



Département d'informatique

Licence fondamentale:

Sciences Mathématiques et Informatique (SMI)

Projet de Fin d'Études

Intitulé:

Conception et réalisation d'une application web de gestion de patrimoines au sein de la FSM

Réalisé par:

Chergui Yassir
Rhazi Chaimae

Encadré par:

Pr. Ali Oubelkacem

Soutenu le 09 / 07 /2024 devant les jurys:

- ★ Pr A.Oubelkacem, Professeur à la Faculté des Sciences-Meknès
- ★ Pr A.Bekri, Professeur à la Faculté des Sciences-Meknès
- ★ Pr ED.El-Allaly, Professeur à la Faculté des Sciences-Meknès

Année Universitaire 2023-2024

Remerciements

Nous tenons tout d'abord à exprimer notre profonde gratitude à Dieu le tout-puissant et miséricordieux, qui nous a accordé la force, la persévérance et la guidance nécessaires pour mener à bien ce projet. Sa lumière a éclairé notre chemin à chaque étape de ce périple académique.

En second lieu, nous souhaitons adresser nos sincères remerciements à nos enseignants et, en particulier, à notre encadrant [M. Ali Oubelkacem](#), pour sa précieuse contribution. Son expertise, ses conseils éclairés et son soutien indéfectible ont été d'une importance capitale pour la réussite de ce travail. Son dévouement et sa disponibilité ont été des sources d'inspiration et d'encouragement tout au long de ce parcours.

Nous tenons également à exprimer notre reconnaissance envers nos proches, nos amis et notre famille, pour leur soutien inconditionnel, leurs encouragements et leur compréhension. Leur présence bienveillante a été un soutien précieux dans les moments de doute et d'effort.

Enfin, nous adressons nos remerciements à toutes les personnes qui, de près ou de loin, ont contribué à la réalisation de ce projet. Leur soutien moral, leurs conseils avisés et leurs encouragements ont été des moteurs essentiels dans notre démarche.

À tous, nous disons merci du fond du cœur pour avoir été à nos côtés dans cette aventure académique.

Résumé

Notre projet de fin d'études porte sur le développement d'une application de gestion des patrimoines. Nous avons divisé notre travail en plusieurs phases, comprenant une phase d'analyse et de conception, suivie d'une phase de développement. Avec l'encadrement et le soutien de notre responsable de projet, nous avons acquis une expérience significative au cours des derniers mois, en utilisant divers outils et en explorant plusieurs concepts clés nécessaires à la réalisation de cette application. Ce rapport détaillera les étapes que nous avons suivies ainsi que les outils que nous avons utilisés pour mener à bien ce projet

Table des Matières

Remerciements	2
Résumé	3
Introduction Général	9
1 Introduction	9
2 Définition de Quelques Concepts	9
3 Contexte et Problématique	10
4 Besoin de l'Institution	11
Objectif du projet	12
Méthodologie	13
1 Analyse des Besoins	13
2 Conception de l'Application	13
3 Développement et Mise en Œuvre.....	13
4 Formation et Déploiement	13
1. Chapitre 1 : Analyse et Conception	15
1.1. Analyse des Besoins	15
1.1.1. Description détaillée des besoins	15
1.1.2. Diagrammes de cas d'utilisation	15
1.2. Spécifications Fonctionnelles	17
1.2.1. Fonctionnalités principales	17
1.2.2. Scénarios d'utilisation	17
1.3. Conception Technique	17
1.3.1. Architecture du système	17
1.3.2. Diagrammes UML	18
1.3.2.1. Diagrammes de cas d'utilisation	18
1.3.2.1.1. Diagramme de cas d'utilisation Authentification.....	18

1.3.2.1.2. Diagramme de cas d'utilisation - Admin	21
1.3.2.1.3. Diagramme de cas d'utilisation - Utilisateur	26
1.3.2.2. Diagrammes de séquence.....	27
1.3.2.2.1 Diagramme de séquence Inscrit/Auth.....	27
1.3.2.2.2 Diagramme de séquence-Admin.....	30
1.3.2.2.3 Diagramme de séquence-Utilisateur.....	34
1.3.2.3. Diagramme de classes..	36
1.3.2.4. Diagrammes d'activités	40
1.3.2.4.1 Diagramme d'activités Inscrit/Auth.....	40
1.3.2.4.2 Diagramme d'activités Admin.....	42
1.3.2.4.3 Diagramme d'activités utilisateur.....	46
2. Chapitre 2 : Modèle de données.....	50
2.1 Introduction.....	50
2.2 Choix des technologies.....	50
2.3 Intégration entre MYSQL et spring boot.....	50
2.4 Conclusion.....	54
3. Chapitre 3 :Frontend avec Angular.....	55
3.1 Introduction.....	55
3.2 Choix d'angular	55
3.3 Concepts Clés d'Angular.....	55
3.3.1 Exemple du composant.....	55
3.3.2 Exemple de service.....	56
3.3.3 Exemple de routage.....	57
3.4 Intégration avec le Backend Spring boot.....	57
3.4.1 Exemple de service Angular pour consommer une API REST.....	57
3.5 Interface Admin.....	58

3.5.1 Page de connexion.....	58
3.5.2 Page d'accueil.....	58
3.5.3 Page gestion des admins.....	60
3.5.4 Exemple traitement d'une demande.....	60
3.5.5 Configuration email de l'application.....	61
3.6 Interface utilisateur.....	62
3.6.1 Page d'inscription.....	62
3.6.2 Page de connexion.....	63
3.6.3 Page d'accueil.....	63
3.6.4 Page demandes achevées.....	64
3.6.5 Créer une demande.....	64
3.6.6 Accessibilité Mobile.....	65
4. Chapitre 4 : Déploiement de l'Application	67
4.1 Introduction	67
4.2 Préparation de l'Environnement de Déploiement	67
4.3 Déploiement du Backend avec Docker	67
4.4 Déploiement du Frontend avec Docker	69
4.5 Conclusion	71
Clôture.....	72

Liste des figures

Figure 1 : UC Globale.....	16
Figure 2 : UC Authentification.....	18
Figure 3 : UC Admin.....	21
Figure 4 : UC Utilisateur.....	26
Figure 5 : Séquence Inscrit/Auth.....	28
Figure 6 : Séquence Admin.....	31
Figure 7 : Séquence Utilisateur.....	34
Figure 8 : Diagramme de classe.....	36
Figure 9 : Activité Inscrit/Auth.....	40
Figure 10 : Activité Admin.....	43
Figure 11 : Activité Utilisateur.....	47
Figure 12 : Table Admin.....	52
Figure 13 : Table Demande.....	53
Figure 14 : Page de connexion.....	58
Figure 15 : Page d'accueil (super_admin).....	59
Figure 16 : Page d'accueil (!super_admin).....	59
Figure 17 : Accès refusé.....	60
Figure 18 : Page gestion des admins.....	60
Figure 19 : Formulaire traitement demande.....	61
Figure 20 : Formulaire de configuration de l'email de l'application.....	61
Figure 21 : Page de vérification.....	62
Figure 22 : Page d'inscription.....	63
Figure 23 : Page de connexion.....	63
Figure 24 : Page d'accueil.....	64
Figure 25 : Page demandes achevées.....	64

Figure 26 : Créer demande.....65

Figure 27 : Interface utilisateur affichée sur un Smartphone.....66

Introduction générale

1 Introduction:

La gestion rationnelle des ressources est un aspect crucial de toute institution moderne. Pour la Faculté des Sciences de Meknès (FSM), cette gestion se situe au carrefour de la gestion du temps et de la gestion du patrimoine. La mise au point d'une application destinée à assurer une circulation rapide de l'information entre les différents acteurs, opérateurs et usagers des équipements de la FSM vise à réduire les délais d'intervention requis pour rendre les dits équipements opérationnels. Cette approche contribue non seulement à sauvegarder le matériel et les équipements mais aussi à enrichir notre savoir-faire en matière d'usage des technologies de l'information et de la communication (TIC).

2 Définition de quelques concepts :

Plusieurs approches peuvent être faites pour définir le patrimoine. Chacune privilégiant un aspect selon l'objet de l'analyse.

Le patrimoine peut être défini comme étant l'ensemble des actifs d'une organisation. On peut distinguer à ce titre, Le Patrimoine matériel, qui est constitué par l'ensemble des immobilisations et des équipements .

Le Patrimoine immatériel qui correspond à la production intellectuelle, le savoir-faire scientifique et technique, en général et dans le cas des entreprises privées, ce sont les brevets d'invention ...

Le patrimoine d'une organisation est intrinsèquement lié au développement,c'est le soubassement logistique de son action. Un patrimoine opérationnel est la garantie de la poursuite de l'action et des objectifs de cette organisation. D'où la nécessité de la sauvegarde de ce patrimoine voir son développement....

La gestion du patrimoine de la FSM est une impérieuse nécessité dictée par plusieurs considérations :

- En premier lieu, toute Institution soucieuse de l'efficacité de son action, doit veiller à sauvegarder la rationalité dans l'usage des moyens dont elle dispose en vue d'optimiser l'affectation des dits moyens aux fins qu'elle se propose d'atteindre.
- En deuxième instance, ce déploiement rationnel des moyens est d'autant plus nécessaire qu'il s'agit d'un établissement public, dont les fonds sont pourvus par le budget général de l'État, donc toute mesure visant à éviter le gaspillage est impérative.
- En troisième lieu, la nature même de l'établissement qu'est la FSM, destiné à dispenser une formation suivant un calendrier bien établi, ne pouvant souffrir du moindre retard dans la mise à la disposition des classes des prestations indispensables au déroulement de leurs travaux (**dans des conditions acceptables**), requiert le respect des grilles temporelles fixées pour chaque tâche.
- La gestion quotidienne des ressources est un processus global, dont les composantes sont interdépendantes les unes des autres, donc tout à-coup ou perturbation dans un niveau, se répercute de manière néfaste sur l'ensemble du dispositif compromettant son fonctionnement normal.

- En d'autres termes, la fluidité des flux d'information et de fournitures et le respect de leur échelonnement temporel jouent un rôle crucial dans la cohérence des interventions et la viabilité du dispositif.
- La carence d'une fourniture, si minime soit-elle, pourrait se traduire, parfois par des pertes de temps et engendrer des perturbations inconsidérées.
- Un micro déréglé
- Une ampoule défectueuse
- Un interrupteur dévissé
- Un dispositif expérimental incomplet ou non préparé dans les délais requis,
- Un siège bancal,

Toutes ces anomalies, si elles ne sont pas traitées, à priori, sont sources de nuisances non seulement en termes de perte de temps, mais aussi en terme de pertes de ressources, car une défectuosité non réparée au moment opportun, pourrait s'amplifier et exiger des dépenses coûteuses.

3 Contexte et Problématique:

a) Contexte:

Dans toute institution académique, la gestion efficace des patrimoines et des ressources est essentielle pour assurer un fonctionnement fluide et une utilisation optimale des moyens disponibles. La Faculté des Sciences de Meknès (FSM) ne fait pas exception à cette règle. En tant qu'institution publique dédiée à l'enseignement et à la recherche, la FSM dispose d'un vaste ensemble de patrimoines matériels et immatériels qui incluent des bâtiments, des équipements de laboratoire, des ressources didactiques, ainsi que des connaissances et compétences spécialisées.

La gestion de ces ressources présente plusieurs défis, notamment en termes de maintenance des équipements, de gestion des infrastructures et de coordination entre les différents services et utilisateurs. Une gestion inefficace peut entraîner des retards dans les opérations, des pannes d'équipements non résolues, et des gaspillages de ressources précieuses. Par conséquent, il est crucial de mettre en place des systèmes et des processus qui facilitent la gestion proactive et réactive des patrimoines de l'institution.

b) Problématique:

La problématique à laquelle nous nous intéressons se situe à l'intersection de la gestion du temps et de la gestion des patrimoines de la FSM. Actuellement, plusieurs défis se posent :

- **Délais de Réparation et de Maintenance** Les délais pour la réparation et la maintenance des équipements sont souvent longs, ce qui perturbe le déroulement des activités pédagogiques et de recherche. Un équipement non opérationnel peut retarder les cours, les travaux pratiques, et les projets de recherche.
- **Communication Inefficace** La communication entre les différents acteurs (enseignants, techniciens, administrateurs) et les services de maintenance est souvent lente et inefficace. Les demandes de réparation ou d'entretien peuvent se perdre ou être retardées en raison de la multiplicité des canaux de communication.

- **Gestion Réactive Plutôt que Proactive** La gestion actuelle des patrimoines est majoritairement réactive. Les interventions de maintenance sont souvent déclenchées après la survenue de pannes, plutôt que par des actions préventives planifiées. Cela conduit à des coûts plus élevés et à des temps d'arrêt plus longs.
- **Utilisation Inefficace des Ressources** Les ressources disponibles ne sont pas toujours utilisées de manière optimale. Le manque de coordination et de planification peut entraîner une sous-utilisation ou un gaspillage des équipements et des infrastructures.

4 Besoin d'institution:

Pour répondre à ces défis, la FSM a besoin d'une solution intégrée qui permette :

- **Une Gestion Centralisée et Automatisée** : Un système centralisé pour gérer toutes les demandes de maintenance et de réparation, suivre les interventions et assurer une communication fluide entre les services.
- **Maintenance Préventive** : Des outils pour planifier et réaliser des actions de maintenance préventive, afin de réduire les pannes imprévues et d'allonger la durée de vie des équipements.
- **Suivi en Temps Réel** : Une surveillance en temps réel des équipements critiques pour détecter les problèmes potentiels avant qu'ils ne deviennent graves.
- **Amélioration de la Communication** : Des canaux de communication efficaces et rapides entre tous les acteurs impliqués dans la gestion des patrimoines.

En développant une application web de gestion des patrimoines, nous visons à répondre à ces besoins de manière efficace et durable, en intégrant les meilleures pratiques de gestion des ressources et les dernières avancées technologiques.

Objectif du projet

L'objectif principal de ce projet est de concevoir et de mettre en œuvre une application web destinée à améliorer la gestion des patrimoines de la FSM. Cette application devra :

- ★ Réduire les délais d'intervention pour les réparations et la maintenance.
- ★ Faciliter une communication rapide et efficace entre les différents acteurs.
- ★ Optimiser l'utilisation des ressources disponibles pour éviter le gaspillage.
- ★ Assurer un suivi détaillé et un historique des interventions de maintenance.

Méthodologie

Pour atteindre les objectifs définis pour ce projet, nous avons suivi une méthodologie en plusieurs étapes structurées. Chaque étape joue un rôle crucial dans la conception, le développement, et la mise en œuvre de l'application web de gestion des patrimoines pour la Faculté des Sciences de Meknès (FSM).

1 Analyse des Besoins

L'analyse des besoins est la première étape et la plus cruciale pour garantir que l'application répondra aux attentes et exigences de tous les utilisateurs. Cette étape comprend les activités suivantes :

- **Recueil des besoins** : Entretiens avec les parties prenantes (enseignants, administrateurs) pour comprendre leurs attentes et leurs problèmes actuels.
- **Priorisation des besoins** : Classement des besoins en termes de priorité pour s'assurer que les fonctionnalités les plus critiques sont développées en premier.

2 Conception de l'Application

La conception de l'application se base sur les résultats de l'analyse des besoins et inclut la planification de l'architecture technique et fonctionnelle de l'application :

- **Spécifications fonctionnelles** : Définition des fonctionnalités de l'application, y compris les cas d'utilisation, les scénarios d'utilisation, et les exigences techniques.
- **Design de l'interface utilisateur** : Création de maquettes et prototypes pour l'interface utilisateur, assurant une expérience utilisateur intuitive et efficace.
- **Design de l'interface admin** : Développement de l'interface spécifique pour les administrateurs, permettant une gestion efficace et intuitive des patrimoines.
- **Architecture du système** : Planification de l'architecture technique, incluant le choix des technologies, la structure des bases de données, et l'organisation des composants logiciels.

3 Développement et Mise en Œuvre

Cette phase inclut le codage de l'application, ainsi que l'intégration et la configuration des différentes composantes du système :

- **Développement** : Programmation des différentes fonctionnalités de l'application en suivant les spécifications définies. Utilisation de technologies et de langages de programmation appropriés, notamment Angular Material pour le frontend et Spring Boot pour le backend.
- **Intégration** : Assemblage des différents modules et composants de l'application, assurant leur fonctionnement harmonieux ensemble.
- **Tests unitaires** : Tests des différents composants individuels pour vérifier leur bon fonctionnement avant l'intégration.

4 Formation et Déploiement

Une fois l'application développée et testée, elle est prête pour être déployée et utilisée par les utilisateurs finaux :

- **Plan de déploiement** : Définition des étapes nécessaires pour installer et configurer l'application dans l'environnement de production.
- **Formation des utilisateurs** : Organisation de sessions de formation pour les utilisateurs finaux, accompagnées de guides et de manuels d'utilisation.

1. Chapitre 1 : Analyse et Conception.

1.1 Analyse des Besoins

1.1.1 Description détaillée des besoins

Dans cette section, nous détaillons les besoins spécifiques identifiés pour la gestion des patrimoines à la Faculté des Sciences de Meknès (FSM). Ces besoins sont basés sur les entretiens avec les parties prenantes.

- **Gestion des équipements :** La FSM possède divers équipements scientifiques et pédagogiques qui nécessitent une gestion rigoureuse pour assurer leur disponibilité et bon fonctionnement.
- **Maintenance préventive et corrective :** Besoin d'un suivi des interventions de maintenance pour minimiser les pannes et prolonger la durée de vie des équipements.
- **Suivi des inventaires :** Une gestion précise des inventaires pour éviter les pertes et optimiser l'utilisation des ressources disponibles.
- **Interface utilisateur intuitive :** L'application doit offrir une interface utilisateur simple et efficace pour faciliter l'utilisation par les différents acteurs (enseignants, étudiants, administrateurs).

1.1.2 Diagrammes de cas d'utilisation

Les diagrammes de cas d'utilisation permettent de visualiser les interactions entre les utilisateurs et le système. Voici le diagramme global des cas d'utilisation pour l'application :

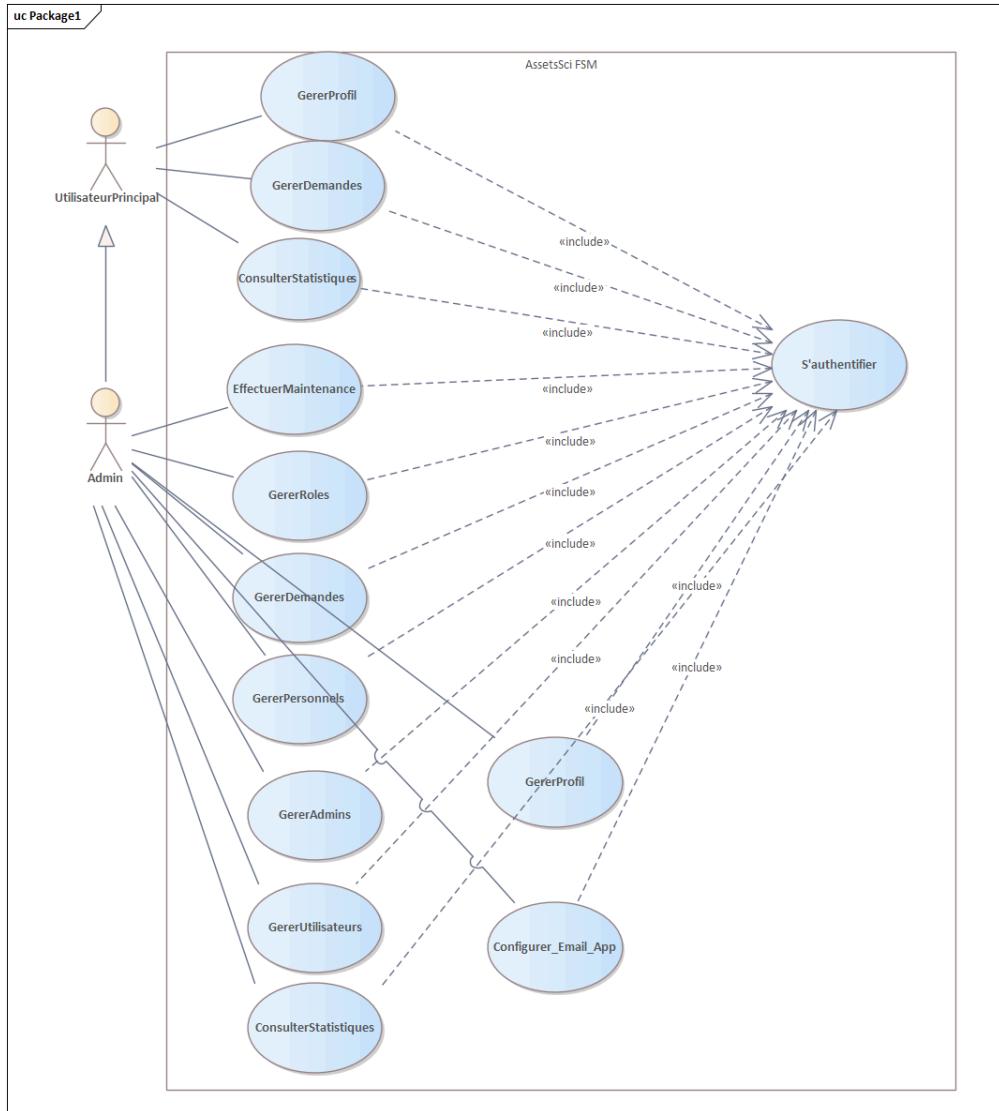


Figure 1 : UC Globale

Description des cas d'utilisation :

Gérer Profil : Permet à l'utilisateur principal et à l'administrateur de gérer leur propre profil.

Gérer Demandes : Permet à l'utilisateur principal et à l'administrateur de gérer les demandes.

Consulter Statistiques : Permet à l'utilisateur principal et à l'administrateur de consulter des statistiques.

Effectuer Maintenance : Permet à l'administrateur d'activer un mode maintenance qui désactive temporairement l'accès des utilisateurs en affichant une page de maintenance.

Gérer Rôles : Permet à l'administrateur de gérer les rôles des utilisateurs.

Gérer Personnels : Permet à l'administrateur de gérer les informations du personnel.

Configurer email de l'application : Permet à l'administrateur de configurer l'email de l'application pour l'envoie des notifications aux utilisateurs et aux administrateurs.

Gérer Admins : Permet à l'administrateur de gérer les administrateurs.

Gérer Utilisateurs : Permet à l'administrateur de gérer les utilisateurs.

S'authentifier : Inclut les actions d'authentification nécessaires pour accéder aux fonctionnalités de l'application.

1.2 Spécifications Fonctionnelles

1.2.1 Fonctionnalités principales

Les spécifications fonctionnelles détaillent les fonctionnalités principales que l'application doit offrir :

- **Gestion des demandes de maintenance** : Interface pour soumettre, suivre et traiter les demandes de maintenance.
- **Notifications** : Système de notifications pour informer les utilisateurs des mises à jour sur les demandes de maintenance.
- **Mode maintenance** : Fonctionnalité permettant à l'administrateur d'activer un mode maintenance, désactivant temporairement l'accès des utilisateurs et affichant une page de maintenance.

1.2.2 Scénarios d'utilisation

Les scénarios d'utilisation illustrent comment les utilisateurs interagiront avec le système dans différentes situations :

- ➔ **Scénario 1** : Un enseignant découvre une panne dans un laboratoire et soumet une demande de maintenance via l'application. L'administrateur reçoit une notification par e-mail ainsi que la demande ajoutée dans la liste des demandes dans son interface. Ensuite, l'administrateur attribue la demande à un personnel approprié en modifiant le statut de la demande à "en cours" et en notifiant l'utilisateur par email. L'utilisateur peut voir cette mise à jour soit par email soit via le site
- ➔ **Scénario 2** : Après avoir attribué la demande, l'administrateur contacte le personnel par téléphone pour lui expliquer la nature de la demande. Une fois que le personnel a réparé l'équipement, l'administrateur modifie le statut de la demande à "traitée" et notifie l'utilisateur par email que la demande a été traitée. L'utilisateur peut voir cette mise à jour soit par email soit via le site.
- ➔ **Scénario 5** : L'administrateur active le mode maintenance pour effectuer des mises à jour sur le système. Les utilisateurs voient une page de maintenance et ne peuvent pas accéder à leurs fonctionnalités habituelles jusqu'à la désactivation de ce mode.

1.3 Conception Technique

1.3.1 Architecture du système

L'architecture du système décrit les composants principaux et leur interaction. L'application sera basée sur une architecture multicouche, incluant :

- **Interface utilisateur/administrateur (Frontend)** : Développée en Angular avec Angular Material pour assurer une interface utilisateur intuitive et réactive.

- **Serveur d'applications (Backend)** : Utilisation de Spring Boot pour gérer la logique métier, les services web et l'accès aux données.
- **Base de données** : Une base de données relationnelle pour stocker les informations sur les équipements, les interventions de maintenance et les utilisateurs.
- **API de communication** : API RESTful pour la communication entre le frontend et le backend.

1.3.2. Diagrammes UML

1.3.2.1 Diagrammes de cas d'utilisation

1.3.2.1.1. Diagramme de cas d'utilisation Authentification

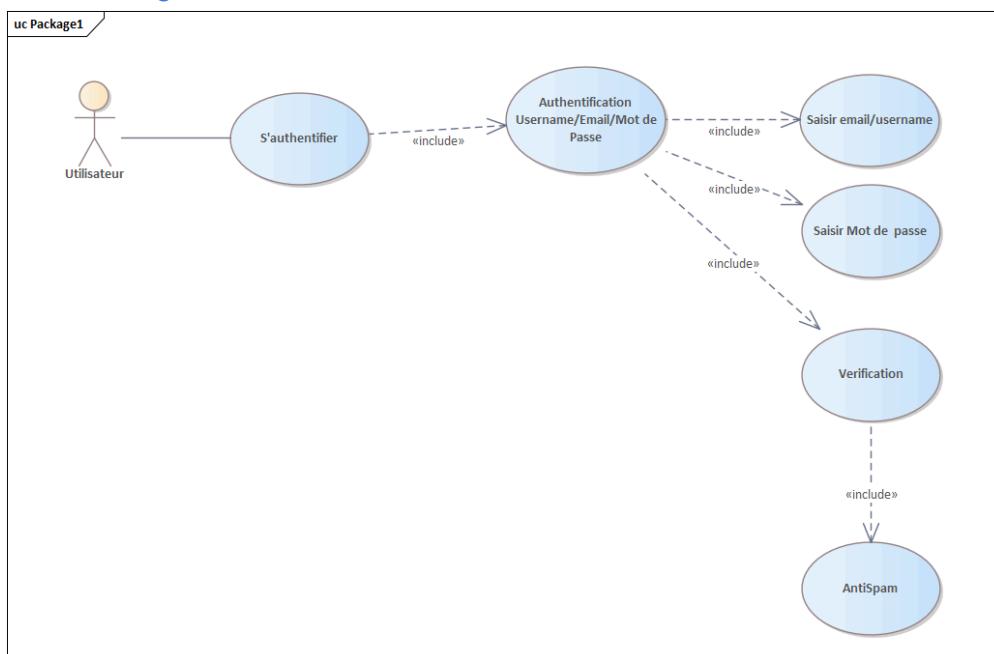


Figure 2 :UC Authentification

Le diagramme montre le processus d'authentification pour deux types d'utilisateurs : les utilisateurs normaux et les administrateurs. Voici une description détaillée des éléments et des interactions représentés dans le diagramme.

1) Acteurs:

- **Utilisateur Principal** : L'utilisateur normal qui s'authentifie en utilisant un nom d'utilisateur et un mot de passe.
- **Admin** : un mot de passe.L'administrateur qui s'authentifie en utilisant un email et

Cas d'Utilisation

1. S'authentifier (Utilisateur)

- **Description** : L'utilisateur normal initie le processus d'authentification.
- **Inclut** :
 - Authentification par Username et Mot de Passe

2. S'authentifier (Administrateur)

- **Description** : L'administrateur initie le processus d'authentification.
- **Inclut** :
 - Authentification par Email et Mot de Passe

3. Authentification par Username et Mot de Passe

- **Description** : Ce cas d'utilisation gère l'authentification de l'utilisateur en utilisant un nom d'utilisateur et un mot de passe.
- **Inclut** :
 - Saisir username
 - Saisir Mot de passe
 - Vérification

4. Authentification par Email et Mot de Passe

- **Description** : Ce cas d'utilisation gère l'authentification de l'administrateur en utilisant un email et un mot de passe.
- **Inclut** :
 - Saisir email
 - Saisir Mot de passe
 - Vérification

5. Saisir username

- **Description** : L'utilisateur entre son nom d'utilisateur.

6. Saisir email

- **Description** : L'administrateur entre son email.

7. Saisir Mot de Passe

- **Description** : L'utilisateur ou l'administrateur entre son mot de passe.

8. Vérification

- **Description** : Le système vérifie les informations d'identification fournies (nom d'utilisateur/email et mot de passe).
- **Inclut** :
 - AntiSpam

9. AntiSpam

- **Description** : Le système effectue des vérifications supplémentaires pour prévenir les tentatives de spam ou d'attaques par force brute. Si une tentative de spam est détectée, le système bloque l'utilisateur pendant 30 secondes avant de permettre une nouvelle tentative de connexion.

Scénario de Base pour Utilisateur

- ➔ L'utilisateur demande à s'authentifier.
- ➔ Le système lui demande de saisir son nom d'utilisateur.
- ➔ L'utilisateur saisit son nom d'utilisateur.
- ➔ Le système lui demande de saisir son mot de passe.
- ➔ L'utilisateur saisit son mot de passe.

- Le système vérifie les informations d'identification fournies.
- Le système passe par un processus AntiSpam pour vérifier qu'il ne s'agit pas d'une tentative de spam.
- Si toutes les vérifications sont réussies, l'utilisateur est authentifié et peut accéder au système.

Scénario de Base pour Administrateur

- L'administrateur demande à s'authentifier.
- Le système lui demande de saisir son email.
- L'administrateur saisit son email.
- Le système lui demande de saisir son mot de passe.
- L'administrateur saisit son mot de passe.
- Le système vérifie les informations d'identification fournies.
- Le système passe par un processus AntiSpam pour vérifier qu'il ne s'agit pas d'une tentative de spam.
- Si toutes les vérifications sont réussies, l'administrateur est authentifié et peut accéder au système.

Scénarios Alternatifs

- **Nom d'utilisateur/Email incorrect** : Si le nom d'utilisateur ou l'email est incorrect, le système informe l'utilisateur ou l'administrateur et lui redemande de saisir les informations correctes.
- **Mot de passe incorrect** : Si le mot de passe est incorrect, le système informe l'utilisateur ou l'administrateur et lui redemande de saisir le mot de passe.
- **Détection de spam** : Si le système détecte une tentative de spam, il bloque l'authentification de l'utilisateur ou de l'administrateur pendant 30 secondes avant de permettre une nouvelle tentative de connexion.

Conclusion

Ce cas d'utilisation montre un processus d'authentification distinct pour les utilisateurs normaux et les administrateurs, chacun utilisant des méthodes différentes (nom d'utilisateur/mot de passe pour les utilisateurs normaux et email/mot de passe pour les administrateurs), avec une vérification AntiSpam pour sécuriser le processus contre les tentatives de spam ou d'attaques par force brute.

1.3.2.1.2. Diagramme de cas d'utilisation – Admin

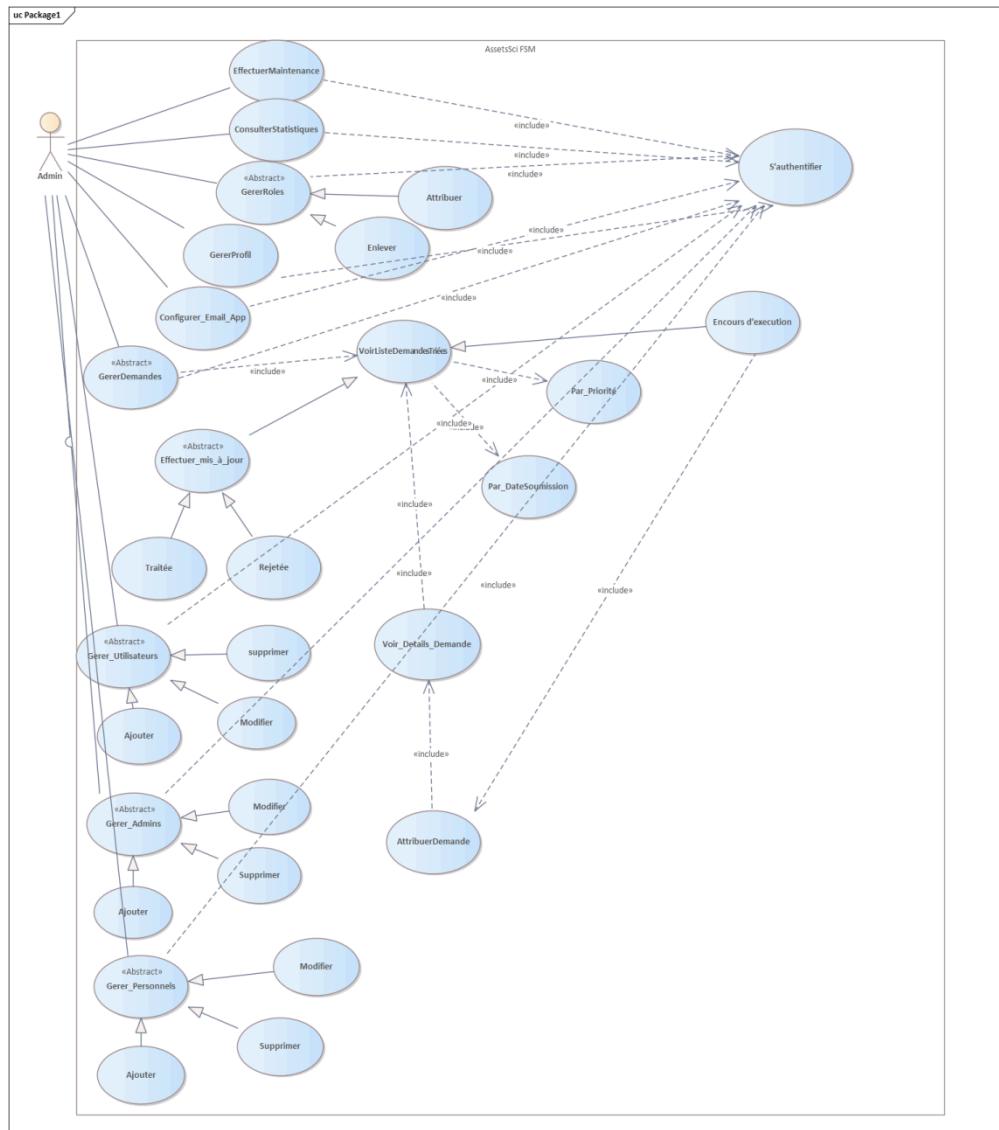


Figure 3 : UC Admin

Le diagramme de cas d'utilisation pour l'administrateur (Admin) présente les différentes fonctionnalités accessibles à ce type d'utilisateur au sein de l'application AssetsSci FSM. L'administrateur joue un rôle central dans la gestion et la maintenance de l'application. Voici les principaux cas d'utilisation identifiés :

1) Acteurs

Admin : L'administrateur qui s'authentifie et accède à diverses fonctionnalités de gestion.

Cas d'Utilisation

1. S'authentifier

- **Description** : L'administrateur initie le processus d'authentification en utilisant un email et un mot de passe.
- **Inclut** :
 - Saisir email
 - Saisir Mot de passe
 - Vérification
 - AntiSpam

2. Gérer Profils

- **Description** : L'administrateur peut gérer son propre profil, y compris le modifier.

3. Effectuer Maintenance

- **Description** : L'administrateur peut activer l'état de maintenance du site. Lorsque le site est en maintenance, les utilisateurs voient une page indiquant que le site est en maintenance.

4. Consulter Statistiques

- **Description** : L'administrateur peut consulter diverses statistiques sur l'utilisation du système.

5. Gérer Rôles

- **Description** : L'administrateur peut gérer les rôles des utilisateurs dans le système.

6. Gérer Demandes

- **Description** : L'administrateur peut gérer les demandes de maintenance soumises par les utilisateurs.
- **Sous-fonctions** :
 - Voir Liste Demandes
 - Attribuer Demandes
 - Traiter Demandes
 - Rejeter Demandes
 - En cours demandes

5. Gérer Utilisateurs

- **Description** : L'administrateur peut gérer les comptes des utilisateurs, y compris les ajouter, les modifier ou les supprimer.

6. Gérer Admins

- **Description** : L'administrateur peut gérer les comptes des autres administrateurs, y compris les ajouter, les modifier ou les supprimer.

7. Gérer Personnels

- **Description** : L'administrateur peut gérer les comptes des personnels, y compris les ajouter, les modifier ou les supprimer.

8. Configurer l'email de l'application

- **Description** : L'administrateur peut configurer l'email de l'application afin définir les paramètres nécessaires pour l'envoi des notifications aux utilisateurs et aux administrateurs

Détails des Cas d'Utilisation

Authentification

1. **Saisir email**
 - **Description** : L'administrateur entre son email.
2. **Saisir Mot de Passe**
 - **Description** : L'administrateur entre son mot de passe.
3. **Vérification**
 - **Description** : Le système vérifie les informations d'identification fournies (email et mot de passe).
4. **AntiSpam**
 - **Description** : Le système effectue des vérifications supplémentaires pour prévenir les tentatives de spam ou d'attaques par force brute. Si une tentative de spam est détectée, le système bloque l'utilisateur pendant 30 secondes avant de permettre une nouvelle tentative de connexion.

Gérer Demandes

1. **Voir Liste Demandes déjà triées par le système**
 - **Description** : L'administrateur peut voir la liste des demandes de maintenance soumises par les utilisateurs.
2. **Attribuer Demandes**
 - **Description** : L'administrateur peut attribuer des demandes de maintenance à des personnels appropriés.

- **Inclut** : Afficher Détails Demande

3.Traiter Demandes

- **Description** : L'administrateur peut marquer une demande comme traitée après qu'elle a été résolue.
- **Inclut** : Afficher Détails Demande, Mettre à Jour Statut

4. Rejeter Demandes

- **Description** : L'administrateur peut rejeter une demande de maintenance.
- **Inclut** : Afficher Détails Demande

5. En cours demandes

- **Description** : L'administrateur peut marquer une demande comme en cours après qu'elle a été attribuée à un personnel.
- **Inclut** : Afficher Détails Demande

Gérer Utilisateurs

1. Ajouter Utilisateur

- **Description** : L'administrateur peut ajouter de nouveaux utilisateurs au système.

2. Modifier Utilisateur

- **Description** : L'administrateur peut modifier les informations d'un utilisateur existant.

3. Supprimer Utilisateur

- **Description** : L'administrateur peut supprimer un utilisateur du système.

Gérer Admins

1. Ajouter Admin

- **Description** : L'administrateur peut ajouter de nouveaux administrateurs au système.

2. Modifier Admin

- **Description** : L'administrateur peut modifier les informations d'un administrateur existant.

3. Supprimer Admin

- **Description** : L'administrateur peut supprimer un administrateur du système.

Gérer Personnels

1. Ajouter Personnel

- **Description** : L'administrateur peut ajouter de nouveaux personnels au système.
2. **Modifier Personnel**
 - **Description** : L'administrateur peut modifier les informations d'un personnel existant.
 3. **Supprimer Personnel**
 - **Description** : L'administrateur peut supprimer un personnel du système.

Authentification

- L'administrateur demande à s'authentifier.
- Le système lui demande de saisir son email.
- L'administrateur saisit son email.
- Le système lui demande de saisir son mot de passe.
- L'administrateur saisit son mot de passe.
- Le système vérifie les informations d'identification fournies.
- Le système passe par un processus AntiSpam pour vérifier qu'il ne s'agit pas d'une tentative de spam.
- Si toutes les vérifications sont réussies, l'administrateur est authentifié et peut accéder au système.

Scénarios Alternatifs

1. **Email incorrect**
 - Si l'email est incorrect, le système informe l'administrateur et lui redemande de saisir les informations correctes.
2. **Mot de passe incorrect**
 - Si le mot de passe est incorrect, le système informe l'administrateur et lui redemande de saisir le mot de passe.
3. **Détection de spam**
 - Si le système détecte une tentative de spam, il bloque l'authentification de l'administrateur pendant 30 secondes avant de permettre une nouvelle tentative de connexion.

Conclusion

Ce cas d'utilisation montre un ensemble complet de fonctionnalités accessibles par l'administrateur, y compris la gestion de son propre profil, des utilisateurs, des demandes de maintenance, et des autres administrateurs et personnels, avec une vérification AntiSpam pour sécuriser le processus contre les tentatives de spam ou d'attaques par force brute.

1.3.2.1.3. Diagramme de cas d'utilisation - Utilisateur

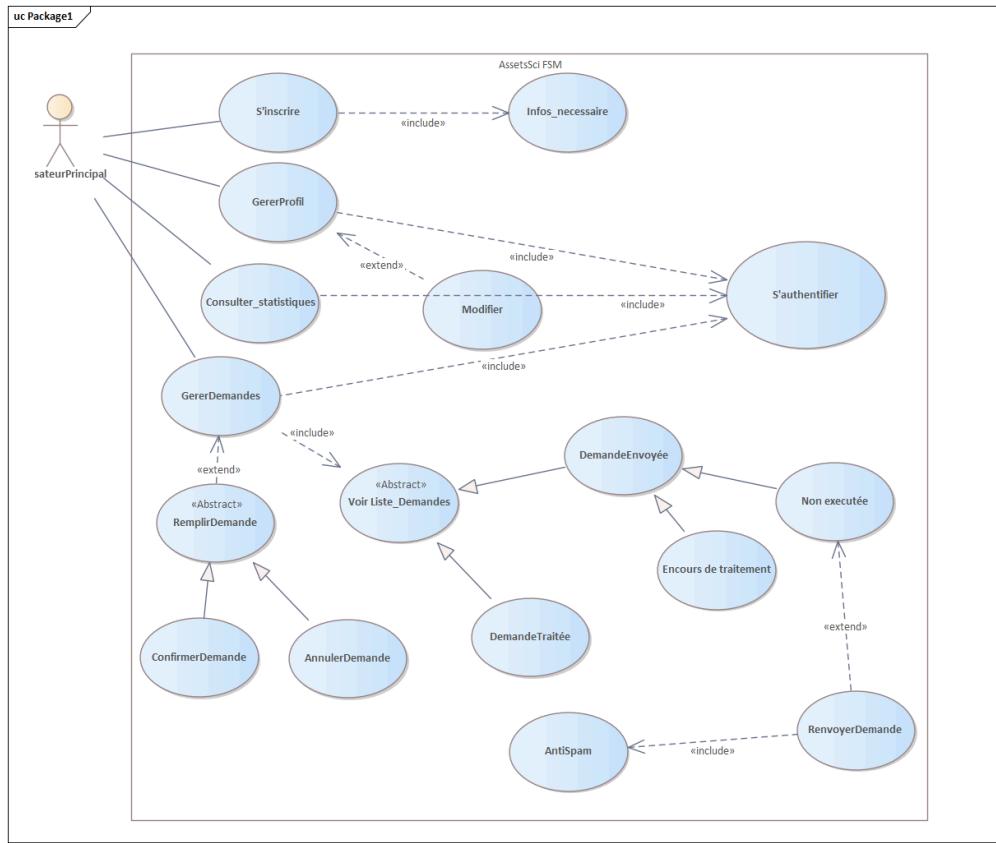


Figure 4 : UC Utilisateur

Utilisateur Principal

L'Utilisateur Principal est celui qui interagit avec le système pour différentes fonctions.

Cas d'Utilisation Principaux

1. **S'authentifier**
 - **Description:** L'utilisateur s'identifie pour accéder au système.
2. **S'inscrire**
 - **Description:** L'utilisateur s'enregistre dans le système en fournissant les informations requises.
 - **Inclut:** Fournir les informations nécessaires pour l'inscription.
3. **Gérer Profil**
 - **Description:** L'utilisateur gère son profil.
 - **Inclut:** Modifier les informations du profil.
4. **Consulter Statistiques**
 - **Description:** L'utilisateur consulte les statistiques disponibles dans le système.
5. **Gérer Demandes**
 - **Description:** L'utilisateur gère les demandes qu'il a soumises.

- **Sous-cas:**
 - Remplir Demande
 - Voir Liste Demandes
- **Inclut:** Voir les demandes envoyées, traitées, en cours de traitement .
- **Extend:** Renouveler Demande, Anti-Spam, Confirmer Demande, Annuler Demande.

Relations d'Inclusion et d'Extension

- **Inclusion de S'authentifier** dans plusieurs cas d'utilisation, indiquant que l'authentification est nécessaire pour accéder à ces fonctions.
- **Inclusion de Infos_necessaire** dans S'inscrire pour fournir les informations nécessaires.
- **Inclusion de Modifier** dans Gérer Profil pour permettre les modifications du profil.
- **Inclusion de Voir Liste Demandes** avec plusieurs sous-cas pour voir et gérer les demandes soumises.
- **Extension de Renouveler Demande** à Non exécutée pour permettre le renouvellement des demandes non exécutées.

Diagramme de Cas d'Utilisation

Le diagramme illustre les interactions entre l'Utilisateur Principal et les fonctionnalités du système, ainsi que les relations d'inclusion et d'extension entre ces fonctionnalités.

Conclusion

L'Utilisateur Principal peut effectuer diverses fonctions, de la gestion de son profil à la gestion des demandes soumises. Chaque fonction est décomposée en sous-cas d'utilisation détaillant les opérations spécifiques. Les relations d'inclusion et d'extension assurent une compréhension claire du flux de travail pour l'Utilisateur Principal.

1.3.2.2. Diagramme de séquence Inscrit/Auth :

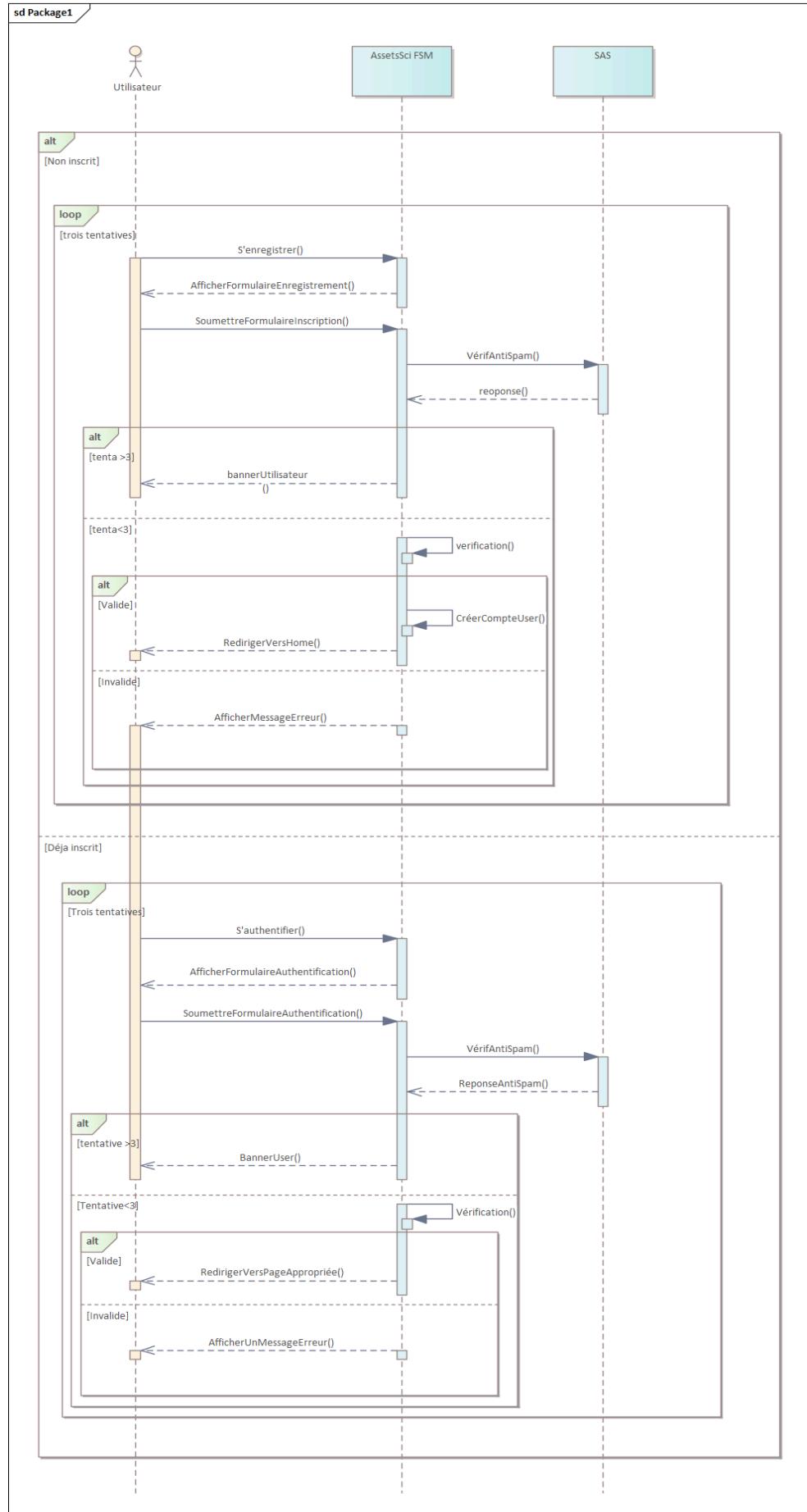


Figure 5 : Séquence Inscrit/Auth

Acteurs et Objets

- ★ Utilisateur : l'utilisateur qui interagit avec le système.
- ★ AssetSci FSM : Module de gestion des utilisateurs.
- ★ SAS : Système d'authentification anti-spam.

Scénario d'Inscription

Condition initiale: L'utilisateur n'est pas encore inscrit.

- **Boucle d'Inscription :** L'utilisateur a trois tentatives pour s'inscrire.
 - *S'enregistrer* : L'utilisateur déclenche le processus d'inscription.
 - *AfficherFormulaireEnregistrement()* : Affichage du formulaire d'inscription par le système.
 - *SoumettreFormulaireInscription()* : L'utilisateur soumet le formulaire.
 - *VérifAntiSpam()* : Envoi des informations au module anti-spam pour vérification.
 - *RéponseAntiSpam()* : Le système anti-spam renvoie une réponse sur la validité des informations.
 - **Conditions Alternatives :**
 - *Si Tentative >= 3* : Le système bloque l'utilisateur après trois tentatives infructueuses.
 - *Si Tentative < 3* :
 - *Vérification()* : Le système vérifie les informations soumises.
 - *Si valide* : Le compte utilisateur est créé et l'utilisateur est redirigé vers la page d'accueil.
 - *Si invalide* : Affichage d'un message d'erreur.

Scénario d'Authentification

Condition initiale: L'utilisateur est déjà inscrit.

1. **Boucle d'Authentification :** L'utilisateur a trois tentatives pour s'authentifier
 - *S'authentifier* : L'utilisateur déclenche le processus d'authentification.
 - *AfficherFormulaireAuthentification()* : Affichage du formulaire d'authentification par le système.
 - *SoumettreFormulaireAuthentification()* : L'utilisateur soumet le formulaire.
 - *VérifAntiSpam()* : Envoi des informations au module anti-spam pour vérification.
 - *RéponseAntiSpam()* : Le système anti-spam renvoie une réponse sur la validité des informations.

Conditions Alternatives :

- *Si Tentative >= 3* : Le système bloque l'utilisateur après trois tentatives infructueuses.

- *Si Tentative < 3 :*
 - *Vérification()* : Le système vérifie les informations soumises.
 - *Si valide* : Redirection de l'utilisateur vers la page appropriée.
 - *Si invalide* : Affichage d'un message d'erreur.

Conclusion

Ce diagramme de séquence illustre le processus d'enregistrement et d'authentification, mettant en évidence l'interaction entre l'utilisateur, le module de gestion des utilisateurs et le système de vérification du spam. Une boucle est utilisée pour gérer les tentatives répétées et utiliser une condition alternative pour bloquer l'utilisateur après trois tentatives infructueuses.

1.3.2.2 Diagramme de séquence-Admin

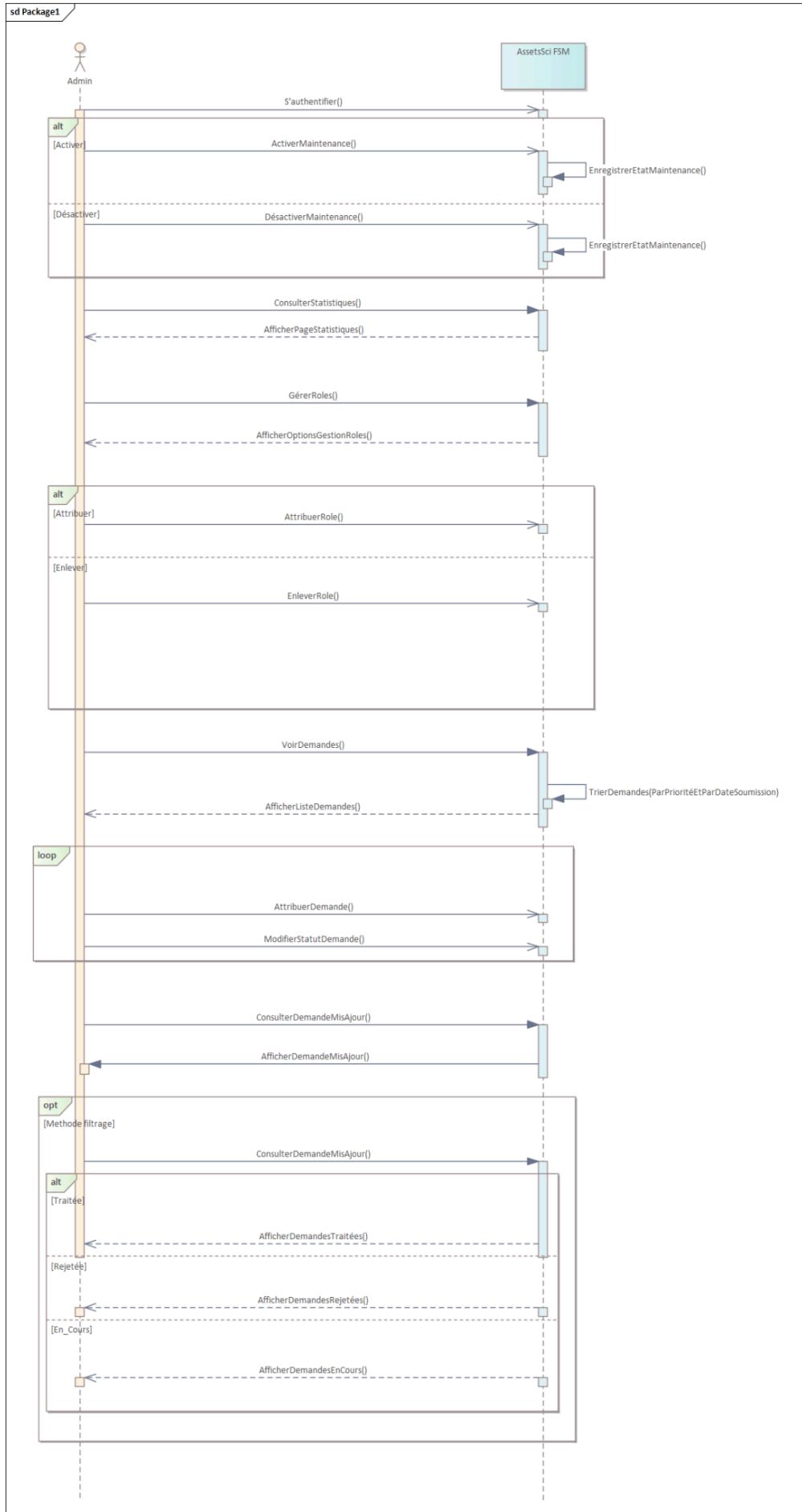


Figure 6 : Séquence Admin

Ce diagramme de séquence illustre l'interaction entre l'administrateur (acteur), le système de gestion des demandes (AssetsSci FSM) et les différentes opérations liées à la gestion des demandes et des rôles.

Acteurs et Objets

- ★ Administrateur : administrateur système.
- ★ AssetsSci FSM : Système de gestion des demandes.

Scénario de Gestion des Demandes par l'Administrateur

1. Authentification

- Admin -> AssetSci FSM : S'authentifier()
 - L'administrateur s'authentifie pour accéder au système.

2. Maintenance

● alt [Activer/Désactiver]

- Admin -> AssetsSci FSM : ActiverMaintenance()
 - L'administrateur active le mode maintenance.
- AssetsSci FSM -> Admin : EnregistrerEtatMaintenance()
 - Le système enregistre l'état de la maintenance.
- Admin -> AssetsSci FSM : DésactiverMaintenance()
 - L'administrateur désactive le mode maintenance.
- AssetsSci FSM -> Admin : EnregistrerEtatMaintenance()
 - Le système enregistre l'état de la maintenance

3. Statistiques

- Admin -> AssetsSci FSM : ConsulterStatistiques()
 - L'administrateur consulte les statistiques du système.
- AssetsSci FSM -> Admin : AfficherPageStatistiques()
 - Le système affiche la page des statistiques.

4. Gestion des Rôles

- Admin -> AssetsSci FSM : GérerRôles()
 - L'administrateur gère les rôles des utilisateurs.
- AssetsSci FSM -> Admin : AfficherOptionsGestionRoles()
 - Le système affiche les options de gestion des rôles.

5. Attribution et Retrait de Rôles

● alt [Attribuer/Enlever]

- Admin -> AssetsSci FSM : AttribuerRole()
 - L'administrateur attribue un rôle à un utilisateur ou administrateur.
- Admin -> AssetsSci FSM : EnleverRole()

- L'administrateur enlève un rôle à un utilisateur ou administrateur

6. Consultation des Demandes

- Admin -> AssetsSci FSM : VoirDemandes()
 - L'administrateur consulte la liste des demandes.
- AssetsSci FSM -> Admin : AfficherListeDemandes()
 - Le système trie les demandes par priorité et par date de soumission, puis affiche la liste des demandes.

7. Attribution et Modification de Demandes

- *loop*
 - Admin -> AssetsSci FSM : AttribuerDemande()
 - L'administrateur attribue une demande à un utilisateur.
 - Admin -> AssetsSci FSM : ModifierStatutDemande()
 - L'administrateur après modifie le statut de la demande.

8. Consultation de Demandes Mises à Jour

- Admin -> AssetsSci FSM : ConsulterDemandeMisAJour()
 - L'administrateur consulte les demandes mises à jour.
- AssetsSci FSM -> Admin : AfficherDemandeMisAJour()
 - Le système affiche les demandes mises à jour.

9. Filtrage des Demandes Mises à Jour

- **opt [Méthode de filtrage]**
 - Admin -> AssetsSci FSM : ConsulterDemandeMisAJour()
 - L'administrateur consulte les demandes mises à jour.
 - **alt [Traitée/Rejetée/En_Cours]**
 - AssetsSci FSM -> Admin : AfficherDemandesTraitée()
 - Le système affiche les demandes traitées.
 - AssetsSci FSM -> Admin : AfficherDemandesRejetée()
 - Le système affiche les demandes rejetées.
 - AssetsSci FSM -> Admin : AfficherDemandesEnCours()
 - Le système affiche les demandes en cours.

Conclusion

Ce diagramme de séquence montre comment un administrateur interagit avec le système de gestion des demandes (AssetsSci FSM) pour effectuer diverses tâches administratives, telles que l'activation et la désactivation du mode maintenance, la consultation des statistiques, la gestion des rôles, la consultation et l'attribution des demandes, ainsi que le filtrage et la visualisation des demandes mises à jour. Les différentes options et conditions alternatives sont clairement illustrées pour montrer les différentes possibilités d'interaction.

1.3.2.1.3. Diagramme de séquence- Utilisateur

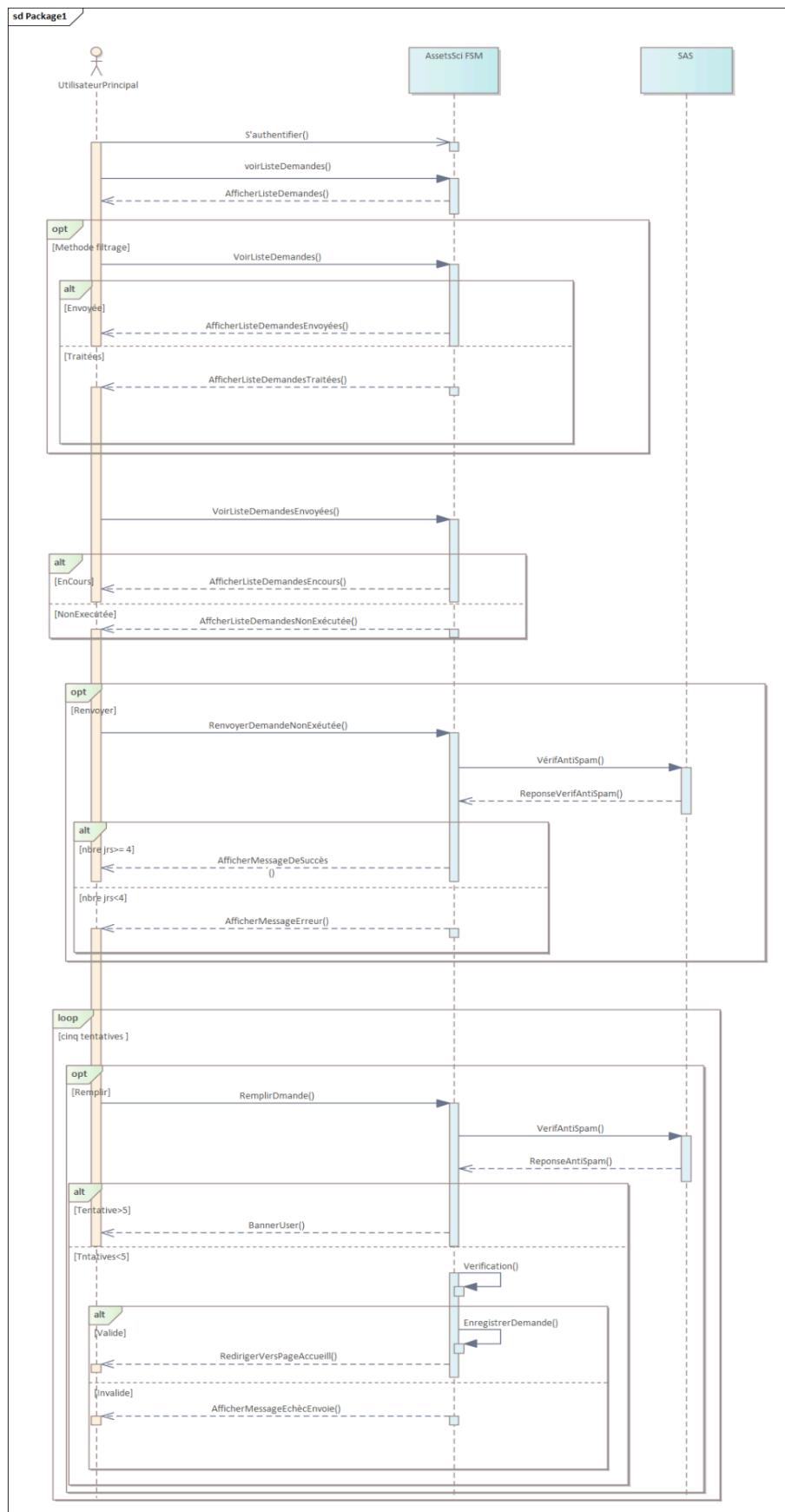


Figure 7 : Séquence Utilisateur

Le diagramme de séquence ci-joint présente l'interaction entre l'utilisateur principal, le système AssetSci FSM et le service Anti-Spam (SAS) pour la gestion des demandes des utilisateurs. Voici une explication détaillée des différentes interactions représentées dans le diagramme.

Acteurs

- **Utilisateur Principal**
- **AssetSci FSM** : Système principal de gestion.
- **SAS** : Service Anti-Spam.

Interactions Principales

1. **S'authentifier**
 - L'utilisateur principal s'authentifie auprès du système AssetSci FSM.
2. **Voir Liste Demandes**
 - Après authentification, l'utilisateur demande à voir la liste de ses demandes.
 - Le système AssetSci FSM affiche la liste des demandes.
3. **Voir Liste Demandes**
 - L'utilisateur peut appliquer des méthodes de filtrage pour voir les demandes envoyées ou traitées.
 - Selon l'état des demandes, le système affiche soit les demandes envoyées, soit les demandes traitées.

Option: Méthode de Filtrage

5. **Renvoi Demande Non Exécutée**
 - Si une demande n'a pas été exécutée, l'utilisateur peut choisir de la renvoyer.
 - Le système envoie la demande au service Anti-Spam pour vérification.
 - Si la vérification échoue(**nbre jrs <4**), un message d'erreur est affiché à l'utilisateur.
Sinon, un message de succès est affiché.

Remplir et Soumettre Demande

6. **Remplir Demande**
 - L'utilisateur remplit une nouvelle demande.
 - La demande est envoyée au service Anti-Spam pour vérification.
 - Si la demande échoue après cinq tentatives, l'utilisateur est banni pendant **une heure**.
Sinon, la demande passe à l'étape de vérification.

Vérification et Enregistrement

7. **Vérification et Enregistrement**
 - Une fois la demande vérifiée, le système procède à l'enregistrement de la demande.
 - Si la demande est valide, l'utilisateur est redirigé vers la page d'accueil. Si elle est invalide, un message d'échec est affiché.

Résumé des Scénarios de Séquence

- 1. Authentification et Affichage des Demandes**
 - L'utilisateur s'authentifie et consulte la liste des demandes envoyées ou traitées.
- 2. Gestion des Demandes Non Exécutées**
 - L'utilisateur renvoie les demandes non exécutées et passe par le service Anti-Spam pour vérification.
- 3. Crédit et Soumission de Demandes**
 - L'utilisateur crée, soumet et fait vérifier de nouvelles demandes. Selon les résultats de la vérification, les demandes sont enregistrées et des messages de statut sont affichés.

1.3.2.3. Diagramme de classes.

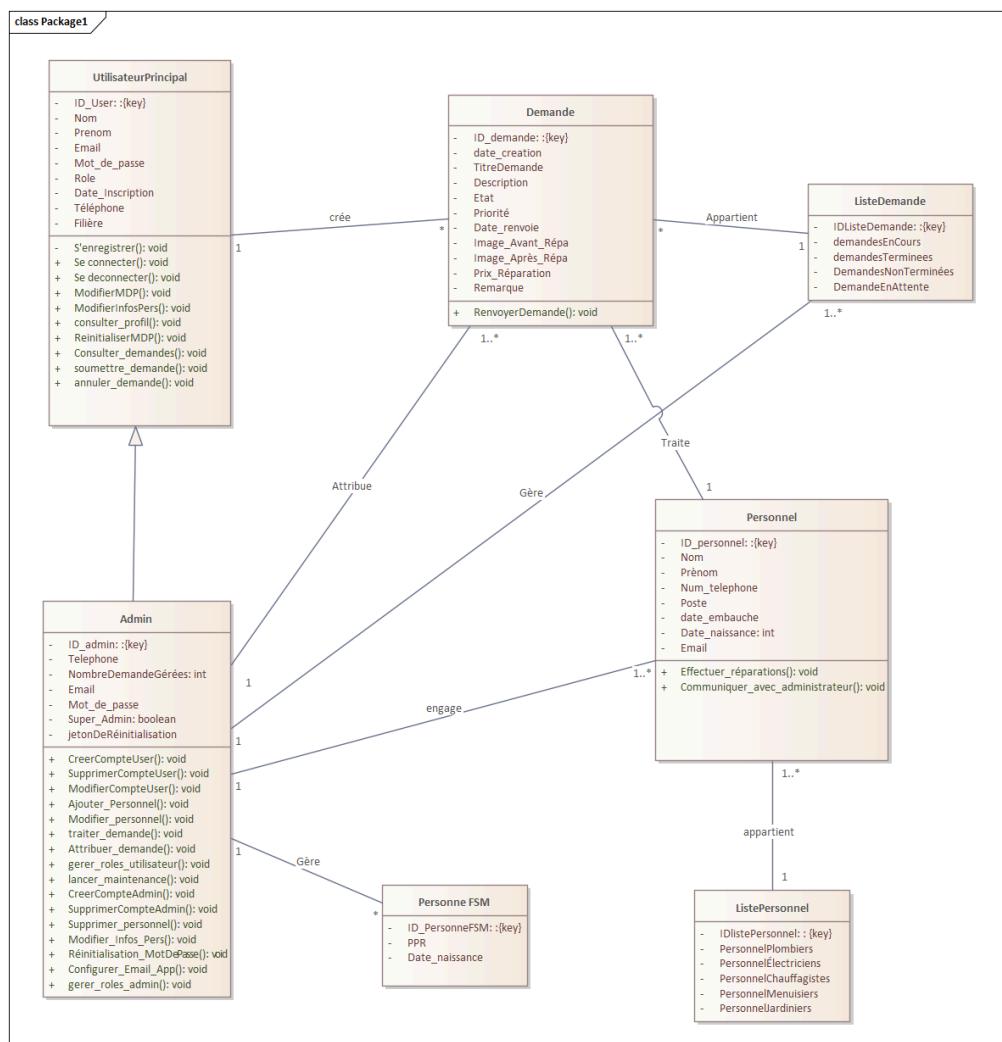


Figure 8 : Diagramme de classe

Ce diagramme de classes représente les entités et leurs relations dans un système de gestion des demandes, comprenant les utilisateurs principaux, les administrateurs, le personnel, les demandes, et les listes associées.

UtilisateurPrincipal

- **Attributs :**

- ID_User : {key}
- Nom
- Prenom
- Email
- Mot_de_passe
- Role
- Date_Inscription
- Téléphone
- Filière

Méthodes :

- S'enregistrer() : void
- Se connecter() : void
- Se déconnecter() : void
- ModifierMDP() : void
- ModifierInfosPers() : void
- consulter_profil() : void
- RéinitialiserMDP() : void
- Consulter_demandes() : void
- soumettre_demande() : void
- annuler_demande() : void

Admin

- **Attributs :**

- ID_admin : {key}
- Telephone
- NombreDemandesGérées : int
- Email
- Mot_de_passe
- Super_Admin : boolean
- jetonDeRéinitialisation

- **Méthodes :**

- CreerCompteUser() : void
- SupprimerCompteUser() : void
- ModifierCompteUser() : void
- Ajouter_Personnel() : void
- Modifier_personnel() : void
- traiter_demande() : void
- Attribuer_demande() : void
- gerer_roles_utilisateur() : void

- lancer_maintenance() : void
- CreerCompteAdmin() : void
- SupprimerCompteAdmin() : void
- ModifierCompteAdmin() : void
- Modifier_Infos_Pers() : void
- Réinitialisation_MotDePasse() : void
- Configurer_Email_App() : void
- gerer_roles_admin() : void

Demande

- **Attributs :**
 - ID_demande : {key}
 - date_creation
 - TitreDemande
 - Description
 - Etat
 - Priorité
 - Date_revue
 - Image_Avant_Répa
 - Image_Après_Répa
 - Prix_Réparation
 - Remarque
- **Méthodes :**
 - RenvoyerDemande() : void

ListeDemande

- **Attributs :**
 - IDListeDemande : {key}
 - demandesEnCours
 - demandesTerminées
 - demandesNonTerminées
 - DemandesEnAttente

Personnel

- **Attributs :**
 - ID_personnel : {key}
 - Nom
 - Prenom
 - Num_telephone
 - Poste
 - date_embauche
 - Date_naissance : int
 - Email
- **Méthodes :**
 - Effectuer_réparations() : void
 - Communiquer_avec_administrateur() : void

Personne FSM

- **Attributs :**
 - ID_PersonneFSM : {key}
 - PPR
 - Date_naissance

ListePersonnel

- **Attributs :**
 - IDListePersonnel : {key}
 - PersonnelPlombiers
 - PersonnelElectriciens
 - PersonnelChauffagistes
 - PersonnelMenuisiers
 - PersonnelJardiniers

Relations entre les Classes

- **UtilisateurPrincipal** crée des **Demandes**
- **Admin** attribue des **Demandes**
- **Admin** gère des **Personne FSM**
- **Demande** appartient à une **ListeDemande**
- **Personnel** traite des **Demandes**
- **Personnel** appartient à une **ListePersonnel**
- **Admin** engage des **Personnels**

Conclusion

Ce diagramme de classes modélise les entités et leurs relations dans un système de gestion des demandes. Les classes principales incluent les utilisateurs, les administrateurs, les demandes, le personnel, et les listes associées. Les relations entre ces classes définissent les rôles et les interactions dans le système, offrant une vue d'ensemble claire des responsabilités et des opérations dans ce contexte.

1.3.2.4. Diagrammes d'activités

1.3.2.4.1 Diagramme d'activités Inscrit/Auth

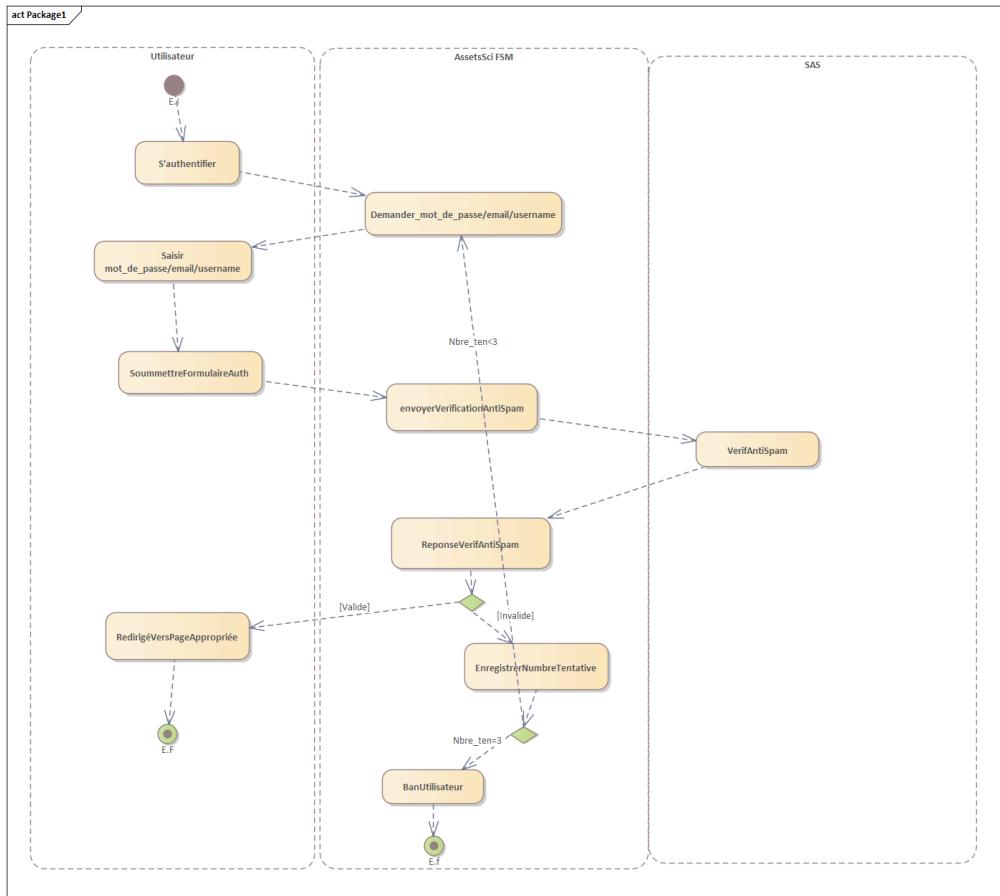


Figure 9 : Activité Inscrit/Auth

Le diagramme d'activité ci-joint illustre le processus d'authentification d'un utilisateur au sein du système AssetSci FSM, incluant une vérification AntiSpam pour améliorer la sécurité.

Acteurs

- **Utilisateur** : Initiateur du processus d'authentification.
- **AssetsSci FSM** : Système de gestion de l'authentification.
- **SAS** : Système AntiSpam.

Activités et État du Processus

1. **S'authentifier**
 - L'utilisateur initie le processus d'authentification.
2. **Saisir mot de passe/email**
 - L'utilisateur saisit son mot de passe et son email/username.
3. **Soumettre Formulaire d'Authentification**
 - L'utilisateur soumet le formulaire d'authentification contenant ses informations d'identification.

Demander mot de passe/email

- AssetsSci FSM reçoit la demande d'authentification et extrait le mot de passe et l'email fournis.

Envoyer Vérification AntiSpam

- AssetsSci FSM envoie les informations à vérifier par le système AntiSpam (SAS).

Vérifier AntiSpam

- Le SAS effectue la vérification pour s'assurer que l'authentification n'est pas une tentative de spam.
- **[Décision]** Nombre de tentatives < 3 :
 - Si oui, continuer la vérification.
 - Si non, interdire l'utilisateur.

Réponse Vérification AntiSpam

- Le SAS renvoie le résultat de la vérification au AssetsSci FSM

Enregistrer Nombre de Tentatives

- Si la vérification échoue, AssetsSci FSM enregistre une tentative d'authentification ratée.

Ban Utilisateur

- Si le nombre de tentatives dépasse trois, l'utilisateur est banni du système pour une durée définie.

Rediriger Vers Page Appropriée

- Si l'authentification est réussie, l'utilisateur est redirigé vers la page appropriée du système.
- Sinon, une notification d'échec d'authentification est affichée.

Scénarios de Base

Authentification Réussie

- L'utilisateur saisit son email et son mot de passe.
- Le formulaire est soumis.
- Le SAS vérifie les informations et les valide.
- L'utilisateur est redirigé vers la page appropriée.

Authentification Échouée

- L'utilisateur saisit des informations incorrectes.
- Le formulaire est soumis.
- Le SAS détecte des erreurs et enregistre la tentative.

- Si moins de trois tentatives, l'utilisateur peut réessayer.
- Si trois tentatives échouent, l'utilisateur est banni temporairement.

Scénarios Alternatifs

Tentative de Spam

- Le système détecte une tentative de spam après plusieurs tentatives infructueuses.
- L'utilisateur est banni du système pour une durée définie avant de pouvoir réessayer.

Le diagramme fournit une vue claire du processus d'authentification, mettant en évidence les étapes cruciales et les décisions prises pour assurer la sécurité du système.

1.3.2.4.2 Diagramme d'activités Admin

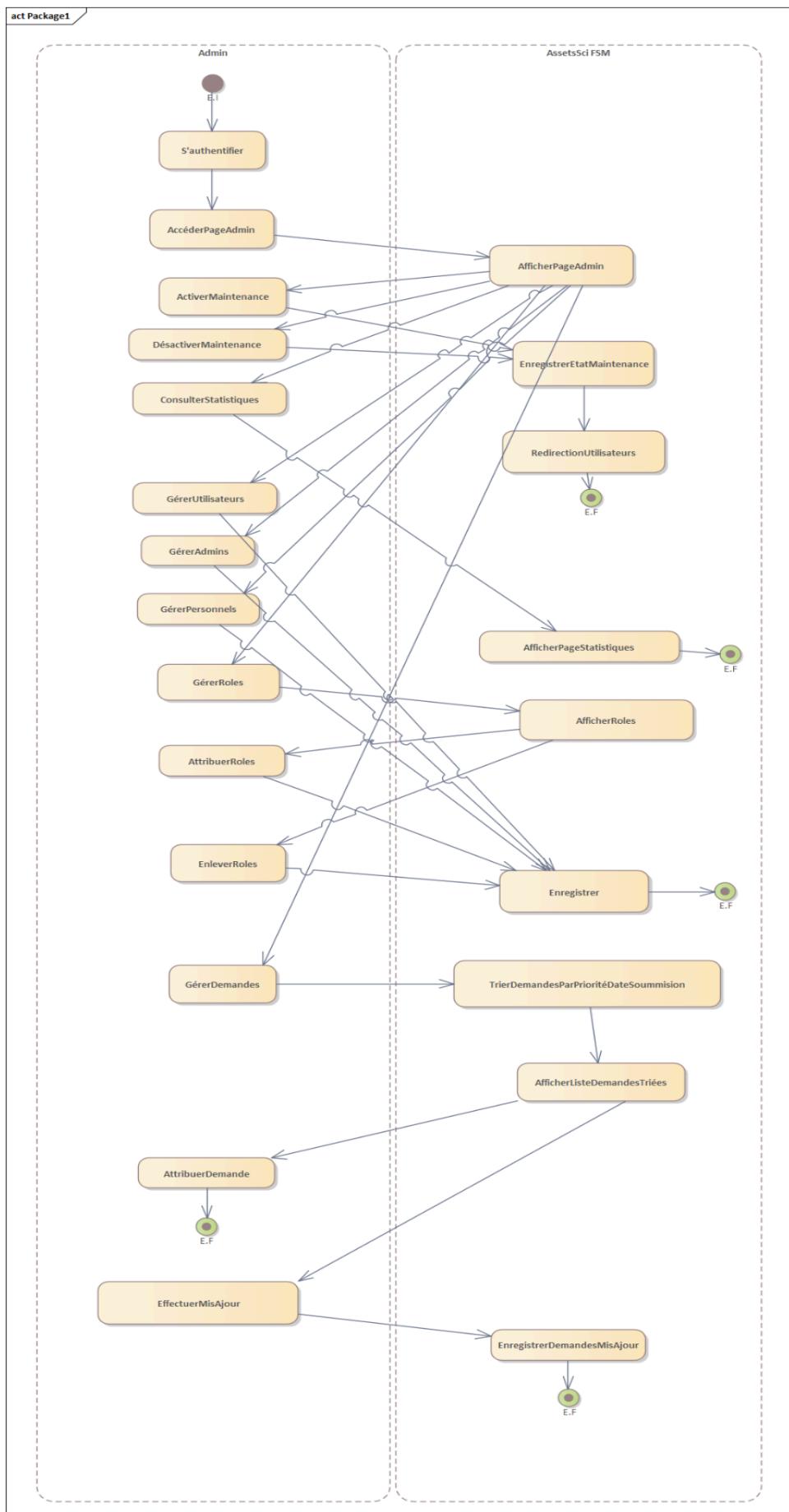


Figure 10 : Activité Admin

Le diagramme d'activité ci-joint décrit le processus de gestion des tâches administratives par un administrateur au sein du système AssetSci FSM. Ce processus inclut diverses activités telles que l'activation de la maintenance, la gestion des utilisateurs, et la gestion des rôles et des demandes.

Acteurs

- **Admin** : L'administrateur du système.

Activités et État du Processus

1. **S'authentifier**
 - L'administrateur initie le processus en s'authentifiant dans le système.
2. **Accéder Page Admin**
 - Après une authentification réussie, l'administrateur accède à la page d'administration.
3. **Afficher Page Admin**
 - La page d'administration est affichée, permettant l'accès à diverses fonctionnalités administratives.

Gestion de la Maintenance

4. **Activer Maintenance**
 - L'administrateur peut activer la maintenance du système.
 - **Enregistrer État Maintenance**
 - L'état de maintenance est enregistré.
 - **Redirection Utilisateurs**
 - Les utilisateurs sont redirigés selon l'état du système (en maintenance ou opérationnel).

Désactiver Maintenance

- L'administrateur peut désactiver la maintenance.

Consultation des Statistiques

6. **Consulter Statistiques**
 - L'administrateur consulte les statistiques du système.
 - **Afficher Page Statistiques**
 - Les statistiques sont affichées sur la page dédiée.

Gestion des Utilisateurs, Admins, et Personnels

7. **Gérer Utilisateurs**
 - L'administrateur peut gérer les profils des utilisateurs.
 - Activités incluent Ajouter, Modifier, et Supprimer des utilisateurs.

8. Gérer Admins

- L'administrateur peut gérer les autres administrateurs du système.
- Activités incluent Ajouter, Modifier, et Supprimer des administrateurs.

9. Gérer Personnels

- L'administrateur peut gérer le personnel associé au système.
- Activités incluent Ajouter, Modifier, et Supprimer du personnel.

Gestion des Rôles

10. Gérer Rôles

- L'administrateur peut gérer les rôles au sein du système.
- **Attribuer Rôles**
 - L'administrateur attribue des rôles aux utilisateurs.
- **Enlever Rôles**
 - L'administrateur retire des rôles aux utilisateurs.
- **Afficher Rôles**
 - Les rôles attribués sont affichés.

Gestion des Demandes

11. Gérer Demandes

- L'administrateur gère les demandes soumises par les utilisateurs.
- **Trier Demandes Par Priorité/Date Soumission**
 - Les demandes sont triées par priorité et date de soumission.
- **Afficher Liste Demandes Triées**
 - La liste des demandes triées est affichée.
- **Attribuer Demande**
 - La demande est attribuée à un personnel.
- **Effectuer Mise à Jour**
 - L'état de la demande est mis à jour après traitement.
- **Enregistrer Demandes Mis à Jour**
 - Les informations de la demande mise à jour sont enregistrées.

Scénarios de Base

1. Activation de la Maintenance

- L'administrateur s'authentifie, accède à la page d'administration, active la maintenance, et les utilisateurs sont redirigés.

2. Consultation des Statistiques

- L'administrateur consulte les statistiques du système via la page d'administration.

3. Gestion des Utilisateurs

- L'administrateur ajoute, modifie ou supprime des utilisateurs via la page d'administration.

4. Traitement des Demandes

- Le système trie les demandes, l'admin les attribue, et met à jour leur état après traitement.

Ce diagramme fournit une vue claire et détaillée des différentes activités que l'administrateur peut effectuer pour gérer efficacement le système AssetSci FSM.

[**1.3.2.4.3 Diagramme d'activités utilisateur**](#)

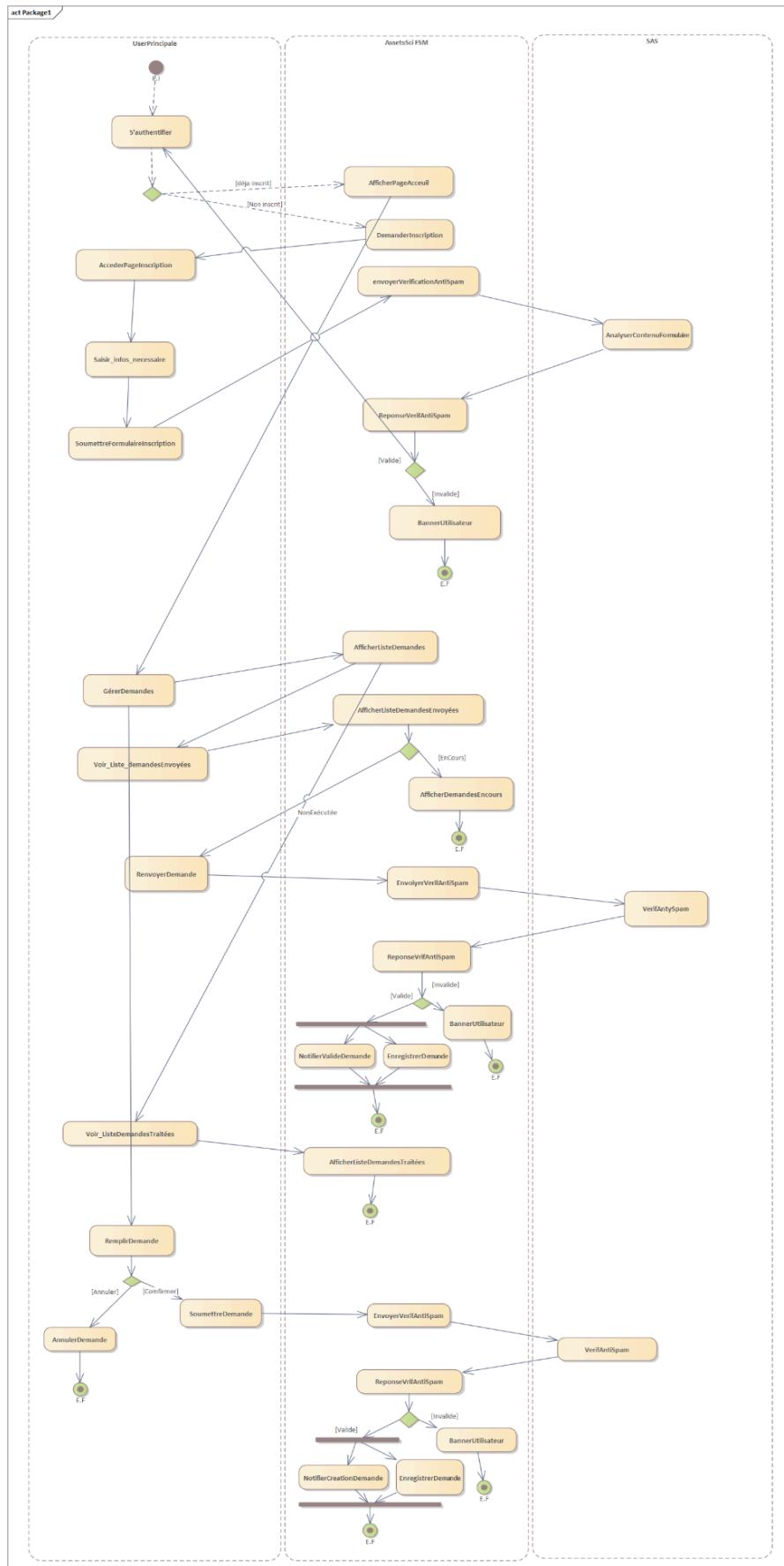


Figure 11 : Activité Utilisateur

Le diagramme d'activité ci-joint présente le processus de gestion des demandes des utilisateurs au sein du système AssetSci FSM. Ce diagramme détaille les étapes impliquées, de l'authentification de l'utilisateur à la gestion et au traitement des demandes.

Acteurs

- **UserPrincipal** : L'utilisateur principal du système.
- **AssetSci FSM** : Le système principal où les actions sont effectuées.
- **SAS** : Service Anti-Spam.

Activités et Processus

1. **S'authentifier**
 - L'utilisateur principal commence par s'authentifier dans le système.
2. **[déjà inscrit] / [non inscrit]**
 - Si l'utilisateur est déjà inscrit, il accède directement à la page d'accueil.
 - Si l'utilisateur n'est pas inscrit, il doit passer par le processus d'inscription.

Processus d'Inscription

3. **Accéder Page Inscription**
 - L'utilisateur accède à la page d'inscription.
4. **Saisir Infos Nécessaire**
 - L'utilisateur entre les informations nécessaires pour l'inscription.
5. **Soumettre Formulaire Inscription**
 - Le formulaire d'inscription est soumis pour vérification.
6. **Envoyer Vérification Anti-Spam**
 - Une vérification anti-spam est envoyée.
7. **Réponse Vérif Anti-Spam**
 - Si la vérification est valide, l'utilisateur est inscrit avec succès.
 - Si la vérification échoue, l'utilisateur est banni.

Gestion des Demandes

8. **Gérer Demandes**
 - Une fois authentifié, l'utilisateur peut gérer ses demandes.
9. **Voir Liste Demandes Envoyées**
 - L'utilisateur peut voir la liste des demandes qu'il a envoyées.
10. **Envoyer Demande**
 - Pour envoyer une nouvelle demande, l'utilisateur doit remplir un formulaire et le soumettre.
11. **Envoyer Vérif Anti-Spam**
 - Chaque demande envoyée passe par une vérification anti-spam.
12. **Réponse Vérif Anti-Spam**
 - Si la vérification échoue, l'utilisateur est banni.
 - Si la vérification est valide, la demande est enregistrée.

Suivi et Traitement des Demandes

- 13. Afficher Liste Demandes**
 - L'utilisateur peut voir la liste de toutes les demandes envoyées.
- 14. Afficher Liste Demandes Traitées**
 - L'utilisateur peut voir la liste des demandes qui ont été traitées.
- 15. Remplir Demande**
 - L'utilisateur peut remplir des demandes spécifiques.
- 16. Soumettre Demande**
 - La demande remplie est soumise pour traitement.
- 17. Annuler Demande**
 - L'utilisateur peut annuler une demande avant soumission.

Notifications et Mise à Jour

- 18. Notifier Validation Demande**
 - L'utilisateur est notifié de la validation de sa demande.
- 19. Enregistrer Demande**
 - Les demandes validées sont enregistrées dans le système.
- 20. Afficher Demandes en Cours**
 - L'utilisateur peut voir les demandes actuellement en cours de traitement.

Scénarios de Base

- 1. Inscription et Authentification**
 - Un utilisateur non inscrit s'inscrit, passe par une vérification anti-spam, et accède au système après validation.
- 2. Envoi et Gestion de Demandes**
 - Un utilisateur authentifié envoie une demande, qui passe par une vérification anti-spam, et est notifié de son statut.
- 3. Suivi et Traitement des Demandes**
 - L'utilisateur consulte les demandes envoyées, en cours, et traitées. Il peut également annuler ou soumettre de nouvelles demandes.

Ce diagramme offre une vue détaillée et structurée des différentes activités et états que traverse un utilisateur lors de la gestion de ses demandes au sein du système AssetSci FSM

2. Chapitre 2 : Modèle de données

2.1 Introduction

Ce chapitre décrit le modèle de données utilisé dans notre projet, mettant l'accent sur les technologies employées pour gérer et structurer les données. Nous avons opté pour MySQL(**MySQL Workbench**) comme système de gestion de base de données et Spring Boot pour le développement backend. Nous aborderons également la méthode utilisée pour stocker et récupérer les images en base64, en particulier pour l'entitéte **Demande**.

2.2 Choix des Technologies

MySQL :

- **Justification** : MySQL est un SGBDR populaire pour sa performance et sa fiabilité. Il est capable de gérer de grandes quantités de données tout en offrant des transactions robustes et une intégrité des données.
- **Avantages** : MySQL offre des fonctionnalités telles que les index, les clés étrangères et les transactions, ce qui le rend adapté pour des applications nécessitant une gestion complexe des données.

Spring Boot :

- **Justification** : Spring Boot est un framework Java qui facilite le développement d'applications autonomes de production prêtes à l'emploi. Il simplifie la configuration et le déploiement des applications Java.
- **Avantages** : Avec Spring Boot, nous bénéficions d'une configuration par défaut qui suit les bonnes pratiques, d'une intégration facile avec les bases de données via Spring Data JPA, et d'une architecture modulaire et évolutive.

2.3 Intégration entre MySQL et Spring Boot

L'intégration entre Spring Boot et MySQL est réalisée via Spring Data JPA, qui permet de mapper les classes Java aux tables de la base de données sans nécessiter de code SQL explicite pour les opérations courantes. Voici comment cela est mis en œuvre :

Configuration de la Base de Données

Dans le fichier **application.properties** de Spring Boot, les paramètres de connexion à la base de données MySQL sont spécifiés.

```
spring.application.name=AssetsSci
spring.datasource.url=jdbc:mysql://localhost:3306/users-db?useSSL=false&serverTimezone=UTC&allowPublicKeyRetrieval=true
spring.datasource.username=root
spring.datasource.password=UMI-FSP@$$w0rd
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
```

Entités JPA

Les entités JPA représentent les tables de la base de données sous forme de classes Java. Par exemple, les entités Admin,Demande pourraient être définies comme suit :

```
public class Demande {  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Long id;  
    private String title;  
    private String description;  
    @Column(name = "created_at")  
    private Date createdAt;  
    private Date dateResubmission;  
    private DemandeStatus status;  
    private DemandePriority priority;  
    private Double prixReparation;  
    private String file_imageBefore;  
    private String file_imageAfter;  
    private String remarque;  
    @ManyToOne  
    private User user;  
    @ManyToOne  
    private Personnel personnel;  
}
```

```

33 usages
@Entity
@NoArgsConstructor
@AllArgsConstructor
@ToString
@Builder
@Getter
@Setter
@Table(name="admin")
public class Admin {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(unique = true, nullable = false)
    private String email;
    @Column(nullable = false)
    private String password;
    @Column(nullable = false)
    private boolean superAdmin;
    private String resetToken;
}

```

Encodage des images en base64 :

Pour stocker des images dans la base de données, nous les avons encodées en base64. Cette méthode permet de convertir les images en chaînes de caractères, facilitant ainsi leur stockage dans des champs de type **String**.

- Voici comment ces classes sont représentées dans la base de données MySQL Workbench:

The screenshot shows the MySQL Workbench interface. At the top, there is a toolbar with various icons. Below it is a 'Query 1' tab. The query editor contains the following SQL code:

```

1 • use `users-db`;
2 • select * from admin
3 ;
4

```

Below the query editor is a 'Result Grid' table. The table has columns labeled 'id', 'email', 'password', 'reset_token', and 'super_admin'. There is one row of data:

	id	email	password	reset_token	super_admin
1	1	pfegestionpatrimoine@gmail.com	pfeGestionpatrimoineFSM	1627cf4f-779b-4bcd-be92-bc84092a9981	1

Figure 12 : Table Admin

id	created_at	date_resubmission	description	file_image_after	file_image_before	priorit	prix_re	remarq	status
7	2024-05-23 11:07:08.195000	NULL	Demande d'installation de caméras d...	NULL	NULL	1	NULL	NULL	2
8	2024-05-23 11:07:08.306000	NULL	Demande d'installation de caméras d...	NULL	NULL	1	NULL	NULL	2
9	2024-05-23 11:24:44.010000	NULL	Description par défaut	data:image/jpeg;base64...	0	NULL	tres ...	0	
10	2024-05-23 12:14:15.271000	NULL	ampoule defectueuse [local block6:4]	data:image/jpeg;base64...	0	NULL	tres ...	0	

Figure 13 : Table Demande

Repositories Spring Data JPA

Les interfaces de repository étendent [JpaRepository](#), permettant d'effectuer des opérations CRUD sans écrire de code SQL. Par exemple, un repository pour une entité [Admin](#) pourrait ressembler à ceci :

```
5 usages
public interface AdminRepository extends JpaRepository<Admin, Long> {
    1 usage
    boolean existsByEmail(String email);
    2 usages
    Optional<Admin> findByEmail(String email);
    ⚡ 1 usage
    Optional<Admin> findByResetToken(String resetToken);
    1 usage
    @Query("SELECT COUNT(a) FROM Admin a WHERE a.superAdmin = true")
    long countSuperAdmins();
}
```

Services

Les services encapsulent la logique métier et utilisent les repositories pour accéder aux données. Un exemple de service pour gérer les admins:

```
@PostMapping("/createAdmin")
public ResponseEntity<Admin> createAdmin(@RequestBody Admin admin) throws MessagingException {
    if (adminRepository.existsByEmail(admin.getEmail())) {
        return ResponseEntity.status(HttpStatus.CONFLICT).body(null);
    }
    Admin savedAdmin = adminRepository.save(admin);
    sendAdminCreationEmail(savedAdmin, admin.getPassword());
    return ResponseEntity.status(HttpStatus.CREATED).body(savedAdmin);
}
```

```

    @DeleteMapping("deleteAdmin/{id}")
    public ResponseEntity<Void> deleteAdmin(@PathVariable Long id) {
        Optional<Admin> optionalAdmin = adminRepository.findById(id);
        if (optionalAdmin.isEmpty()) {
            return ResponseEntity.notFound().build();
        }
        adminRepository.deleteById(id);
        return ResponseEntity.noContent().build();
    }

```

```

    @PutMapping("/updateAdmin/{id}")
    public ResponseEntity<Admin> updateAdmin(@PathVariable Long id,
                                              @RequestBody Admin updatedAdmin)
        throws MessagingException {
        Optional<Admin> optionalAdmin = adminRepository.findById(id);
        if (optionalAdmin.isEmpty()) {
            return ResponseEntity.notFound().build();
        }
        Admin admin = optionalAdmin.get();
        admin.setEmail(updatedAdmin.getEmail());
        admin.setPassword(updatedAdmin.getPassword());
        admin.setSuperAdmin(updatedAdmin.isSuperAdmin());
        Admin savedAdmin = adminRepository.save(admin);
        sendAdminUpdateEmail(savedAdmin);
        return ResponseEntity.ok(savedAdmin);
    }

```

2.4 Conclusion

Le modèle de données dans notre projet est structuré de manière efficace grâce à l'utilisation de MySQL pour la gestion des données et de Spring Boot pour le développement backend. Cette approche permet de garantir une performance optimale, une intégrité des données, et une évolutivité de l'application. Le couplage de Spring Boot avec MySQL via Spring Data JPA simplifie les opérations CRUD et la gestion des transactions, tout en offrant une grande flexibilité pour le développement de nouvelles fonctionnalités.

3. Chapitre 3 : Frontend avec Angular

3.1 Introduction

Ce chapitre traite la partie frontend de notre projet, développée avec le framework Angular. Nous explorerons les principales raisons du choix d'Angular, les concepts clés, ainsi que la structure et l'implémentation de notre application Angular. Des captures d'écran illustreront les différentes interfaces de l'application.

3.2 Choix d'Angular

Justification :

- **Modularité** : Angular permet de structurer l'application en modules, rendant le code plus maintenable et réutilisable.
- **Performance** : Grâce à son moteur de rendu, Angular assure des performances optimales pour les applications web dynamiques.
- **Outils et Écosystème** : Angular CLI (Command Line Interface) facilite la création, la gestion et le déploiement de projets Angular. De plus, Angular dispose d'un vaste écosystème de bibliothèques et d'extensions.
- **TypeScript** : Angular est construit en TypeScript, un surensemble de JavaScript qui apporte des fonctionnalités de typage statique, améliorant ainsi la qualité du code et la productivité des développeurs.

3.3 Concepts Clés d'Angular

1. Composants :

- Les composants sont les blocs de construction de base d'une application Angular. Chaque composant contrôle une partie de l'interface utilisateur (UI).

8.3.1 Exemple du composant:

```
export class ForgotPasswordComponent {
  public forgotPasswordForm: FormGroup;
  no usages
  constructor(
    private authService: AuthService,
    private router: Router,
    private fb: FormBuilder,
    private _snackBar: MatSnackBar
  ) {
    this.forgotPasswordForm = this.fb.group( controls: {
      email: ['', [Validators.required, Validators.email]]
    });
  }
}
```

```

    submitForgotPasswordForm(): void {
      if (this.forgotPasswordForm.valid) {
        const email = this.forgotPasswordForm.value.email;
        this.authService.resetPassword(email).subscribe(
          next: (response: any) => {
            if (response && response.message) {
              this._snackBar.open(response.message, action: 'Fermer', config: {
                duration: 5000,
              });
              this.router.navigate(commands: ['/login']);
            } else {
              console.error('Erreur lors de l\'envoi de la demande de réinitialisation de mot de passe :', response);
            }
          },
          error: (error: any) => {
            // Gestion des erreurs HTTP
            console.error('Erreur lors de l\'envoi de la demande de réinitialisation de mot de passe :', error);
            this._snackBar.open( message: "Adresse e-mail non trouvée.", action: 'Fermer', config: {
              duration: 5000,
            });
          }
        );
      }
    }

  1+ usages
  getErrorMessage(fieldName: string, error: ValidationErrors): string {
    if (error['required']) {
      return `${fieldName} est requis`;
    } else if (error['email']) {
      return `Adresse email invalide`;
    } else {
      return "";
    }
  }
}

```

2. Services et Injection de Dépendances :

- Les services sont utilisés pour partager des données ou des fonctionnalités entre différentes parties de l'application.

8.3.2 Exemple de service:

```

@Injectable({
  providedIn: 'root'
})
export class PersonneFSMService{
  private baseUrl : string = 'http://172.30.70.16:8082';

  no usages
  constructor(private http: HttpClient) {
  }

  1+ usages
  createUser( personneFSM:PersonneFSM) : Observable<object>{
    return this.http.post(`url: ${this.baseUrl}/addPersonneFsm` , personneFSM)
  }
}

```

4. Routage :

- Angular fournit un module de routage qui permet de naviguer entre différentes vues ou composants.

8.3.3 Exemple de routage :

```

const routes: Routes = [
  {path : "",component: LoginComponent, canActivate: [MaintenanceGuard]},
  {path : "login",component: LoginComponent, canActivate: [MaintenanceGuard]},
  {path:"signup",component:SignupComponent, canActivate: [MaintenanceGuard]},
  {path : "user",component: UserTemplateComponent ,canActivate : [AuthGuard, MaintenanceGuard], children : [
    {path : "home",component: HomeComponent, canActivate: [MaintenanceGuard]},
    {path : "profile",component: ProfileComponent, canActivate: [MaintenanceGuard]},
    {path : "add-request",component: AddRequestComponent, canActivate: [MaintenanceGuard]},
    {path : "current-requests",component: CurrentRequestsComponent, canActivate: [MaintenanceGuard]},
    {path : "dashboard",component: DashboardComponent, canActivate: [MaintenanceGuard]},
    {path : "requests-completed",component: RequestsCompletedComponent, canActivate: [MaintenanceGuard]},
  ]},
  { path: 'maintenance' , component: MaintenanceComponent }
];

```

3.4 Intégration avec le Backend Spring Boot

Pour que l'application Angular communique avec le backend Spring Boot, nous utilisons le module HttpClient d'Angular.

8.3.4 Exemple de service Angular pour consommer une API REST :

```


export class UserService {
    private baseUrl : string = 'http://172.30.70.16:8082';

    no usages
    constructor(private httpClient: HttpClient) { }

    1+ usages
    addUsers( user : UserDTO ) : Observable<object>{
        return this.httpClient.post( url: `${this.baseUrl}/ADMIN/addUsers` , user)
    }
    1+ usages
    getUserId(userId: string) : Observable<Object> {
        const url : string = `${this.baseUrl}/usersById/${userId}`;
        return this.httpClient.get(url);
    }
}


```

3.5 Interface Admin:

Voici quelques captures d'écran illustrant les différentes parties de l'application Angular :

8.5.1 Page de Connexion :

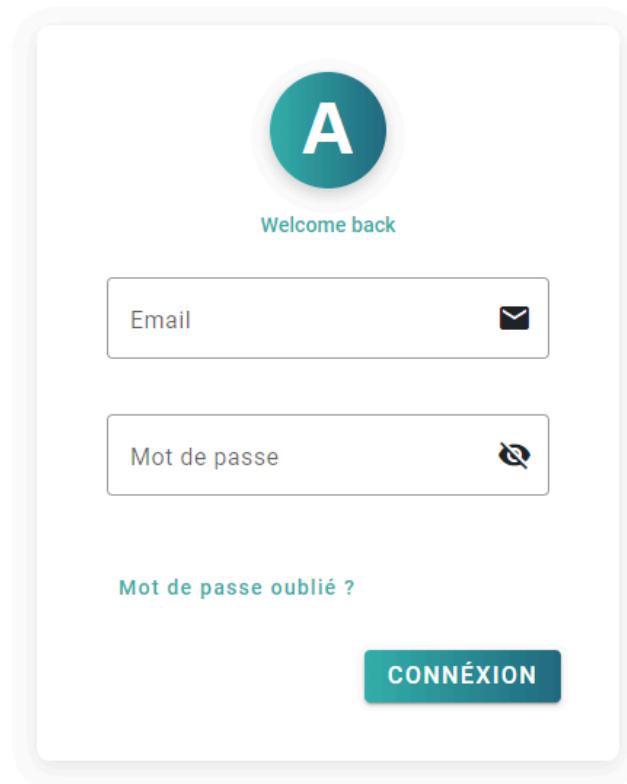


Figure 14 : Page de connexion

3.5.2 Page d'accueil:

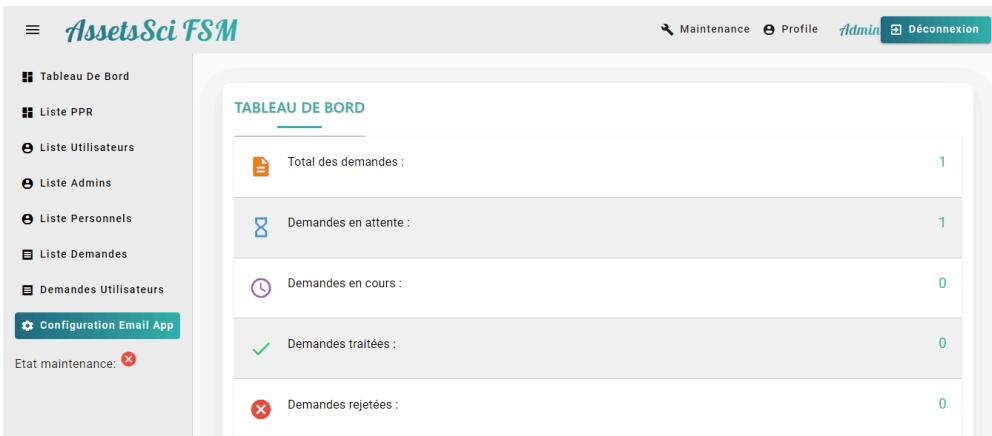


Figure 15 : Page d'accueil (super_admin)

- Lorsqu'un administrateur est authentifié en tant que super administrateur, il a la capacité de gérer entièrement le site. Cela inclut la possibilité de lancer la maintenance (activer/désactiver) et de gérer les autres administrateurs.
- En revanche, si un administrateur n'est pas un super administrateur, il peut effectuer toutes les tâches administratives sauf lancer la maintenance et gérer les autres administrateurs. Dans ce cas :
 - Le bouton de maintenance ne s'affiche pas.
 - La page de la liste des administrateurs ne lui est pas accessible.

Cette image montre les différences d'interface entre un super administrateur et un administrateur standard, notamment l'affichage des boutons de maintenance et l'accès à la page de la liste des administrateurs.



Figure 16 : Page d'accueil (!super_admin)

- L'accès à la liste des admins pour les administrateurs qui n'ont pas un rôle super admin

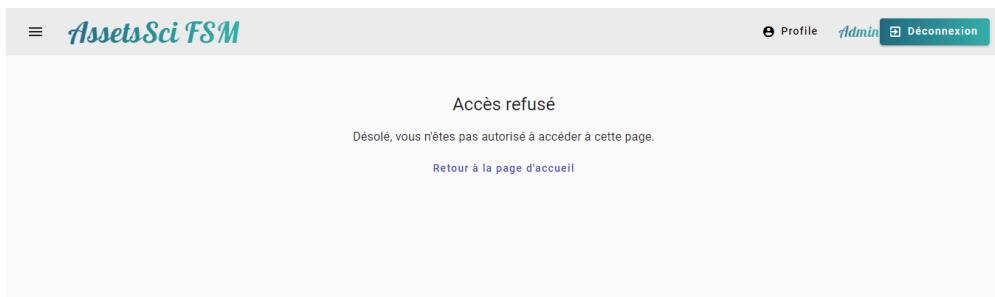


Figure 17 : Accès refusé

3.5.3 Page gestion des admins:

LISTE DES ADMINS					Ajouter Admin
ID	Email	Mot de passe	Super admin	Actions	
1	pfegestionpatrimoine@gmail.com	pfeGestionpatrimoineFSM	true		
2	Mohammed_123@gmail.com	Mohammed	false		
Items per page: <select>5</select> 1 - 2 of 2 < < > >					

Figure 18 : Page gestion des admins

3.5.4 Exemple traitement d'une demande:

Modifier Demande

ID 3	Nom Complet chaimae_chaimae
Email chaimae_123@gmail.co	Filière smi
Role ÉTUDIANT	Téléphone 0612364596
Titre Ampoule défectueuse	Description ampoule defectueuse [local block6:4]
Avant Reparation	Priorité URGENTE

The screenshot shows a user interface for managing a request. At the top left is a dropdown menu labeled "Statut" with "EN_ATTENTE" selected. To its right is a section for "Après Reparation" which includes a "Choisir un fichier" button and a message "Aucun fi". Below these are two input fields: one for "Remarque" (Remarks) and another for "Save".

Figure 19 : Formulaire traitement d'une demande

Ce formulaire permet de traiter la demande reçue par l'utilisateur chaimae_chaimae. L'administrateur peut :

- Modifier le statut de la demande (EN_COURS, TRAITÉE ou REJETÉE).
- Insérer une image après réparation pour confirmer le traitement, si la demande inclut une image avant réparation.(cas général)
- Ajouter des remarques pertinentes.

3.5.5 Configuration email de l'application

This screenshot shows a configuration page for application emails. It features a header "Configuration email de l'application". Below it are two input fields: "Email*" containing "pfegestionpatrimoine@gmail.com" and "Clé publique*" containing "mhac spur yjej gmik". A blue "Enregistrer" button is at the bottom.

Figure 20 : Formulaire de configuration de l'email de l'application

Ce formulaire permet de modifier les paramètres d'email de l'application, notamment l'adresse email et la clé publique. Il est conçu pour être utilisé lorsque l'email initialement configuré devient non fonctionnel ou doit être changé pour d'autres raisons.

Champs du Formulaire :

- **Email :** Ce champ permet de saisir une nouvelle adresse email qui sera utilisée par l'application pour envoyer des messages. Il est essentiel de fournir une adresse email valide pour garantir la bonne réception et l'envoi des emails.
- **Clé publique :** Ce champ est destiné à la saisie de la clé publique associée à l'email. La clé publique est utilisée à des fins de sécurité, permettant de chiffrer les communications ou d'authentifier les emails envoyés.

Fonctionnalité :

En fournissant un moyen simple et sécurisé de mettre à jour l'email et la clé publique, ce formulaire assure que l'application peut continuer à envoyer des emails sans interruption, même si l'adresse email originale devient obsolète ou invalide. Cela garantit une meilleure gestion des communications et améliore la fiabilité du système de messagerie de l'application.

3.6 Interface utilisateur

L'interface utilisateur inclut plusieurs fonctionnalités essentielles, telles que :

3.6.1 Page d'inscription

Pour que l'utilisateur puisse s'inscrire, il doit tout d'abord passer par une vérification de son PPR et de sa date de naissance afin de pouvoir accéder à la page d'inscription et la remplir. Voici à quoi ressemble la page de vérification :

Figure 21 : Page de vérification

Après avoir vérifié les informations, il sera redirigé vers la page d'inscription :

AssetsSci

Créer un Compte

Nom d'utilisateur*

Prénom* Nom* Rôle*

Numéro de téléphone* Email*

Filière* Code*
1234

Mot de passe* Confirmez le Mot de Pass

Enregistrer

< Retour à la Connexion

Figure 22 : Page d'inscription

3.6.2 Page de connexion

U Authentication

Username*

Mot de passe*

Login

Créer un compte

Mot de passe oublié ?

Figure 23 : Page de connexion

3.6.3 Page d'accueil:

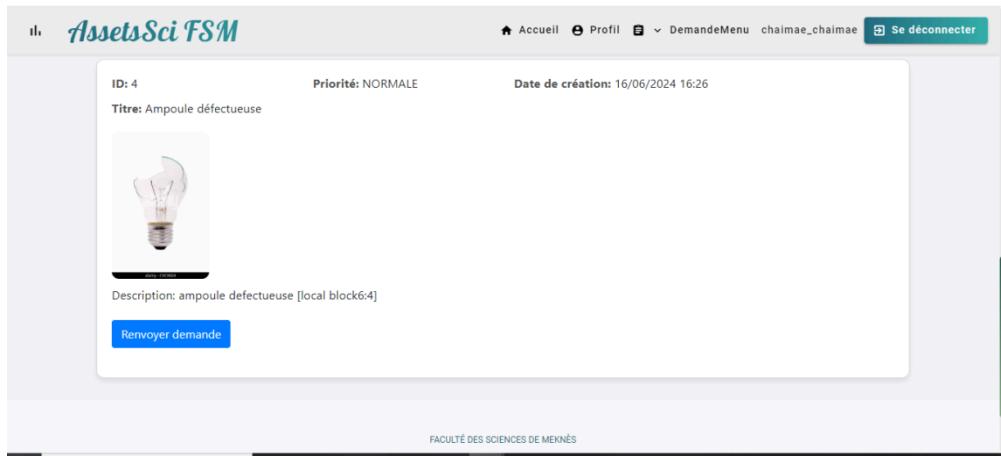


Figure 24 :Page d'accueil

- Les demandes créées ont un statut par défaut “EN_ATTENTE” et sont accessibles sur la page d'accueil.
- Ce bouton(**Renvoyer demande**) permet à un utilisateur de renvoyer une demande.
- La possibilité de renvoyer une demande est conditionnée par le fait que 4 jours se soient écoulés depuis la création de la demande.
- Lorsqu'un utilisateur tente de cliquer sur le bouton "Renvoyer Demande", le système vérifie la date de création de la demande.
- Si plus de 4 jours se sont écoulés depuis la création de la demande, le bouton est actif et permet l'envoi.
- Sinon, le bouton est désactivé et un message s'affiche à l'utilisateur indiquant qu'il ne peut pas renvoyer la demande avant que 4 jours ne se soient écoulés.
- les demandes avec un statut “EN_COURS” sont accessibles sur la page des demandes en cours ,Une fois traitées, elles seront accessibles sur la page des demandes achevées.

3.6.4 Page demandes achevées

- Les demandes avec un statut “TRAITÉE” ressemblent à ceci :

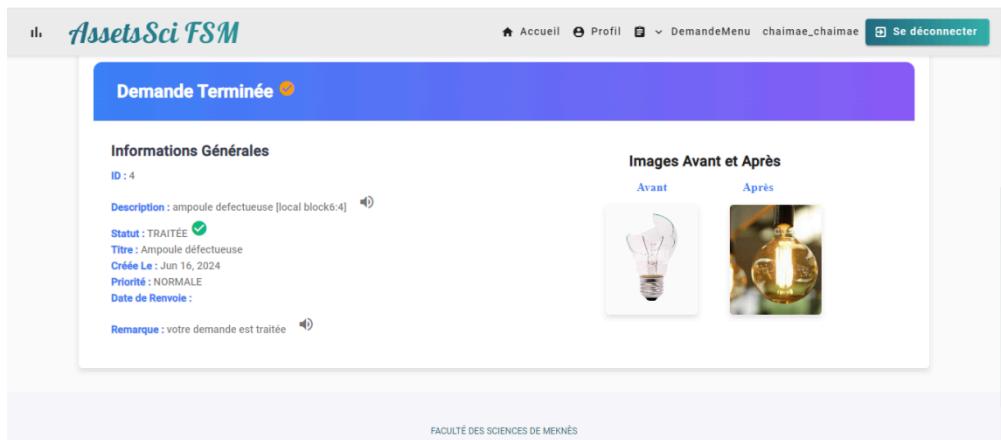


Figure 25 : Page demandes achevées

3.6.5 Créer une demande:

Espace de création des demandes

Créer une demande

Titre*

Description*
ampoule defectueuse
[local block6:4]

Priorité*

Choisir une image Browse

Figure 26 : Créer demande

Cet espace permet aux utilisateurs de créer des demandes de réparation. Les champs obligatoires à remplir sont :

- **Titre** : une brève description du problème.
- **Description** : des détails supplémentaires sur le problème, incluant éventuellement la localisation.
- **Priorité** : le niveau d'urgence de la demande.
- **Choisir une image** : ajouter une image illustrant le problème avant réparation (optionnel).

Ce formulaire facilite la gestion et le suivi des demandes de réparation, en fournissant toutes les informations nécessaires à l'administrateur.

Note : Si un utilisateur soumet plus de 5 demandes, il sera banni pendant une heure.

3.6.6 Accessibilité mobile

L'application est conçue pour être accessible via des appareils mobiles. L'interface utilisateur est entièrement responsive, offrant une expérience utilisateur optimale sur différents types de dispositifs. Grâce à l'utilisation de techniques de design responsive, nous avons pu assurer que les fonctionnalités principales de l'application sont accessibles et utilisables sur les smartphones et les tablettes. Les composants Angular sont adaptatifs et s'ajustent en fonction de la taille de l'écran, garantissant ainsi une navigation fluide et intuitive pour les utilisateurs mobiles.

Maroc Telecom 15:55 52 %

AssetsSci FSM

Espace de création des demandes

Créer une demande

Titre*

Description*
ampoule defectueuse
[local block6:4]

Priorité*

Choisir une image Browse

> Envoyer

The screenshot shows a mobile application interface for creating a demand. At the top, there is a status bar with signal strength, time (15:55), battery level (52%), and a Maroc Telecom logo. Below the status bar is the application's header, "AssetsSci FSM". Underneath the header is the title "Espace de création des demandes". A horizontal line separates the title from the main form area. The form is titled "Créer une demande". It contains several input fields: "Titre*" (Title*) with a placeholder "ampoule defectueuse"; "Description*" (Description*) with a placeholder "[local block6:4]"; "Priorité*" (Priority*) with a dropdown arrow; and file selection buttons "Choisir une image" (Select an image) and "Browse". At the bottom of the form is a large grey button labeled "> Envoyer" (Send).

Figure 27 : Interface utilisateur affichée sur un Smartphone

4. Chapitre 4 : Déploiement de l'application

4.1 Introduction:

Le déploiement de notre application implique la mise en place du backend développé avec Spring Boot et des frontends développés avec Angular. Nous avons utilisé Docker pour faciliter ce processus, en garantissant une portabilité et une isolation des environnements.

4.2 Préparation de l'Environnement de Déploiement

Avant de procéder au déploiement, nous avons préparé l'environnement de déploiement en installant Docker et Docker Compose sur le serveur de production. Voici les étapes principales :

1. Installation de Docker et Docker Compose.
2. Configuration des fichiers Docker nécessaires.
3. Vérification de la connectivité réseau et des permissions.

4.3 Déploiement du Backend avec Docker

Le backend de notre application, développé avec Spring Boot, a été conteneurisé à l'aide de Docker. Voici les étapes suivies :

1. Création du Dockerfile pour le Backend :

```
FROM openjdk:22-oracle
VOLUME /tmp
COPY target/*.jar app.jar
EXPOSE 8082
ENTRYPOINT ["java", "-jar", "/app.jar"]
```

2. Compilation et empaquetage de l'application :

Avant de créer l'image Docker, nous avons besoin de compiler et d'empaqueter l'application. Nous utilisons le script Maven Wrapper pour garantir l'utilisation de la bonne version de Maven :

```

PS C:\Users\Administrateur.SRV\Downloads\PFE_DIR\PFE_DIR\AssetsSci> ./mvnw clean package -DskipTests
[INFO] Scanning for projects...
[WARNING]
[WARNING] Some problems were encountered while building the effective model for fsm.assets:AssetsSci:jar:0.0.1-SNAPSHOT
[WARNING] 'dependencies.dependency.(groupId:artifactId:type:classifier)' must be unique: org.springframework.boot:spring-boot-starter-web:jar -> duplicate declaration of version (?) @ line 62, column 15
[WARNING] It is highly recommended to fix these problems because they threaten the stability of your build.
[WARNING]
[WARNING] For this reason, future Maven versions might no longer support building such malformed projects.
[WARNING]
[INFO]
[INFO] -----< fsm.assets:AssetsSci >-----
[INFO] Building AssetsSci 0.0.1-SNAPSHOT
[INFO]   from pom.xml
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- clean:3.3.2:clean (default-clean) @ AssetsSci ---

```

Cette commande effectue les actions suivantes :

- **clean** : Supprime les fichiers générés par les compilations précédentes pour s'assurer d'un environnement propre.
- **package** : Compile le code source et crée un fichier JAR de l'application.
- **-DskipTests** : Ignore l'exécution des tests unitaires, accélérant ainsi le processus de compilation.

3. Construction de l'image Docker :

Une fois l'application empaquetée, nous utilisons Docker pour créer une image de conteneur :

```

PS C:\Users\Administrateur.SRV\Downloads\PFE_DIR\PFE_DIR\AssetsSci> docker build -t assetssci:latest .
2024/06/14 11:17:44 http2: server: error reading preface from client //./pipe/docker_engine: file has already been closed
[+] Building 0.8s (7/7) FINISHED
  => [internal] load build definition from Dockerfile                               docker:default
  => => transferring dockerfile: 155B                                         0.0s
  => [internal] load metadata for docker.io/library/openjdk:22-oracle           0.0s
  => [internal] load .dockerignore                                              0.7s
  => => transferring context: 2B                                              0.0s
  => [internal] load build context                                             0.0s
  => => transferring context: 858                                            0.0s
  => [1/2] FROM docker.io/library/openjdk:22-oracle@sha256:08b2d714025cbba08c787f5395d931bae89345a856e4ab1be20 0.0s
  => CACHED [2/2] COPY target/*.jar app.jar                                     0.0s
  => exporting to image                                                       0.0s
  => => exporting layers                                                       0.0s
  => => writing image sha256:b0af414154c0b856d041947d45df1816d0d3864e10f70374d7488d1f62c19921 0.0s
  => => naming to docker.io/library/assetssci:latest                           0.0s

```

4. Exécution du conteneur Docker :

Enfin, nous exécutons le conteneur Docker avec l'image créée :

```

PS C:\Users\Administrateur.SRV\Downloads\PFE_DIR\PFE_DIR\AssetsSci> docker run -p 8082:8082 assetssci:latest
 .   _--_
 / \ / _ _ \ \ \ \ \ \ \ \ \
 ( ) \ \ \ \ \ \ \ \ \ \ \ \ \ \
 \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
 ' \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
 == \ \ \ \ \ \ \ \ \ \ \ \ \ \ \
 :: Spring Boot ::           (v3.2.4)

2024-06-14T10:18:27.891Z INFO 1 --- [AssetsSci] [           main] f.assets.assetsci.AssetsSciApplication : Starting AssetsSciApplication v0.0.1-SNAPSHOT using Java 22 with PID 1 (/app.jar started by root in /)
2024-06-14T10:18:27.896Z INFO 1 --- [AssetsSci] [           main] f.assets.assetsci.AssetsSciApplication : No active profile set, falling back to 1 default profile: "default"
2024-06-14T10:18:29.143Z INFO 1 --- [AssetsSci] [           main] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2024-06-14T10:18:29.227Z INFO 1 --- [AssetsSci] [           main] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 73 ms. Found 6 JPA repository interfaces.
2024-06-14T10:18:30.051Z INFO 1 --- [AssetsSci] [           main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8082 (http)

```

5. Configuration de la base de données MySQL :

Nous avons utilisé MySQL Workbench pour configurer la base de données et les utilisateurs nécessaires.

6. Connexion du Backend à la Base de Données :

Les paramètres de connexion à la base de données ont été configurés dans le fichier **application.properties** de Spring Boot.

4.4 Déploiement du Frontend avec Docker

Les frontends de notre application, développés avec Angular, ont également été conteneurisés et déployés. Les étapes suivantes montrent comment cela a été fait, en utilisant des captures d'écran du frontend utilisateur et du frontend admin pour illustrer les étapes.

1. Création du Dockerfile pour le Frontend :

Pour conteneuriser notre frontend, nous avons créé un Dockerfile. Bien que les étapes soient identiques pour le frontend utilisateur et le frontend admin, la capture d'écran ci-dessous montre le processus pour le frontend utilisateur.

```

FROM node:20.14.0 AS build

WORKDIR /app

COPY package*.json ./

RUN npm install

COPY . .

RUN npm run build -- --output-path=../dist/frontend-ang

FROM nginx:alpine as production-stage

COPY --from=build /app/dist/frontend-ang/browser /usr/share/nginx/html
COPY nginx.conf /etc/nginx/conf.d/default.conf

EXPOSE 4201

CMD ["nginx", "-g", "daemon off;"]

```

2. Construction de l'image Docker :

Bien que la construction de l'image Docker soit identique pour les deux frontends, la capture d'écran ci-dessous montre le processus pour le frontend admin.

```

PS C:\Users\Administrateur.SRV\Downloads\ADMIN_APP> docker build -t admin_app:latest .
2024/06/14 11:02:00 http2: server: error reading preface from client //./pipe/docker_engine: file has already been closed
[+] Building 21.5s (15/15) FINISHED
  => [internal] load build definition from Dockerfile                               docker:default
  => => transferring dockerfile: 1.05kB                                         0.0s
  => [internal] load metadata for docker.io/library/nginx:alpine                 0.0s
  => [internal] load metadata for docker.io/library/node:20.14.0                  0.0s
  => [internal] load .dockerignore                                              0.0s
  => => transferring context: 28                                                 0.0s
  => [build 1/6] FROM docker.io/library/node:20.14.0@sha256:d0a9a2399581a9de1ff962a48a28b5cfe700678a0a5df8e31a 0.0s
  => [internal] load build context                                              1.6s
  => => transferring context: 4.13MB                                            1.6s
  => [production-stage 1/3] FROM docker.io/library/nginx@sha256:69f8c2c72671490607f52122be2af27d4fc0965 0.0s
  => CACHED [build 2/6] WORKDIR /app                                           0.0s
  => CACHED [build 3/6] COPY package*.json ./                                     0.0s
  => CACHED [build 4/6] RUN npm install                                         0.0s
  => [build 5/6] COPY . .                                                 5.3s

```

3. Exécution du conteneur Docker :

Pour lancer et exécuter le conteneur Docker, les mêmes étapes sont suivies pour les deux frontends. La capture d'écran ci-dessous montre le processus pour le frontend utilisateur :

```
PS C:\Users\Administrateur.SRV\Downloads\PFE_DIR\PFE_DIR\frontend-ang> docker run -p 172.30.70.16:4201:80 frontend
ng:latest
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: /etc/nginx/conf.d/default.conf differs from the packaged version
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2024/06/14 09:51:06 [notice] 1#1: using the "epoll" event method
```

4. Configuration de Nginx :

Nous avons utilisé Nginx comme serveur web pour servir les fichiers statiques générés par Angular.

4.5 Conclusion

Le déploiement de notre application a été réalisé avec succès en utilisant Docker pour conteneuriser à la fois le backend et le frontend. Cette approche a permis de simplifier le processus de déploiement et d'assurer une meilleure portabilité et gestion des environnements.

Clôture

Ce projet marque une étape importante de notre parcours académique et a permis de mettre en pratique les compétences acquises. Merci à tous ceux qui ont rendu cela possible.