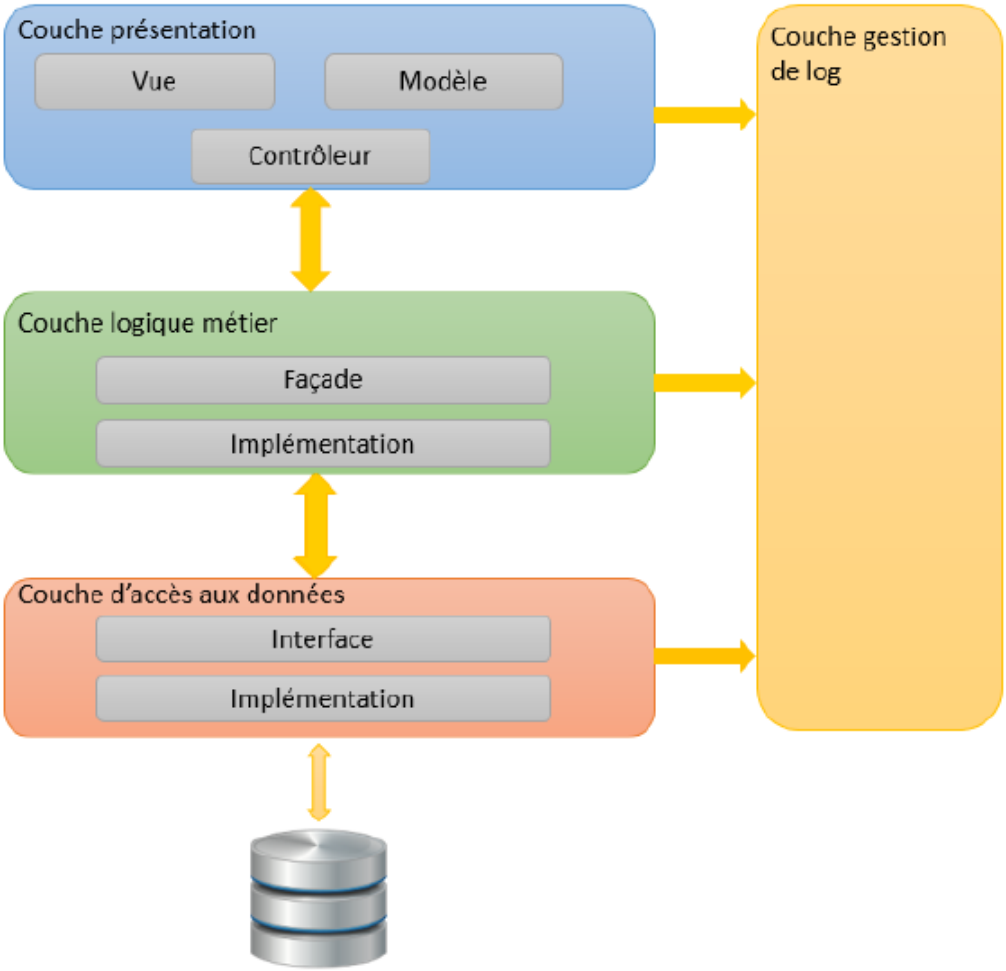

Figure 3.1 : Architecture opérationnelle de la solution

3.2 L'architecture logicielle

Dans cette partie nous présentons l'architecture MVC de l'application web.

L'architecture MVC

Nous avons développé notre application web en se basant sur le framework Java Spring Boot, qui se base à son tour sur l'architecture MVC 5 (Model-View-Controller).
Cette architecture MVC est un modèle qui a l'avantage de séparer les données, les traitements et la présentation.

- **Le modèle :** contient toutes les données qui sont soit extraites à partir de la vue, soit à partir de la base de données. Il se modifie sur ordre du contrôleur.
- **Le contrôleur :** est chargé de la synchronisation du modèle et de la vue. Il reçoit les actions de l'utilisateur de la vue puis il utilise les données du modèle, les traiter en fonction de l'action de l'utilisateur, et les envoyer à la vue afin qu'elle les affiche. En cas de besoin,

il ordonne les éventuelles modifications au modèle.

- **La vue :** représente l'interface avec l'utilisateur. Son rôle consiste à afficher les données reçues auprès du contrôleur.

  La figure 3.2 représente un schéma descriptif du modèle MVC.

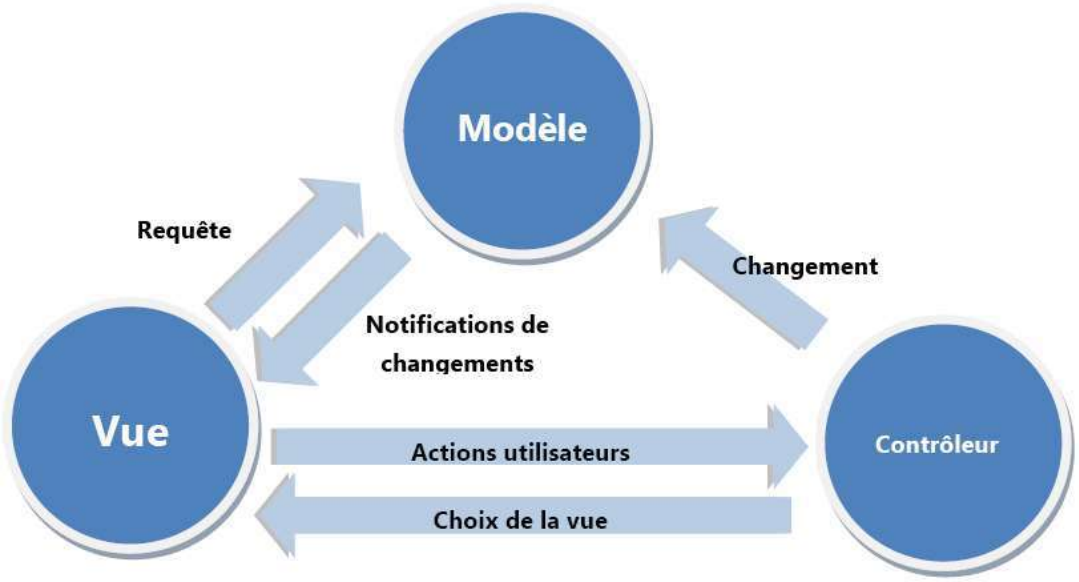

Figure 3.2 : Architecture logicielle du site « MVC »

3.3 Les diagrammes de communication

Dans cette section, nous présentons le diagramme de communication de l'application web.

3.4 Les diagrammes de classe de conception

Dans cette section, nous présentons les diagrammes de classe de conception du site.

3.5 Jeton Web JSON (JSON Web Token (JWT))

Pour assurer la sécurité de notre projet, nous avons utiliser Jeton Web JSON (JWT) pour l'authentification et l'échange d'informations des web services.

Jeton Web JSON (JWT) est un standard ouvert (**RFC 7519**) qui définit une manière compacte et autonome de transmettre de manière sécurisée des informations entre des parties en tant qu'objet JSON. Cette information peut être vérifiée et fiable car elle est signée numériquement. Les JWT peuvent être signés en utilisant un secret (avec l'algorithme **HMAC**) ou une paire de clés publique / privée utilisant **RSA** [3].

Bien que les fichiers JWT puissent être cryptés pour assurer également le secret entre les parties, nous nous concentrerons sur les jetons signés. Les jetons signés peuvent vérifier l'intégrité des réclamations qui y sont contenues, tandis que les jetons chiffrés cachent ces réclamations à d'autres parties. Lorsque des jetons sont signés à l'aide de paires de clés publiques / privées, la signature certifie également que seule la partie qui détient la clé privée est celle qui l'a signée.

3.5.1 Scénarios des JWT

- **Authentification :** Une fois l'utilisateur connecté, chaque requête ultérieure inclura le JWT, permettant à l'utilisateur d'accéder aux routes, services et ressources autorisés avec ce jeton. "Single Sign On" est une fonctionnalité qui utilise largement JWT de nos jours, en raison de sa faible surcharge et de sa capacité à être facilement utilisé dans différents domaines.
- **Web services** Les Jetons Web JSON sont un bon moyen de transmettre des informations en toute sécurité entre les parties. Étant donné que les JWT peuvent être signés (par exemple, en utilisant des paires de clés publiques / privées), nous pouvons être sûr que les expéditeurs sont ceux qu'ils prétendent être. De plus, comme la signature est calculée à l'aide de l'en-tête et de la charge utile, nous pouvons également vérifier que le contenu n'a pas été altéré.

3.5.2 Structure des JWT

Dans sa forme compacte, les Jetons Web JSON se composent de trois parties séparées par des points (.), qui sont :

- **Header** (Entête) : L'en-tête se compose généralement de deux parties : le type du jeton, qui est JWT, et l'algorithme de hachage utilisé, tel que HMAC SHA256 ou RSA.
- **Payload** (Charge utile) : La deuxième partie du jeton est la charge utile, qui contient les

réclamations. Les réclamations sont des déclarations concernant une entité (généralement l'utilisateur) et des métadonnées supplémentaires. Il existe trois types de réclamations : les réclamations enregistrées, publiques et privées.

- **Signature** : Pour créer la partie de signature, nous devons prendre l'en-tête codé, la charge utile codée, un secret, l'algorithme spécifié dans l'en-tête, et signer cela.

La figure 3.3 dans la page 29 représente la structure des JWT avec des exemples.

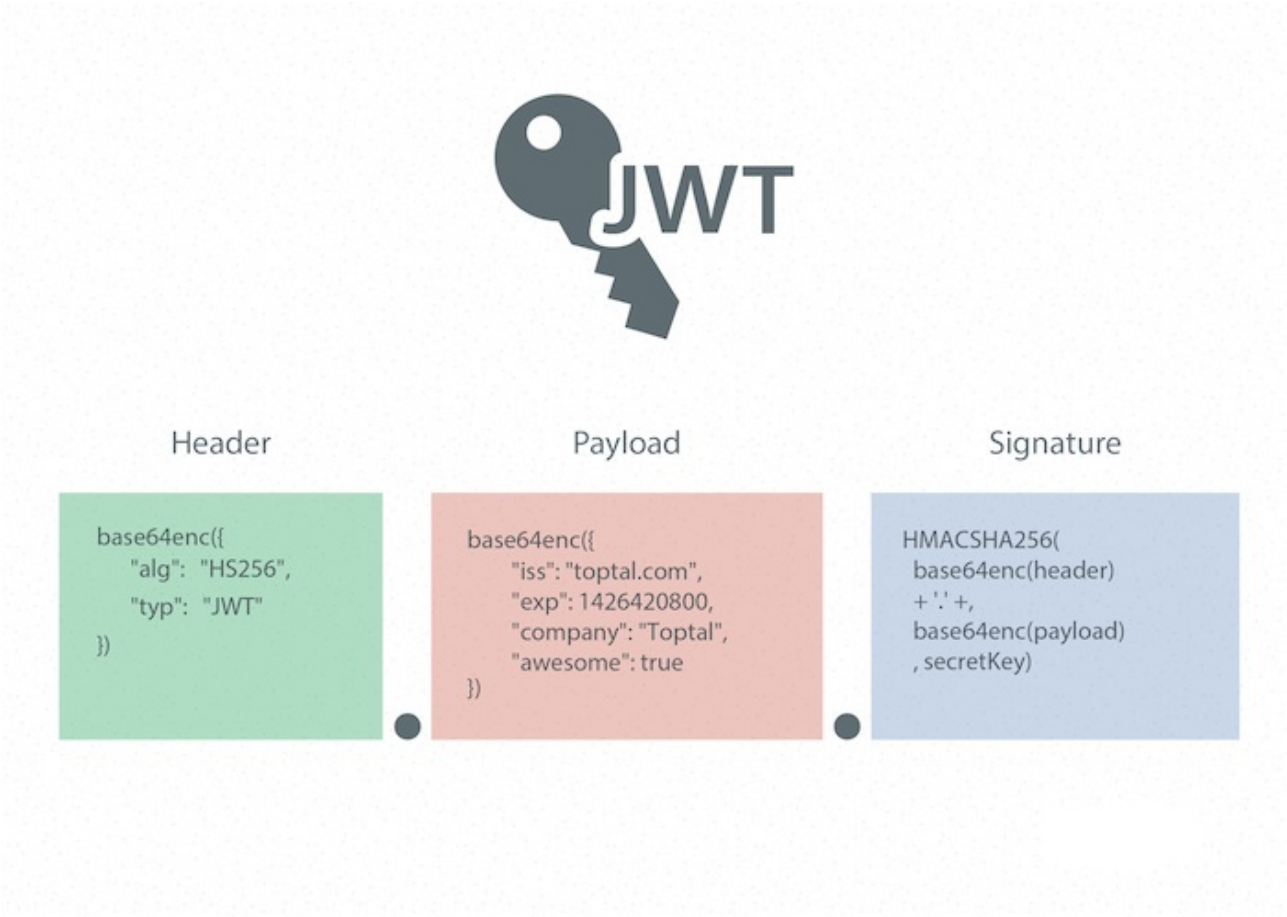

Figure 3.3 : Structure des Jetons Web JSON (JWT)

3.5.3 Avantages des JWT

- Comme JSON est moins verbeux que XML, lorsqu'il est codé, sa taille est également plus petite, rendant JWT plus compact. Cela fait de JWT un bon choix à passer dans les environnements HTML et HTTP.
- Les analyseurs JSON sont courants dans la plupart des langages de programmation car ils sont directement mappés aux objets. Inversement, XML n'a pas de mappage naturel de document à objet. Cela rend plus facile de travailler avec JWT.
- En ce qui concerne l'utilisation, JWT est utilisé à l'échelle Internet. Cela met en évidence la facilité du traitement côté client du Jeton Web JSON sur plusieurs plates-formes, en particulier sur les mobiles.

3.5.4 Fonctionnement des JWT

Dans l'authentification, lorsque l'utilisateur se connecte à l'aide de ses informations d'identification, un Jeton Web JSON est renvoyé et doit être enregistré localement (généralement dans le stockage local, mais les cookies peuvent également être utilisés) au lieu de l'approche traditionnelle. serveur et renvoyer un cookie.

Chaque fois que l'utilisateur veut accéder à un itinéraire protégé ou à une ressource, l'agent utilisateur doit envoyer le JWT, généralement dans l'en-tête **Authorization** en utilisant le schéma **Bearer**. Le contenu de l'en-tête devrait ressembler à ceci :

- **Authorization : Bearer <token>**

C'est un mécanisme d'authentification sans état puisque l'état de l'utilisateur n'est jamais sauvegardé dans la mémoire du serveur. Les routes protégées du serveur vérifient la présence d'un JWT valide dans l'en-tête Authorization, et s'il est présent, l'utilisateur sera autorisé à accéder aux ressources protégées. Comme les JWT sont autonomes, toutes les informations nécessaires sont présentes, ce qui réduit le besoin d'interroger la base de données plusieurs fois.

Cela nous permet de nous fier entièrement aux API de données qui sont sans état et même de faire des demandes aux services en aval. Peu importe les domaines desservant nos API, le partage de ressources d'origine croisée (Cross-Origin Resource Sharing (CORS)) ne pose pas de problème car il n'utilise pas de cookies.

Le diagramme dans la figure 3.4 montre ce processus :

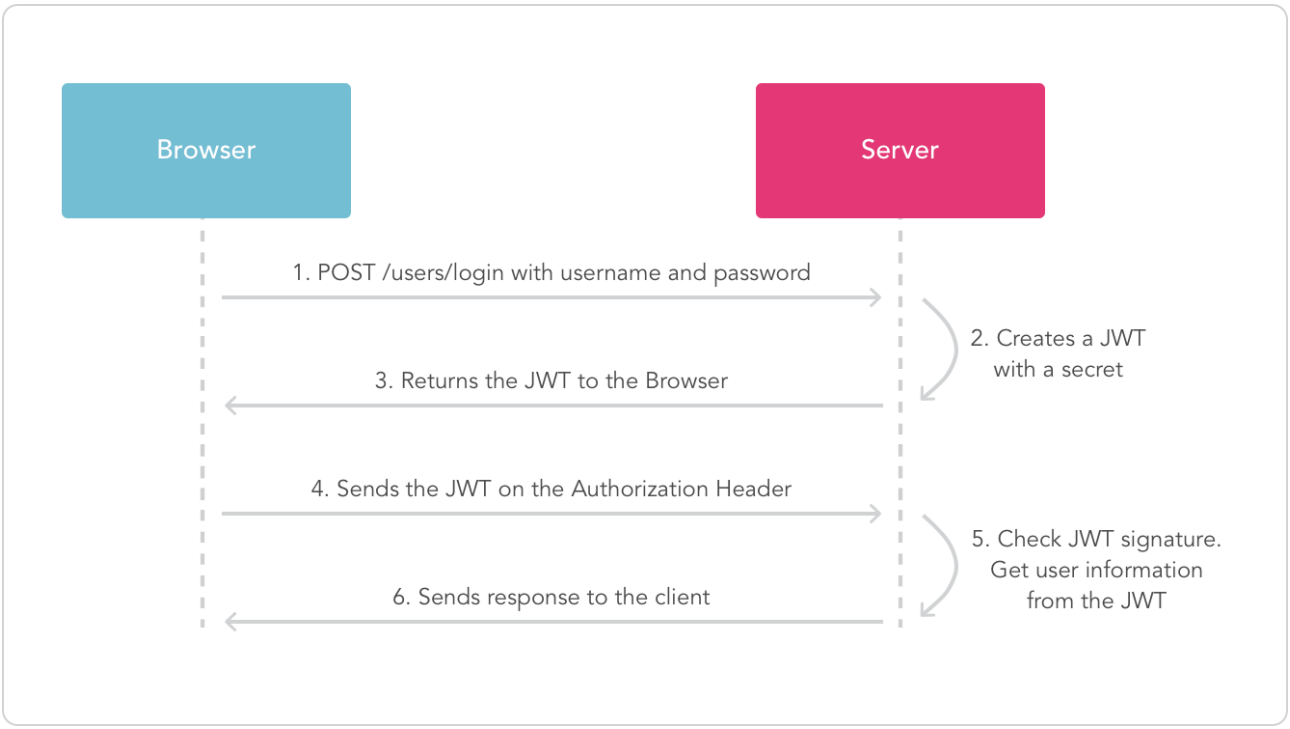

Figure 3.4 : Processus des JSON Web Tokens

3.6 L'architecture d'authentification et web services

Pour assurer l'authentification et les web services de notre projet, nous avons utiliser une architecture **Spring Security** qui intègre par défaut un ensemble de filtres permettant de configurer différents mécanismes de sécurité proposés par le module.. La figure 3.5 dans la page 32 représente l'architecture de sécurité de notre projet.

La SecurityFilterChain par défaut dans SpringBoot est la DefaultSecurityFilterChain qui sans aucune configuration contient 16 filtres différents. Parmis ces filtres nous allons voir quelques-uns pour comprendre un peu mieux comment Spring Security intervient sur notre requête et implémente ses protocoles d'authentification et de sécurité.

3.6.1 Le filtre : SecurityContextHolderFilter

Le **SecurityContext** est au cœur du fonctionnement de l'authentification, c'est à l'intérieur de celui-ci que l'identité de l'utilisateur, ses identifiants ainsi que ses droits seront injectés dans le **SecurityContextHolder** afin d'être accessible au sein de la chaine de validation. Cette information permettra à Spring de définir si l'utilisateur a le droit d'accéder à la ressource appelée ou non.

Le **SecurityContextHolderFilter** est chargé d'appliquer la stratégie de contexte propre

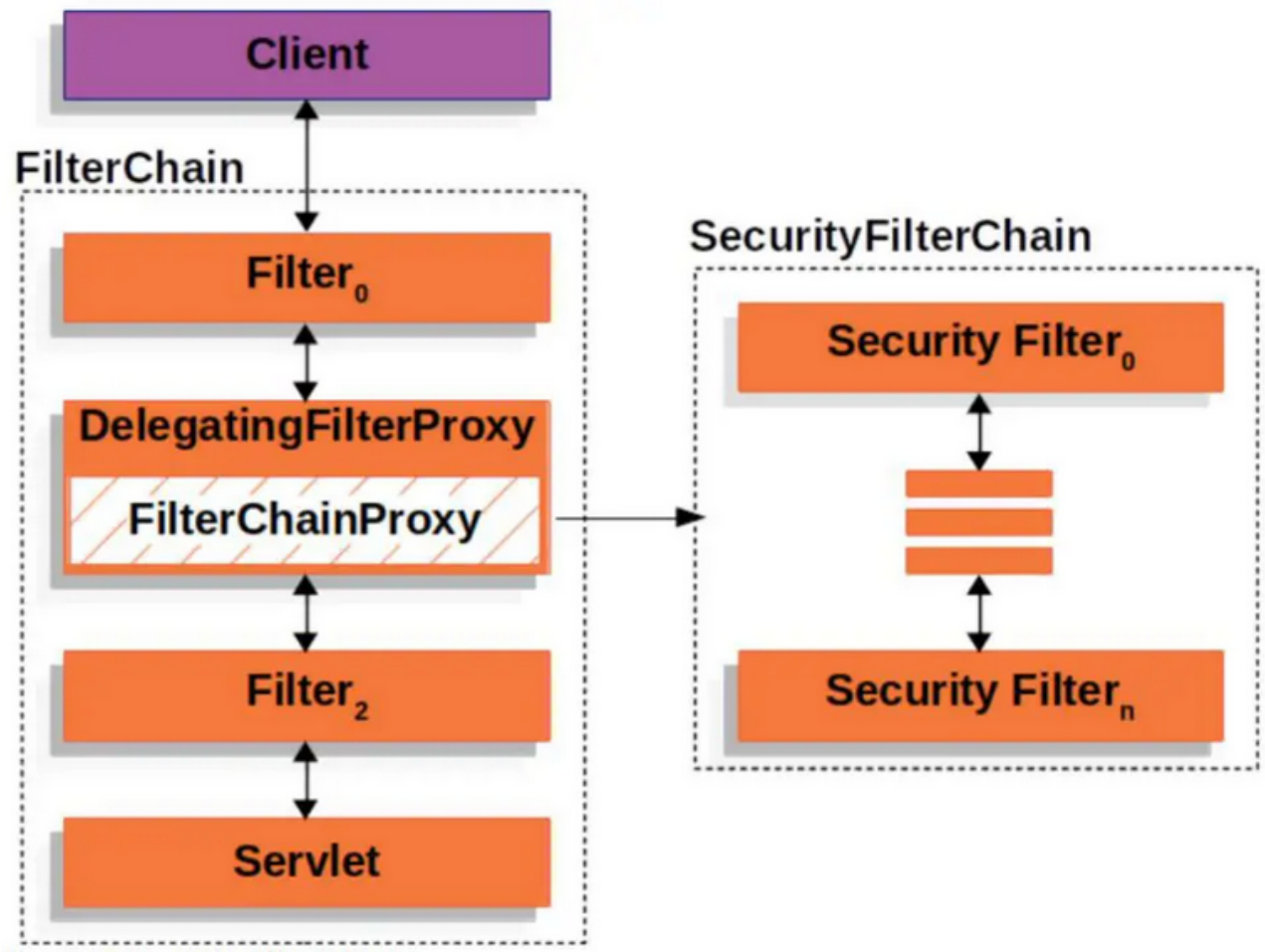

Figure 3.5 : Architecture de sécurité du projet

à notre requête, la configuration par défaut de Spring Security applique la **ThreadLocalSecurityContextHolderStrategy** qui stocke le contexte dans un **ThreadLocal** permettant d'isoler celui-ci des autres threads pouvant être exécuté par Spring. Le contexte est disponible pour toutes les méthodes présentes au sein du thread courant et il n'est donc pas nécessaire de conserver manuellement une référence pour pouvoir y accéder.

La figure 3.6 dans la page 33 représente ce filtre.

3.6.2 Le filtre : BasicAuthenticationFilter

Si l'authentification est nécessaire, le filtre commence l'exécution du cheminement standard de Spring Security. Ce protocole va permettre à Spring Security d'appliquer une méthode de vérification à la méthode d'authenfication. La figure 3.7 représente ce scénario :

- Le token qui a été généré par notre filtre implémente la classe **AbstractAuthenticationToken** qui implémente elle-même la classe **Authentication**, elle représente une forme standard et valide d'authentification.

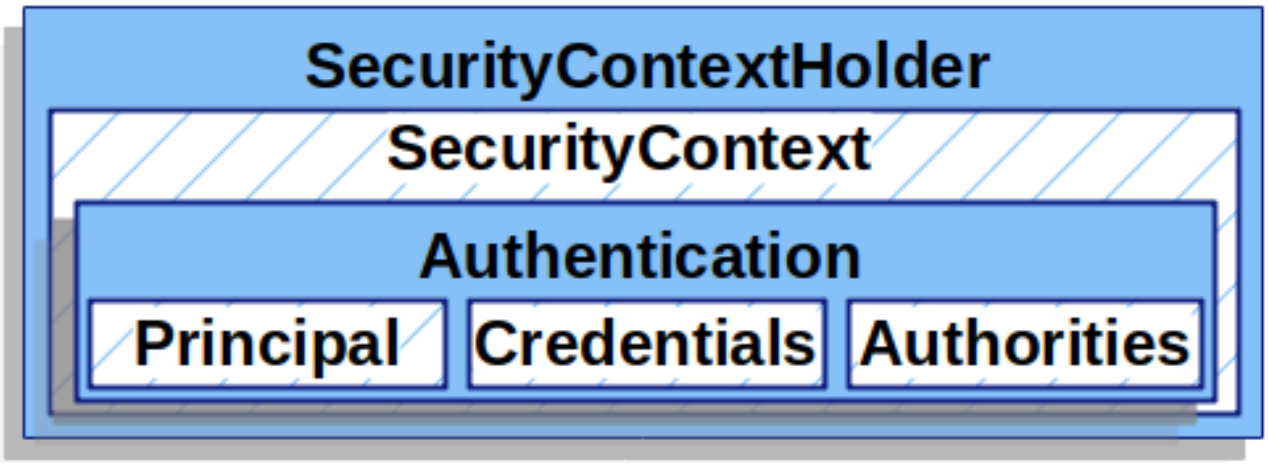

Figure 3.6 : Représentation de SecurityContextHolder

- Le filtre va envoyer le token au **AuthenticationManager**. Par défaut c'est un **ProviderManager** qui contient une liste d'**AuthenticationProvider**, une classe chargée de recevoir un objet Authentication et de fournir une réponse à la question suivante : L'Authentication fournie est-elle valide et si oui, quel droit possède-t-elle ?
- Cependant, il peut exister plusieurs manières de répondre à ces questions dans une application en fonction du type d'authentification utilisé. Un nombre illimité de providers peut être ajouté mais un seul provider sera utilisé.

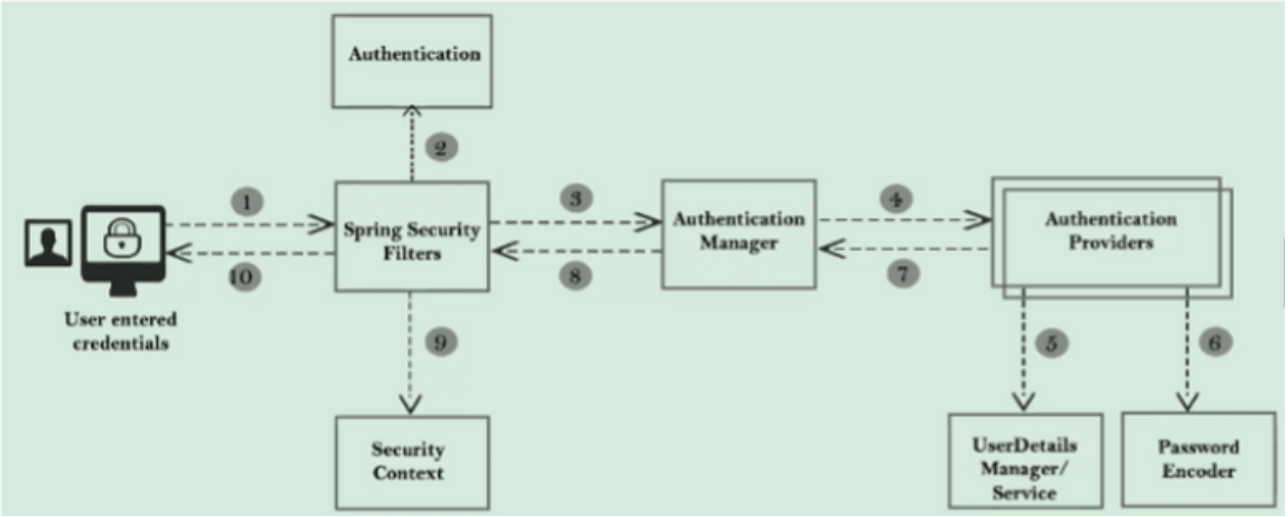

Figure 3.7 : Scénario de l'authentification

Conclusion

Après avoir décortiquer la partie conception et la partie sécurité dans ce chapitre, il s'agit désormais de se focaliser sur l'aspect pratique de notre application dans le chapitre suivant.

Chapitre 4

Réalisation

Conclusion générale et Perspectives

Bibliographique

[1] Le Langage de Modélisation Unifié (UML). https://openclassrooms.com/courses/debutez-l-analyse-logicielle-avec-uml/uml-c-est-quoi. [En ligne ; consulté le 15-Avril-2018].

[2] UML 2. http://laurent-audibert.developpez.com/Cours-UML/?page=diagramme-cas-utilisation. [En ligne ; consulté le 20-Avril-2018].

[3] JSON Web Tokens. https://jwt.io. [En ligne ; consulté le 21-Mai-2018].

ملخّص

كلمات مفاتيح :

Résumé

Mots clés :

Abstract

Keywords :