

République algérienne démocratique et populaire

Ministère de l'enseignement supérieur de la recherche scientifique



Université Saad Dahled Blida 1

Faculté des sciences

Département informatique



**Mémoire de fin d'études pour l'obtention du diplôme de Master
en Informatique**

Option : Système informatique et réseaux

**Proposition d'un système de Contrôle d'Accès
Distribué appliqué aux données médicales**

Réaliser par

- SARSAG Sabrine
- SAID Mounia

Encadré par

- Mme. ARKAM Meriem

Devant le jury :

Mme BOUTOUMI Bachira	MAA	Présidente	USDB1
Mme GHEBGHOUB Yasmine	MCB	Examinatrice	USDB1
Mme ARKAM Meriem	MAA	Promotrice	USDB1

Année Universitaire 2021/2022

Remerciement

Avant toute chose, on tient à remercier Allah, le tout puissant, pour nous avoir donné la force et la patience d'achever ce modeste travail, grâce à notre foi, nous croyons au destin, nous pouvons traverser les moments difficiles en regardant toujours le bon côté de la chose.

On exprime nos profonds remerciements à Mme. ARKAM, d'avoir accepté de nous encadrer, pour sa gentillesse et son soutien et aussi pour L'orientation, la patience qui a constitué un apport considérable sans lequel ce travail n'aurait pas pu être mené au bon port.

Nous tenons à remercier sincèrement aux membres du jury pour l'intérêt Qu'ils ont porté à notre recherche en acceptant d'examiner notre travail.

Nos plus sincères remerciements pour notre chef de spécialité M. OULED KHAOUA Pour votre disponibilité, vos conseils si précieux et dirigés, pour les connaissances que vous nous avez apportées. Veuillez croire en notre profond respect

Nous remercions également le chef de département ainsi tous le corps d'enseignants du département d'informatique et surtout les enseignants des spécialités Systèmes d'Informatique et réseaux qui ont contribué de près ou de loin à notre formation pendant les cinq ans qu'on a passé ElHamduli'Allah

Dédicace

En tout premier lieu, je remercie le dieu tout puissant de m'avoir donné la force pour dépasser toutes les difficultés, permis de mener à bien ce travail.

Je dédie ce modeste travail en signe de respect, de reconnaissance et de remerciement :

A mon cher père Mourad à qui je dois ma vie, ma réussite et tout mon respect.

A mon adorable maman Hadjira, qui ne cesse de me donner avec amour le nécessaire pour que je puisse arriver là où je suis.

A mes grands frères et ma petite sœur, pour leur affections et soutient durant toutes ces années.

A toute ma famille

A mes chères meilleures amies Amira et Celia qui ont toujours été là pour m'aider et m'encourager,

Ainsi que tous mes amis qui ont été présents durant tous mon parcours.

A mon binôme Sabrine pour sa contribution, sa compréhension, sa patience et sa clémence.

Mounia.

Dédicace

Je rends HOMMAGE A Feu BOUKROUD Rabah mon petit cousin qui je l'aime trop

Paix a son âme que dieu l'accueille dans son vaste paradis ALLAH YARAHMOU

Je dédie ce travaille

A ma maman, ma meilleure amie, ma source d'énergie positive, celle qui n'a jamais cessé de m'encourager, de me soutenir, et de m'épauler. Ce travaille est un aboutissement pour elle autant que pour moi, quoi que je dis ou je fasse je ne saurais point la remercier comme il se doit

A mon papa, l'homme de ma vie, ce travail est pour lui, lui qui ma tant aidé et accompagné toute ma vie tout au long de mes étude, que dieu lui accorde une bonne santé et longue vie.

A la femme de ma vie, mon âme sœur, mon exemple de force et persévération, il me faudrait toute une vie pour la décrire et la remercier, ma grande sœur Amina.

A mon grand frère Abdelwahab, Tout l'encre du monde ne pourrait suffire pour exprimer mes sentiments envers un être très cher. Vous avez toujours été mon école de patience, de confiance et surtout d'espoir et d'amour. Vous êtes et vous resterez pour moi ma référence, la lumière qui illumine mon chemin.

A mes frères Omar & Ishak, pour leur dévouement, leur compréhension et leur grande tendresse, qui en plus de m'avoir encouragé tout le long de mes études.

A mon beau frère Mourad, pour sa bonne humeur et sa joie de vivre.

A mes copines Nihed, Lynda et Tati qui m'ont encouragé et qui sont toujours la dans les meilleurs moments comme dans les pires.

A mes chères neveux Iyad, Younes, Youcef, Anes, Fares et ouis.

A mes adorables nièces, Racha & Lina.

A mes amis Marouane, Ayoub, Imad, Moumen, Saber.

A mon chère binôme Mounia, un exemple de positivité et de force,

Merci de m'avoir soutenue et d'avoir supporté mes sautes d'humeur.

Je vous aime tant.

Sabrina.

Résumé

Grâce à l'évolution technologique des matériels, des dispositifs mobiles, des logiciels et de la connectivité, les systèmes de santé permettent d'assurer aux utilisateurs une accessibilité transparente aux ressources médicales à tout moment, depuis n'importe où et avec n'importe quel dispositif, ces accès doivent disposer de plusieurs limites pour la protection de la vie personnelle des individus selon la loi qui considère les données médicales comme données sensibles très particulières.

L'accès et le partage sécurisés des données médicales entre différentes institutions médicales publiques et/ou privées connaissent actuellement une évolution technologique rapide. Les technologies fournies sont généralement mises en œuvre sur la base de systèmes et d'applications de contrôle d'accès. Ceci rend l'interopérabilité entre les différentes applications médicales une tâche très fastidieuse d'autant plus qu'il faudra prendre en considération toutes les politiques de sécurité de chaque application, les préférences de confidentialité des patients ainsi que les droits d'accès attribués par utilisateur sans omettre qu'un même utilisateur se trouvera souvent attribuer plusieurs comptes d'accès pour chaque application.

Dans notre projet, nous cherchons à proposer une solution médiatrice permettant le contrôle d'accès des différents utilisateurs aux différentes sources de données d'une manière centralisée et autonome de toute technologie utilisée dans les applications locales.

Mot clés : Système de santé, Contrôle d'accès, Middleware, XML.

ملخص

من خلال التطور التكنولوجي للأجهزة والأجهزة المحمولة والبرامج والاتصال ، تساعد الأنظمة الصحية على ضمان وصول المستخدمين بسهولة إلى الموارد الطبية في أي وقت وفي أي مكان ، مع أي جهاز ، يجب أن يكون لهذه الوصول عدة قيود لحماية الحياة الشخصية الأفراد وفقاً للقانون الذي يعتبر البيانات الطبية بيانات حساسة للغاية

يخضع الوصول الآمن إلى البيانات الطبية ومشاركتها بين مختلف المؤسسات الطبية العامة و / أو الخاصة حالياً للتطور التكنولوجي السريع. يتم تنفيذ التقنيات المقدمة بشكل عام على أساس أنظمة التحكم في الوصول والتطبيقات. هذا يجعل التشغيل البيئي بين التطبيقات الطبية المختلفة مهمة شاقة للغاية ، خاصة أنه سيكون من الضروري مراعاة جميع السياسات الأمنية لكل تطبيق ، وتفضيلات السرية للمرضى وكذلك حقوق الوصول المعينة من قبل المستخدم دون إغفال ذلك غالباً ما يتم تعيين عدة حسابات وصول لكل تطبيق للمستخدم نفسه

نسعى في مشروعنا إلى اقتراح حل وسيط يسمح بالتحكم في وصول المستخدمين المختلفين إلى مصادر البيانات المختلفة بطريقة مركزية ومستقلة لأي تقنية مستخدمة في التطبيقات المحلية

الكلمات المفتاحية: التحكم في الوصول ، البرامج الوسيطة ، XML ، النظام الصحي.

Through the technological evolution of hardware, mobile devices, software and connectivity, healthcare systems enable users to ensure seamless accessibility to medical resources anytime, anywhere and with any which device, these accesses must have several limits for the protection of the personal life of individuals according to the law which considers medical data as very specific sensitive data.

The secure access and sharing of medical data between different public and/or private medical institutions is currently undergoing rapid technological development. The technologies provided are generally implemented on the basis of access control systems and applications. This makes interoperability between the different medical applications a very tedious task, especially since it will be necessary to take into consideration all the security policies of each application, the confidentiality preferences of the patients as well as the access rights assigned by user without omitting that the same user will often be assigned several access accounts for each application.

In our project, we seek to propose a mediating solution allowing the access control of the different users to the different data sources in a centralized and autonomous way of any technology used in the local applications.

Keywords: Health system, Access control, Middleware, XML.

Table des matières

Chapitre 1 : Contrôle d'accès

1	Introduction.....	18
2	Définition de la sécurité informatique	18
2.1	L'authentification	18
2.2	La confidentialité	19
2.3	L'intégrité.....	19
2.4	La disponibilité	20
2.5	La non-répudiation	20
3	Mécanisme de sécurité	20
4	Contrôle d'accès	22
4.1	Définition	22
5	L'objectif du contrôle d'accès.....	22
6	Différentes approches	24
6.1	Contrôle d'accès Discrétionnaire	24
6.1.1	Définition	24
6.1.2	Les avantages du contrôle d'accès DAC.....	25
6.1.3	Les inconvénients du contrôle d'accès DAC.....	25
6.2	Contrôle d'accès Obligatoire	25
6.2.1	Définition	25
6.2.2	Les avantages de contrôle d'accès MAC.....	26
6.2.3	Les inconvénients de contrôle d'accès MAC	26
6.2.4	Contrôle d'accès Basé Sur Les Rôles.....	26
6.2.5	Définition	26
6.2.6	Les avantages de contrôle d'accès RBAC.....	27
6.2.7	Les inconvénients de contrôle d'accès RBAC	28
6.3	Contrôle d'accès Basé Sur Les Attributs	28
6.3.1	Définition	28
6.3.2	Les avantages de contrôle d'accès ABAC	28
6.3.3	Inconvénients de contrôle d'accès ABAC	30
6.4	Comment choisit entre RBAC et ABAC.....	30
7	Politique de contrôle d'accès	32
7.1	Politique de contrôle d'accès statique	32
7.2	Politique de contrôle d'accès dynamique	32

8	Comparaison entre les différentes méthodes de contrôle d'accès :	33
9	LE CHOIX DU BON CONTRÔLE D'ACCÈS.....	33
10	Conclusion	34

Chapitre 2: Généralité sur Middleware

1	Introduction.....	36
2	Définition du middleware.....	36
3	Origines et objectifs.....	36
4	Les cas d'utilisation des middlewares	37
4.1	Développement de jeux	37
4.2	Matériel électronique.....	37
4.3	Développement de logiciels	37
4.4	Transmission de données.....	37
4.5	Applications distribuées	38
5	L'architecture middleware	38
5.1	Console de gestion	38
5.2	Interface client	39
5.3	Cadre de messagerie commun	39
5.4	Interface interne de logiciel	39
5.5	Interface de plateforme	39
5.6	Gestionnaire de contrat	39
5.7	Gestionnaire de session	40
5.8	Gestionnaire de base de données.....	40
5.9	Moniteur d'exécution.....	40
6	Fonctionnement du middleware.....	40
7	Le rôle du middleware.....	40
8	Type du middleware.....	41
8.1	Passage de messages.....	41
8.2	Middleware d'appel de procédure.....	41
8.3	Middleware orienté objets.....	41
8.4	Middleware de composant	41
8.5	Middleware orienté message.....	42
8.6	Logiciels intermédiaires Publier-Souscrire	42
8.7	Middleware transactionnel	42
8.8	Middleware orienté base de données	42
8.9	Mémoire partagée distribuée	43
8.10	Middleware web.....	43

9	Conclusion	43
---	------------------	----

Chapitre 3: Modélisation et Etude conceptuelle

1	Introduction.....	45
2	Domaine d'application de la solution.....	45
3	Problèmes et objectifs.....	45
4	Modélisation du RBAC.....	46
4.1	Hiérarchie des rôles.....	47
5	Description de la solution.....	47
5.1	Schématisation de la solution	47
5.1.1	Position du Middleware	48
5.1.2	Identification des rôles	49
5.1.3	Serveur d'application.....	49
5.1.4	Systèmes de Gestion de base de données	50
5.2	Spécification des droits d'accès.....	50
5.2.1	Construction des fichiers XML.....	50
5.2.2	Enregistrement des XML	64
5.3	Processus effectué par le middleware	64
5.3.1	Interception des requêtes http	64
5.3.2	Traitement des requêtes http	65
5.3.3	Renvoi des réponses http.....	67
6	Etude conceptuelle de l'exemple d'application	67
6.1	Diagramme de cas d'utilisation	67
6.1.1	Diagramme de cas d'utilisation de l'utilisateur	68
6.1.2	Diagramme de cas de l'administrateur	68
6.1.3	Diagramme de cas du médecin	69
6.1.4	Diagramme de cas de l'assistante	70
6.1.5	Diagramme de cas du patient	70
6.2	Diagramme de séquence.....	71
6.2.1	Diagramme de séquence pour l'authentification	71
6.2.2	Diagramme de séquence pour la spécification	72
6.2.3	Diagramme de séquence pour vérification d'autorisation	72
7	Conclusion	75

Chapitre 4: Implémentation et Réalisation

1	Introduction.....	77
2	Outils et environnement de développement.....	77

2.1	Spring Boot Framework.....	77
2.2	Apache maven.....	77
2.3	MySQL.....	77
2.4	Java développement kit.....	78
2.5	XSD et XML.....	78
2.6	JSON.....	78
2.7	JDBC.....	78
2.8	MVC.....	79
2.9	Html, CSS et Javascript.....	79
2.10	Thymeleaf.....	79
2.11	Bootstrap.....	79
3	Présentation de l'application.....	80
3.1	Rôle patient.....	80
3.1.1	Informations clinique.....	80
3.1.2	Spécification d'autorisation.....	82
3.1.3	Consultation des données.....	83
3.2	Rôle assistante.....	85
3.2.1	Consultation des patients et rendez vous.....	85
3.2.2	L'ajout d'un patient.....	87
3.3	Rôle médecin.....	88
3.3.1	Consultation des patients et rendez vous.....	88
3.3.2	Consultation des données d'un patient.....	89
3.3.3	Ajouter des données pour un patient.....	89
3.4	Rôle Administrateur (Chef clinique).....	90
3.4.1	Enregistrer un médecin et/ou assistante.....	90
3.4.2	Consulter la liste des médecins et assistantes.....	92
3.4.3	Supprimer un médecin et/ou assistante.....	92
3.4.4	Consulter les notifications.....	93
3.5	Interface d'utilisateur.....	93
4	Simulation de la solution.....	96
5	Conclusion.....	102
Conclusion et perspectives		
1	Conclusion et perspective.....	104
1	Bibliographie.....	106

Table des Figures

1	Figure 1. Identification et authentification.	10
2	Figure 2. Mécanisme de sécurité.	21
3	Figure 3. Type de contrôle d'accès.	23
4	Figure 4 . Définition de contrôle d'accès.	24
5	Figure 5. Les différents architectes de contrôle d'accès.	32
6	Figure 6. Architecture du middleware [28]	38
7	Figure 7. Modélisation du RBAC	46
8	Figure 8. Hiérarchie des rôles	47
9	Figure 9. Architecture générale	48
10	Figure 10. Illustration de la structure XML.	53
11	Figure 11. Logigramme.	66
12	Figure 12. Diagramme de cas d'utilisateur	68
13	Figure 13. Diagramme de cas de l'administrateur	69
14	Figure 14. Diagramme de cas de gestion de médecin	69
15	Figure 15. Diagramme de cas d'utilisation de l'assistant	70
16	Figure 16. Diagramme de cas d'utilisation du patient	70
17	Figure 17 : Diagramme de séquence d'authentification	71
18	Figure 18. Diagramme de séquence de spécifications	72
19	Figure 19. Diagramme de séquence de vérification d'autorisation	73
20	Figure 20. Diagramme de classe.	74
21	Figure 21. Accueil	80
22	Figure 22. Départements.	80
23	Figure 23. Les médecins	80
24	Figure 24. Formulaire de spécification de droits d'accès	81
25	Figure 25. Récupération des droits	81
26	Figure 26. Création du XML.	82
27	Figure 27. Consultation des maladies.	83
28	Figure 28. Consultation des vaccins	83
29	Figure 29. Consultations des analyses.	83
30	Figure 30. Les patients d'un médecin.	84
31	Figure 31. Les rendez-vous d'un médecin	84
32	Figure 32. Modifier ou supprimer un rendez-vous.	85

33	Figure 33. Prendre rendez vous ou retirer un patient	85
34	Figure 34. Affecter un patient a un médecin	86
35	Figure 35. Enregistrer un patient	86
36	Figure 36. Consulter patients	87
37	Figure 37. Consulter rendez vous	87
38	Figure 38. Consulter informations patient	88
39	Figure 39. Ajouter une maladie	88
40	Figure 40. Ajouter un vaccin	89
41	Figure 41. Nouvel utilisateur	89
42	Figure 42. Nouveau médecin	90
43	Figure 43. Nouvelle assistante	90
44	Figure 44. Consulter médecins/assistantes	91
45	Figure 45. Suppression d'utilisateur	91
46	Figure 46. Consultation des notifications	92
47	Figure 47. Interface d'authentification	92
48	Figure 48. Consulter informations	93
49	Figure 49. Modifier informations personnelles	93
50	Figure 50. Code d'enregistrement d'un patient	94
51	Figure 51. Accès non autorisé	94
52	Figure 52. Sélection des données	95
53	Figure 53. Résultat de l'opération	95
54	Figure 54. Résultat de changement de donnée	96
55	Figure 55. Sélection de données autorisées et non autorisées	96
56	Figure 56. Résultat d'échec d'opération	97
57	Figure 57. Fichier XML du patient id=33	98
58	Figure 58. Ajouter une maladie chronique au nom diabète	99
59	Figure 59. Réexécution de l'opération de figure 56	99
60	Figure 60. Résultat de la réexécution	100
61	Figure 61. Code de vérification d'autorisation	100

Liste des Tableaux

1	Tableau 1. Comparaison entre RBAC et ABA	31
2	Tableau 2. Comparaison entre les différentes approches de contrôle d'accès.	33
3	Tableau 3. HashMap.....	61
4	Tableau 4. Information rangée dans HashMap.....	62

A decorative graphic of a scroll with a black outline and a light gray shadow. The scroll is unrolled in the middle, with the top and bottom edges curving upwards and downwards respectively. The text is centered within the unrolled portion.

Introduction Générale

Introduction Générale

De plus en plus, les systèmes de santé adoptent les nouvelles technologies pour la génération, le traitement, le stockage et la consultation de données médicales. Cette adoption vise à assurer une meilleure qualité de service et un fonctionnement plus efficace pour une meilleure consultation des données dans différents contextes. Prenant en compte la quantité d'informations distribuées dans différents référentiels physiques, l'accès aux ressources médicales en temps réel est devenu prioritaire.

En conséquence et grâce à l'évolution technologique des matériels, des dispositifs mobiles, des logiciels et de la connectivité, les systèmes de santé trouvent leur voie vers l'ubiquité qui leur permettra d'assurer aux utilisateurs une accessibilité transparente aux ressources médicales à tout moment, depuis n'importe où et avec n'importe quel dispositif.

Dans un premier temps, la sensibilité et la confidentialité des données médicales justifient le fait qu'elles soient conservées dans leurs ressources d'origine et distribuées dans différents sous-systèmes (hôpitaux, laboratoires d'analyse, cabinets de médecin, etc.). Cette décentralisation influence la gestion d'accès aux ressources médicales qui passe par la distribution des privilèges d'accès selon le rôle de l'utilisateur dans la hiérarchie du système en utilisant le standard RBAC – RoleBased Access Control.

La nature évolutive de données médicales est très intéressante car elle reflète non seulement l'avancement d'une situation d'un patient en prenant en compte l'axe temporel mais aussi, les différentes interventions qui ont eu lieu par les membres de l'équipe médicale. Sachant qu'une grande partie des données médicales est générée et traitée en temps réel, l'administration des privilèges d'accès doit être centralisée pour assurer l'intégrité du système et une prise de décision fiable.

Dans un second temps, nous soulignons l'importance et le besoin du contrôle d'accès au dossier médical non seulement par rapport au rôle de l'utilisateur (RBAC) mais aussi par rapport à son contexte (localisation, temps, machine, connexion, etc.) et à la situation d'utilisation du système (urgence, accès depuis une base de donnée non familière, etc.).

En analysant les caractéristiques des systèmes de santé, nous pouvons dire qu'ils sont de plus en plus centrés sur l'utilisateur (médecin, infirmière, patient, etc.) et qu'ils utilisent des technologies orientées service pour garantir une certaine qualité. Dans ces systèmes, la qualité de services est critique car elle touche la vie du patient.

Cependant, un patient est le propriétaire de ses données médicales, il a le droit d'autoriser ou d'interdire aux utilisateurs d'accéder à ses informations sauf pour un intérêt vital.

Malheureusement le système de contrôle d'accès actuel ne se concentre pas sur les préférences du patient.

Notre travail consiste à proposer une solution médiatrice permettant le contrôle d'accès des différents utilisateurs aux différentes sources de données d'une manière centralisée et autonome de toute technologie utilisée dans les applications locales.

Pour cela, nous organisons notre mémoire en 4 chapitres structuré comme suit :

Chapitre 1 : Dans ce chapitre nous exposons des généralités sur la sécurité informatique en donnant quelques définitions, Nous présentons les différents modèles de contrôle d'accès aux systèmes d'information.

Chapitre 2 : Nous présentons la couche médiatrice les middlewares.

Chapitre 3 : Nous couvrons dans ce chapitre la conception et la modélisation de la solution proposée.

Chapitre 4 : Ce chapitre représente la réalisation et la mise en œuvre de la couche de sécurité

Et enfin dans la conclusion générale, on résume les limites et les perspectives futures de notre travail.



Chapitre 1 : contrôle d'accès

1 Introduction

Généralement, dans un système informatique on considère souvent la confidentialité, l'intégralité et la disponibilité des données comme les trois propriétés fondamentales à respecter

La sécurisation des systèmes d'information est un aspect très important qui est devenue un enjeu important pour les différents systèmes ainsi que pour l'ensemble des acteurs qui l'entourent, dont l'objectif est la protection des ressources informatiques contre l'utilisation non-autorisée, le mauvais usage, la divulgation et la modification, tout en garantissant l'accès pour les utilisateurs légitimes.

Pour assurer cette sécurité, plusieurs techniques ont été proposées dont nous présentons dans ce chapitre les notions de base de la sécurité informatique et les différents modèle de contrôle d'accès, Ainsi la politique du contrôle d'accès.

2 Définition de la sécurité informatique

La sécurité informatique est l'ensemble des moyens techniques, organisationnels, juridiques et humains nécessaire mis en œuvre pour réduire la vulnérabilité d'un système informatique contre les menaces accidentelles ou intentionnelles. L'objectif de la sécurité informatique est d'assurer que les ressources matérielles et/ou logicielles d'un parc informatique soient uniquement utilisées dans le cadre prévu et par des utilisateurs autorisés [1]. Il convient d'identifier les exigences fondamentales en sécurité informatique, qui caractérisent ce à quoi s'attendent les utilisateurs de ce systèmes informatiques au regard de la sécurité à savoir :

2.1 L'authentification

L'authentification doit permettre de vérifier l'identité d'une entité afin de s'assurer, entre autres, de l'authenticité de celle-ci. Pour cela, l'entité devra prouver son identité, le plus souvent en donnant une information spécifique qu'elle est censée être seule à détenir telle que, par exemple, un mot de passe ou une empreinte biométrique. [2] Ensuite on passe à la gestion des droits et des permissions comme illustré dans figure si dessous :

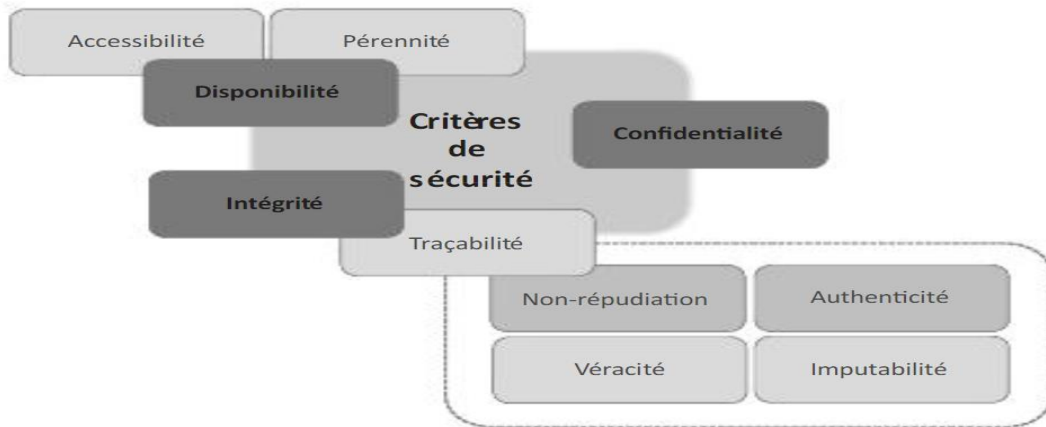


Figure 1. Identification et authentification.[2]

2.2 La confidentialité

La notion de **confidentialité** est liée au maintien du secret, elle est réalisée par la protection des données contre une divulgation non autorisée. Il existe deux types d'actions complémentaires permettant d'assurer la confidentialité des données.[2]

- Limiter et contrôler leur accès afin que seules les personnes habilitées à les lire ou à les modifier puissent le faire.
- Les rendre inintelligibles en les chiffrant de telle sorte que les personnes qui ne sont pas autorisées à les déchiffrer ne puissent les utiliser.

2.3 L'intégrité

Le critère **d'intégrité** des ressources physiques et logiques (équipements, données, traitements, transactions, services) est relatif au fait qu'elles sont demeurées intactes, qu'elles n'ont pas été détruites (altération totale) ou modifiées (altération partielle) à l'insu de leurs propriétaires tant de manière intentionnelle qu'accidentelle. Une fonction de sécurité appliquée à une ressource pour préserver son intégrité permet de la protéger contre une menace de corruption ou de destruction.[2]

Se prémunir contre l'altération des données et avoir la certitude qu'elles n'ont pas été modifiées lors de leur stockage, de leur traitement ou de leur transfert contribue à la qualité des prises de décision basées sur celles-ci.

2.4 La disponibilité

La **disponibilité** d'une ressource est relative à la période de temps pendant laquelle le service qu'elle offre est opérationnel. Le volume potentiel de travail susceptible d'être pris en charge durant la période de disponibilité d'un service détermine la capacité d'une ressource à être utilisée (serveur par exemple).

Il ne suffit pas qu'une ressource soit disponible, elle doit pouvoir être utilisable avec des temps de réponse acceptables. Sa disponibilité est indissociable de sa capacité à être accessible par l'ensemble des ayants droit.[2]

La disponibilité des services, systèmes et données est obtenue par un dimensionnement approprié et une certaine redondance des infrastructures ainsi que par une gestion opérationnelle et une maintenance efficaces des infrastructures, ressources et services.

2.5 La non-répudiation

La non-répudiation est le fait de ne pouvoir nier ou rejeter qu'un événement (action, transaction) a eu lieu. À ce critère de sécurité peuvent être associées les notions d'imputabilité, de traçabilité ou encore parfois d'auditabilité. [2]

- **L'imputabilité** se définit par l'attribution d'une action (un événement) à une entité déterminée (ressource, personne).
- La **traçabilité** permet de suivre la trace numérique laissée par la réalisation d'un événement (message électronique, transaction commerciale, transfert de données...).
- **L'auditabilité** se définit par la capacité d'un système à garantir la présence d'informations nécessaires à une analyse ultérieure d'un événement (courant ou exceptionnel) effectuée dans le cadre de procédures de contrôle spécifiques et d'audit.

3 Mécanisme de sécurité

Un mécanisme de sécurité est conçu pour détecter, prévenir et lutter contre une attaque de sécurité en d'autre terme il fournit et supporte les services de sécurité, on en distingue deux classes : mécanismes spécifiques à certains services et mécanismes génériques.[3]

Voici l'organigramme en dessous qui présente les différents types de mécanisme de sécurité.[3]

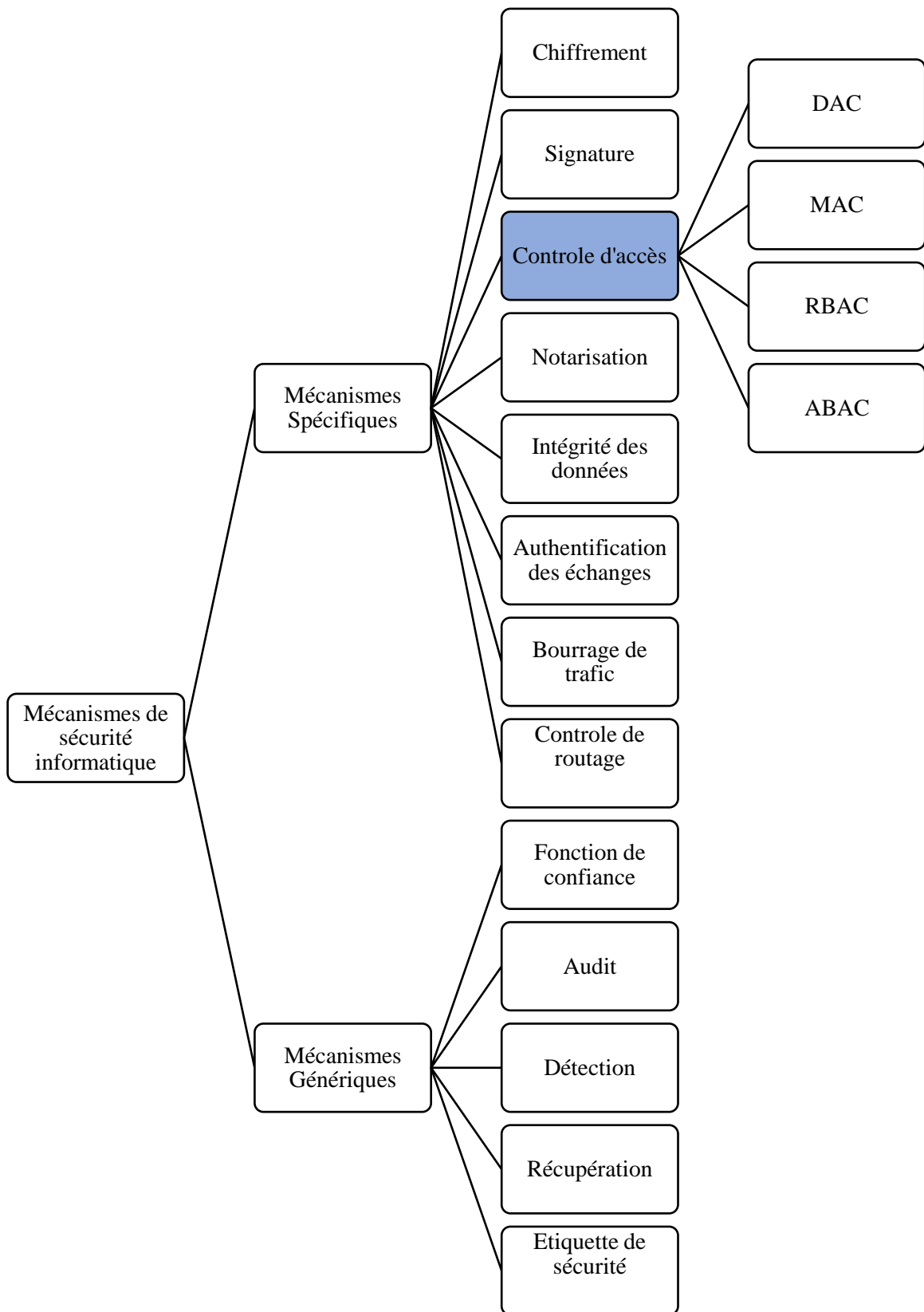


Figure2. Mécanisme de sécurité.

4 Contrôle d'accès

4.1 Définition

Le contrôle d'accès est le processus consistant à accorder ou refuser l'accès à une ressource ou une fonctionnalité spécifique, en fonction des droits initialement attribués. Souvent exploitées par les attaquants, les vulnérabilités liées au manque de contrôle d'accès permettent aux attaquants d'accéder aux informations d'autres utilisateurs, d'obtenir des accès utilisateurs à privilèges élevés voire des accès administrateurs.

Il est important de rappeler que les premières mesures de sécurité informatique consistent à contrôler tous les accès possibles à un système d'information et à autoriser ou non un nombre précis d'opérations en fonction de l'utilisateur. Il s'agit de limiter l'accès aux ressources du système d'information aux seuls utilisateurs, programmes, processus ou systèmes autorisés. Le contrôle d'accès est défini comme tout mécanisme par lequel le système accorde ou refuse à une entité active (sujet) le droit d'accéder à une entité passive (objet) ou d'effectuer une action. Il s'agit notamment de vérifier que l'entité (personne, ordinateur, etc.) souhaitant accéder à la ressource dispose des autorisations nécessaires pour le faire. [6]

5 L'objectif du contrôle d'accès

Contrôle d'accès est un processus qui garantit un niveau de protection la confidentialité, l'intégrité et la disponibilité des ressources consultées (objets) Par l'utilisateur (sujet) via les méthodes suivantes :

- **Identification** : C'est le processus d'établissement de l'identité de l'utilisateur. Il permet de répondre à la question : "Qui es-tu ?". L'utilisateur utilise un identifiant "nom d'utilisateur" pour s'identifier et est Attribué individuellement. Cet identifiant est unique.
- **Authentification** : Il s'agit d'un processus qui vise à prouver l'identité d'un utilisateur. Cela se passe après la phase d'identification. Elle permet de répondre à la question : "Êtes-vous vraiment cette personne ?". L'utilisateur utilise un authentifiant ou "mot de passe" que lui seul connaît.
- **Autorisation** : Il s'agit d'un processus qui permet d'associer formellement et légalement des sujets et des ressources par le biais de droits ou d'autorisations.

– **Responsabilité** : il s'agit d'un processus qui permet de suivre les détails. Assurez-vous que les actions effectuées sur la ressource sont conformes aux étapes d'autorisation de la politique de l'autorité de certification. [7]

Il existe trois types de contrôle d'accès : physiques, logiques et administratifs [7] :

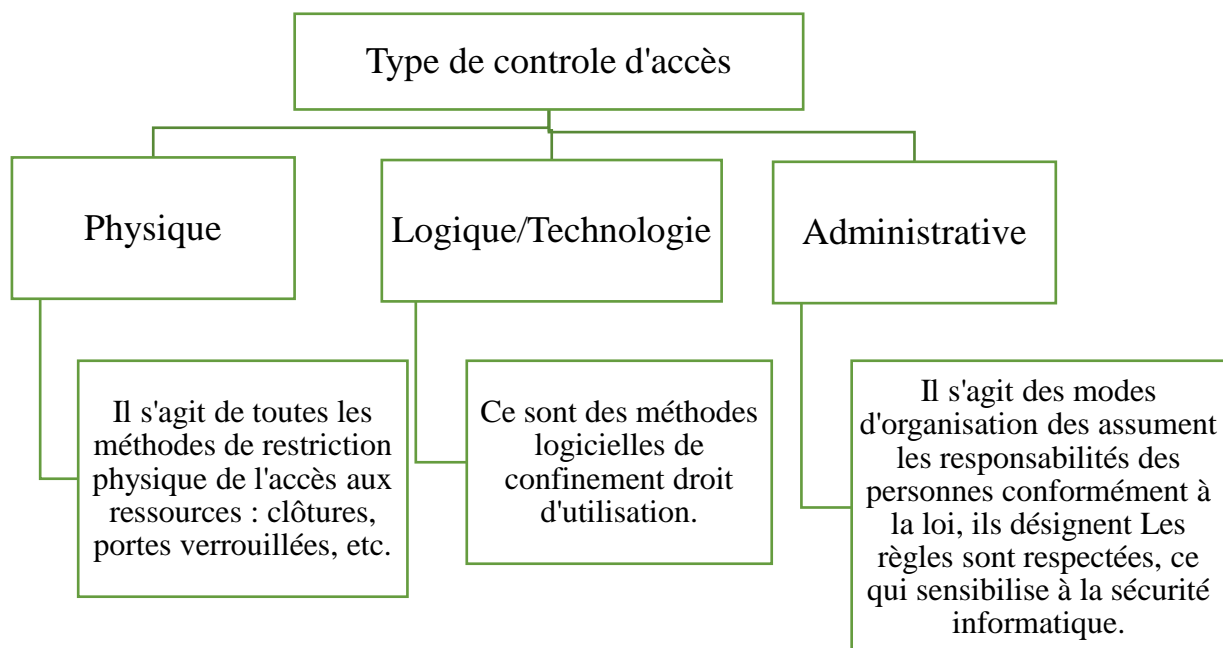


Figure 3. Type de contrôle d'accès.

Dans le cadre de ce travail, nous nous intéressons plus particulièrement aux techniques logiques de contrôle d'accès.

Le modèle de contrôle d'accès est défini par les éléments suivants : Sujet, Contrôle, Action, Objet. Un sujet peut avoir la permission, l'interdiction, l'obligation ou le conseil d'effectuer une action sur un ou plusieurs objets.

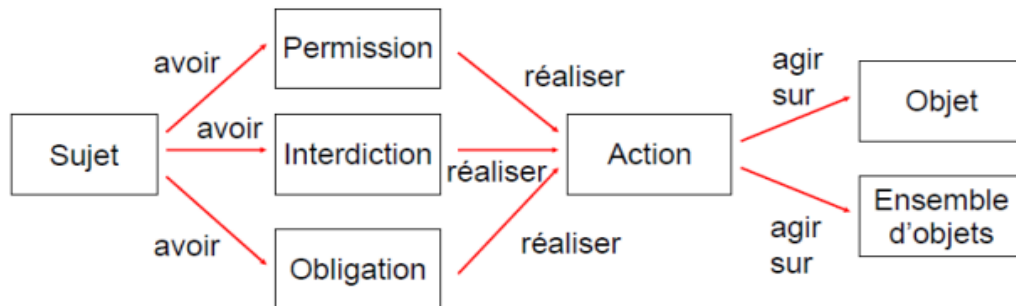


Figure 4. Définition de contrôle d'accès.

6 Différentes approches

Il existe quatre approches du contrôle d'accès aux données : (a) modèle de contrôle d'accès Discrétionnaire (DAC) ; (b) Modèle de contrôle d'accès obligatoire (MAC) ; (c) de contrôle d'accès basé sur les rôles (RBAC) et (d) modèle de contrôle d'accès basé sur les attributs (ABAC).

6.1 Contrôle d'accès Discrétionnaire

6.1.1 Définition

Modèle de contrôle d'accès discrétionnaires ou « Discretionary Access Control». il est basé sur les concepts de sujet, objet, et les droits d'accès.

Le sujet a un contrôle complet sur tous les objets qui lui appartiennent, il peut changer les permissions d'accès, transférer des objets authentifiés ou des accès à l'information à d'autres sujets. C'est pourquoi il est dit discrétionnaire.

Dans ce modèle, les autorisations sont attribuées directement à des sujets en fonction de leur identité ; l'inconvénient d'une telle approche est que, dans les grands systèmes, déterminer l'octroi de l'autorisation sur une ressource donnée à des utilisateurs individuels, est laborieux et difficile à gérer.

La révocation de la permission est également complexe lorsque l'utilisateur quitte l'entreprise ou change de fonction, par exemple. L'information peut être copiée d'un objet à un autre, de sorte que l'accès à une copie est possible même si le propriétaire initial ne donne pas accès à l'originale. Puis que les politiques du DAC peuvent être facilement modifiées par le propriétaire, un programme malveillant s'exécutant en son nom pourra aussi changer ces

mêmes politiques, ce qui constitue une faiblesse de ce système.[8]

6.1.2 Les avantages du contrôle d'accès DAC

Convivialité : l'interface utilisateur est facile à utiliser et il est très facile de gérer les données et les autorisations avec ce système.

Flexibilité : comme mentionné précédemment, le contrôle d'accès discrétionnaire est extrêmement flexible, permettant le plus grand nombre d'allocations et la possibilité d'accorder facilement l'accès à d'autres également.

Maintenance minimale : ces systèmes ne nécessitent pas d'entretien et de maintenance réguliers et imposent moins de responsabilité à l'administration pour les gérer.[9]

6.1.3 Les inconvénients du contrôle d'accès DAC

Moins sécurisé : étant donné que l'accès peut facilement être accordé d'une personne à une autre, le DAC n'est pas le système le plus sécurisé et des informations peuvent être divulguées à l'extérieur de l'organisation.

Difficile de garder une trace : le DAC n'étant pas centralisé, le seul moyen de surveiller les flux de données et les autorisations d'accès est de passer par l'ACL (accès control List). Ceci, cependant, ne peut être fait que dans le cas d'une petite organisation avec peu d'employés.[9]

6.2 Contrôle d'accès Obligatoire

6.2.1 Définition

Le contrôle d'accès obligatoire (MAC) est le plus couramment utilisé dans les systèmes où la confidentialité des données est la principale préoccupation. Le MAC consiste à attribuer une étiquette de classement à chaque ressource du fichier. La classification est effectuée par catégorie d'informations et niveau de sensibilité, tels que confidentiel, classifié ou top secret. Les sujets sont catégorisés de la même manière : chaque sujet se voit attribuer des droits d'accès. Lorsqu'un principal tente d'accéder à une ressource particulière, le système vérifie les autorisations du principal pour déterminer si l'accès est autorisé ; les droits d'accès accordés au sujet dépendent de la classification de la ressource.

Le haut niveau de confidentialité et d'intégrité du Mandatory Access Control ou Contrôle d'Accès obligatoire a imposé son usage dans les secteurs utilisant des données sensibles, notamment dans les environnements exposés sur le plan de la sécurité. C'est le cas par exemple de l'armée, des autorités gouvernementales, du secteur de la politique, du commerce

extérieur, du domaine de la santé et du service de renseignements. Il n'est pas rare non plus que des entreprises ordinaires aient recours au MAC. Le système d'exploitation Security-Enhanced Linux (SELinux) est par exemple bâti sur l'implémentation d'un contrôle d'accès obligatoire dans le noyau de Linux.[10]

6.2.2 Les avantages de contrôle d'accès MAC

Le Mandatory Access Control ou Contrôle d'Accès obligatoire fait partie des contrôles d'accès les plus sécurisés, car on ne peut quasiment pas l'enfreindre.

Le système MAC ne permet aux utilisateurs d'apporter aucune modification.

Le contrôle et l'attribution des droits d'accès se font de façon parfaitement automatique par le système.

Le Mandatory Access Control offre un haut niveau de confidentialité.

Le système est par ailleurs caractérisé par une excellente intégrité.

Sans autorisation préalable, il est impossible de modifier les données qui sont donc parfaitement à l'abri d'un usage malveillant. [11]

6.2.3 Les inconvénients de contrôle d'accès MAC

La mise en œuvre d'un contrôle d'accès obligatoire demande cependant une planification détaillée en amont, et son administration après implémentation impose un suivi important.

Chaque assignation de droit sur des objets ou des utilisateurs demande une vérification et une réactualisation permanentes.

Il en va de même des tâches d'entretien courantes, parmi lesquelles on compte l'ajout de nouvelles données, de nouveaux utilisateurs, la prise en compte de modifications dans la catégorisation ou dans la classification.

Généralement, l'attribution de ces droits repose sur une seule personne. Cela garantit certes un niveau de sécurité élevé, mais représente souvent une dose de travail importante pour l'administrateur. [11]

6.2.4 Contrôle d'accès Basé Sur Les Rôles

6.2.5 Définition

Role-Based Access Control a été proposé pour fournir un modèle et des outils qui permettent de gérer le contrôle d'accès dans un système complexe avec un très grand nombre d'utilisateurs et d'objets.[12]

Dans un modèle de contrôle d'accès basé sur les rôles (RBAC), l'accès à une ressource est déterminé par la relation du demandeur avec l'organisation ou la personne contrôlant la ressource, le rôle ou la fonction du demandeur détermine si l'accès est autorisé ou refusé.

Chaque utilisateur se voit attribuer un ou plusieurs permissions sont attribuées à chaque rôle. Les permissions ne sont plus associées de manière directe aux sujets, mais par le biais de rôles, qui regroupent des sujets qui remplissent les mêmes fonctions.

Le contrôle d'accès basé sur les rôles (RBAC) devient l'une des méthodes de contrôle les plus largement adoptées. Pour certains, RBAC vous permet de regrouper des individus et d'attribuer des autorisations pour des rôles spécifiques. Si vous décidez d'utiliser RBAC, vous pouvez également ajouter des rôles dans des groupes ou directement aux utilisateurs.

Dans de nombreuses organisations, le contrôle d'accès basé sur les rôles s'est révélé la meilleure procédure d'administration des autorisations d'accès. Cependant, le modèle RBAC peut être utilisé même dans les systèmes d'exploitation et les autres logiciels, notamment pour le service de répertoire Microsoft Windows Server Active Directory, pour le système d'exploitation Linux SELinux optimisé dans le domaine de la sécurité ou le système d'exploitation Unix Solaris.[13]

Selon certaines conditions, le contrôle d'accès basé sur les rôles s'est établi comme un modèle de bonnes pratiques. Si le concept de rôles et d'autorisation est défini et appliqué de manière obligatoire dans toute l'entreprise, le RBAC présente de nombreux avantages et aussi des inconvénients :

6.2.6 Les avantages de contrôle d'accès RBAC

Flexibilité : l'entreprise attribue seulement un ou plusieurs rôles à un collaborateur selon les besoins. Les modifications au sein de la structure de l'organisation ou des autorisations sont rapidement transmises à tous les collaborateurs tandis que l'entreprise adapte les rôles correspondants.

Faible charge administrative : le RBAC rend obsolète l'attribution coûteuse d'autorisations individuelles.

Risque d'erreurs réduit : les autorisations individuelles sont plus coûteuses et aussi plus sujettes aux erreurs que l'attribution d'autorisations d'accès à base de rôles.

Augmentation de l'efficacité : en réduisant la charge et le risque d'erreurs, l'efficacité du système informatique et des autres collaborateurs augmente. Les modifications manuelles, le

traitement des erreurs, les délais d'attente ainsi que les demandes individuelles de droits disparaissent.

Sécurité : les droits d'accès sont définis exclusivement selon le concept de rôles, ce qui évite la sur-autorisation de collaborateurs individuels. Ceci correspond au principe du PoLP, c'est-à-dire le principe du moindre privilège.

Transparence : la désignation des rôles est le plus souvent aisément compréhensible, ce qui renforce la transparence et la compréhension par les utilisateurs. [14]

6.2.7 Les inconvénients de contrôle d'accès RBAC

Élaboration complexe : le transfert des structures de l'organisation dans le modèle RBAC requiert beaucoup de travail.

Attribution temporaire : si un utilisateur a besoin temporairement de droits d'accès étendus, il est plus facile d'oublier l'attribution avec le RBAC qu'avec une attribution individuelle de droits.

Utilisation : pour les petites entreprises, l'élaboration et la maintenance des rôles nécessitent davantage de travail que la répartition des droits individuels. Par conséquent, le modèle RBAC est utilisé seulement à partir d'un certain nombre de rôles et de collaborateurs. Et pourtant, même pour les grandes entreprises, le Role Based Access Control a pour inconvénient de multiplier les rôles. Si une entreprise a dix services et dix rôles, il en résulte déjà 100 groupes différents. [14]

6.3 Contrôle d'accès Basé Sur Les Attributs

6.3.1 Définition

Le modèle de contrôle d'accès basé sur les attributs (ABAC) utilise des mécanismes tels que les listes de contrôles d'accès (ACL) qui contiennent les attributs des sujets ainsi que les opérations autorisées sur une ressource donnée. Lorsqu'un attribut correspond à celui figurant dans la liste ACL, le sujet se voit attribuer le droit d'effectuer sur la ressource les opérations mentionnées pour cet attribut dans la liste ACL. [15]

6.3.2 Les avantages de contrôle d'accès ABAC

Élaboration de politiques granulaires mais flexibles : Le principal avantage d'ABAC est sa flexibilité. Essentiellement, la limite de l'élaboration des politiques réside dans les attributs dont il faut tenir compte et dans les conditions que le langage informatique peut exprimer.

ABAC permet au plus grand nombre de sujets d'accéder à la plus grande quantité de ressources sans obliger les administrateurs à spécifier les relations entre chaque sujet et objet. Prenez les étapes suivantes à titre d'exemple :

Lorsqu'un sujet rejoint une organisation, un ensemble d'attributs de sujet lui est attribué (par exemple, John Doe est consultant pour le service de radiologie).

Un objet, une fois créé, se voit attribuer ses attributs (par exemple, un dossier avec des fichiers de test d'imagerie cardiaque pour les patients cardiaques).

L'administrateur ou le propriétaire de l'objet crée ensuite une règle de contrôle d'accès (par exemple, "Tous les consultants du service de radiologie peuvent afficher et partager des fichiers de test d'imagerie cardiaque pour les patients cardiaques").

Les administrateurs peuvent modifier davantage ces attributs et règles de contrôle d'accès pour répondre aux besoins d'une organisation. Par exemple, lors de la définition de nouvelles politiques d'accès pour des sujets externes tels que des sous-traitants et des fournisseurs, ils peuvent le faire sans modifier manuellement chaque relation sujet-objet. ABAC permet une grande variété de situations d'accès avec peu de supervision administrative.

Compatibilité avec les nouveaux utilisateurs : Avec ABAC, les administrateurs et les propriétaires d'objets peuvent créer des politiques permettant à de nouveaux sujets d'accéder aux ressources. Tant que les nouveaux sujets se voient attribuer les attributs nécessaires pour accéder aux objets (par exemple, tous les consultants du service de radiologie se voient attribuer ces attributs), il n'est pas nécessaire de modifier les règles existantes ou les attributs des objets.

Les modèles ABAC permettent aux organisations d'être agiles lors de l'intégration de nouveaux employés et de l'activation de partenaires externes.

Sécurité et confidentialité strictes : Grâce à son utilisation d'attributs, ABAC permet aux décideurs de contrôler de nombreuses variables situationnelles, en sécurisant l'accès sur une base fine. Dans un modèle RBAC, par exemple, les équipes RH peuvent toujours avoir accès aux informations sensibles des employés, telles que les données de paie et les informations personnellement identifiables. Avec ABAC, les administrateurs peuvent mettre en place des restrictions d'accès intelligentes qui tiennent compte du contexte. Par exemple, les employés des RH peuvent n'avoir accès à ces informations qu'à certains moments ou uniquement pour le personnel de la succursale concernée.

En conséquence, ABAC permet aux organisations de combler efficacement les failles de sécurité et de respecter la confidentialité des employés, tout en respectant efficacement les exigences de conformité réglementaire. [16]

6.3.3 Inconvénients de contrôle d'accès ABAC

Complexe à concevoir et à mettre en œuvre : ABAC peut être difficile à faire décoller. Les administrateurs doivent définir manuellement les attributs, les attribuer à chaque composant et créer un moteur de politique central qui détermine ce que les attributs sont autorisés à faire, en fonction de diverses conditions ("si X, alors Y"). L'accent mis par le modèle sur les attributs rend également difficile l'évaluation des autorisations disponibles pour des utilisateurs spécifiques avant que tous les attributs et règles ne soient en place.

Cependant, bien que la mise en œuvre d'ABAC puisse prendre beaucoup de temps et de ressources, l'effort est payant. Les administrateurs peuvent copier et réutiliser des attributs pour des composants et des postes d'utilisateurs similaires, et l'adaptabilité d'ABAC signifie que le maintien des politiques pour les nouveaux utilisateurs et les situations d'accès est une affaire relativement "sans intervention". [16]

6.4 Comment choisir entre RBAC et ABAC

Pour les petites et moyennes entreprises, RBAC est une alternative plus simple à ABAC. Chaque utilisateur se voit attribuer un rôle unique, avec des autorisations et des restrictions correspondantes. Lorsqu'un utilisateur passe à un nouveau rôle, ses autorisations sont remplacées par celles du nouveau poste. Cela signifie que, dans les hiérarchies où les rôles sont clairement définis, il est facile de gérer un petit nombre d'utilisateurs internes et externes.

Cependant, lorsque de nouveaux rôles doivent être créés manuellement, cela n'est pas efficace pour les grandes organisations. Une fois les attributs et les règles définis, les politiques d'ABAC sont beaucoup plus faciles à appliquer lorsque les utilisateurs et les parties prenantes sont nombreux, tout en réduisant les risques de sécurité.[17]

RBAC	ABAC
Une petite ou moyenne entreprise	Une grande organisation avec de nombreux utilisateurs
Politiques de contrôle d'accès sont large	Des capacités de contrôle d'accès approfondis et spécifiques
Les rôles de votre organisation sont clairement définis, et le nombre d'utilisateurs externes est peu	Garantir le respect de la confidentialité et de la sécurité

Tableau 1. Comparaison entre RBAC et ABAC

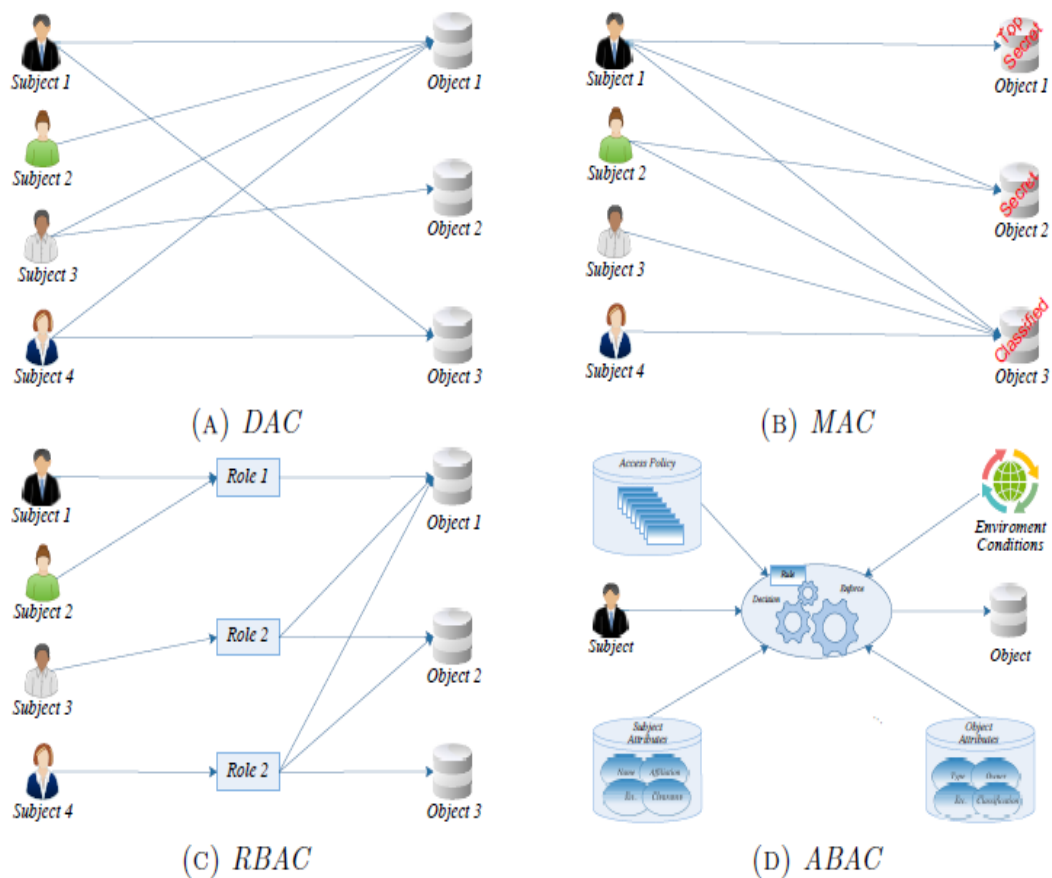


Figure 5. Les différents architecteur de contrôle d'accès.

7 Politique de contrôle d'accès

Les contraintes qui régissent les accès aux ressources d'un système peuvent être de nature statique ou dynamique. On distingue ainsi deux types de politiques : les politiques de contrôle d'accès statique et les politiques de contrôle d'accès dynamique.

7.1 Politique de contrôle d'accès statique

Pour un système d'information donné, une politique de contrôle d'accès statique est caractérisée par le fait que son état ne change pas par rapport à l'évolution dynamique du système, car elle comporte seulement des contraintes statiques. Celle-ci peut être mise à jour pour refléter divers changements dans l'organisation (par exemple, un changement d'affectation dans une organisation qui entraîne une augmentation des privilèges d'un utilisateur). L'initiation d'une action par un utilisateur déclenche l'évaluation de l'état de la politique. En fonction des autorisations accordées par la politique dans son état courant, l'exécution de l'action est permise ou pas. Dans la plupart des implémentations, une politique de contrôle d'accès statique associe les utilisateurs du système à leurs privilèges.[18]

7.2 Politique de contrôle d'accès dynamique

Pour un système d'information donné, une politique de contrôle d'accès dynamique possède plusieurs états, car elle est associée à l'évolution du système. L'autorisation de l'exécution d'une action est basée sur l'évaluation de l'état courant du système et sur la définition même de la politique. L'état courant est mis à jour lors de chaque exécution d'une action contrôlée. La version élémentaire de ce type de politiques de contrôle d'accès utilise un historique des actions exécutées par le système d'information qui est mise à jour par le gestionnaire de mise en œuvre de la politique. Les langages formels qui supportent les traces d'événements, comme les langages basés sur une algèbre de processus, se prêtent bien à l'expression de contraintes dynamiques.[18]

8 Comparaison entre les différentes méthodes de contrôle d'accès :

Attribut /Type de contrôle d'accès	DAC	MAC	RBAC	AB ² AC
Facilité d'utilisation ou commodité	Haute	Varie	Haute	Haute
Performance	Bas	Varie selon les niveaux de sécurité	Haute	Haute
Réutilisabilité	Oui	Non	Oui	Oui
Défaillance ponctuelle	Echec de l'autorisation	Moins	Moins	-
Echec de l'authentification	Moins	Varie	Moins	Moins

Tableau 2. Comparaison entre les différentes approches de contrôle d'accès.

Dans le tableau au dessus on récapitule la comparaison entre les différents approches de contrôle d'accès par rapport au quelques critères on trouve que le DAC, RBAC et ABAC sont les plus facile a utilisée par rapport aux MAC aussi sont réutilisable par rapport au MAC on trouve aussi que le RBAC et le ABAC sont les plus performant par rapport au DAC et la performance de MAC varie selon le niveau de la sécurité.

Selon la comparaison nous voyons que les faiblesses de DAC est l'échec de l'autorisation, mais le MAC et RBAC ils ont moins de faiblesses. A la fin on trouve que l'échec de l'authentification est peut dans RBAC, DAC et RBAC.

9 LE CHOIX DU BON CONTRÔLE D'ACCÈS

Lorsqu'il s'agit de choisir le bon contrôle d'accès, il n'y a pas d'approche exacte. La sélection dépend de plusieurs facteurs et vous devez choisir celui qui convient à vos besoins et exigences uniques.

Par exemple, si nous cherchons la flexibilité et la facilité d'utilisation, optez pour un système de contrôle d'accès discrétionnaire (DAC). Pour une sécurité maximale, un système de contrôle d'accès obligatoire (MAC) serait préférable. Si vous souhaitez un équilibre entre

sécurité et facilité d'utilisation, vous pouvez envisager le contrôle d'accès basé sur les rôles (RBAC).

Certains facteurs à prendre en compte incluent la nature de votre propriété, le nombre d'utilisateurs sur le système et les procédures de sécurité existantes au sein de l'organisation. Voici quelques questions fondamentales que vous devez vous poser avant de prendre une décision.

10 Conclusion

Dans ce chapitre nous avons présenté quelque notion de sécurité de système d'information tel que la confidentialité, l'authentification et la disponibilité, Ainsi la sécurité des documents XML et le fameux standard XACML. Ensuite nous parlons sur les différents modèles de contrôle d'accès existants qui ont été conçus pour assurer la sécurité des données et gères les droits d'accès en prennent le DAC, MAC, RBAC, ABAC. Ainsi la politique de contrôle d'accès. Il est difficile de considérer qu'un modèle de contrôle d'accès est meilleur qu'un autre, cela dépend essentiellement du domaine d'application et du type de l'organisation qui le met en œuvre.

La sécurité devant être assurée du front-end au back-end, nous avons jugé utile d'aborder la notion de middleware car c'est le niveau intermédiaire entre les deux parties justement citées, ça fera l'objet du prochain chapitre.



***Chapitre 2 : Généralité sur
Middleware***

1 Introduction

Dans les environnements informatiques qui se trouvent partout et en tous lieux, les systèmes doivent percevoir les informations de contexte dynamiques et changeantes et agir en conséquence. Par conséquent, pour répondre à ces exigences, il est nécessaire de séparer la capacité des dispositifs informatiques de détecter les aspects de l'environnement local de l'utilisateur et les dispositifs informatiques eux même, de les interpréter et d'y réagir et le développement d'applications, c'est-à-dire de séparer le traitement de données de détection sous-jacent et les applications de haut niveau sensible au contexte. Ce qu'il faut c'est une couche intermédiaire à savoir un middleware conscient du contexte. [19]

Certaines des techniques et des outils les plus réussis conçus pour améliorer la réutilisation du centre logiciel sur les middlewares qui aide à gérer la complexité et l'hétérogénéité des applications distribuées. Ce middleware de calcul distribué (désormais appelé simplement middleware) fournit des logiciels réutilisables qui font un pont fonctionnel entre (1) les exigences fonctionnelles de bout en bout des applications et (2) les systèmes d'exploitation de niveau inférieur, les piles de protocoles de réseau, bases de données et dispositifs matériels. [20]

2 Définition du middleware

Le terme a été introduit au début des années 1980. Il englobe des solutions logicielles complexes qui modernisent les anciens systèmes, généralement les ordinateurs centraux, au moyen de nouvelles fonctions, comme des logiciels et des composants d'application. Initialement, il était uniquement utilisé pour étendre la couche séparant les couches réseau et application. Par la suite, son utilisation s'est étendue pour servir de couche au-dessus du système d'exploitation et de la couche réseau, et sous la couche d'application. Cela signifie que le middleware pourrait désormais faciliter la communication générique entre le composant d'application et le réseau distribué. Plusieurs définitions ont été utilisées en fonction du domaine de recherche. Certains décrivent le middleware comme la couche qui attache le logiciel par différents composant du système d'autres comme l'intégration des connexions réseau tolérant les pannes et vérifiant les erreurs etc. [21]

3 Origines et objectifs

Les middlewares sont des logiciels de connectivité qui se composent d'un ensemble de services habilitants qui permettent à plusieurs processus sur une ou plusieurs machines

d'interagir sur un réseau. Ils sont essentiels à la migration des applications mainframe vers les applications client/serveur et à la communication entre plateforme hétérogènes. Cette technologie a évolué au cours des années 1990 pour assurer l'interopérabilité à l'appui du passage aux architectures client/serveur. [22]

Les initiatives de middleware les plus médiatisées sont :

- Open Software Foundation Distributed Computing Environment (DCE).
- Object Management Group's Common Object Request Broker Architecture (CORBA).
- Microsoft's COM/DCOM (Component Object Model).

4 Les cas d'utilisation des middlewares

Les solutions de middleware sont une large catégorie qui inclut tous les types de services, des serveurs Web aux systèmes d'authentification et outils de courriel. Voici les principaux cas d'utilisation du middleware en développement moderne :

4.1 Développement de jeux

Les développeurs de jeux utilisent les middlewares comme moteur de jeu. Le logiciel doit communiquer avec divers serveurs d'images, d'audio et de vidéo ainsi qu'avec des systèmes de communication. Le moteur de jeu facilite cette communication et rend le développement du jeu plus efficace. [23]

4.2 Matériel électronique

Les ingénieurs utilisent des middlewares pour intégrer divers types de capteurs à leurs contrôleurs. La couche des middlewares permet aux capteurs de communiquer avec le contrôleur par le biais d'un cadre de messagerie commun. [24]

4.3 Développement de logiciels

Les middlewares offrent une interface de programme d'application (API) standard pour gérer l'entrée et la sortie de données requises du composant. La liaison interne avec le composant est cachée à l'utilisateur. Les développeurs utilisent des API pour intégrer différents composant logiciels à d'autres applications. [25]

4.4 Transmission de données

Les applications logicielles utilisent des middlewares pour envoyer et recevoir des flux de données de manière fiable. Les flux de données consistent en une transmission à grande vitesse de données continues. Ils sont importants pour une diffusion vidéo et audio fiable. [26]

4.5 Applications distribuées

Une application distribuée (ou application répartie) est une application composée d'un ensemble de processus possédant leur propre mémoire locale et communiquant par passage de messages. [27]

5 L'architecture middleware

La plupart des middlewares suivent la conception de l'architecture orientée service (SOA) ou sont conçus comme une solution de plateforme-service (PaaS). SOA est un style architectural qui tente de réaliser des applications logicielles vaguement couplées qui interagissent entre elles pour fonctionner dans son ensemble. Il est adopté par les organisations qui tentent de découpler toutes leurs unités opérationnelles, en fonction de l'intégration et de la réutilisation pour les opérations quotidiennes. SOA permet aux organisations d'utiliser les investissements dans les applications et les systèmes existants. [28]

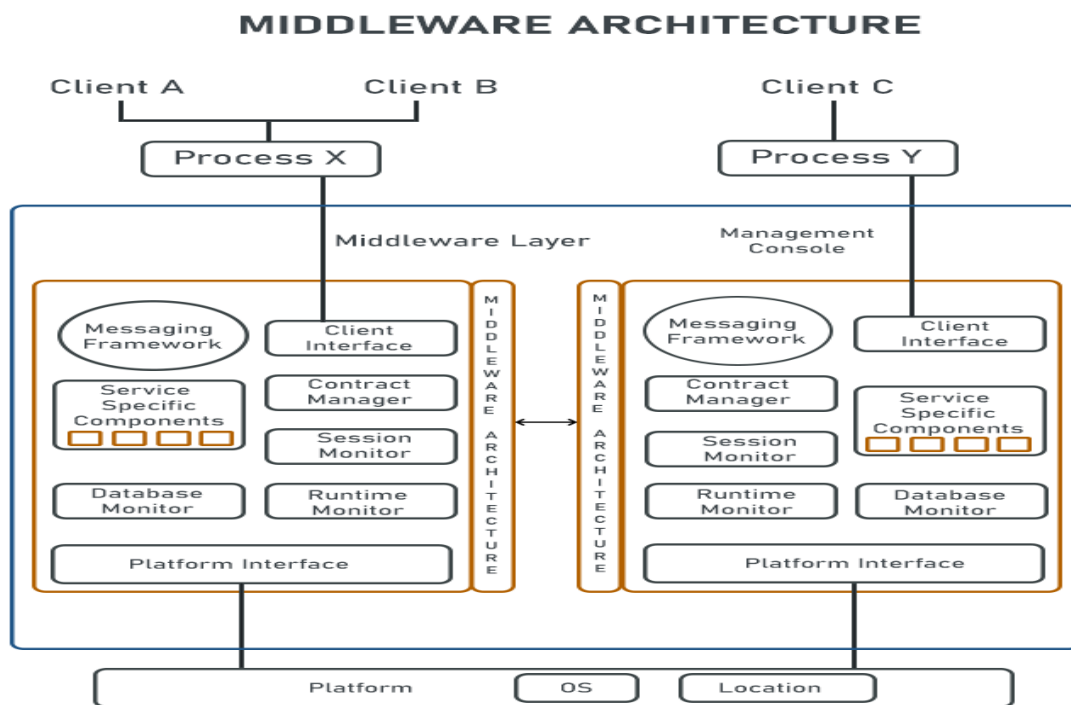


Figure 6. Architecture du middleware [28]

5.1 Console de gestion

La console de gestion fournit un support déclaratif pour caractériser de façon concise les activités, les règles logicielles et les configurations du système middleware. [29]

5.2 Interface client

L'interface client est la partie extérieure du logiciel middleware qui communique avec les applications. Afin d'accéder aux services hébergés dans des réseaux « éloignés », systèmes de stockage ou interagir avec d'autres applications, la demande de l'application cliente doit être acheminée par un ou plusieurs nœuds de passerelle, cette fonction est fournie par l'interface client pour être utilisée par les développeurs. [30]

5.3 Cadre de messagerie commun

Les middlewares nécessitent des services de messagerie pour communiquer avec les services, les applications et les plateformes. La plupart de ces cadres s'appuient sur des standards existants tels que le protocole d'accès simple aux objets (SOAP), le transfert d'état représentatif (REST) ou la notation Javascript des objets (JSON). Ce cadre doit être soigneusement conçu, en gardant à l'esprit les caractéristiques existantes et prévues. La communication elle-même passe par des services Web ou des interfaces de programmation d'application (API). Les API et les services Web utilisent des normes comme JSON pour recevoir, envoyer et transférer de l'information entre les différentes couches. [31]

5.4 Interface interne de logiciel

Les composants de middleware utilisent l'interface interne pour fonctionner de manière cohérente avec leur propre protocole. Elle agit comme une colle de logiciel liant les différents composants entre eux. [32]

5.5 Interface de plateforme

L'interface de middleware contient des composants logiciels qui fonctionnent avec différents types de systèmes d'exploitation garantissant ainsi que le programme de middleware est compatible avec diverses plateformes. [33]

5.6 Gestionnaire de contrat

Le gestionnaire de contrat définit les règles d'échange de données dans le système middleware. Il observe et consigne les événements d'interaction réalisés à l'aide du middleware, il envoie une alerte ou une exception lorsque ces échanges ne respectent pas les règles spécifiques. Par exemple, le gestionnaire de contrat renverra une exception si les données échangées sont des mots alors que des nombres sont attendus. [34]

5.7 Gestionnaire de session

Le gestionnaire de session établit un canal de communication entre les applications et le middleware. Un gestionnaire des communications est chargé d'activer les séances de découverte des services et de récupérer les attributs des services disponibles dans l'environnement. Ce composant coopère avec le précédent pour le stockage des informations de maintenance. [35]

5.8 Gestionnaire de base de données

Le gestionnaire de base de données contrôle les connexions à la base de données, en modifiant l'ajout, la modification et la suppression des données en fonction du service de base de données utilisé. Étant donné que cela varie considérablement selon le type de base de données utilisée, l'endroit où elle se trouve et la sensibilité des données, la sécurité est une considération importante pour cette composante. [36]

5.9 Moniteur d'exécution

Le moniteur d'exécution détecte et signale les activités inhabituelles aux développeurs en surveillant continuellement les mouvements des données dans le middleware. [37]

6 Fonctionnement du middleware

Le middleware abstrait le processus de communication sous-jacent entre les composants. Cela signifie que l'application front-end communique uniquement avec le middleware et ne doit pas apprendre la langue des autres composants logiciels back-end. Il existe plusieurs types de middleware dans le marché implémentés différemment et utilisent des mécanismes de partage de ressources et de communication différents. [38] Ces fonctionnalités principales sont :

- Echange de données.
- Formatage de données.
- Fonctions de sécurité.
- Localisation de données.

7 Le rôle du middleware

Les middlewares sont utilisés pour favoriser l'interaction entre les différents aspects d'une application ou entre les applications elle mêmes. Parmi les avantages qu'on puisse tirer des middlewares [39] :

- La gestion facile de la complexité et de l'hétérogénéité des applications.
- La simplicité de la conception logicielle.
- La réutilisation de l'infrastructure logicielle.
- Permettre l'interopérabilité, la portabilité et l'intégration de composants utilisant différentes technologies.

8 Type du middleware

Les middlewares peuvent implémenter un ou plusieurs modèles de communication distribués. Ces modèles de communication appartiennent à différents niveaux d'abstraction et ne sont pas évidents pour classer et comparer. D'après [40] les middlewares peuvent être divisés en classe principales qui sont :

8.1 Passage de messages

Les expéditeurs peuvent envoyer des messages de façon asynchrone et poursuivre leur traitement, tandis que les destinataires attendent les messages et les reçoivent de façon synchrone. Les opérations d'envoi/réception peuvent être bloquantes, non bloquantes ou une combinaison des deux. Des interactions de niveau supérieur peuvent être construites au-dessus de ces primitives comme des invocations à distance. Les expéditeurs et les récepteurs sont couplés dans le temps et l'espace, car ils doivent fonctionner en même temps et l'expéditeur doit connaître l'adresse du destinataire. La transmission de messages n'assure généralement pas l'accès et la transparence de l'emplacement, car la formation et l'adressage des données sont explicites au niveau de l'application.

8.2 Middleware d'appel de procédure

Remote Procedure Call (RPC) est utilisé de manière synchrone ou asynchrone pour faire appel à des services à partir de systèmes distants.

8.3 Middleware orienté objets

Middleware Oriented Object (OOM) est basé sur Remote Method Invocation (RMI). Les applications utilisent un agent de requête d'objet ou un middleware de requête d'objet pour envoyer des requêtes sans savoir où l'application cible est hébergée.

8.4 Middleware de composant

Les middlewares de composants sont une extension aux middlewares d'objets distribués. Une entité distribuée n'est plus un objet unique, mais une agrégation d'objets avec des

dépendances explicites appelées composant. Les exemples sont OMG CORBA Component Model CCM, Sun Enterprise JavaBeans (EJB), Microsoft Distributed Component Object Model DCOM 19.

8.5 Middleware orienté message

C'est une infrastructure qui prend en charge la transaction de messages entre les éléments répartis. Message Oriented Middleware (MOM) inclut Message Queuing MQ, ces middlewares fournissent un ensemble fixe de primitives de messagerie asynchrones prédéfinies pour envoyer et recevoir des messages anonymement d'un expéditeur à un ou plusieurs destinataires via des canaux de communication indirects, explicites et nommés appelés files d'attente de messages.

8.6 Logiciels intermédiaires Publier-Souscrire

Dans les middlewares Publier-Souscrire, les récepteurs de messages appelés abonnés enregistrent leur intérêt dans le middleware pour recevoir un type particulier d'information généralement appelé événement. Quand un producteur de message appelé éditeur génère cette information, le middleware notifie c.-à-d. livrer l'événement désiré à tous les abonnés inscrits. Exemples de ces middlewares : Tuxedo, Apache ActiveMQ, CORBA Event and Notification services et JMS. Publier/Souscrire prend en charge la transparence d'emplacement, car les expéditeurs et les destinataires ne connaissent pas l'adresse physique des autres.

8.7 Middleware transactionnel

Les middlewares transactionnels, tels que les moniteurs de processus transactionnels, garantissent que chaque étape d'un processus transactionnel logiciel est exécutée correctement.

8.8 Middleware orienté base de données

Les middlewares de base de données facilitent l'accès aux systèmes de gestion de base de données (SGBD) comme Microsoft Access, les bases de données Oracle ou MySQL en utilisant des API standard comme l'Open Database Connectivity (ODBC), qui est indépendant des langages de programmation, des systèmes d'exploitation et du SGBD. SQL (StructuredQueryLanguage) est un langage de requête déclaratif avec des extensions de programmation procédurale qui est utilisé pour créer, insérer, mettre à jour, récupérer et supprimer des données dans le SGBD relationnel.

8.9 Mémoire partagée distribuée

Distributed Shared Memory (DSM) fournit un espace d'adresse logique global, qui est partagé par les producteurs et les consommateurs dans les espaces d'adresse distribués.

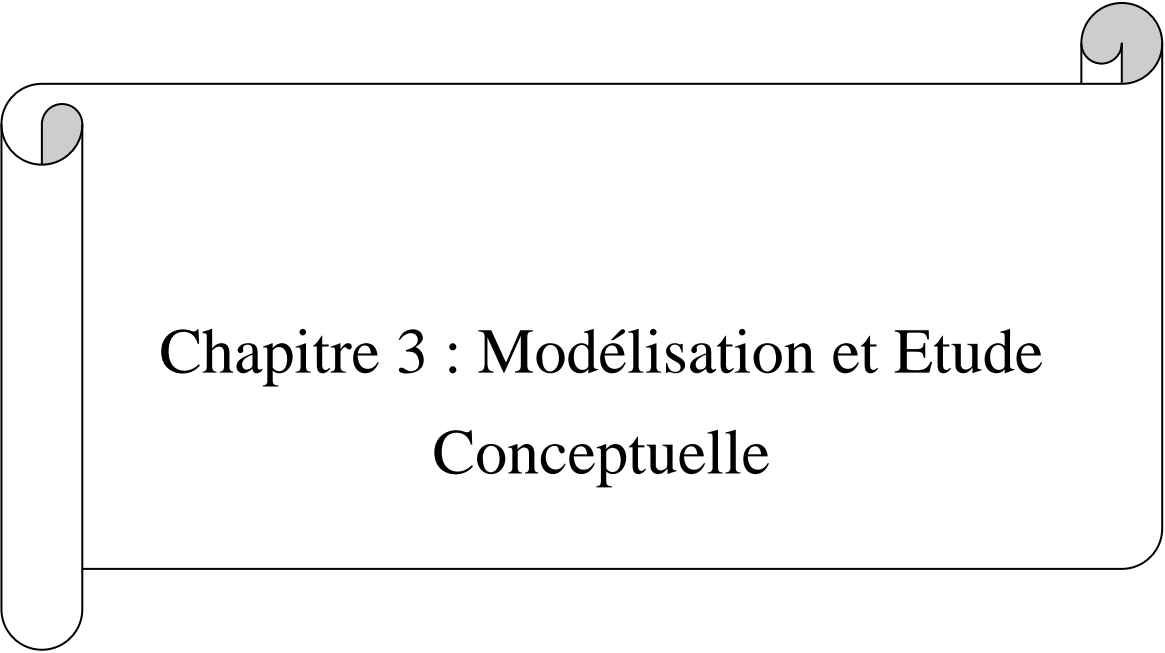
8.10 Middleware web

Les middlewares Web sont basés sur des technologies et des protocoles Internet tels que XML et HTTP pour offrir une vue uniforme des applications Web indépendamment des langages, des systèmes d'exploitation ou des bases de données. Les Services Web sont basés sur XML pour la définition des services utilisant le Langage de Définition des Services Web (WSDL) et le protocole de message utilisant le Simple Object Access Protocol (SOAP).

La séparation entre ces différents types de middlewares est plutôt floue. En effet, base de données middlewares peuvent avoir une orientation objet. Les middlewares objet comme CORBA fournissent des services pour la transaction, la file d'attente, la messagerie et publier-souscrire. Enfin, messages en file d'attente middlewares peuvent également prendre en charge transaction, DBMS et publier-souscrire. Par exemple, le Java Message Service (JMS) supporte à la fois les files d'attente de messages et les modèles publication-abonnement.

9 Conclusion

Les middlewares étant une technologie en croissance rapide offrant de nombreux avantages aux informaticiens pour le développement des applications distribuées et divers d'autres domaine de systèmes informatiques. Cependant, choisir les bons outils pour l'implémentation d'un middleware reste une décision importante pour la conception d'une telle technologie. Dans ce travail, on cherche à profiter des avantages de cette technologie pour l'orienter vers les contrôles d'accès et concevoir une solution médiatrice qui puisse être utilisée pour la sécurisation des données personnelles.



Chapitre 3 : Modélisation et Etude Conceptuelle

1 Introduction

Les contrôles d'accès sont les plus populaires et les plus utilisés pour assurer la sécurité, ils empêchent les personnes non autorisées d'accéder à certains endroits. Ce principe s'applique à la sécurité des personnes, des biens ou des données.

Disponibilité, fiabilité, flexibilité et interopérabilité sont des enjeux majeurs pour tous systèmes, les middlewares existent pour organiser et rationaliser tous ces mesures.

Basant sur l'étude des chapitres précédents, on va développer une solution médiatrice qui permet de contrôler l'accès des différents utilisateurs aux différentes sources de données indépendamment de leurs technologies utilisées. Ce chapitre commence par expliquer les raisons du choix du domaine médical pour ensuite établir le mécanisme de sécurité RBAC sur un exemple d'application de gestion médicale. Pour enfin démontrer les étapes du fonctionnement d'une combinaison des deux technologies définies précédemment et la tester.

2 Domaine d'application de la solution

E-sante ou information numérique sur la sante, est un domaine où de nombreuses informations sont manipulées et partagées régulièrement. Cette énorme base d'information, son stockage et partage sont très importants et difficiles en raison de la sensibilité du contenu et des facteurs restrictifs comme la sécurité et la confidentialité.

En raison de l'implication de plusieurs médecins de différents domaines d'expertise pendant un processus d'hospitalisation, une information devient facilement accessible et manipulable, sa modification, suppression ou son partage peut induire à une mauvaise utilisation de cette dernière. Par conséquent, son propriétaire peut se retrouver dans une situation menaçante ou même en danger en cas de mauvaise prescription de médicament ou de traitement.

3 Problèmes et objectifs

Un système E-sante composé de plusieurs départements implique le besoin de la circulation de l'information dans tous les services du système. En appliquant le modèle RBAC on réduit le flux de propagation des données personnelles des patients par besoin, une assistante n'est pas obligée de connaître ou de consulter les dossiers médicaux des patients, ce qui fait que dans chaque service ne circule que les informations nécessaires pour son fonctionnement.

Dans le cas précédent, la restriction n'est appliquée que pour les services externes. Certains malades souhaitent interdire à certains de leurs médecins d'accéder à certaines données qu'ils

précisent eux même. Notre objectif c'est de pouvoir mettre à disposition des patients une solution qui leur permette de spécifier leurs exigences, interdire et accorder l'accès par fonctionnalités et par données au demandeur d'accès.

4 Modélisation du RBAC

Un service appartenant à un système E-sante se compose de plusieurs individus ayant des rôles différents. Chaque individu peut effectuer des opérations spécifiques selon le rôle qu'il lui est attribué. C'est le concept du modèle RBAC, la figure ci-dessous représente sa modélisation:

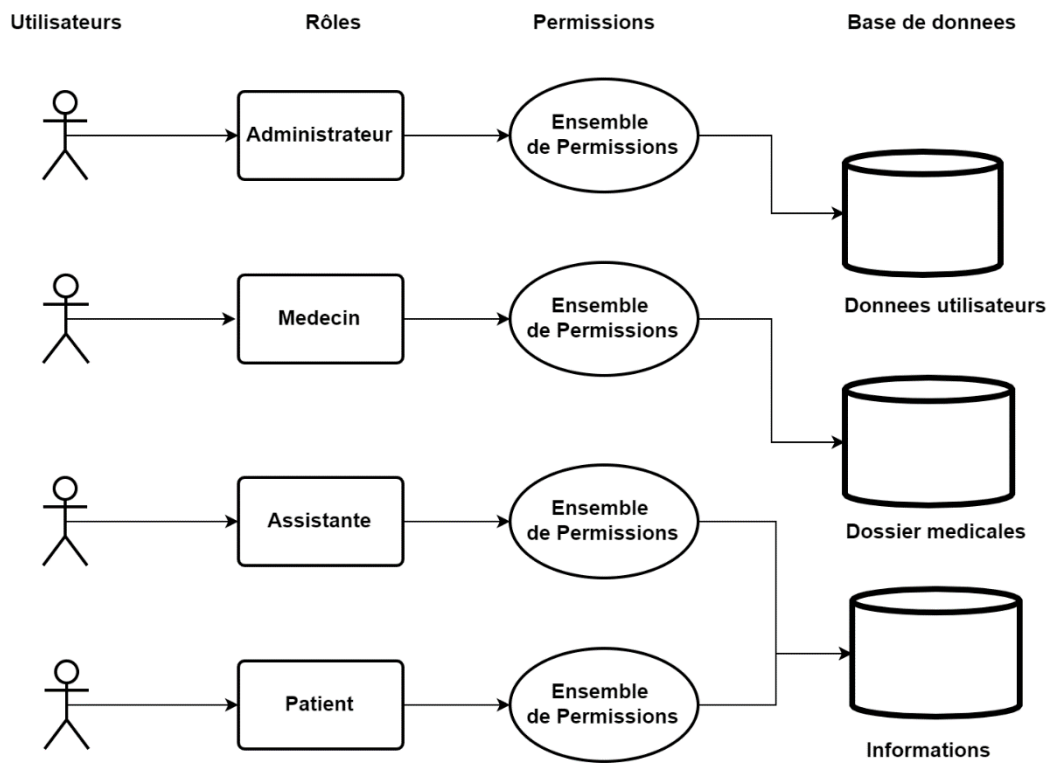


Figure 7. Modélisation du RBAC

4.1 Hiérarchie des rôles

Le concept de la hiérarchie des rôles permet aux individus de rang supérieur d'hériter les permissions de ceux du rang inférieur. Un administrateur pouvant être le chef du service peut effectuer tous les opérations permises aux médecins, assistantes et patients. La hiérarchie du service est représentée comme suit :

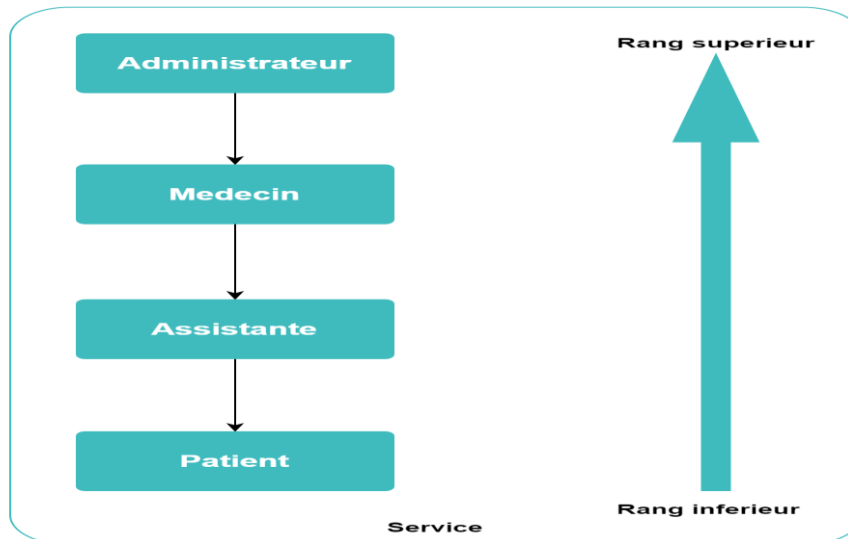


Figure 8. Hiérarchie des rôles

5 Description de la solution

L'inconvénient avec le modèle RBAC, comme expliqué dans la partie 3 de ce chapitre, c'est qu'un utilisateur ayant le rôle médecin peut accéder à toutes les données personnels de ces patients bien qu'il soit généraliste, ophtalmologue ou radiologue alors que certains préfèrent restreindre l'accessibilité à quelques données. Ils souhaiteraient que leurs informations comme leur analyses, maladies chroniques ou antécédents restent confidentielles pour tous à l'exception des médecins concernés par ces renseignements et dont le contenu est d'une extrême nécessité pour l'étude de leur santé et la prescription des traitements.

En répondant aux problèmes exposés, on a proposé une solution intermédiaire, un middleware, permettant d'atteindre un niveau de sécurité élevée que le modèle RBAC.

5.1 Schématisation de la solution

La figure suivante représente l'architecture générale de tout le système E-santé combiné avec la solution proposée :

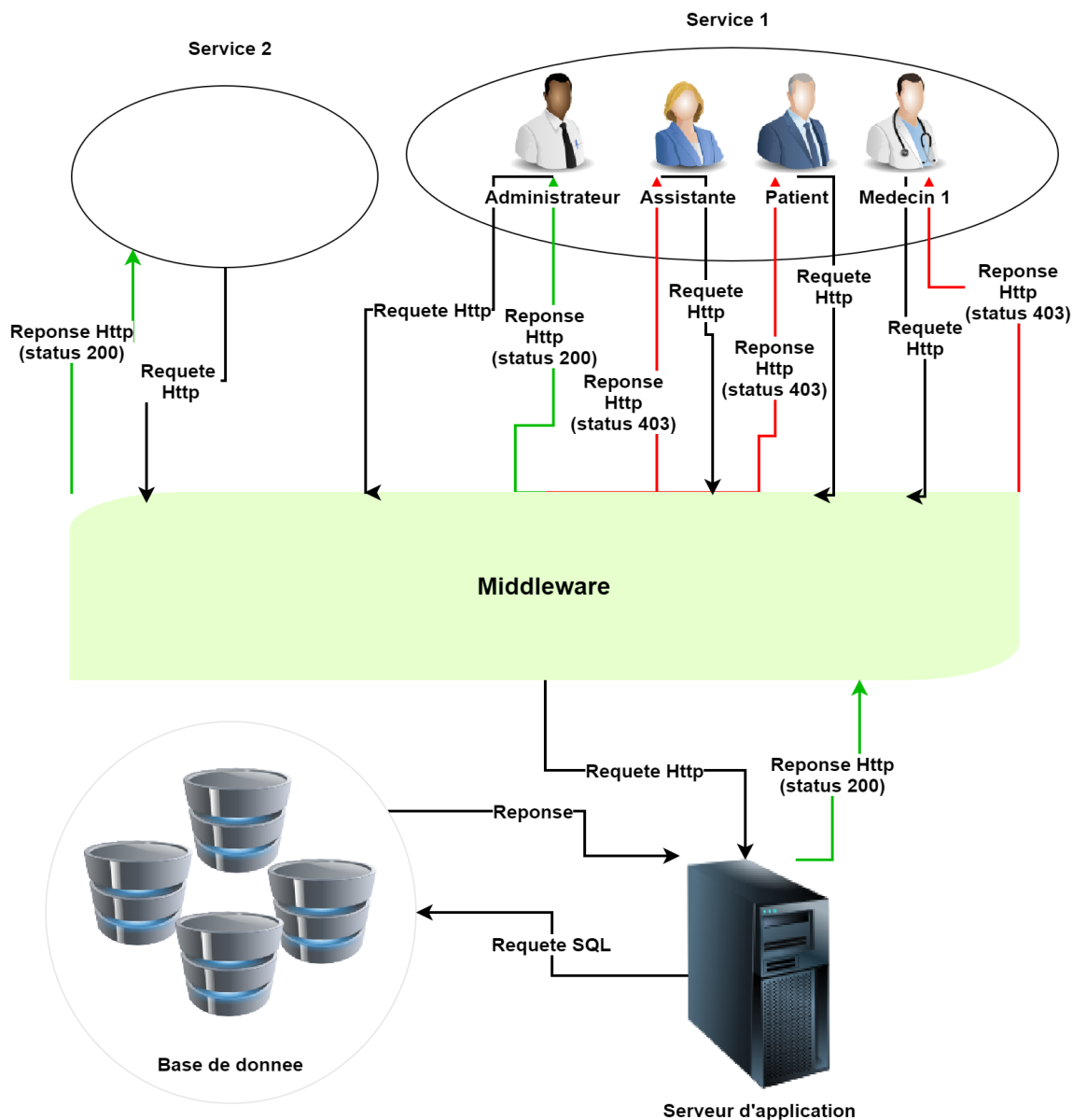


Figure 9. Architecture générale

5.1.1 Position du Middleware

Le middleware se situe entre l'application front-end du système et son back-end, il joue le rôle de courtier entre ces deux derniers. Toutes les demandes envoyées par les utilisateurs indépendamment de leurs rôles sont accueillies par le middleware pour ensuite effectuer un ensemble d'opérations avec les paramètres incluses dans celles-ci afin de décider de la continuation de ces demandes.

La suite des demandes est basée sur le résultat de ce traitement dont les règles ont été spécifiées précédemment par le propriétaire de la donnée sur la quelle vont agir ces demandes. Le résultat du traitement du middleware peut aboutir aux décisions suivantes :

- **Laisser passer la demande** : cette décision permet de poursuivre le processus d'exécution de la demande afin de recevoir le résultat souhaitée par l'utilisateur.
- **Ne pas laisser passer la demande** : cette décision permet d'ignorer le processus d'exécution de la demande et la rejeter puis renvoyer un message d'échec comme résultat à l'utilisateur.

5.1.2 Identification des rôles

Un ensemble d'opérations, à travers lesquelles les demandes aux serveurs sont envoyées, est mis à la disposition des utilisateurs en considération des rôles qui leurs sont attribués. Les rôles prédéfinis des utilisateurs sont les suivants :

- **Administrateur** : L'échelon le plus élevé de la hiérarchie, il possède l'accès à tous le système et la permission d'effectuer tous type d'opérations à l'exception de celles qui ne lui ont pas été spécifiées.
- **Médecin** : Le niveau sous administrateur, son accès est plus restreint que celui du niveau supérieur et ne peut effectuer les opérations sur un dossier patient que ce qui lui a été permis par son propriétaire. Il hérite aussi des permissions des rôles de plus bas niveau.
- **Assistante** : L'accès du rôle assistante est beaucoup plus restreint que ceux qui précédent, les opérations qui lui sont permises sont plus limitées, toute fois, ce rôle hérite des droits d'accès du patient.
- **Patient** : Le plus bas niveau de la hiérarchie du système est le rôle patient et ne peut hériter des droits de ces prédécesseurs.

5.1.3 Serveur d'application

Une architecture trois tiers s'utilise couramment pour construire une application web. Le navigateur (interface utilisateur) est le premier tiers. Vient ensuite le serveur d'application, tiers du milieu, qui se connecte avec la base de données qui est le troisième tiers.

Le serveur d'application est la machine qui s'occupe d'exécuter toutes les opérations effectuées par les utilisateurs, se charge d'interrogation de la base de données et de la récupération de ces derniers pour renvoyer les résultats.

Ce serveur utilise le protocole http pour la communication avec les utilisateurs, il reçoit les requêtes et renvoie les réponses http avec les informations demandée à l'intérieur.

La connexion avec la base de données se fait via le protocole TCP/IP, en-dessus de la couche transport, pour dialoguer et gérer les transactions, ces deux derniers recourent à un protocole appelé middleware. Il existe plusieurs types de middleware qui permette la communication avec une BDD, indépendamment de son type.

Dans notre application, on a utilise le type standard JDBC, qu'on va définir et expliquer en détails dans le chapitre suivant.

5.1.4 Systèmes de Gestion de base de données

Le système de gestion de base de données est l'entrepôt où sont stockés les dossiers médicaux des patients, les informations des utilisateurs et tout autre type de donnée.

Dans notre application, on a utilisé comme exemple le système de gestion de base de données MySQL.

5.2 Spécification des droits d'accès

Le patient est celui qui décide qui ou non peut consulter et manipuler telle ou telle donnée, il peut donc préciser les fonctionnalités qui peuvent être effectuée sur chacune de ces données personnelles. Ces spécifications sont définies dans des fichiers XML.

Le XML permet de structurer les informations dans des fichiers texte. On a utilise le XML pour faciliter le traitement informatique toute en conservant un support texte lisible et éditable.

5.2.1 Construction des fichiers XML

Le middleware doit extraire les règles de control d'accès depuis des fichiers XML propre pour chaque patient. Chaque patient possède un fichier XML dont il spécifie lui-même les règles, ce document est construit juste après l'enregistrement du patient à l'application contenant déjà les fonctionnalités par défaut autorisée par tous, et modifié à la demande de son propriétaire.

Un formulaire regroupant toutes les opérations possibles d'être effectuées par un utilisateur avec le rôle médecin est mis à la disposition du patient pour pouvoir designer ses exigences pour chaque médecin qu'il consulte qui seront traduit à des règles de control d'accès.

5.2.1.1 Définition de la structure XSD

Le fichier XML du patient est composé d'éléments complexes qui contiennent les informations nécessaires au middleware pour les traitements des requêtes. Bien que le contenu de chaque fichier diffère de l'autre, leur construction suit la même structure.

La structure de construction du fichier XML du patient est définie dans un autre fichier XSD. Ce fichier contient la définition qui permet à tous les fichiers XML de suivre la même structure. Son contenu est le suivant :

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Donnee" type="xs:string" />
  <xs:element name="NomFonction" type="xs:string"/>
  <xs:element name="Fonction" >
    <xs:complexType mixed="true">
      <xs:sequence>
        <xs:element ref="NomFonction" minOccurs="1" maxOccurs="1"/>
        <xs:element ref="Donnee" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Permission">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Fonction" minOccurs="1" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="utilisateur_id" type="xs:int"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="Patient">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Permission" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="patient_id" type="xs:int"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

On a constitué deux types d'éléments :

- **Élément simple** : Ce type est constitué d'élément de type primitif et possède un attribut "name" qui représente son nom pour le distinguer des autres éléments. Les éléments simples que contient le fichier XSD sont :
 - Donnee de type chaîne de caractère.
 - NomFonction de type chaîne de caractère.
- **Élément complexe** : Ce type est constitué de plusieurs éléments simples définis précédemment possédant un attribut "name" représentant le nom de cet élément pour le distinguer des autres. Les éléments de type complexe que contient le fichier XSD sont :
 - Fonction : constitué de l'élément "NomFonction" qui ne peut apparaître qu'une seule fois grâce aux attributs "minOccurs=1" et "maxOccurs=1", peut aussi contenir une infinité d'éléments "Donnee" grâce à l'attribut "maxOccurs=unbounded" ou aucun grâce à l'attribut "minOccurs=0". L'utilisation de l'attribut "mixed=true" force les deux types d'éléments précédent d'apparaître dans l'ordre de leur définition c.à.d. "NomFonction" puis "Donnee" (s'il existe).
 - Permission : constitué d'au moins un élément de type complexe "Fonction" jusqu'à une infinité grâce aux attributs "minOccurs=1" et "maxOccurs=unbounded". On lui a aussi défini un attribut qu'on a nommé "utilisateur_id" de type primitif entier.
 - Patient : peut être constitué d'une infinité d'éléments de type complexe "Permission". Un attribut "patient_id" de type primitif lui a été défini.

5.2.1.2 Illustration de la structure XSD

Une vision de la structure est toujours bien compréhensible qu'un texte. La figure ci-dessous illustre la structure du fichier XML présenté dans 5.2.1.1 en tant que schéma :

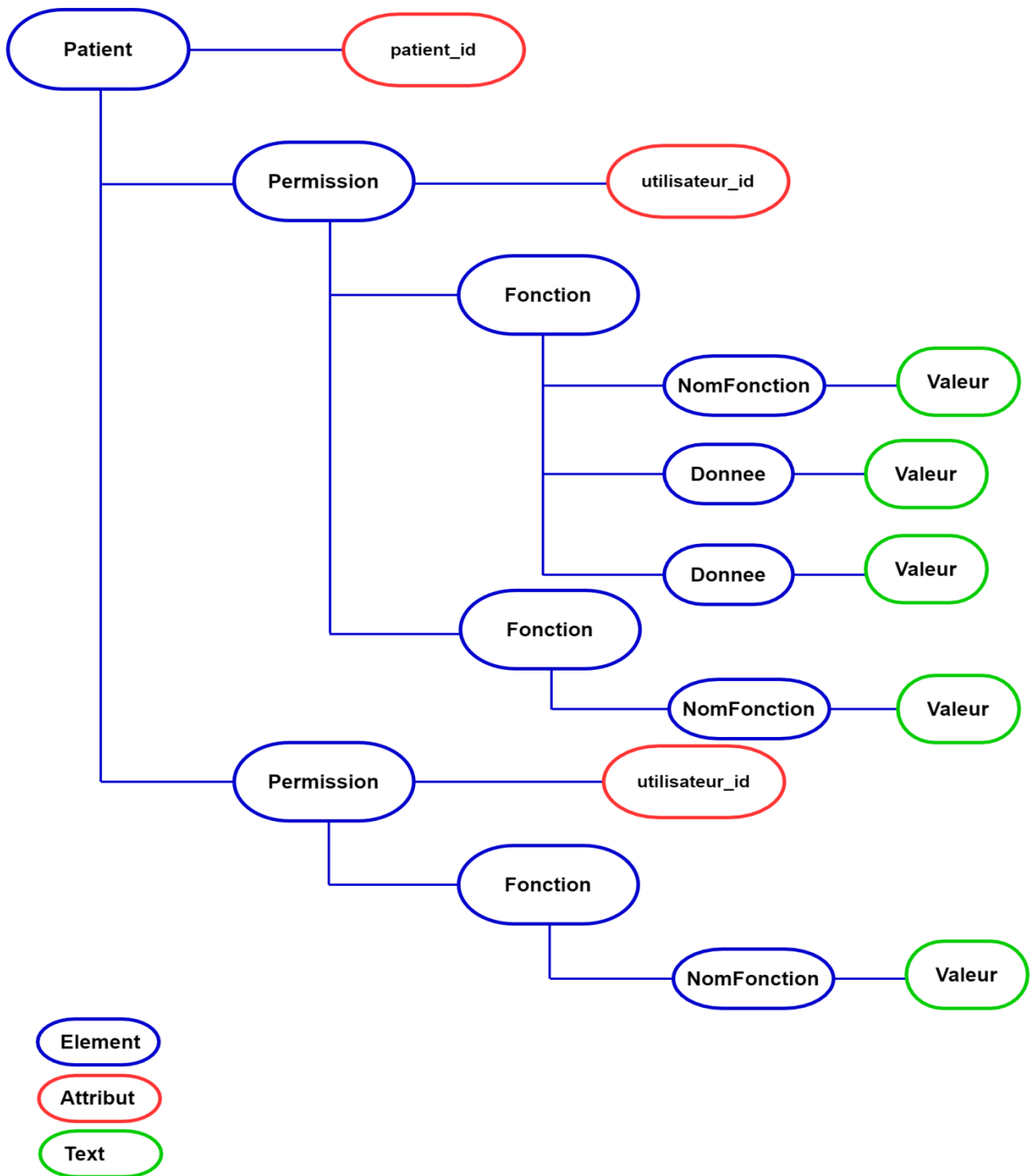


Figure 10. Illustration de la structure XML

Une structure XML est composée d'un élément racine et de plusieurs sous éléments dont chacun peut contenir d'autre sous éléments. Voici une explication de chacun des éléments

présentés dans la figure 10 et la description de leur part de participation dans le travail effectué par le middleware :

❖ **Patient** : L'élément racine représente le propriétaire du fichier courant, que le middleware peut identifier avec l'attribut **patient_id** dont la valeur est l'identifiant attribué au patient lors de son enregistrement à l'application. Grâce à l'élément `<Patient patient_id=" ? ">` le middleware peut déterminer qu'un tel document appartient à un tel patient.

❖ **Permission** : L'élément fils de l'élément "Patient" (sous-element). Il contient l'ensemble des opérations qui ont été permise par patient_id à un certain utilisateur identifiable avec l'attribut de cet élément **utilisateur_id** dont la valeur lui a été attribuée lors de son enregistrement à l'application. L'élément `<Permission utilisateur_id=" ?">` permet au middleware de savoir qui peut effectuer les fonctionnalités qu'il contient.

❖ **Fonction** : Le sous-élément de "Permission", il représente les informations de la fonctionnalité à être effectuée par **utilisateur_id**.

❖ **NomFonction** : Cet élément contient le nom de la fonctionnalité à être exécutée.

❖ **Donnee** : L'élément Donnée contient la valeur des champs sur lesquelles les fonctionnalités vont opérer.

5.2.1.3 Pseudo-Algorithmme

Depuis le formulaire de spécification du patient, les opérations et les données désignées sont représentée en texte clair et seront envoyées ensuite au middleware par une méthode POST. Cette requête sera décortiquée par le middleware pour en tirer les informations qu'elle contienne afin d'effectuer le traitement de l'algorithme suivant :

Variables

Patient_id: Chaîne de caractère ;
Donnees : Liste<Chaîne_de_caractere> ;
MapFonction : HashMap<Chaîne_de_caractere, Donnees> ;
MapPermission : HashMap<Chaîne_de_caractere, MapFonction> ;
ModeleSpecification : MapPermission ;

Début

```
//instancier un nouveau Document en utilisant JDOM
Document document= CreerUnDocumentXML() ;

//création de l'élément racine Patient
Element Patient= CreerElement('Patient') ;

//création des attributs de l'élément racine
Attribut xmlns:xs= CreerAttribut('xmlns:xs') ;
Attribut schemaxsd= CreerAttribut('xs:noNamespaceSchemaLocation') ;
Attribut patient_id= CreerAttribut('patient_id') ;

//Ajouter des valeurs aux attributs de l'élément racine
AjouterValeurAttribut(xmlns:xs, 'http://www.w3.org/2001/XMLSchema-instance') ;
AjouterValeurAttribut(schemaxsd, 'Localisation du fichier schema.xsd') ;
AjouterValeurAttribut(patient_id, Patient_id) ;
AjouterAttribut(Patient, xmlns:xs) ; AjouterAttribut(Patient, schemaxsd) ;
AjouterAttribut(Patient, patient_id) ;
/*Création des éléments permission
* Les itérations sur les HashMap se font par Clé
* /
Pour Cle_id = première clé de MapPermission à dernière clé de MapPermission faire :
    Element Permission= CreerElementPermission(document, Cle_id) ;
    MapFonction MF=InstancierVariableMapFonction() ;
    MF=RecupererLaValeurDeMapPermission(Cle_id, MapPermission) ;

/*Création des éléments Fonction
*Itération sur la HashMap MapFonction MF
* /
Pour Cle_f= première clé de MF a dernière clé de MF faire :
    Donnees D= InstancierVariableDonnees() ;
    Element Fonction= CreerElementFonction(document, Cle_f, D) ;
    //Ajouter un sous élément Fonction à l'élément Permission
    AjouterUnElement(Permission, Fonction) ;
FinPour ;
//Ajouter un sous élément Permission à l'élément Patient
AjouterUnElement(Patient, Permission) ;
FinPour ;
EnregistrerXML(document) ;
```

Fin.


```
//Les procédures et les fonctions
```

```
Fonction CreerUnDocumentXML() {
```

```
  Début
```

```
    Retourner un nouveau Document ;
```

```
  }
```

```
Fonction CreerElement(nomElement : Chaîne de caractère) : Element {
```

```
  Début
```

```
    Créer un nouveau élément nommé nomElement ;
```

```
    Retourner nomElement ;
```

```
  Fin.
```

```
}
```

```
Fonction CreerAttribut(nomAttribut : Chaîne de caractère ) : Attribut {
```

```
  Début
```

```
    Créer un nouveau attribut nommé nomAttribut ;
```

```
    Retourner nomAttribut ;
```

```
  Fin.
```

```
}
```

```
Procédure AjouterValeurAttribut(attribut : Attribut, valeur :Chaîne de caractère) {
```

```
  Début
```

```
    attribut.valeur= valeur ;
```

```
  Fin.
```

```
}
```

```
Procédure AjouterAttribut(elmt : Element, attr : Attribut) {
```

```
  Début
```

```
    elmt.attribut= attr ;
```

```
  Fin.
```

```
}
```

```
//Les procédures et les fonctions
```

```
Fonction CreerElementPermission(document : Document, Cle_id : Chaîne de caractère) : Element {
```

```
Variables var, nomAttribut : chaîne de caractère ;
```

```
    permission : Element ; utilisateur_id : Attribut ;
```

```
Début
```

```
    var= 'Permission' ; nomAttribut='utilisateur_id' ;
```

```
    permission=CreerElement(var) ; utilisateur_id=CreerAttribut(nomAttribut) ;
```

```
    AjouterValeurAttribut(utilisateur_id, Cle_id) ;
```

```
    AjouterAttribut(permission, utilisateur_id) ;
```

```
    Retourner permission ;
```

```
Fin.
```

```
}
```

```
Fonction InstancierVariableMapFonction() : MapFonction{
```

```
Début
```

```
    Retourner une instance de type MapFonction ;
```

```
Fin.
```

```
}
```

```
Fonction RecupererLaValeurDeMapPermission(Cle_Id : Chaîne de caractère, MP :MapPermission) :  
MapFonction{
```

```
Début
```

```
    Retourner la valeur dont la clé correspond à Cle_id depuis MP ;
```

```
Fin.
```

```
}
```

```
Fonction InstancierVariableDonnees() : Donnees {
```

```
Début
```

```
    Retourner une instance de type Donnees ;
```

```
Fin.
```

```
}
```

//Les procédures et les fonctions

Fonction **CreerElementFonction**(document : Document, Cle_f : Chaîne de caractère, D : Donnees) :
Element {

Variables

 nomFonction, donnee, fonction : Chaîne de caractère ; i : entier ;
 Fonction, NomFonction : Element ;

Début

 nomFonction='NomFonction' ; donnee='Donnee' ; fonction='Fonction' ;

 Fonction=**CreerElement**(fonction) ;

 NomFontion=**CreerElement**(nomFonction) ; **AjouterTextElement**(NomFonction, Cle_f) ;

AjouterUnElement(Fonction, NomFonction) ;

 /* Création des éléments Donnee depuis la liste Donnees*/

 Pour i=0 à i < Donnees.longueur faire :

 Element Donnee=**CreerElement**(donnee) ;

AjouterTextElement(Donnee, Donnees[i]) ;

AjouterUnElement(Fonction, Donnee) ;

 finpour.

 Retourner Fonction ;

Fin.

}

Procédure **AjouterUnElement**(element : Element ; sousElement : Element){

Début

 Element.souselement=sousElement ;

}

Procédure **AjouterTextElement**(element : Element, sousElement : Chaîne de caractère){

Début

 element.text=sousElement ;

}

Procédure **EnregistrerXML**(document : Document){

Début

 Créer un nouveau fichier dans le serveur avec l'extension .xml ;

 Lire le contenu du document et l'écrire dans le fichier créé ;

Fin.}

On utilise une HashMap pour conserver les informations récupérées depuis la requête afin d'obtenir un accès par Clé/Valeur, la clé représente l'identifiant du médecin que l'on autorise à effectuer les opérations représentées dans la valeur qui correspond à cette clé. Cette valeur quant à elle est une autre HashMap Clé/Valeur dont la clé représente le nom de l'opération et sa valeur correspondante contient une liste de chaîne de caractère représentant les données à être manipulées par cette opération.

L'identifiant du patient est récupéré par le middleware depuis les informations de l'utilisateur en session.

Les informations dans la MapPermission qui est utilisée par l'algorithme précédent sont récupérées depuis la requête et y sont stockées en suivant certaines étapes démontrées dans le pseudo-algorithme ci-dessous :

Variables

Nom_Fonctions, Donnees, Utilisateurs : Liste <Chaine de caractère> ;
MapFonction : HashMap<String, Donnees> ;
MapPermission : HashMap<String, MapFonction> ;

Début

Nom_Fonctions=**RecupererNomsFonctionnalitees()** ;

Pour i ← 0 à i < Longueur de (Nom_Fonctions) faire :

Si (**ExisteDansRequete**(Nom_Fonctions[i])) faire :

Donnees=**RecupererDonneeFonctionnalitees**(Nom_Fonction[i]) ;

Pour j ← 0 à j < Longueur de (Donnees) faire :

Utilisateurs=**RecupererUtilisateursDonnees**(Donnees[j]) ;

Pour k ← 0 à k < Longueur de (Utilisateurs) faire :

Si (MapPermission.contientCle(Utilisateurs[k])) Alors:

MapFonction=MapPermission.Cle(Utilisateurs[k]) ;

Si (MapFonction.contientCle(Nom_Fonctions[i]) Alors :

MapPermission.Cle(Utilisateurs[k]).Cle(Nom_Fonctions[i])

.Ajouter(Donnees[j]) ;

Sinon

MapPermission.Cle(Utilisateur[k]).**Ajouter**(Nom_Fonctions[i]) ;

MapPermission.Cle(Utilisateur[k]).Cle(Nom_Fonctions[i])

.Ajouter(Donnees[j]) ;

Sinon

MF←**InstancierVariableMapFonction()** ;

MF.**AjouterCle**(Nom_Fonction[i]) ;

MF.Cle(Nom_Fonctions[i]).**Ajouter**(Donnees[j]) ;

MapPermission.**AjouterCleEtValeur**(Utilisateurs[k], MF) ;

Finsi ;

Finpour ;

finPour ;

Finsi ;

finPour ;

Retourner MapPermission ;

Fin.

Les noms des fonctionnalités sont récupérés depuis le système de gestion de base de données ainsi que les données et les utilisateurs pour être affichées aux patients dans un formulaire démontrant chaque fonctionnalité les données sur les quelles elle opère, ainsi que l'utilisateur pouvant effectuer cet opération sur cette donnée.

En sélectionnant ces fonctionnalités avec les données correspondante et les utilisateurs et enregistrer, il envoie ces informations en tant que des paramètres dans une requête, le middleware saura décortiquer cette requête pour en extraire les informations et les reproduire dans un modèle spécifié (HashMap) pour simplifier le traitement de constitution du fichier XML du patient.

Le tableau ci-dessous représente une vue globale de la HashMap :

Clé HM1	Valeur HM1	
	Clé HM2	Valeur HM2
Médecin 1	Opération 1	{Donnée 1, Donnée 2, Donnée 3}
	Opération 2	{Donnée 1, Donnée 2}
Médecin 2	Opération 1	{Donnée 1, Donnée 3}
	Opération 3	/

Tableau3.HashMap

Faisons la correspondance entre ces informations et les éléments du fichier XML expliqués dans la partie 5.2.1.2.

- Donnée 1, Donnée 2, Donnée 3, sont représentées chacune dans un élément <Donnee>.
- Opération 1, Opération 2, Opération 3 représentent le nom de la fonctionnalité équivalente dans le XML avec l'élément <NomFonction>.
- L'élément <Fonction> englobe les deux cités en-dessus, ce qui équivaut à une Clé/Valeur de la sous HashMap HM2.
- Médecin 1 et Médecin 2 représente la valeur de l'attribut utilisateur_id contenu dans l'élément <Permission utilisateur_id=""> englobant ainsi une Clé/Valeur d'une HashMap HM1.
- L'élément <Patient> dont de son attribut patient_id a été récupéré depuis la session, englobe la totalité des paires Clé/Valeur de la HashMap HM1.

✓ **Exemple :**

Supposons qu'un certain patient avec l'identifiant 9 a donné les autorisations aux médecins suivants d'effectuer les opérations suivantes sur les données suivantes (ces informations sont rangées dans une HashMap représentée par le tableau ci-dessous):

Identifiantmédecin	Operations et données	
	NomFonction	Donnee
78	Consulter info patient	/
	Ajouter maladie	/
	Consulter maladies chroniques	{diabetes, bronchite, epilepsies}
	Consulter les vaccins	{BCG Pasteur, COMVAX}
6	Consulter info patient	/
	Consulter les analyses	{bilan lipidique, TSH}
	Ajouter maladie	/

Tableau3.Informations rangées dans HashMap

Le fichier XML construit depuis ces informations suivant le pseudo-algorithme précédent est ceci :

```

<?xml version="1.0" encoding="UTF-8"?>
<Patient xmlns:xs="http://www.w3.org/2001/XMLSchema-instance"
  xs:noNamespaceSchemaLocation="Localisation du fichier XSD"
  patient_id="9">
  <Permission utilisateur_id="78">
    <Fonction>
      <NomFonction>Consulter info patient</NomFonction>
    </Fonction>
    <Fonction>
      <NomFonction>Ajouter maladie</NomFonction>
    </Fonction>
    <Fonction>
      <NomFonction>Consulter maladies chroniques</NomFonction>
      <Donnee>diabete</Donnee>
      <Donnee>bronchite</Donnee>
      <Donnee>epilepsie</Donnee>
    </Fonction>
    <Fonction>
      <NomFonction>Consulter les vaccins</NomFonction>
      <Donnee>BCG Pasteur</Donnee>
      <Donnee>COMVAX</Donnee>
    </Fonction>
  </Permission>
  <Permission utilisateur_id="6">
    <Fonction>
      <NomFonction>Consulter info patient</NomFonction>
    </Fonction>
    <Fonction>
      <NomFonction>Consulter les analyses</NomFonction>
      <Donnee>bilan lipidique</Donnee>
      <Donnee>TSH</Donnee>
    </Fonction>
    <Fonction>
      <NomFonction>Ajouter maladie</NomFonction>
    </Fonction>
  </Permission>
</Patient>

```

Observation : Le médecin avec l'identifiant 78 a la permission d'effectuer la fonctionnalité "Consulter maladies chroniques" sur les données "diabete", "bronchite" et "epilepsie" et la fonctionnalité "Consulter les vaccins" sur les données "BCG Pasteur" et "COMVAX" du patient avec l'identifiant 9.

Le médecin avec l'identifiant 6 a la permission d'effectuer la fonctionnalité "Consulter les analyses " sur les données "bilan lipidique" et "TSH".

N.B : Les fonctionnalités "Consulter info patient" et "Ajouter maladie" sont permises pour les deux médecins parce que c'est des fonctionnalités qui seront ajoutées au fichier XML du patient par défaut. Effectivement tous les médecins que le patient consulte ont le droit d'effectuer certaines opérations qui leurs sont nécessaires en dépendants de leur spécialités.

5.2.2 Enregistrement des XML

Le fichier XML est une ressource critique par ce qu'elle détient des informations que seul son propriétaire a le droit de connaître. Ce document doit être donc mis à la disposition du middleware seulement, il doit donc savoir sa localisation s'il existe, comment le récupérer et où le placer après sa création.

5.2.2.1 Emplacement

Il existe autant de document XML que le nombre de patient conservés dans le même lieu. Pour faciliter leur accès au middleware, on a fait en sorte de stocker ces fichiers dans le serveur d'application, ainsi que le fichier XSD qui doit être aussi dans le même endroit.

Pour la récupération de l'un d'entre eux le middleware doit faire la distinction, quel fichier appartient à un tel patient, pour cela le nom de chaque fichier est enregistré par le mot "Patient" avec l'identifiant attribué au propriétaire de ce fichier séparés par un tiret "_".

Exemple : Patient_9.xml.

5.3 Processus effectué par le middleware

Le travail du middleware passe par plusieurs étapes avant d'arriver à la prise de décision, celle de continuer à exécuter l'opération effectuée par l'utilisateur ou non. Dans cette partie on va expliquer ces étapes en détails jusqu'à arrivé à la plus importante qui est la vérification de droit d'accès.

5.3.1 Interception des requêtes http

L'utilisateur envoie une requête http depuis l'interface mis à sa disposition au serveur d'application pour pouvoir effectuer une opération, au lieu d'aller directement à sa destination, elle sera interceptée par une technologie intermédiaire qui est notre **Middleware**. De ce fait, l'utilisateur ignore totalement que la requête n'a pas suivi son parcours normal.

5.3.2 Traitement des requêtes http

La méthode des requêtes réceptionnées est de type Get, les informations nécessaires au traitement sont donc été envoyées par une url en tant que paramètre.

5.3.2.1 Récupération des paramètres

Depuis l'url, le middleware récupère les valeurs des paramètres concernés par le traitement. On a donné un nom pour chacun de ces paramètres qui correspond à son élément présent dans le fichier XML. Les paramètres essentiels pour le middleware sont les suivants :

- **NomFonction** : sa valeur correspond au nom de la fonctionnalité que l'utilisateur voudrait effectuer.
- **Patient_id** : sa valeur correspond à l'identifiant du patient concerné.
- **Donnee** : sa valeur correspond à la donnée du patient qui va être manipulée.

Le paramètre le plus important est récupéré depuis les informations concernant la session courante, après l'authentification, la valeur du paramètre **utilisateur_id** qui correspond à l'attribut de l'élément <Permission> est celui de l'utilisateur qui va effectuer l'opération **NomFonction**.

5.3.2.2 Vérification d'autorisation

Après avoir récupéré les paramètres cités dans 5.3.2.1, le middleware applique une suite finie de règles dans un ordre déterminé à ces valeurs pour arriver à un certain résultat impliquant une prise de décision qui est permettre ou non à l'utilisateur d'effectuer l'opération souhaitée. Le logigramme ci-dessous est un schéma qui représente les étapes successives de ce processus :

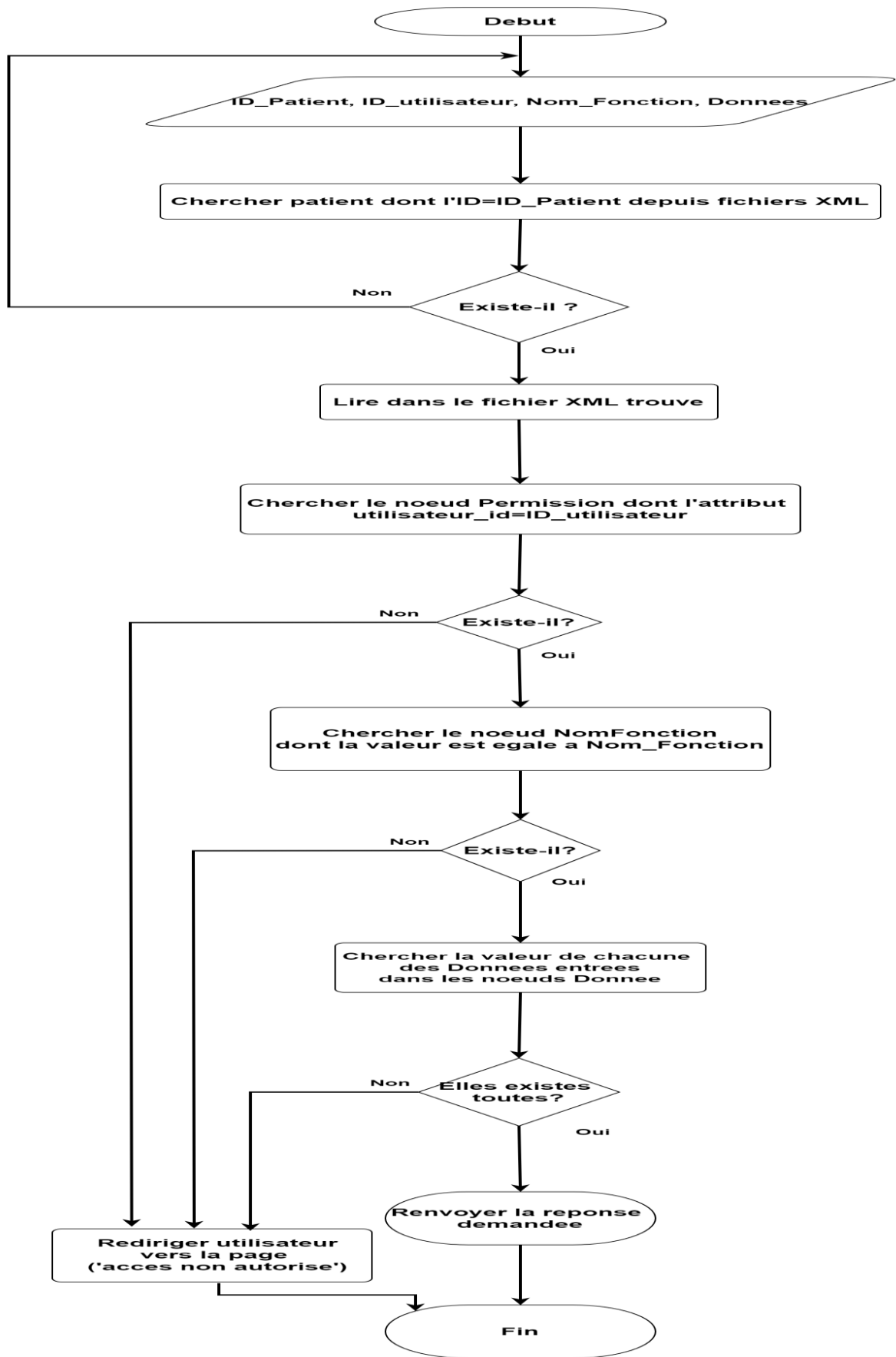


Figure 11. Logigramme

5.3.3 Renvoi des réponses http

A la fin du processus de vérification d'autorisation, le middleware peut renvoyer deux types de réponses http comprenant une ligne de statut précisant l'état du traitement de la requête à l'aide d'un code et un texte explicatif.

Comme mentionné dans la partie 5.1.1, il peut laisser passer la demande pour effectuer l'opération souhaitée, la réponse correspondante comprend un élément représentant le code de statut 200 et un message Ok, désignant le bon déroulement de la requête, ainsi que le corps de la réponse qui contient les informations demandée par l'utilisateur. Il peut ne pas laisser passer la demande et renvoyer une réponse http comprenant un élément représentant le code de statut 403 et un message FORBIDDEN qui veut dire que l'accès à la ressource demandée est tout simplement interdit.

Par exemple, si l'utilisateur avec l'identifiant 6 demande d'effectuer la fonctionnalité Consulter les vaccins en précisant la valeur BCG Pasteur du patient avec l'identifiant 9 de l'exemple de la partie 5.2.1.3 il sera redirigé vers une page précisant que l'action demandée ne lui est pas permise (statut 403 FORBIDDEN). Par contre, si le même utilisateur veut effectuer la fonctionnalité Consulter les analyses en précisant l'une ou les deux données bilan lipidique et TSH, les documents contenant ces informations lui seront renvoyés (statut 200 Ok).

6 Etude conceptuelle de l'exemple d'application

Pour mettre en place notre solution, on va développer une application qui doit satisfaire les besoins fonctionnels principaux suivants :

- Stockage des données dans un SGBD.
- Récupération des données depuis un SGBD.
- Suppression et modification des données présentes dans un SGBD.
- Exécution des opérations sur les dossiers médicaux présents dans un SGBD.
- Spécification des autorisations sur les données personnelles.
- Connexion et l'authentification à l'application par rôle.

6.1 Diagramme de cas d'utilisation

Le diagramme de cas d'utilisation représente le comportement fonctionnel d'un système. Dans le cas de notre application, on distingue quatre acteurs, héritant les mêmes fonctionnalités d'un simple utilisateur.

6.1.1 Diagramme de cas d'utilisation de l'utilisateur

Chaque utilisateur enregistré dans l'application peut établir un nombre de fonctionnalité comme illustré dans la figure ci-dessous :

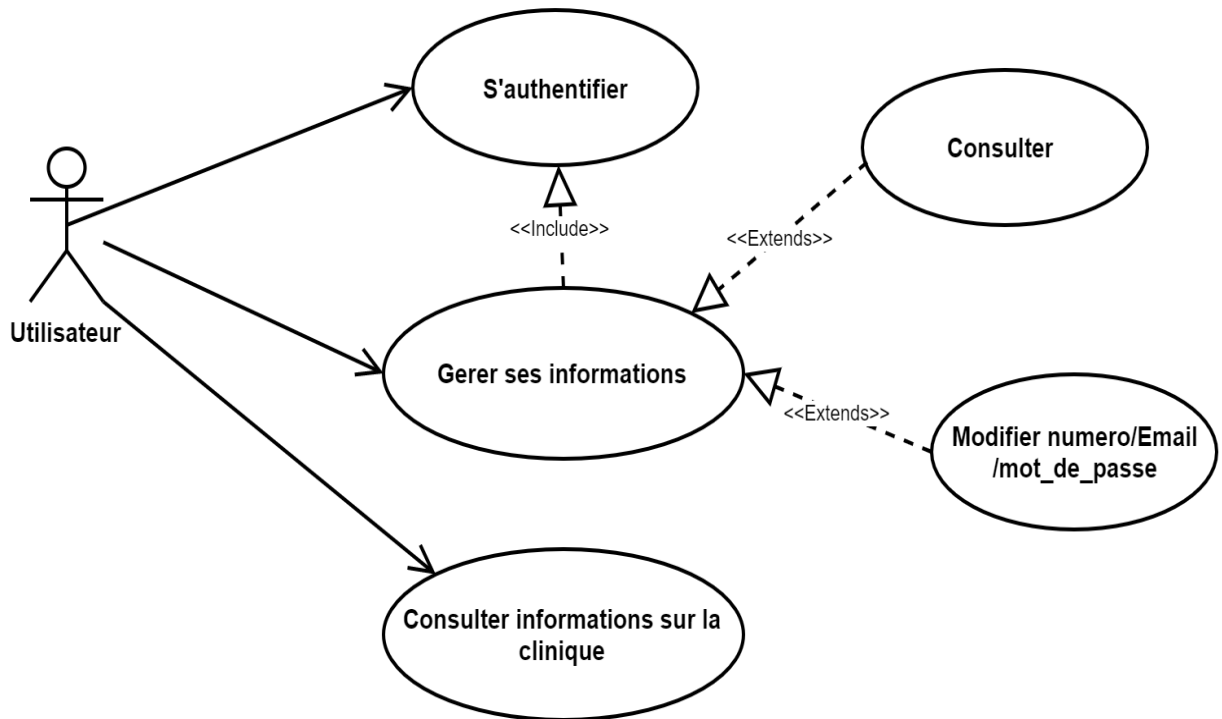


Figure 12. Diagramme de cas d'utilisateur

6.1.2 Diagramme de cas de l'administrateur

Un utilisateur avec un rôle administrateur peut effectuer les fonctionnalités illustrées dans le diagramme suivant :

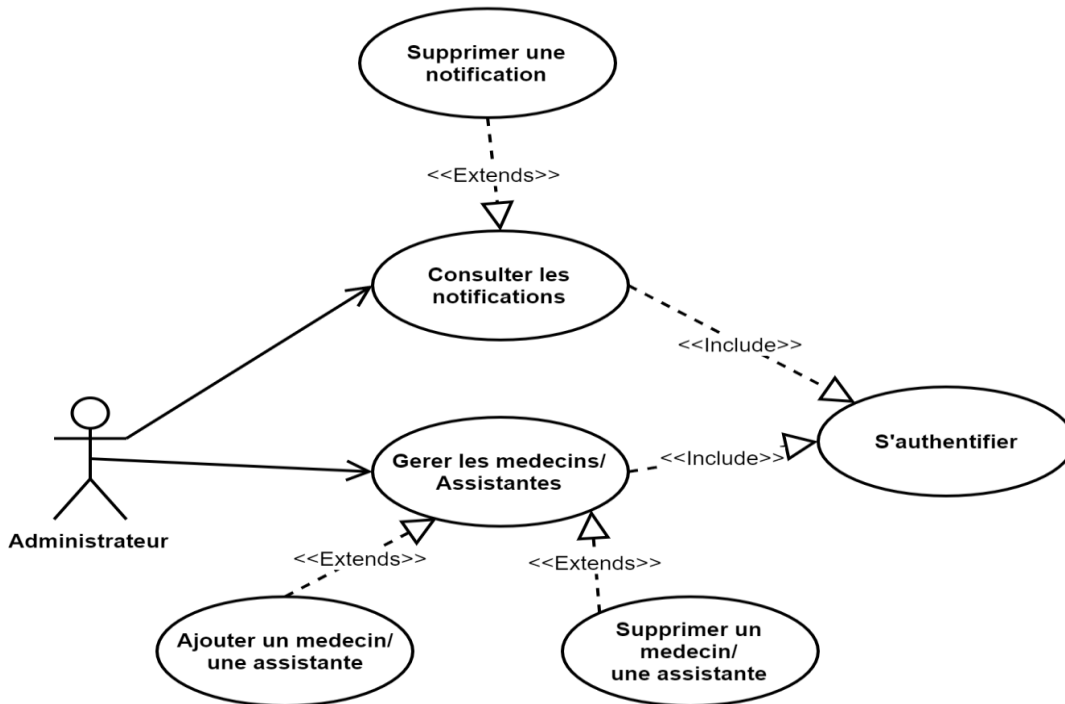


Figure 13. Diagramme de cas de l'administrateur

6.1.3 Diagramme de cas du médecin

Un utilisateur avec un rôle médecin peut effectuer les opérations illustrées en détails dans les figures qui suivent :

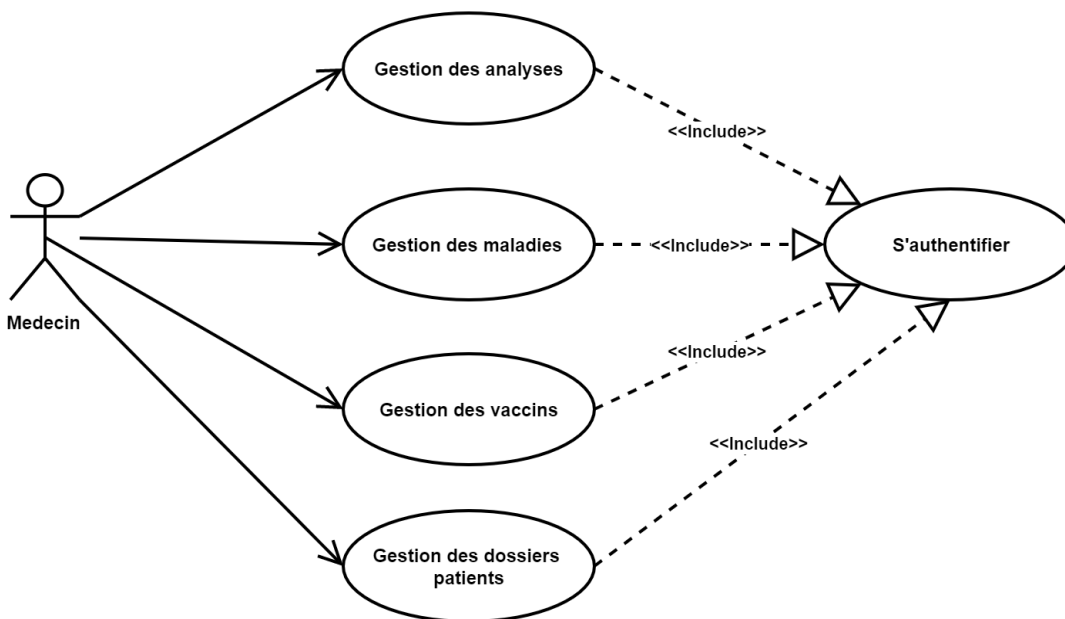


Figure 14. Diagramme de cas du médecin

6.1.4 Diagramme de cas de l'assistante

Une assistante possède un ensemble d'opération qu'elle puisse effectuer, montrées dans le diagramme de cas d'utilisation suivant :

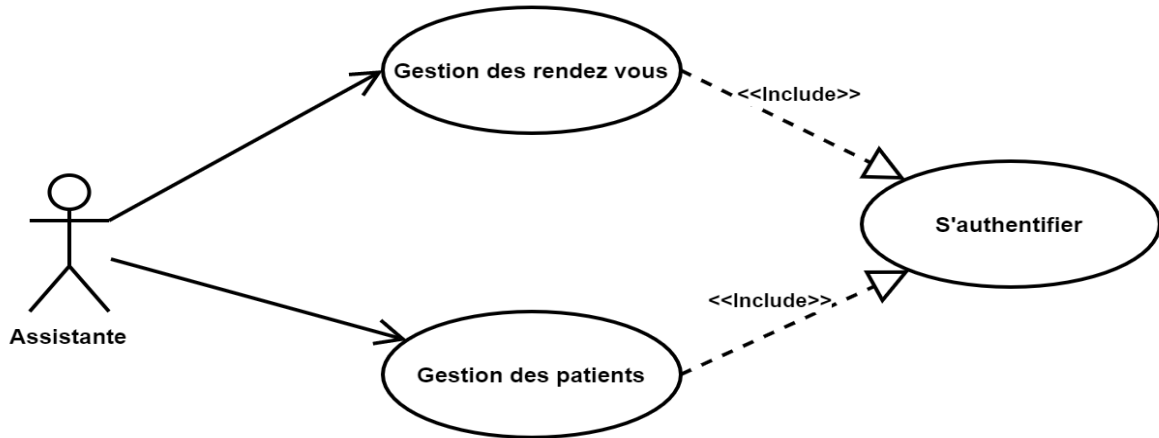


Figure 15. Diagramme de cas d'utilisation de l'assistante.

6.1.5 Diagramme de cas du patient

Pour les patients, chacun possède de multiples opérations illustrées dans le diagramme de cas d'utilisation ci-dessous :

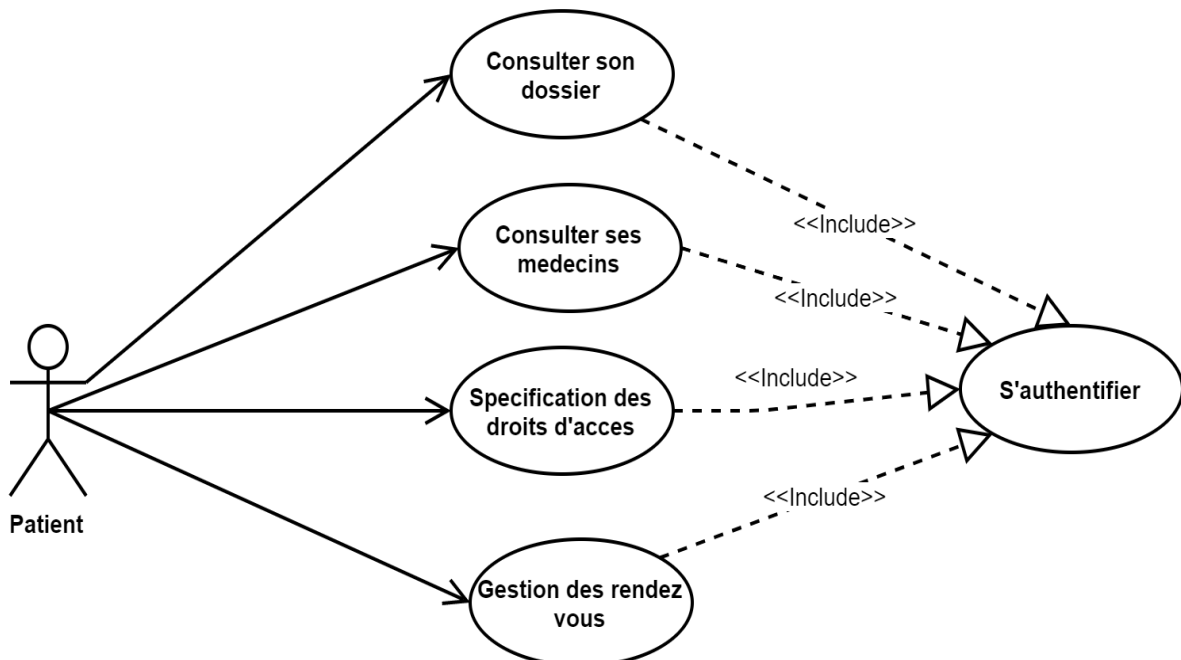


Figure 16. Diagramme de cas d'utilisation du patient

6.2 Diagramme de séquence

Dans cette partie on va décrire le fonctionnement de notre application ainsi que le middleware à l'aide du diagramme de séquence qui expose en détail la façon dont les opérations principales sont effectuées.

6.2.1 Diagramme de séquence pour l'authentification

Pour pouvoir effectuer les opérations permises, les utilisateurs doivent s'authentifier en entrant un email et un mot de passe, s'ils sont correctes, ils seront redirigés vers une interface contenant des fonctionnalités dépendamment de leur rôles.

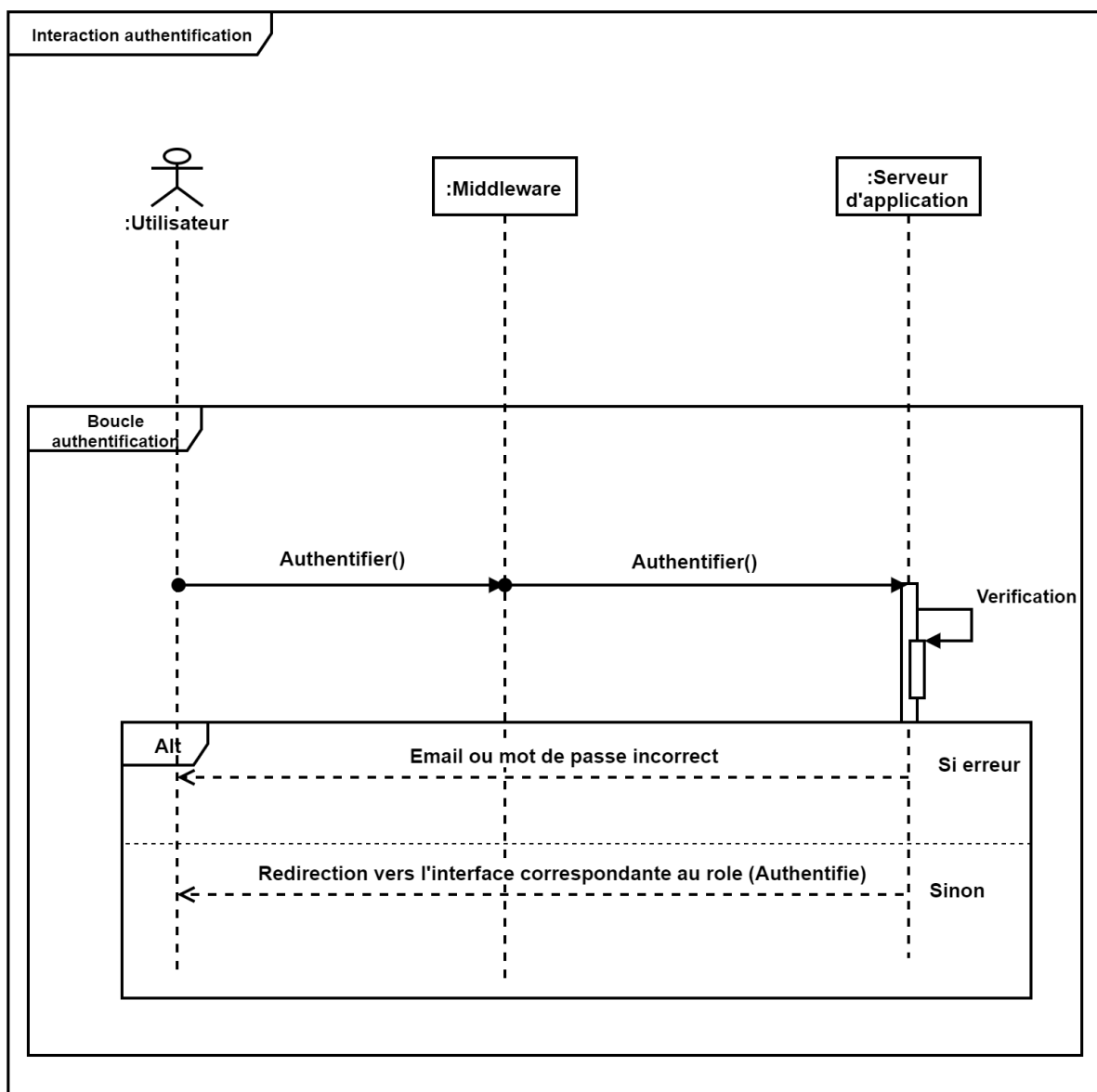


Figure 17. Diagramme de séquence authentification

6.2.2 Diagramme de séquence pour la spécification

Un formulaire contenant toutes les opérations qui puissent être effectuées sur le dossier d'un patient est exposé avec la possibilité de choisir parmi ces fonctionnalités celle qui est permise pour un certain médecin sur une donnée spécifique. Cette spécification suit les étapes détaillées ci-dessous :

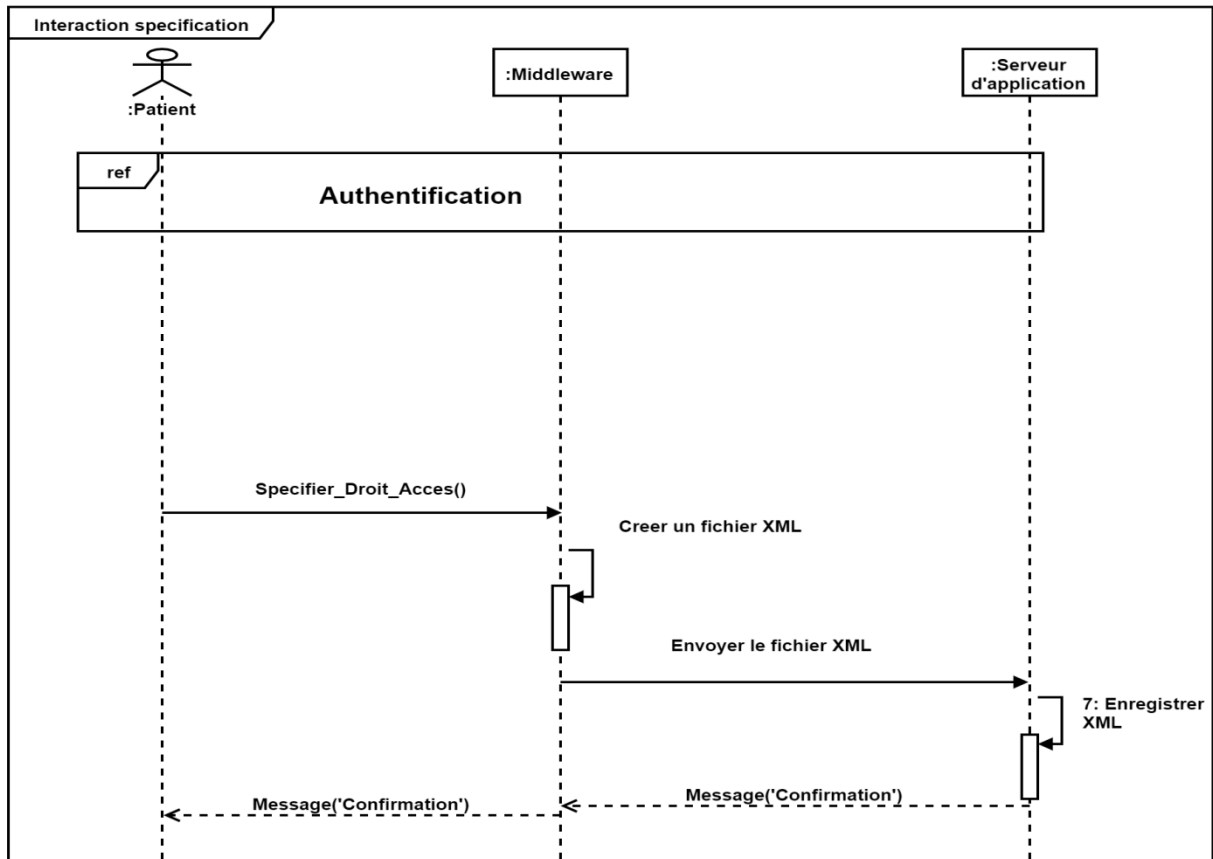


Figure 18. Diagramme de séquence de spécification

6.2.3 Diagramme de séquence pour vérification d'autorisation

Pour pouvoir effectuer une opération sur un dossier patient, il faut que celle-ci soit présente dans le fichier XML de ce dernier ainsi que les données et l'utilisateur demandant à l'effectuer. En d'autre terme, l'opération doit être déjà spécifiée par le propriétaire des données pour cet utilisateur suivant les étapes de la figure 17. Cette vérification suit les étapes démontrées dans le diagramme suivant :

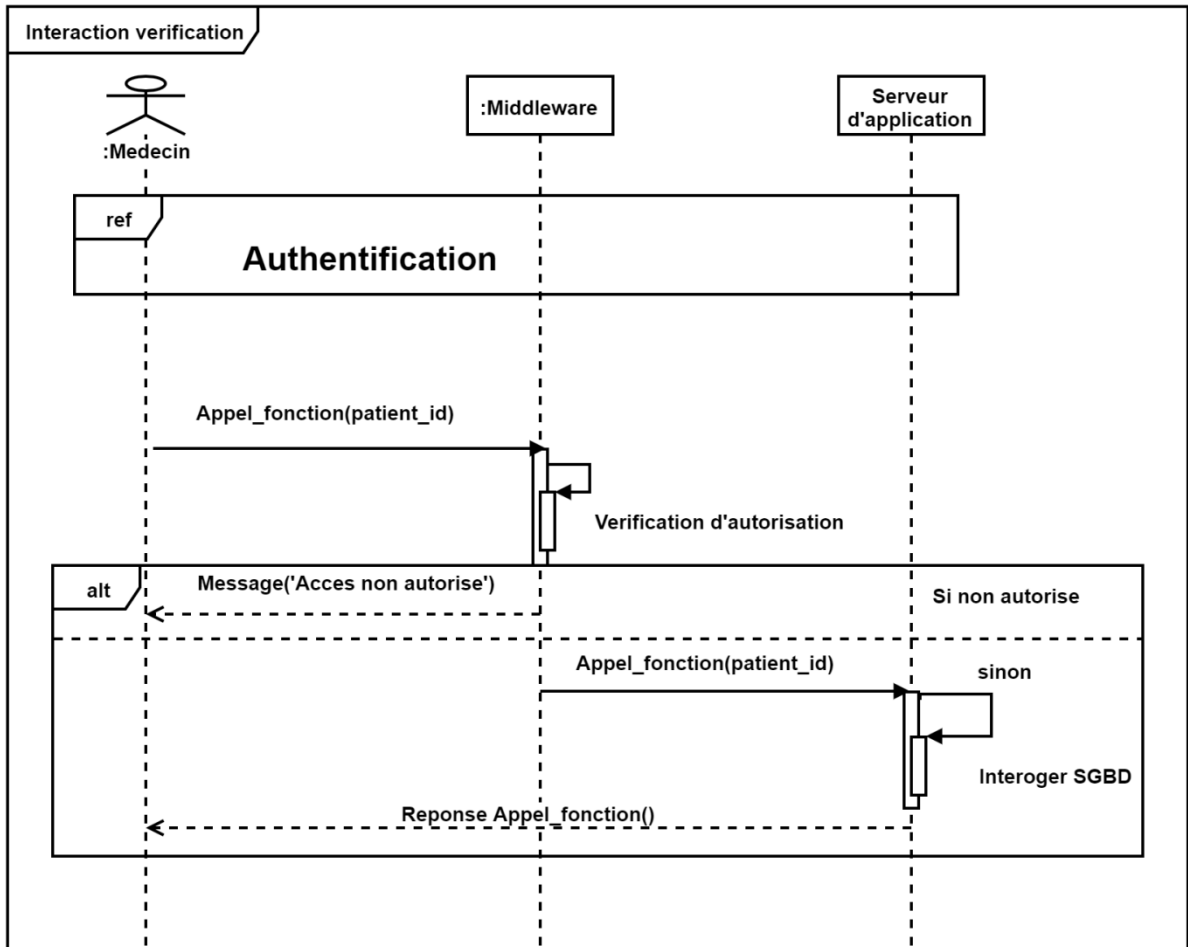


Figure 19.Diagramme de séquence vérification d'autorisation

6.3. Diagramme de classe

Le diagramme suivant décrit la structure de notre application et représente les entités correspondantes à chacun des acteurs concernés par notre model de contrôle d'accès RBAC+ :

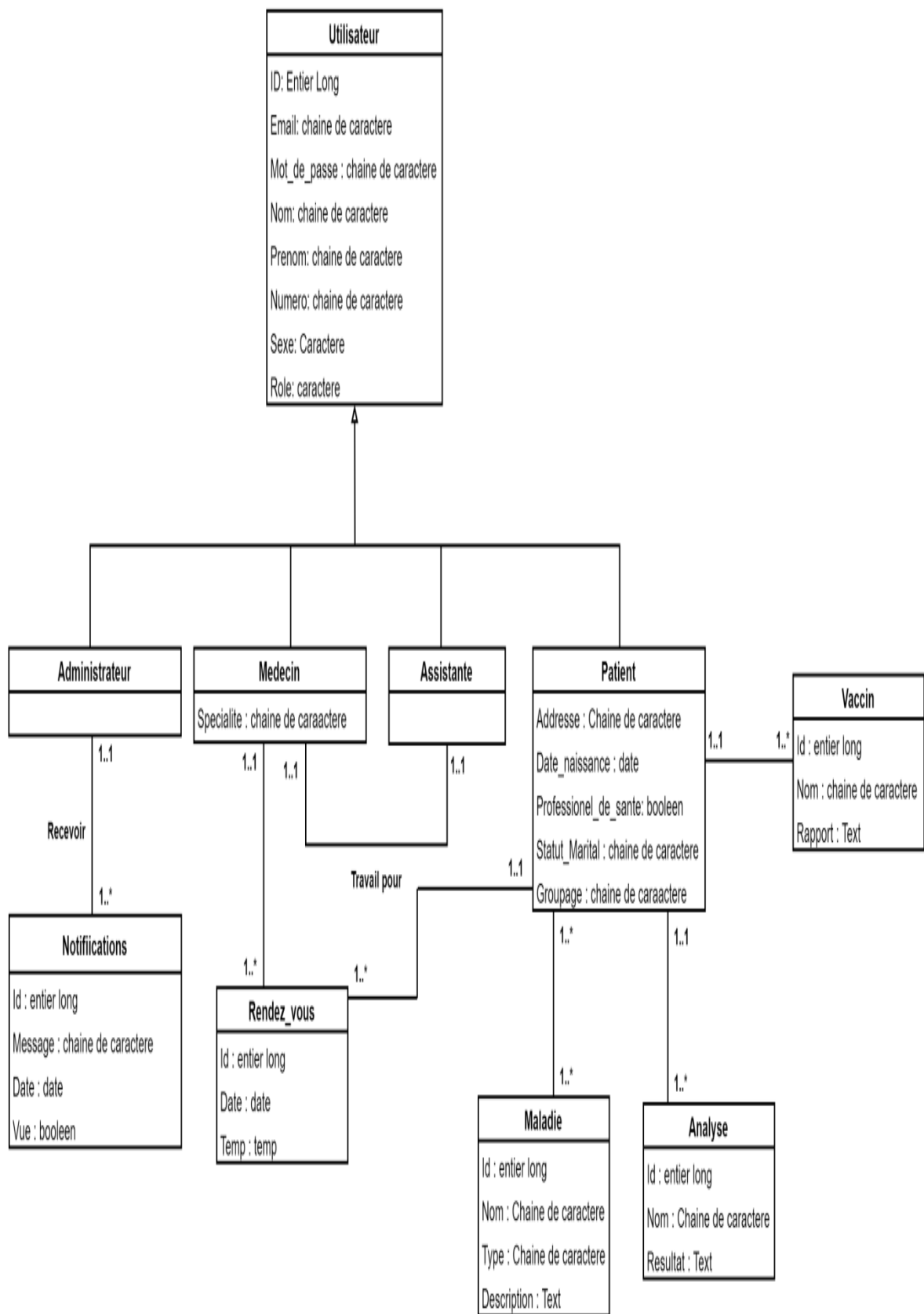


Figure 20.Diagramme de classe

7 Conclusion

Dans ce chapitre, on a décrit notre approche pour une solution médiatrice pour les applications distribuées. Cette approche est basée sur le modèle de control d'accès RBAC avec plus de spécificités, offrant ainsi un niveau de sécurité élevé et une garanti pour la confidentialité des données personnelles. Pour mettre en place cette solution, un exemple d'application a été conçu avec toutes les fonctionnalités nécessaires permettant son bon fonctionnement.



Chapitre 4 : Implémentation et Réalisation

1 Introduction

En se basant sur l'analyse et la conception qu'on a obtenue dans les chapitres précédents, on va présenter notre application qui consiste en la réalisation d'un contrôleur d'accès basé sur les rôles et faire une simulation en suivant les réactions de notre middleware à différentes contraintes, ce qui est l'objectif de ces expérimentations.

2 Outils et environnement de développement

Un environnement de développement se réfère à une suite d'outils et d'applications qu'on a installé sur la machine pour effectuer des essais et des tests pour arriver à développer notre solution. Ces outils sont les suivants :

2.1 Spring Boot Framework

Java Spring Framework (Spring Framework) est un cadre d'entreprise populaire et open source pour la création d'applications autonomes de niveau production qui s'exécutent sur la machine virtuelle Java (JVM).

Java Spring Boot (Spring Boot) est un outil qui accélère et facilite le développement d'applications Web et de micro services avec Spring Framework grâce à trois fonctionnalités principales : configuration automatique, une approche avisée de la configuration, la possibilité de créer des applications autonomes. [41]



2.2 Apache maven

Apache Maven est un outil de gestion et de compréhension de projet logiciel. Basé sur le concept d'un modèle d'objet de projet (POM), Maven peut gérer la construction, les rapports et la documentation d'un projet à partir d'une information centrale. [42]

2.3 MySQL

MySQL est un système de gestion de base de données relationnelle (SGBDR) open source, basé sur le langage SQL (StructuredQueryLanguage).



Cette solution, parmi les plus populaires au monde, est connue pour délivrer de hautes performances dans le stockage de larges volumes de données (notamment dans le big data). [43]

2.4 Java développement kit

Le Java Development Kit (JDK) désigne un ensemble de bibliothèques logicielles de base du langage de programmation Java, ainsi que les outils avec lesquels le code Java peut être compilé, transformé en bytecode destiné à la machine virtuelle Java.[44]

2.5 XSD et XML

Diverses organisations utilisent différents systèmes. XML est une méthode qui peut être utilisée pour transférer des données entre différents programmes et plates-formes. XSD est lié à XML. Cet article a présenté la différence entre XML et XSD. La différence entre XML et XSD réside dans le fait que XML est un langage de balisage qui constitue une méthode flexible de création et de partage de données sur des systèmes incompatibles, alors que XSD est utilisé pour définir la structure et le contenu d'un document XML.[44]

2.6 JSON

JSON est un format de texte complètement indépendant du langage mais utilise des conventions familières aux programmeurs de la famille C de langages, y compris C, C++, C#, Java, JavaScript, Perl, Python et bien d'autres. Ces propriétés font de JSON un langage d'échange de données idéal.[45]

2.7 JDBC

La connectivité de base de données Java™ (JDBC) est la spécification JavaSoft d'une interface de programmation d'application (API) standard qui permet aux programmes Java d'accéder aux systèmes de gestion de base de données. L'API JDBC consiste en un ensemble d'interfaces et de classes écrites dans le langage de programmation Java.

À l'aide de ces interfaces et classes standard, les programmeurs peuvent écrire des applications qui se connectent à des bases de données, envoyer des requêtes écrites en langage de requête structuré (SQL) et traiter les résultats.

Étant donné que JDBC est une spécification standard, un programme Java qui utilise l'API JDBC peut se connecter à n'importe quel système de gestion de base de données (SGBD), tant qu'un pilote existe pour ce SGBD particulier. [50]

2.8 MVC

Model View Controller (MVC) est un modèle de conception pour les logiciels informatiques. Il peut être considéré comme une approche pour distinguer le modèle de données, le contrôle du traitement et l'interface utilisateur. Il sépare parfaitement l'interface graphique affichée pour l'utilisateur du code qui gère les actions de l'utilisateur. L'objectif est de fournir un cadre qui applique une conception meilleure et plus précise.[46]

2.9 Html, CSS et Javascript

HTML est le langage de balisage que nous utilisons pour structurer et donner un sens à notre contenu Web, par exemple en définissant des paragraphes, des titres et des tableaux de données, ou en incorporant des images et des vidéos dans la page.



CSS est un langage de règles de style que nous utilisons pour appliquer un style à notre contenu HTML, par exemple en définissant des couleurs et des polices d'arrière-plan et en disposant notre contenu dans plusieurs colonnes.



JavaScript est un langage de script qui vous permet de créer du contenu mis à jour dynamiquement, de contrôler le multimédia, d'animer des images et à peu près tout le reste. [47]

2.10 Thymeleaf

Thymeleaf est un moteur de modèle Java côté serveur moderne pour les environnements Web et autonomes.



L'objectif principal de Thymeleaf est d'apporter des modèles naturels élégants à votre flux de travail de développement - HTML qui peut être correctement affiché dans les navigateurs et également fonctionner comme des prototypes statiques, permettant une collaboration plus étroite au sein des équipes de développement.[48]

2.11 Bootstrap

Bootstrap est un Framework Web open-source gratuit et ouvert pour



La conception de sites Web et d'applications Web. Il contient des modèles de conception basés sur HTML et CSS pour la typographie, les formulaires, les boutons, la navigation et d'autres composants d'interface, ainsi que des extensions JavaScript optionnelles. Contrairement à de nombreux Framework Web, il se préoccupe uniquement du développement frontal. [49]

3 Présentation de l'application

Dans cette partie, on va présenter notre application en commençant par niveau hiérarchique le plus bas.

Notre Application commence d'abord par l'authentification des utilisateurs.

L'authentification nous dirige vers quatre espaces à l'aide d'un middleware qui vérifie le rôle de l'authentifiant

Nous pouvons dire que l'application est composée de quatre parties :

- L'espace d'administrateur
- L'espace des médecins
- L'espace des assistantes
- L'espace des patients

Les **middlewares** permettent de filtrer les requêtes HTTP. Par exemple, ils permettent de vérifier si l'utilisateur est authentifié ou non. Ils peuvent être utilisés à chaque chargement de page ou seulement à certains moments précis.

3.1 Rôle patient

3.1.1 Informations clinique

Les interfaces suivantes représentent la partie statique contenant les informations sur le personnel soignant, département et renseignements de la clinique accessible sans besoin d'authentification.



Figure 21. Accueil

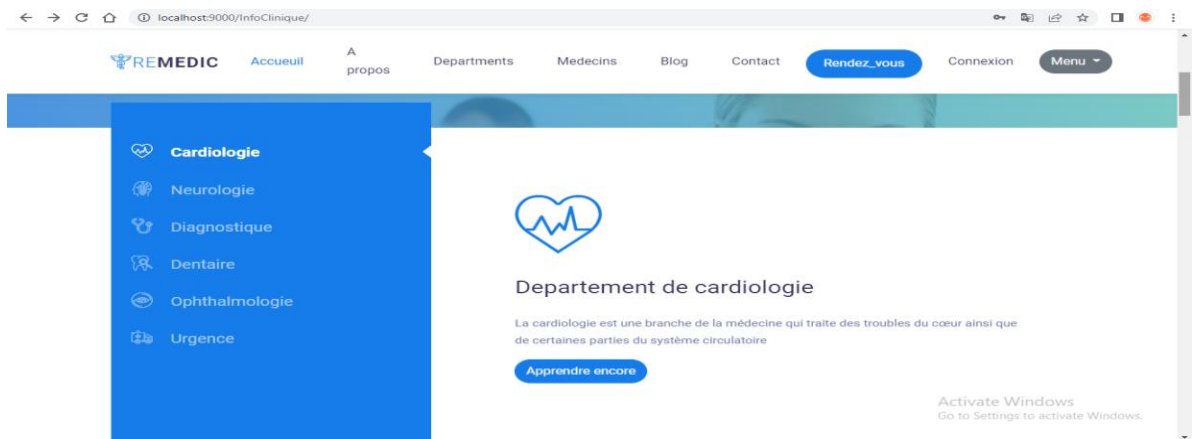


Figure 22. Départements

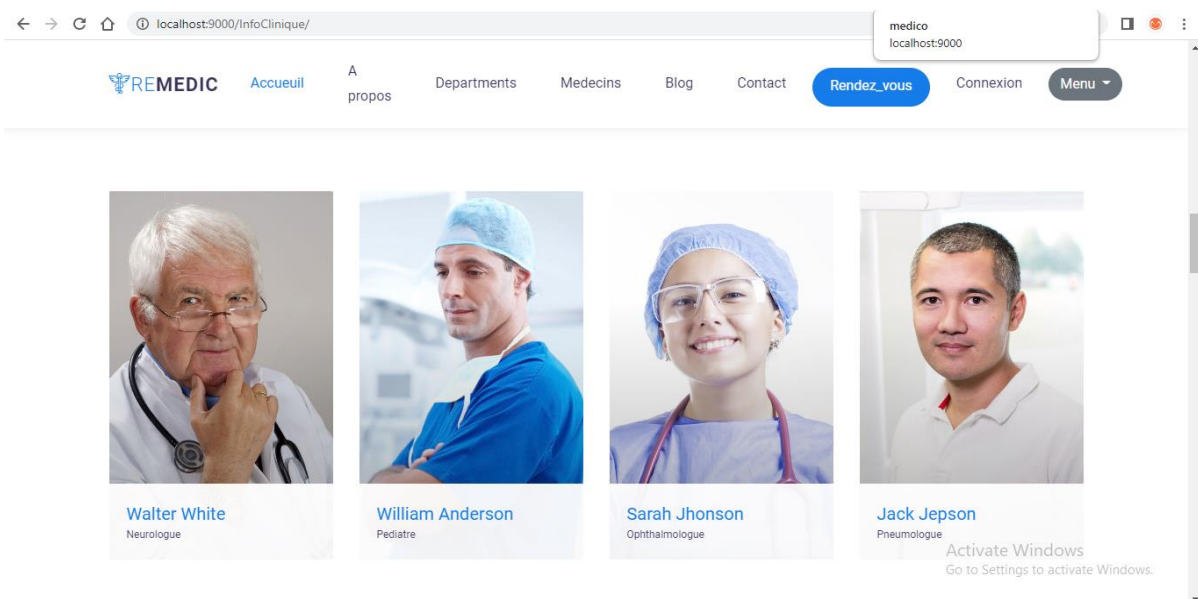


Figure 23. Les médecins

3.1.2 Spécification d'autorisation

Pour pouvoir spécifier les autorisations pour les médecins qu'il consulte, le patient après s'être authentifié, doit passer par le formulaire suivant :

Figure 24. Formulaire de spécification de droits d'accès

Voici le code exécuté après la sauvegarde de ce formulaire :

```
129 .....
130 .....
131 @PostMapping("/EnregistrerXML")
132 //ResponseBody
133 public ModelAndView haja(Principal principal,@ModelAttribute ModeleSpecification specification,@RequestParam MultiValueMap<String, String> requestParams) {
134
135
136 List<String> fct=maladieService.getTypesMaladie();
137 fct.replaceAll(s-> "Consulter maladies "+s);
138 fct.add("Consulter les vaccins");fct.add("Consulter les analyses");
139
140 for(String fonction:fct) {
141 System.err.println(fonction);
142 if(requestParams.get(fonction)!=null){//verifier s'il ya une maladie selectionnee dans la section
143
144 List<String> list = requestParams.get(fonction); //récupérer toutes les maladies sélectionnées dans une liste
145
146 for(String maladie:list) { //parcourir toutes les maladies sélectionnées
147
148 List<String> list1 =requestParams.get("d-"+maladie); //récupérer les ids des médecins autorisec de consulter cette maladie
149
150 System.out.println("les médecin autoriser pour voir la maladie "+maladie+" sont:"); //
151
152 for(String id:list1) { //parcourir tous les ids médecins
153
154 if(specification.getPermission().containsKey(id)) {
155 if(specification.getPermission().get(id).containsKey(fonction)) {
156 specification.getPermission().get(id).get(fonction).add(maladie);
157 }
158 }
159
160 specification.getPermission().get(id).put(fonction, new ArrayList<>());
161 specification.getPermission().get(id).get(fonction).add(maladie);}
162 }
163 }
164 ModeleSpecification sp = new ModeleSpecification();
165 sp.getFonction().put(fonction, new ArrayList<String>());
166 sp.getFonction().get(fonction).add(maladie);
167 specification.getPermission().put(id, sp.getFonction() );
168
169
170 specification.getPermission().put(id, sp.getFonction() );
171 //specification.getPermission().get(id).get("appareil_degistif").add(maladie);
172
173 }
174
175 //récupere les médecin a partir d'une BD par id
176 // ajouter les médecins aux fichier xml
177 System.out.println("medecin "+id+" + maladie);
178
179
180 }
181
182 Patient pt=patientService.getPatient(principal.getName());
183 List<Medecin> mds=pt.getMedecins();
184 for(Medecin m: mds) {
185 String id=String.valueOf(m.getId());
186 if (!specification.getPermission().containsKey(id)) {
187 specification.getPermission().put(id, new HashMap<String, List<String>>());
188 }
189 }
190
191 try {
192 Utils.Creer_XML_Patient(specification, String.valueOf(pt.getId()));
193 } catch (ParserConfigurationException e) {
194
195 e.printStackTrace();
196 } catch (SAXException e) {
197
198 e.printStackTrace();
199 } catch (IOException e) {
200
201 e.printStackTrace();
202 }
203
204 return new ModelAndView("redirect:/Patient/rdv");
205 }
```

Figure 25. Récupération des droits

```

60
61
62 public Boolean Creer_XML_Patient(ModelesSpecification specification, String patient_id) throws ParserConfigurationException, SAXException, IOException {
63
64     String P1="P1", P2="P2", P3="P3", P4="P4";
65     String F1="F1", F2="F2", F3="F3", F4="F4";
66     String D1="D1", D2="D2", D3="D3", D4="D4";
67     String utilisateur_id="utilisateur_id"; //, patient_id="9";
68
69
70
71     DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
72     DocumentBuilder builder=factory.newDocumentBuilder();
73
74     Document document=builder.newDocument();
75
76     HashMap<String,List<String>> MapFonction=new HashMap<String,List<String>>();
77
78     HashMap<Long,HashMap<String,List<String>>>MapPermission=new HashMap<Long, HashMap<String,List<String>>>();
79
80     ArrayList<String> Donnees =new ArrayList<String>();
81     Donnees.add("Nom");Donnees.add("Prenom");
82
83
84     MapFonction.put("Consulter details patient",Donnees );
85
86     MapPermission.put((long)77, MapFonction);
87     MapPermission.put((long) 67, MapFonction);
88     // document.setDocumentURI("C:\\Users\\DELL\\Desktop\\V02 SIR\\VPE\\XML\\Pati.xml");
89     Element element= Creer_Element_Patient(document, patient_id);
90
91     for(String i: specification.getPermission().keySet()) {
92
93         Element Permission=Creer_Element_Permission(document, i);
94
95         Map<String,List<String>> MF=specification.getPermission().get(i);
96
97         Map<String,List<String>> MF=specification.getPermission().get(i);
98
99         MF.put("Ajouter une maladie", null);
100        MF.put("Ajouter un vaccin", null);
101        MF.put("Ajouter une analyse", null);
102        for(String f: MF.keySet()) {
103            Iterator<String> D = null;
104            if(MF.get(f)!=null)
105                D=MF.get(f).iterator();
106
107
108            Element Fonction=Creer_Element_Fonction(document, f, D);
109
110            Permission.appendChild(Fonction);
111
112        }
113
114        element.appendChild( Permission);
115    }
116
117
118
119
120
121    //System.out.println("***** "+document.getDocumentElement() + " *****"+element.getNodeName());
122
123
124    System.out.println("*****");
125
126
127    TransformerFactory transformerFactory = TransformerFactory.newInstance();
128    Transformer transformer;
129    try {transformer = transformerFactory.newTransformer();
130        transformer.setOutputProperty(OutputKeys.INDENT, "yes");
131        transformer.setOutputProperty("http://xml.apache.org/xslt#indent-amount", "4");
132
133        DOMSource source = new DOMSource(document);
134
135        //File fichiersXML = new ClassPathResource("connexion.jpg").getFile();
136        //File fichiersXML=ResourceUtils.getFile("classpath:static/fichiersXML").getAbsolutePath();
137        System.out.println("ServletContext.getResource(\"/static/\" + \" \");
138        //FileWriter fil=new FileWriter(new File("file:/static/fichiersXML/Patient_"+patient_id+".xml"));
139        FileWriter fil= new FileWriter(new File("C:\\Users\\DELL\\Desktop\\V02 SIR\\VPE\\XML\\Patient_"+patient_id+".xml"));
140        StreamResult result = new StreamResult(fil);
141        transformer.transform(source, result);
142
143
144        System.out.println( result.getWriter()+ " $$$$$$");
145
146
147    } catch (TransformerConfigurationException e) {
148        // TODO Auto-generated catch block
149        e.printStackTrace();
150    } catch (TransformerException e) {
151        // TODO Auto-generated catch block
152        e.printStackTrace();
153    }
154    return false;
155 }
156

```

Figure 26. Création du XML

3.1.3 Consultation des données

Les figures ci-dessous représentent les interfaces affichant les données personnelles pour un patient :

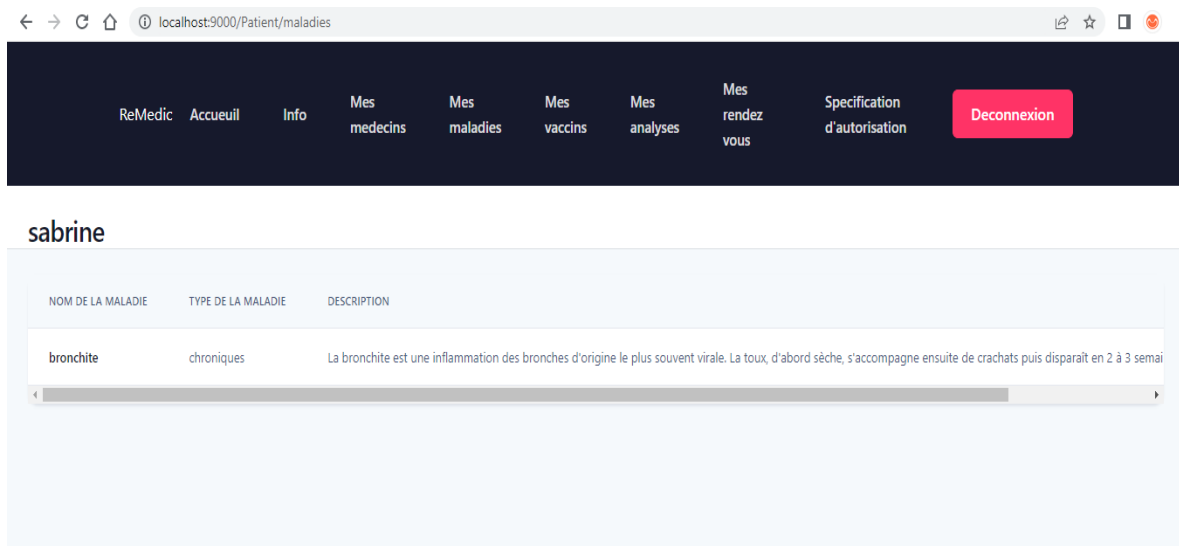


Figure 27. Consultation des maladies

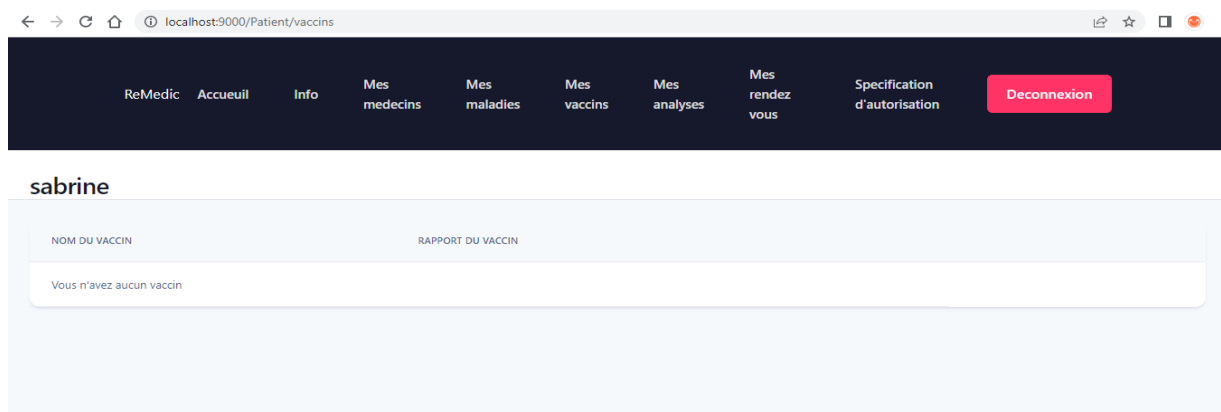


Figure 28. Consultation des vaccins

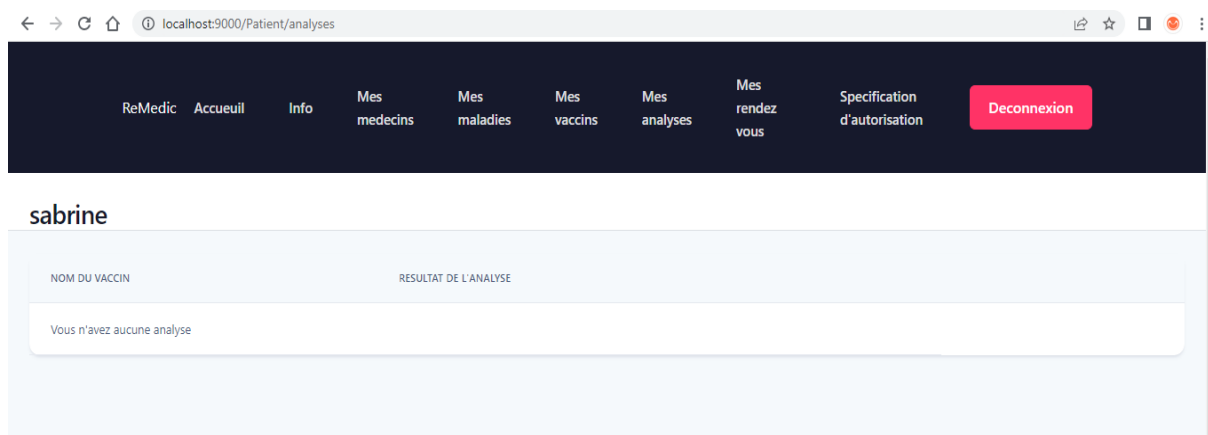


Figure 29. Consultations des analyses

3.2 Rôle assistante

Pour pouvoir effectuer les fonctionnalités suivantes l'assistante doit d'abord s'authentifier.

3.2.1 Consultation des patients et rendez vous

Les figures ci-dessous représentent la consultation des patients et rendez vous du médecin pour le quelle l'assistante authentifiée travail :

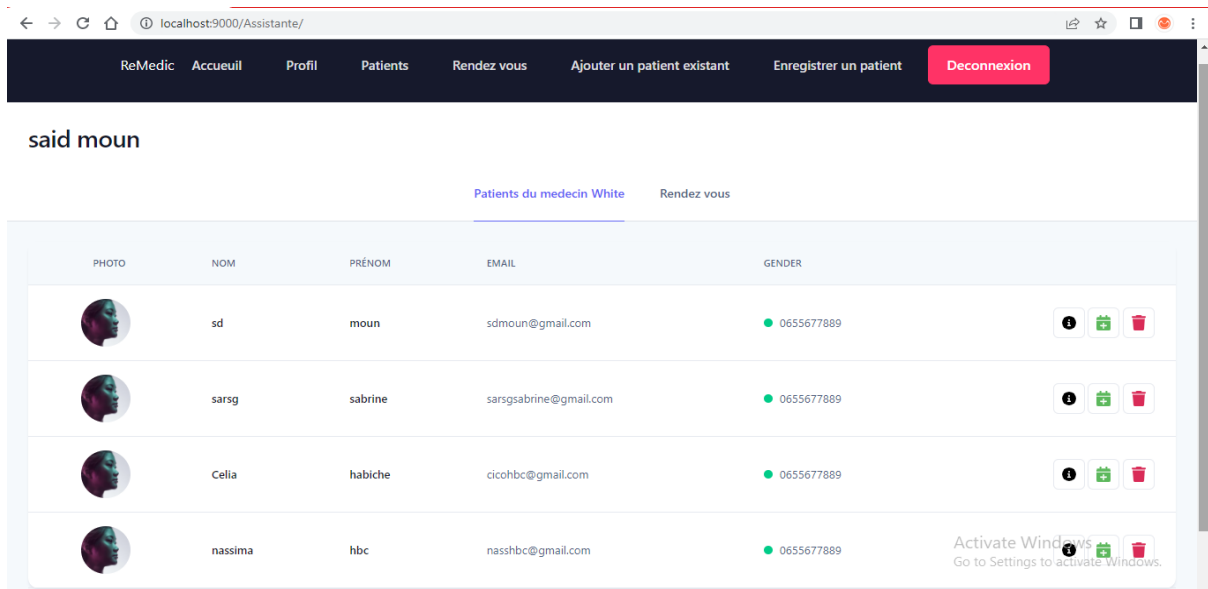


Figure 30. Les patients d'un médecin

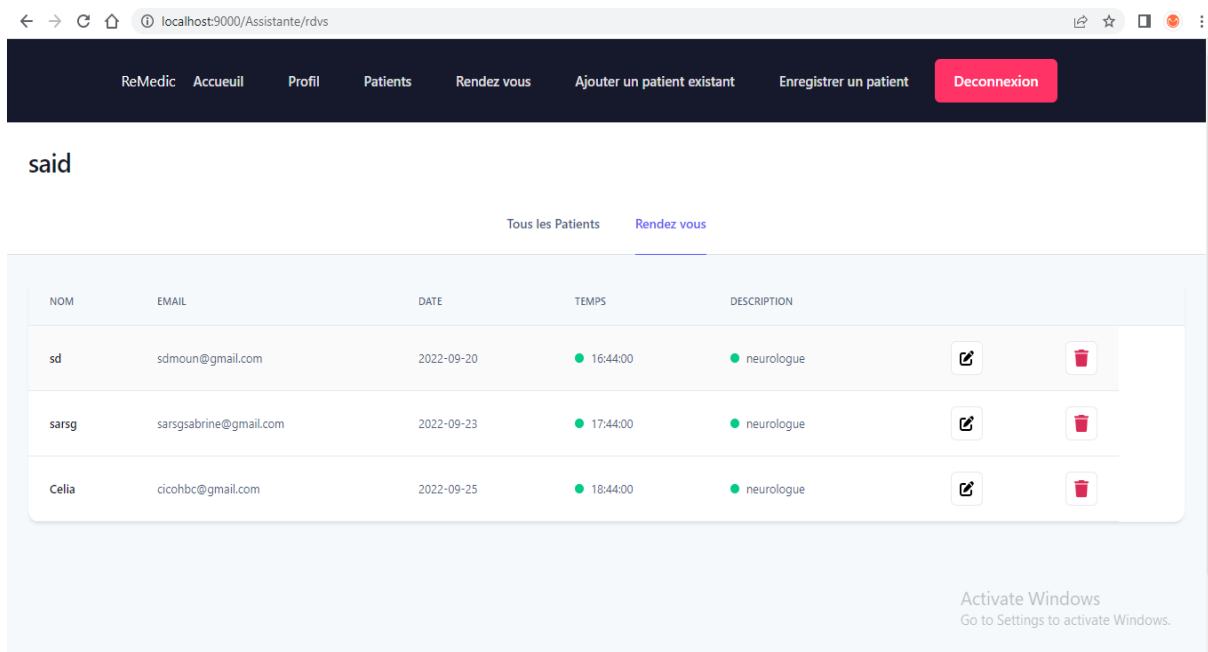


Figure 31. Les rendez vous d'un médecin

Elle peut aussi supprimer ou modifier un rendez vous pour un certain patient depuis les formulaires suivants :

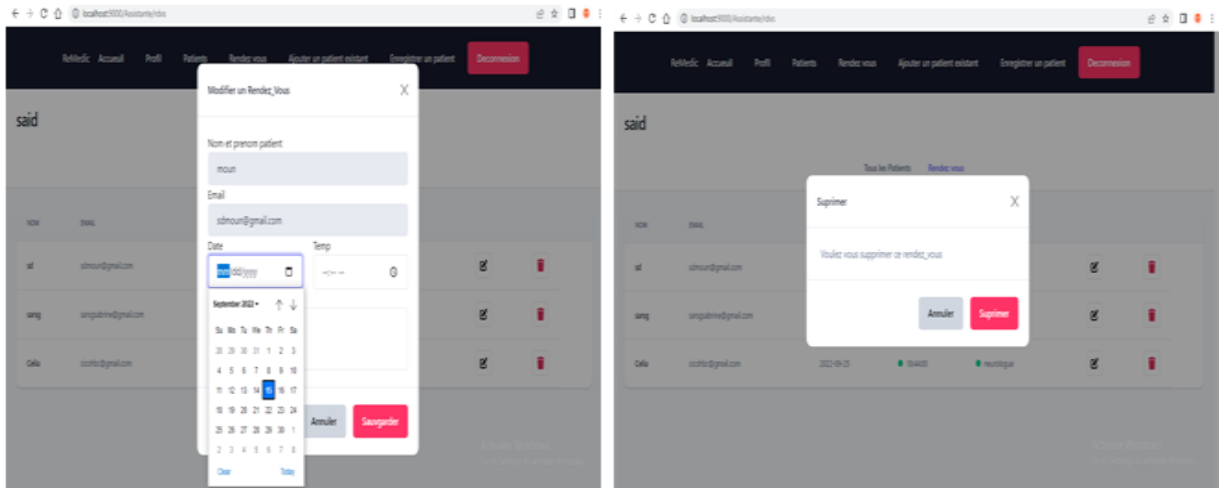


Figure 32. Modifier ou supprimer un rendez vous

Elle peut retirer un patient de la liste du médecin par le bouton le plus à droite dans la ligne des informations ou bien lui prendre un rendez vous et confirmer depuis les interfaces suivantes :

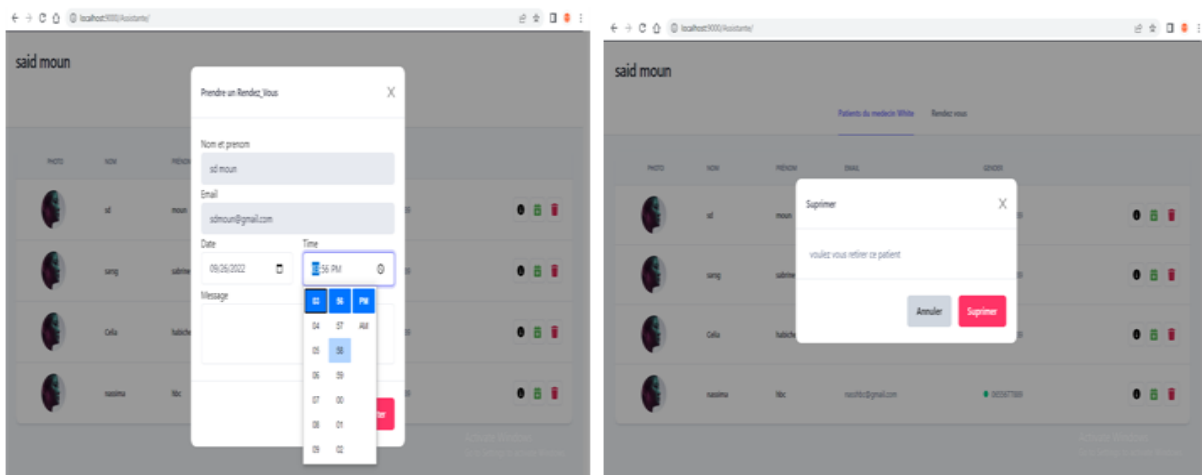


Figure 33. Prendre rendez vous ou retirer un patient

3.2.2 L'ajout d'un patient

Le champ présenté dans le formulaire suivant c'est pour affecter un patient qui s'est déjà enregistré dans l'application à un médecin :

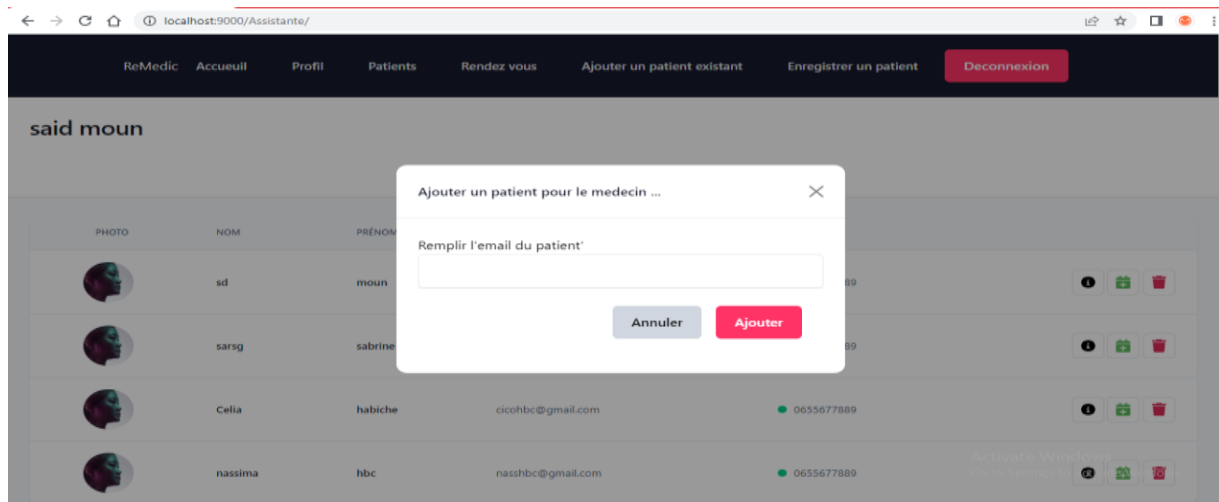


Figure 34. Affecter un patient a un médecin

Le formulaire suivant, c'est pour enregistrer un patient dans l'application et en même temps l'affecter à un médecin :

A screenshot of a web application showing a patient registration form. The form is titled 'Information d'enregistrement' and 'Formulaire de renseignement d'un Patient'. It includes a sub-header 'Veuillez remplir les informations concernant le patient'. The form fields are: Date d'aujourd'hui (09/15/2022), Nom du Patient * (Nom and Prenom), Sexe * (Masculin), Téléphone * (masked with #), Email * (MMMM@gmail.com), Date de Naissance * (mm/dd/yyyy), Date de Naissance * (mm/dd/yyyy), Le Patient est un professionnel de Santé (Non), Statut Marital * (Célibataire, Divorcé(e), Marié(e), Veuf/ve), Adresse du Patient * (Adresse Rue), and Groupage * (Groupage). A button 'Enregistrer patient' is at the bottom. There is also a watermark 'Activate Windows Go to Settings to activate Windows' on the right side.

Figure 35. Enregistrer un patient

3.3 Rôle médecin

3.3.1 Consultation des patients et rendez vous

Depuis l'interface ci-dessous un médecin peut consulter ses patients ainsi que ses rendez vous les modifier ou supprimer par les boutons à droite:

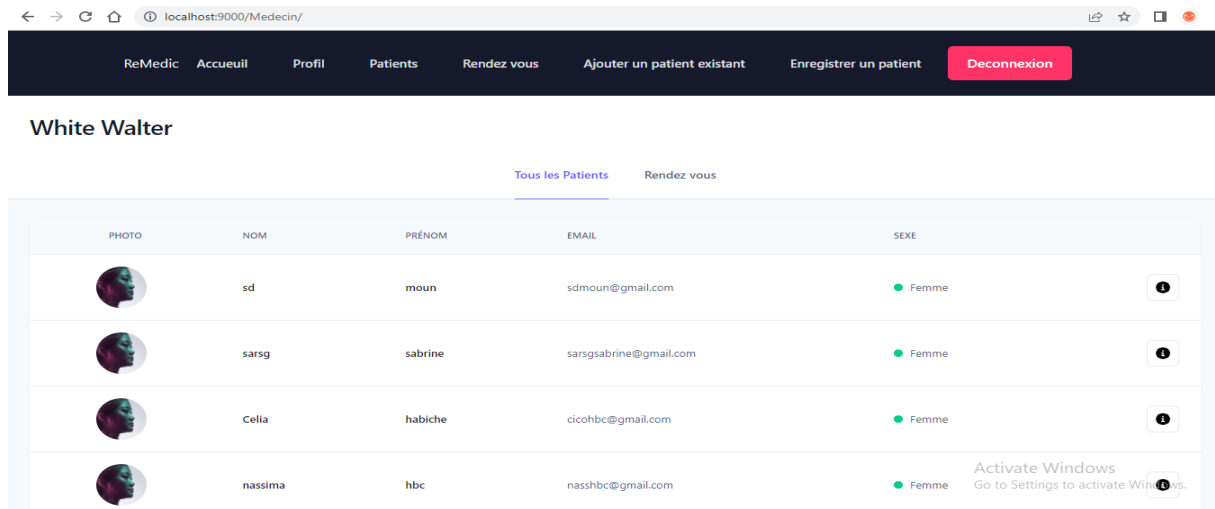


Figure 36. Consulter patients

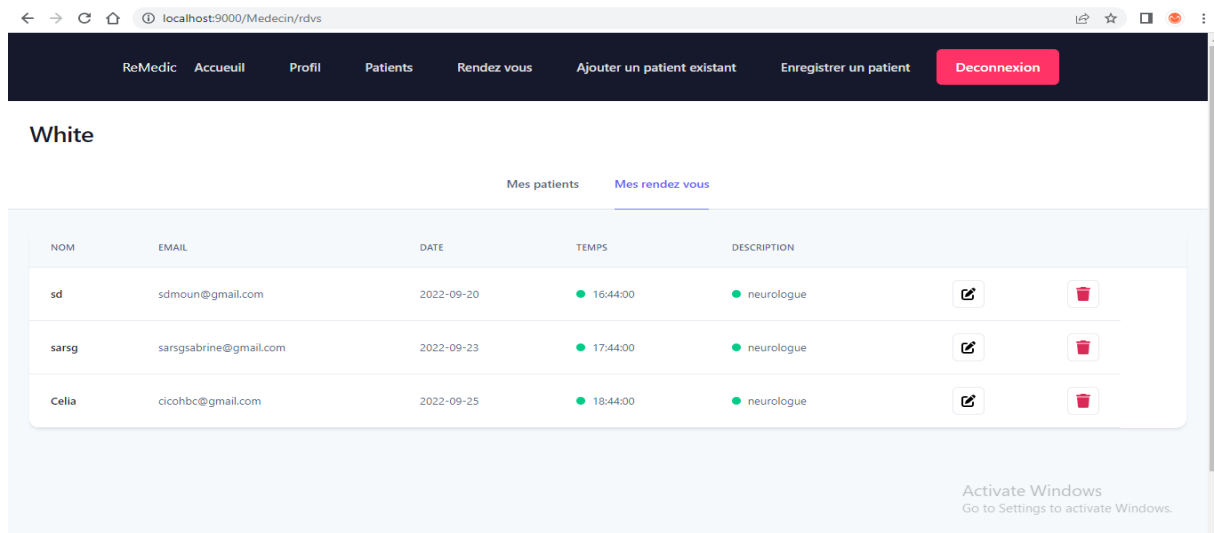


Figure 37. Consulter rendez vous

3.3.2 Consultation des données d'un patient

Dans la liste des patients du médecin depuis le bouton à droite il peut accéder à l'interface qui lui permet de consulter les données et informations du patient souhaité ainsi que d'effectuer certaines fonctionnalités sur ses données. L'interface contenant ces informations est la suivante :

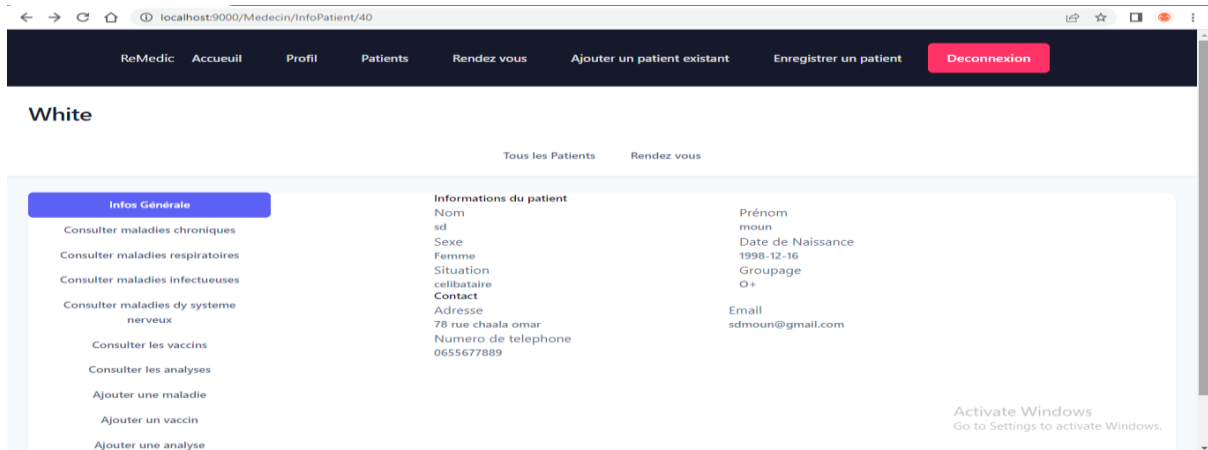


Figure 38. Consulter informations patient

3.3.3 Ajouter des données pour un patient

Depuis les formulaires présentés dans la figure suivante un médecin peut remplir un dossier d'un patient :

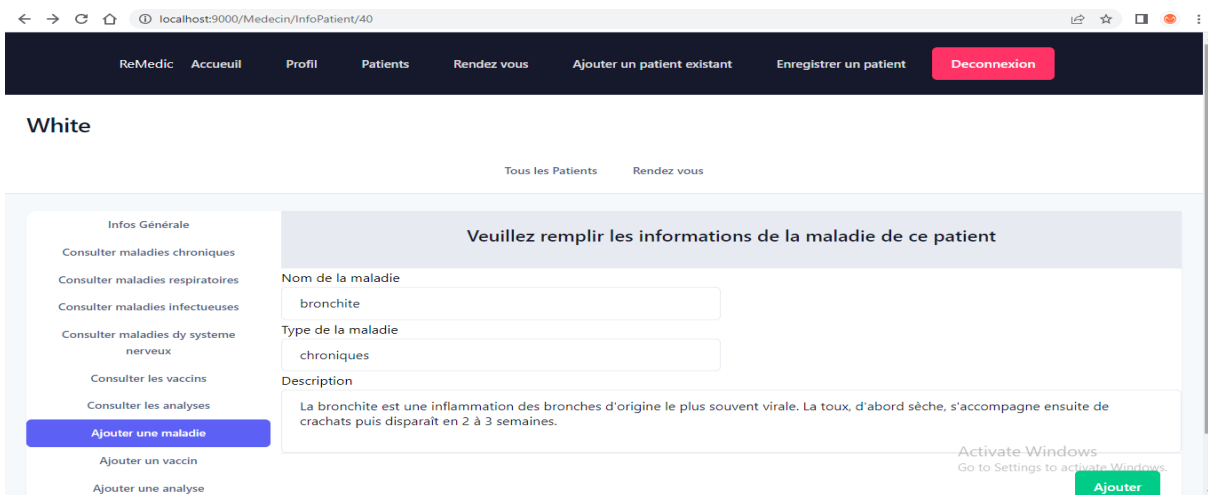


Figure 39. Ajouter une maladie

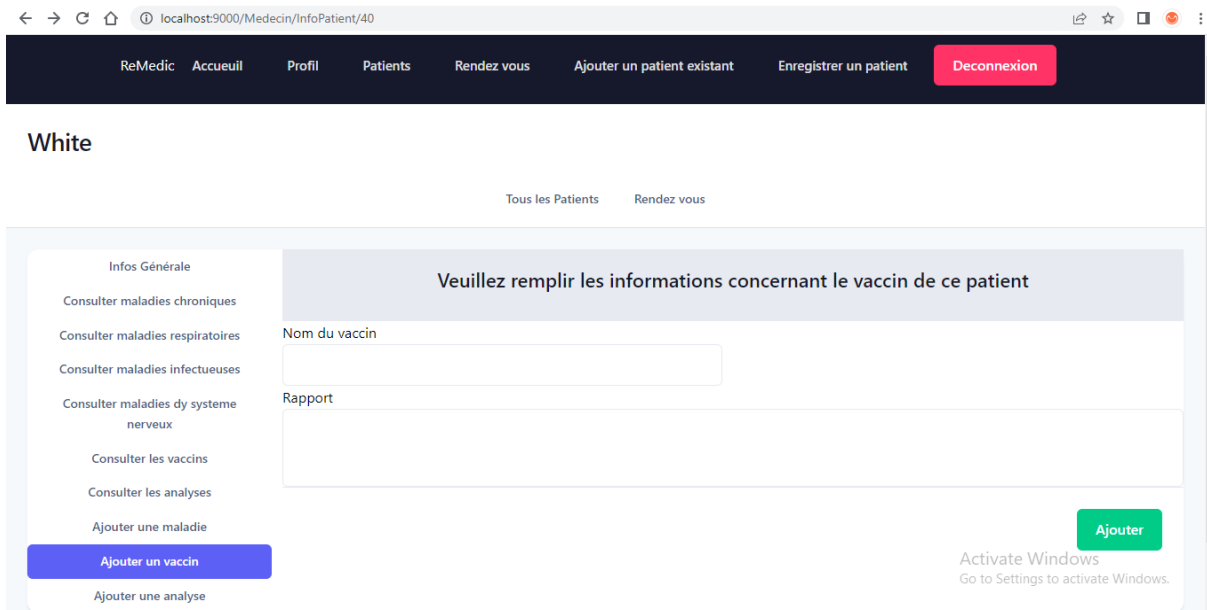


Figure 40.Ajouter un vaccin

3.4 Rôle Administrateur (Chef clinique)

3.4.1 Enregistrer un médecin et/ou assistante

Depuis les interfaces suivantes un administrateur peut enregistrer un médecin ou bien une assistante et l'attribuer à un médecin :

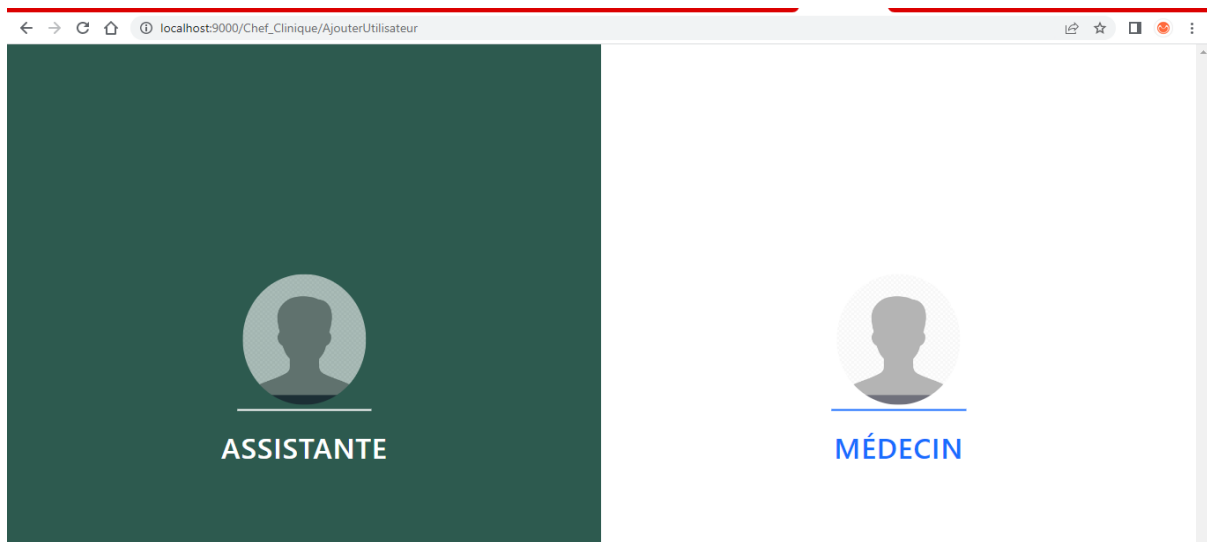


Figure 41.Nouvel utilisateur

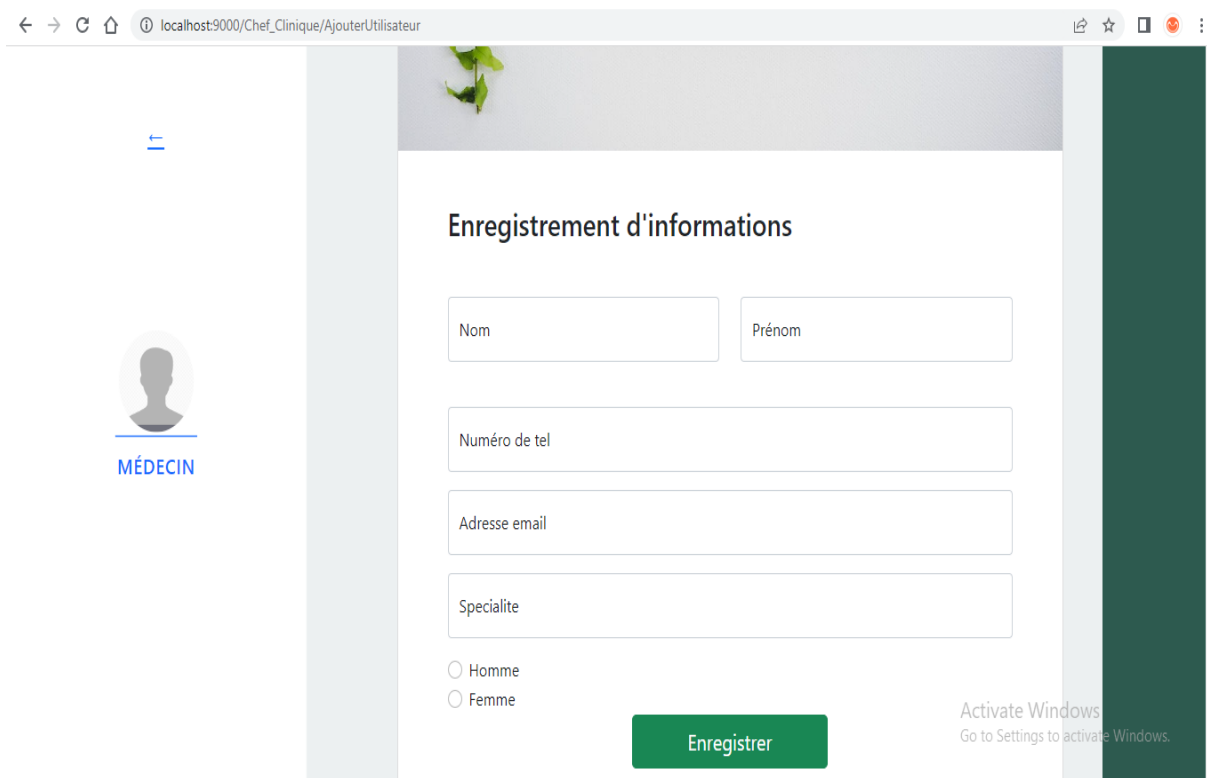


Figure 42. Nouveau médecin

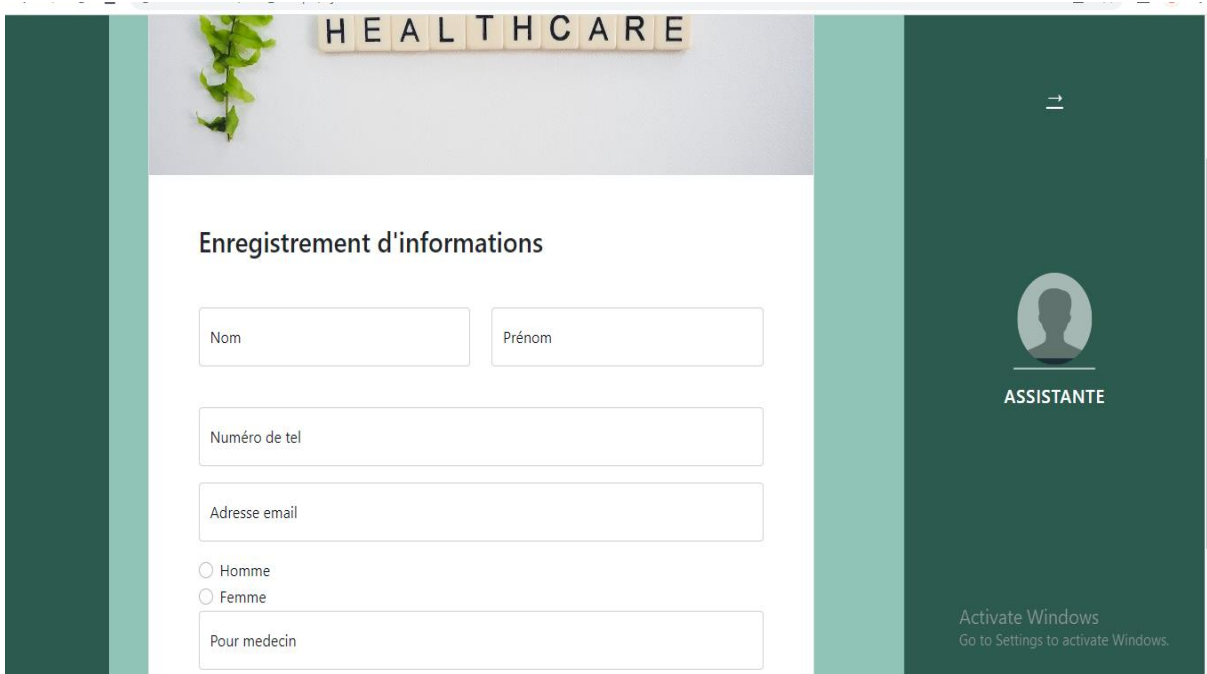


Figure 43. Nouvelle assistante

3.4.2 Consulter la liste des médecins et assistantes

L'interface pour consulter les médecins et assistantes enregistrés dans l'application est celle-ci-dessous :

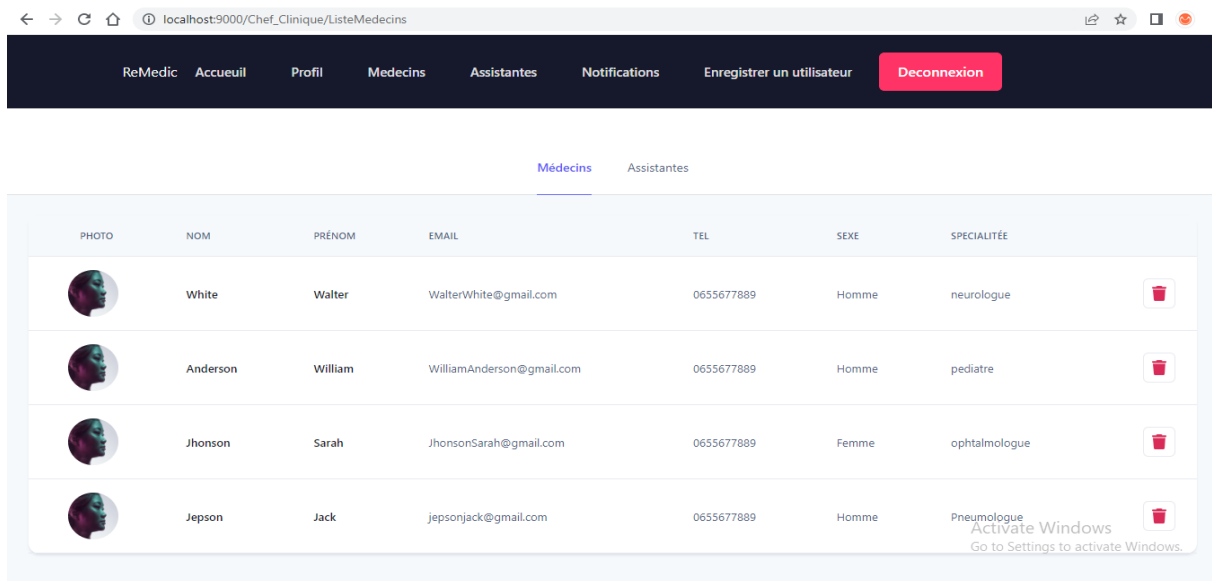


Figure 44. Consulter médecins/assistantes

3.4.3 Supprimer un médecin et/ou assistante

Depuis le bouton à droite dans l'interface de consultations des médecins/assistantes un administrateur a la possibilité de supprimer un utilisateur de la base de données et confirmer la suppression comme montré dans la figure suivante :

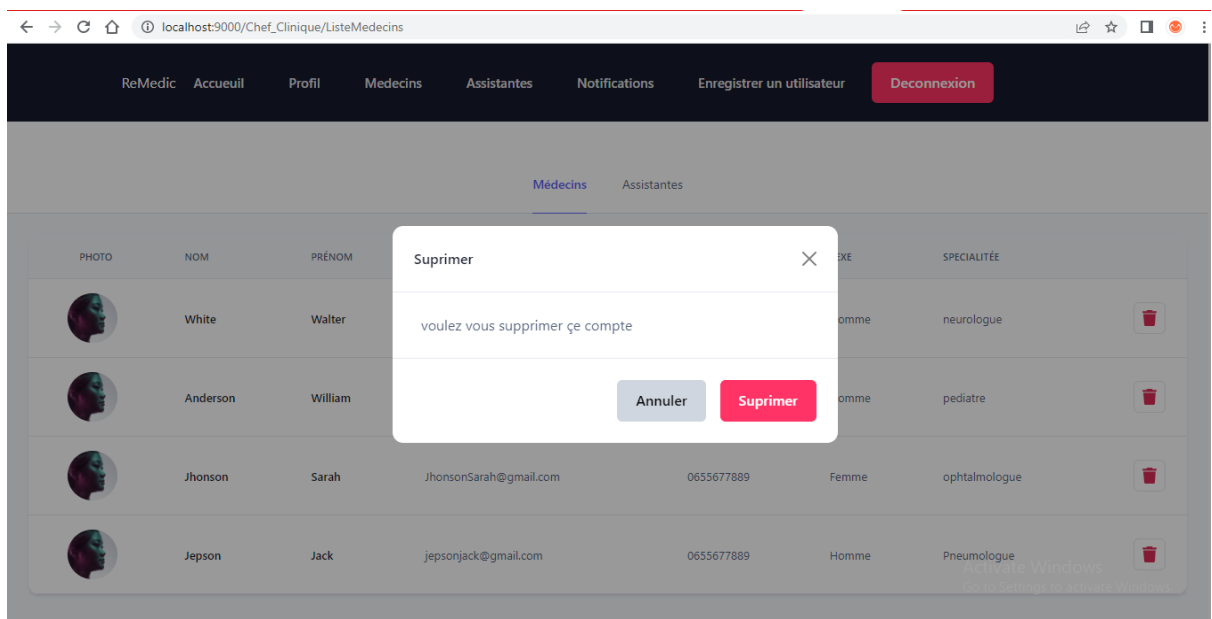


Figure 45. Suppression d'utilisateur

3.4.4 Consulter les notifications

Si un utilisateur essaye d'effectuer une fonctionnalité dont le droit ne lui a pas été spécifié l'administrateur sera informé en recevant une notification qui puisse être consultée depuis l'interface suivante :

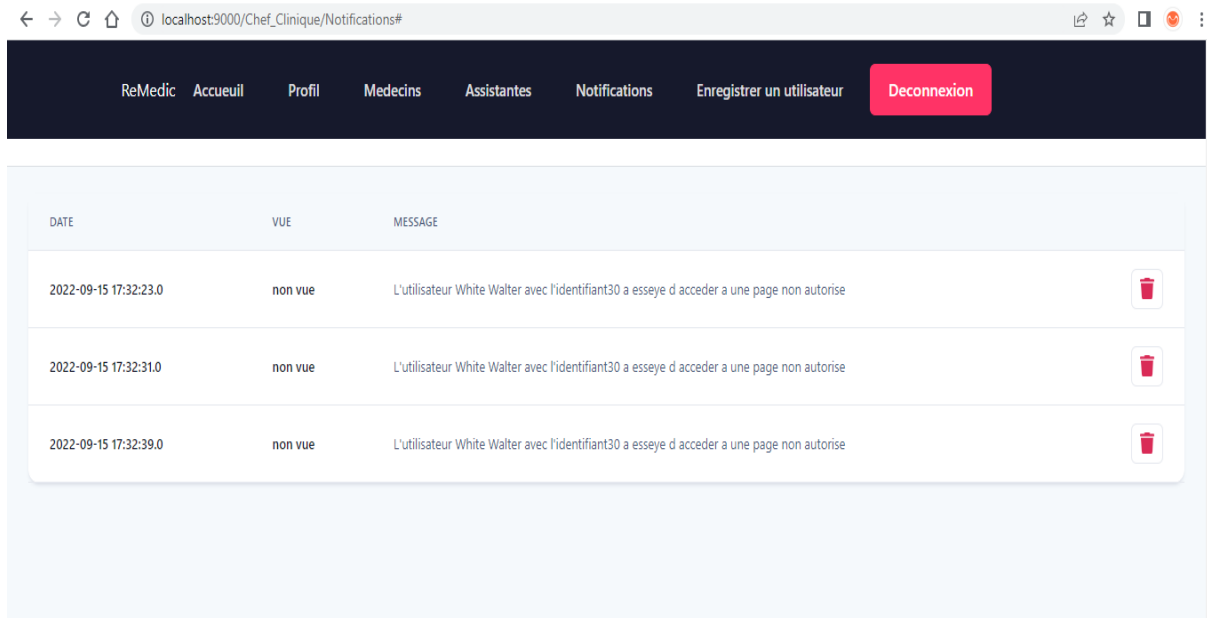


Figure 46. Consultation des notifications

3.5 Interface d'utilisateur

On va présenter les interfaces d'authentification, de consultation de profil et celle qui représente un accès non autorisé à un utilisateur :

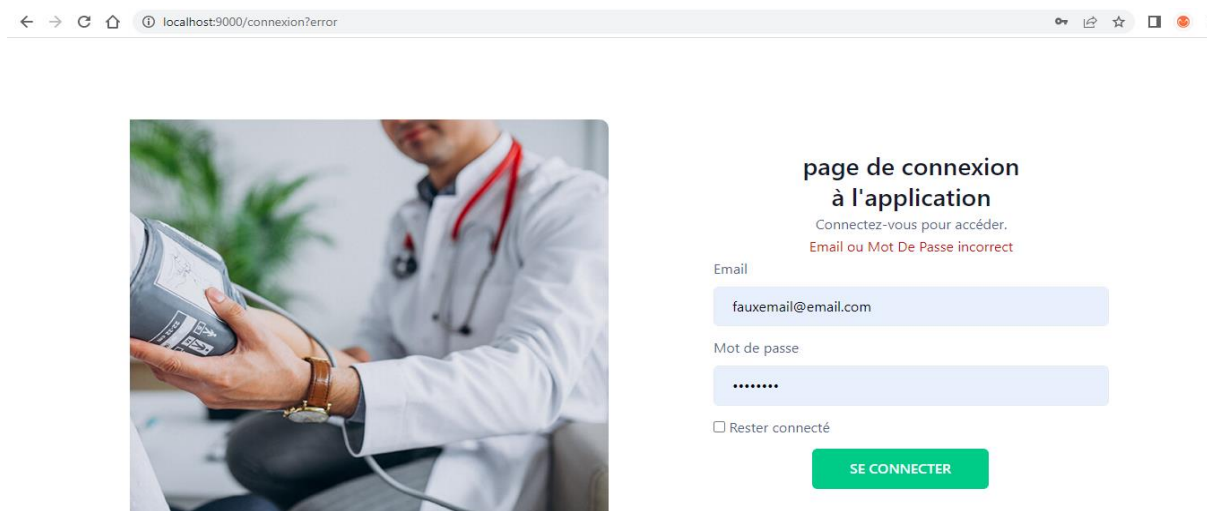


Figure 47. Interface d'authentification

Depuis l'interface (figure 47) d'authentification, l'utilisateur doit remplir ses coordonnées exactes pour pouvoir se connecter.

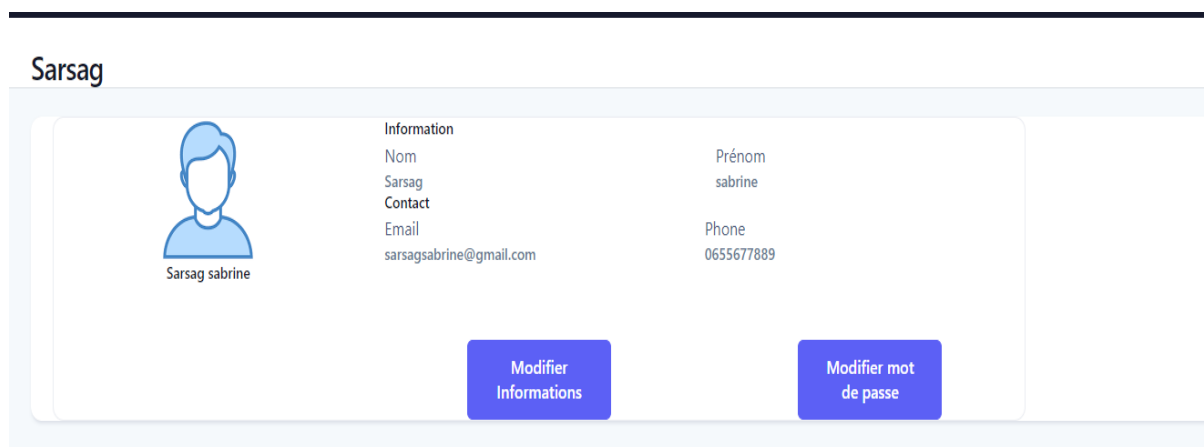


Figure 48. Consulter informations

Chaque utilisateur peut consulter son profil et modifier ses informations ainsi que son mot de passe en cliquant sur les boutons en bas de page de l'interface de la figure 48 pour ensuite remplir les formulaires suivants :

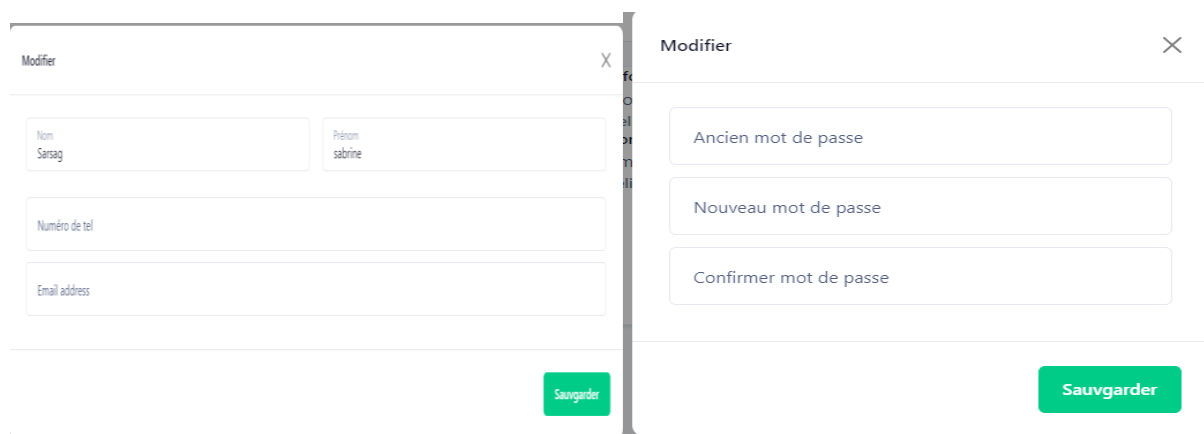


Figure 49. Modifier informations personnelles

Les mots de passes des utilisateurs sont générés automatiquement après l'enregistrement pour ensuite être modifiés par le propriétaire du compte.

Voici le code d'enregistrement d'un utilisateur (patient) :

```
135
136
137 /*-----Enregistrer Patient--> retour Accueil-----*/
138
139 /*_Reafficher les patient propre a chaque medecin_*/
140
141 @PostMapping("/EnregistrerPatient")
142 public ModelAndView AjouterPatient(@ModelAttribute Patient patient, Principal principal){
143     Assistante assistante=assistanteService.getAssistante(principal.getName());
144     patient.setMDP(passwordEncoder.passwordEncoder().encode(patient.getNom()+patient.getPrenom()));
145     patient.getMedecins().add(assistante.getMedecin());// recup depuis la session de l'assistante le medecin
146     patientservice.EnregistrerPatient(patient);
147     System.err.println("*****");
148
149     //patientservice.EnregistrerPatient(patient);
150     return new ModelAndView("redirect:/Assistante/");
151 }
152
```

Figure 50. Code d'enregistrement d'un patient

Les mots de passe des utilisateurs sont enregistrés dans le SGBD après leur chiffrement avec la fonction BCrypt.



Figure 51. Accès non autorisé

Si un utilisateur essaye d'accéder à une page dont il n'a pas la permission, il sera renvoyé vers l'interface de la figure 51.

4 Simulation de la solution

D'après l'interface de spécification d'autorisations présentée dans la figure 24, on remarque que le patient a sélectionné la donnée diabète pour la consultation des maladies chroniques pour les médecins White et Jepson et seulement la donnée bronchite pour le médecin Jepson. Ainsi, on va se connecter avec le compte d'utilisateur Jepson pour effectuer les fonctionnalités permises par le modèle RBAC dans la figure 38 et vérifier si les contraintes de spécifications sont praticables.

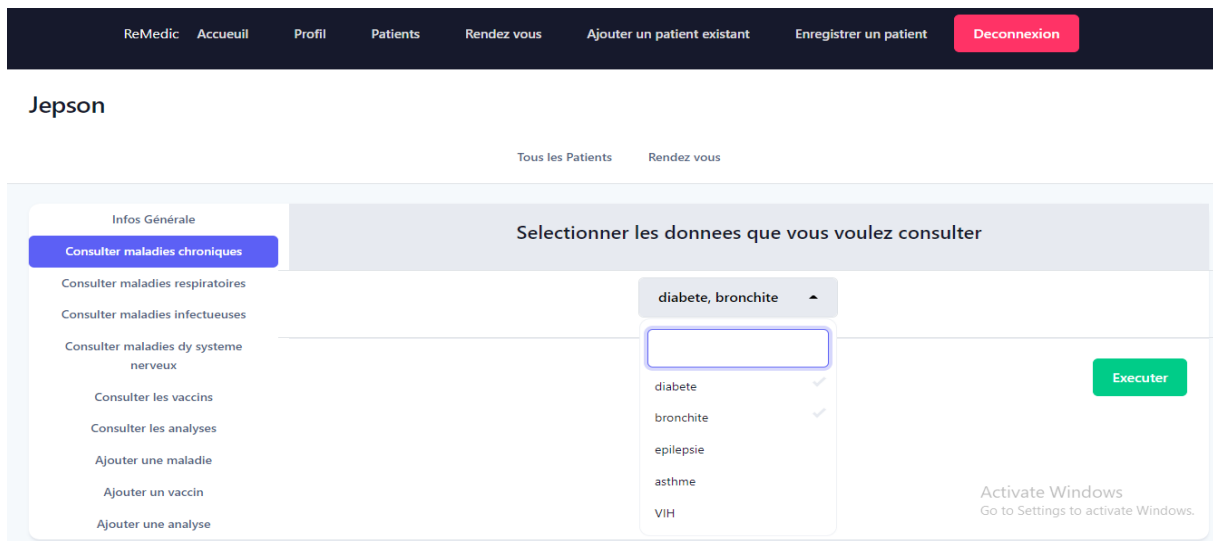


Figure 52. Sélection des données

Avec le compte du médecin Jepson on a sélectionné les données spécifiées par le patient et exécuter la fonction consulter maladies chroniques, le résultat est dans la figure suivantes :



Figure 53. Résultat de l'opération

Les maladies affichées sont celles présentes dans la figure 27, la maladie chronique diabète ne s'affiche pas à l'écran du médecin bien qu'il a le droit de la consulter parce que le patient ne l'a pas. S'il essaye de consulter seulement cette donnée en sélectionnant diabète seulement depuis l'interface 52, voici ce qui sera affiché :



Figure 54. Résultat de changement de donnée

Maintenant, si on essaye d'effectuer une autre opération ou bien essayer d'effectuer la même en sélectionnant des données non spécifiées, voici le résultat de l'exécution :

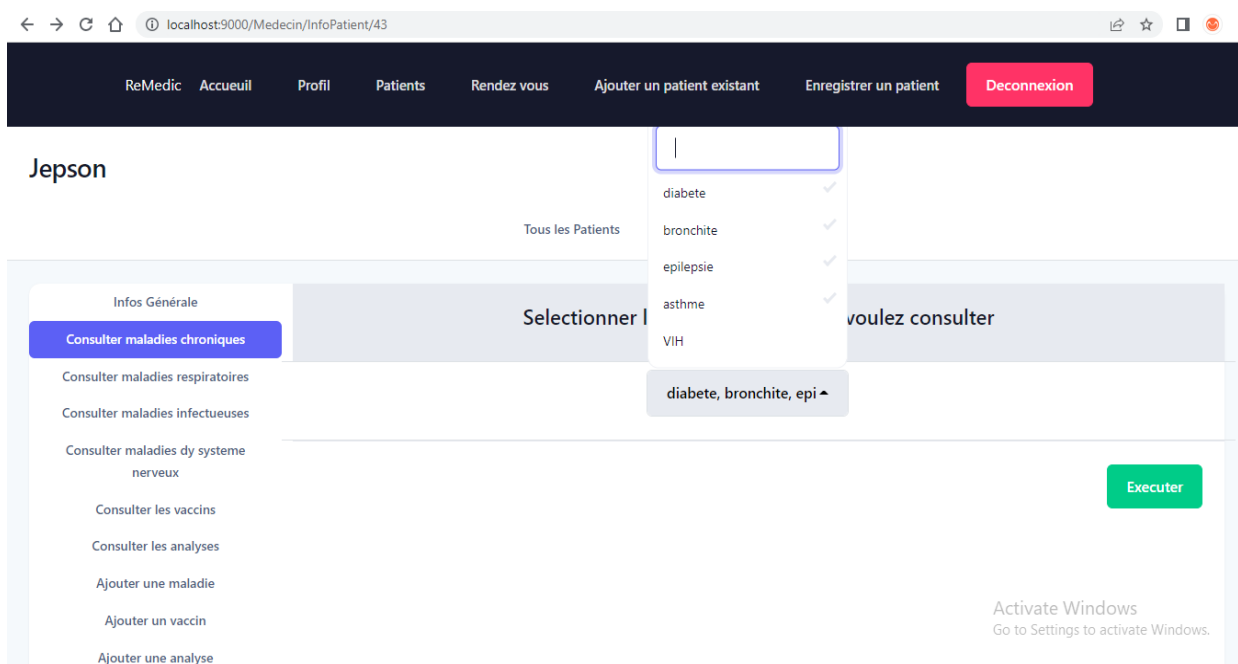


Figure 55. Sélection de données autorisées et non autorisées

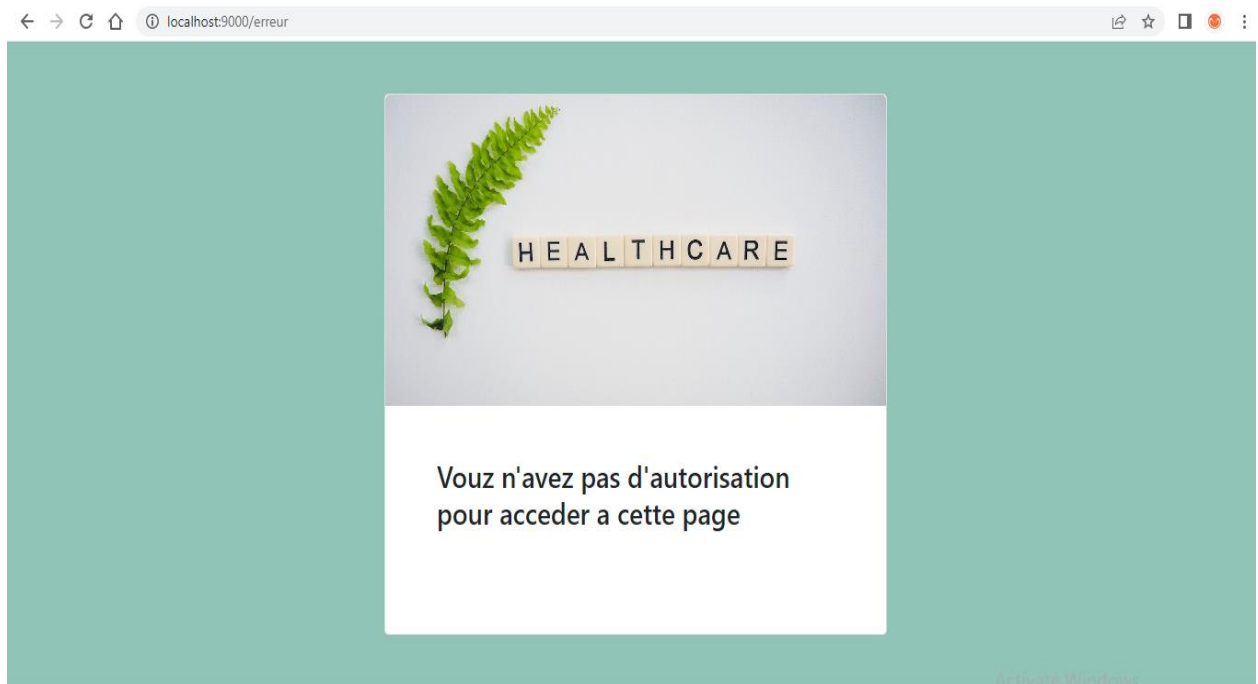


Figure 56. Résultat d'échec d'opération

Pour vérifier, voici le contenu du fichier XML du patient :

```

1  [?xml version="1.0" encoding="UTF-8" standalone="no"?>
2  <Patient xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" patient_id="43" xsi:noNamespaceSchemaLocation="C:\Users\DELL\Desktop\12 SIR\1PFE\XML\Patient.xsd">
3  <Permission utilisateur_id="33">
4  <Fonction>
5  <NomFonction>Ajouter une maladie</NomFonction>
6  </Fonction>
7  <Fonction>
8  <NomFonction>Ajouter un vaccin</NomFonction>
9  </Fonction>
10 <Fonction>
11 <NomFonction>Ajouter une analyse</NomFonction>
12 </Fonction>
13 <Fonction>
14 <NomFonction>Consulter maladies chroniques</NomFonction>
15 <Donnee>diabete</Donnee>
16 <Donnee>bronchite</Donnee>
17 </Fonction>
18 </Permission>
19 <Permission utilisateur_id="30">
20 <Fonction>
21 <NomFonction>Ajouter une maladie</NomFonction>
22 </Fonction>
23 <Fonction>
24 <NomFonction>Ajouter un vaccin</NomFonction>
25 </Fonction>
26 <Fonction>
27 <NomFonction>Ajouter une analyse</NomFonction>
28 </Fonction>
29 <Fonction>
30 <NomFonction>Consulter maladies chroniques</NomFonction>
31 <Donnee>diabete</Donnee>
32 </Fonction>
33 </Permission>
34 <Permission utilisateur_id="31">
35 <Fonction>
36 <NomFonction>Ajouter une maladie</NomFonction>
37 </Fonction>
38 <Fonction>
39 <NomFonction>Ajouter un vaccin</NomFonction>
40 </Fonction>
41 <Fonction>
42 <NomFonction>Ajouter une analyse</NomFonction>
43 </Fonction>
44 </Permission>
45 <Permission utilisateur_id="32">
46 <Fonction>
47 <NomFonction>Ajouter une maladie</NomFonction>
48 </Fonction>
49 <Fonction>
50 <NomFonction>Ajouter un vaccin</NomFonction>
51 </Fonction>
52 <Fonction>
53 <NomFonction>Ajouter une analyse</NomFonction>
54 </Fonction>
55 </Permission>
56 </Patient>
57

```

```

34 <Permission utilisateur_id="31">
35 <Fonction>
36 <NomFonction>Ajouter une maladie</NomFonction>
37 </Fonction>
38 <Fonction>
39 <NomFonction>Ajouter un vaccin</NomFonction>
40 </Fonction>
41 <Fonction>
42 <NomFonction>Ajouter une analyse</NomFonction>
43 </Fonction>
44 </Permission>
45 <Permission utilisateur_id="32">
46 <Fonction>
47 <NomFonction>Ajouter une maladie</NomFonction>
48 </Fonction>
49 <Fonction>
50 <NomFonction>Ajouter un vaccin</NomFonction>
51 </Fonction>
52 <Fonction>
53 <NomFonction>Ajouter une analyse</NomFonction>
54 </Fonction>
55 </Permission>
56 </Patient>
57

```

Figure 57. Fichier XML du patient id=33

On remarque la présence de certaines fonctionnalités qui n’ont pas été spécifiées depuis le formulaire, ces fonctions sont ajoutées au fichier XML par défaut, des opérations que chaque médecin a le droit d’effectuer sur un dossier de son patient.

On va essayer d’effectuer l’opération ajouter une maladie de type chroniques de nom diabète et consulter cette donnée encore une fois et voir le résultat. Voici les figures montrant ces étapes :

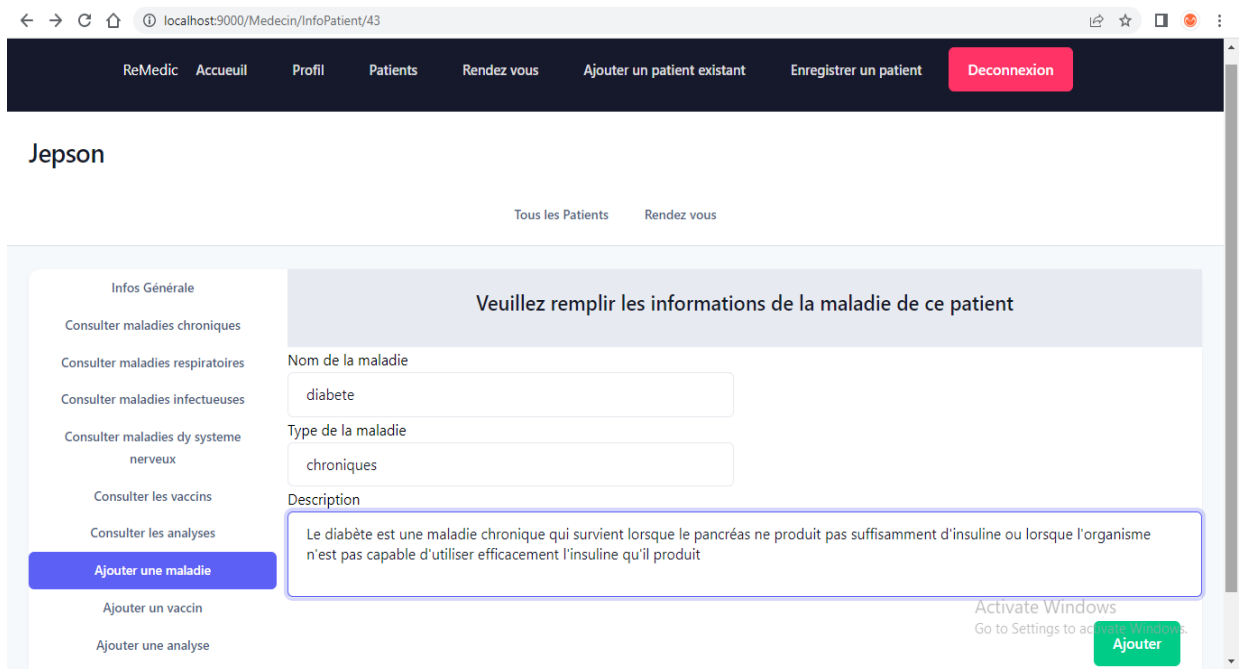


Figure 58. Ajouter une maladie chronique au nom diabète

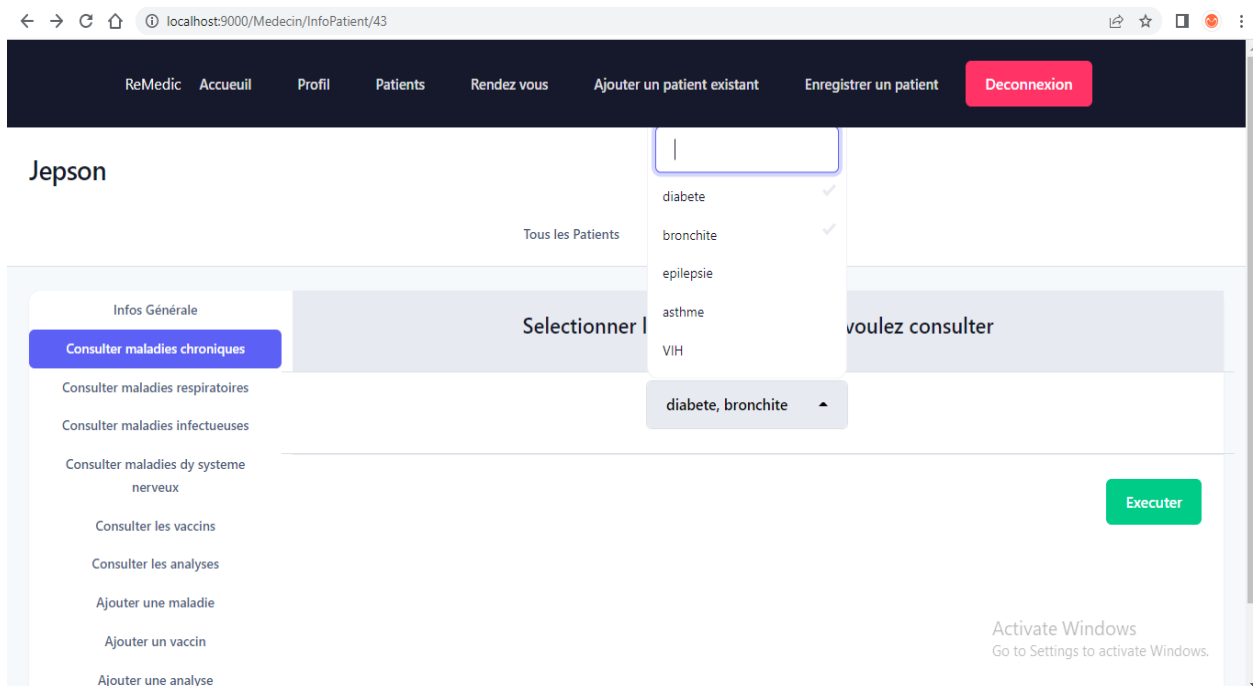


Figure 59. Réexécution de l'opération de figure 51

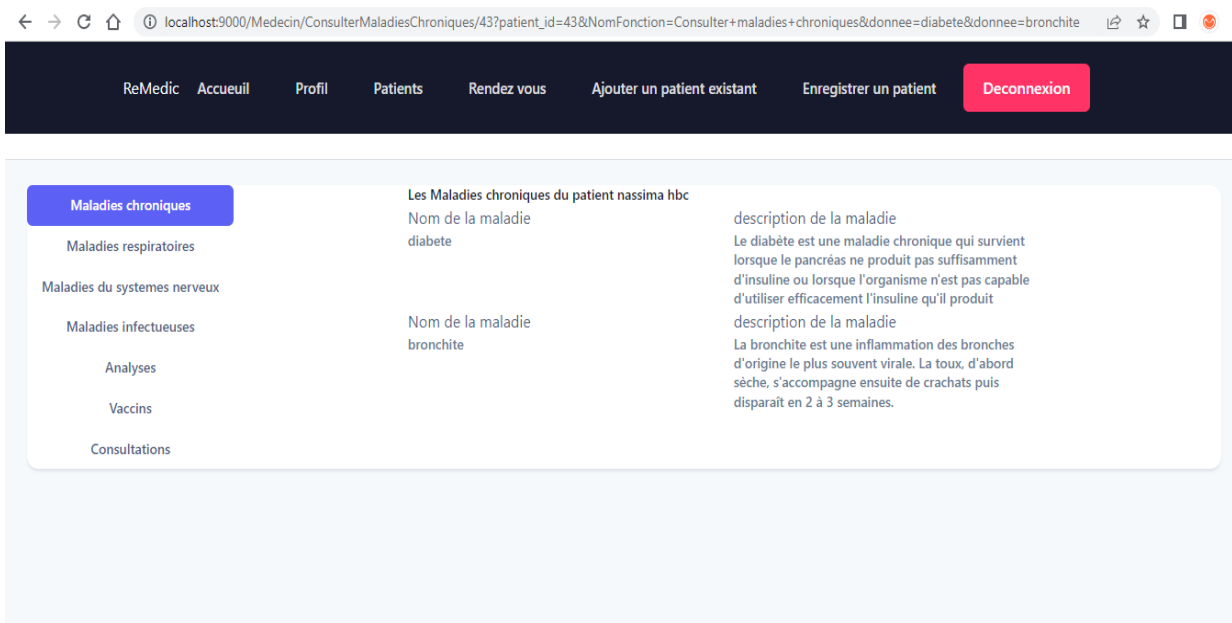


Figure 60. Résultat de la réexécution

Voici une partie du code exécuté pour vérifier le droit d'effectuer les opérations précédentes pour un certain utilisateur sur un dossier d'un certain patient :

```

159
160 public Boolean autorisation(String NomFonction,List<String>liste, Long patient_id, Long utilisateur_id) throws XPathExpressionException {
161     String fonctionnalite="Consulter vaccins";
162     XPathFactory xpf=XPathFactory.newInstance();
163     XPath path=xpf.newXPath();
164
165     //liste.add("Nom");
166
167     //liste.add("Age");
168
169     Node permission=verifier_permission_utilisateur(utilisateur_id, patient_id, path);
170     //System.out.println("_____");
171
172     // System.out.println("permission "+permission);
173     if(permission!=null) {
174         System.out.println("permission not null ");
175         Node Fonction=verifier_fonctionnalite(NomFonction, permission, path);
176
177         System.out.println("verifier fonct "+Fonction);
178         if(Fonction!=null) {
179             System.out.println(Fonction);
180             Boolean donnees=(Boolean) path.evaluate("Donnee", Fonction, XPathConstants.BOOLEAN);
181             // System.out.println(donnees);
182             //System.out.println("esQ kaian un noeud donnees dakhal la fonction "+donnees);
183
184             if(donnees) {
185
186                 System.err.println("donnee");
187                 //ajouterdonnee(Fonction, path);
188                 //System.out.println(donnees);
189                 if(!liste.isEmpty()) {
190                     //System.out.println("liste non vide"+ !liste.isEmpty());
191
192                     if(verifier_donnees(liste, Fonction, path)) return true;
193                     else return false;
194                     // System.out.println("vous n'etes pas autorise a acceder au donnees demandees");
195
196
197                 }else return true;
198                 //else return false;
199                 //System.out.println("vous n'etes pas autorise a effectuer cette fonction sans preciser les donnees");
200                 }else return true;
201                 //System.out.println("vous etes autorise");
202                 }else return false; //System.out.println("vous n'etes pas autorise a effectuer cette fonction");
203
204             }else return false; //System.out.println("vous n'avez pas cette permission");
205
206         }
207     }
208 }

```

Figure 61. Code de vérification d'autorisation

5 Conclusion

Dans ce chapitre, on a parlé des outils de développement ainsi que les technologies utilisées pour la réalisation de notre application, on a présenté les différentes interfaces graphiques qui composent les espaces des utilisateurs de notre système en exposant les opérations qui constituent chacun. On a démontrée le travail du middleware avec une simulation et en effectuant des essayes soumis à plusieurs contraintes.



Conclusion Générale et Perspective

1 Conclusion et perspective

L'objectif visé à travers ce travail est de s'intéresser à la problématique de la gestion des accès aux données médicales afin de pouvoir partager les données médicales sensibles entre les professionnels de santé autorisés tout en préservant la confidentialité, la déontologie médicale et la vie privée des patients.

Pour répondre à cette problématique, nous avons fait une étude bibliographique que nous avons répartie sur quatre chapitres.

Dans le premier chapitre nous avons parlé sur les concepts de base de sécurité et les différentes approches de contrôle d'accès. Ainsi la politique de contrôle d'accès. Ensuite dans le deuxième chapitre nous avons présenté les concepts de la couche médiatrice entre le back-end et le front-end, le middleware.

Après dans le chapitre trois, nous avons proposé une solution de la couche de sécurité qui combine les deux approches de sécurité ; La première approche Concerne Le Contrôle d'accès basé sur les rôles (RBAC) et la deuxième approche concerne le formulaire de spécification.

D'autre part, ces deux approches assurent la protection des données du patient car elles permettent aux utilisateurs autorisés l'accès de ces derniers et interdites utilisateurs non autorisés. Cette solution fournit une couche de sécurité.

Pour mettre en œuvre notre solution, nous avons implémenté les deux approches dans une application web (Gestion d'accès aux données médicale) en utilisant le Framework Spring boot (langage java).

Par ailleurs, notre solution de soulève un certain nombre de questions ouvertes intéressantes telles que:

- Tester la solution dans différents domaines de système ouvert par exemple : les I.O.T.
- Mise à l'échelle
- L'utilisation de plusieurs plateformes et différents systèmes de SGBD SQL et NOSQL en même temps.



BIBLIOGRAPHIE

1 Bibliographie

- [1] Raphael Yende. « Support de cours de sécurité informatique et crypto », Master.CongoKinshasa. 2018.cel-01965300
- [2] GHERNAOUTI Solange, « Sécurité informatique et réseaux », Dunod 5ème édition, Livre, Année 2016.
- [3] KHALIL, A., MBAREK, N., & TOGNI, O. (2022). Adaptation du contrôle d'accès pour la sécurité de l'IoT. La gestion et le contrôle intelligents des performances et de la sécurité dans l'IoT, 169.
- [4] GHERNAOUTI Solange, « Cyber sécurité analyser les risques mettre en œuvre les solutions », Dunod 6ème édition, Livre, Année 2019.
- [5] Ferraiolo, D., Chandramouli, R., Kuhn, R., & Hu, V. (2016, March). Extensible access control markup language (XACML) and next generation access control (NGAC). In Proceedings of the 2016 ACM International Workshop on Attribute Based Access Control (pp. 13-24).
- [6] TORCHI, N. Contrôle d'accès basé sur les rôles (RBAC) et les attributs (ABAC) dans le Cloud Computing.
- [7] MATALLAH, A., BABAHADJ, A., & KADDI, M. (2017). SYSTÈME DE CONTROLE D'ACCES PHYSIQUE (Doctoral dissertation, Université Ahmed Draïa-Adrar).
- [8] Uttha, W. (2016). Etude des politiques de sécurité pour les applications distribuées: le problème des dépendances transitives: modélisation, vérification et mise en œuvre (Thèse de Doctorat, Université d'Aix-Marseille), pp. 24-27.
- [9] Bertrand, Y. (2017). Gestion du contrôle de la diffusion des données d'entreprises et politiques de contrôles d'accès (Doctoral dissertation, Université Côte d'Azur (ComUE)).
- [10] Alban Gabillon. Contrôler les accès aux données numériques, 2013, pp. 2-4.
- [11] YUCEF, K. Une approche hybride pour le contrôle d'accès.
- [12] David, F., & Richard, K. (1992). Role-Based access controls. In Proceedings of 15th NIST-NCSC National Computer Security Conference (Vol. 563). Baltimore, Maryland: NIST- NCSC.

- [13] Kumar, A., Karnik, N., & Chafle, G. (2002). Context sensitivity in Role-Based access control. *ACM SIGOPS Operating Systems Review*, 36(3), pp. 53-66.
- [14] Baïna, A. (2009). Contrôle d'accès pour les grandes infrastructures critiques. Application au réseau d'énergie électrique (Doctoral dissertation, INSA de Toulouse).
- [15] Hu, V. C., Ferraiolo, D., Kuhn, R., Friedman, A. R., Lang, A. J., Cogdell, M. M., ... & Scarfone, K. (2013). Guide to attribute based access control (abac) definition and considerations (draft). NIST special publication, 800(162), 1-54.
- [16] Ferraiolo, D., Chandramouli, R., Hu, V., & Kuhn, R. (2016). A comparison of attribute based access control (ABAC) standards for data service applications. NIST Special Publication, 800, 178.
- [17] Das, S., Mitra, B., Atluri, V., Vaidya, J., & Sural, S. (2018). Policy engineering in RBAC and ABAC. In *From Database to Cyber Security* (pp. 24-54). Springer, Cham.
- [18] M. Jiague (2012). Mise en œuvre de politique de contrôle d'accès formelles pour des application basées sur une architecture orientée services (Thèse de Doctorat, université de Sherbrooke), pp. 35-36
- [19] Tao Xu. The context-aware middleware in ambient intelligence. Other. Ecole Centrale de Lyon, 2013. English. ⟨NNT : 2013ECDL0047⟩. ⟨tel-01001647⟩
- [20] Schmidt, Douglas C. and Frank Buschmann. "Patterns, frameworks, and middleware: their synergistic relationships." *25th International Conference on Software Engineering, 2003. Proceedings.* (2003): 694-704.
- [21] Alexandros Gazis and Eleftheria Katsiri. 2022. Middleware 101. *Commun. ACM* 65, 9 (September 2022), 38–42.
- [22] Bernstein, Philip A. "Middleware: A Model for Distributed Services." *Communications of the ACM* 39, 2 (February 1996): 86-97.
- [23] Gilberto Echeverria, Séverin Lemaignan, Arnaud Degroote, Simon Lacroix, Michael Karg, et al.. Simulating complex robotic scenarios with MORSE. *3rd International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, Nov 2012, Tsukuba, Japan. pp.197-208, [10.1007/978-3-642-34327-8_20](https://doi.org/10.1007/978-3-642-34327-8_20). [hal-00988475](https://hal.archives-ouvertes.fr/hal-00988475)

- [24]Thinh Le Vinh, Samia Bouzefrane, Jean-Marc Farinone, Amir Attar. Middleware to Integrate Mobile Devices, Sensors and Cloud Computing. *ANT 2015 - 6th International Conference on Ambient Systems, Networks and Technologies*, Jun 2015, Greenwich,London, United Kingdom. [10.1016/j.procs.2015.05.061](https://doi.org/10.1016/j.procs.2015.05.061). [hal-01160989](https://hal.archives-ouvertes.fr/hal-01160989)
- [25]Romain Rouvoy. Une démarche à granularité extrêmement fine pour la construction de canevas intergiciels hautement adaptables : application aux services de transactions. *domain_stic.inge*. Université des Sciences et Technologie de Lille - Lille I, 2006. Français. [tel-00119794](https://tel.archives-ouvertes.fr/tel-00119794)
- [26] Emil - Mircea Andriescu, Roberto Speicys Cardoso, Valérie Issarny. AmbiStream: A Middleware for Multimedia Streaming on Heterogeneous Mobile Devices. *12th International Middleware Conference (MIDDLEWARE)*, Dec 2011, Lisbon, Portugal. pp.249-268, [10.1007/978-3-642-25821-3_13](https://doi.org/10.1007/978-3-642-25821-3_13). [hal-00639633](https://hal.archives-ouvertes.fr/hal-00639633)
- [27] Yves Mahéo. Intergiciels pour applications distribuées sur réseaux dynamiques. Réseaux et télécommunications [cs.NI]. Université de Bretagne Sud; Université Européenne de Bretagne, 2011. [fftel00633253f](https://tel.archives-ouvertes.fr/fftel00633253f)
- [28] <https://www.spiceworks.com/tech/cloud/articles/what-is-middleware/> consulter le 30/07/2022
- [29]Wilfried Jouve, Julien Lancia, Nicolas Palix, Charles Consel, Julia L. Lawall. A Domain-Specific IDL and its Compiler for Pervasive Computing Applications. [Research Report] RR-6213, INRIA. 2007. [inria-00153375v3](https://hal.archives-ouvertes.fr/inria-00153375v3)
- [30] Marco Autili, Mauro Caporuscio, Valérie Issarny. A Reference Model for Service Oriented Middleware. [Research Report] 2008. [inria-00326479](https://hal.archives-ouvertes.fr/inria-00326479)
- [31]Lionel Seinturier, Philippe Merle, Romain Rouvoy, Daniel Romero, Valerio Schiavoni, et al.. A Component-Based Middleware Platform for Reconfigurable Service-Oriented Architectures. *Software: Practice and Experience*, Wiley, 2012, 42 (5), pp.559-583. [10.1002/spe.1077](https://doi.org/10.1002/spe.1077). [inria-00567442](https://hal.archives-ouvertes.fr/inria-00567442)
- [32]Philippe Merle. Intergiciel d'intergiciels adaptable à base de Services, Composants et Aspects. Calcul parallèle, distribué et partagé [cs.DC]. Université Lille 1 Sciences et Technologies, 2015. [tel-01206244](https://tel.archives-ouvertes.fr/tel-01206244)

- [33]Puder, Arno, Kay Römer, and Frank Pilhofer. *Distributed systems architecture: a middleware approach*. Elsevier, 2011.
- [34]Carlos Molina-Jimenez and Santosh Shrivastava. 2009. Model checking correctness properties of a middleware service for contract compliance. In Proceedings of the 4th International Workshop on Middleware for Service Oriented Computing (MWSOC '09). Association for Computing Machinery, New York, NY, USA, 13–18.
- [35]BissyandéTegawendé, Laurent Réveillère, Yérom-David Bromberg. UbiGate: A Gateway to Transform Discovery Information into Presence Information. 4th ACM International Workshop on Services Integration in Pervasive Environments, Jul 2009, London, United Kingdom. pp.NC. [\(hal-00411146\)](#)
- [36]Mehdi Kessiss. Gestion intégrée et multi-échelles des systèmes répartis : Architecture et canevas intergiciel orientés composants. Génie logiciel [cs.SE]. Université de Grenoble, 2010. Français. [\(tel-00742118\)](#)
- [37]Wouter De Borger, Bert Lagaisse, Wouter Joosen. A Generic Solution for Agile Run-Time Inspection Middleware. *12th International Middleware Conference (MIDDLEWARE)*, Dec 2011, Lisbon, Portugal. pp.451-470, [\(10.1007/978-3-642-25821-3_23\)](#). [\(hal-01597758\)](#)
- [38]Moser, V. *Mini Middleware*. No. REP_WORK. 2004.
- [39]Cyril Briand, Olivier Stasse, Michel Taïx, Thierry Germa, Fabien Marco, et al.. ROS : une solution middleware adaptée pour l'industrie du futur ?. 17ème colloque national S-mart *AIP-PRIMECA*, Université Polytechnique Hauts-de-France [UPHF], Mar 2021, LAVAL VIRTUAL WORLD, France. [\(hal-03296139\)](#)
- [40]Gregory Gailliard. Towards a common hardware/software specification and implementation approach for distributed, real time and embedded systems, based on middlewares and object-oriented components. Engineering Sciences [physics]. Université de CergyPontoise, 2010. English. [\(tel-00524737\)](#)
- [41]Webb, P., Syer, D., Long, J., Nicoll, S., Winch, R., Wilkinson, A., ... & Frederick, S. (2020). Spring Boot reference documentation. Retrieved June, 22.
- [42]Maven, A. (2014). Apache maven. URL <http://maven.apache.org/>. Accessed, 11-07.
- [43]Zhang, P. (2017). Practical Guide to Oracle SQL, T-SQL and MySQL. CRC Press.

[44]Fong, F., &Raed, M. (2021). Performance comparison of GraalVM, Oracle JDK andOpenJDK for optimization of test suite execution time.

[45]<https://www.json.org/json-en.html> consulter le 31/08/2022

[46] <https://www.geeksforgeeks.org/mvc-design-pattern/> consulter le 24/08/2022

[47]https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript consulter le 25/08/2022

[48]<https://www.thymeleaf.org/documentation.html> consulter le 03/09/2022

[49] <https://getbootstrap.com/docs/5.0/getting-started/introduction/> consulter le 01/09/2022

[50]<https://www.ibm.com/docs/en/informix-servers/12.10?topic=started-what-is-jdbc> consulter le 01/09/2022