Université Assane Seck de Ziguinchor
UFR Sciences et Technologies
Département Informatique

Mémoire de fin d'études

Pour l'obtention du diplôme de Master

Mention : Informatique

Spécialité : Génie Logiciel

***Thème** : Conception et Mise en place d'une plateforme web d'un environnement de gestion et suivi des stages hospitaliers des étudiants à l'UFR des Sciences de la Santé (UFR 2S) de l'UASZ*

Soutenu le : 25/06/2024

Présenté et soutenu par : M. Saliou TOURE

Sous la direction de : Dr El Hadji Malick NDOYE, Pr Ousmane DIALLO, M. Diéré DIEDHIOU

Sous la supervision de : Pr Ousmane DIALLO

Membres du jury

Pr Ibrahima DIOP	Professeur Assimilé	Président, Examinateur	UASZ
M. Malaw NDIAYE	Assistant	Rapporteur	UASZ
Dr El Hadji Malick NDOYE	Maitre de conférences Assimilé	Encadrant	UASZ
Pr Ousmane DIALLO	Professeur Assimilé	Co-Encadrant	UASZ

Année académique : 2022-2023

Résumé

Ce mémoire de master s'inscrit dans le cadre d'un stage à l'Unité de Formation et de Recherche (UFR) des sciences de la santé (2S) de l'université Assane Seck de Ziguinchor (UASZ). En effet, depuis 2013, année du démarrage des stages des étudiants, la gestion de ceux-ci s'est toujours faite de manière manuelle (format papier) par le service pédagogique. Ce dernier effectue un travail de plus en plus pénible au fil des ans du fait du nombre pléthorique des étudiants. Ce qui entraine des limites telles que le retard des notes des étudiants, la perte des informations comme les fiches de présence dans les hôpitaux, les notes de services. A cela s'ajoute le temps de recherche élevé pour consulter les informations spécifiques d'un étudiant donné.

Pour améliorer cette situation, notre projet vise l'automatisation de la gestion et du suivi de ces stages en développant une plateforme web dédiée. Ainsi, l'objectif principal de ce mémoire est de proposer une plateforme web d'un environnement de gestion et suivi des stages hospitaliers des étudiants à l'UFR 2S qui se base principalement sur trois modules : Un module pour la gestion des comptes utilisateurs, un module pour la gestion des années académiques qui coordonne tous les aspects concernant les années académiques, notamment les semestres, les modules, ainsi que les affiliations et les assignations, et enfin un module de gestion des informations relatives aux établissements hospitaliers.

Pour concrétiser ce projet, nous avons adopté une approche AGILE, plus précisément SCRUM. La phase de spécification des besoins a impliqué l'identification des acteurs, la subdivision des modules en sous-modules chacun, et l'élaboration des fonctionnalités spécifiques à chaque sous-module. Dans la phase d'analyse des besoins et de conception, différents diagrammes sont élaborés pour représenter le système sous divers angles.

Le développement du front-end est réalisé en utilisant l'architecture Model-View- Controller (MVC), tandis que le back-end a suivi le modèle en couches. La notation Unified Modeling Language(UML) et l'outil PowerAMC sont employés pour concevoir les diagrammes, et l'outil Git a facilité la collaboration et la gestion des versions.

L'application est développée avec la technologie MERN (MySQL, Express, React, Node), les tests sont effectués à l'aide de l'outil Postman et le design est réalisé en utilisant un template Bootstrap téléchargé sur internet et personnalisé à notre guise.

Résumé

Mots-clés : stage, Unité de Formation et de Recherche (UFR), sciences de la santé (2S), université Assane Seck de Ziguinchor (UASZ), gestion manuelle, service pédagogique, automatisation, plateforme web, gestion des comptes utilisateurs, gestion des années académiques, établissements hospitaliers, approche AGILE, SCRUM, Model-View-Controller (MVC), back-end, Unified Modeling Language (UML), PowerAMC, Git, MERN, MySQL, Express, React, Node, Postman, Bootstrap.

Abstract

This master's thesis is part of an internship at the Faculty of Health Sciences (UFR 2S) of Assane Seck University of Ziguinchor (UASZ). Since 2013, when student internships began, the management of these internships has always been done manually (in paper format) by the educational service. Over the years, this manual process has become increasingly burdensome due to the large number of students. This has led to issues such as delays in student grades, loss of information such as attendance sheets in hospitals, and service notes. Additionally, there is a high search time required to consult specific information about a given student.

To improve this situation, our project aims to automate the management and monitoring of these internships by developing a dedicated web platform. Therefore, the main objective of this thesis is to propose a web platform for the management and monitoring of student hospital internships at UFR 2S. This platform will primarily be based on three modules: a user account management module, an academic year management module that coordinates all aspects related to academic years, including semesters, modules, affiliations, and assignments, and finally, a hospital information management module.

To bring this project to fruition, we adopted an AGILE approach, specifically SCRUM. The requirements specification phase involved identifying stakeholders, subdividing the modules into sub-modules, and developing specific functionalities for each sub-module. During the requirements analysis and design phase, various diagrams were developed to represent the system from different perspectives.

The front-end development was carried out using the Model-View-Controller (MVC) architecture, while the back-end followed the layered model. Unified Modeling Language (UML) notation and the PowerAMC tool were used to design the diagrams, and the Git tool facilitated collaboration and version control.

The application was developed using the MERN technology stack (MySQL, Express, React, Node), tests were performed using the Postman tool, and the design was created using a Bootstrap template downloaded from the internet and customized to our needs.

Keywords: internship, Faculty of Health Sciences (UFR), Assane Seck University of Ziguinchor (UASZ), manual management, educational service, automation, web platform, user

account management, academic year management, hospital information management, AGILE approach, SCRUM, Model-View-Controller (MVC), back-end, Unified Modeling Language (UML), PowerAMC, Git, MERN, MySQL, Express, React, Node, Postman, Bootstrap.

Hommages et Dédicaces

Par la grâce de DIEU, le Tout-Puissant, source infinie de guidance et de bénédiction, je dédie ce modeste travail à des êtres chers, des étoiles qui continuent de briller dans ma vie

- *À ma grand-mère paternelle, Ndèye Fatou SOW, incarnation vivante de la sagesse et de l'amour inconditionnel. Sa présence, toujours vibrante, est une source constante d'inspiration. Puissent ses jours être doux et emplis de bénédictions.*
- *À ma défunte sœur, Absa GUEYE, une étoile partie trop tôt. Son souvenir illumine mes jours, et son impact sur ma vie résonne toujours. En dédiant ce mémoire à sa mémoire, je rends hommage à la beauté de son âme. Paix à son âme.*

Remerciements

En signe de profonde reconnaissance, je souhaite exprimer mes plus sincères remerciements à toutes les personnes dont la contribution, directe ou indirecte, a été déterminante pour le succès de ce mémoire et la réalisation de ce modeste travail. Je tiens à adresser mes remerciements particuliers à :

Encadrement :

- *Dr El Hadji Malick NDOYE, éminent docteur à l'Université Assane SECK de Ziguinchor, pour son encadrement avisé et ses conseils éclairés qui ont grandement enrichi ce travail.*
- *Pr Ousmane DIALLO, éminent professeur à l'Université Assane SECK de Ziguinchor, en tant que co-encadrant, pour sa précieuse contribution à l'élaboration de ce mémoire.*
- *Pr. Ibrahima DIOP, Président du jury, pour son expertise académique et ses critiques constructives qui ont permis d'améliorer la qualité de ce mémoire. Son engagement rigoureux et son dévouement à l'excellence ont été une source d'inspiration constante.*
- *M. Malaw NDIAYE, Assistant, pour son soutien logistique et académique tout au long de ce projet. Sa disponibilité et son assistance précieuse ont grandement facilité le bon déroulement de ce travail.*
- *M. Diéré DIEDHIOU, mon maître de stage, dont la contribution à mon parcours professionnel a été inestimable, sa disponibilité exemplaire a été une source d'inspiration constante. Toujours prêt à partager son expertise et à répondre à nos questions, il a su créer un espace où le dialogue et l'apprentissage étaient encouragés. Son humilité remarquable a rendu notre collaboration des plus agréable. Malgré son expérience et ses compétences, il a su instaurer un climat de confiance où chacun se sentait libre de s'exprimer et d'apprendre.*
- *Au personnel de mon lieu de stage, notamment le service de la pédagogie de l'UFR 2S, pour leur accueil chaleureux, leur collaboration et leur soutien tout au long de cette période. Leur assistance et leur dévouement ont grandement facilité la réalisation de ce mémoire.*

Corps professoral :

- *L'ensemble des enseignants de l'Université Assane SECK de Ziguinchor, avec une mention spéciale pour ceux du Département Informatique, qui ont grandement contribué à la qualité exceptionnelle de ma formation.*

Famille :

- *Ma mère (Fatou GUEYE THIAM), une figure exemplaire : Sa bienveillance infinie et son soutien indéfectible ont été des piliers constants dans notre vie. Son rôle incommensurable va bien au-delà de l'éducation ; elle est le symbole même de la force et de la dévotion maternelle. Ma mère est l'étoile qui a guidé chacun de mes pas.*
- *Mon défunt père (Singbe KOUROUMA) : Son sacrifice en faveur de ses enfants a été une véritable aubaine. Sa mémoire continue de nous inspirer, et son héritage de valeurs inestimables demeure vivant dans nos cœurs. Chaque accomplissement que je réalise porte la marque de son amour et de ses enseignements.*
- *Mes frères et sœurs : Mame Astou KOUROUMA, Mouhamed KOUROUMA et Sadibou TOURE, Pape Ndiassé THIAM.*

Famille tutrice à Ziguinchor :

Je tiens également à exprimer ma reconnaissance envers ma famille tutrice qui a joué un rôle essentiel tout au long de mon parcours universitaire.

- *Salimata DIEDHIOU (Ta Mignone) : Sa facilité à me faire sentir chez moi à Ziguinchor a été un véritable bienfait. Sa bienveillance et son soutien ont grandement contribué à mon adaptation dans un nouvel environnement.*
- *Fatou BADJI (Igna) et Yaya DIEDHIOU (Baba) : Les parents de Ta Mignone m'ont accueilli avec une générosité sans égale. Leur traitement, empreint de chaleur et d'affection, m'a souvent donné le sentiment d'être leur propre fils. Leur dévouement exceptionnel a créé un cadre familial où je me suis senti privilégié et choyé, parfois même au détriment de leurs propres enfants.*
- *Je tiens également à rendre hommage au reste de la famille tutrice, dont la présence bienveillante a été une source constante de réconfort et d'encouragement. Chacun d'eux a joué un rôle significatif dans mon parcours, contribuant à faire de cette expérience une période inoubliable.*

Proches et Amis :

- *Mes camarades de classe, amis (Aissatou SEYDI, Abdou Lahad NDIAYE, Malick FAYE, Ousseynou NDOYE, Moustapha FALL), à mon mentor (M. Cheikh Tidiane GUEYE) et*

tout mon entourage, dont la contribution physique et morale a été cruciale pour la réussite de ce travail.

Table des matières

Liste des figures

Liste des tableaux

Liste des Abréviations

2S : Sciences de la Santé

API : Application Programming Interface

CD : Continuous Deployment/Delivery

CI : Continuous Integration

GUI : Graphical User Interface

IaC : Infrastructure en tant que Code

IDE : Integrated Development Environment

JWT : JSON Web Tokens

MERN : MySQL Express React Node

MVC : Model-View-Controller

PCEM : Premier Cycle des Etudes Médicales

RAD : Rapid Application Development

SGBD : Système de Gestion de Base de Données

UASZ : Université Assane Seck de Ziguichor

UFR : Unité de Formation et de Recherche

UML : Unified Modeling Language

URL : Uniform Resource Locator

VCS : Versionning Control System

VSCode : Visual Studio Code

Introduction Générale

L'Unité de Formation et de Recherche des sciences de la santé (UFR 2S) de l'Université Assane Seck de Ziguinchor (UASZ) est un acteur clé dans la formation des professionnels de la santé. Depuis 2013, les étudiants de cette unité effectuent des stages hospitaliers essentiels pour leur formation pratique. Cependant, la gestion de ces stages est toujours réalisée de manière manuelle, à l'aide de supports papier. Avec l'augmentation constante du nombre d'étudiants, cette méthode est devenue inefficace, entraînant divers problèmes tels que des retards dans la transmission des notes, la perte d'informations importantes comme les fiches de présence dans les hôpitaux, et des difficultés accrues pour retrouver des informations spécifiques sur un étudiant donné.

Face à ces défis, notre projet vise à automatiser la gestion et le suivi des stages hospitaliers des étudiants de l'UFR 2S en développant une plateforme web dédiée. Cette solution vise à améliorer l'efficacité et la fiabilité du processus de gestion des stages, en remplaçant les méthodes manuelles par un système numérique centralisé et sécurisé.

Au cours de ce stage de fin d'études, notre mission capitale réside dans la conception et la mise en œuvre des modules déterminants pour le bon fonctionnement du système. Ces modules fondamentaux sont les suivants :

- Gestion des Utilisateurs :
  - ❖ Objectif : Administrer et superviser les comptes des utilisateurs du système.
  - ❖ Portée : Permettre une gestion efficace des accès et des autorisations, assurant une sécurité optimale.
- Gestion des Années Académiques :
  - ❖ Objectif : Piloter l'ensemble des éléments liés aux années académiques, aux semestres, aux modules, ainsi qu'aux affiliations et aux assignations.
  - ❖ Portée : Offrir une gestion complète et structurée des différents éléments temporels et organisationnels du cursus académique.
- Gestion des Hôpitaux :
  - ❖ Objectif : Administrer les hôpitaux, les spécialités et les services associés.
  - ❖ Portée : Faciliter la gestion des partenariats hospitaliers, des spécialités médicales, et des services disponibles.

Ce mémoire est structuré autour de cinq chapitres. Dans un premier temps, le chapitre I présente le contexte général du projet ainsi que la problématique à laquelle il vise à répondre. Ensuite, le chapitre II détaille les processus de conception et les choix méthodologiques adoptés pour la réalisation du projet. Le chapitre III se penche sur la spécification des besoins et l'identification des acteurs clés, tandis que le chapitre IV analyse les besoins de manière approfondie et présente les résultats des phases de conception. Enfin, le chapitre V expose les outils d'implémentation utilisés et offre un aperçu des interfaces de l'application développée.

Chapitre I : Contexte justificatif du Sujet

Introduction

Ce chapitre présente la structure d'accueil, qui est l'UFR des sciences de la santé (2S) de l'Université Assane Seck de Ziguinchor (UASZ), où nous avons réalisé notre stage au sein du Service Pédagogique. Ensuite, il fournit une description générale du projet de gestion des stages en exposant la problématique traitée et les objectifs visés au cours de cette période de stage.

I.1. L'UFR des Sciences de la Santé

Pédagogiquement ouverte en **Février 2007** l'université de Ziguinchor est depuis mars 2014 baptisée **Université Assane SECK de Ziguinchor**. C'est par le décret numéro **2008-537 du 22 mai 2008** portant création et organisation que l'université est reconnue comme institution publique[1].

Avec deux UFR (l'UFR Sciences et Technologies et l'UFR Sciences Economiques et Sociales), l'Université a donc démarré avec 257 étudiants, 20 enseignants-chercheurs et 30 personnels administratifs, techniques et de service.

Elle s'est enrichie au courant de l'année académique 2011-2012 d'une UFR supplémentaire l'UFR des sciences de la santé qui, depuis son ouverture, s'est engagée dans le système LMD et a démarré ses activités avec 43 étudiants. Elle est aujourd'hui dirigée par une directrice.

I.1.1. Mission de l'UFR

Elle est un des établissements d'enseignement et de recherche de l'Université de Ziguinchor. Sa mission essentielle est de former des ressources humaines de haut niveau pour le Sénégal. L'ouverture de l'UFR des Sciences de la Santé, la dernière-née de l'institution, s'explique par la nécessité de satisfaire les besoins de couverture sanitaire à cause de l'éloignement et de l'enclavement de la Région Naturelle de La Casamance. En effet, elle permet de relever le plateau médical de la région dont les structures sanitaires sont de plus en plus sollicitées par des patients venus de la sous-région. Cette UFR procure aussi à l'institution des spécialistes qui pourront en plus des enseignements, intervenir dans les hôpitaux de la région afin de prendre en charge un certain nombre de pathologies[2].

I.1.2. Conditions d'accès à l'UFR

Pour être admis en première année du Premier Cycle des Etudes Médicales (PCEM), les candidats doivent être titulaires du Baccalauréat scientifique ou tout diplôme admis en équivalence, avoir une moyenne de 14/20 dans les matières Fondamentales suivantes (Physique-Chimie, Sciences Naturelles, Mathématiques) et être âgés de 19 ans au maximum. L'admission aux Masters ne concerne que les étudiants régulièrement inscrits au Deuxième Cycle des Etudes Médicales. Les candidats au diplôme d'Etat de Docteur en Médecine prennent une inscription au début de chaque quadrimestre (inscription quadrimestrielle). En 1ère Année (PCME I), seule une inscription à un quadrimestre par session est autorisée. Par contre en 2ème Année (PCMEII), une inscription à deux (02) quadrimestres par session est autorisée.

I.1.3. Organisation

L'organisation de l'UFR 2S repose sur une structure hiérarchique comprenant plusieurs services, chacun étant dirigé par un chef de division placé sous l'autorité d'un directeur.

À sa tête se trouve le directeur, responsable de l'ensemble des activités académiques, administratives et stratégiques de l'unité.

D'un côté, nous trouvons le chef de service administratif, supervisant le service financier et les services généraux. De l'autre côté, nous avons le directeur adjoint, responsable des départements de médecine, de chirurgie, de biologie et d'explorations fonctionnelles, ainsi que des sciences paramédicales. Entre ces deux entités se trouve le responsable pédagogique, sous la direction du chef de service administratif et du directeur adjoint.

Pour faciliter le travail, le directeur est secondé par un assistant, le responsable pédagogique est accompagné d'un secrétaire de service pédagogique, et présentement un seul secrétaire est désigné pour tous les départements, assurant ainsi une coordination administrative efficace.

La figure ci-dessous (***figure 1***) est une représentation de l'organigramme de l'UFR des sciences de la santé.

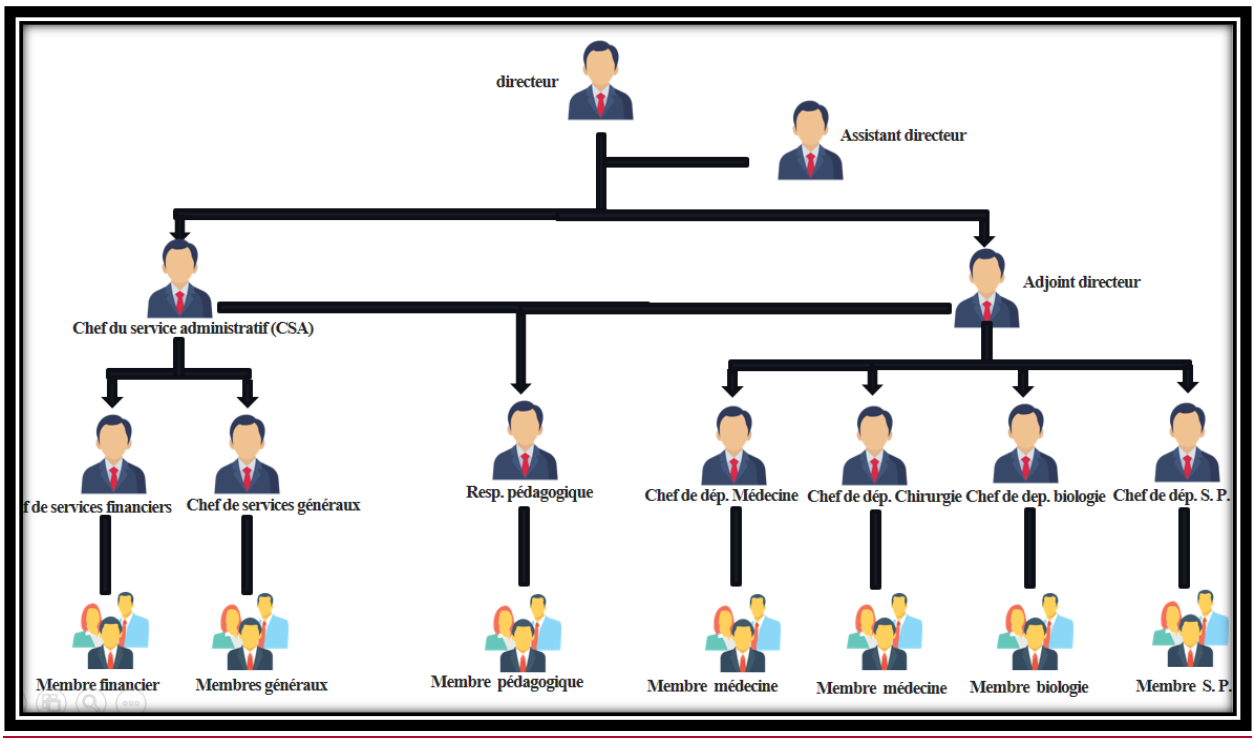

Figure 1 : Organigramme de l'UFR des sciences de la santé

Comme l'atteste la figure 1 ci-dessus l'organigramme est composé de plusieurs services dont chacun a une mission explicite.

I.1.3.1. Le directeur

Le directeur, en tant que chef de file de l'UFR 2S, est responsable de la supervision globale des activités académiques, administratives et stratégiques. Il joue un rôle crucial dans l'élaboration et la mise en œuvre des politiques de l'UFR, ainsi que dans la coordination des programmes d'enseignement et de recherche. Le directeur est chargé de veiller à la réalisation des objectifs de l'unité, en s'assurant que toutes les opérations sont conformes aux normes académiques et aux réglementations institutionnelles.

De plus, le directeur supervise le développement stratégique de l'UFR, en s'engageant dans des initiatives visant à améliorer la qualité de l'enseignement et de la recherche. Il est également responsable de la gestion des relations avec les partenaires externes, les organismes de financement et les autres parties prenantes, afin de promouvoir les intérêts de l'UFR.

I.1.3.2. Le chef de service administratif

Le chef de service administratif, en tant que pivot central de l'administration, est chargé de superviser les aspects financiers, administratifs et généraux de l'UFR 2S. Il joue un rôle crucial

dans la gestion des ressources financières, y compris la planification budgétaire, le suivi des dépenses et la gestion des finances. De plus, il est responsable de la coordination des activités administratives de l'UFR, y compris la gestion des ressources humaines, la gestion des infrastructures matérielles et la maintenance des équipements. En outre, le chef de service administratif assure la mise en œuvre des politiques et des procédures administratives de l'UFR, en veillant à ce qu'elles soient conformes aux normes et aux réglementations en vigueur. Son rôle est essentiel pour assurer le bon fonctionnement global de l'UFR et pour soutenir efficacement les activités académiques et de recherche.

I.1.3.3. L'adjoint au directeur

L'adjoint du directeur, en sa qualité de relais entre le directeur et les différents départements, supervise également les activités des départements de médecine, de chirurgie, de sciences paramédicales, de biologie et d'explorations fonctionnelles. Il travaille en étroite collaboration avec les chefs de département de ces unités pour assurer la coordination efficace des programmes académiques, la gestion des ressources humaines et matérielles, ainsi que le suivi des projets de recherche et de développement. En outre, il veille à ce que les objectifs stratégiques de chaque département soient alignés sur la vision et les priorités de l'UFR dans son ensemble, contribuant ainsi à la cohérence et à l'harmonisation des activités universitaires.

I.1.3.4. Le responsable pédagogique

Le responsable pédagogique, en tant que figure centrale de l'UFR, assume la responsabilité de superviser toutes les activités académiques et pédagogiques au sein de l'unité, sous les ordres du chef de service administratif et du directeur adjoint. Son rôle principal consiste à garantir la qualité et la pertinence des programmes d'enseignement et de formation dispensés par l'UFR. À cette fin, il collabore étroitement avec les chefs de département et les enseignants pour élaborer les plans de cours, concevoir les programmes d'études et assurer la mise en œuvre efficace des méthodes pédagogiques. De plus, le responsable pédagogique est chargé d'évaluer et de suivre les progrès des étudiants, de coordonner les examens et les évaluations, ainsi que de promouvoir un environnement d'apprentissage inclusif et stimulant. En outre, il représente l'UFR dans les instances universitaires et professionnelles pertinentes, contribuant ainsi à renforcer la réputation et le rayonnement de l'unité.

I.2. Description du Sujet

Cette section explore le contexte et la problématique associés à la gestion des stages hospitaliers à l'UFR des sciences de la santé de l'Université Assane Seck de Ziguinchor, ainsi que les objectifs visés.

I.2.1. Contexte et Problématique

La gestion des stages hospitaliers à l'UFR des sciences de la santé de l'Université Assane Seck de Ziguinchor a comme principal objectif de faciliter le travail du personnel impliqué à cette gestion à savoir le responsable pédagogique, les chefs de département, les maitres de stage au niveau des hôpitaux et les secrétaires mais aussi de sécuriser les données des étudiants.

En effet depuis 2013, année de l'inauguration des stages des étudiants, la gestion de ces derniers a toujours été manuelle, s'effectuant sur support papier par le service pédagogique. Au fil des ans, cette méthode s'est avérée de plus en plus laborieuse en raison de l'augmentation constante du nombre d'étudiants, imposant ainsi un fardeau croissant au service pédagogique.

Ainsi, l'UFR a toujours fonctionné en utilisant le format classique (papier) pour les stages. En conséquence, l'UFR, sous cette forme, présente de nombreuses limites qui peuvent entraver sa mission.

En voici quelques-unes :

- Accès limité : Les dossiers de stage des étudiants peuvent être conservés dans des lieux physiques éloignés, rendant leur accès difficile. Cela peut compliquer le partage rapide d'informations entre les membres du personnel impliqués.
- Perte de Données : Les dossiers de stage au format papier sont vulnérables à la perte, aux dommages ou au vol. En cas de catastrophe naturelle ou d'incident, les informations peuvent être irrémédiablement perdues. Par exemple lors des événements pré-électoraux de juin 2023, des manifestants ont vandalisé et incendié certaines parties de l'UFR, ce qui a entraîné d'importantes pertes d'informations.
- Temps de recherche élevé : La recherche d'informations spécifiques dans les dossiers de stage papier peut être longue et fastidieuse, entraînant une perte de temps pour le personnel de l'UFR.

- Espace de stockage physique : les dossiers de stage au format papier prennent beaucoup de place physique, ce qui peut poser des problèmes d'espace dans les établissements de l'UFR. De plus, stocker de grandes quantités de dossiers peut être coûteux et encombrants.
- Difficulté de Mise à Jour : les dossiers de stage papier peuvent être difficiles à mettre à jour. Les entrées manuelles peuvent causer des erreurs ou des omissions lorsqu'il s'agit d'ajouter ou de modifier.
- Confidentialité : il existe toujours un risque de violation de la confidentialité des dossiers de stage au format papier, malgré le fait que ces dossiers sont généralement stockés dans des endroits sécurisés. Les personnes non autorisées peuvent y avoir accès.
- Difficulté dans l'évaluation : Les dossiers de stage papier rendent difficile la réalisation de bilans semestriels ou annuels, ainsi que l'évaluation du nombre d'étudiants ayant validé leur stage pour un niveau donné ou globalement.

Fort de ce constat, nous avons jugé nécessaire qu'une structure d'une telle envergure a besoin de s'acquérir de moyens plus performants et rapides améliorant pleinement sa gestion dans sa globalité. C'est pourquoi nous avons décidé de concevoir et de mettre en place un site web pour la gestion des stages hospitaliers à l'UFR des sciences de la santé.

I.2.2. Objectifs

Cette section établit les objectifs généraux et spécifiques du projet, visant à fournir une vision claire et structurée de ce que le système de gestion des stages hospitaliers de l'UFR des sciences de la santé doit accomplir.

I.2.2.1. Objectif Général

L'objectif général de ce stage est de mettre en place une plateforme web d'un environnement de gestion et suivi des stages hospitaliers des étudiants à l'UFR 2S. Ce système devrait permettre une gestion efficace et transparente des utilisateurs, des années académiques, ainsi que des hôpitaux impliqués dans les stages des étudiants.

I.2.2.2. Objectifs spécifiques

Les objectifs spécifiques de ce système sont les suivants :

- Permettre la création, la visualisation, la modification et la suspension des comptes utilisateurs, garantissant ainsi un contrôle total sur les autorisations d'accès.
- Faciliter l'enregistrement, l'affichage et la modification des années académiques, fournissant ainsi une base solide pour le démarrage et la gestion des sessions d'apprentissage.
- Assurer la création, la visualisation, la modification et la suspension des hôpitaux, offrant la flexibilité nécessaire pour gérer les partenariats et les collaborations.
- Permettre la gestion des spécialités médicales et chirurgicales dans chaque hôpital, avec la possibilité de créer, visualiser, modifier et suspendre ces spécialités selon les besoins.
- Faciliter la gestion des services hospitaliers, permettant ainsi une répartition efficace des étudiants dans les différents services disponibles.
- Offrir une flexibilité dans la gestion des semestres académiques, avec la possibilité de créer, visualiser et modifier les semestres en cas de besoin.
- Assurer la création, la visualisation et la modification des modules de stage, permettant ainsi une organisation efficace des activités pédagogiques.
- Permettre l'affiliation des secrétaires aux hôpitaux respectifs, avec la possibilité de modifier ces affiliations au besoin.

Faciliter l'assignation des modules aux enseignants responsables, garantissant ainsi une répartition équitable des tâches pédagogiques.

Conclusion

Ce chapitre a présenté notre structure d'accueil qui est l'UFR des sciences de la santé, le contexte du projet de gestion des stages dans lequel nous avons travaillé. Il a aussi exposé la problématique et les objectifs de ce travail.

Pour atteindre ces objectifs, il est essentiel d'adopter une méthodologie de développement appropriée. Dans le prochain chapitre, les processus de développement sont revisités, puis le processus choisi pour ce mémoire est décrit en détail.

Chapitre II : Processus de Développement

Introduction

L'orchestration du développement d'une application est une symphonie structurée d'étapes cruciales, formant le socle de la conception, de la création, des tests et du déploiement des applications logicielles. Chacune de ces étapes, complexe en soi, réclame des compétences techniques pointues. Le choix de la méthodologie de développement, qu'elle soit Agile, DevOps, en Cascade, ou autre, devient ainsi impératif, adapté aux besoins spécifiques du projet et à la dynamique de l'équipe de développement.

Ces méthodologies, bien que distinctes dans leurs approches respectives, convergent toutes vers un objectif commun : atténuer les risques et maîtriser les coûts inhérents à la création logicielle.

Notre exploration de ces méthodologies débutera par une incursion concise, avant de dévoiler les critères subtils ayant motivé notre choix méthodologique pour le développement novateur de notre site web.

I. Méthode en cascade

I.1. Qu'est-ce que le modèle en cascade ?

Le modèle en cascade (en anglais : *waterfall model*) est un modèle de gestion linéaire qui divise les processus de développement en phases de projet successives. Contrairement aux modèles itératifs, chaque phase est effectuée une seule fois. Les sorties de chaque phase antérieure sont intégrées comme entrées de la phase suivante. Le modèle en cascade est principalement utilisé dans le développement de logiciels[3].

I.2. Comment fonctionne le modèle en cascade ?

On doit le développement du modèle en cascade classique à l'informaticien Winston Walker Royce. Royce n'en est toutefois pas l'inventeur. En effet, son essai publié en 1970 sous le titre « **Managing the Development of Large Software Systems** » présente plutôt une **analyse critique des modèles linéaires**. Royce proposait comme alternative un modèle itératif et

incrémental dans lequel chaque phase reposerait sur la précédente et en vérifierait les résultats[3].

Il recommandait un modèle en sept phases qui se déroulaient en plusieurs étapes (itérations) : les exigences système, les exigences logicielles, l'analyse, la conception, l'implémentation, le test et l'exploitation

Le modèle de gestion que l'on appelle modèle en cascade est fondé sur les phases définies par Royce, **mais prévoit une seule itération**.

Dans son essai, Royce n'évoque pas une seule fois le terme cascade (*waterfall*).

En pratique, plusieurs versions du modèle en cascade sont utilisées. Les modèles les plus courants divisent les processus de développement en cinq phases. Les phases 1, 2 et 3 définies par Royce sont parfois regroupées en une seule et même phase, qualifiée d'analyse des besoins.

- **Analyse :** planification, analyse et spécification des besoins
- **Conception :** conception et spécification du système
- **Implémentation :** programmation et tests des modules
- **Test :** intégration du système, tests du système et de l'intégration
- **Exploitation :** livraison, maintenance, amélioration

I.3. Les phases du modèle en cascade

Dans le modèle en cascade, les différentes phases d'un processus de développement s'enchaînent. Chaque phase se termine par un résultat intermédiaire (étape)[3]**.**

- **Analyse** : chaque projet logiciel commence par une phase d'analyse comprenant une étude de faisabilité et une définition des besoins. Les coûts, le rendement et la faisabilité du projet logiciel sont estimés lors de l'**étude de faisabilité**.
- **Conception :** la phase de conception sert à l'élaboration d'un concept de résolution concret sur la base des besoins, des tâches et des stratégies déterminées au préalable. Au cours de cette phase, les développeurs élaborent l'**architecture logicielle** ainsi qu'un **plan de construction détaillé du logiciel**.
- **Implémentation :** l'architecture logicielle élaborée pendant la phase de conception est réalisée lors de la **phase d'implémentation** qui comprend la **programmation du logiciel**, la **recherche d'erreurs** et les **tests de modules**.

- **Test :** la phase de test comprend l'intégration du logiciel dans l'environnement cible souhaité. En règle générale, les produits logiciels sont tout d'abord livrés à une sélection d'utilisateurs finaux sous la forme d'une **version bêta** (bêta-tests). Il est alors déterminé si le logiciel répond aux besoins préalablement définis à l'aide des **tests d'acceptation** développés lors de la phase d'analyse. Un produit logiciel ayant passé avec succès les bêta-tests est prêt pour la mise à disposition.
- **Exploitation :** la dernière phase du modèle en cascade qui inclut la **livraison** la **maintenance** et l'**amélioration du logiciel**.

La figure ci-après ***(figure 2)*** représente les phases de la méthode en cascade

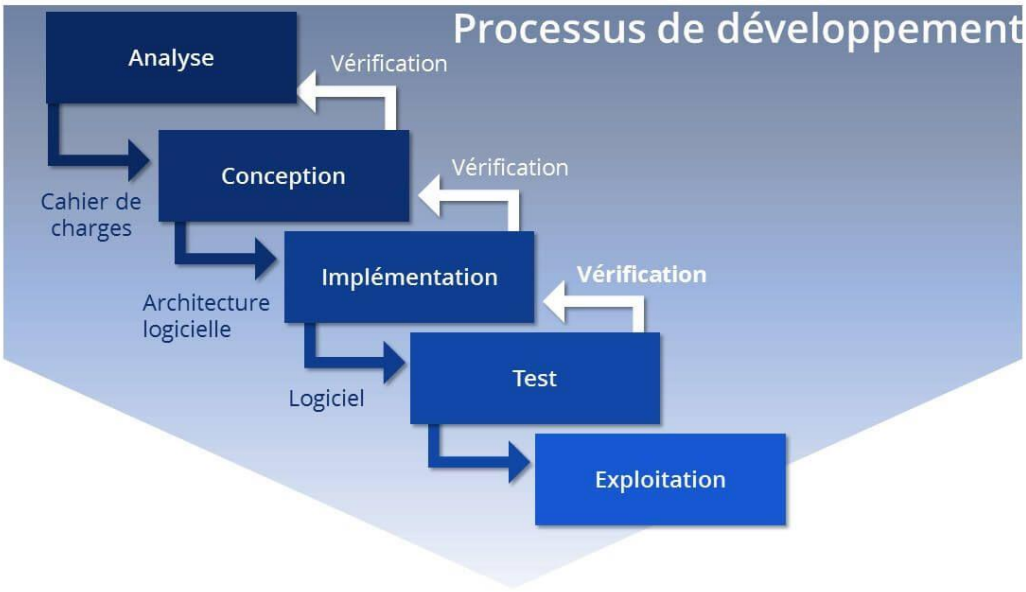

Figure 2 : Modèle en cascade*[3]*

I.4. Evaluation du modèle en cascade

Le modèle en cascade offre une structure hiérarchique claire pour les projets de développement dans laquelle les différentes phases du projet sont clairement délimitées. Étant donné que chaque phase se termine par une étape, le processus de développement peut être suivi facilement. Le point fort du modèle se trouve dans la documentation des étapes du processus. Les connaissances acquises sont consignées dans les documents d'exigences et de conception[3].

En théorie, le modèle en cascade doit créer les conditions pour une réalisation rapide et peu coûteuse des projets par une planification minutieusement élaborée au préalable. Toutefois, les avantages du modèle en cascade sont sujets à controverse dans la pratique. D'une part, les phases du projet sont rarement délimitées clairement dans le développement logiciel. Pour les projets logiciels complexes notamment, les développeurs sont souvent confrontés au fait que les différents composants d'une application se trouvent dans différentes phases de développement au même moment. D'autre part, le déroulement linéaire du modèle en cascade ne correspond souvent pas aux conditions réelles.

Le modèle en cascade ne prévoit pas à proprement parler des adaptations en cours de projet. Un projet de logiciel, dans lequel l'ensemble des détails du développement sont déjà définis au début du projet, peut toutefois uniquement aboutir lorsque beaucoup de temps et d'argent ont été investis dès le départ dans l'analyse et la conception. S'y ajoute le fait que les projets logiciels plus vastes s'étendent parfois sur plusieurs années et sans un ajustement régulier aux développements actuels, ils fourniraient des résultats déjà obsolètes lors de leur introduction.

I.5. Vue d'ensemble des avantages et inconvénients du modèle en cascade

Le tableau ci-dessous (***tableau 1***) montre une vue d'ensemble des avantages et des inconvénients du modèle en cascade

Avantages	**Inconvénients**
Une structure simple grâce à des phases de projet clairement identifiées	Les projets complexes ou à plusieurs niveaux ne peuvent que rarement être divisés en phases de projet clairement définies
Une bonne documentation du processus de développement par des étapes clairement définies	Une faible marge pour les ajustements du déroulement du projet en raison d'exigences modifiées

Les coûts estimés et la charge de travail peuvent être dès le début du projet	L'utilisateur final est uniquement intégré dans le processus de production après la programmation
Les projets structurés d'après le modèle en cascade peuvent être représentés facilement sur un axe temporel	Les erreurs sont parfois détectées uniquement à la fin du processus de développement

Tableau 1 : Avantages et inconvénients du modèle en cascade*[3]*

Le modèle en cascade est principalement utilisé dans les projets pour lesquels les besoins et les processus peuvent être définis de façon précise dès la phase de planification et pour lesquels on peut supposer que les hypothèses changeront peu tout au long du déroulement du projet.

Passons maintenant à l'approche Agile, une méthodologie de développement qui a gagné en popularité ces dernières années.

II. Méthodologie ou Approche Agile

II.1. Définition

La méthode agile est une méthode de gestion de projet**.** L'idée, lorsque l'on utilise cette approche, est d'apporter souplesse et performance à la gestion de projet. Centrée sur l'humain et la communication, elle permet aux clients de participer au développement d'un produit tout au long de l'avancement du projet[4].

II.2. Principes de base

La méthode Agile se base sur un cycle de développement qui porte le client au centre. Le client est impliqué dans la réalisation du début à la fin du projet. Grâce à la méthode agile le demandeur obtient une meilleure visibilité de la gestion des travaux qu'avec une méthode classique.

L'implication du client dans le processus permet à l'équipe d'obtenir un feedback régulier afin d'appliquer directement les changements nécessaires.

Cette méthode vise à accélérer le développement d'un logiciel. De plus, elle assure la réalisation d'un logiciel fonctionnel tout au long de la durée de sa création.

Le principe de base consiste à proposer une version minimale du logiciel puis à intégrer des fonctionnalités supplémentaires à cette base, par processus itératif. Le processus itératif regroupe une séquence d'instructions à répéter autant de fois que possible, selon le besoin[5].

II.3. Les principales méthodes agiles

En effet, lorsque l'on emploie le terme « méthode agile » au singulier on parle d'un concept. Cependant il existe plusieurs méthodes agiles qui se différencient les unes des autres. En voici quelques-unes [5] :

II.3.1. Scrum

Aujourd'hui « Scrum » est la méthode agile la plus populaire. Ce terme signifie « mêlée » au rugby. La méthode scrum s'appuie sur des « sprints » qui sont des espaces temps assez courts pouvant aller de quelques heures jusqu'à un mois. Généralement et de préférence un sprint s'étend sur deux semaines. À la fin de chaque sprint, l'équipe présente ce qu'elle a ajouté au produit. Scrum regroupe trois acteurs :

- Le Product Owner (ou « Directeur de produit ») : il communique les objectifs premiers des clients et utilisateurs finaux, coordonne l'implication des utilisateurs et des parties prenantes, et se coordonne lui-même avec les autres product owners pour assurer une cohérence.
- Le Scrum Master : membre de l'équipe, il a pour but d'optimiser la capacité de production de l'équipe. Pour se faire, le scrum master aide l'équipe à travailler de façon autonome tout en s'améliorant davantage.
- L'équipe opérationnelle (qui regroupe idéalement moins de dix personnes) : la particularité d'une équipe scrum est qu'elle est dépourvue de toute hiérarchie interne. Une équipe scrum est auto-organisée.

D'autres termes sont à connaître pour comprendre la méthode scrum:

- Le product backlog (carnet du produit) : ce document contient les exigences initiales dressées puis hiérarchisées avec le client en début de projet. Néanmoins il va évoluer tout au long de la durée du projet, en fonction des divers besoins du client.

- Le sprint backlog (carnet de sprint) : en chaque début de sprint, l'équipe définit un but. Puis lors de la réunion de sprint, l'équipe de développement choisit les éléments du carnet à réaliser. L'ensemble de ces éléments constitue alors le sprint backlog.
- User story : ce terme désigne les fonctionnalités décrites par le client.
- La mêlée (scrum) : c'est une réunion d'avancement organisée de manière quotidienne durant le sprint.

La figure ***(figure 3)*** ci-dessous illustre le schéma du processus de Scrum

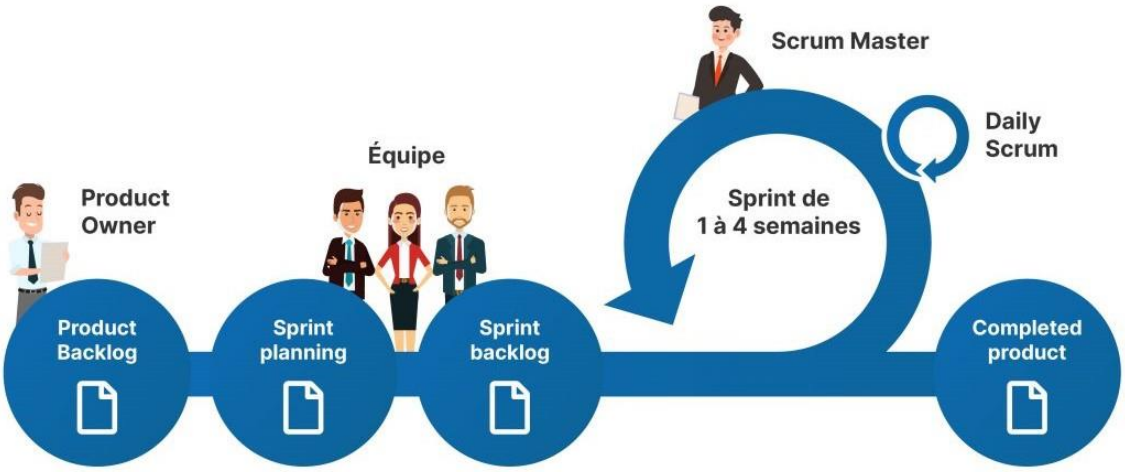

Figure 3 : Méthode Scrum Agile*[6]*

II.3.2. EXtreme Programming (XP)

Cette méthode très réactive destinée à des petits ou moyens projets, permet de réduire les coûts du changement. Dans cette méthode, le client pilote le projet grâce à des cycles itératifs d'une à deux semaines. C'est lui qui choisit les fonctionnalités à implémenter. Il transmet ses exigences à l'équipe sous forme de scénario susceptible d'être implémenté en une itération.

Dans la méthode XP l'équipe se décompose en binômes qui revoient régulièrement les codes. On parle de responsabilité collective du code, le but étant que chaque développeur soit capable d'intervenir n'importe où dans la structure interne du logiciel. Auquel s'ajoutent des tests automatisés mis en place afin de vérifier chacune des fonctionnalités demandées par le client et de garantir une bonne qualité du produit[5].

II.3.3. Rapid Application Development (RAD)

Cette méthode agile est la plus ancienne de toutes les autres méthodes agiles. Elle se base sur un cycle de développement allant de 90 à 120 jours, incluant cinq phases, à savoir [5]:

- L'initialisation (préparation de l'organisation et plan de communication)
- Le cadrage (analyse et expression des exigences)
- Le design (conception et modélisation)
- La construction (réalisation et prototypage qui représentent environ 50% du projet)
- Le contrôle final de qualité par site pilote

On peut également citer d'autres méthodes agiles comme Rational Unified Process (RUP), Feature Driven Development (FDD) ou Dynamic systems development method (DSDM).

II.4. Avantages et inconvénients de la méthodologie

II.4.1. Avantages

Ce processus compte de nombreux avantages pour une équipe. Il apporte beaucoup de valeurs en matière de gestion de projet et de transformation organisationnelle d'entreprise. Voici les principaux bénéfices qui ressortent d'une bonne application[7]:

- Un gain en termes de contrôle sur le produit final à livrer. La méthode de travail est particulièrement efficace. Elle permet de modifier et d'adapter le plan d'action à tout moment. Il est plus aisé de rebondir et de trouver des solutions adaptées, notamment grâce au Product Owner et au Scrum Master.
- Une efficacité décuplée avec des équipes qui s'organisent entre elles et qui sont indépendantes. L'ambiance se veut collaborative et les équipes sont totalement responsabilisées. Leur engagement est fort, tout comme leurs performances globales. Tout le monde avance à l'unisson avec un objectif commun dans des bureaux entièrement équipés.
- Des commandes de haute qualité grâce aux nombreux tests des différentes fonctionnalités effectuées lors du développement. Les modifications utiles et nécessaires sont alors effectuées pour avoir des produits toujours parfaits.
- Une plus grande satisfaction des utilisateurs grâce à cette collaboration solide entre les équipes. Les échanges sont plus simples et les désirs des clients, mieux suivis. Un bon

retour sur investissement généré, avec une philosophie qui permet de créer un produit rapidement commercialisable grâce à des outils adaptés. Cette technique permet de réduire les risques et donc les coûts.

II.4.2. Inconvénients

Malgré ses nombreux avantages et toutes les valeurs ajoutées apportées, la technique compte également quelques limites. On retrouve notamment dans ses points faibles[7] :

- La mauvaise compréhension de l'agilité et de la signification du terme. Pour une bonne application de cette technique, il faut y investir du temps en entreprise. Elle doit être comprise mais aussi bien apprivoisée pour des résultats notables. Ces changements organisationnels demandent de l'investissement afin d'éviter l'échec.
- La réduction de la documentation dans le développement. Il est important de travailler suffisamment la documentation du produit afin de réaliser une bonne passation avec les équipes. Les nouveaux membres, notamment, ont besoin de renseignements et de détails sur ses fonctionnalités afin de pouvoir l'appliquer. Cela peut engendrer certaines difficultés mais aussi une adaptation plus longue et fastidieuse pour les nouveaux arrivants en ce qui concerne l'agilité.
- La difficulté à adopter la culture que cela implique. La technique occasionne des changements d'organisation drastique au sein des entreprises. Elle n'est pas toujours simple à prendre en main, ce qui peut causer une certaine frustration.
- Le manque de visibilité. En effet, la manière de faire se base sur des feuilles de route allant de 12 à 24 mois. Il peut parfois être difficile de prévoir à la fois les coûts, le temps et les ressources nécessaires.
- La complexité de mise en place de la méthodologie pour les grandes entreprises. La technique fonctionne mieux au sein des petites ou moyennes entreprises. Il est préférable de former des équipes de 5 à 9 personnes. Avec des grosses équipes, il est plus compliqué de coordonner les différents collaborateurs.

En résumé, l'approche Agile se distingue par sa capacité à offrir une flexibilité accrue et une visibilité optimisée dans la gestion de projet. Dans un contexte où la personnalisation revêt une importance croissante, cette méthodologie gagne de plus en plus d'adeptes. Sa souplesse s'illustre particulièrement dans les projets qui nécessitent une réalisation rapide ou dont les paramètres évoluent rapidement.

Cette méthodologie s'avère être une alliée idéale pour les projets appelés à évoluer en cours de route, fournissant la structure nécessaire sans entraver l'efficacité des équipes. Elle autorise des ajustements et des optimisations constants, contribuant ainsi à une gestion de projet plus agile, classe et professionnelle.

Poursuivons notre exploration méthodologique en abordant l'approche DevOps, qui se concentre sur l'amélioration de la collaboration et de l'intégration entre les équipes de développement et d'exploitation.

III. L'Approche DevOps

III.1. Définition

L'approche DevOps est une forme de culture d'entreprise qui peut être adoptée au sein des entreprises dans le domaine des logiciels et du développement de logiciel. Il s'agit d'une combinaison d'approches, de pratiques et d'outils grâce auxquels les entreprises peuvent considérablement accélérer la mise au point des logiciels et leur applicabilité tout en améliorant la qualité[8].

III.2. Principes de DevOps

Le DevOps repose avant tout sur un ensemble de principes. En 2010, Damon Edwards et John Willis ont résumé ces principes à travers l'acronyme "CAMS" : Culture, Automation, Measurement, Sharing (culture, automatisation, mesure, partage).

Dans cette section, nous explorerons les bases du DevOps, mettant en lumière ses principes fondamentaux[9].

- **Intégration Continue (CI) :** La CI consiste à fusionner régulièrement les changements de code de plusieurs développeurs dans un référentiel partagé. Ce processus garantit que tous les changements sont testés, intégrés et validés tôt, minimisant les problèmes d'intégration ultérieurs.

- **Livraison Continue (CD) :** La CD se concentre sur l'automatisation de la mise en production et du déploiement de logiciels. En adoptant la CD, les organisations peuvent livrer des logiciels de haute qualité plus fréquemment et avec moins d'effort manuel.

- **Infrastructure en tant que Code (IaC) :** L'IaC consiste à gérer l'infrastructure et à provisionner des ressources à l'aide de code plutôt que d'une configuration manuelle. Cela permet l'automatisation et le contrôle de version de l'infrastructure, facilitant la réplication et le déploiement cohérent des environnements.

- **Surveillance et Journalisation :** La surveillance continue et la journalisation sont essentielles dans le DevOps pour obtenir une visibilité sur les performances de l'application, détecter les problèmes et assurer une résolution rapide. Cela permet aux équipes d'identifier et de résoudre proactivement les goulots d'étranglement ou les défaillances potentiels.

- **Collaboration et Communication :** Une collaboration et une communication efficaces sont cruciales dans un environnement DevOps. Les équipes doivent partager des connaissances, travailler étroitement ensemble et utiliser des outils qui facilitent la communication pour favoriser une culture de collaboration et de transparence.

III.3. Méthodes DevOps

Afin d'accélérer et d'améliorer le développement et le lancement de leurs produits, les entreprises disposent de plusieurs méthodologies et pratiques DevOps de développement logiciel. Les méthodes Scrum, Kanban et Agile sont les plus couramment utilisées[10].

- **Scrum.** La méthode Scrum définit la manière dont les membres de l'équipe doivent collaborer pour accélérer les projets de développement et d'assurance qualité. Les pratiques Scrum utilisent des workflows clés, une terminologie spécifique (sprint, time box, daily scrum) et des rôles désignés (Scrum Master, product owner ou responsable de produit).
- **Kanban.** Développée par Toyota pour améliorer l'efficacité de ses usines de montage, la méthode Kanban repose sur un suivi des travaux en cours (TEC) dans un projet logiciel à l'aide d'un tableau de Kanban.
- **Agile.** Les premières méthodes Agile de développement logiciel agile continuent d'influencer largement les pratiques et les outils DevOps. De nombreuses approches DevOps, notamment Scrum et Kanban, intègrent des éléments de la programmation agile. Certaines pratiques agiles offrent une meilleure réactivité face à l'évolution des besoins en documentant les exigences sous forme de user stories, en organisant des

réunions quotidiennes (daily standups) et en intégrant les retours des clients de manière continue. La méthodologie agile préconise également des cycles de développement logiciel plus courts, à l'encontre des méthodes classiques dites « en cascade » plus chronophages.

III.4. Chaine d'outils DevOps

Les adeptes de la pratique Développement et Opérations s'appuient souvent sur des outils DevOps compatibles dans leur «chaîne d'outils» pour rationaliser, accélérer et automatiser davantage les différentes étapes du workflow (ou «pipeline») de fourniture des logiciels. Ces outils renforcent les principes fondamentaux du DevOps tels que l'automatisation, la collaboration et l'intégration entre les équipes chargées du développement et des opérations. Voici quelques exemples d'outils employés à différentes étapes du cycle de vie DevOps[10].

- **Planification :** Cette phase permet de définir la valeur commerciale et les exigences. Jira et Git peuvent être utilisés pour le suivi des problèmes connus et la gestion des projets.
- **Code :** Cette phase inclut la conception logicielle et la création du code logiciel à l'aide des logiciels GitHub, GitLab, Bitbucket ou Stash, par exemple.
- **Création :** Cette phase consiste à gérer les versions logicielles et à exploiter des outils automatisés pour compiler et intégrer le code en vue de sa mise en production. Des référentiels de code source ou de package « empaquettent » aussi l'infrastructure requise pour la livraison du produit à l'aide des logiciels Docker, Ansible, Puppet, Chef, Gradle, Maven ou JFrog Artifactory, par exemple.
- **Test :** Cette phase comprend des tests continus, qu'ils soient manuels ou automatisés, et vise à assurer une qualité de code optimale à l'aide des logiciels JUnit, Codeception, Selenium, Vagrant, TestNG ou BlazeMeter, par exemple.
- **Déploiement :** Cette phase peut inclure des outils de gestion, de coordination, de planification et d'automatisation de la mise en production des produits, avec Puppet, Chef, Ansible, Jenkins, Kubernetes, OpenShift, OpenStack, Docker ou Jira, par exemple.
- **Exploitation :** Cette phase permet de gérer les logiciels en production à l'aide des logiciels Ansible, Puppet, PowerShell, Chef, Salt ou Otter, par exemple.
- **Supervision :** Cette phase permet d'identifier les problèmes affectant une version logicielle en production et de collecter les informations correspondantes à l'aide des

logiciels New Relic, Datadog, Grafana, Wireshark, Splunk, Nagios ou Slack, par exemple.

- La figure ***(figure 4)*** ci-dessous montre les phases de l'approche DevOps avec les outils que l'on peut utiliser pour chaque phase

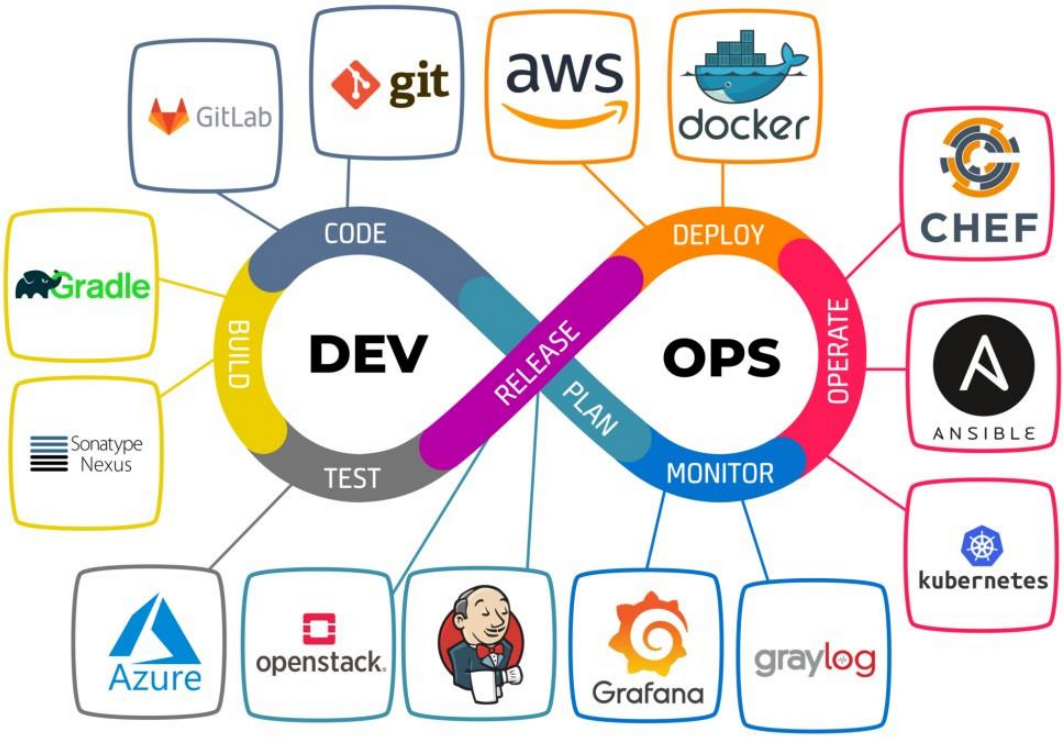

Figure 4 : Approche DevOps[11]

III.5. Avantages du DevOps

Pour les entreprises qui suivent les pratiques DevOps, les avantages commerciaux et techniques sont évidents et la plupart contribuent à améliorer la satisfaction des clients[10] :

- Accélération et amélioration de la fourniture des produits
- Résolution plus rapide des problèmes et complexité réduite
- Plus grande évolutivité et disponibilité inégalée
- Stabilité accrue des environnements d'exploitation
- Meilleure utilisation des ressources
- Automatisation accrue
- Meilleure visibilité sur les résultats du système
- Innovation renforcée

III.6. Inconvénients du DevOps

Bien que DevOps présente de nombreux avantages en termes d'efficacité, de collaboration et de rapidité de déploiement, il présente également des défis et des inconvénients potentiels. Voici quelques-uns des inconvénients fréquemment évoqués[12]:

- **Complexité accrue** : l'adoption de DevOps peut souvent entraîner une complexité accrue au sein de l'infrastructure informatique d'une organisation. En intégrant plusieurs outils et technologies dans le processus de développement logiciel, les entreprises peuvent créer un environnement de production plus complexe, difficile à gérer et à dépanner. En outre, DevOps peut obliger les organisations à investir dans des ressources matérielles et logicielles supplémentaires, ce qui peut encore accroître la complexité et les coûts.
- **Manque de normalisation :** Il n'existe actuellement aucune standardisation à l'échelle de l'industrie pour DevOps. Par conséquent, les entreprises qui adoptent DevOps peuvent avoir besoin de créer leurs propres processus et ensembles d'outils personnalisés, ce qui peut s'avérer long et coûteux. De plus, le manque de standardisation peut entraîner une confusion parmi les employés quant à la meilleure façon d'utiliser les techniques DevOps.
- **Coûts accrus** : l'adoption de DevOps peut souvent entraîner une augmentation des coûts pour les entreprises. En exigeant l'achat de ressources matérielles et logicielles supplémentaires et l'embauche de professionnels DevOps expérimentés, les entreprises peuvent constater une augmentation significative de leurs dépenses informatiques. De plus, la complexité d'un environnement DevOps peut souvent entraîner des problèmes de fiabilité et de performances des applications critiques pour l'entreprise.

En résumé, de nombreuses méthodes DevOps destinées à rationaliser le développement et le déploiement des logiciels reposent sur des modèles agiles tels que la programmation Lean.

Après avoir examiné attentivement les caractéristiques et les avantages propres à chaque méthodologie, nous allons à présent dresser un tableau comparatif pour mieux visualiser les différences entre la méthode en Cascade, Agile et DevOps. Cette analyse approfondie devrait nous permettre de déterminer celle qui s'aligne le mieux avec les exigences et les réalités de notre projet.

Caractéristique	Méthodologie en Cascade	Méthodologie Agile	Méthodologie DevOps
Flexibilité	Faible	Elevée	Elevée
Itérations	Non itérative	Itérative	Itérative
Livraison rapide	Non adapté	Adapté	Adapté
Communication	Faible	Elevée	Elevée
Risque de changement	Elevé	Faible	Faible
Adaptabilité	Faible	Elevée	Elevée
Gestion du changement	Difficile	Souple	Souple
Collaboratif	Faible	Elevé	Elevé
Retours fréquents	Non adapté	Adapté	Adapté
Risque de dépassement du budget	Elevé	Faible	Faible

Tableau 2 : Tableau comparatif des différentes méthodologies étudiées

IV. Adoption de la méthode Scrum Agile

L'essence même de la méthode Agile réside dans une collaboration intensive avec le client, requérant des échanges fréquents avec les utilisateurs finaux. Dès l'amorce de notre projet au sein de l'UFR des sciences de la santé de l'Université, la diversité des profils des utilisateurs a imposé une considération minutieuse des données à traiter dans notre application. La nécessité

de prendre en compte cette diversité dès les premières phases du projet a cristallisé l'évidence de l'adoption d'une approche AGILE.

Notre choix s'est naturellement porté sur SCRUM Agile, une méthode qui se prête parfaitement à la gestion dynamique des besoins émergents. Les premières rencontres ont mis en lumière une variété de perspectives, confirmant la pertinence d'une approche itérative et collaborative. Dans ce contexte, les chefs de département, le directeur, et le responsable pédagogique de l'UFR ont assumé le rôle crucial de Product Owner.

L'équipe de développement, composée de trois membres, a été soigneusement constituée. Le chef d'équipe, également notre encadreur, a endossé la fonction de Scrum Master, tandis que mon binôme et moi-même avons assumé le rôle de développeurs.

Conclusion

Ce chapitre a dévoilé une pléthore de méthodologies de développement, chacune portant son lot de nuances et de spécificités. Notre choix s'est résolument ancré dans le royaume des méthodologies agiles, avec une inclination particulière pour Scrum, une approche qui guide l'intégralité de notre travail. Ce choix s'est forgé sur la nécessité d'une implication accrue du client tout au long du processus de développement.

Au-delà du choix méthodologique, l'importance cruciale de la communication et de la collaboration au sein de l'équipe est soulignée. Ces éléments jouent un rôle tout aussi essentiel que la méthodologie elle-même dans la réussite d'un projet.

Le prochain chapitre, consacré à la spécification des besoins, est le terrain où cette méthodologie agile sera mise en pratique.

Chapitre III : Spécification des besoins

Introduction

Ce chapitre dévoile l'anatomie de la spécification des besoins. Notre première démarche consiste à mettre en lumière les utilisateurs, minutieusement identifiés à travers la subdivision du système en sous-modules, offrant ainsi une toile de fond propice à ce travail. Une fois les bases posées, un choix judicieux de langages de modélisation, éléments clés du processus, est effectué.

L'essence de ce chapitre se révèle dans la dissection des exigences fonctionnelles de l'application. À cet égard, les diagrammes de cas d'utilisation prennent le devant de la scène, déployant une cartographie visuelle détaillée du comportement fonctionnel de notre application. Cet outil devient notre allié privilégié pour exprimer et identifier les contours précis de chaque fonctionnalité.

I. Les acteurs du système

Au cœur de tout système, qu'il soit informatique, social, économique, ou de toute autre nature, réside l'acteur. Cette entité, aux contours définis, interagit de manière cruciale avec le système dans son ensemble. Les acteurs, véritables piliers, peuvent être des individus, des groupes, des organisations, des entités, ou des éléments, tous jouant un rôle majeur ou exerçant une influence significative sur le fonctionnement, les performances, ou les résultats du système.

I.1. Profils des Acteurs

Au sein de ce système, un écosystème dynamique s'anime à travers des acteurs aux profils distincts. Chacun de ces acteurs contribue de manière cruciale à l'orchestration harmonieuse du système. Voici un aperçu des profils des acteurs :

- **L'Administrateur** : En tant que superviseur du système, il est responsable de la gestion globale, y compris le contrôle des autorisations, la gestion des utilisateurs et la maintenance du système.

- **Le Responsable Pédagogique** : Il joue un rôle crucial en tant que garant de l'aspect pédagogique du système, assurant la supervision des activités académiques et le suivi des performances des étudiants.
- **Le Chef de Département** : Il agit en tant que dirigeant du département concerné, coordonnant les activités départementales et interagissant avec le système pour répondre à des besoins spécifiques.

II. Les modules du système

La clarté et la précision des besoins fonctionnels nécessitent une subdivision rigoureuse en sous-modules. Cette démarche vise à rendre chaque composant du système plus explicite, offrant ainsi une vision détaillée de chaque aspect. Les trois modules qui retiennent particulièrement notre attention se déclinent en plusieurs sous-modules stratégiques, chaque subdivision étant minutieusement orchestrée pour mieux répondre à leurs exigences spécifiques.

Modules	**Sous-modules**	**Description**
Authentification		Ce sous-module se distingue en tant que pièce maîtresse, détenant une importance capitale au sein de notre système. Sa fonction primordiale réside dans la vérification minutieuse de l'identité de chaque personne cherchant à se connecter, établissant ainsi un contrôle précis des niveaux d'habilitation.
Gestion des utilisateurs	Comptes utilisateurs	Ce sous-module octroie à l'administrateur la maîtrise des comptes utilisateurs du système. Il se profile comme l'outil central permettant la gestion fluide et efficace des profils au sein de l'écosystème.
Gestion des hôpitaux	Hôpitaux	Pour le sous-module "Hôpitaux", le responsable pédagogique est habilité à créer et modifier les hôpitaux du système, assurant ainsi une gestion flexible et personnalisée de ces entités.
	Spécialités	Quant au sous-module "Spécialités", il confère au responsable pédagogique la capacité de créer et de modifier les spécialités au niveau des hôpitaux, offrant ainsi une adaptation précise des domaines de compétence dans le système.

	Services	Le sous-module "Services" octroie au responsable pédagogique la possibilité de créer et de modifier les services au sein des spécialités au niveau des hôpitaux, garantissant une gestion fine des différentes prestations offertes par ces établissements.
Gestion des années académiques	Années et Semestres	Ce sous-module dédié à la gestion des années confère au responsable pédagogique le pouvoir de structurer et de superviser le déroulement des années académiques. Il permet une gestion efficace du démarrage des semestres, assurant ainsi une planification optimale.
	Modules	Ce sous-module offre la fonctionnalité de lancement des modules de manière flexible, adaptée à une année spécifique et à un semestre donné. Il donne au chef de département le contrôle sur la mise en œuvre des enseignements.
	Assignations et Affiliations	Ce sous-module donne au chef de département la possibilité d'attribuer de manière ajustable des modules à des enseignants, offrant une flexibilité dans la répartition des tâches. Parallèlement, il permet au responsable pédagogique d'affilier de manière adaptable un secrétaire à un hôpital, garantissant une gestion dynamique du personnel

Tableau 3 : Liste des modules et leurs descriptions

III. Langage de modélisation UML

III.1. Qu'est-ce que le langage UML ?

Le langage UML (Unified Modeling Language, ou langage de modélisation unifié) a été pensé pour être un langage de modélisation visuelle commun, et riche sémantiquement et syntaxiquement. Il est destiné à l'architecture, la conception et la mise en œuvre de systèmes logiciels complexes par leur structure aussi bien que leur comportement. L'UML a des applications qui vont au-delà du développement logiciel, notamment pour les flux de processus dans l'industrie[13].

III.2. Les différents types de diagramme UML

Pour les néophytes, le nombre de diagrammes UML peut paraître infini. En vérité, les normes en identifient 13 types, eux-mêmes répartis en deux groupes, tels que décrits ci-dessous[14].

- **Diagrammes UML structurels :** Les diagrammes UML structurels, comme leur nom l'indique, illustrent la structure d'un système, notamment les classes, les objets, les paquetages, les composants, etc., et les relations entre ces éléments.
  - ❖ **Diagramme de classes**
  - ❖ **Diagramme de composants**
  - ❖ **Diagramme de déploiement**
  - ❖ **Diagramme de structure composite**
  - ❖ **Diagramme d'objets**
  - ❖ **Diagramme de paquetages**

- **Diagrammes de profil :** Récemment ajoutés à UML 2.0, les diagrammes de profil sont uniques et rarement utilisés dans les spécifications. Un diagramme de profil est mieux compris comme un mécanisme d'extensibilité pour personnaliser les modèles UML pour des domaines et des plates-formes spécifiques.
- **Diagrammes UML comportementaux :** Ces diagrammes UML représentent la manière dont le système se comporte et interagit avec lui-même et avec les utilisateurs, les autres systèmes et les autres entités.
  - ❖ **Diagramme de temps**
  - ❖ **Diagramme d'aperçu des interactions**
  - ❖ **Diagramme de communication**
  - ❖ **Diagramme de cas d'utilisation**
  - ❖ **Diagramme de séquence**
  - ❖ **Diagramme d'activités**

IV. Les diagrammes de cas d'utilisation

IV.1. Présentation

En langage UML, les diagrammes de cas d'utilisation modélisent le comportement d'un système et permettent de capturer les exigences du système.

Les diagrammes de cas d'utilisation décrivent les fonctions générales et la portée d'un système. Ces diagrammes identifient également les interactions entre le système et ses acteurs. Les cas d'utilisation et les acteurs dans les diagrammes de cas d'utilisation décrivent ce que le système fait et comment les acteurs l'utilisent[15].

IV.2. Représentation des sous-modules

Dans cette section, est effectuée une représentation visuelle des sous-modules de notre système à l'aide de diagrammes de cas d'utilisation. Les figures 5 à 11 illustrent respectivement les sous-modules suivants : Comptes utilisateurs, Hôpitaux, Spécialités, Services, Années et semestres, Modules, Assignations et Affiliations.

- **Gestion des utilisateurs : Comptes utilisateurs**

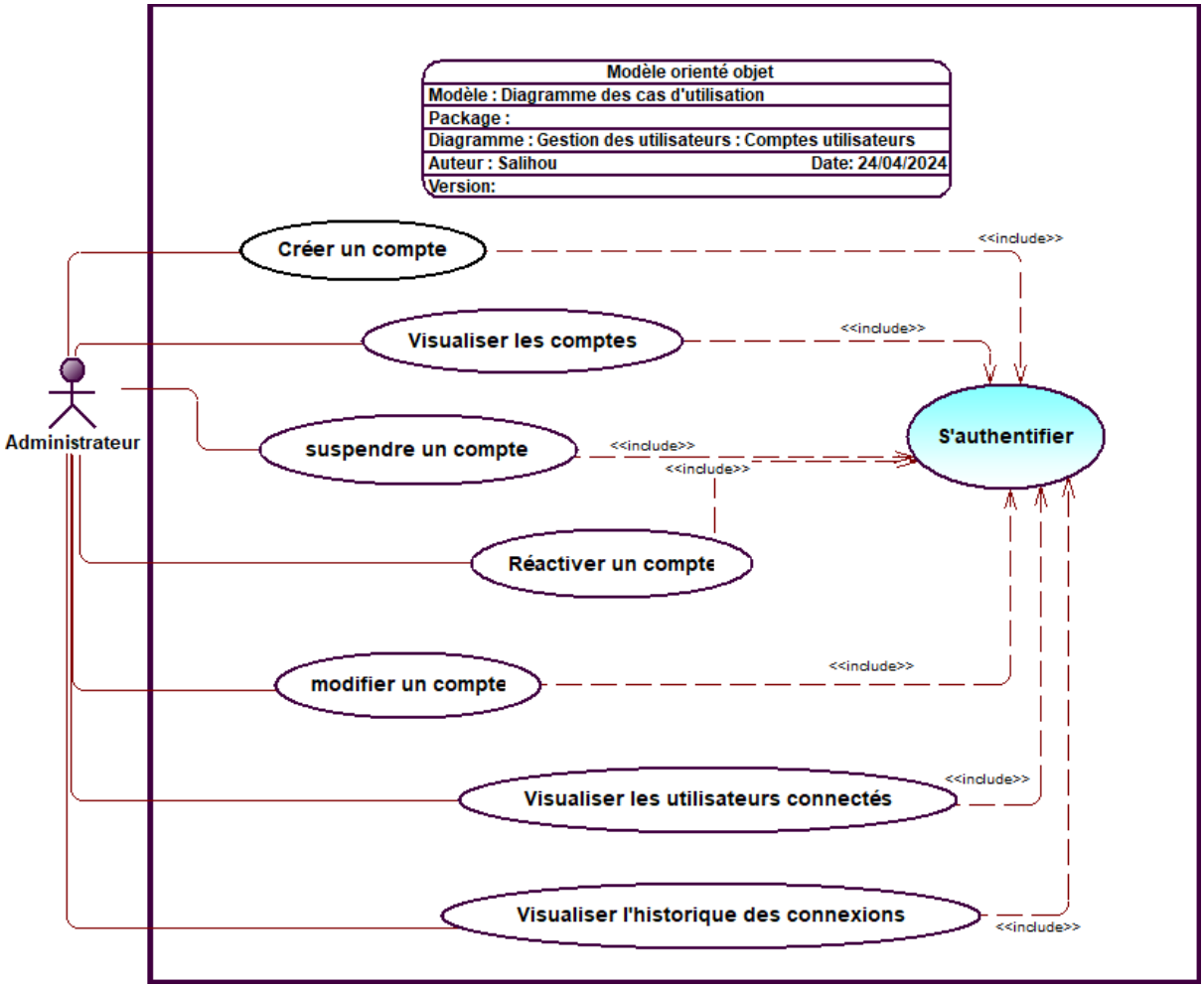

Figure 5 : Diagramme des cas d'utilisation pour la gestion des comptes utilisateurs

Explication des fonctionnalités de la figure 5 :

- **Créer un compte :** L'administrateur exerce la faculté de créer des comptes utilisateurs en leur attribuant le profil correspondant.
- **Modifier un compte :** L'administrateur est en mesure de modifier les comptes utilisateurs, intervenant en cas d'erreurs ou de nécessité de mise à jour.
- **Visualiser les comptes :** L'administrateur a la capacité de visualiser l'ensemble des comptes utilisateurs, assurant ainsi une gestion globale et transparente du système.
- **Suspendre un compte :** L'administrateur peut suspendre un compte, le rendant inactif dans le système en cas de besoin de restriction d'accès.
- **Réactiver un compte :** L'administrateur a la possibilité de réactiver un compte précédemment suspendu, le rendant à nouveau actif pour une utilisation normale du système.

- **Gestion des hôpitaux : Hôpitaux**

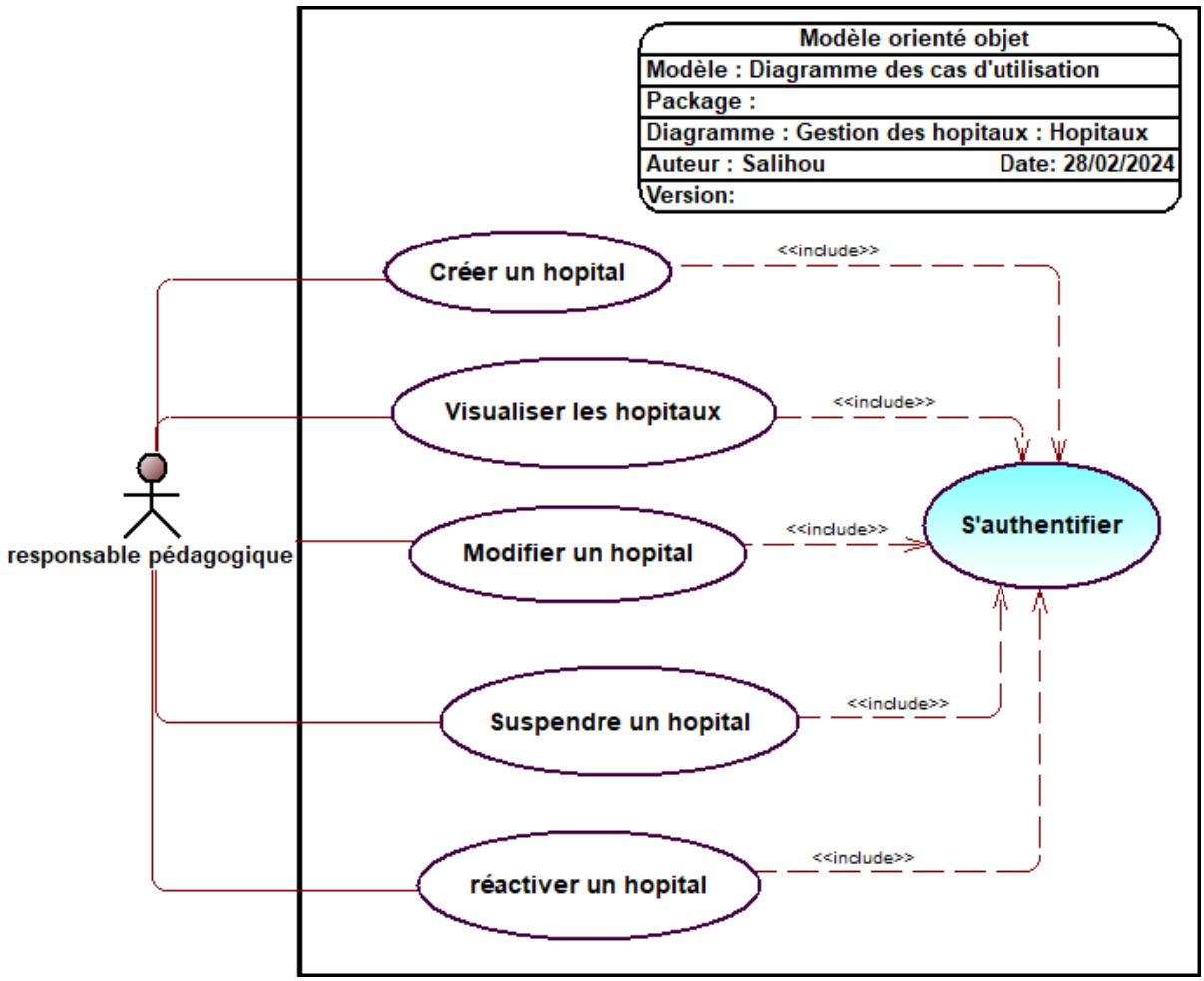

Figure 6 : Diagramme des cas d'utilisation pour la gestion des hôpitaux

Explication des fonctionnalités de la figure 6 :

- **Créer un hôpital :** Le responsable pédagogique a la compétence d'ajouter un nouvel hôpital au système.
- **Modifier un hôpital :** Il est en mesure de modifier les informations (noms) relatives aux hôpitaux, collaborant étroitement avec l'UFR en cas de nécessité.
- **Visualiser les hôpitaux :** Le responsable pédagogique dispose de la capacité de visualiser l'ensemble des hôpitaux.
- **Suspendre un hôpital :** Il a la faculté de suspendre un hôpital, le rendant temporairement non opérationnel selon les besoins.
- **Réactiver un hôpital :** Le responsable pédagogique peut réactiver un hôpital précédemment suspendu, lui redonnant son statut opérationnel.

- **Gestion des : hôpitaux : Spécialités**

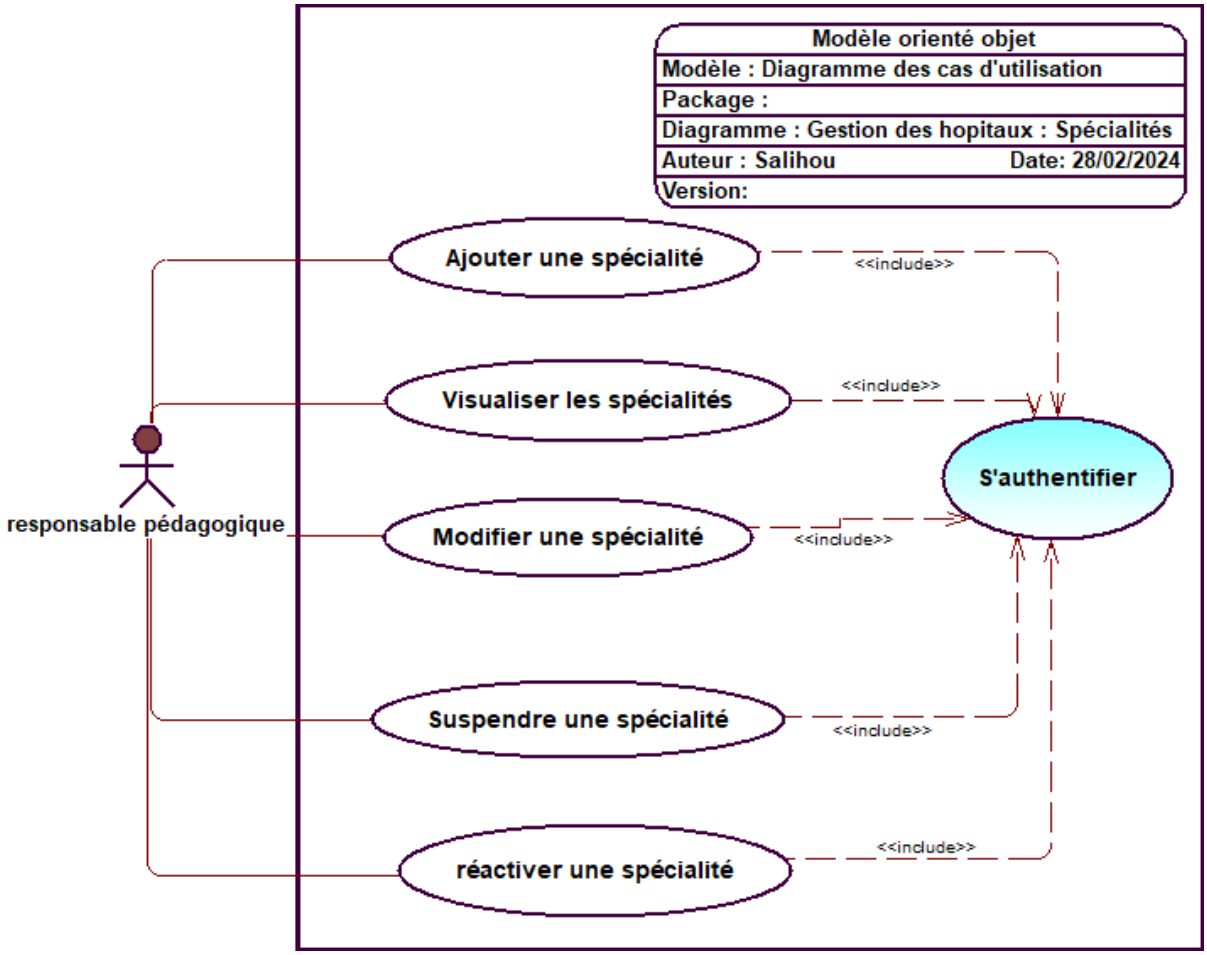

Figure 7 : Diagramme des cas d'utilisation pour la gestion des spécialités

Explication des fonctionnalités de la figure 7 :

- **Ajouter une spécialité :** Le responsable pédagogique est habilité à ajouter une nouvelle spécialité à un hôpital (Médicale ou Chirurgicale).

- **Modifier une spécialité :** Il est en mesure de modifier les informations relatives à la spécialité d'un hôpital, notamment les noms associés.
- **Visualiser les spécialités :** Le responsable pédagogique dispose de la capacité de visualiser l'ensemble des spécialités.
- **Suspendre une spécialité :** Il a la faculté de suspendre une spécialité, la rendant temporairement non opérationnelle selon les besoins.
- **Réactiver une spécialité :** Le responsable pédagogique peut réactiver une spécialité précédemment suspendue, lui redonnant son statut opérationnel. Il est important de noter que cette tâche ne peut être effectuée que si l'hôpital en question est opérationnel.

- **Gestion des hôpitaux : Services**

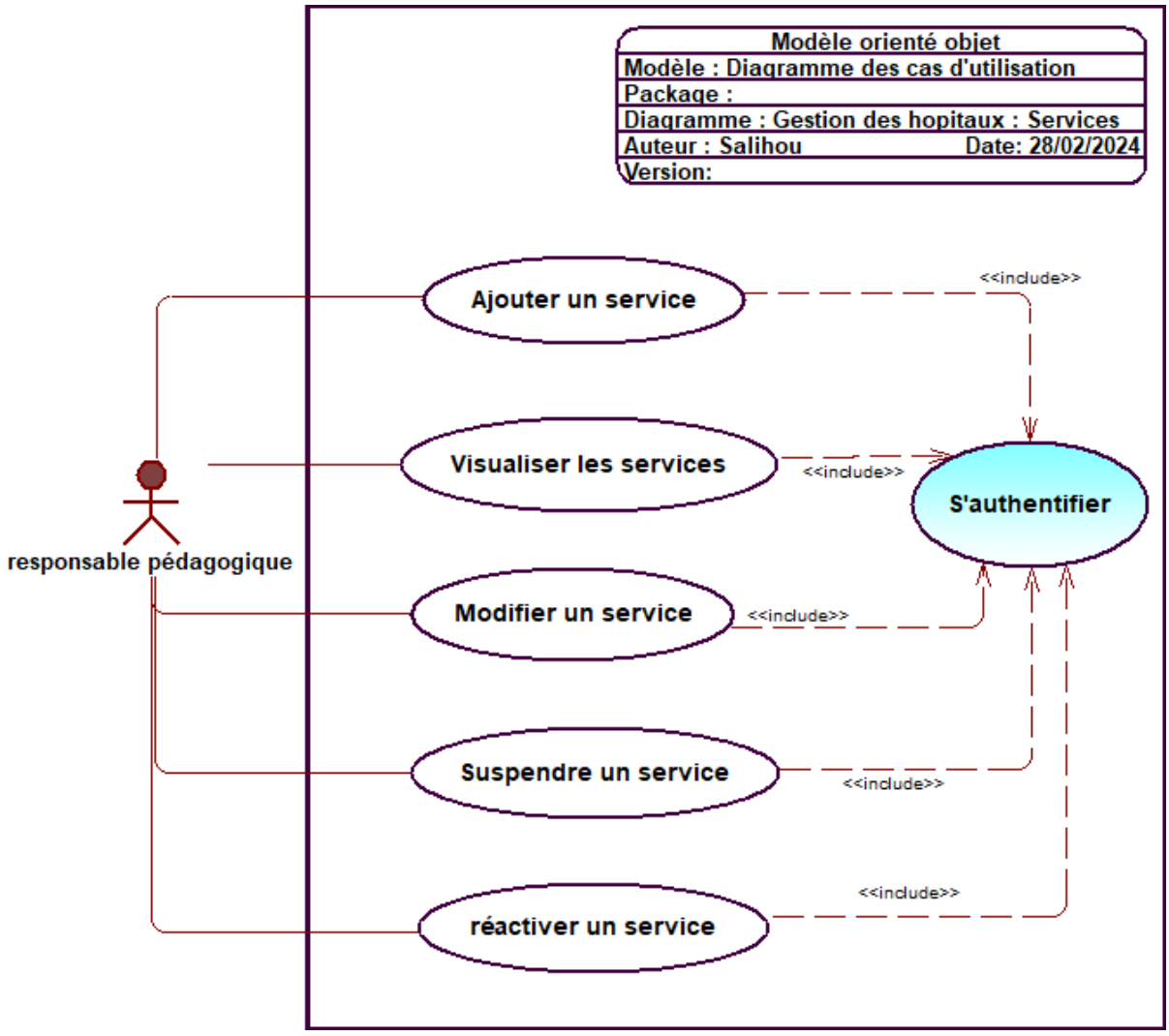

Figure 8 : Diagramme des cas d'utilisation pour la gestion des services

Explication des fonctionnalités de la figure 8 :

- ❖ **Ajouter un service :** Le responsable pédagogique est habilité à intégrer un nouveau service à une spécialité existante.
- ❖ **Modifier un service :** Il est en mesure de modifier les informations relatives à un service, notamment les noms associés.
- ❖ **Visualiser les services :** Le responsable pédagogique dispose de la capacité de visualiser l'ensemble des services.
- ❖ **Suspendre un service :** Il a la faculté de suspendre un service, le rendant temporairement non opérationnel selon les besoins.
- ❖ **Réactiver un service :** Le responsable pédagogique peut réactiver un service précédemment suspendu, lui redonnant son statut opérationnel. Il est à noter que cette tâche ne peut être effectuée que si la spécialité à laquelle se trouve le service en question est opérationnelle.

- **Gestion des années académiques : Années et Semestres**

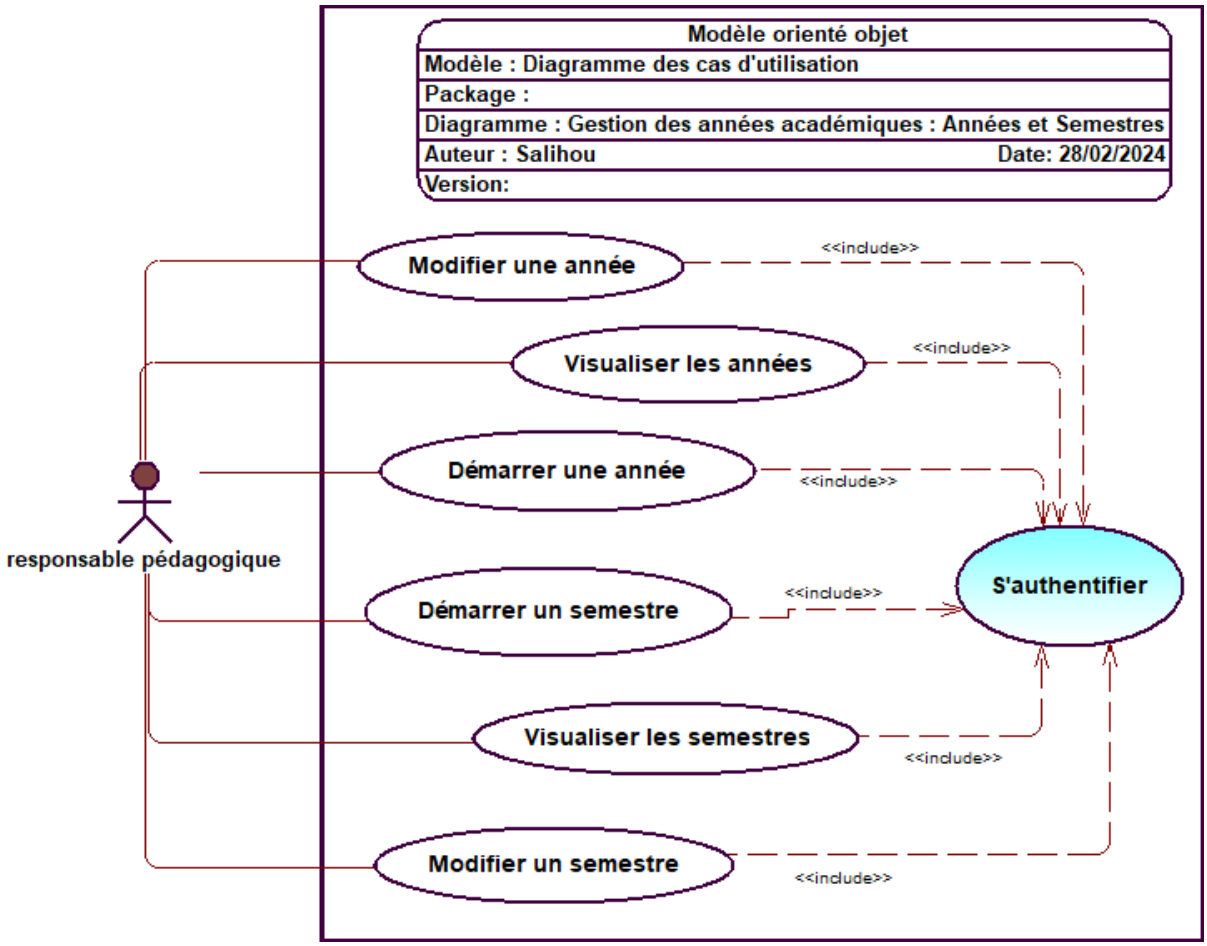

Figure 9 : Diagramme des cas d'utilisation pour la gestion des années et semestres

Explication des fonctionnalités de la figure 9 :

- **Démarrer une année :** Le responsable pédagogique est habilité à ajouter une nouvelle année académique.
- **Modifier une année :** Il est en mesure de modifier les informations relatives à une année existante.
- **Visualiser les années :** Le responsable pédagogique dispose de la capacité de visualiser l'ensemble des années académiques.
- **Démarrer un semestre :** Le responsable pédagogique est habilité à démarrer un nouveau semestre (Semestre 1 ou semestre 2) mais ce démarrage est fonction de l'existence de l'année.
- **Visualiser les semestres :** Il peut visualiser les semestres enregistrés dans le système.
- **Modifier un semestre :** Il peut effectuer des modifications sur un semestre existant en cas d'erreurs ou de changements nécessaires.

- **Gestion des années académiques : Modules**

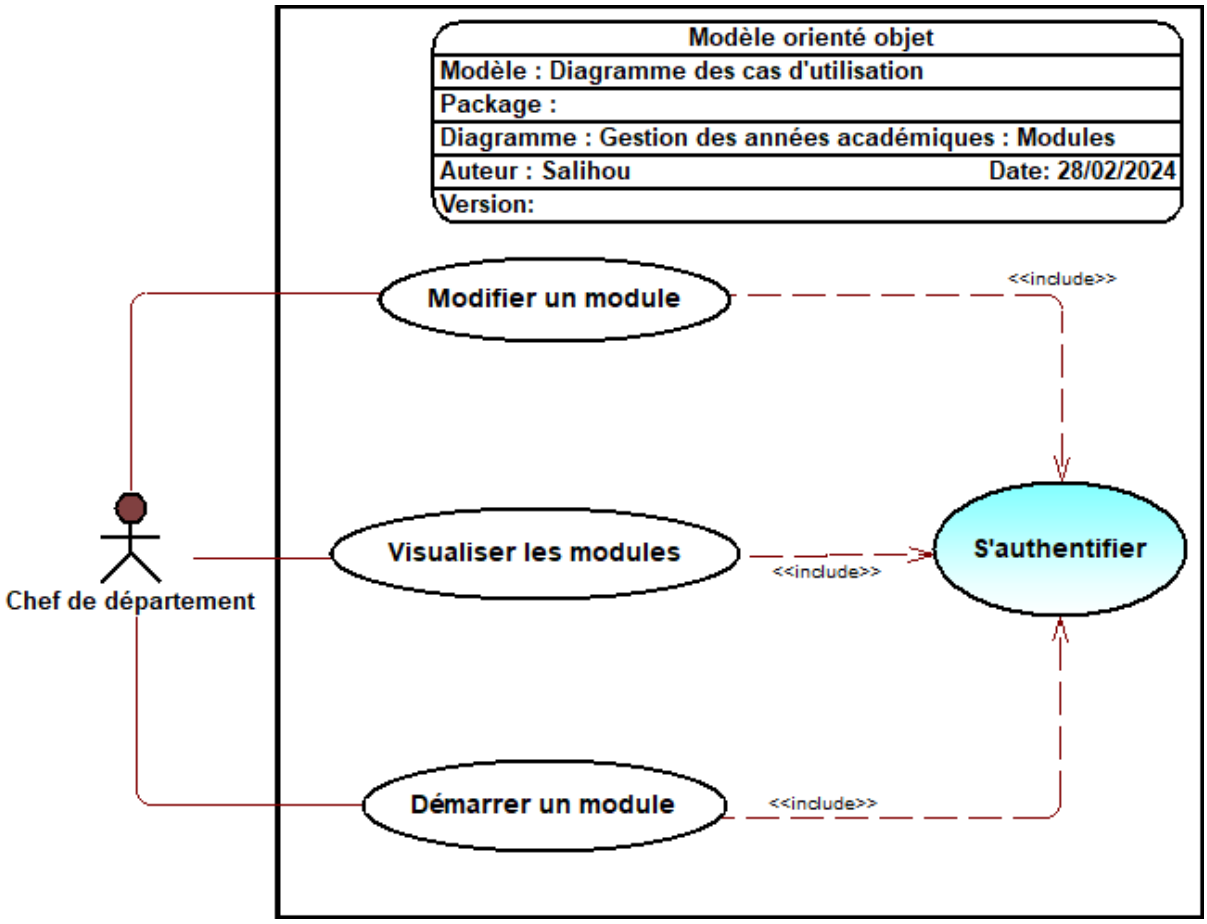

Figure 10 : Diagramme des cas d'utilisation pour la gestion des modules

Explication des fonctionnalités de la figure 10 :

- ❖ **Démarrer un module :** Cette fonctionnalité accorde au chef de département le pouvoir de lancer un nouveau module, adapté à un niveau spécifique, en tenant compte de l'année académique et du semestre en cours.
- ❖ **Modifier un module :** Il est en mesure de modifier les informations relatives à un module existant.
- ❖ **Visualiser les modules :** Le chef de département dispose de la capacité de visualiser l'ensemble des modules pour une année académique.

- **Gestion des années académiques : Assignations et Affiliations :**

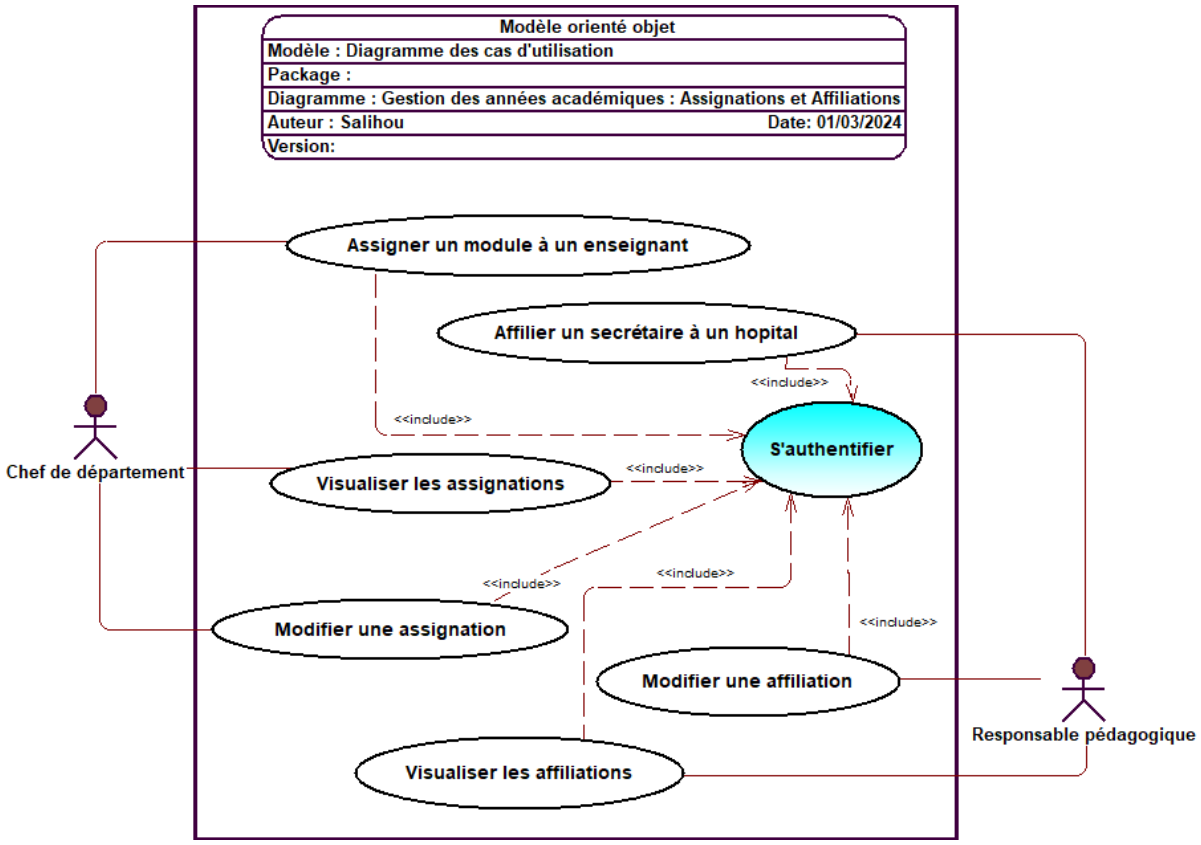

Figure 11 : Diagramme des cas d'utilisation pour la gestion des assignations et affiliations

Explication des fonctionnalités de la figure 10 :

- ❖ **Assigner un module à un enseignant :** Le chef de département est habilité à attribuer un nouveau module à un enseignant, contribuant ainsi à la gestion des enseignements.
- ❖ **Visualiser les assignations :** Le chef de département peut visualiser toutes les informations relatives aux assignations existantes, offrant une vue d'ensemble des tâches attribuées.
- ❖ **Modifier les assignations :** Le chef de département dispose de la capacité de modifier les détails d'une assignation si des ajustements sont nécessaires.

- **Affilier un secrétaire à un hôpital :** Le responsable pédagogique peut affilier un secrétaire à un hôpital, facilitant la gestion administrative.
- **Visualiser les affiliations :** Le responsable pédagogique peut visualiser toutes les affiliations entre les secrétaires et les hôpitaux, assurant un suivi efficace.
- **Modifier les affiliations :** Le responsable pédagogique dispose de la capacité de modifier les affiliations existantes, permettant une flexibilité dans la gestion du personnel.

IV.3. Tableau récapitulatif des sous-modules et leurs cas d'utilisation

Sous-modules	**Cas d'utilisation**						
Authentification	**Authentification**						
Gestion des utilisateurs	Créer un compte	Visualiser un compte	Modifier un compte	Suspendre un compte	Réactiver un compte	Visualiser les utilisateurs connectés	Visualiser l'historique des connxions
hôpitaux	Ajouter un hôpital	Visualiser un hôpital	Visualiser un hôpital	Suspendre un hôpital	Réactiver un hôpital		
Spécialités	Ajouter une spécialité	Visualiser une spécialité	Modifier une spécialité	Suspendre une spécialité	Réactiver une spécialité		
Services	Ajouter un service	Visualiser un service	Modifier un service	Suspendre un service	Réactiver un service		
Années	Démarrer une année		Visualiser une année		Modifier une année		
Semestres	Démarrer un semestre		Visualiser un semestre		Modifier un semestre		
Modules	Démarrer un module		Visualiser un module		Modifier un module		
Gestion des affiliations	Assigner un module à un enseignant	Visualiser les assignations	Modifier une assignation	Affilier un secrétaire à un hôpital	Visualiser les affiliations	Modifier une affiliation	

Tableau 4 : Les sous-modules et leurs cas d'utilisation

Conclusion

En synthèse, ce chapitre permet d'identifier les acteurs clés de notre système, à savoir l'administrateur, le responsable pédagogique, et le chef de département. Les besoins

fonctionnels sont regroupés en modules, illustrés à l'aide de diagrammes de cas d'utilisation pour une visualisation claire et structurée. Un tableau récapitulatif des cas d'utilisation par module est également fourni. Cette étape préliminaire constitue la base solide sur laquelle nous allons conduire l'analyse et la conception dans la section suivante.

Chapitre IV : Analyse des besoins et Conception du Système

Introduction

La phase d'analyse et de conception constitue un pilier essentiel du processus de développement, visant à formaliser les premières étapes du système pour l'ajuster aux besoins spécifiques du client. Ce chapitre explore en profondeur l'analyse des besoins et les divers éléments de la conception, à savoir la conception générale et la conception détaillée. La partie analyse mettra en évidence certains diagrammes d'activités sans oublier quelques diagrammes de séquences pour une compréhension approfondie, tandis que la conception générale dévoilera l'architecture physique et logique sélectionnée pour l'application. Enfin, certains diagrammes de classes seront présentés pour illustrer la conception détaillée du système.

I. Analyse des besoins

En matière de définition, on parle d'analyse du besoin ou de recueil des besoins voire d'expression des besoins. Ces formulations renvoient au même concept. L'analyse des besoins vise à identifier les exigences du projet et à faire le point sur les éléments attendus. Il s'agit de contextualiser le projet et d'analyser les attentes pour donner un cadre au projet. On parle également de cadrage de projet[16].

Dans cette section, nous dévoilerons deux diagrammes d'activités, le premier dédié à la gestion des comptes utilisateurs et le second à celle des modules. S'ensuivront deux diagrammes de séquences : le premier explorant le processus d'authentification, le second se consacrant à la fonctionnalité d'ajout d'un hôpital.

I.1. Les diagrammes d'activités

I.1.1. Définition

Un diagramme d'activité, une facette dynamique et intégrale du langage de modélisation unifié (UML), est défini comme une représentation visuelle sophistiquée dans l'ingénierie logicielle et divers domaines. Il excelle à illustrer le flux continu d'activités, d'actions et de

processus au sein de systèmes complexes, de flux de travail d'entreprise ou de tout autre processus dynamique[17].

I.1.2. Diagramme d'activités pour la gestion des comptes utilisateurs

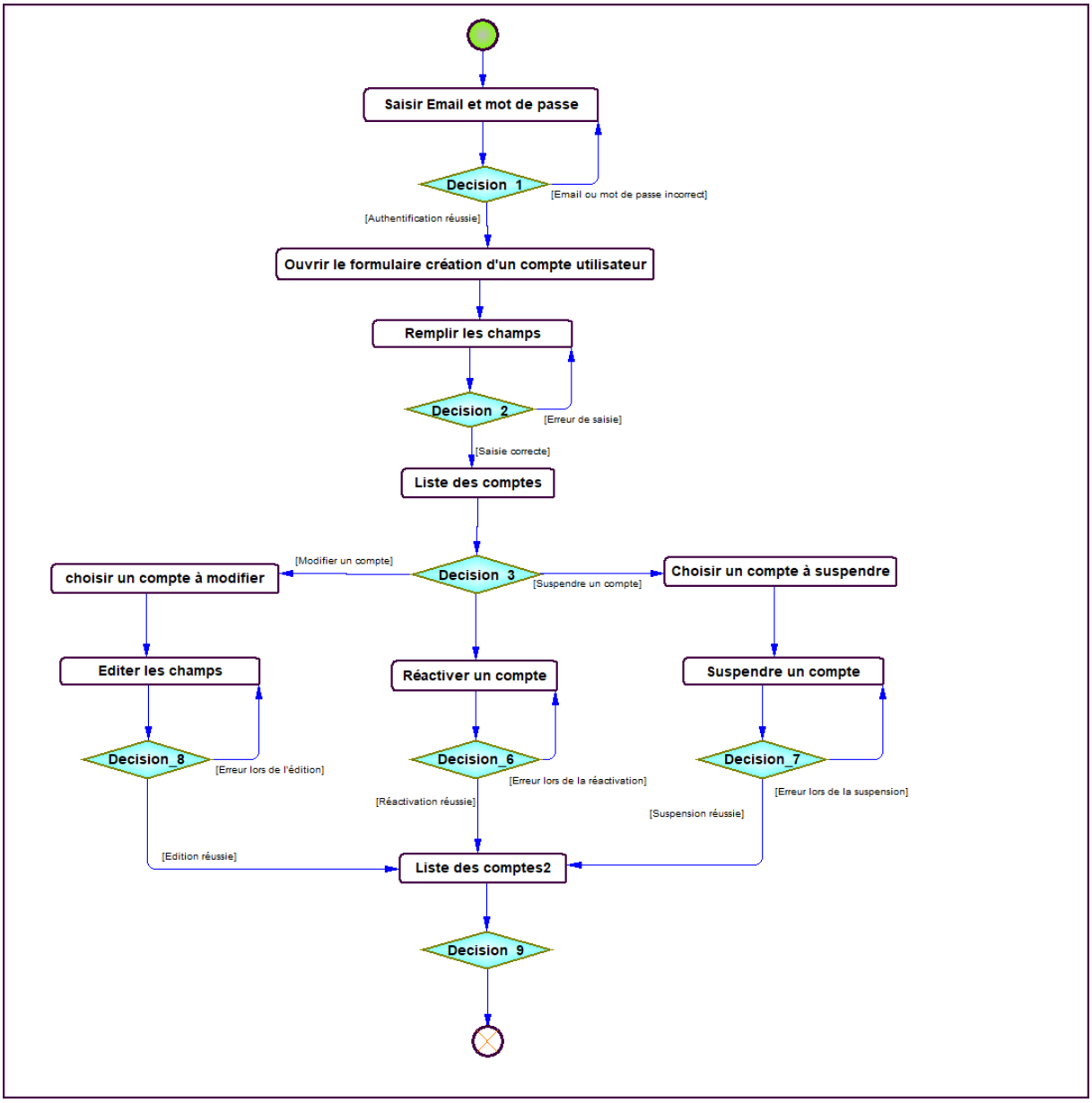

Figure 12 : Diagramme d'activités pour la gestion des comptes utilisateurs

Pour la gestion des comptes, l'administrateur doit s'authentifier pour avoir accès à cette interface. La figure ci-dessus (***figure 12***) offre une vision détaillée de ces étapes.

I.1.3. Diagramme d'activités pour la gestion des modules

Avant de pouvoir accéder à l'interface dédiée à la gestion des modules, une authentification en tant que chef de département est impérative. La figure *(figure 12)*, présentée ci-dessous, offre une visualisation complète des étapes à suivre.

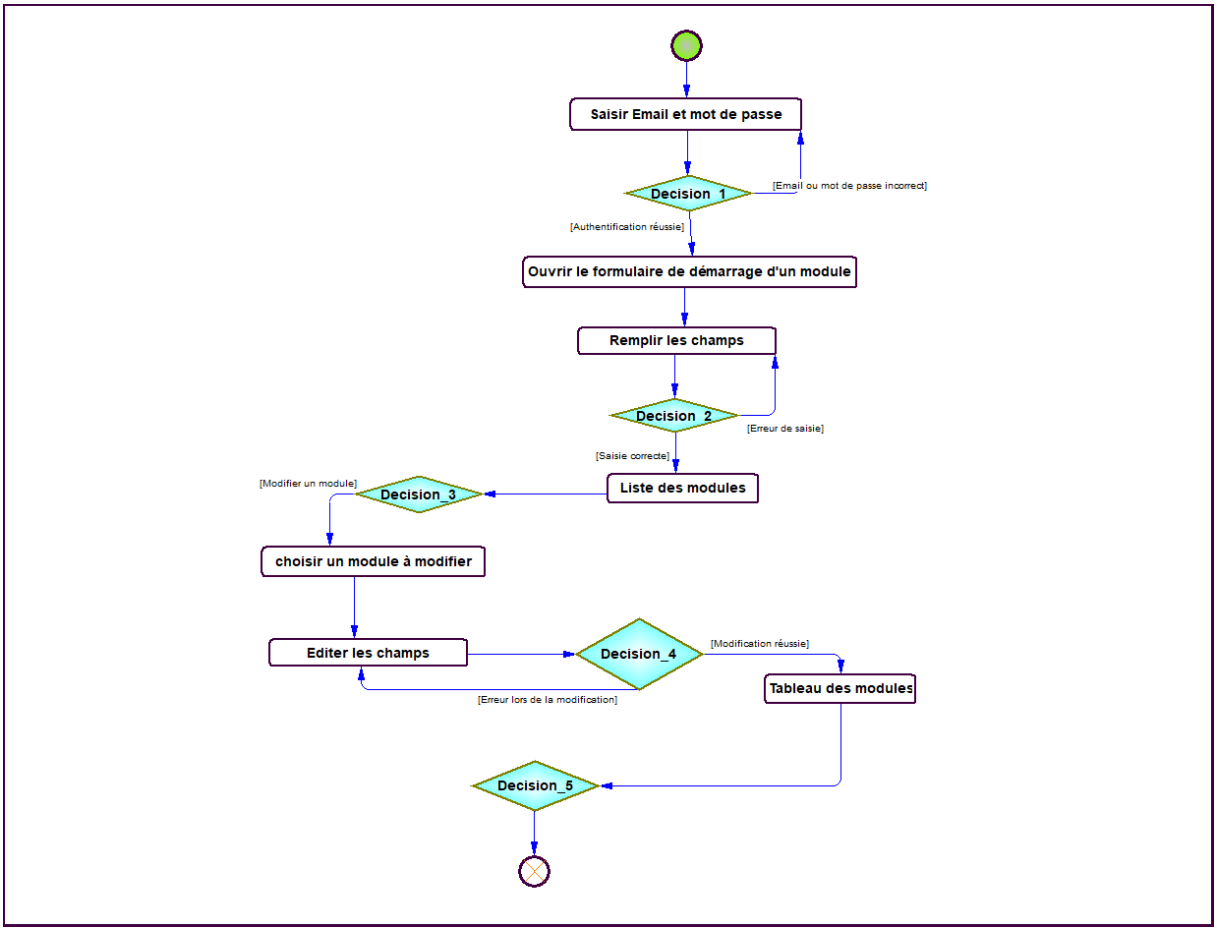

Figure 13 : Diagramme d'activités pour la gestion des modules

I.2. Les diagrammes de séquence

I.2.1. Définition

Un diagramme de séquence est défini comme un type de diagramme UML (Unified Modeling Language) utilisé dans le génie logiciel et la conception de systèmes pour visualiser les interactions et la communication entre divers composants ou objets au sein d'un système. Les diagrammes de séquence sont particulièrement utiles pour décrire le comportement dynamique d'un système et la manière dont les différents composants

collaborent pour réaliser une tâche ou un objectif spécifique. Ils sont couramment utilisés pendant les phases de conception et de documentation du développement de logiciels[18].

I.2.2. Diagramme de séquences pour l'authentification

La figure ci-dessous (***figure 14***) représente le diagramme de séquences du processus d'authentification d'un utilisateur (l'administrateur).

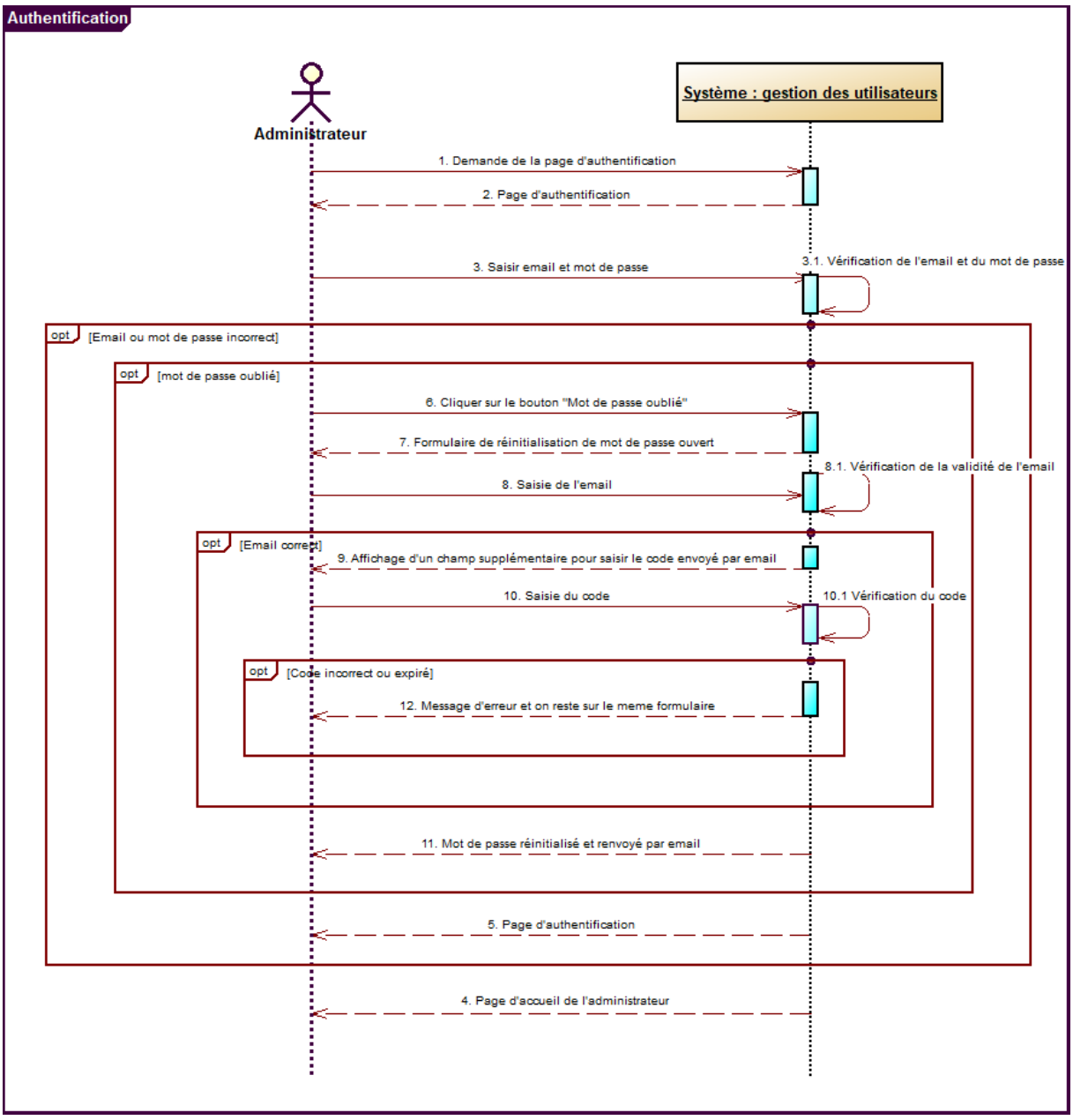

Figure 14 : Diagramme de séquences pour l'authentification

Le tableau ci-***dessous (tableau 5)*** donne une description détaillée de la ***figure 14*** ci-dessus

Scénario principal	
1. Demande de la page d'authentification	2. Ouverture de la page d'authentification contenant le formulaire de connexion
3. Saisir l'email et le mot de passe	3.1 Le système procède à la vérification des informations saisies
4. Page d'accueil de l'administrateur (si l'authentification s'est bien passée)	
Scénarios secondaires	
Si l'email et/ou le mot de passe sont incorrects	5. Le système affiche un message d'erreur 'Email ou mot de passe incorrect' et on reste sur la page d'authentification
Si le mot de passe est oublié	6. Cliquez sur le bouton 'Mot de passe oublié'
7. Un formulaire de réinitialisation de mot de passe avec le champ email s'ouvre	8. L'utilisateur renseigne l'email
8.1 Le système procède à la vérification de l'email Si l'email est valide	9. Un champ code vient compléter le formulaire de réinitialisation de mot de passe et parallèlement un email contenant un code est envoyé à l'utilisateur
10. L'utilisateur saisit le code	10.1 Le système procède à la vérification du code saisi
Si le code est valide	11. Le mot de passe est réinitialisé et automatiquement envoyé par email
Si le code n'est pas valide	12. Un message d'erreur 'Code invalide' est envoyé et on reste sur le même formulaire

Tableau 5 : Description détaillée du diagramme de séquences de l'authentification

I.2.3. Diagramme de séquences pour l'ajout d'un hôpital

La figure ci-dessous (***figure 15***) est le diagramme de séquences du processus d'ajout d'un nouvel hôpital.

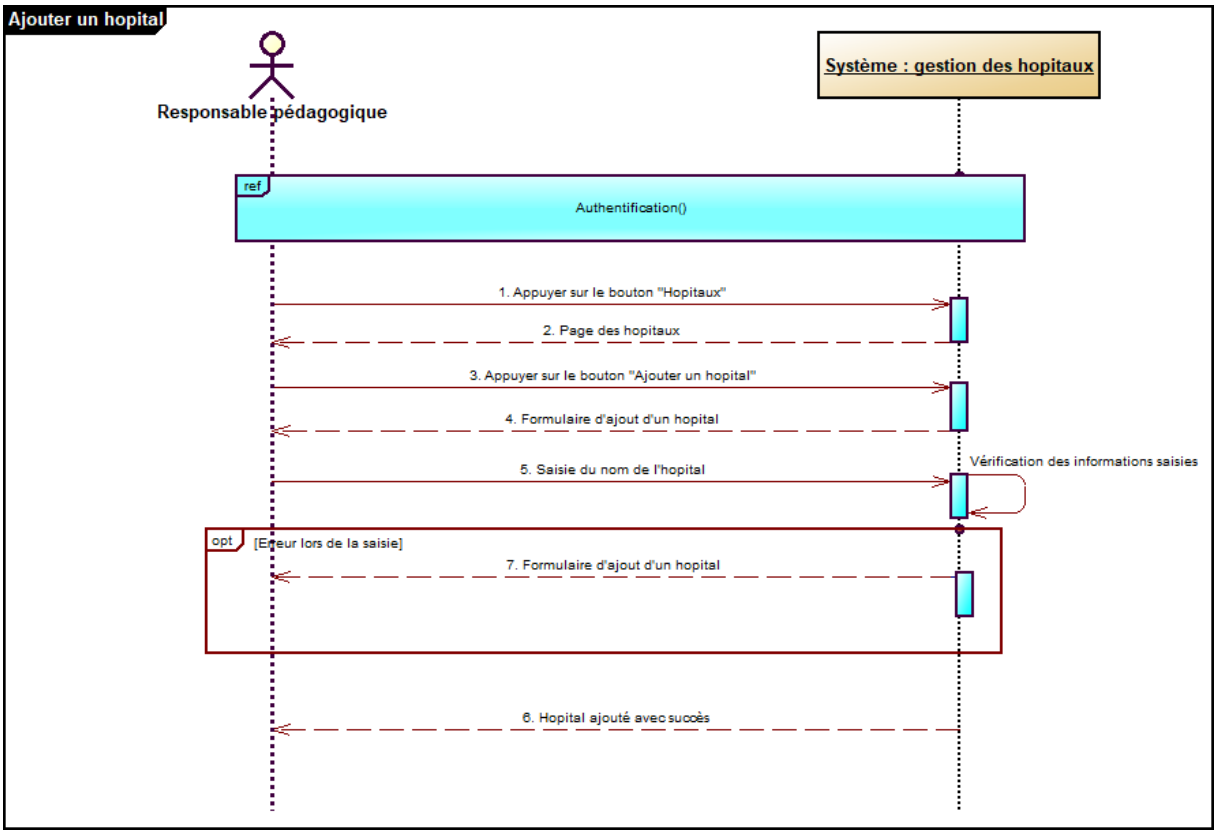

Figure 15 : Diagramme de séquences pour le processus d'ajout d'un hôpital

Le tableau suivant ***(tableau 7)*** donne une description explicite de la figure 15

Scénario principal	
0. Authentification	1. Appuyer sur le sous-menu «Hôpitaux»
2. Redirection vers la page des hôpitaux	3. Appuyer sur le bouton 'Ajouter un hôpital'
4. Un formulaire d'ajout s'affiche	5. Le responsable pédagogique renseigne un nom d'hôpital
5.1 Le système procède à la vérification des informations saisies	6. Renvoi d'un message de succès 'Hôpital ajouté avec succès' (si les informations sont correctes)
Scénarios secondaires	
S'il y a erreur lors de la saisie	7. Le système renvoie un message d'erreur et on reste sur le même formulaire

Tableau 6 : Description détaillée du diagramme de séquences du processus d'ajout d'un hôpital

II. Conception du Système

Dans la démarche de Processus Unifié, la phase de conception suit immédiatement la phase d'Analyse, par ailleurs la conception de logiciel est un art qui nécessite de l'expérience, et elle consiste à traduire les besoins en spécifiant comment l'application pourra les satisfaire avant de procéder à sa réalisation[19].

II.1. Conception générale

II.1.1. Architecture physique du système

Dans le domaine informatique, l'architecture physique (également nommée architecture technique) décrit l'ensemble des composants matériels supportant l'application[20]. Ces composants peuvent être

- des calculateurs (ou serveurs matériels)
- des postes de travail
- des équipements de stockage (baie de stockage, SAN)
- des équipements de sauvegarde

Dans le vaste éventail des modèles d'architectures physiques pour le développement d'applications, l'exploration dépasse les frontières des schémas conventionnels. Parmi ces modèles, émerge l'architecture 2-tiers.

En effet dans une architecture deux tiers, encore appelée client-serveur de première génération ou client-serveur de données, le poste client se contente de déléguer la gestion des données à un service spécialisé. Le cas typique de cette architecture est une application de gestion fonctionnant sous Windows ou Linux et exploitant un SGBD centralisé[21].

Ce type d'application permet de tirer parti de la puissance des ordinateurs déployés en réseau pour fournir à l'utilisateur une interface riche, tout en garantissant la cohérence des données, qui restent gérées de façon centralisée.

La gestion des données est prise en charge par un SGBD centralisé, s'exécutant le plus souvent sur un serveur dédié. Ce dernier est interrogé en utilisant un langage de requête qui, le plus souvent, est SQL. Le dialogue entre client et serveur se résume donc à l'envoi de requêtes et au retour des données correspondant aux requêtes.

La figure suivante ***(figure 16)*** présente le schéma d'une architecture 2-tiers

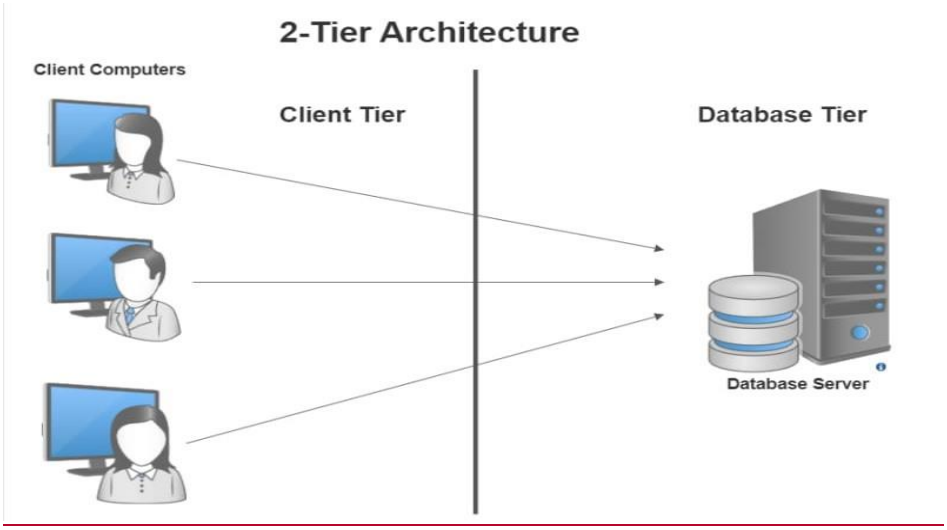

Figure 16 : Architecture 2-tiers

Nous avons aussi l'architecture 3-tiers, également baptisée architecture à trois niveaux qui est une architecture d'application logicielle bien établie qui organise les applications dans trois niveaux informatiques logiques et physiques : le niveau Présentation, ou interface utilisateur, le niveau Application, où les données sont traitées et le niveau Données, où les données associées à l'application sont stockées et gérées[22].

La figure suivante ***(figure 17)*** montre le schéma de l'architecture 3-tiers

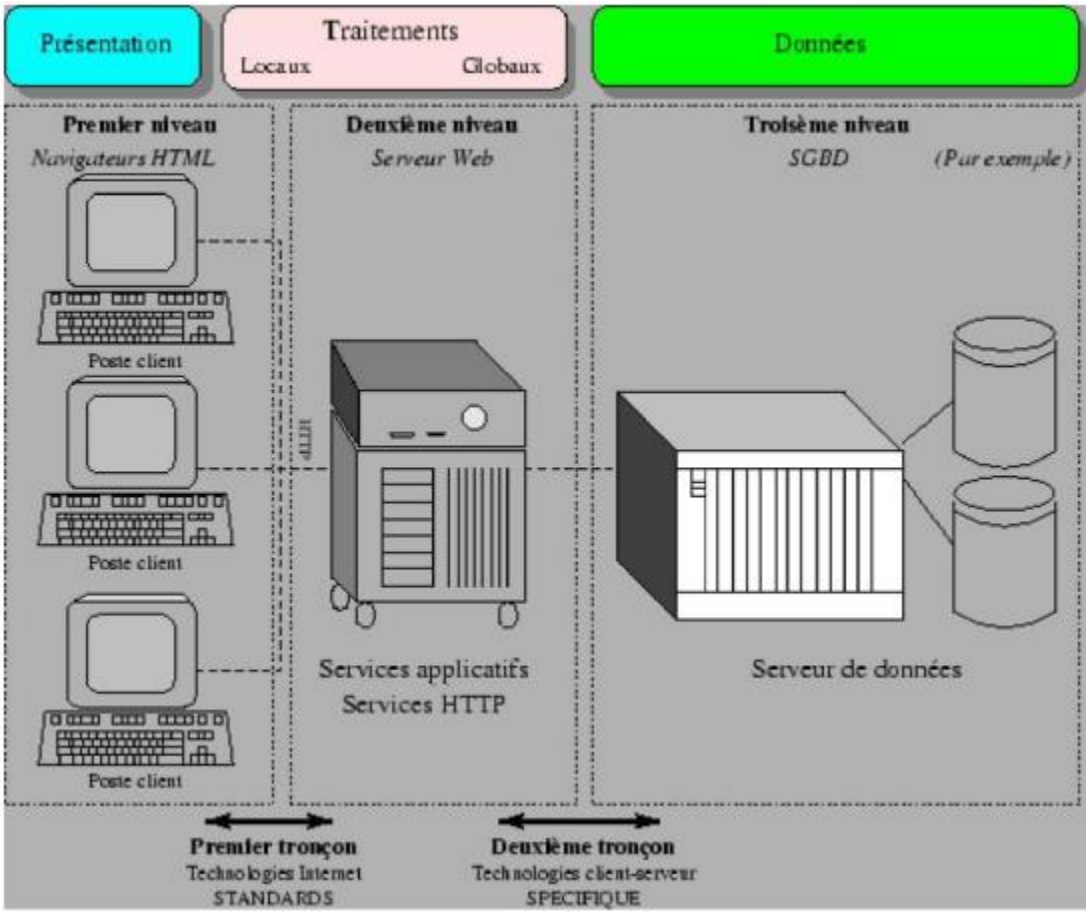

Figure 17 : Architecture 3-tiers*[21]*

II.1.2. Adoption de l'architecture 3-tiers : Justification

L'orientation vers l'architecture 3-tiers n'est pas le fruit du hasard, mais plutôt une décision stratégique dictée par des considérations spécifiques visant à optimiser le développement et la pérennité de notre application.

Les justifications de ce choix sont aussi robustes que les trois couches qui composent cette architecture éclairée[22]:

- **Développement plus rapide** : comme chaque niveau peut être développé simultanément par des équipes différentes, une organisation peut mettre l'application sur le marché plus rapidement, et les programmeurs peuvent utiliser les meilleurs langages et outils les plus récents pour chaque niveau.
- **Évolutivité accrue :** n'importe quel niveau peut être mis à l'échelle indépendamment des autres selon les besoins.

- **Fiabilité accrue** : une indisponibilité dans un niveau est moins susceptible d'avoir un impact sur la disponibilité ou les performances des autres niveaux.
- **Sécurité renforcée** : comme les niveaux Présentation et Données ne peuvent pas communiquer directement, un niveau Application bien conçu peut fonctionner comme une sorte de pare-feu interne, empêchant les injections SQL et d'autres exploits malveillants.

II.1.3. Architecture logique du système

La phase de conception logique du cycle de vie de la solution consiste à définir une architecture logique qui décrit les relations existantes entre les composants logiques de la solution. Cette architecture, associée à l'analyse d'utilisation établie lors de la phase de spécification technique, constitue un scénario de déploiement, qui sert de base pour la phase de conception du déploiement[23].

Naviguer parmi les diverses architectures logicielles demande une évaluation minutieuse des avantages, des inconvénients et de l'adéquation à notre contexte spécifique. Chacune de ces architectures représente une philosophie de conception. Voici quelques-unes des architectures logicielles considérées, chacune avec ses caractéristiques distinctives :

- **Monolithique :** Dans une architecture monolithique, tous les éléments de l'application (de l'interface utilisateur et de la logique métier au code d'accès aux données) sont construits et regroupés dans une base de code et un référentiel unique. Cette architecture utilise généralement des concepts tels que des modèles/thèmes et le modèle de conception Model-View-Controller (MVC)[24].

La figure ***(figure 18)*** ci-après montre la structure d'une architecture monolithique

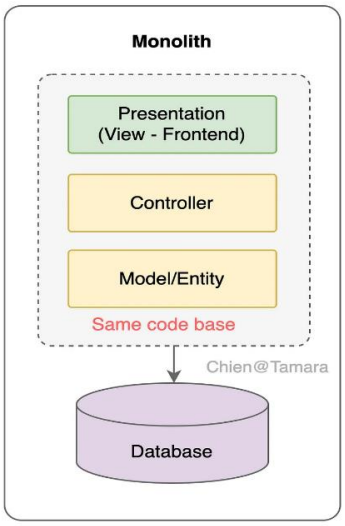

Figure 18 : Architecture monolithique*[24]*

- **Micro-services :** Une architecture de microservices est un type d'architecture d'application dans laquelle l'application est développée sous la forme d'un ensemble de services. Elle fournit le framework permettant de développer, déployer et gérer de manière indépendante des diagrammes et des services d'architecture de microservices[25].

Ci-dessous le schéma ***(figure 19)*** d'un micro-service

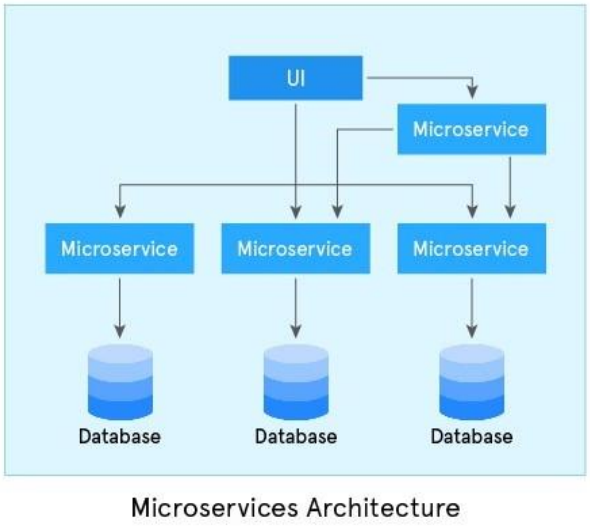

Figure 19 : Architecture micro-services*[26]*

- **Architecture Orientée Événements :** L'architecture pilotée par les événements fonctionne selon le principe suivant : elle produit, détecte et agit selon les événements du système pertinents pour les utilisateurs. Si aucun événement ne se produit, rien ne se passe dans le système. Cette architecture définit ces événements indispensables comme des déclencheurs. Lorsque ces déclencheurs s'activent, ils provoquent des comportements spécifiques, comme l'envoi d'une alerte lorsque le prix de l'article que vous voulez baisse[27].

La capture ***(figure 20)*** suivante illustre l'architecture orientée Evénement

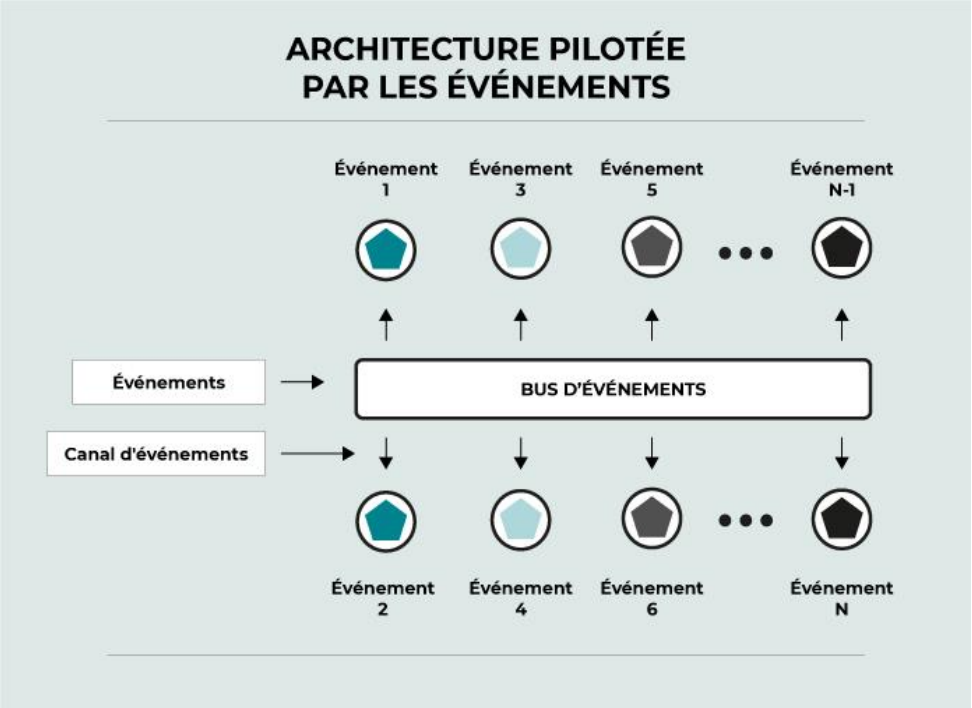

Figure 20 : Architecture Orientée Evénement *[27]*

II.1.4. Adoption de l'architecture monolithique : Justification

Étant donné la relative simplicité et la taille réduite de notre application de gestion des stages, ainsi que notre équipe de développement restreinte composée de seulement deux membres, nous avons pris la décision stratégique de développer notre module sous forme de services et d'adopter une architecture monolithique pour chaque composante de l'application. Cette approche, avec le modèle en couches pour le back-end et le modèle MVC pour le front-end, offre une mise en œuvre rapide et nécessite des investissements minimaux, ce qui la rend particulièrement adaptée à nos besoins.

Avec le modèle en couches, chaque couche dialogue avec une autre au travers d'un contrat d'interface. Par convention, la couche de données est la couche la plus basse et la couche de présentation la couche la plus haute. Chaque couche expose alors des services à la couche supérieure[28] comme le montre la figure ***(figure 21)*** ci-après.

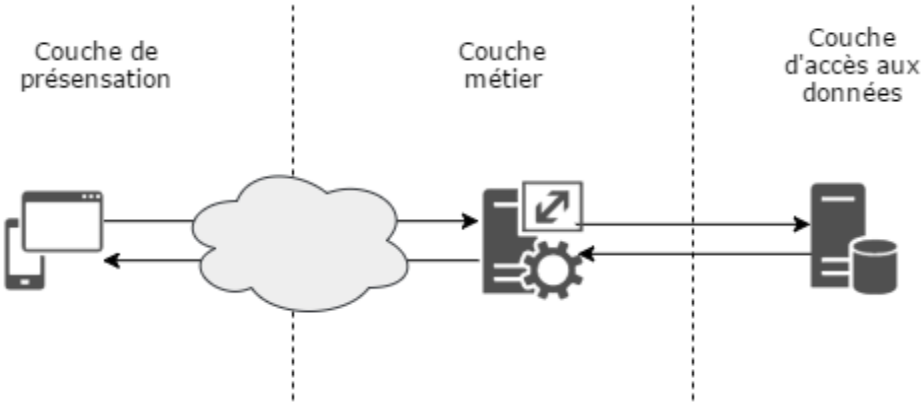

Figure 21 : Architecture en couches*[28]*

Concernant le patron d'architecture logicielle **modèle-vue-contrôleur** (en abrégé **MVC**, en anglais *model-view-controller*), tout comme les patrons **modèle-vue-présentation** ou **présentation, abstraction, contrôle**, il est un modèle destiné à répondre aux besoins des applications interactives en séparant les problématiques liées aux différents composants au sein de leur architecture respective[29].

Ce paradigme regroupe les fonctions nécessaires en trois catégories :

- un **modèle** (modèle de données)
- une **vue** (présentation, interface utilisateur)
- un **contrôleur** (logique de contrôle, gestion des événements, synchronisation).

La figure ***(figure 22)*** suivante montre les différentes interactions de l'architecture MVC.

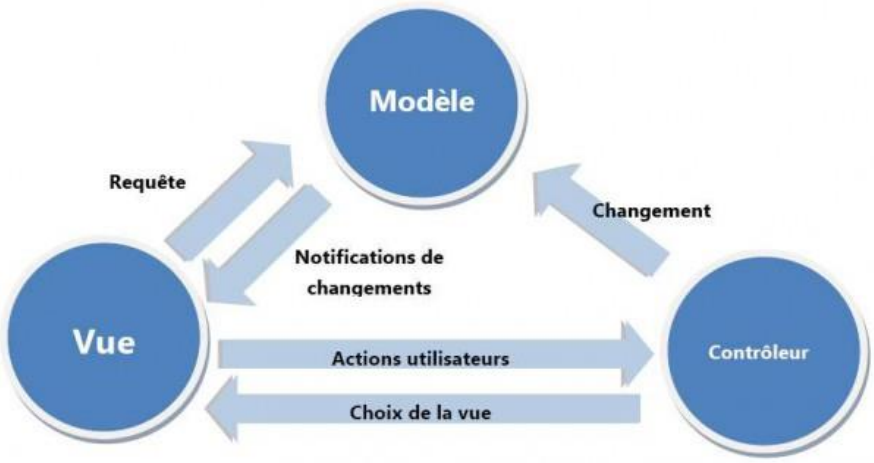

Figure 22 : Architecture MVC*[29]*

La figure ci-dessous (***figure 23***) représente l'architecture globale de notre système.

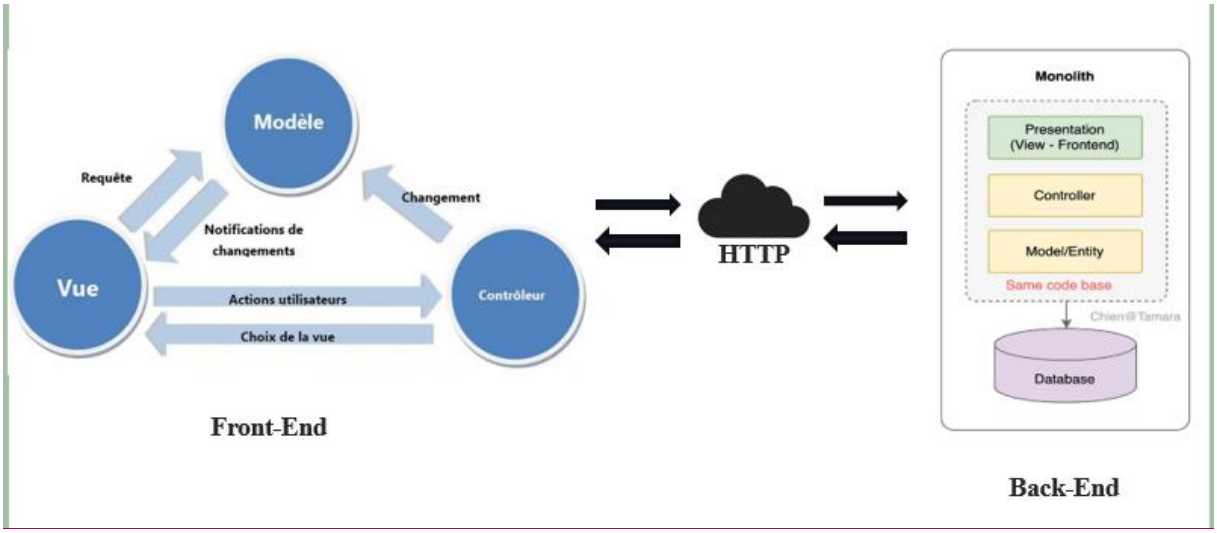

Figure 23 : Architecture globale du système

II.2. Conception détaillée

La conception détaillée est un ensemble des activités consistant à fragmenter les résultats de la conception architecturale jusqu'à un niveau de granularité permettant le codage du logiciel[30].

Dans cette section, nous proposerons un certain nombre de diagrammes de classes portant sur la gestion des hôpitaux et la gestion des modules.

II.2.1. Diagramme de classes

Les diagrammes de classes sont l'un des types de diagrammes UML les plus utiles, car ils décrivent clairement la structure d'un système particulier en modélisant ses classes, ses attributs, ses opérations et les relations entre ses objets[31].

II.2.2. Diagramme de classes pour la gestion des hôpitaux

La figure ***(figure 24)*** ci-dessous nous montre le diagramme des classes du sous-module de la gestion des hôpitaux

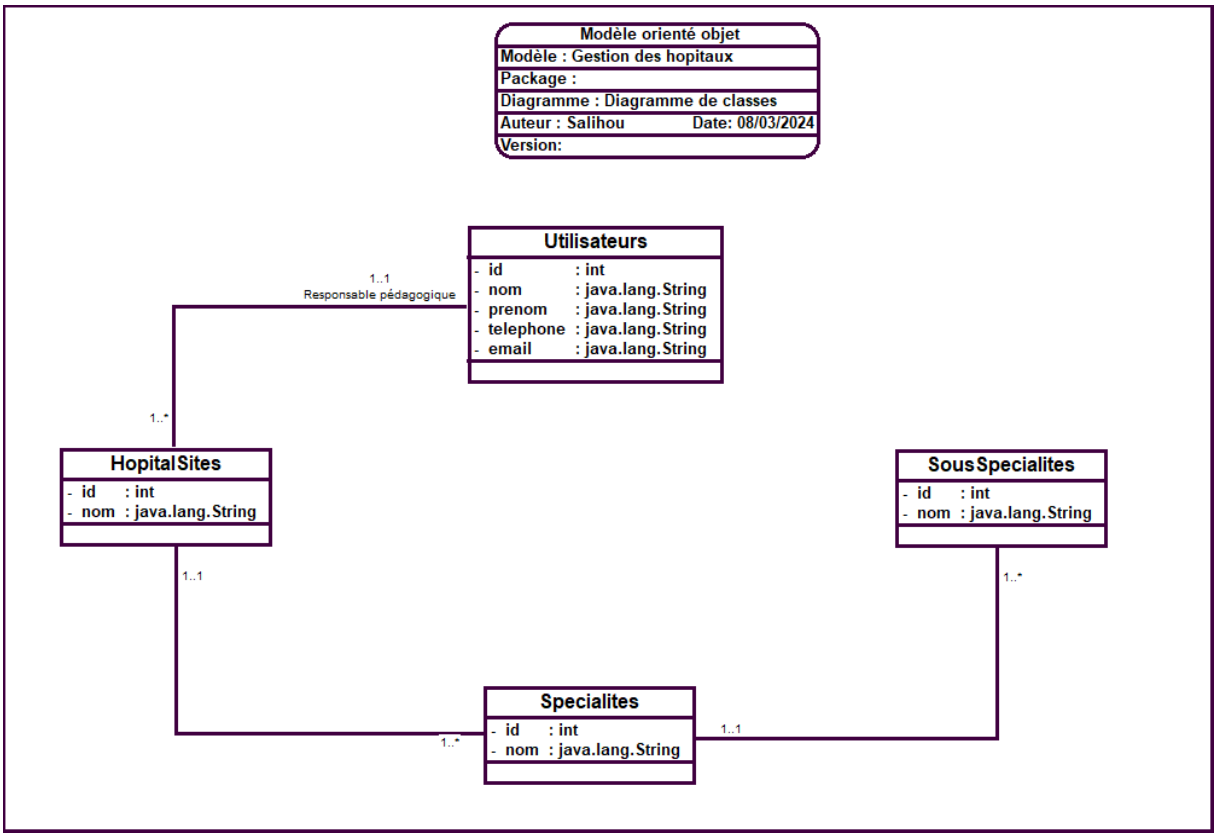

Figure 24 : Diagramme de classes pour la gestion des hôpitaux

Le tableau ci-dessous ***(tableau 8)*** décrit le diagramme des classes de la figure ci-dessus ***(figure 23)***

Classe	Description de la classe
Utilisateurs	Cette classe rassemble les informations partagées par tous les utilisateurs de notre système, notamment le responsable pédagogique.
HopitalSites	Cette classe encapsule les détails des hôpitaux créés par le responsable pédagogique dans notre système.
Specialites	Cette classe contient les informations relatives aux différentes spécialités (médicale ou chirurgicale) disponibles dans les hôpitaux.
SousSpecialites	Cette classe englobe les données concernant les divers services qui font partie des spécialités des hôpitaux.

Tableau 7 : Description du diagramme de classes de la gestion des hôpitaux

II.2.3. Diagramme de classes pour la gestion des modules

Ici nous présentons le diagramme des classes du sous module de gestion des modules

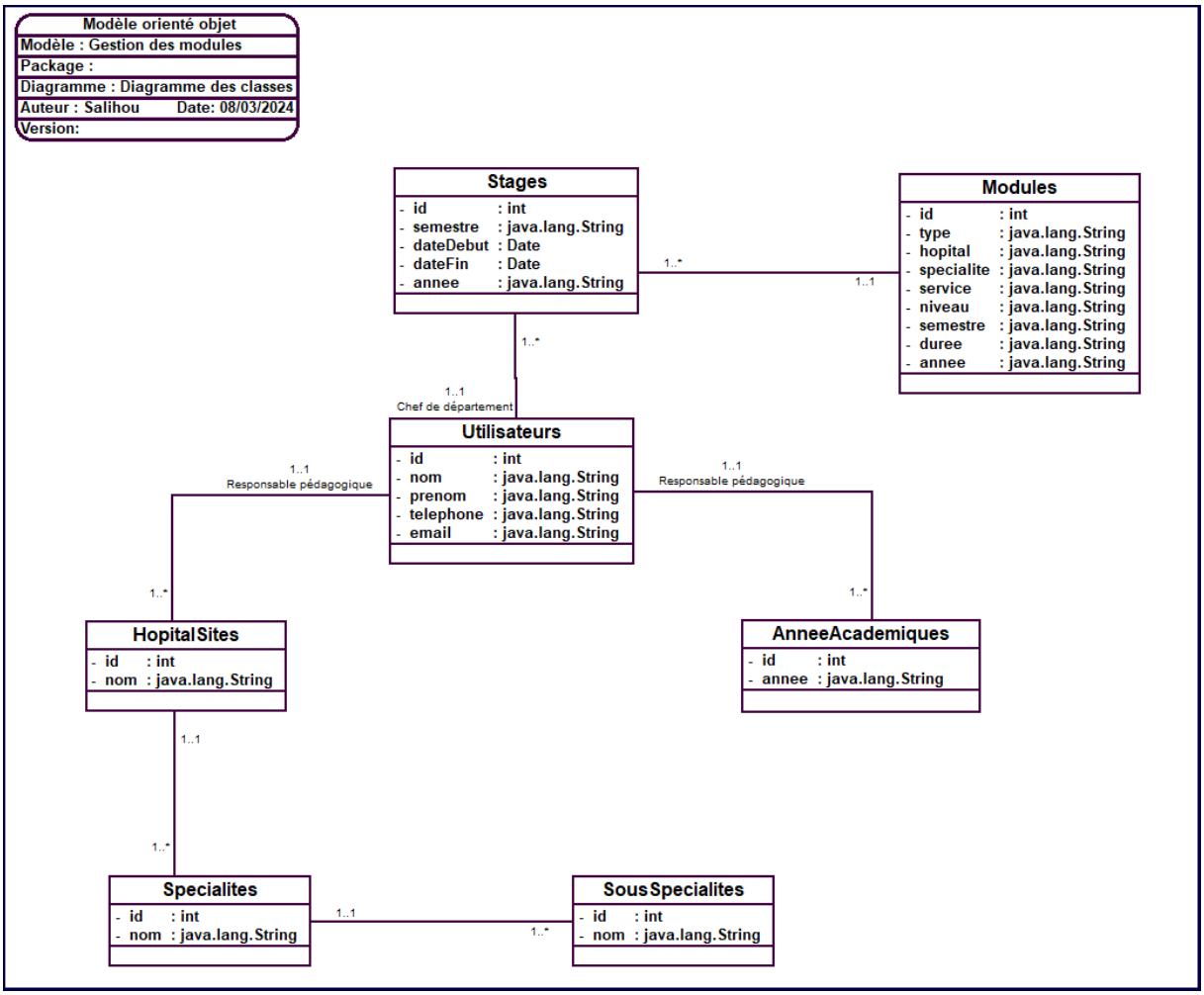

Figure 25 : Diagramme de classes pour la gestion des modules

A présent passons à la description de la figure ***(figure 25)*** dans le tableau ***(tableau 8)*** suivant :

Classe	Description de la classe
Utilisateurs	Cette classe regroupe les informations partagées par tous les utilisateurs du système, incluant le responsable pédagogique et le chef de département.
HopitalSites	Cette classe revêt une importance particulière car elle représente les hôpitaux où des modules sont ouverts dans notre système.
Specialites	Chaque module ouvert est affilié à une spécialité (Médicale ou Chirurgicale) existante dans un hôpital sélectionné, justifiant ainsi l'existence de cette classe.
SousSpecialites	Cette classe permet l'ouverture de modules, chaque module étant lié à une sous-spécialité ou service spécifique..
AnneeAcademiques	Cette classe contient les données relatives à l'année académique en cours, créées par le responsable pédagogique pour organiser les activités du programme.

Stages	Cette classe concerne les informations sur les stages semestriels pour ne pas dire le semestre, démarrés par le responsable pédagogique.
Modules	Cette classe contient les détails des modules ouverts, créés par le chef de département pour structurer le programme d'études.

Tableau 8 : Description du diagramme de classes de la gestion des modules

Conclusion

Ce chapitre débute par une analyse approfondie des besoins, illustrée à travers des diagrammes d'activités et des diagrammes de séquences. Par la suite, lors de la conception générale, sont définies avec rigueur les architectures physique et logique du système. Enfin, la phase de conception détaillée permet de cristalliser les idées à travers la création et la description minutieuse de divers diagrammes de classes, représentant certains modules exposés précédemment. Ces étapes méthodiques conduisent à élaborer un modèle de base de données robuste, ouvrant ainsi la voie à l'implémentation de l'application dans la section suivante.

Chapitre V : Implémentation et Présentation de l'application

Introduction

L'implémentation est la réalisation, l'exécution ou la mise en pratique d'un plan, d'une méthode ou bien d'un concept, d'une idée, d'un modèle, d'une spécification, d'une norme ou d'une règle dans un but précis. L'implémentation est donc l'action qui doit suivre une réflexion pour la concrétiser.

Dans le contexte des technologies de l'information, l'implémentation d'un logiciel ou matériel englobe tous les processus impliqués dans le bon fonctionnement d'un élément dans son environnement[32].

Dans ce chapitre, une présentation des outils choisis pour mettre en œuvre l'application est d'abord effectuée. Ensuite, les processus de test sont explorés, et quelques-unes des interfaces essentielles de l'application sont mises en avant pour conclure.

I. Présentation des outils de développement

Puisque notre application repose sur une architecture client-serveur, avec le client en front-end et le serveur en back-end, nous avons opté pour la technologie **MERN (M**ySQL**, E**xpress**, R**eact**, N**ode**)**.

Dans cette section, nous allons introduire la technologie que nous avons choisie pour développer notre application, ainsi que l'Environnement de Développement Intégré (IDE), VSCode, en expliquant les raisons derrière nos choix respectifs.

I.1. La technologie MERN : Présentation

La technologie MERN est un acronyme représentant un ensemble de technologies (à la fois côté serveur et côté client) permettant de développer des applications web full stack[33]. Voici les 4 technologies utilisées par cette stack :

- **M**ySQL
- **E**xpress
- **R**eact

- Node

I.1.1. MySQL

MySQL est un système de gestion de base de données relationnelles (SGBDR) basé sur SQL (Structured Query Language)[34].

Il fonctionne sur pratiquement toutes les plates-formes, y compris Linux, Unix et Windows. Il est entièrement multi-thread avec un noyau de threads, et fournit des API (Application Programming Interface) pour de nombreux langages de programmation, notamment C, C + +, Eiffel, Java, Perl, PHP, Python, et Tcl.

Il est utilisé dans une large gamme d'applications, Le commerce électronique, les bases de données Web...

MySQL bénéficie d'un large public, car :

- **Il est facile à comprendre :** Sa syntaxe simple en fait un langage facile à comprendre pour les programmeurs et des débutants.
- **Le Langage est fonctionnel :** MySQL fonctionne sur de nombreuses plates-formes différentes.
- **Dispose d'une vaste bibliothèque de fonctions et d'API :** API pour C, C + +, Eiffel, Java, Perl, PHP, Python, Ruby et Tcl sont disponibles. Les fonctions SQL sont mises en place en utilisant une bibliothèque de classes optimisées.
- **Multi Thread :** Complètement multi-thread utilisant un noyau de threads.
- **Haute capacité de storage :** Pour vous donner une idée : De grosses entreprises actuelles utilisent le serveur MySQL avec plus de 100 000 tables et 1 000 000 000 d'enregistrements.

I.1.2. Node

Node.js est un environnement d'exécution JavaScript open source et multiplateforme qui se concentre sur les applications côté serveur et réseau[35].

Node.js permet aux développeurs de créer des applications réseau rapides et évolutives à l'aide de code facile à comprendre. Il s'exécute sous Windows OS, Mac OSX, Linux, Unix et d'autres

systèmes d'exploitation. Il prend en charge les processeurs ARM, tels que Raspberry Pi ou Beaglebone Black.

L'idée derrière Node a été conçue début 2009 par Ryan Dahl. Il avait déjà développé Web Machine. C'était la première structure d'application Web écrite dans le navigateur Chrome de Google.

Node.js est une technologie back-end qui permet aux développeurs d'utiliser JavaScript côté serveur de leurs applications. On peut l'utiliser pour des tâches telles que la collecte de données à partir de capteurs, contrôle de votre thermostat domestique.

Cependant, Node propose également des modules et des packages spécialement conçus pour le développement frontal. Vous pouvez ainsi créer des applications Web à page unique à l'aide de structures telles que ReactJS.

I.1.3. React

React n'est pas à proprement parler un framework mais se présente comme une bibliothèque JavaScript pour créer des interfaces utilisateurs. React est la réponse de Facebook à un problème récurrent : créer des interfaces réutilisables[36].

Tout d'abord, React est basé sur virtual-dom : un composant React ne crée pas de HTML mais une représentation sous forme d'objets et de nœuds de ce à quoi le HTML final doit ressembler. Virtual-dom va prendre en compte cette représentation, la comparer au DOM réel et en déduire les opérations minimales à exécuter pour que le DOM réel soit conforme au virtuel. C'est grâce à cet outil que React peut partir du principe qu'il est plus simple de "remplacer" toute l'interface quand elle doit être modifiée plutôt que de modifier au fur et à mesure le DOM comme jQuery ou AngularJS pouvaient le faire.

L'intérêt de cette approche est assez simple. On reproche souvent à JavaScript d'être lent alors que c'est DOM qui l'est. Avoir une représentation sous forme d'arbre en JavaScript permet de réaliser beaucoup plus d'opérations, d'utiliser les meilleurs algorithmes de comparaison d'arbres et, cerise sur le gâteau, de faire toutes les modifications du DOM en une opération plutôt qu'au fur et à mesure. Virtual-dom est également bien plus facile à mettre à jour et à améliorer que les différentes implémentations de DOM dans les navigateurs.

I.1.4. Express

Express.js, parfois aussi appelé « Express », est un framework backend Node.js minimaliste, rapide et de type Sinatra qui offre des fonctionnalités et des outils robustes pour développer des applications backend évolutives. Il vous offre le système de routage et des fonctionnalités simplifiées pour étendre le framework en développant des composants et des parties plus puissants en fonction des cas d'utilisation de votre application[37].

Ce framework fournit un ensemble d'outils pour les applications web, les requêtes et les réponses HTTP, le routage et les intergiciels permettant de créer et de déployer des applications à grande échelle, prêtes pour l'entreprise.

Il fournit également un outil d'interface de ligne de commande (CLI) appelé Node Package Manager (NPM), où les développeurs peuvent s'approvisionner en paquets développés. Il oblige également les développeurs à suivre le principe Don't Repeat Yourself (DRY).

Le principe DRY vise à réduire la répétition des modèles logiciels, en les remplaçant par des abstractions, ou en utilisant des normalisations de données pour éviter la redondance.

I.2. La technologie MERN : Fonctionnement

Le Stack MERN suit une séparation claire entre le front-end et le back-end, permettant aux développeurs de travailler indépendamment des deux côtés tout en assurant une communication fluide entre eux.

Voici comment fonctionnent tous les composants du Stack MERN :

- Le front-end est construit à l'aide de React.js. Il gère toute la logique côté client et rend l'application web interactive. Il envoie des requêtes HTTP au back-end pour récupérer, ajouter et manipuler des données.
- Express.js, s'exécutant sur Node.js, sert de back-end. Il gère les requêtes HTTP entrantes depuis le front-end, communique avec la base de données MySQL pour récupérer ou stocker des données, et envoie des réponses appropriées.
- MySQL stocke les données de l'application. Il fournit une solution de base de données qui stocke les données tables.

- Node.js agit comme intermédiaire entre le front-end et la base de données MySQL. Il reçoit les demandes du front-end, les traite, interagit avec la base de données et renvoie les données nécessaires.

I.3. La technologie MERN : Justification

La technologie MERN, constituée de MySQL, Express.js, React et Node.js, offre un ensemble complet d'outils pour le développement d'applications web modernes.

Voici quelques-unes des raisons qui justifient le choix de la stack MERN pour la création d'applications évolutives et réactives[38]:

- JavaScript Full-Stack : Le Stack MERN permet aux développeurs d'utiliser un seul langage de programmation pour toute l'application, que ce soit le front-end ou le back-end. Cela donne aux développeurs JavaScript un superpouvoir.
- Réutilisabilité : Avec l'architecture basée sur les composants de React, les développeurs appliquent l'un des principes les plus importants du développement, le DRY (Ne vous répétez pas). Avec React, les développeurs peuvent créer des éléments d'interface utilisateur réutilisables, réduisant ainsi la répétition et économisant du temps.
- Applications en temps réel : L'architecture orientée événements de Node.js permet la création d'applications en temps réel, comme les applications de chat et les outils collaboratifs. Vous pouvez utiliser des bibliothèques comme 'socket.io' pour cela.
- Scalabilité : La capacité de MySQL à gérer de grandes quantités de données, combinée à l'I/O non bloquant de Node.js, rendent ce stack adapté à la création d'applications évolutives.

I.4. Outil de modélisation

Un outil de modélisation, dans le contexte du développement logiciel ou de la conception de systèmes, est une application informatique spécialement conçue pour faciliter la création, la visualisation, et la modification de modèles. Ces modèles représentent généralement des aspects abstraits ou concrets d'un système, tels que sa structure, son comportement, ou ses interactions.

PowerAMC, l'outil élu pour la conception de nos organigrammes et de tous les diagrammes qui enrichissent ce système, se présente comme une baguette magique dans le royaume complexe

de la modélisation. Voici un aperçu de cet outil avant que nous ne plongions dans la justification éclairée de notre choix :

I.4.1. PowerAmc : Présentation

PowerAMC est un logiciel de modélisation. Il permet de modéliser les traitements informatiques et leurs bases de données associées. Créé par SDP sous le nom AMC*Designor, racheté par Powersoft, ce logiciel est produit par Sybase depuis le rachat par cet éditeur en 1995. Hors de France, la version internationale est commercialisée par Sybase sous la marque **PowerDesigner**[39]**.**

PowerAMC offre la possibilité de réaliser divers types de modèles informatiques, parmi lesquels on peut citer :

- Modèle Conceptuel de Données (MCD)
- Modèle Libre (MLB)
- Modèle Orienté Objet (MOO)
- Modèle Physique de Données (MPD)
- Modèle XML (MSX)

I.4.2. PowerAmc : Justification de ce choix

Le recours à PowerAMC pour la modélisation de nos organigrammes et de l'ensemble des diagrammes présentés dans ce document découle d'une sélection réfléchie motivée par plusieurs avantages[40] :

- Power AMC est un outil simple à utiliser. Le déploiement d'un poste suffit à rendre l'outil efficient.
- L'outil fonctionne nativement avec tous les SGBD courants du marché (ORACLE, SQL SERVEUR, DB2/UDB).
- L'outil permet une documentation des développements.
- L'outil permet une rétro-documentation de l'existant.

L'outil génère des graphiques exportables et importables facilement via un format XML.

I.5. Environnement de Développement Intégré (IDE)

Un environnement de développement intégré, ou IDE, est un logiciel de création d'applications, qui rassemble des outils de développement fréquemment utilisés dans une seule interface utilisateur graphique (GUI)[41].

Parmi les Environnements de Développement Intégré (IDE) couramment utilisés, on peut mentionner Visual Studio Code, PyCharm, Eclipse, IntelliJ IDEA, et bien d'autres encore. Notre choix s'est porté sur Visual Studio Code, et nous exposerons les raisons de ce choix après l'avoir présenté.

I.5.1. Visual Studio Code : Présentation

Visual Studio Code est un éditeur de code source qui peut être utilisé avec une variété de langages de programmation, notamment Java, JavaScript, Go, Node.js et C++[42].

Il est présenté lors de la conférence des développeurs Build d'avril 2015 comme un éditeur de code multiplateforme, open source et gratuit.

Il est basé sur Electron, une structure utilisée pour déployer des applications Node.js pour le bureau exécuté sur le moteur Blink. Bien qu'il utilise le framework Electron, le logiciel n'utilise pas Atom mais utilise le même composant éditeur (nommé Monaco) utilisé dans Azure DevOps (anciennement appelé Visual Studio Online et Visual Studio Team Services).

I.5.2. Visual Studio Code : Justification

Dans cette section, nous allons examiner les raisons qui nous ont poussé à choisir Visual Studio Code comme environnement de développement pour notre projet.

Visual Studio Code (VSCode) offre plusieurs avantages qui en font un choix populaire parmi les développeurs :

- **Léger et rapide :** VSCode est connu pour sa légèreté et sa rapidité d'exécution, ce qui en fait un choix idéal pour les environnements de développement.
- **Personnalisable :** Il offre une large gamme d'extensions et de thèmes personnalisables, permettant aux développeurs de l'adapter à leurs besoins spécifiques.
- Intégration Git native : VSCode intègre nativement des fonctionnalités Git, ce qui facilite la gestion et la collaboration sur les projets Git.

- **Débogage intégré :** Il prend en charge le débogage intégré pour plusieurs langages de programmation, ce qui simplifie le processus de débogage des applications.
- **Auto complétion intelligente :** VSCode offre une fonctionnalité d'autocomplétion intelligente qui suggère des mots-clés, des fonctions et des variables pendant la saisie du code, ce qui améliore la productivité des développeurs.
- **Intégration avec les outils de développement populaires :** Il offre une intégration transparente avec une variété d'outils de développement populaires, tels que Node.js, TypeScript, Python, etc.
- **Grande communauté et support actif :** VSCode bénéficie d'une large communauté d'utilisateurs et de développeurs, ce qui signifie qu'il existe de nombreuses ressources, tutoriels et extensions disponibles pour aider les utilisateurs.

En résumé, les avantages de Visual Studio Code résident dans sa légèreté, sa personnalisation, son intégration avec les outils de développement populaires, son support Git intégré et son auto complétion intelligente, ce qui en fait un choix attrayant pour de nombreux développeurs.

I.6. Outil de collaboration

Un outil collaboratif ou plateforme collaborative désigne une solution destinée au travail à distance ou en équipe. Il permet de partager, de traiter et de gérer des documents, des fichiers et différents types de données entre divers utilisateurs[43].

Étant donné que notre équipe est constituée de deux développeurs, et compte tenu de l'importance du développement collaboratif dans la création de logiciels modernes, il était impératif de privilégier cette approche. C'est ainsi que nous avons choisi d'utiliser l'outil de collaboration Git.

I.6.1. Git : Définition

Git est un système de contrôle de version qui a été inventé et développé par **Linus Torvalds**, également connu pour l'invention du noyau Linux, en 2005.

Il s'agit d'un outil de développement qui aide une équipe de développeurs à gérer les changements apportés au code source au fil du temps. Les logiciels de contrôle de version gardent une trace de chaque changement apporté au code dans un type spécial de base de données. Git est le

plus connu des **VCS (V**ersionning **C**ontrol **S**ystem**),** c'est un projet open source très puissant qui est utilisé par l'ensemble de la communauté des développeurs[44].

I.6.2. Git : Justification

Dans cette partie, nous discuterons des raisons fondamentales qui ont conduit à l'adoption de Git pour la gestion de notre code source[45].

- **Workflow de branche par fonctionnalité :** Les capacités de branching de Git constituent l'un de ses plus grands avantages. Contrairement aux systèmes de contrôle de version centralisés, les branches Git sont bon marché et faciles à merger. Cela permet d'utiliser le workflow de branche de fonctionnalité populaire auprès de nombreux utilisateurs de Git.
- **Développement distribué :** Dans SVN, chaque développeur obtient une copie de travail qui pointe vers un dépôt centralisé unique. Git, toutefois, est un système de contrôle de version distribué. Au lieu d'une copie de travail, chaque développeur obtient son propre dépôt local, avec un historique complet des commits.
- **Pull requests :** De nombreux outils de gestion du code source tels que Bitbucket améliorent la fonction de base de Git grâce à des pull requests. Une pull request est un moyen de demander à un autre développeur de merger l'une de vos branches dans son dépôt. Cela permet non seulement aux responsables du projet de suivre les changements plus facilement, mais aussi aux développeurs d'engager des discussions sur leur travail avant de l'intégrer au reste du code.
- **Communauté :** Dans de nombreux cercles, Git est devenu le système de contrôle de version attendu pour les nouveaux projets. Si votre équipe utilise Git, vous n'aurez pas à former de nouveaux employés à votre workflow, car ils seront déjà familiarisés avec le développement distribué.
- **Cycle de livraison plus rapide** : Les branches de fonctionnalité, le développement distribué, les pull requests et la communauté stable se traduisent par un cycle de livraison plus rapide. Ces capacités favorisent un workflow Agile où les développeurs sont encouragés à partager plus fréquemment les moindres changements. À leur tour, les changements peuvent être pushés plus rapidement vers le pipeline de déploiement qu'en utilisant les versions monolithiques courantes avec des systèmes de contrôle de version centralisés.

I.7. Outil de sécurité

La sécurité informatique représente les mesures et pratiques mises en place pour protéger les systèmes informatiques**,** les données ou les réseaux contre les accès non autorisés ou les dommages[46].

Dans le cadre de notre système, où l'authentification, l'autorisation et la sécurisation des routes sont des aspects cruciaux, nous avons opté pour l'utilisation de JSON Web Tokens (JWT).

I.7.1. JWT : Définition

Un JSON Web Token est un access token (jeton d'accès) aux normes RFC 7519 qui permet un échange sécurisé de donnée entre deux parties. Il contient toutes les informations importantes sur une entité, ce qui rend la consultation d'une base de données superflue et la session n'a pas besoin d'être stockée sur le serveur (stateless session)[47].

I.7.2. JWT : Justification

L'adoption des JSON Web Tokens (JWT) pour la sécurité de notre application repose sur deux principales raisons[48] :

- **Réduction de la surcharge du serveur :** le grand nombre de données de session ne sont pas stockées côté serveur. Nous pouvons stocker davantage de propriétés utilisateur sur les données de session côté client pour réduire le nombre d'accès à la base de données sans se soucier de la surcharge de mémoire sur le serveur.
- **Evolution facile :** étant donné que les données de session sont stockées côté client, tant que tous les serveurs principaux partagent la même clé privée, tous les serveurs ont la même capacité de vérifier la validité de la session, ce qui présente une bonne solution pour les architectures Micro-Services.

II. Implémentation de l'application

II.1. Implémentation du Back-end

L'implémentation de notre back-end a été précédée de séances hebdomadaires au cours desquelles, en étroite collaboration avec notre maître de stage, nous avons méticuleusement

recueilli les besoins, jetant ainsi les bases du cahier des charges. Une fois ce dernier établi, nous avons entamé la conception et l'implémentation du back-end en créant un dépôt Git. À l'intérieur de ce dépôt, nous avons organisé un dossier nommé "Full-Stack", qui contient spécifiquement un sous-dossier dédié au back-end.

II.2. Implémentation du Front-end

Dans le cadre de l'implémentation de notre front-end, nous avons suivi une démarche analogue à celle du back-end, en ajoutant un nouveau sous-dossier dédié au front-end dans le répertoire "Full-Stack".

En ce qui concerne le design, nous avons téléchargé un template[49] gratuit sur Internet qu'on a personnalisé à notre guise.

III. Tests

En termes simples, les tests de logiciels consistent à étudier les propriétés d'un système logiciel afin de déterminer sa faisabilité pour une tâche ou un objectif donné. Les tests de logiciels vérifient si le logiciel fonctionne comme prévu, dans une grande variété de circonstances différentes[50].

Étant donné que notre application comprend à la fois un back-end et un front-end, nous avons réalisé des tests dans les deux cas.

III.1. Test du Back-end

Pour tester notre Back-end nous avons utilisé **Postman** qui est une plateforme API permettant de créer et d'utiliser des API. Postman simplifie chaque étape du cycle de vie des API et rationalise la collaboration afin que vous puissiez créer de meilleures API plus rapidement[51].

Les API fournissent un moyen structuré permettant à une application d'accéder aux fonctionnalités d'une autre application. Généralement, cette communication s'effectue sur Internet via un serveur API. Une application client (comme une application mobile) envoie une requête au serveur et, une fois la requête traitée, le serveur renvoie une réponse au client[52].

Une requête inclut l'URL du point de terminaison de l'API et une méthode de requête HTTP. La méthode indique l'action que vous souhaitez que l'API effectue. Voici quelques-unes des méthodes les plus courantes :

- **GET** récupère les données d'une API.
- **POST** envoie de nouvelles données à une API.
- **PATCH** et **PUT** mettre à jour les données existantes
- **DELETE** supprime les données existantes.

III.2. Test du Front-end

Pour cette étape, nous avons créé des comptes représentant les profils des utilisateurs pour tester les fonctionnalités de notre front-end.

IV. Présentation de quelques interfaces de l'application

Notre application offre cinq profils utilisateur différents : l'administrateur, le responsable pédagogique, le chef de département, le maître de stage et le secrétaire. Dans cette section, nous vous proposons un aperçu de quelques interfaces utilisateur représentatives de ces différents profils.

IV.1. La page d'accueil de la plateforme

La figure ***(figure 26)*** ci-dessous représente la page d'accueil de la plateforme avec ses différentes rubriques (« Docteurs », « Etudiants » , « Hôpitaux » et « Spécialités ») et ses différents onglets(« Accueil » , « Spécialités », « Apropos » et « Connexion ») qui nous permettent de naviguer d'une page à une autre.

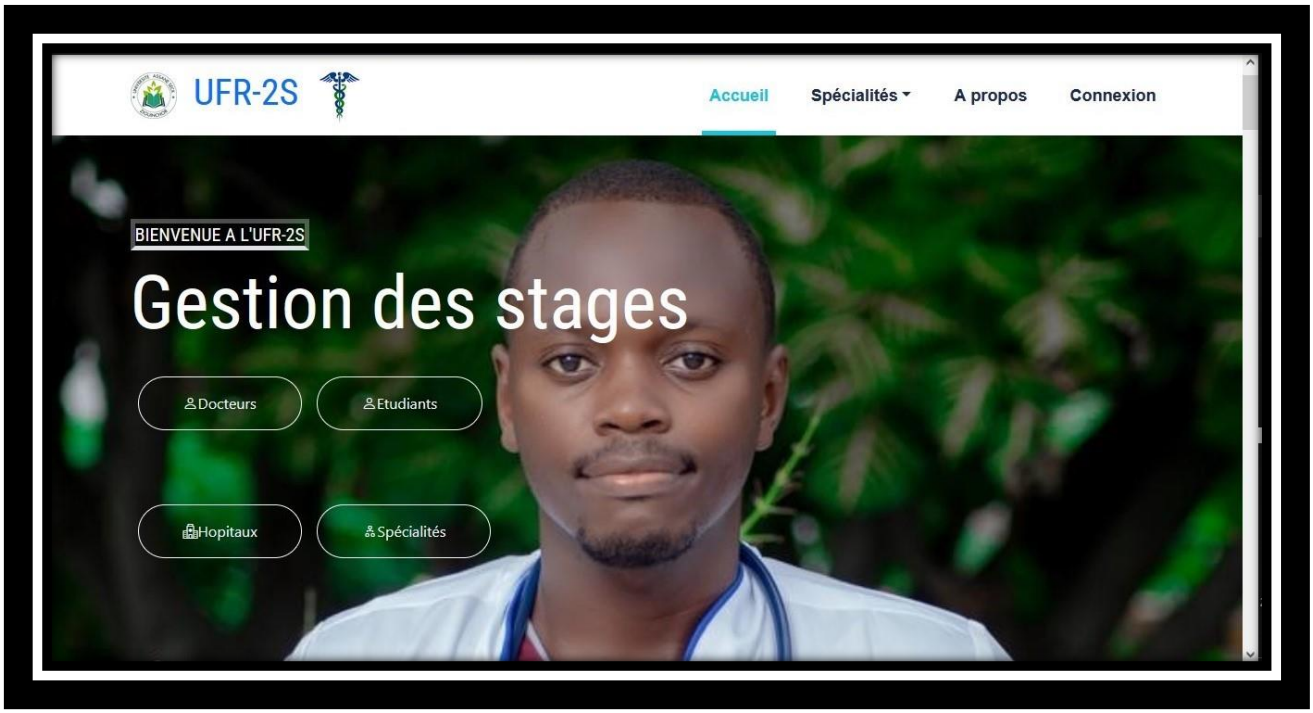

Figure 26 : Page d'accueil de la plateforme

IV.2. La page d'authentification

La figure ***(figure 27)*** ci-après est une capture de la page de connexion qui permet aux utilisateurs de s'authentifier afin qu'ils soient redirigés à leur espace de travail

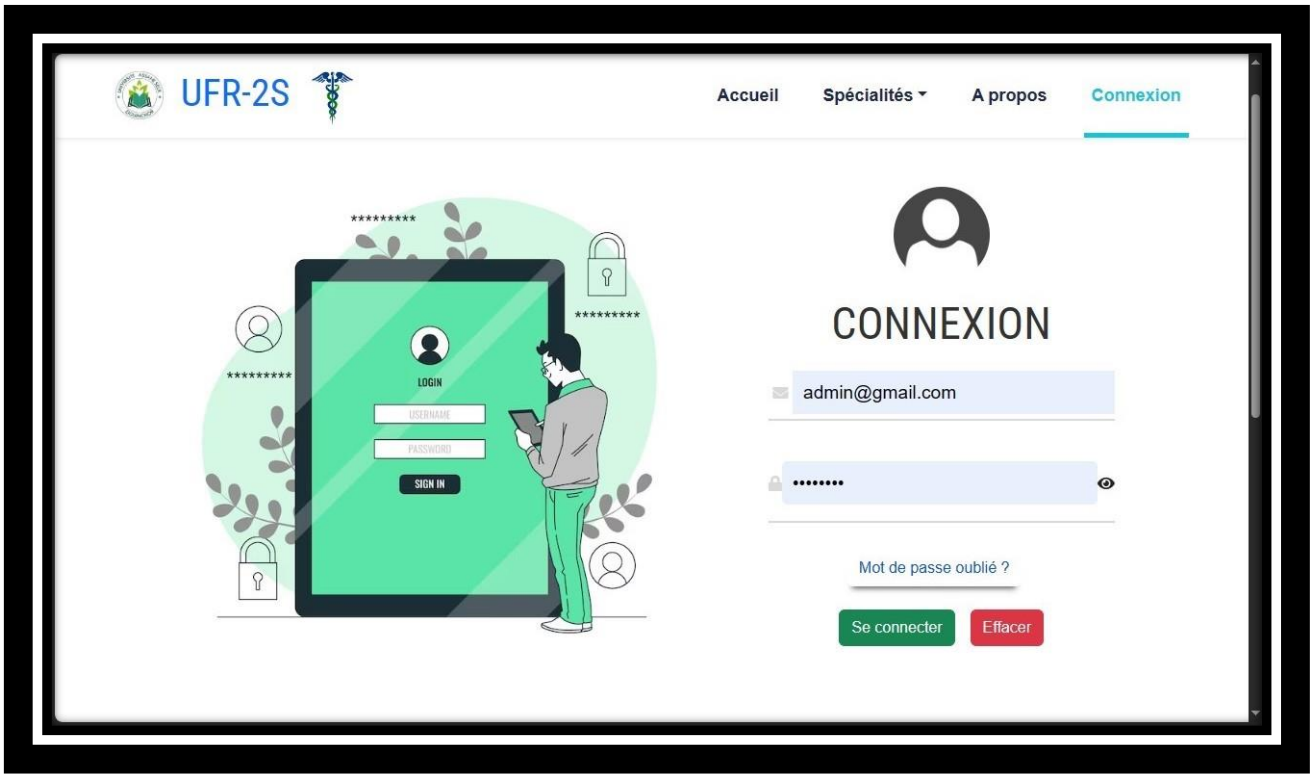

Figure 27:Page d'authentification

IV.3. L'espace administrateur

La figure ***(figure 28)*** suivante nous montre l'espace de travail de l'administrateur qui assure la gestion globale des utilisateurs

Figure 28 : Espace administrateur

IV.4. Liste des utilisateurs

L'image suivante montre la liste des utilisateurs créés par l'administrateur

Figure 29 : Liste des utilisateurs

IV.5. Formulaire de création d'un utilisateur

La figure ***(figure 30)*** montre le formulaire de création d'un utilisateur.

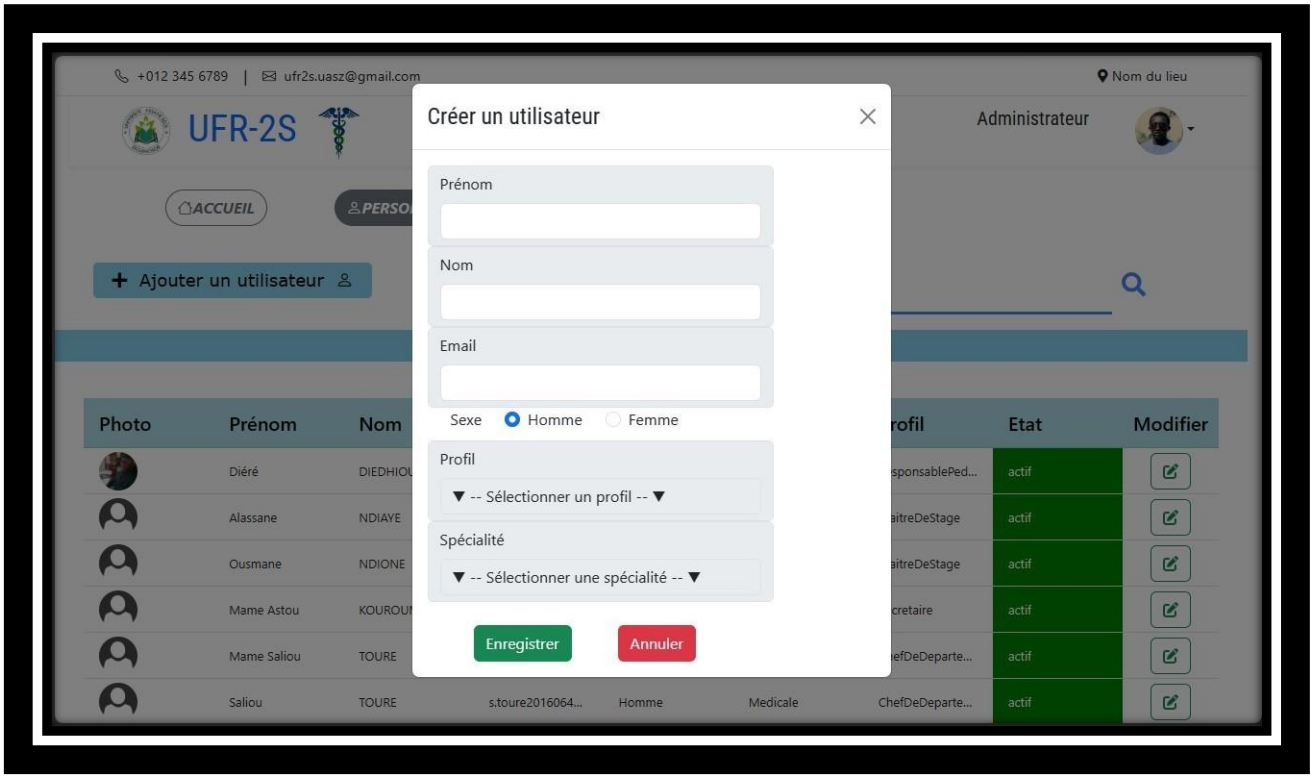

Figure 30 :Formulaire de création d'un utilisateur

IV.6. Liste des utilisateurs connectés

La capture suivante dévoile la liste des utilisateurs connectés en temps réel

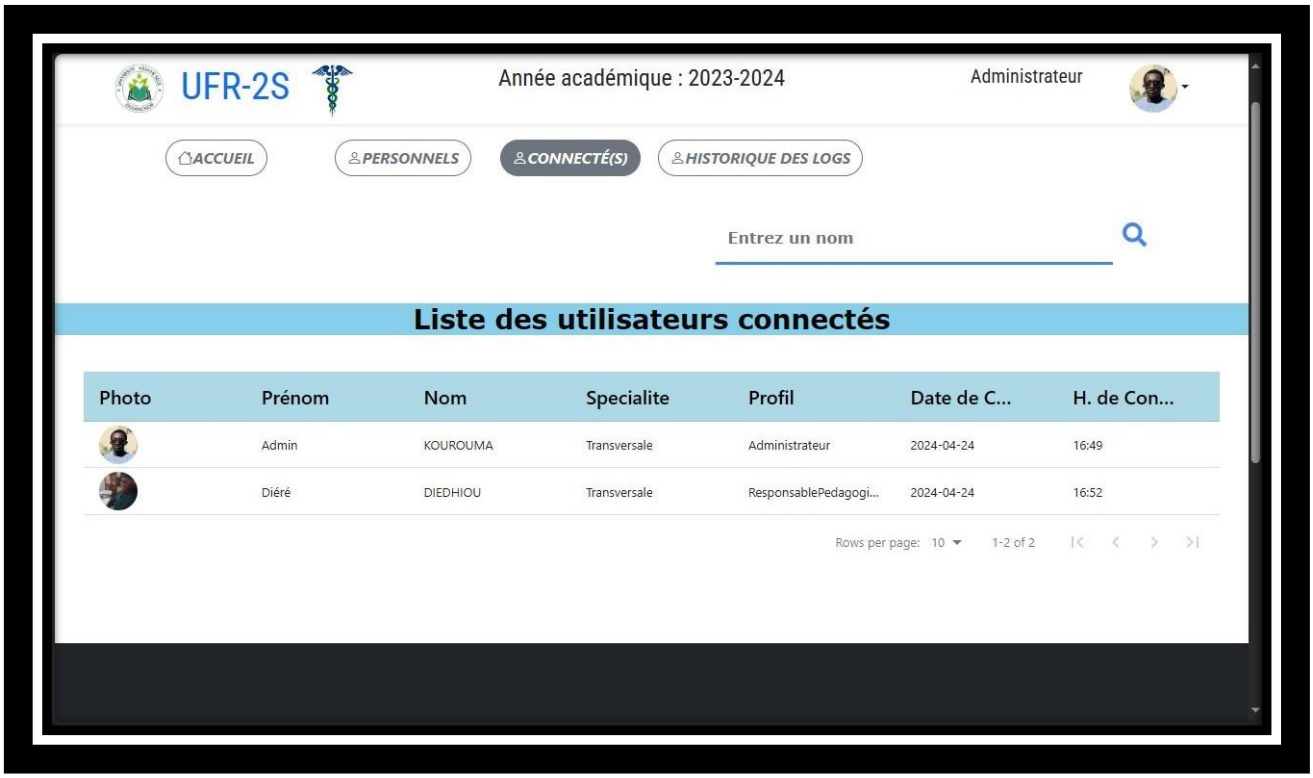

Figure 31 : Liste des utilisateurs connectés

IV.7. Espace du responsable pédagogique

Voici l'interface d'accueil du responsable pédagogique qui est chargé de gérer les années, les semestres, les hôpitaux entre autres.

Figure 32: Page d'accueil du responsable pédagogique

IV.8. Liste des semestres

Cette figure montre la liste des semestres démarrés par le responsable pédagogique

Figure 33: Liste des semestres

IV.9. Formulaire de démarrage d'un semestre

Voici l'aperçu du formulaire de démarrage d'un semestre

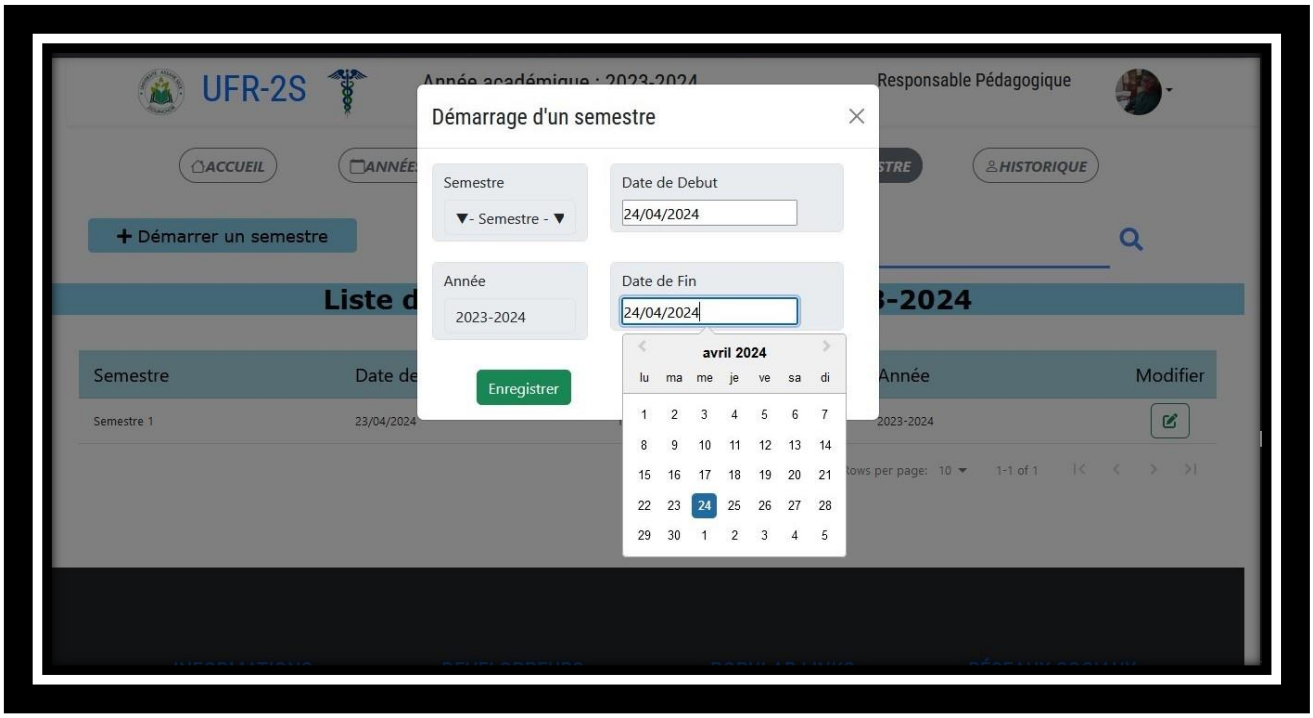

Figure 34 : Formulaire de démarrage d'un semestre

IV.10. Liste des hôpitaux

Voici la liste des hôpitaux créés ou qui sont en étroite collaboration avec l'UFR 2S

Figure 35 : Liste des hôpitaux

IV.11. Espace du chef de département en chirurgie

Cette figure révèle l'espace de travail du chef de département de chirurgie qui assure la gestion des modules en les assignant aux maitres de stage appropriés

Figure 36 : Page d'accueil du chef de département en chirurgie

IV.12. Liste des stages

La figure ci-après nous montre la liste des stages (modules) ouverts par le chef de département

Figure 37 : Liste des stages

IV.13. Formulaire de démarrage d'un stage

Voici le formulaire de démarrage d'un stage (module) chirurgical

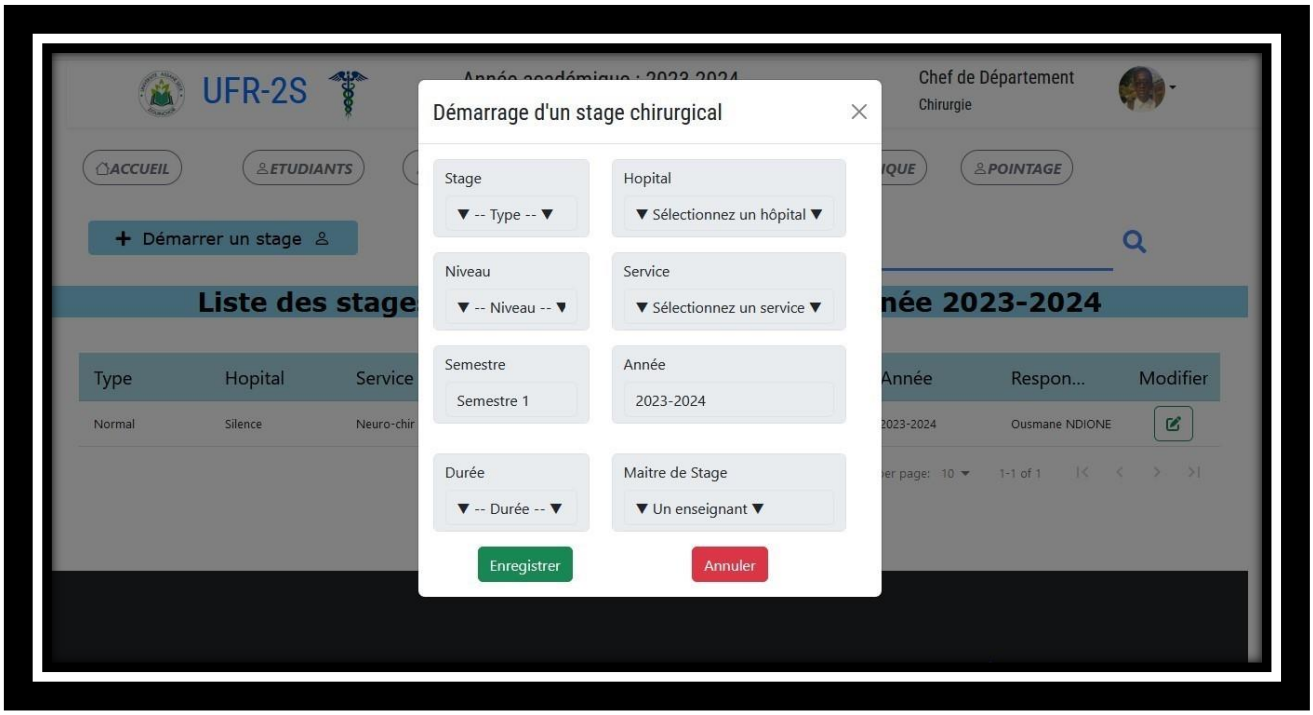

Figure 38 : Formulaire de démarrage d'un stage

Conclusion

Ce chapitre met en lumière les choix technologiques effectués pour la réalisation de l'application ainsi que les interfaces utilisateur conçues en fonction des différents profils du système. Ces décisions sont cruciales pour garantir le succès du projet en termes de fonctionnalités offertes, d'expérience utilisateur et de performance globale de l'application.

Conclusion Générale et Perspectives

En tant qu'étudiant en Master Génie logiciel, j'ai eu le privilège d'effectuer mon stage au sein de l'UFR des sciences de la santé (UFR 2S) de l'Université Assane Seck de Ziguinchor (UASZ), dans le cadre du Projet de gestion des stages hospitaliers des étudiants. Au cours de cette expérience enrichissante, j'ai eu à contribuer à trois modules essentiels : la gestion des utilisateurs, la gestion des années académiques et la gestion des hôpitaux.

La gestion des stages des étudiants revêt une importance cruciale dans le cursus universitaire des étudiants en médecine. Malheureusement, lorsqu'elle est réalisée sur support papier, cette gestion présente de nombreux inconvénients et limites qui entravent le travail du personnel impliqué. Face à cette réalité, nous avons conclu qu'une informatisation de ce processus s'avère indispensable pour surmonter ces obstacles.

Ce mémoire a débuté par la présentation de l'UFR des sciences de la santé et l'exposition de la problématique sous-jacente. Nous avons ensuite opté pour la méthodologie Agile dans notre processus de conception, car elle s'alignait parfaitement avec notre approche de travail, nécessitant une forte interaction avec le client et une collaboration étroite entre les membres de l'équipe. Par la suite, nous avons entrepris la modélisation et la conception en procédant à une analyse approfondie des besoins pour identifier les différents intervenants du système. En découpant notre application en sous-modules, nous avons identifié les fonctionnalités associées à chacun à l'aide de diagrammes de cas d'utilisation. Dans la phase de conception, nous avons exposé les architectures choisies pour la réalisation de l'application, puis nous avons présenté nos diagrammes de séquences, d'activités et de classes pour clore cette étape avec succès.

À la suite de cette phase, nous avons entamé la mise en œuvre de l'application, optant pour la technologie MERN (MySQL, Express, React, Node). Enfin, nous avons conclu cette étape en présentant quelques interfaces clés de l'application.

Pendant la réalisation de ce travail, nous avons mis en pratique les connaissances et compétences acquises tout au long de notre formation. Nous avons utilisé des méthodologies telles que l'UML pour la modélisation, ainsi que des compétences pratiques dans l'utilisation de la technologie MERN, parmi d'autres outils et technologies pertinentes.

En Outre, nous avons utilisé l'outil de développement VSCode, ainsi que PowerAMC pour créer les différents diagrammes. Ce stage nous a également offert une expérience précieuse dans un environnement professionnel, renforçant nos compétences en travail d'équipe et en gestion de projet.

Nous sommes heureux de constater que nous avons largement atteint nos objectifs initiaux en réalisant les trois modules mentionnés précédemment. Cette réalisation représente une avancée significative dans la facilitation du travail du personnel de l'UFR des sciences de la santé.

Dans le cadre de ce travail, plusieurs perspectives prometteuses s'ouvrent pour des travaux futurs. Parmi celles-ci, nous envisageons notamment :

- L'établissement de statistiques détaillées pour faciliter la gestion des bilans des stages.
- L'adaptation de l'application en une version mobile, offrant ainsi une accessibilité accrue.
- L'intégration des spécialités des sciences paramédicales et de biologie ainsi que des explorations fonctionnelles pour une couverture plus complète.
- La collaboration avec des opérateurs téléphoniques afin de permettre aux étudiants de recevoir des notifications relatives à leurs stages par message.
- Et bien d'autres possibilités encore.

Ces perspectives ouvrent de nouvelles opportunités pour améliorer encore davantage l'efficacité et la praticité de notre application.

Bibliographie

[1] « Historique de L'UASZ | Université Assane Seck de Ziguinchor ». Consulté le: 15 avril 2024. [En ligne]. Disponible sur: https://uasz.sn/?page_id=891

[2] « UFR Sciences de la Santé | Université Assane Seck de Ziguinchor ». Consulté le: 15 avril 2024. [En ligne]. Disponible sur: https://uasz.sn/?page_id=137521

[3] « Le modèle en cascade (waterfall model) », IONOS Digital Guide. Consulté le: 23 février 2024. [En ligne]. Disponible sur: https://www.ionos.fr/digitalguide/sites-internet/developpement-web/modele-en-cascade/

[4] « Méthode agile : Qu'est ce que c'est en gestion de projet ? | Slack ». Consulté le: 22 février 2024. [En ligne]. Disponible sur: https://slack.com/intl/fr-fr/blog/collaboration/methode-agile

[5] ideematic, « Définition des méthodes Agiles », Idéematic. Consulté le: 22 février 2024. [En ligne]. Disponible sur: https://www.ideematic.com/actualites/2015/01/methodes-agiles-definition/

[6] « Scrum, une méthode de développement agile », CFI SA. Consulté le: 23 février 2024. [En ligne]. Disponible sur: https://www.cfi.ch/en/scrum-une-methode-de-developpement-agile/

[7] « Méthode Agile : Le Guide Complet », Le Blog LeHibou. Consulté le: 22 février 2024. [En ligne]. Disponible sur: https://www.lehibou.com/communaute/methode-agile

[8] « Qu'est-ce que le DevOps ? », IONOS Digital Guide. Consulté le: 22 février 2024. [En ligne]. Disponible sur: https://www.ionos.fr/digitalguide/sites-internet/developpement-web/quest-ce-que-le-devops/

[9] A. L. ADECHINA, « Article 1 : Comprendre les Fondamentaux du DevOps », Lawal ALAO. Consulté le: 22 février 2024. [En ligne]. Disponible sur: https://lawalalao.com/article-1-comprendre-les-fondamentaux-du-devops

[10] « DevOps : définition, pratiques et avantages | NetApp ». Consulté le: 22 février 2024. [En ligne]. Disponible sur: https://www.netapp.com/fr/devops-solutions/what-is-devops/

[11] « What is DevOps and where is it applied? », SHALB. Consulté le: 23 février 2024. [En ligne]. Disponible sur: https://shalb.com/blog/what-is-devops-and-where-is-it-applied/

[12] « Disadvantages of Devops? Why Is It Challenging? » Consulté le: 22 février 2024. [En ligne]. Disponible sur: https://www.orientsoftware.com/blog/advantages-and-disadvantages-of-devops/

[13] « Qu'est-ce que le langage UML (langage de modélisation unifié) ? », Lucidchart. Consulté le: 26 février 2024. [En ligne]. Disponible sur: https://www.lucidchart.com/pages/fr/langage-uml

[14] « Les types de diagrammes UML | Blog Lucidchart ». Consulté le: 26 février 2024. [En ligne]. Disponible sur: https://www.lucidchart.com/blog/fr/types-de-diagrammes-UML

[15] « IBM Documentation ». Consulté le: 26 février 2024. [En ligne]. Disponible sur: https://www.ibm.com/docs/fr/rational-soft-arch/9.5?topic=diagrams-use-case

[16] « Analyse des besoins, étude du besoin — méthode IAFACTORY ». Consulté le: 2 mars 2024. [En ligne]. Disponible sur: https://www.iafactory.fr/service-ux/consulting-ux/analyse-du-besoin.php

[17] M. Alam, « Qu'est-ce qu'un diagramme d'activités ? Définition, exemples et diagramme d'activité en UML », IdeaScale. Consulté le: 2 mars 2024. [En ligne]. Disponible sur: https://ideascale.com/fr/blogues/quest-ce-que-le-diagramme-dactivite/

[18] M. Alam, « Qu'est-ce qu'un diagramme de séquence ? Diagrammes de définition et de séquence en UML », IdeaScale. Consulté le: 5 mars 2024. [En ligne]. Disponible sur: https://ideascale.com/fr/blogues/quest-ce-que-le-diagramme-de-sequence/

[19] « Memoire Online - Conception et réalisation d'une application web pour la gestion des stocks cas d'étude magasin de la faculté des sciences exactes de l'université de Bejaia - Laaziz LAHLOU », Memoire Online. Consulté le: 7 mars 2024. [En ligne]. Disponible sur: https://www.memoireonline.com/07/10/3700/m_Conception-et-realisation-dune-application-web-pour-la-gestion-des-stocks-cas-detude-magasin23.html

[20] « Architecture physique », *Wikipédia*. 13 décembre 2022. Consulté le: 7 mars 2024. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Architecture_physique&oldid=199456801

[21] « leClientServeur1.pdf ». Consulté le: 7 mars 2024. [En ligne]. Disponible sur: http://mariepascal.delamare.free.fr/IMG/pdf/leClientServeur1.pdf

[22] « Qu'est-ce que l'architecture à trois niveaux | IBM ». Consulté le: 7 mars 2024. [En ligne]. Disponible sur: https://www.ibm.com/fr-fr/topics/three-tier-architecture

[23] « Chapitre 4 Conception logique (Guide de planification du déploiement de Sun Java Enterprise System 2005Q4) ». Consulté le: 7 mars 2024. [En ligne]. Disponible sur: https://docs.oracle.com/cd/E19636-01/819-3453/6n5m27hbl/index.html

[24] C. Hoang, « Monolith Architecture », Medium. Consulté le: 7 mars 2024. [En ligne]. Disponible sur: https://tech.tamara.co/monolith-architecture-5f00270f384e

[25] « Qu'est-ce que l'architecture de microservices ? | Google Cloud », Google Cloud. Consulté le: 7 mars 2024. [En ligne]. Disponible sur: https://cloud.google.com/learn/what-is-microservices-architecture?hl=fr

[26] « (27) Microservices vs. Monolithic Architecture: Choosing the Right Path for Your Software | LinkedIn ». Consulté le: 7 mars 2024. [En ligne]. Disponible sur: https://www.linkedin.com/pulse/microservices-vs-monolithic-architecture-choosing-right-kakapakala/

[27] « Apprenez l'architecture pilotée par les événements », OpenClassrooms. Consulté le: 7 mars 2024. [En ligne]. Disponible sur: https://openclassrooms.com/fr/courses/7210131-definissez-votre-architecture-logicielle-grace-aux-standards-reconnus/7370416-apprenez-larchitecture-pilotee-par-les-evenements

[28] SeB, « L'architecture 3-tiers à l'heure du serverless », Le weblogue de SeB. Consulté le: 7 mars 2024. [En ligne]. Disponible sur: https://blog.lecacheur.com/2017/01/26/larchitecture-3-tiers-a-lheure-du-serverless/

[29] MouraDev, « Le pattern MVC », MouraDev. Consulté le: 7 mars 2024. [En ligne]. Disponible sur: https://mouradev.wordpress.com/2015/10/29/le-pattern-mvc/

[30] « conception détaillée ». Consulté le: 8 mars 2024. [En ligne]. Disponible sur: https://vitrinelinguistique.oqlf.gouv.qc.ca/fiche-gdt/fiche/8373240/conception-detaillee

[31] « Diagramme de classes UML : définition, avantages, composants et exemples », Lucidchart. Consulté le: 8 mars 2024. [En ligne]. Disponible sur: https://www.lucidchart.com/pages/fr/diagramme-de-classes-uml

[32] « Que signifie Implémentation? - Definition IT de LeMagIT », LeMagIT. Consulté le: 11 mars 2024. [En ligne]. Disponible sur: https://www.lemagit.fr/definition/Implementation

[33] « Qu'est-ce que la MERN Stack ? Tout ce qu'il faut savoir », Kicklox. Consulté le: 11 mars 2024. [En ligne]. Disponible sur: https://www.kicklox.com/blog-talent/mern-stack-definition-enjeux-avantages/

[34] « Présentation de MySQL ». Consulté le: 11 mars 2024. [En ligne]. Disponible sur: https://phpsources.net/presentation/mysql/index

[35] « Présentation de Node.js ». Consulté le: 11 mars 2024. [En ligne]. Disponible sur: https://developer.oracle.com/fr/learn/technical-articles/1481879245890-103-what-is-node-js

[36] V. Billey, « Introduction à React ». Consulté le: 11 mars 2024. [En ligne]. Disponible sur: https://www.synbioz.com/blog/tech/introduction-a-react

[37] « Qu'est-ce qu'Express.js ? Tout ce que vous devez savoir », Kinsta®. Consulté le: 11 mars 2024. [En ligne]. Disponible sur: https://kinsta.com/fr/base-de-connaissances/qu-est-express-js/

[38] R. L. Dev, « Qu'est-ce que le Stack MERN ? Introduction et Ressources », Medium. Consulté le: 11 mars 2024. [En ligne]. Disponible sur: https://refaireledev.medium.com/quest-ce-que-le-stack-mern-introduction-et-ressources-3a92d02d9b17

[39] « 🔎 PowerAMC : définition et explications », Techno-Science.net. Consulté le: 26 février 2024. [En ligne]. Disponible sur: https://www.techno-science.net/definition/764.html

[40] « SAP Power AMC ». Consulté le: 26 février 2024. [En ligne]. Disponible sur: https://www.next-decision.fr/autres-editeurs/modelisation/sap-power-amc

[41] « Qu'est-ce qu'un environnement de développement intégré (IDE) ? » Consulté le: 11 mars 2024. [En ligne]. Disponible sur: https://www.redhat.com/fr/topics/middleware/what-is-ide

[42] « Visual Studio Code », *Wikipédia*. 15 février 2024. Consulté le: 13 mars 2024. [En ligne]. Disponible sur: https://fr.wikipedia.org/w/index.php?title=Visual_Studio_Code&oldid=212497259#Pr%C3%A9sentation

[43] P. Attitude, « Les outils collaboratifs à avoir pour travailler à distance ! » Consulté le: 14 mars 2024. [En ligne]. Disponible sur: https://blog.parisattitude.com/fr/les-outils-collaboratifs-a-avoir-pour-travailler-a-distance

[44] « Qu'est-ce que Git ? » Consulté le: 14 mars 2024. [En ligne]. Disponible sur: https://www.next-decision.fr/wiki/qu-est-ce-que-git

[45] Atlassian, « Pourquoi utiliser Git ? | Atlassian Git Tutorial », Atlassian. Consulté le: 14 mars 2024. [En ligne]. Disponible sur: https://www.atlassian.com/fr/git/tutorials/why-git

[46] « Sécurité informatique : définition », Konica Minolta Digital Center. Consulté le: 28 avril 2024. [En ligne]. Disponible sur: https://digital-solutions.konicaminolta.fr/infrastructure-informatique/article-securite-informatique-definition/

[47] « Présentation des JSON Web Token (JWT) », IONOS Digital Guide. Consulté le: 28 avril 2024. [En ligne]. Disponible sur: https://www.ionos.fr/digitalguide/sites-internet/developpement-web/json-web-token-jwt/

[48] « Sécurité des applications Web avec JSON Web Token - S2IAD ». Consulté le: 29 avril 2024. [En ligne]. Disponible sur: https://s2iad.com/jwt-10

[49] « Free & Premium HTML Website Templates Download », HTML Codex. Consulté le: 15 avril 2024. [En ligne]. Disponible sur: https://htmlcodex.com/

[50] K. Veerappan, « Test de logiciel | Définition et cycle de vie • Zuci Systems », Zuci Systems. Consulté le: 14 mars 2024. [En ligne]. Disponible sur: https://www.zucisystems.com/be/blog/fondamentaux-des-tests-de-logiciels/

[51] « About Postman », Postman API Platform. Consulté le: 14 mars 2024. [En ligne]. Disponible sur: https://www.postman.com/company/about-postman/

[52] « Send your first API request », Postman Learning Center. Consulté le: 14 mars 2024. [En ligne]. Disponible sur: https://learning.postman.com/docs/getting-started/first-steps/sending-the-first-request/

Annexes

Dictionnaire des données

Utilisateurs		
Colonne	**Type**	**Null**
id(clé primaire)	bigint(20)	Non
nom	varchar(255)	Non
prenom	varchar(255)	Non
sexe	varchar(255)	Non
email	varchar(255)	Non
telephone	varchar(255)	Non
Comptes		
Colonne	**Type**	**Null**
id(clé primaire)	bigint(20)	Non
login	varchar(255)	Non
Mot_de_passe	varchar(255)	Non
etat	varchar(255)	Non
profil	varchar(255)	Non
photo	varchar(255)	Non
utilisateurId	bigint(20)	Non
Logs		
Colonne	**Type**	**Null**
id(clé primaire)	bigint(20)	Non
dateDeLog	date	Non
heureDeConn	varchar(255)	Non
heureDeDeconn	varchar(255)	Non
utilisateurId	bigint(20)	Non
photo	varchar(255)	Non
utilisateurId	bigint(20)	Non
AnneeAcademiques		
Colonne	**Type**	**Null**
id(clé primaire)	bigint(20)	Non
annee	varchar(255)	Non
utilisateurId	bigint(20)	Non
HopitalSites		
Colonne	**Type**	**Null**
id(clé primaire)	bigint(20)	Non
nom	varchar(255)	Non
etat	varchar(255)	Non
Specialites		
Colonne	**Type**	**Null**
id(clé primaire)	bigint(20)	Non

nom	varchar(255)	Non
etat	varchar(255)	Non
hopitalSiteId	bigint(20)	Non
SousServices		
Colonne	**Type**	**Null**
id(clé primaire)	bigint(20)	Non
nom	varchar(255)	Non
etat	varchar(255)	Non
specialiteId	bigint(20)	Non
Stages		
Colonne	**Type**	**Null**
id(clé primaire)	bigint(20)	Non
dateDebut	varchar(255)	Non
dateFin	varchar(255)	Non
annee	varchar(255)	Non
utilisateurId	bigint(20)	Non
Etudiants		
Colonne	**Type**	**Null**
id(clé primaire)	bigint(20)	Non
matricule	varchar(255)	Non
nom	varchar(255)	Non
prenom	varchar(255)	Non
sexe	varchar(255)	Non
niveau	varchar(255)	Non
email	varchar(255)	Non
dateDeNaissance	varchar(255)	Non
utilisateurId	bigint(20)	Non
hopitalSiteId	bigint(20)	Non
specialiteId	bigint(20)	Non
sousServiceId	bigint(20)	Non
Modules		
Colonne	**Type**	**Null**
id(clé primaire)	bigint(20)	Non
type	varchar(255)	Non
hopital	varchar(255)	Non
specialite	varchar(255)	Non
service	varchar(255)	Non
niveau	varchar(255)	Non
semestre	varchar(255)	Non
duree	varchar(255)	Non
annee	varchar(255)	Non
utilisateurId	bigint(20)	Non
stageId	bigint(20)	Non
Evaluations		
Colonne	**Type**	**Null**
id(clé primaire)	bigint(20)	Non
note	double	Non
utilisateurId	bigint(20)	Non

anneeAcademiqueId	bigint(20)	Non
sousServiceId	bigint(20)	Non
moduleId	bigint(20)	Non
Pointages		
Colonne	**Type**	**Null**
id(clé primaire)	bigint(20)	Non
datePointage	date	Non
heureArrivee	varchar(255)	Non
heureDepart	varchar(255)	Non
etat	varchar(255)	Non
utilisateurId	bigint(20)	Non
etudiantId	bigint(20)	Non
Affiliations		
Colonne	**Type**	**Null**
id(clé primaire)	bigint(20)	Non
hopitalSiteId	bigint(20)	Non
utilisateurId	bigint(20)	Non
Historiques		
Colonne	**Type**	**Null**
id(clé primaire)	bigint(20)	Non
etudiantId	bigint(20)	Non
evaluationId	bigint(20)	Non
date	date	Non
type	varchar(255)	Non
hopital	varchar(255)	Non
specialite	varchar(255)	Non
service	varchar(255)	Non
semestre	varchar(255)	Non
duree	varchar(255)	Non
annee	varchar(255)	Non