

# Projet Fin D'étude

## Licence Sciences Mathématiques et Informatique



### Développement d'un Chatbot Pédagogique pour la Faculté des Sciences Ben M'Sick : Application de la Génération Augmentée par Récupération (RAG) aux Supports de Cours Universitaires

#### Encadré par :

Pr. BENLAHMAR EL HABIB

Mr. ELFAKIR ZAKARIA

Mr. KAICH OUSSAMA

**Soutenu le 14 juin 2025**

#### Devant le jury :

Pr. ZAHOUR Omar

Pr. KANDALI Khalid

Pr. BENLAHMAR EL HABIB

Mr. ELFAKIR ZAKARIA

Mr. KAICH OUSSAMA

Mme. HANOUNI Salma

#### Réalisé par :

Mr. MOURABIH MAROUANE

Mr. HAKAM SALLAHDINE

Mr. JAFFAL MOHAMMED

**Année Universitaire : 2024/2025**

# Dédicaces

Grâce à la bénédiction de Dieu et son aide nous avons pu accomplir ce  
modeste travail.

Nous dédions ce travail :

À nos chers parents, pour tous leurs sacrifices, leur amour, leur tendresse,  
leur soutien et leurs prières tout au long de nos études.

À nos chères sœurs et frères pour leurs encouragements permanents, et  
leur  
soutien moral.

À tous les étudiants de la Faculté des Sciences Ben M'Sick, en espérant que  
ce

projet puisse leur être d'une aide précieuse dans leur parcours académique.

À tous ceux qui œuvrent pour l'amélioration de l'accès à la connaissance et  
au  
soutien pédagogique

# Remerciement

Au préambule à ce projet de fin d'étude, nous remercions Dieu qui nous aide et nous donne la patience et le courage durant ces années d'études.

Il est souvent difficile de remercier les gens qui vous aident à accomplir les tâches qui vous sont données, et pourtant nous nous devons d'exprimer l'entière gratitude que nous ressentons envers eux.

Nous tenons à exprimer notre profonde gratitude à notre encadrant principal, Professeur BENLAHMAR EL HABIB, pour son encadrement, sa disponibilité, ses conseils avisés et ses efforts pertinents, qu'il nous a accordés tout au long de ce projet.

Nous exprimons également notre gratitude envers nos co-encadrants, Monsieur Zakaria El Fakir et Monsieur Oussama Kaich, pour leur aide précieuse et leurs conseils durant la réalisation de ce projet.

Nos remerciements s'adressent également à l'ensemble du Corps Professoral de la FSBM pour tout leur savoir-faire qu'ils ont su nous procurer.

Sans oublier nos parents, nos familles, pour leurs contributions, leurs soutiens, leurs patiences et encouragements au cours de la réalisation de ce projet de fin d'étude, pour que nous puissions arriver là où nous en sommes.

Nous remercions également nos amis et camarades de promotion pour leur soutien mutuel et les moments partagés.

# Résumé

Ce projet porte sur la conception et la mise en œuvre d'un assistant académique intelligent capable de répondre aux questions des étudiants en se basant sur le contenu de leurs cours.

Face à l'abondance de supports pédagogiques (polycopiés, diapositives, documents numériques) et à la difficulté d'y retrouver rapidement une information précise, nous proposons une solution innovante reposant sur le principe du Retrieval-Augmented Generation (RAG).

L'assistant utilise une base de connaissances vectorielle construite à partir des documents de cours de la FSBM (modules S1 à S6) et un modèle de langage (LLM) pour formuler des réponses précises, contextualisées et exclusivement fondées sur le contenu fourni.

Une interface web conviviale (développée avec Streamlit) permet à l'utilisateur de poser ses questions en français de manière naturelle. Le système recherche d'abord les informations pertinentes dans les documents vectorisés grâce à des embeddings sémantiques et à l'index FAISS, puis génère une réponse en langage naturel en citant les sources consultées.

Le prototype réalisé a démontré la faisabilité de cette approche, offrant aux étudiants un outil interactif pour faciliter la révision et la compréhension de leurs cours. Les résultats sont prometteurs, bien que dépendants de la qualité des documents sources, de la pertinence de l'indexation, et des capacités du modèle de langage utilisé.

À l'avenir, des améliorations sont envisagées, telles que l'extension aux modules spécialisés, l'intégration d'une gestion multilingue, ou encore l'ajout d'un système de feedback pour renforcer l'efficacité pédagogique de l'assistant.

# ABSTRACT

This project focuses on the design and implementation of an intelligent academic assistant capable of answering students' questions based on the content of their university courses.

Given the abundance of educational materials (handouts, slides, digital documents) and the difficulty of quickly retrieving specific information, we propose an innovative solution based on the Retrieval-Augmented Generation (RAG) approach.

The assistant leverages a vector knowledge base built from the course documents of the FSBM (modules S1 to S6), combined with a Large Language Model (LLM) to generate accurate, contextualized responses strictly grounded in the provided content.

A user-friendly web interface (developed with Streamlit) allows users to naturally ask questions in French. The system first searches for relevant information within the vectorized documents using semantic embeddings and FAISS indexing, then generates a natural language response while citing the consulted sources.

The developed prototype has demonstrated the feasibility of this approach, offering students an interactive tool to support their course review and comprehension. The results are promising, though they depend on the quality of the source documents, the efficiency of the indexing process, and the capabilities of the chosen language model.

Future improvements are planned, including support for specialized modules, multilingual interaction, and the integration of a feedback system to enhance the assistant's educational effectiveness.

هذا المشروع يتمحور حول تصميم وتنفيذ مساعد أكاديمي ذكي قادر على الإجابة على أسئلة الطلبة اعتمادًا على محتوى دروسهم.

أمام كثرة الموارد البيداغوجية (مطويات، شرائح، وثائق رقمية (وصعوبة العثور بسرعة على معلومات دقيقة، نقترح حلاً مبتكرًا يستند إلى مبدأ\* التوليد المعزز بالاسترجاع - Retrieval-Augmented Generation (RAG)\*.

يعتمد هذا المساعد على قاعدة معرفية شعاعية تم إنشاؤها انطلاقًا من وثائق دروس كلية العلوم بن مسيك (FSBM) للفصول الدراسية من S1 إلى S6 ، بالإضافة إلى نموذج لغوي (LLM) لتوليد إجابات دقيقة، وسياقية، ومستندة فقط إلى المحتوى المتوفر.

تُوفّر للمستخدم واجهة ويب سهلة الاستخدام تم تطويرها باستخدام Streamlit تسمح له بطرح أسئلته باللغة الفرنسية بشكل طبيعي. يقوم النظام أولاً بالبحث عن المعلومات الملائمة داخل الوثائق التي تم تحويلها إلى تمثيلات شعاعية باستخدام تقنيات التضمين الدلالي (semantic embeddings) وفهرس FAISS ، ثم يُولّد إجابة بلغة طبيعية مرفقة بالمصادر المستعملة.

لقد أظهر النموذج الأولي (prototype) المنجز قابلية تطبيق هذا النهج، حيث وفر للطلبة أداة تفاعلية تُسهّل عملية المراجعة وفهم الدروس. النتائج كانت مشجعة، لكنها تبقى مرتبطة بجودة الوثائق المصدرية، ودقة الفهرسة، وقدرات النموذج اللغوي المُستخدم.

في المستقبل، يُخطط لإدخال تحسينات، مثل توسيع المساعد ليشمل الوحدات المتخصصة، دعم تعدد اللغات، أو إضافة نظام للتغذية الراجعة (feedback) بهدف تعزيز الفعالية البيداغوجية للمساعد.

# Table des Matières



Dédicaces .....	1
Remerciements .....	2
Résumé .....	3
Abstract (English) .....	4
ملخص (Arabe) .....	5
Table des Matières .....	6
Liste des Figures .....	9
Liste des Abréviations .....	11
Introduction Générale .....	12
<b>Chapitre I : Contexte Général du Projet .....</b>	<b>13</b>
1. Introduction .....	14
2. Présentation de la Faculté des Sciences Ben M'Sick .....	14
3. Contexte Pédagogique à la FSBM .....	15
4. Problématique .....	16
5. Objectifs du Projet .....	17
6. Périmètre du Projet .....	17
1. Fonctionnalités Incluses .....	17
2. Aspects Exclus .....	17
7. Besoins Fonctionnels .....	18
1. Pour l'Étudiant .....	19
2. Pour l'Enseignant .....	19
3. Pour l'Administrateur .....	19
8. Besoins Non Fonctionnels .....	19
9. Méthodologie et Formalismes Adoptés .....	19
1. Méthode Agile : SCRUM .....	19
2. Diagramme de Gantt (Planification) .....	20
10. Conclusion du Chapitre .....	20
<b>Chapitre II : État de l'Art .....</b>	<b>21</b>
1. Introduction.....	22
2. Les Modèles de Langage Larges (LLM).....	22

1. Meta-Llama 4 Scout via OpenRouter : Performance et Intégration.....	22
3. Retrieval Augmented Generation (RAG) .....	22
1. Principes et Architectures .....	23
2. Prétraitement et Stockage Vectoriel (FAISS) .....	23
3. Recherche et Récupération d'Information .....	23
4. Techniques d'Évaluation des Chatbots .....	23
5. Conclusion du Chapitre .....	23
<b>Chapitre III : Étude Préalable du Projet .....</b>	<b>24</b>
1. Introduction .....	25
2. Approfondissement sur les LLM .....	25
1. Définition et Principes de Base .....	25
2. Défis et Limites des LLM .....	25
2. Focus sur LLAMA-4-Scout .....	26
1. Introduction et Spécificités .....	26
2. Comparaison avec GPT-3.5 . .....	26
3. Le Rôle du Chatbot dans le Projet .....	27
4. Constitution du Dataset et Préparation des Données .....	27
5. Conclusion du Chapitre .....	28
<b>Chapitre IV : Conception du Chatbot .....</b>	<b>29</b>
1. Introduction .....	30
2. Définition de l'UML (Unified Modeling Language) .....	30
3. Conception du Chatbot FSBM avec RAG .....	30
3.1. Diagrammes de Cas d'Utilisation .....	30
3.1.1. Rôle Étudiant .....	30
3.1.2. Rôle Administration .....	31
3.1.3. Rôle Professeur .....	32
3.2. Diagrammes de Séquence .....	33
3.2.1. Connexion Utilisateur.....	33
3.2.2. Étudiant : Poser une Question .....	34
3.2.3. Administrateur : Activer/Désactiver Compte .....	35



3.2.4. Professeur : Gérer les Documents .....	36
3.3. Diagramme d'Activité .....	38
3.4. Diagramme d'États de Transitions .....	39
3.5. Diagramme de Classes .....	40
4. Conclusion du Chapitre .....	40
<b>Chapitre V : Réalisation du Chatbot .....</b>	<b>43</b>
1. Introduction .....	44
2. Technologies et Outils Utilisés .....	44
2.1. Outils de Développement et de Gestion .....	44
2.2. Technologies Logicielles .....	46
(Python, HTML, Tailwind CSS, TypeScript, Spring Framework, Java)	
3. Préparation du Dataset .....	47
i. Regroupement des données .....	47
ii. Structuration et Nettoyage .....	49
iii. Mise en Forme du Dataset pour le Système RAG .....	50
4. Scripts Python de gestion de la mémoire vectorielle .....	51
i. create_memory_for_llm_simplified.py .....	52
5. Évaluation du système de mémoire vectorielle .....	54
a) Quelques réponses du modèle réglé .....	54
6. Développement de l'Interface du Chatbot .....	56
i. Développement du Back-end .....	56
ii. Développement du Front-end .....	62
7. Conclusion du Chapitre .....	68
Conclusion Générale .....	69
Perspectives et Recommandations .....	70
Référence .....	71



Figure 1 : Organigramme FSBM – Faculté des Sciences Ben M'Sick .....	15
Figure 2 : Méthode SCRUM .....	19
Figure 3 : Diagramme de Gantt.....	20
Figure 4 : Architecture générale d'un modèle RAG .....	25
Figure 5 : Logo officiel LLAMA .....	26
Figure 6 : Comparaison schématique LLAMA-4-Scout vs GPT-3.5.....	27
Figure 7 : Diagramme de cas d'utilisation (Rôle Étudiant).....	31
Figure 8 : Diagramme de cas d'utilisation (Rôle Administrateur ).....	32
Figure 9 : Diagramme de cas d'utilisation (Rôle Professeur).....	33
Figure 10 : Diagramme de séquence : Connexion Utilisateur (Détailé).....	35
Figure 11 : Diagramme de séquence : Étudiant - Poser une Question.....	36
Figure 12 : Diagramme de séquence : Administrateur - Activer/Désactiver un Compte Utilisateur .....	37
Figure 13 : Diagramme de séquence : Professeur - Gérer les Documents.....	38
Figure 14 : Diagramme d'activité : Pipeline d'Ingestion de Document.....	39
Figure 15 : Diagramme d'état de transitions : Étudiant utilisant le chatbot académique	40
Figure 16 : Diagramme de classes du système Chatbot .....	41
Figure 17 : Paramètres globaux .....	52
Figure 18 : Splitter récursif pour la segmentation des documents .....	52
Figure 19 : Chargement paresseux des ressources .....	53
Figure 20 : Sélection des documents pertinents et retriever dédié .....	53
Figure 21 : Prompt pour l'assistant universitaire .....	53
Figure 22 : Exemple de question sur les curseurs en PL/SQL .....	55
Figure 23 : Exemple de question sur JSP dans un contexte JEE .....	55
Figure 24 : Schéma de l'Architecture Backend et Structure des Packages .....	57
Figure 25 : Schéma de l'architecture configuration .....	57
Figure 26 : Schéma de l'architecture Controller .....	58
Figure 27 : Schéma de l'architecture Service .....	59
Figure 28 : Schéma de l'architecture Model .....	59
Figure 29 : Schéma de l'architecture Repository .....	60
Figure 30 : Schéma de l'architecture DTO .....	60

Figure 31 : Schéma de l'architecture Exception .....	61
Figure 32 : Schéma de l'architecture Security .....	61
Figure 33 : Authentification Stateless .....	62
Figure 34 : Landing page (Page d'accueil) .....	63
Figure 35 : Page d'inscription Étudiant .....	63
Figure 36 : Page d'inscription Professeur .....	64
Figure 37 : Page de vérification d'e-mail .....	65
Figure 38 : Page de Gestion des Documents pour le professeur .....	66
Figure 39 : Page Principale de Chat de l'Assistant .....	67
Figure 40 : Barre Latérale de l'Application de Chat .....	67

# Liste des Abréviations

Abréviation	Description
<b><i>LLAMA</i></b>	Large Language Model Meta AI (Meta IA pour les Modèles de Langage Large)
<b><i>LLM</i></b>	Large Language Model (Modèle de Langage Large)
<b><i>NLP</i></b>	Natural Language Processing (Traitement Automatique du Langage Naturel)
<b><i>GPT</i></b>	Generative Pre-trained Transformers (Générateur Pré-entraîné de Transformer)
<b><i>AI</i></b>	Artificial Intelligence (Intelligence Artificielle)
<b><i>UML</i></b>	Unified Modeling Language (langage de modélisation unifié)
<b><i>RAG</i></b>	Retrieval-Augmented Generation (Génération augmentée par récupération)
<b><i>FAISS</i></b>	Facebook AI Similarity Search (Bibliothèque de recherche vectorielle rapide)
<b><i>Streamlit</i></b>	Framework Python pour créer des applications web interactives
<b><i>API</i></b>	Application Programming Interface (interface de programmation d'applications)
<b><i>DTO</i></b>	Data Transfer Object (objet de transfert de données)

# INTRODUCTION GENERALE

La transformation numérique de l'enseignement supérieur a engendré une profusion de supports de cours au format numérique (documents PDF, présentations, notes de cours, etc.). Les étudiants de la Faculté des Sciences Ben M'Sik (FSBM), par exemple, accumulent semestre après semestre une vaste collection de polycopiés et de documents pédagogiques pour chacun de leurs modules. Face à cet amas d'informations, il devient difficile pour un étudiant de retrouver rapidement une notion précise ou une explication particulière enfouie dans ses documents de cours. Les méthodes de recherche traditionnelles (lecture manuelle, surlignage, recherche de mots-clés) sont chronophages et peu efficaces, ce qui peut freiner la révision et l'apprentissage autonome.

Parallèlement, les avancées récentes en Intelligence Artificielle (IA) ouvrent de nouvelles perspectives pour améliorer l'accès à l'information. En particulier, les chatbots et les modèles de langage de dernière génération (appelés LLM) sont désormais capables de comprendre des questions en langage naturel et de générer des réponses pertinentes. Ces progrès suggèrent qu'il serait possible de doter les étudiants d'un assistant virtuel capable de les aider à naviguer à travers leurs contenus de cours de manière intelligente et interactive.

C'est dans ce contexte qu'est né le présent projet de fin d'études. L'objectif principal est de concevoir et développer FSBM Scholar Assistant, un assistant académique intelligent sous la forme d'un chatbot capable de répondre aux questions des étudiants en se basant exclusivement sur le contenu de leurs supports de cours. La solution envisagée s'appuie sur le paradigme du Retrieval-Augmented Generation (RAG), combinant un moteur de recherche documentaire et un modèle de génération de texte. Concrètement, pour chaque question posée par un étudiant, l'assistant recherche d'abord les informations pertinentes dans la base de documents fournis (polycopiés, diapositives, etc.), puis formule une réponse en langage naturel en s'appuyant sur ces éléments contextuels. Cette approche hybride permet de garantir que les réponses proviennent du contenu validé des cours, réduisant ainsi les risques d'erreurs ou de hors-sujet souvent associés aux modèles de langage non contraints.

Ce rapport détaille la démarche suivie pour la réalisation de cet assistant intelligent. Le Chapitre I établira le contexte général du projet, en définissant la problématique, les objectifs visés et le périmètre de l'étude. Le Chapitre II sera consacré à l'état de l'art, explorant les travaux existants sur les chatbots éducatifs et les systèmes RAG. Le Chapitre III détaillera la conception de notre système, incluant l'architecture RAG, la préparation des données, et la modélisation de l'interface. Le Chapitre IV décrira la phase de réalisation et d'implémentation, présentant les technologies et outils utilisés. Le Chapitre V portera sur l'évaluation du prototype et l'analyse des résultats obtenus. Enfin, une Conclusion Générale viendra synthétiser les contributions du projet et proposer des perspectives d'amélioration.

# Chapitre I

## Contexte général du projet

## 1. Introduction :

L'avènement du numérique a profondément transformé les méthodes d'enseignement et d'apprentissage dans l'enseignement supérieur. Si cette transition offre des avantages indéniables en termes d'accessibilité et de diversité des ressources, elle engendre également de nouveaux défis pour les étudiants. Ce premier chapitre a pour objectif de dresser un panorama complet de l'environnement dans lequel s'inscrit notre projet de fin d'études. Nous débuterons par une analyse détaillée du contexte pédagogique au sein de la Faculté des Sciences Ben M'Sik (FSBM), en mettant en lumière les spécificités de la diffusion et de l'utilisation des supports de cours. Cette analyse nous conduira naturellement à l'identification précise de la problématique centrale que ce projet vise à résoudre : la difficulté pour les étudiants d'exploiter efficacement la masse d'informations à leur disposition. Par la suite, nous formulerons de manière explicite les objectifs, tant principaux que secondaires, que nous aspirons à atteindre à travers le développement de notre solution. Une délimitation rigoureuse du périmètre du projet sera ensuite établie pour clarifier ce qui sera inclus et exclu de notre réalisation. Ce chapitre se poursuivra par une spécification des besoins fonctionnels et non fonctionnels essentiels, qui serviront de cahier des charges pour la conception et l'implémentation du système. Enfin, nous présenterons la méthodologie de gestion de projet que nous avons privilégiée pour structurer notre démarche et assurer le suivi de nos avancées.

## 2. Présentation de la Faculté des Sciences Ben M'Sick :

La Faculté des Sciences Ben M'Sick (FSBM) (2), relevant de l'Université Hassan II de Casablanca, a été créée en 1984. Depuis sa fondation, elle s'est imposée comme un pilier de l'enseignement supérieur scientifique au Maroc, alliant formation académique de qualité et développement actif de la recherche scientifique.

Durant ses premières années, et jusqu'en 2003, la faculté proposait des formations diversifiées allant du premier cycle (DEUG) au second cycle (licence sciences Bac+4), couvrant plusieurs disciplines scientifiques. Dès 1989, elle s'engage dans la formation de troisième cycle (CEA et DES), rendue possible grâce à la mise en place de plusieurs équipes et laboratoires de recherche dynamiques.

L'année 1997 marque un tournant important avec la création des Unités de Formation et de Recherche (UFR). Cette restructuration vise à renforcer la synergie entre formation et recherche, en fédérant les chercheurs autour de thématiques émergentes et porteuses d'innovation.

En 2003, dans le cadre de la réforme de l'enseignement supérieur et l'adoption du système LMD (Licence – Master – Doctorat), la faculté réorganise ses programmes autour d'une formation modulaire et semestrielle, favorisant ainsi une meilleure flexibilité pédagogique et une harmonisation avec les standards internationaux.

La faculté franchit une nouvelle étape en 2008 avec la création du Centre d'Études Doctorales

(CED) intitulé « Sciences et Applications ». Ce centre regroupe l'ensemble des structures de

recherche accréditées et constitue un levier stratégique pour la formation doctorale et la production scientifique de haut niveau.

Enfin, depuis décembre 2015, dans le cadre d'une initiative de structuration renforcée lancée

par l'Université Hassan II, la FSBM a procédé à une réorganisation complète de ses équipes

et laboratoires. Cette démarche a abouti à la mise en place de 19 laboratoires de recherche, témoignant de la richesse, de la diversité et de la vitalité scientifique de l'établissement. Grâce à ses infrastructures, à la qualité de son encadrement, et à son engagement constant en

faveur de la recherche et de l'innovation, la Faculté des Sciences Ben M'Sick constitue aujourd'hui un acteur majeur du paysage scientifique national et régional.

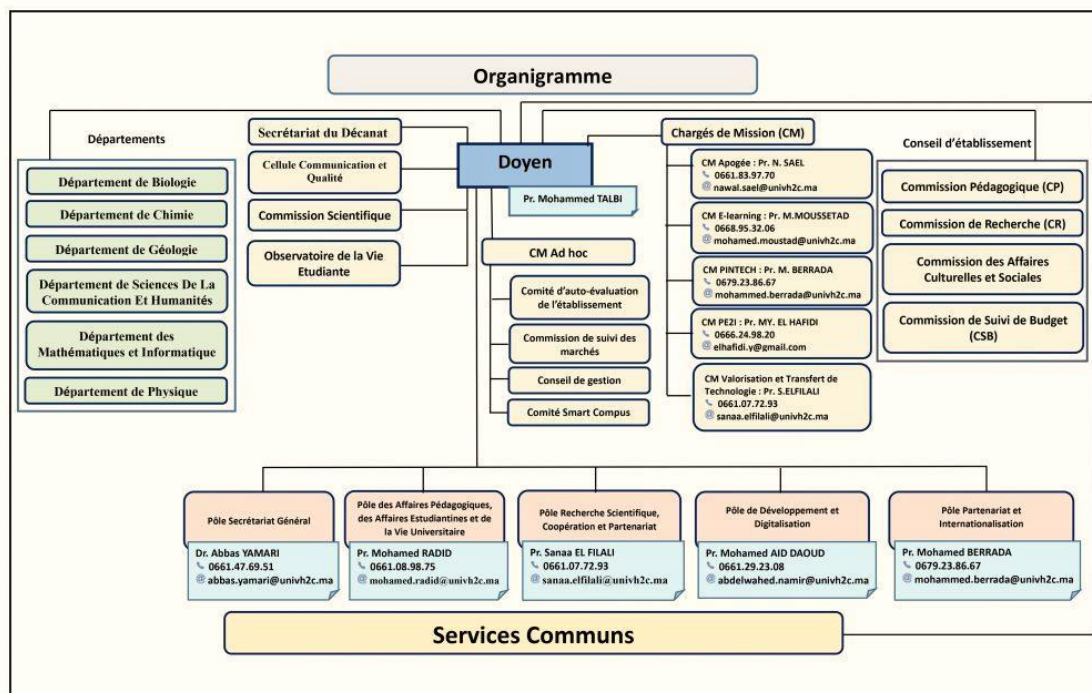


Figure 1: Organigramme FSBM – Faculté des Sciences Ben M'Sick

### 3. Contexte:

L'enseignement supérieur vit depuis une décennie une transformation numérique profonde. À la FSBM, les modules sont souvent accompagnés de supports numériques riches mais éparpillés :

polycopiés en PDF, présentations PowerPoint, documents Word, etc. Ces documents sont

souvent transmis via Google Drive, WhatsApp, e-mails, ou déposés dans des plateformes Moodle.

Bien que ces ressources soient utiles, leur volume élevé rend difficile leur exploitation. Un étudiant peut disposer de plusieurs centaines de pages de contenu pour un seul semestre. Dans ce contexte, accéder rapidement à une définition, un théorème, ou une explication devient un défi.



Les outils de recherche classiques (Ctrl+F dans un PDF, surlignage, lecture linéaire) sont manuels, fastidieux et peu intelligents, car ils ne tiennent pas compte du contexte ni du sens de la question posée

## 4.Problématique :

L'objectif principal du projet est de fournir une solution académique intelligente, personnalisée et fiable aux étudiants de la FSBM. Plus concrètement, nous visons à :

Concevoir un chatbot académique capable de répondre à des questions sur les modules S1 à S6.

Utiliser la technologie RAG (Retrieval-Augmented Generation), qui garantit que chaque réponse est basée uniquement sur les documents fournis (cours, TD, TP...).

- Employer un LLM open-source tel que LLAMA 3, permettant un meilleur contrôle et une adaptation locale.
- Développer une interface simple et intuitive (via Streamlit) pour faciliter l'utilisation par des étudiants non-techniciens.

## 5.Objectifs :

Les étudiants expriment souvent un besoin de clarté et de rapidité d'accès à l'information. Voici quelques problèmes concrets rencontrés :

Difficulté à retrouver rapidement une explication vue au début du semestre.

- Confusion entre différents documents portant sur le même module.
- Perte de temps lors des périodes de révision à parcourir tous les fichiers.
- Frustration face à des outils de recherche qui ne comprennent pas les questions formulées en langage naturel.

D'un autre côté, les modèles de langage (comme GPT) sont performants, mais ne garantissent pas la véracité de leurs réponses, car ils génèrent du contenu basé sur leurs propres données internes (pré-entraînement). Il est donc risqué de leur faire confiance pour des contenus académiques spécifiques.

## 6. Périmètre du projet :

Le périmètre du projet définit les limites fonctionnelles et techniques du système que nous avons choisi de concevoir dans le cadre de ce projet de fin d'études. Il précise ce que le système prend en charge, ainsi que les aspects qui ont été volontairement exclus afin de cadrer la portée du travail dans les délais impartis.

Dans cette version du projet, le système se concentre exclusivement sur les documents pédagogiques textuels (principalement en format PDF) fournis par les enseignants de la Faculté des Sciences Ben M'Sik (FSBM) pour les modules allant du semestre S1 jusqu'au S6. Le chatbot est conçu pour répondre à des questions en français, en se basant uniquement sur le contenu extrait, indexé et vectorisé à partir de ces documents.

Le périmètre couvre notamment les fonctionnalités suivantes :

- Le prétraitement et l'extraction du texte à partir des documents PDF.
- La création d'une base de connaissances vectorielle à l'aide d'embeddings sémantiques.
- L'intégration d'un moteur de recherche intelligente basé sur l'algorithme FAISS.
- L'utilisation d'un modèle de langage LLM (tel que LLAMA 3) dans un pipeline de type RAG pour générer des réponses contextualisées.
- Le développement d'une interface web simple (via Streamlit) pour interagir avec le chabot.

En revanche, certains aspects ont été exclus du périmètre du projet, notamment :

- La prise en charge de documents multimédias (vidéos, audio, images).
- La gestion multilingue (le chabot ne répond qu'en français).
- Le déploiement sur un serveur public ou dans un environnement cloud sécurisé.
- L'intégration d'un système de gestion de comptes utilisateurs avancé (rôles, permissions, suivi personnalisé).

Ce cadrage clair du périmètre nous a permis de concentrer nos efforts sur les aspects fondamentaux du projet, tout en laissant la porte ouverte à des extensions futures qui pourraient enrichir le système.

## 7. Besoins Fonctionnels :

Le système développé distingue deux types d'utilisateurs : l'étudiant et l'enseignant, chacun ayant des droits et des fonctionnalités spécifiques.

Tout d'abord, l'utilisateur (qu'il soit étudiant ou enseignant) doit se connecter à la plateforme à l'aide de son email et de son mot de passe. Si l'utilisateur ne possède pas encore de compte, il peut s'inscrire en remplissant un formulaire contenant un nom, un email valide et un mot de passe sécurisé. Un lien de validation est ensuite envoyé à l'adresse email renseignée pour activer le compte.

Une fois connecté, l'étudiant accède à une interface simple composée d'un champ de saisie pour poser des questions en langage naturel, et d'une zone d'affichage pour visualiser la réponse générée par le chabot. Il peut également consulter son historique de conversations, le renommer, le supprimer ou l'archiver selon ses besoins. L'étudiant peut se déconnecter à tout moment via le menu principal

Quant à l'enseignant, en plus des fonctionnalités précédentes, il dispose d'un espace dédié lui permettant de gérer les documents pédagogiques. Il peut ajouter, modifier ou supprimer des fichiers PDF associés à ses modules. Ces documents sont automatiquement pris en compte par le système et intégrés dans la base de données vectorielle utilisée par le chabot

Ainsi, le périmètre du projet englobe :

- La gestion des comptes utilisateurs avec deux rôles distincts : étudiant et enseignant.
- La sécurisation de l'inscription et de l'authentification. .
- L'interface de chat académique avec recherche et génération de réponse.
- La consultation et gestion de l'historique côté étudiant.
- La gestion des documents PDF côté enseignant (upload, modification, suppression).

Ce fonctionnement permet de garantir une expérience personnalisée et contrôlée selon le rôle de chaque utilisateur dans le système.

## 8. Besoins Non Fonctionnels :

En plus des fonctionnalités principales que le système doit assurer, plusieurs exigences non fonctionnelles doivent être respectées afin de garantir une qualité d'utilisation, une fiabilité du service et une expérience utilisateur fluide.

Voici les besoins non fonctionnels identifiés :

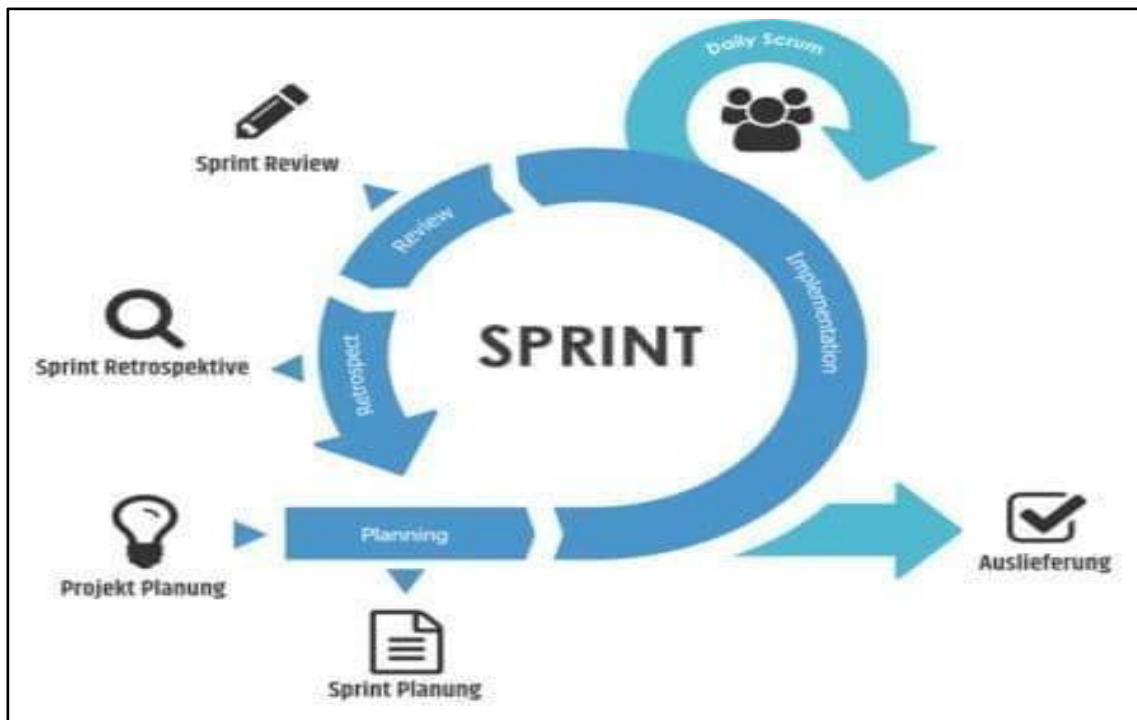
- **Performance** : Le système doit fournir une réponse à chaque question en moins de 2 secondes, afin de garantir une interaction fluide et agréable pour l'utilisateur.
- **Fiabilité** : Le chatbot doit toujours fournir des réponses basées uniquement sur les documents indexés, sans générer de contenu inventé. Cela garantit la cohérence académique des réponses.
- **Sécurité** : L'accès aux documents pédagogiques doit être restreint aux utilisateurs autorisés. Les enseignants doivent être les seuls à pouvoir modifier ou supprimer les fichiers PDF. Les mots de passe sont hachés et les connexions sont protégées.
- **Accessibilité** : L'interface doit être simple et intuitive, accessible aussi bien sur ordinateur que sur mobile. Aucune compétence technique n'est requise pour utiliser l'outil.
- **Modularité** : L'architecture du système doit permettre d'ajouter de nouveaux modules ou documents sans perturber le fonctionnement global.
- **Scalabilité** : Le système doit être prêt à gérer plusieurs utilisateurs simultanément sans perte significative de performance.
- **Maintenabilité** : Le code source doit être organisé et documenté, facilitant ainsi les futures évolutions, corrections de bugs ou mises à jour du système.
- **Confidentialité** : Les informations personnelles des utilisateurs ainsi que les documents de cours doivent être traités de manière confidentielle et ne doivent jamais être exposés à des tiers

## 9.Méthodologie et formalismes adoptés Fonctionnels :

### 9.1 . Méthode SCRUM:

SCRUM est un cadre de travail agile populaire utilisé dans le développement logiciel, mais également dans d'autres domaines où la gestion de projet nécessite flexibilité et adaptation. Voici quelques-unes de ses caractéristiques principales :

- Itératif et Incrémental
- Rôles Définis
- Réunions Flexibilité
- Transparence et Inspection
- Auto-organisation



*Figure 2: méthode SCRUM*

### 9.2 . Diagramme de Gantt:

L'un des outils les plus efficaces pour représenter visuellement l'état d'avancement des différentes activités (tâches) qui constituent un projet, couramment utilisé en gestion de projet. On a fait une planification avec Gantt :

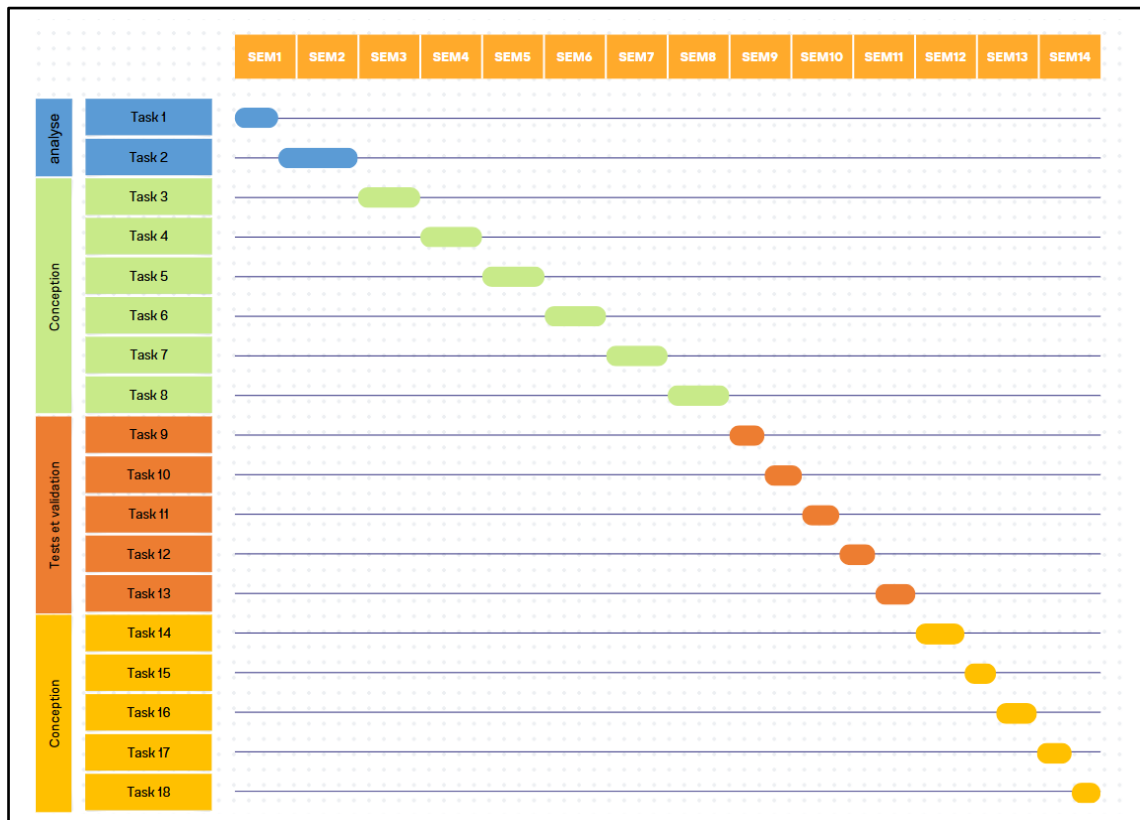


Figure 3:diagramme de GANTT

## 10. Conclusion :

Ce chapitre nous a permis de situer notre projet dans un besoin réel exprimé par les étudiants de la FSBM, et de proposer une réponse technologique innovante et adaptée. Grâce à une méthodologie agile, des outils modernes, et des objectifs clairement définis, nous avons posé les fondations solides pour la suite du développement.

Le prochain chapitre détaillera l'état de l'art, à savoir les technologies existantes, les approches RAG, les modèles de langage, et les exemples comparables déjà développés dans le domaine de l'éducation.

## Chapitre II

### L'état de l'art

## 1- Introduction :

L'état de l'art constitue une analyse approfondie des technologies et des approches actuelles dans le domaine des chatbots intégrant le retrieval augmented generation (RAG). Dans le cadre de notre projet, maîtriser ces évolutions techniques récentes est fondamental, nous permettant d'intégrer les meilleures pratiques et d'assurer une réalisation scientifique rigoureuse et pertinente. Cette démarche garantit non seulement une continuité avec les travaux existants, mais aussi une contribution originale en adéquation avec les dernières avancées technologiques.

## 2- Meta-Llama 4 Scout via OpenRouter : Performance et intégration :

➤ **Pré-entraînement** : Le modèle Llama-4-Scout repose sur l'architecture éprouvée de la famille des modèles Llama de Meta, avec des améliorations substantielles en termes de taille contextuelle et de capacités de compréhension du langage naturel, adaptées spécifiquement aux besoins des systèmes de questions-réponses. Son entraînement se base sur un corpus massif diversifié, permettant de traiter efficacement des questions complexes et nuancées.

➤ **Affinement et intégration API** : Notre choix stratégique d'utiliser Llama-4-Scout via l'API OpenRouter facilite l'accès à une infrastructure robuste et performante, simplifiant l'intégration au sein de notre application tout en assurant la scalabilité. Cette approche permet un affinement ciblé via des instructions claires intégrées directement dans le prompt, renforçant la pertinence des réponses fournies aux utilisateurs.

➤ **Sécurité et gestion des biais** : OpenRouter garantit des mécanismes efficaces pour contrôler la génération de réponses, limitant les biais et les dérives potentiellement nuisibles. De plus, le modèle Llama-4-Scout, grâce à ses mécanismes internes optimisés, réduit les risques de toxicité et d'inexactitudes, garantissant ainsi des interactions sécurisées et fiables avec les utilisateurs finaux.

➤ **Observations** : L'intégration via OpenRouter permet une gestion fine de la similarité des réponses grâce à des réglages spécifiques (e.g., seuils de similarité cosinus paramétrables). Ceci est particulièrement pertinent dans notre contexte académique où la précision et la neutralité des réponses sont critiques.

## 3- Introduction au Retrieval Augmented Generation (RAG) :

Les architectures RAG combinent efficacement deux approches : la récupération de documents pertinents à partir d'une base de connaissances préétablie et la génération dynamique de réponses par des modèles de langage avancés. Notre projet utilise cette méthode hybride, assurant des réponses pertinentes, précises et contextualisées. Le processus comprend plusieurs étapes critiques :

Prétraitement et stockage : Création d'une base vectorielle FAISS alimentée par des embeddings générés avec le modèle multilingue intfloat/multilingual-e5-small, permettant un accès rapide et précis aux informations pertinentes.

Recherche et récupération : Implémentation d'une recherche de similarité optimisée avec NumPy, offrant performance et transparence dans la sélection des documents à intégrer au prompt.

Génération et présentation des réponses : Le modèle Llama-4-Scout génère ensuite une réponse contextualisée, citant explicitement les sources utilisées, renforçant ainsi la crédibilité et l'utilité du chatbot pour les étudiants

## **4- Techniques avancées d'évaluation des Chatbots :**

L'évaluation des chatbots demeure un défi important, surtout lorsqu'il s'agit d'assurer l'impartialité et l'exactitude des réponses générées. Pour notre projet, nous envisageons d'adopter plusieurs métriques :

Score de similarité cosinus : Permet de mesurer objectivement la pertinence entre la question posée et les réponses issues de la base documentaire.

Évaluation qualitative humaine : Indispensable pour détecter les nuances, les biais subtils et assurer une qualité perçue élevée des réponses générées.

Ces métriques combinées nous offrent une méthode d'évaluation complète, adaptée spécifiquement à nos objectifs pédagogiques.

## **5- Conclusion :**

Ce chapitre présente les bases techniques solides nécessaires à notre projet, tout en soulignant les choix méthodologiques et technologiques adoptés. En intégrant une technologie avancée comme Llama-4-Scout via OpenRouter avec une architecture RAG optimisée, nous sommes en mesure de produire un chatbot performant, fiable et impartial, répondant parfaitement aux exigences académiques et pédagogiques définies dans notre cahier des charges.



## Chapitre III

### Etude préalable du projet

## 1. Introduction

Dans ce chapitre, nous abordons les principes fondamentaux de notre sujet en explorant son contexte et ses bases. Notre objectif est de saisir les concepts clés qui constituent la base de notre travail, tout en établissant clairement le cadre dans lequel notre projet s'inscrit.

## 2. Généralités sur les LLM :

### i. Définition :

Les LLM (Large Language Models) sont des modèles avancés d'intelligence artificielle appartenant au domaine du traitement du langage naturel (NLP). Ces modèles sont capables de comprendre et de générer du texte en se basant sur des modèles statistiques obtenus par l'entraînement sur d'immenses corpus textuels.

### ii. Principes de base :

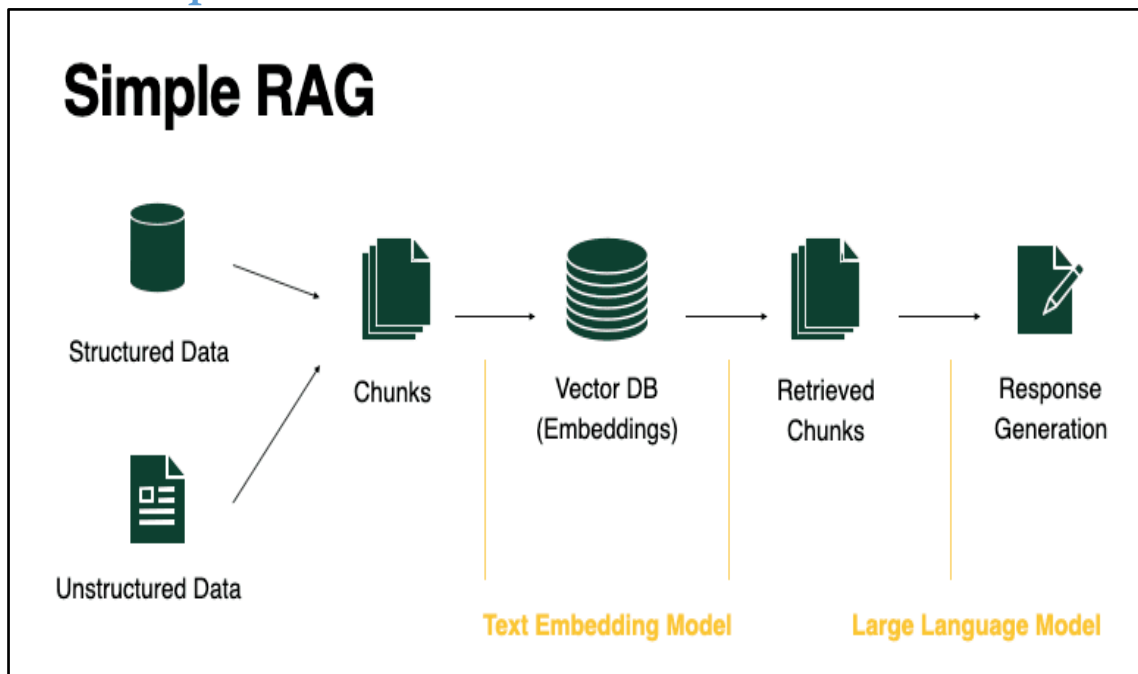


Figure 4 : Architecture générale d'un modèle Transformer

### iii. Défis et limites :

Bien que puissants, les LLM comme LLAMA-4-Scout présentent certaines limites et défis importants :

**Limitation contextuelle :** Malgré leur puissance, les LLM peuvent parfois interpréter incorrectement le contexte d'une requête, entraînant des réponses imprécises ou inadéquates.

**Biais intrinsèques :** Les résultats produits par le modèle peuvent présenter des biais reflétant ceux contenus dans les données initiales, impactant potentiellement la neutralité et l'objectivité des réponses.

Dépendance à la qualité des données d’entraînement : La qualité et l'efficacité du modèle dépendent largement de la quantité et de la qualité des données initiales utilisées durant l'entraînement.

### 3. Généralités sur LLAMA-4-Scout :

#### i. Introduction :

Dans le cadre précis de notre projet, nous avons choisi d’utiliser spécifiquement LLAMA-4-Scout via l’API d’OpenRouter. LLAMA-4-Scout est une déclinaison avancée et optimisée des modèles de la famille LLAMA, spécialement adaptée aux applications de chatbot et à la génération contextualisée de réponses fiables et pertinentes. Le choix de LLAMA-4-Scout répond à notre volonté d’exploiter un modèle performant tout en restant accessible via OpenRouter.



*Figure 5 : Logo officiel LLAMA*

#### ii. Comparaison succincte entre LLAMA et GPT-3.5 :

Critère	LLAMA-4-Scout	GPT-3.5
Taille des données	Modérée	Élevée
Sécurité des réponses	Élevée (optimisé sécurité)	Moyenne
Fraîcheur des données	Données récentes et actualisées	Données jusqu’à fin 2021
Accessibilité via OpenRouter	Directe via OpenRouter	Nécessite API OpenAI

Ainsi, LLAMA-4-Scout est adapté précisément à notre contexte d’utilisation grâce à sa performance équilibrée, sa sécurité accrue et son accès facilité par l’API d’OpenRouter.

Feature	LLaMA-4-Scout	GPT-3.5 Turbo
Parameters	109B total, 17B active (MoE)	Estimated 14 5
MMLU Score	75,2%	154-175B
Output Speed	120,4 tokens/sec	152,3 tokens/sec
Latency (TTFT)	0,35 seconds	0,42 seconds
Open Source	Yes	No

*Figure 6 : Comparaison schématique LLAMA-4-Scout vs GPT-3.5*

## 4. Chatbot :

Un chatbot est une interface conversationnelle qui permet aux utilisateurs d'interagir avec des systèmes numériques via du texte ou de la voix, imitant ainsi une interaction humaine naturelle. Dans notre projet, nous exploitons précisément le potentiel conversationnel de LLAMA-4-Scout en combinaison avec une approche RAG pour permettre des réponses précises basées sur des connaissances extraites directement des documents de référence stockés dans notre base vectorielle FAISS.

## 5. Dataset et préparation des données :

Dans notre projet, le terme Dataset désigne spécifiquement l'ensemble des documents (PDF, DOCX, PPTX, TXT) issus des supports pédagogiques de la FSBM, utilisés pour créer notre base vectorielle. Les étapes concrètes appliquées à ces données dans notre implémentation sont :

Extraction automatique du texte à partir de documents variés (PDF, DOCX, PPTX, TXT).

Découpage en segments adaptés (environ 800 tokens par chunk avec chevauchement de 80 tokens) afin d'optimiser le rappel contextuel lors des interrogations.

Création d'embeddings textuels grâce au modèle d'embedding multilingual-e5-small d'Intfloat (Hugging Face).

Stockage efficace dans une base de données vectorielle FAISS permettant une récupération rapide et pertinente des segments les plus proches lors des requêtes utilisateur.

Ces étapes constituent les fondations de notre méthode de réponse assistée par extraction (RAG).

## 6. Conclusion

Ce chapitre a permis d'établir précisément les fondations conceptuelles et techniques de notre projet. Nous avons présenté les modèles de langage utilisés, justifié leur choix (LLAMA-4-Scout) et détaillé notre approche pour préparer et exploiter les données dans le cadre de notre chatbot basé sur la technologie RAG. Cette clarification approfondie des concepts et outils utilisés est essentielle avant d'aborder concrètement, dans le chapitre suivant, la conception et la réalisation effective du chatbot.

## Chapitre IV

### Conception du Chatbot

## 1. Introduction

La plupart des utilisateurs ne sont pas des experts en informatique, ce qui souligne l'importance de fournir un moyen simple pour exprimer leurs besoins. C'est là que conception interviennent. Dans ce chapitre, nous présenterons la modélisation de notre future application web. Nous illustrerons une vue globale de la solution à l'aide d'un diagramme de cas d'utilisation, une vue statique avec le diagramme de classes, ainsi qu'une vue dynamique à travers des diagrammes de séquence.

## 2. Définition de l'UML :

UML (Unified Modeling Language) est un langage de modélisation standardisé, utilisé en ingénierie logicielle pour visualiser, spécifier, concevoir et documenter les composants d'un système logiciel.

Développé initialement par Grady Booch, James Rumbaugh et Ivar Jacobson, UML permet de représenter un système sous forme de diagrammes, facilitant ainsi la communication entre les parties prenantes (analystes, développeurs, clients, etc.).

UML ne décrit pas comment coder le système, mais comment il est structuré et comment il se comporte à travers différents types de diagrammes.

Objectif de UML :

L'objectif principal de UML est de fournir une représentation claire et compréhensible d'un système logiciel, quel que soit son niveau de complexité, afin de favoriser une conception cohérente, une meilleure communication au sein de l'équipe, et une documentation solide du système.

## 3. Conception du Chatbot FSBM avec RAG :

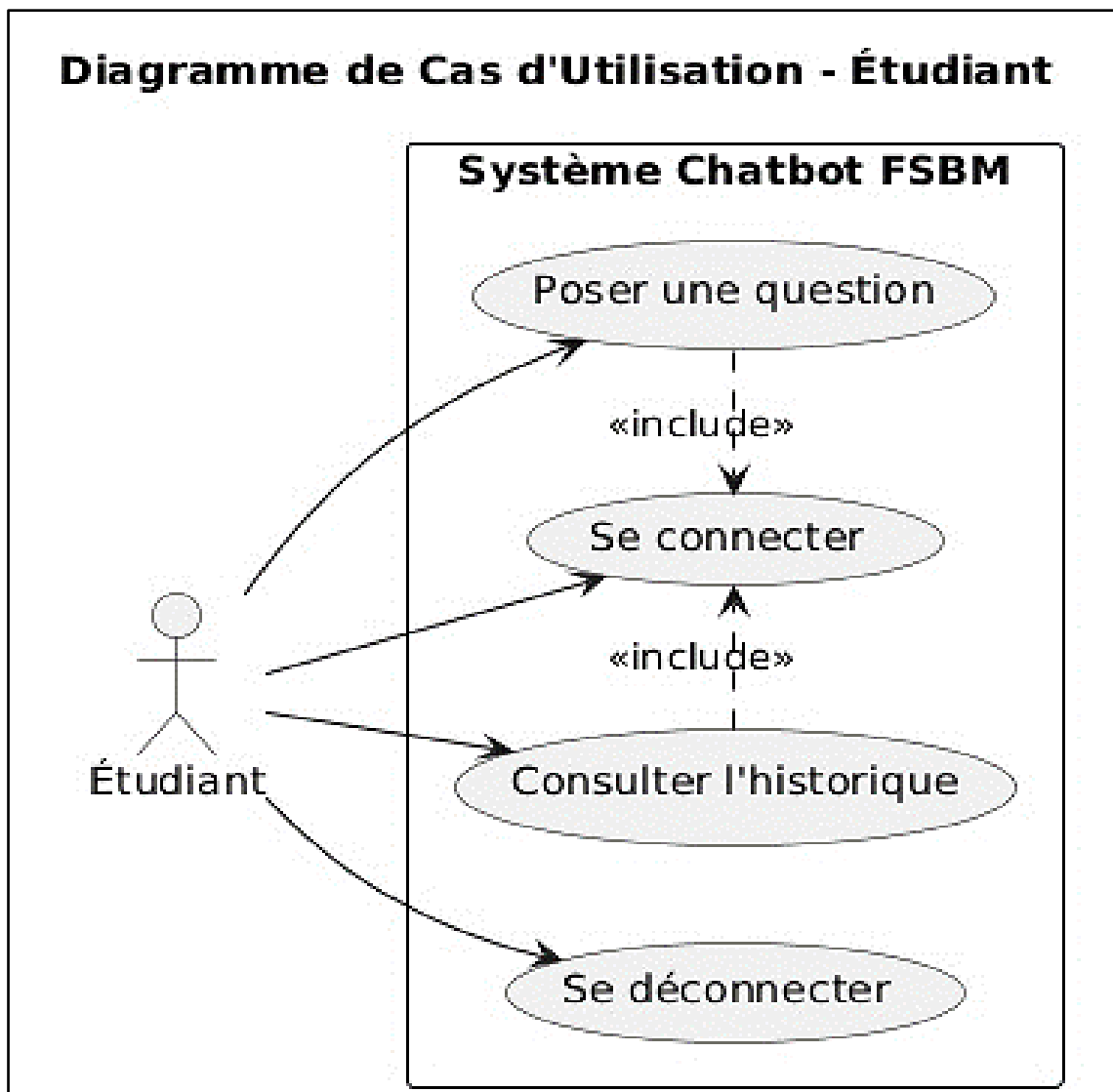
Cette partie vise à affiner la description du fonctionnement de notre chatbot académique en détaillant les aspects clés de sa mise en œuvre, notamment l'intégration du système RAG (Retrieval-Augmented Generation) et la gestion des documents de cours. L'objectif est de parvenir à une représentation technique précise et concrète, très proche de notre future application web. Pour modéliser l'architecture complète du système – de l'interaction utilisateur à la logique interne du RAG – nous allons utiliser le langage UML. Nous emploierons ainsi des diagrammes de cas d'utilisation pour illustrer les fonctionnalités offertes aux différents rôles (Étudiant, Professeur, Admin), un diagramme de classes pour définir la structure statique des composants logiciels (Chatbot, RAGSystem, PDFDocument, etc.) et leurs relations, et enfin des diagrammes de séquence pour décrire le comportement dynamique et la collaboration entre ces composants lors de scénarios clés comme la réponse à une question ou l'ingestion d'un nouveau document PDF.

### i. Diagramme de cas d'utilisation :

Le diagramme de cas d'utilisation montre les interactions entre les utilisateurs (acteurs) et le système. Il décrit les fonctionnalités offertes par le système et les différents scénarios d'utilisation, sans entrer dans les détails techniques.

### *a. étudiant:*

Ce diagramme modélise les interactions fonctionnelles d'un étudiant avec le système de chatbot FSBM. Il établit que l'étudiant a pour objectifs principaux de poser des questions et de consulter son historique. Cependant, l'accès à ces deux fonctionnalités essentielles est conditionné par une authentification préalable, représentée par le cas d'utilisation "Se connecter" qui est inclus (rendu obligatoire) par les deux autres. L'étudiant a également la possibilité de terminer explicitement sa session via "Se déconnecter". Le diagramme fournit une vue d'ensemble claire et concise des capacités offertes à l'étudiant et des prérequis nécessaires (l'authentification) pour les utiliser, servant ainsi de base à la compréhension des exigences fonctionnelles pour ce rôle utilisateur.



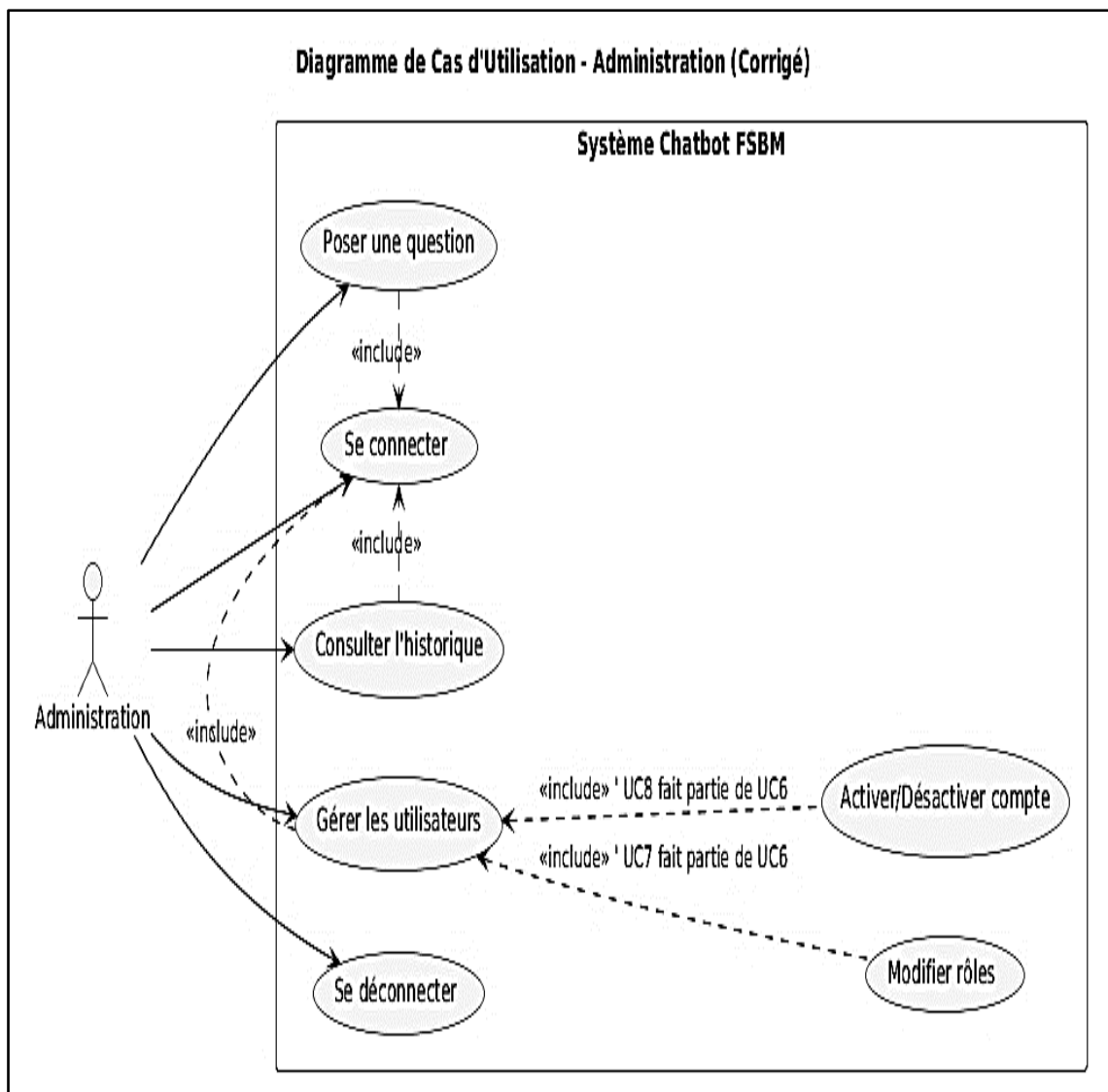
*Figure 7 : Diagramme de cas d'utilisation (étudiant)*

### *b. Administration :*

Ce diagramme met en évidence le rôle central de l'Administration dans la maintenance des utilisateurs du Système Chatbot FSBM. Bien que l'administrateur puisse utiliser des fonctions basiques comme poser des questions ou consulter l'historique (probablement à



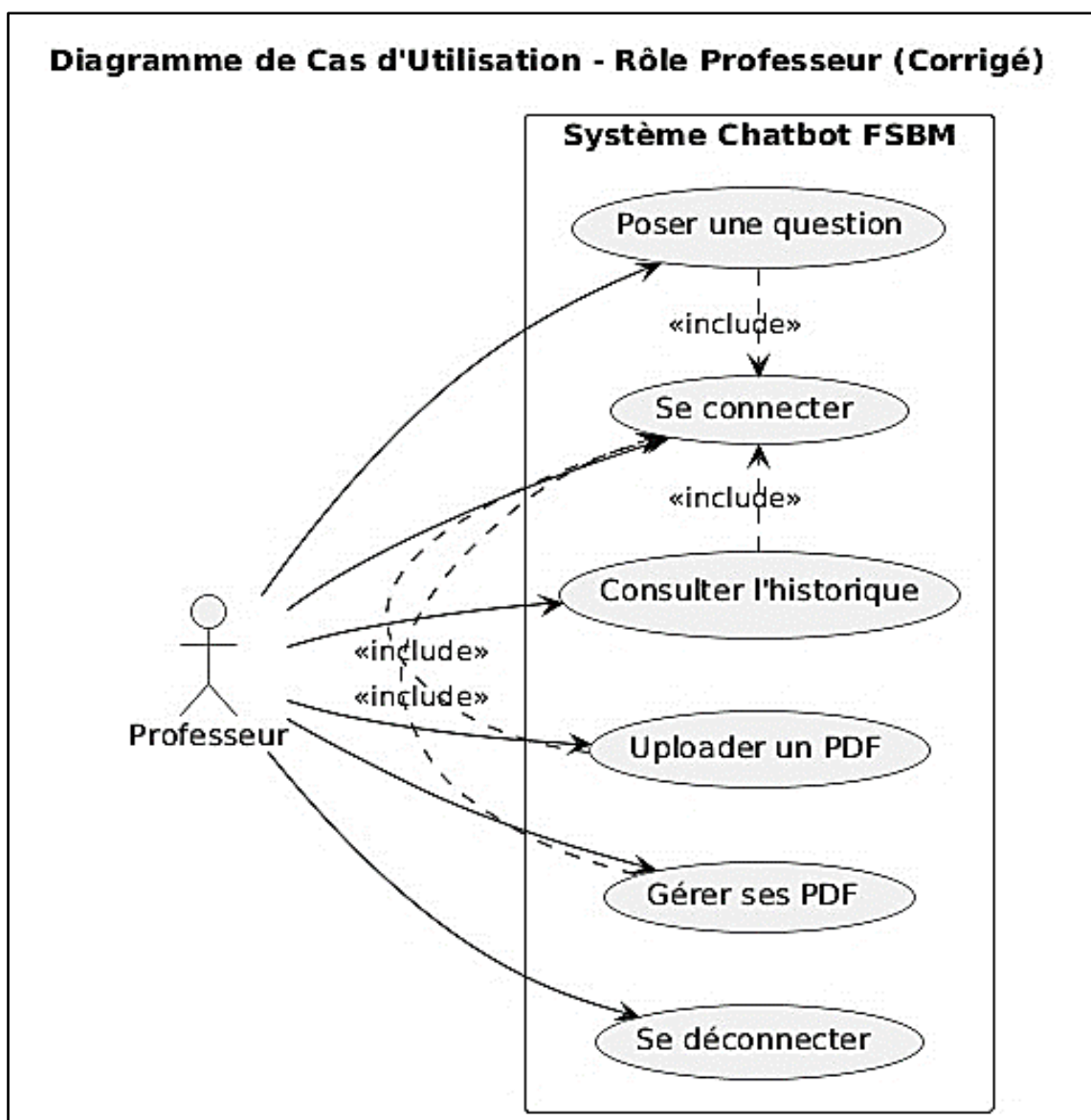
des fins de test ou de support), sa tâche principale est de "Gérer les utilisateurs". Cette gestion, comme l'indiquent les relations <<include>> et les labels associés (malgré leur placement et direction non standards sur le diagramme), englobe des actions spécifiques telles que l'activation/désactivation des comptes et la modification des rôles assignés. L'accès à toutes ces fonctionnalités, et en particulier à la gestion des utilisateurs, est strictement conditionné par une authentification préalable ("Se connecter"). L'action "Se déconnecter" permet de clore la session administrative. Le diagramme, bien qu'ayant des imperfections de représentation UML (placement des use cases spécifiques hors système), communique l'idée que l'administrateur est le garant de la gestion des accès et des permissions au sein de l'application.



*Figure 8 : Diagramme de cas d'utilisation (administrateur)*

### *c. Professeur :*

Ce diagramme illustre les capacités offertes au Professeur par le système Chatbot FSBM. Le Professeur partage certaines fonctionnalités avec l'Étudiant (poser des questions, consulter son historique, se connecter/déconnecter), mais possède également des responsabilités cruciales et spécifiques : l'ajout ("Uploader un PDF") et la maintenance ("Gérer ses PDF") des documents de cours. Ces documents sont essentiels au fonctionnement du système RAG qui alimente les réponses du chatbot. Le diagramme souligne fortement, via les relations <<include>>, que toutes ces actions, qu'elles soient consultatives ou modificatrices de contenu, requièrent une authentification préalable ("Se connecter"). Cela garantit que seules les personnes autorisées peuvent interagir de manière significative avec le système et surtout, gérer la base de connaissances documentaire.



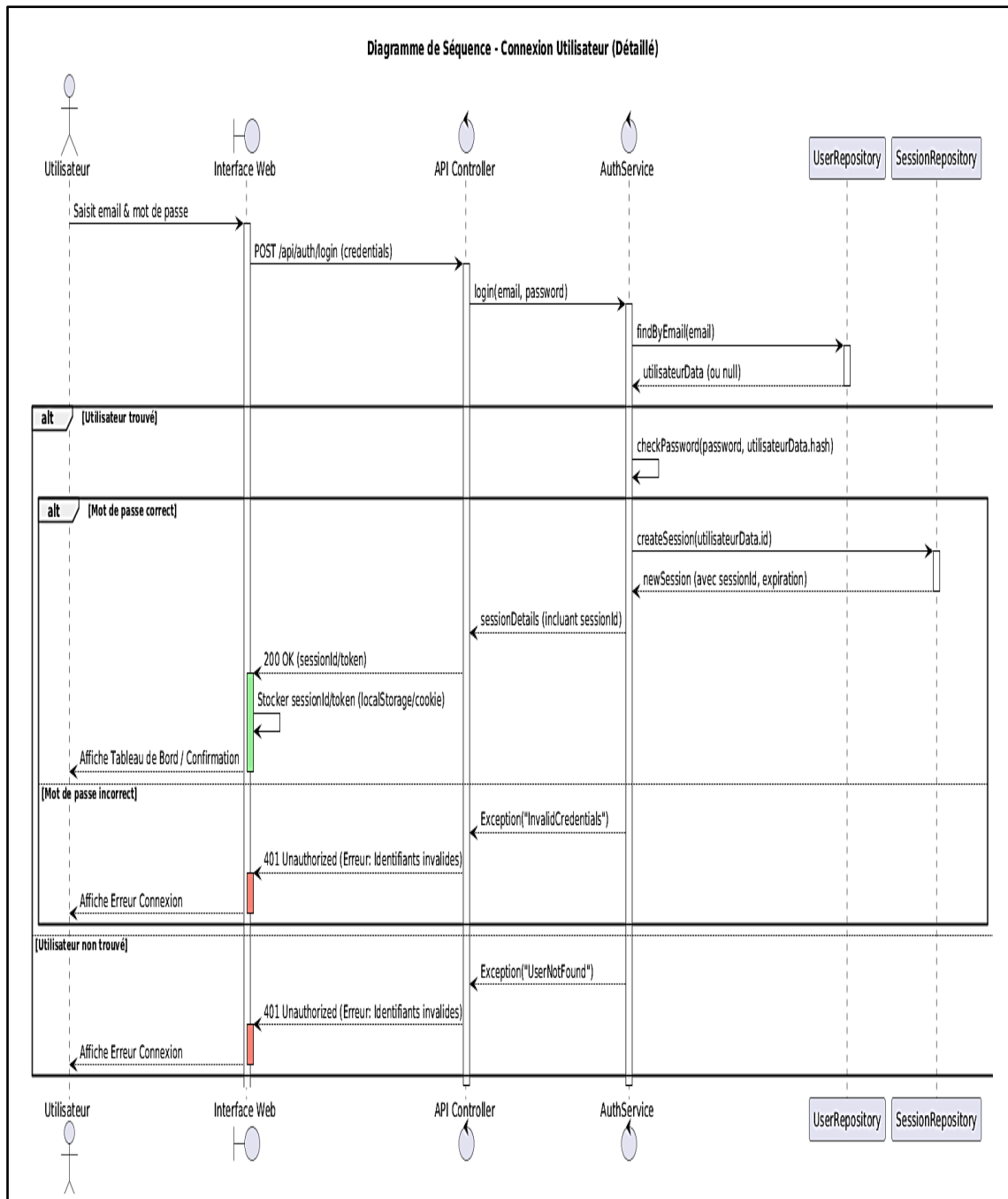
*Figure 9 : Diagramme de cas d'utilisation (professeur)*

## ii. Diagramme de séquence :

Le diagramme de séquence décrit comment les objets interagissent dans un scénario particulier d'un cas d'utilisation. Il montre l'ordre chronologique des messages échangés entre les objets.

### *a. connexion :*

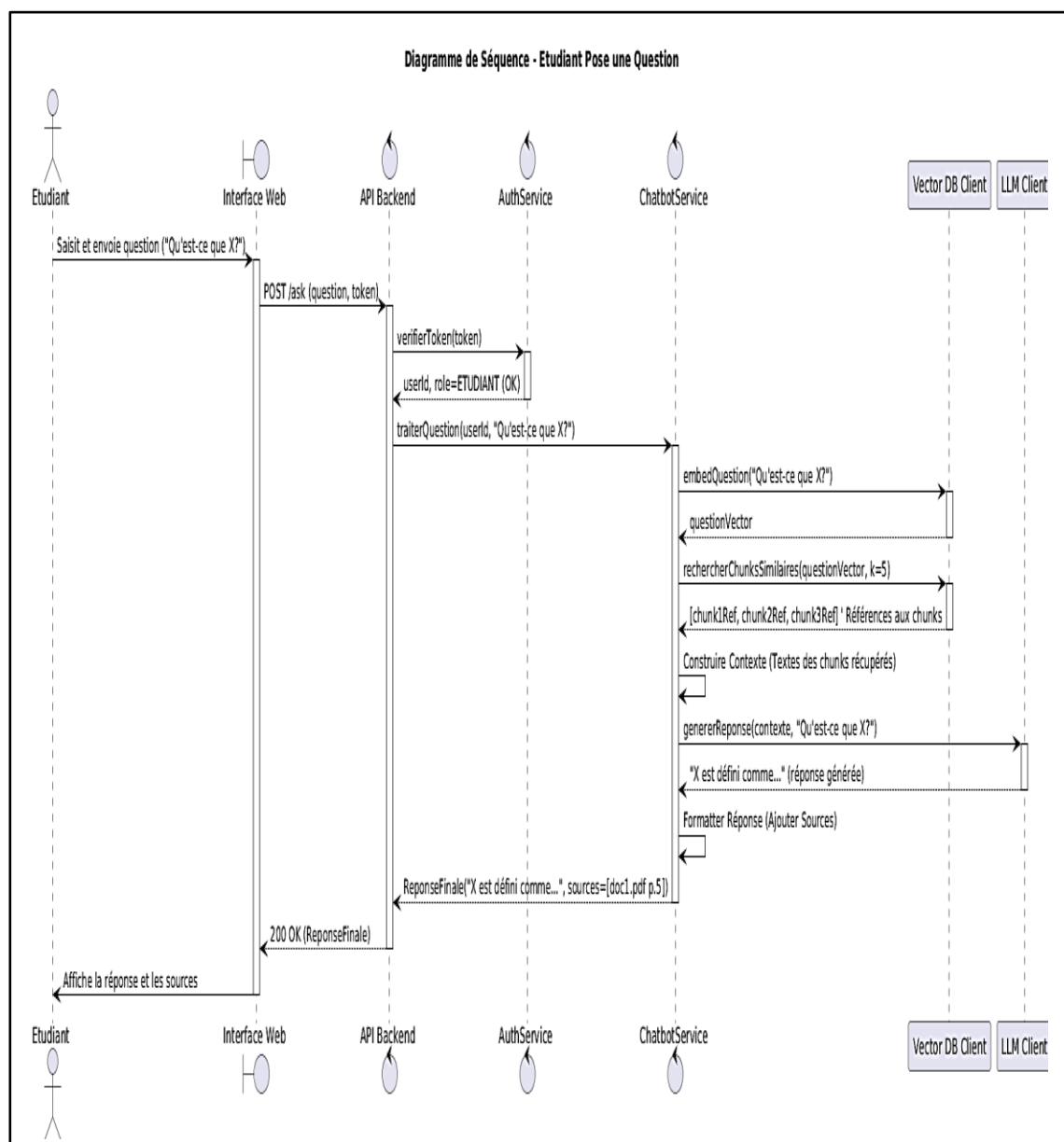
Ce diagramme de séquence décompose méticuleusement le processus de connexion. Il montre comment l'action simple de l'utilisateur (saisir email/mot de passe) déclenche une cascade d'interactions entre le front-end (Interface Web) et différents composants back-end (API Controller, AuthService, UserRepository, SessionRepository). Il illustre clairement la séparation des responsabilités : le contrôleur gère la requête HTTP, le service gère la logique d'authentification, et les dépôts (repositories) gèrent l'accès aux données (utilisateurs et sessions). De manière cruciale, il visualise les différents chemins possibles grâce aux blocs alt, montrant explicitement comment les erreurs (utilisateur non trouvé, mot de passe incorrect) sont gérées et propagées jusqu'à l'utilisateur final, ainsi que le chemin de succès menant à la création d'une session et à la confirmation de la connexion.



*Figure 10 : Diagramme de séquence Connexion*

### ***b. étudiant Poser une question :***

Ce diagramme illustre le flux complet du traitement d'une question par l'étudiant dans un système RAG. Après une vérification d'authentification initiale, la question est transformée en vecteur. Ce vecteur est utilisé pour rechercher les extraits de documents les plus pertinents dans une base de données vectorielle (phase de Retrieval). Ces extraits forment un contexte qui, avec la question initiale, est envoyé à un LLM pour générer une réponse (phase de Generation). Finalement, cette réponse est enrichie avec les sources des informations utilisées avant d'être renvoyée à l'étudiant via l'interface web. Le diagramme met en évidence la collaboration entre le service chatbot, le service d'authentification, la base de données vectorielle et le modèle de langage pour fournir une réponse contextuelle et sourcée.



*Figure 11 : Diagramme de séquence Poser une question v*

### c. Admin activer/désactiver compte :

Ce diagramme illustre une action administrative sensible : la modification du statut d'un compte utilisateur. Le processus commence par une action de l'administrateur sur l'interface web, qui envoie une requête PATCH sécurisée par un token au backend. Le backend vérifie d'abord rigoureusement l'authentification et les permissions de l'administrateur. Ensuite, il tente de trouver l'utilisateur cible. S'il est trouvé, une vérification cruciale empêche l'administrateur de se désactiver lui-même. Si tout est en ordre, le statut de l'utilisateur cible est modifié et sauvegardé en base de données. Finalement, une réponse (succès avec les données mises à jour, ou erreur spécifique 404/403) est renvoyée au frontend pour informer l'administrateur du résultat de son action. Le diagramme met en lumière les étapes de validation, de logique métier et de persistance nécessaires pour cette opération.

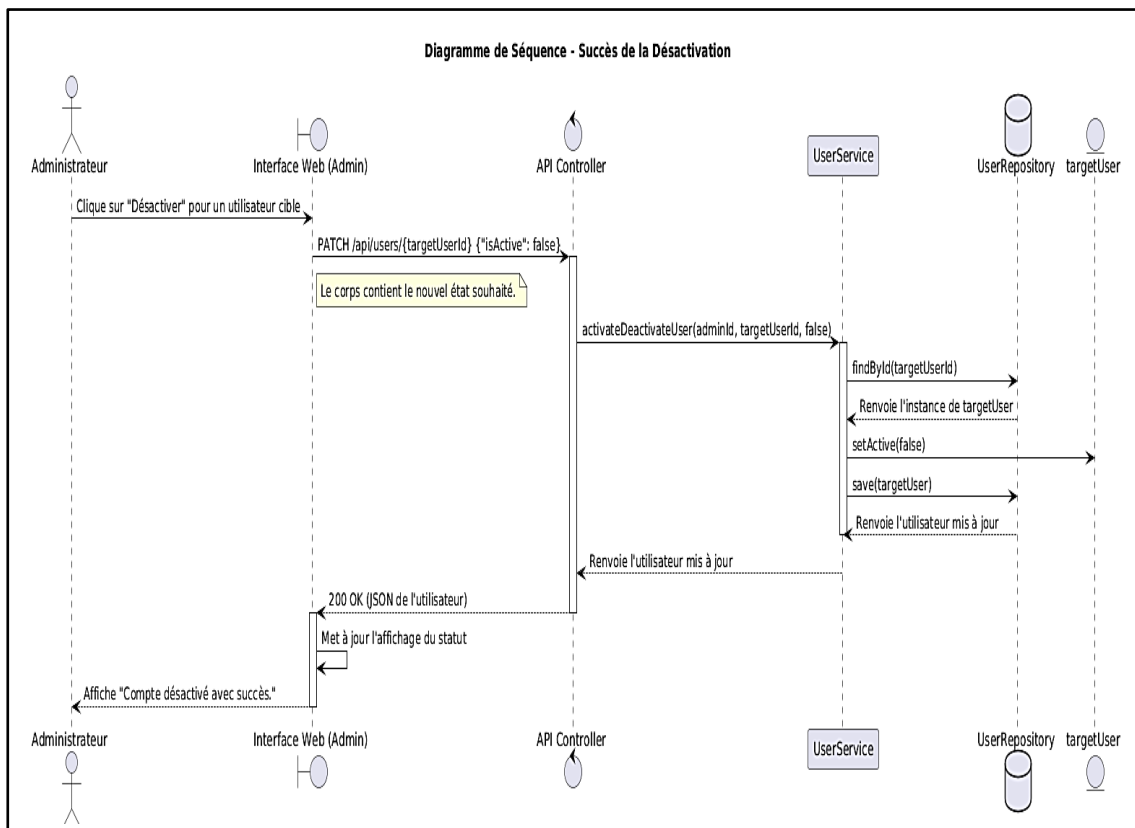
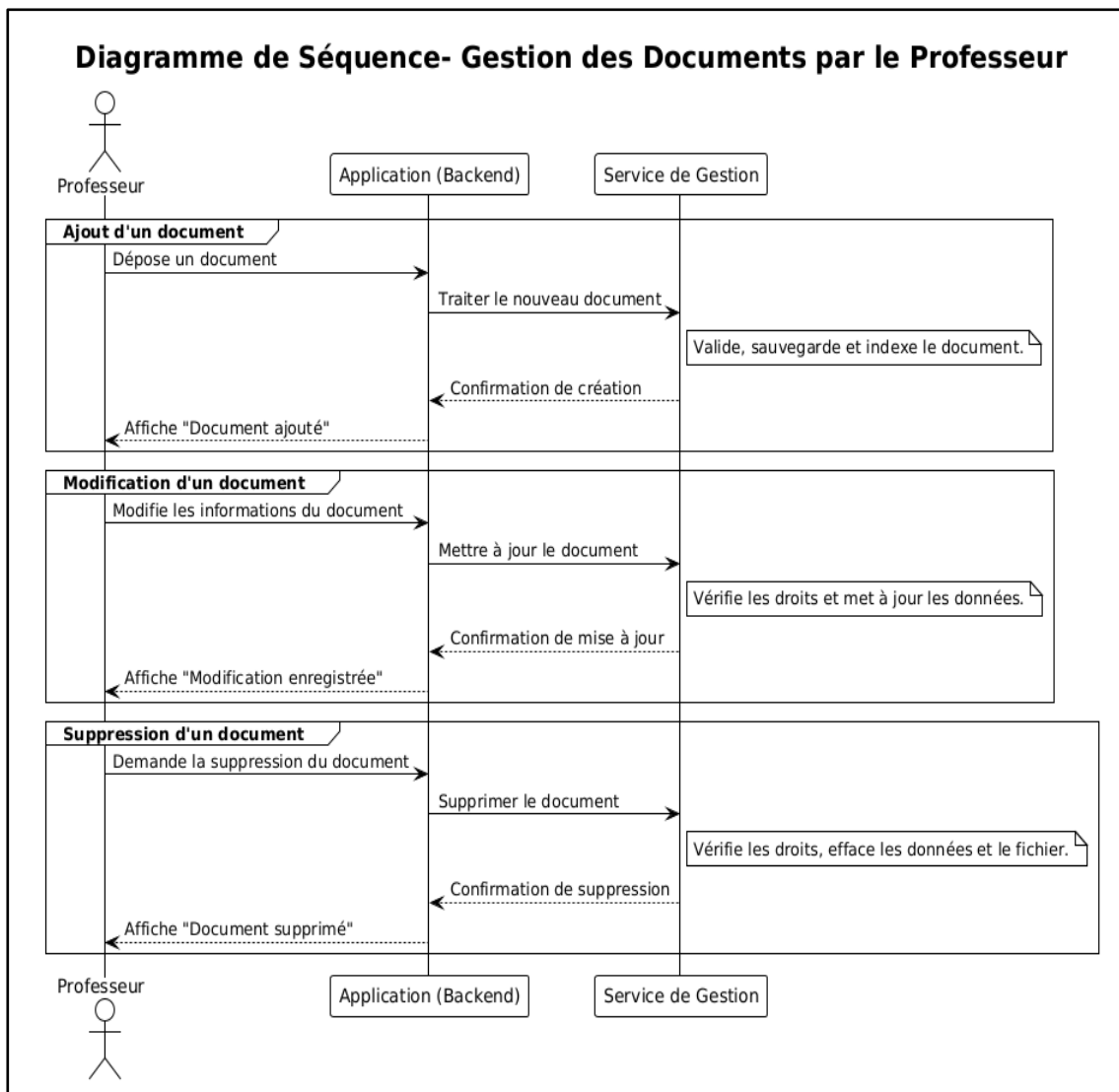


Figure 12 : Diagramme de séquence Gérer les compte

#### *d. Professeur Gérer les document :*

Ce diagramme de séquence très complet détaille les trois opérations de gestion de documents par un professeur. Chaque opération suit un schéma similaire : action de l'utilisateur, requête API avec token, validation de session/permission, appel au service métier (DocumentManagementService), interaction avec les composants de persistance (DocumentPDFRepository, FileStorage, ModuleRepository) et le service RAG (RAGPipelineService), et enfin retour d'une réponse (succès ou erreur) au client. Le diagramme met en évidence des points clés : la nécessité de l'authentification, la vérification des permissions (un professeur ne gère que ses documents), la séparation des responsabilités entre les services, et l'interaction avec le stockage de fichiers et le pipeline RAG (notamment l'indexation asynchrone lors de l'ajout et la désindexation lors de la suppression). Les blocs alt et opt clarifient les différents scénarios possibles et la gestion des erreurs.



*Figure 13 : Diagramme de séquence Gérer les Documents*

### iii. Diagramme d'Activité:

Le diagramme d'activité illustre le déroulement des processus métiers ou des opérations. Il met en évidence les différentes étapes d'une activité, les décisions, les parallélismes et les boucles.

Ce diagramme d'activité illustre de manière claire et structurée le pipeline complexe d'ingestion d'un document PDF soumis par un professeur. Après la soumission et la validation des permissions, le fichier brut est stocké, et ses métadonnées sont enregistrées dans une base de données relationnelle. Parallèlement (ou séquentiellement), le contenu textuel du PDF est extrait. Si l'extraction réussit, le texte est découpé en "chunks", et chaque chunk est transformé en "embedding" vectoriel. Ces paires chunk/embedding sont ensuite indexées dans une base de données vectorielle spécialisée, rendant le contenu interrogeable par le système RAG. Le diagramme met bien en évidence les différentes responsabilités (Interface Web, Backend, Stockage Fichier, BD Relationnelle, BD Vectorielle) et les points de décision critiques (validation, succès/échec de l'extraction de texte) ainsi que les issues possibles (succès, erreur d'authentification, erreur de parsing).

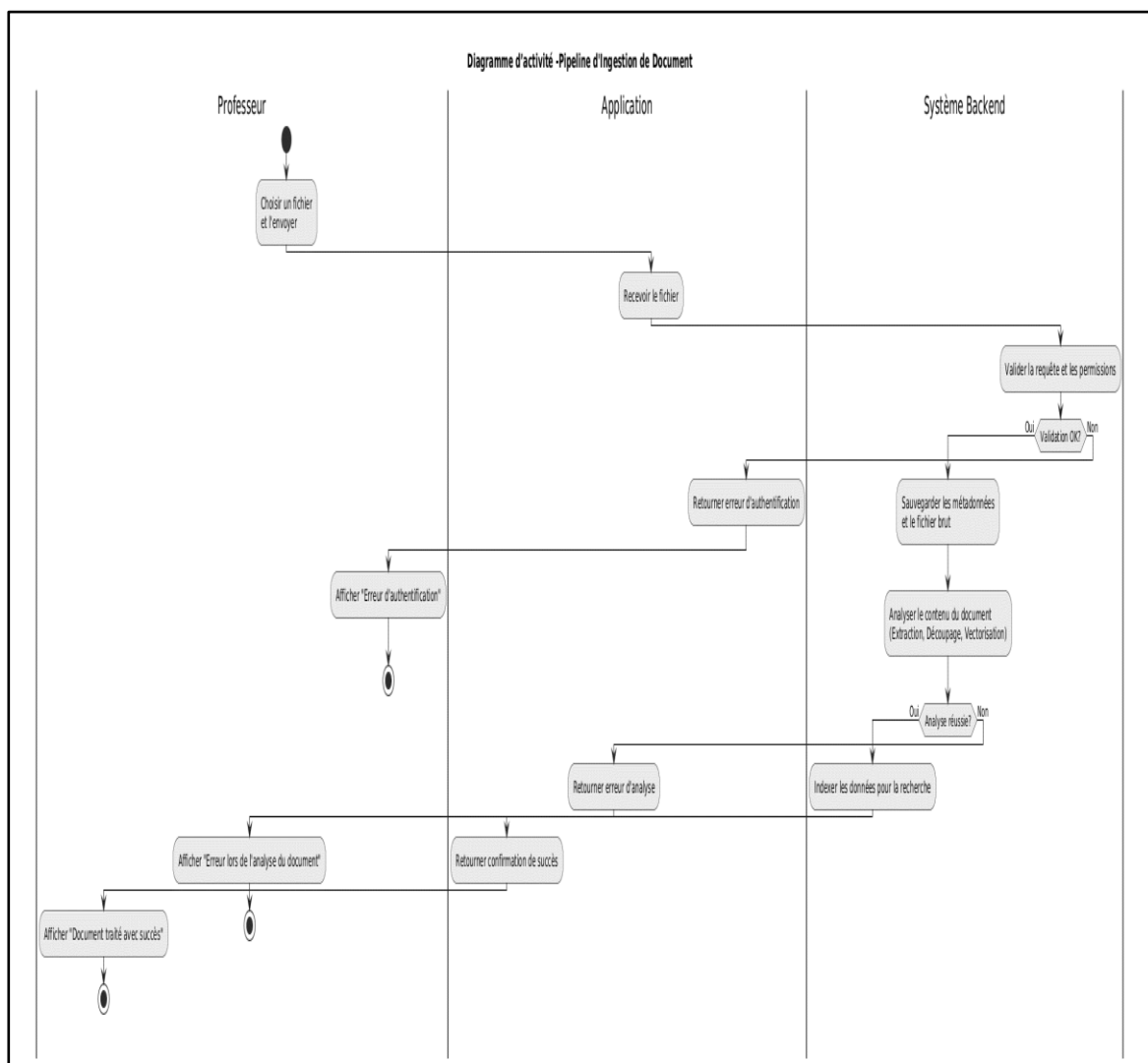


Figure 14 : Diagramme d'activité



### iii. Diagramme d'état de transitions:

Le diagramme d'états-transitions (ou diagramme d'états) modélise les différents états d'un objet au cours de sa vie, ainsi que les transitions déclenchées par des événements ou des conditions spécifiques.

Ce diagramme d'état montre le cycle de vie typique d'une session d'un étudiant avec le chatbot. Il commence dans un état NonAuthentifié, nécessitant une Connexion valide pour entrer dans l'état composite Authentifié. Une fois authentifié, l'étudiant arrive sur une PageAccueil d'où il peut naviguer vers la ConsultationCours (et revenir) ou initier une interaction avec le chatbot via PoserQuestion. Poser une question le fait passer par les états AttenteRéponse et AffichageRéponse avant de revenir à la PageAccueil. Le diagramme montre également deux façons de terminer la session : une action de Déconnexion depuis la PageAccueil et une Fermeture de session plus générale depuis l'état Authentifié global (ou même depuis NonAuthentifié). Ce diagramme simplifie le processus en se concentrant sur les états majeurs du flux de l'étudiant.

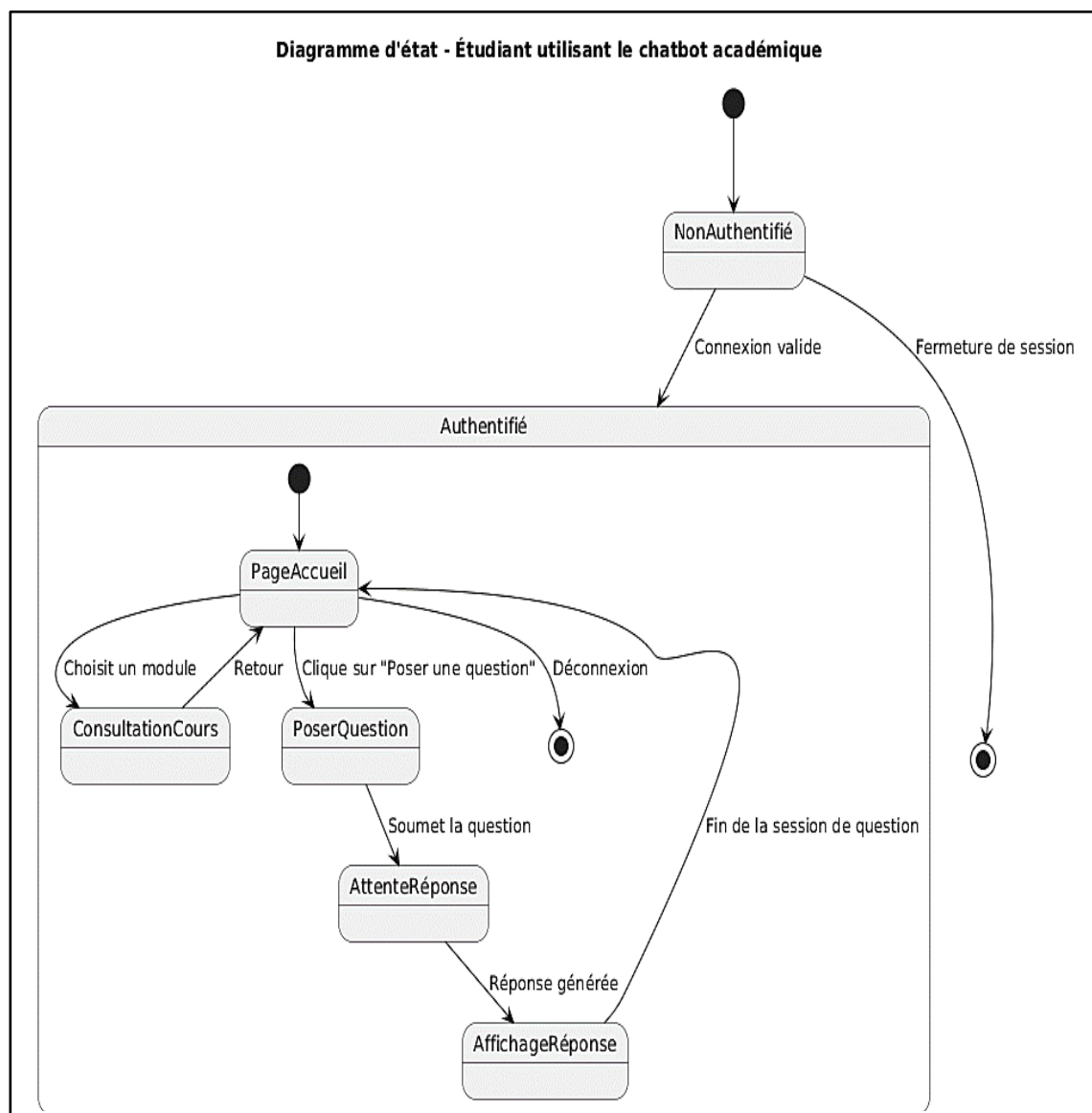


Figure 15 : Diagramme d'état de transitions

#### iv. Diagramme de classes:

Le diagramme de classes représente la structure statique d'un système. Il montre les classes, leurs attributs, leurs méthodes et les relations entre elles (associations, héritages, dépendances, etc.).

Ce diagramme illustre la structure simple de notre chatbot. Nous avons l'Utilisateur, qui peut être un Étudiant, un Professeur (qui a la capacité spécifique de gérer les documents), ou un membre de l'Administration (qui peut gérer les comptes utilisateurs). Au cœur du système se trouve le SystemeRAG, le moteur intelligent qui génère les réponses. Quand un Utilisateur pose une question, le SystemeRAG cherche l'information pertinente dans les Documents Cours. Il crée ensuite une réponse sous forme de Message, qui est ajoutée à la Conversation de l'Utilisateur. Ainsi, l'Utilisateur interagit avec le système via des Conversations contenant des Messages, et le SystemeRAG utilise les Documents pour fournir les réponses.

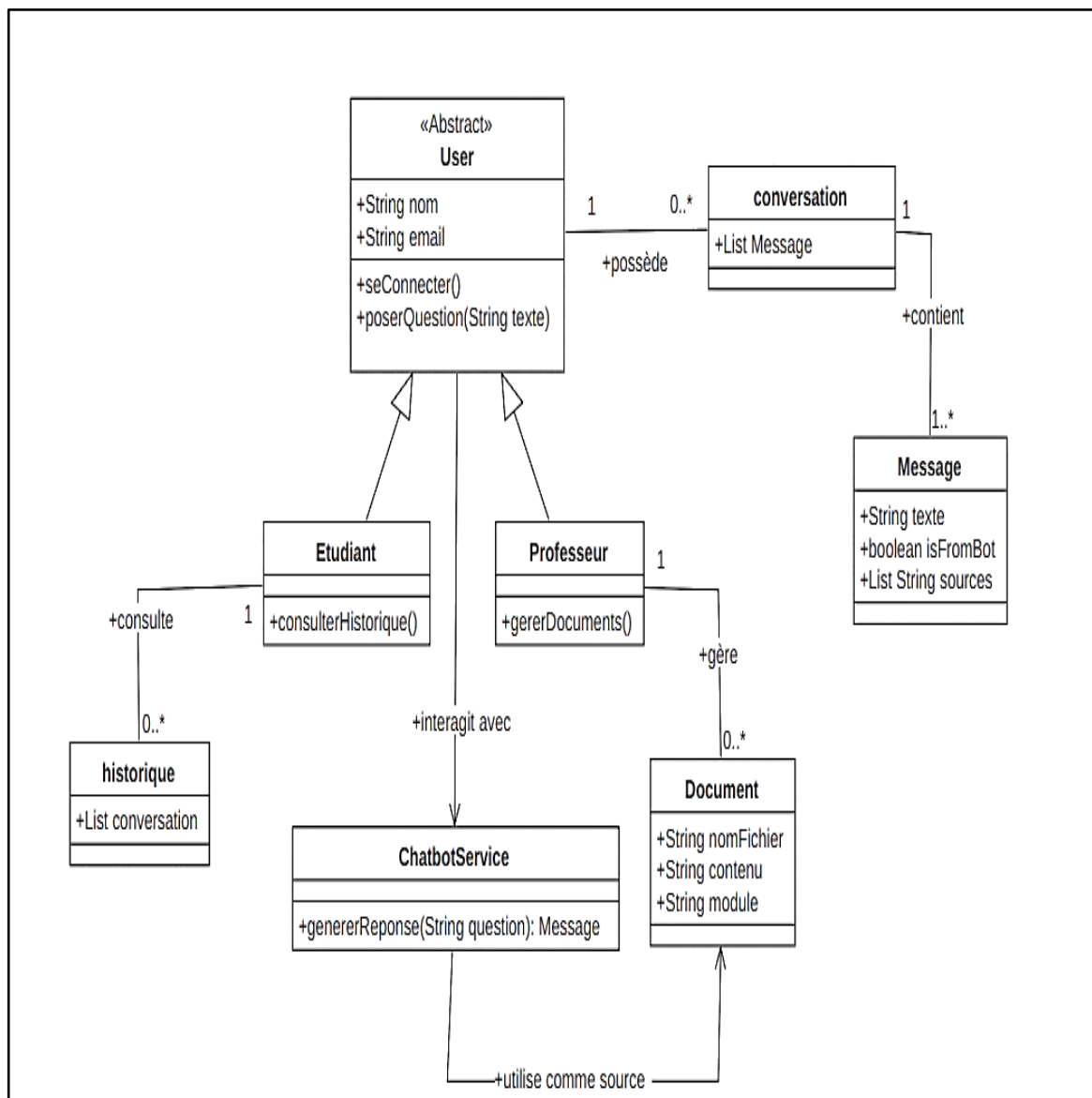


Figure 16 : Diagramme des classes

Nom de la Classe	Attribut / Méthode	Type	Description
<b>User (abstraite)</b>	nom	String	Nom de l'utilisateur
	email	String	Adresse e-mail de l'utilisateur
	seConnecter()	Méthode	Permet à l'utilisateur de se connecter
	poserQuestion(String texte)	Méthode	Permet à l'utilisateur de poser une question
<b>Etudiant</b>	consulterHistorique()	Méthode	Permet de consulter l'historique des conversations
<b>Professeur</b>	gererDocuments()	Méthode	Permet de gérer les documents (ajout, suppression, etc.)
<b>ChatbotService</b>	genererReponse(String question)	Message (Méthode)	Génère une réponse à partir d'une question
<b>Message</b>	texte	String	Contenu textuel du message
	isFromBot	boolean	Vrai si le message est généré par le bot
	sources	List	Liste des sources utilisées pour générer le message
<b>Document</b>	nomFichier	String	Nom du fichier
	contenu	String	Contenu textuel du document
	module	String	Module auquel appartient le document
<b>conversation</b>	messages	List	Liste des messages constituant la conversation
<b>historique</b>	conversation	List	Conversations précédemment consultées par l'étudiant

## 4. Conclusion :

Dans cette partie, nous avons présenté l'analyse et la conception de notre système de chatbot RAG pour les modules FSBM. Nous avons notamment détaillé les interactions entre les différents types d'utilisateurs (Étudiant, Professeur, Administration) et les composants logiciels clés (ChatbotService, RAGSystem, Repositories, etc.) à travers les diagrammes de cas d'utilisation pour la vue fonctionnelle, le diagramme de classes pour la structure statique, et des diagrammes de séquence illustrant des scénarios dynamiques essentiels. Ces modélisations ont permis de mieux comprendre le déroulement des processus, en particulier le pipeline d'ingestion des documents et le flux de réponse basé sur le RAG. Cette conception détaillée sera exploitée comme un plan directeur fiable lors de la phase de réalisation et d'implémentation de notre application.

# Chapitre V

## Réalisation du Chatbot

## 1. Introduction

Ce chapitre présente la mise en œuvre concrète de notre chatbot académique. Après la phase de conception, nous avons procédé au développement technique en suivant une approche agile. Cette réalisation s'appuie sur un pipeline RAG, combinant un modèle LLAMA 3 et une base de connaissances vectorielle issue des supports de cours.

Les étapes clés incluent le prétraitement des documents pédagogiques, la vectorisation du contenu, l'intégration du modèle de génération, ainsi que le développement de l'interface web. Les sections suivantes détaillent ces étapes, les outils utilisés et les résultats obtenus.

## 2. Technologies et Outils Utilisés :

### i. Les outils :

#### *Visual studio code*



Visual Studio (VS) Code est un éditeur de code open-source principalement utilisé pour corriger et réparer les erreurs de codage des applications Cloud et web. VS Code est développé par Microsoft et prend en charge les systèmes d'exploitation MacOS, Linux et Windows. Les outils de VS Code peuvent être utilisés pour améliorer la fonctionnalité de tout code écrit.

#### *Postman*



Postman est un outil de développement d'API (interface de programmation d'application) qui permet de créer, de tester et de modifier des API. Presque toutes les fonctionnalités dont un développeur pourrait avoir besoin sont encapsulées dans cet outil.

#### *MySQL*



MySQL est un système de gestion de bases de données relationnelles SQL open source développé et supporté par Oracle. Le SQL dans "MySQL" signifie "Structured Query Language" : le langage standard pour les traitements de bases de données.

#### *IntelliJ IDEA*



IntelliJ IDEA est un environnement de développement intégré (IDE) Java créé par JetBrains. En mettant l'accent sur la productivité et l'efficacité, IntelliJ IDEA propose une large gamme de fonctionnalités et d'outils pour aider les développeurs à écrire, tester et déboguer leur code plus rapidement et plus facilement. De la détection automatique des erreurs à la restructuration intelligente

du code, nous l'avons utilisé comme IDE pour programmer le Back-end de notre Chatbot.

### ***Figma***



Figma est une application web d'édition graphique qui permet le partage en temps réel sur le même fichier, ce qui signifie que toutes les parties prenantes du projet peuvent interagir et collaborer sur le projet en tenant compte de chaque mise à jour et changement. Cela permet évidemment de gagner du temps et d'augmenter l'efficacité.

### ***Hugging Face***



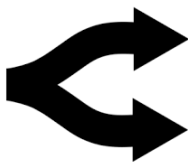
Hugging Face est une plateforme et une communauté qui fournissent des outils pour créer des Datasets, entraîner et déployer des modèles d'apprentissage automatique basés sur la technologie et le code open source. Elle offre également un espace permettant aux chercheurs, ingénieurs et passionnés d'IA de se réunir pour échanger des idées, obtenir du soutien et contribuer à des projets open source.

### ***StarUML***



Logiciel de modélisation open source utilisé dans Unified Modeling Language où la modélisation de systèmes et de logiciels prend en charge les concepts UML en fournissant différents types de diagrammes UML pour générer du code dans différents langages afin de développer des plates-formes rapides et flexibles pour répondre aux besoins et aux exigences.

### ***OpenRouter.ai***



OpenRouter.ai est une plateforme qui agit comme un agrégateur et un point d'accès unifié à une vaste gamme de modèles d'intelligence artificielle (IA), notamment des grands modèles de langage (LLM) provenant de divers fournisseurs (comme OpenAI, Anthropic, Google, etc.). Son principal objectif est de simplifier l'accès et l'utilisation de ces modèles pour les développeurs en leur offrant une API unique. Plutôt que de gérer de multiples intégrations API et clés distinctes, les développeurs peuvent utiliser OpenRouter pour choisir et interagir avec différents modèles de manière flexible, souvent avec un système de crédits unifié, facilitant ainsi l'expérimentation et le déploiement d'IA dans leurs applications.

## ii. Les Technologies :

### *Python*



Python est le langage de programmation informatique le plus populaire et le plus utilisé, notamment dans le domaine de la Data Science et du Machine Learning. De plus, Python est un langage multiplateforme qui fonctionne sur divers systèmes d'exploitation, ce qui en fait un choix idéal pour les développeurs travaillant sur différents environnements.

### *HTML*



HTML signifie « HyperText Markup Language » qu'on peut traduire par « langage de balise pour l'hypertexte ». Il est utilisé afin de créer et de représenter le contenu d'une page web et sa structure. Nous avons utilisé ce langage pour élaborer la structure principale des pages de l'interface.

### *Tailwind CSS*



Tailwind CSS est un framework CSS distinctif par son approche "utility first" (axée sur les utilitaires). Plutôt que de fournir des composants d'interface utilisateur préconçus, il met à disposition une vaste collection de classes CSS de bas niveau, hautement composables et à usage unique, qui s'appliquent directement dans le code HTML pour construire des designs personnalisés. Cette méthodologie permet une personnalisation poussée et un développement rapide d'interfaces uniques, tout en garantissant des fichiers CSS finaux optimisés et légers grâce à l'élimination automatique des styles non utilisés lors de la compilation pour la production.

### *TypeScript*



C'est un langage open source, développé comme un sur-ensemble de JavaScript. Ce qu'il faut comprendre, c'est que tout code valide en JavaScript l'est également en TypeScript. Cependant, ce langage introduit des fonctionnalités optionnelles comme le typage ou la programmation orientée objet.

## Spring



Spring Framework fournit un modèle complet de programmation et de configuration pour les applications d'entreprise modernes basées sur Java, sur tout type de plate-forme de déploiement. Un élément clé de Spring est le support de l'infrastructure au niveau des applications : Spring se concentre sur la « plomberie » des applications d'entreprise afin que les équipes puissent se concentrer sur le logique métier au niveau des applications, sans liens inutiles avec des environnements de déploiement spécifiques. Et avec ce Framework nous avons utilisé plusieurs dépendances : Spring data JPA, oauth2 ressource server, Lombok.

## JAVA



Java est un langage de programmation originellement proposé par Sun Microsystems et maintenant par Oracle depuis son rachat de Sun Microsystems en 2010. Java a été conçue avec deux objectifs principaux :

- Offrir un langage orienté objet avec une bibliothèque standard riche
- Permettre aux développeurs d'écrire des logiciels indépendants de l'environnement hardware d'exécution.

## 3. Préparation du Dataset

Dans cette section, nous explorerons les étapes nécessaires pour préparer un ensemble de données à partir d'une collection de fichiers, d'articles et de revues, extraire le texte pertinent et formater l'ensemble de données pour qu'il soit conforme à LLAMA 4. Ce processus est essentiel pour aligner les programmes LLM.

### i. Regroupement des données

#### a) Identification des Sources de Données

Nous détaillerons comment identifier et collecter les différentes sources de données pertinentes pour notre projet d'assistant de cours FSBM. Nos sources cibles comprennent :

- Documents de cours universitaires (S1 à S6)

Nous avons collecté l'ensemble des supports pédagogiques utilisés dans les filières scientifiques de la Faculté des Sciences Ben M'Sik, couvrant les six semestres de licence. Ces documents constituent la base de connaissances de notre système RAG.

- Sources principales :

- Google Drive partagé par les enseignants de la FSBM
- Collections personnelles d'étudiants de confiance ayant validé l'authenticité des contenus
- Supports officiels fournis par les différents départements de la faculté



➤ Types de documents collectés :

- Polycopiés de cours au format PDF
- Présentations PowerPoint (.ppt, .pptx) utilisées en amphi
- Documents Word (.doc, .docx) contenant des exercices et compléments de cours
- Fichiers texte (.txt) avec des notes de cours structurées

➤ Couverture académique :

- Filières ciblées : SMI (Sciences Mathématiques et Informatique), SMA (Sciences Mathématiques et Applications), et autres filières scientifiques
- Semestres : S1 à S6 (cycle licence complet)
- Modules : Informatique (Java, JEE, Bases de données), Mathématiques, Physique, et autres matières fondamentales

### ***b) Extraction des ressources***

Contrairement aux projets nécessitant du web scraping, notre approche se base entièrement sur des documents locaux déjà disponibles. Cette méthode garantit la fiabilité et l'authenticité des contenus académiques.

Stratégie d'extraction :

Nous avons utilisé des bibliothèques Python spécialisées pour chaque type de document, intégrées dans notre pipeline d'ingestion automatisée :

- PyPDFLoader : Pour l'extraction de texte des fichiers PDF contenant les polycopiés de cours. Cette bibliothèque nous permet de récupérer le contenu textuel tout en préservant la structure documentaire.
- UnstructuredPowerPointLoader : Dédiée au traitement des présentations PowerPoint, elle extrait le texte des diapositives en maintenant la cohérence pédagogique.
- UnstructuredWordDocumentLoader : Pour les documents Word, particulièrement utile pour les exercices et les compléments de cours rédigés par les enseignants.
- UnstructuredFileLoader : Utilisée comme solution de fallback pour les formats moins courants ou les fichiers texte simples.

Particularités techniques :

- Traitement exclusif de documents numériques (pas de reconnaissance OCR nécessaire)
- Système de cache basé sur le hashage SHA-256 pour éviter le retraitement de documents identiques
- Traitement parallélisé avec ThreadPoolExecutor pour optimiser les performances

### ***c) Organisation des ressources***

Nous avons structuré notre corpus documentaire selon une hiérarchie logique reflétant l'organisation académique de la FSBM :

**docs/**

```
└─ smi/
  │ └─ s1/
  │   │ └─ mathematiques/
  │   │   │ └─ cours_analyse.pdf
  │   │   │   └─ td_algebre.docx
  │   │   └─ informatique/
  │   │       └─ intro_programmation.ppt
  │   │       └─ algorithmique.pdf
  │   └─ s2/ ... s6/
  └─ sma/
    │ └─ s1/ ... s6/
    └─ autres_filieres/
        └─ ...
```

➤ Organisation par filière

Chaque filière (SMI, SMA, etc.) dispose de son propre répertoire, facilitant la navigation et le filtrage des contenus selon la spécialisation de l'étudiant

➤ Structuration par semestre

Les documents sont classés de S1 à S6, permettant une progression pédagogique logique et un accès ciblé selon le niveau d'études.

➤ Catégorisation par module

Chaque semestre contient les modules correspondants (Bases de données, Java, JEE, Mathématiques, etc.), offrant une granularité fine pour les requêtes spécialisées.

➤ Diversité des formats

Tous les types de documents pédagogiques sont supportés dans chaque module, préservant la richesse des supports d'enseignement originaux.

## ii. Structuration et Nettoyage

### a) *Les Outils d'Extraction de texte*

Notre pipeline de traitement utilise la bibliothèque LangChain pour une approche unifiée et robuste :

- Extraction des fichiers PDF : Utilisation de PyPDFLoader qui gère efficacement les documents académiques en français, en préservant la mise en forme et les caractères spéciaux mathématiques.
- Extraction des PowerPoint : UnstructuredPowerPointLoader traite les diapositives en extrayant le texte principal tout en ignorant les éléments de mise en forme non pertinents.
- Extraction des documents Word : UnstructuredWordDocumentLoader récupère le contenu textuel des fichiers .doc et .docx, particulièrement adapté aux exercices et aux compléments de cours.
- Traitement des fichiers texte : Lecture directe avec UnstructuredFileLoader pour les notes de cours au format .txt.

### ***b) Découpage et Prétraitement du Texte***

Configuration du découpage :

**CHUNK\_SIZE = 800**                      **# Tokens par chunk**

**CHUNK\_OVERLAP = 80**                      **# Chevauchement entre chunks**

Nous utilisons le RecursiveCharacterTextSplitter de LangChain avec une stratégie de séparation hiérarchique :

- Séparateurs primaires : Paragraphes (\n\n)
- Séparateurs secondaires : Lignes (\n), phrases (.), mots ( )

Cette approche préserve la cohérence sémantique des contenus académiques français tout en optimisant la récupération d'informations.

Métadonnées enrichies : Chaque chunk est enrichi avec :

- Source du document original
- Module d'appartenance
- Semestre et filière
- Type de document
- Numéro de chunk pour la traçabilité

## **iii. Mise en Forme du Dataset pour le Système RAG**

### ***a) Vectorisation avec FAISS***

Choix du modèle d'embeddings : Nous utilisons intfloat/multilingual-e5-small pour plusieurs raisons adaptées à notre contexte :

- Support multilingue : Optimisé pour le français académique
- Efficacité computationnelle : Compatible avec des ordinateurs portables sans GPU
- Qualité sémantique : Performance équilibrée entre précision et rapidité
- Taille réduite : Déploiement local facilité

Architecture FAISS :

# Configuration de base

```
embeddings = HuggingFaceEmbeddings(  
    model_name="intfloat/multilingual-e5-small",  
    model_kwargs={"device": "cpu"},  
    encode_kwargs={"normalize_embeddings": True}  
)  
  
# Construction de l'index vectoriel  
db = FAISS.from_texts(texts, embeddings, metadatas=metadatas)
```

### ***b) Système de Cache et Optimisation***

Mécanisme de cache intelligent :

- Calcul d'empreinte SHA-256 pour chaque document
- Évitement du retraitement des fichiers non modifiés
- Mise à jour incrémentale de la base vectorielle

Stockage structuré :

**vectorstore/**

```
|— db_faiss/          # Index FAISS principal  
| |— index.faiss      # Données vectorielles  
| |— index.pkl        # Métadonnées et configuration  
|— processed_files.json # Cache des fichiers traités
```

Cette organisation permet des mises à jour efficaces du corpus sans retraitement complet, essentiel pour un système académique en évolution constante.

## **4. Scripts Python de gestion de la mémoire vectorielle :**

Dans cette partie, nous présentons pas-à-pas les deux scripts qui constituent la pipeline “mémoire vectorielle” :

- `create_memory_for_llm_simplified.py` – ingestion, découpage et indexation des documents.
- `connect_memory_with_llm_lighter_cos.py` – application Streamlit qui interroge l'index avec un LLM distant.

Le format reprend exactement la mise en page illustrée dans les captures : flèches pour les titres, blocs de code grisés, légendes numérotées en dessous.

## i. create\_memory\_for\_llm\_simplified.py : création / mise à jour de l'index FAISS

### ➤ Configuration globale

```
DATA_DIR      = "docs"                # Dossier des documents à indexer
FAISS_DIR     = "vectorstore/db_faiss" # Emplacement de l'index
PROCESSED_CACHE = "vectorstore/processed_files.json"
CHUNK_SIZE    = 800                   # ≈ tokens par chunk
CHUNK_OVERLAP = 80                    # chevauchement entre chunks
MAX_WORKERS   = min(4, (os.cpu_count() or 4))
```

Figure 17 : Paramètres globaux

Le bloc de code présente la configuration initiale du script Python chargé de créer ou mettre à jour l'index FAISS. On y voit l'affectation de variables globales : `DATA_DIR` pour le dossier des documents, `FAISS_DIR` pour le chemin de l'index, `PROCESSED_CACHE` pour le cache JSON, puis `CHUNK_SIZE` et `CHUNK_OVERLAP` pour la taille et la superposition des segments, et `MAX_WORKERS` pour le parallélisme.

### ➤ Découpage sémantique et préparation des chunks

```
SPLITTER = RecursiveCharacterTextSplitter(
    chunk_size    = CHUNK_SIZE,
    chunk_overlap = CHUNK_OVERLAP,
    separators    = ["\n\n", "\n", ".", " ", ""],
)

def load_and_split(path: str) -> List[Dict]:
    ...

def sha256(path: str) -> str: ...
def load_cache() -> Dict[str, str]: ...
def save_cache(cache: Dict[str, str]) -> None: ...
def discover(root: str) -> List[str]: ...
```

Figure 18 : Splitter récursif pour la segmentation des

Cette figure affiche un extrait du code définissant `RecursiveCharacterTextSplitter`, un composant chargé de découper les textes en segments cohérents. Les paramètres `chunk_size` et `chunk_overlap` contrôlent respectivement la longueur des segments et leur recouvrement, tandis que la liste `separators` indique les délimiteurs successifs à tester. En-dessous, on trouve les signatures des fonctions utilitaires (`load_and_split`, `sha256`, `load_cache`, `save_cache`, `discover`).

### ➤ Mise en cache Streamlit : embeddings & vectorstore

```
@st.cache_resource
def get_embeddings(): ...

@st.cache_resource
def get_vectorstore(): ...


class OpenRouterLLM(LLM):
    api_key: str
    model: str = "meta-llama/llama-4-scout:free"
    temperature: float = 0.7
    ...
```

Figure 19 : Chargement paresseux des ressources

Le visuel illustre la mise en cache paresseuse au sein d'une application Streamlit. Deux décorateurs `@st.cache_resource` entourent les fonctions `get_embeddings` et `get_vectorstore` pour éviter des calculs redondants. À droite, la classe `OpenRouterLLM` (héritée de `LLM`) expose des attributs tels que la clé API, le modèle (`meta-llama/llama-4-scout:free`), la température et d'autres paramètres de connexion au LLM.

### ➤ Fonctions d'extraction / filtrage + retriever custom

```
def retrieve_relevant_docs(db, user_query: str, embeddings, threshold=0.35, top_k=5): ...

class CustomRetriever(BaseRetriever): ...
```

Figure 20 : Sélection des documents pertinents et retriever dédié

Cette figure présente la fonction `retrieve_relevant_docs`, responsable de filtrer et sélectionner les meilleurs chunks en fonction de la requête utilisateur. Elle prend en entrée la base FAISS, la requête texte, les embeddings, un seuil de similarité et un nombre maximal de résultats (`top_k`). En complément, la classe `CustomRetriever` (étendant `BaseRetriever`) montre comment personnaliser le comportement de récupération. focus

### ➤ Prompt template spécialisé (français, citations)

```
def french_course_prompt() -> PromptTemplate:
    template = (
        "Tu es un assistant intelligent spécialisé ..."
        "Réponse:"
    )
    return PromptTemplate(template=template, input_variables=["context", "question"])


def retrieve_relevant_docs(db, user_query: str, embeddings, threshold=0.35, top_k=5): ...

class CustomRetriever(BaseRetriever): ...
```

Figure 21 : Prompt pour l'assistant universitaire

Ici, on voit la définition de la fonction `french_course_prompt()` qui renvoie un `PromptTemplate`. Le template encapsule un message d'instruction en français ("Tu es un assistant intelligent spécialisé ...") suivi de l'indicateur "Réponse:". Les variables d'entrée (`input_variables`) sont explicitement listées ("context", "question"), permettant d'injecter dynamiquement le contexte et la question posée.

#### iv. Évaluation du système de mémoire vectorielle

Pour évaluer l'efficacité de notre système de mémoire vectorielle et de question-réponse, plusieurs méthodes peuvent être utilisées afin de juger la pertinence et la qualité des réponses fournies par le modèle lors de l'interrogation de la base documentaire. Voici une brève description des méthodes et des critères que nous avons utilisés :

Précision des réponses :

La précision consiste à vérifier si la réponse donnée par le système correspond bien à l'information recherchée par l'utilisateur. On pose des questions précises, puis on vérifie si la réponse est correcte et directement liée au contenu indexé.

Pertinence :

La pertinence mesure si la réponse, même si elle n'est pas parfaite, reste utile et en lien avec la question posée. Une réponse pertinente est généralement extraite ou inspirée d'un document présent dans la base, ou fait référence à des informations fiables.

Justification par citation :

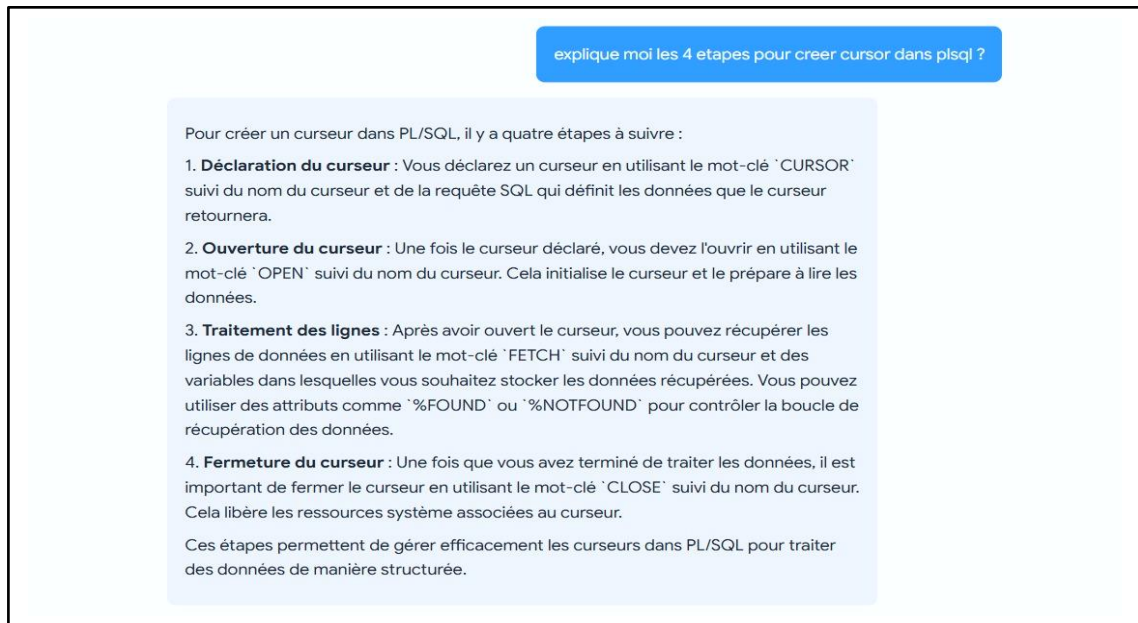
On vérifie également si le système cite correctement la source (le document ou l'extrait) à partir duquel la réponse a été générée. Cela permet à l'utilisateur de remonter à la source et de valider la véracité de la réponse.

Expérience utilisateur :

Il est aussi important d'évaluer la simplicité d'utilisation de l'interface, la rapidité des réponses, et la compréhension des questions même lorsqu'elles sont formulées différemment (robustesse).

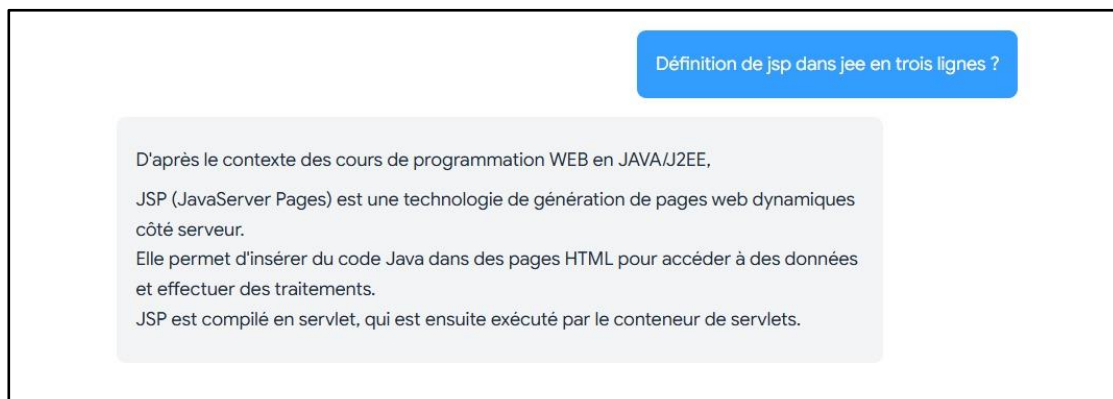
##### *a) Quelques réponses du modèle réglé :*

L'image présente une réponse à la question « explique moi les 4 étapes pour créer un curseur dans PL/SQL ? ». Elle décrit brièvement les étapes suivantes : **déclaration du curseur** avec une requête SQL, **ouverture du curseur** à l'aide du mot-clé `OPEN`, **lecture des données** ligne par ligne avec `FETCH`, puis **fermeture du curseur** avec `CLOSE` pour libérer les ressources. Ces étapes permettent de gérer efficacement les données dans PL/SQL.



*Figure 22 : Exemple de question sur les curseurs en PL/SQL*

L'image montre une interaction où l'utilisateur pose la question : « Définition de JSP dans JEE en trois lignes ? ». En réponse, une brève explication est donnée : JSP (JavaServer Pages) est une technologie de Java EE permettant de générer des pages web dynamiques côté serveur. Elle permet d'insérer du code Java dans des pages HTML pour accéder à des données et exécuter des traitements. Les JSP sont compilées en servlets et exécutées par le conteneur de servlets. Cette réponse illustre une explication synthétique d'un concept en développement web Java.



*Figure 23 : Exemple de question sur JSP dans un contexte JEE*



## 5. Développement de l'Interface du Chatbot :

Lors du développement de cette application web pour l'Assistant Académique FSBM, nous avons mis tout notre intérêt à créer une interface utilisateur facile à naviguer et à comprendre, afin d'offrir une bonne expérience à l'étudiant.

Nous avons nommé notre Chatbot « FSBM Scholar Assistant » (ou un nom de votre choix, ici j'en propose un en lien avec le contexte). Ce choix de nom vise à refléter directement sa fonction d'aide aux études pour les étudiants de la Faculté des Sciences Ben M'Sik et s'inspire du modèle de langage sous-jacent, LLAMA-4-Scout, pour sa capacité à explorer et à fournir des informations pertinentes.

Pour concevoir l'apparence de notre interface, nous avons utilisé Streamlit. Comme défini précédemment, Streamlit est un framework Python qui permet de créer rapidement des applications web interactives. Son approche centrée sur Python et sa simplicité d'utilisation nous ont permis de prototyper et d'itérer efficacement sur la conception de l'interface, en nous concentrant sur la fonctionnalité et l'expérience utilisateur plutôt que sur les complexités du développement web traditionnel. Cela a facilité la collaboration au sein de l'équipe pour unifier nos idées et opinions sur la présentation optimale des informations et des interactions.

### i. Développement du Back-end :

Le back-end de notre Assistant Académique FSBM est une application robuste développée avec le framework Spring Boot. Ce choix a été motivé par la maturité de Spring Boot, son écosystème riche, sa facilité de configuration et sa capacité à créer des API RESTful performantes et sécurisées, essentielles pour la communication avec le front-end (développé avec Streamlit) et la gestion de la logique métier complexe de notre système RAG.

L'architecture du back-end suit les principes de conception en couches pour assurer la modularité, la testabilité et la maintenabilité du code. Les principaux packages et leurs responsabilités sont les suivants :

#### I. *FsbmChatbotBackendApplication (Point d'Entrée) :*

- La classe principale :

`com.fsbmchatbot.fsbmchatbotbackend.FsbmChatbotBackendApplication` initialise et démarre l'application Spring Boot.

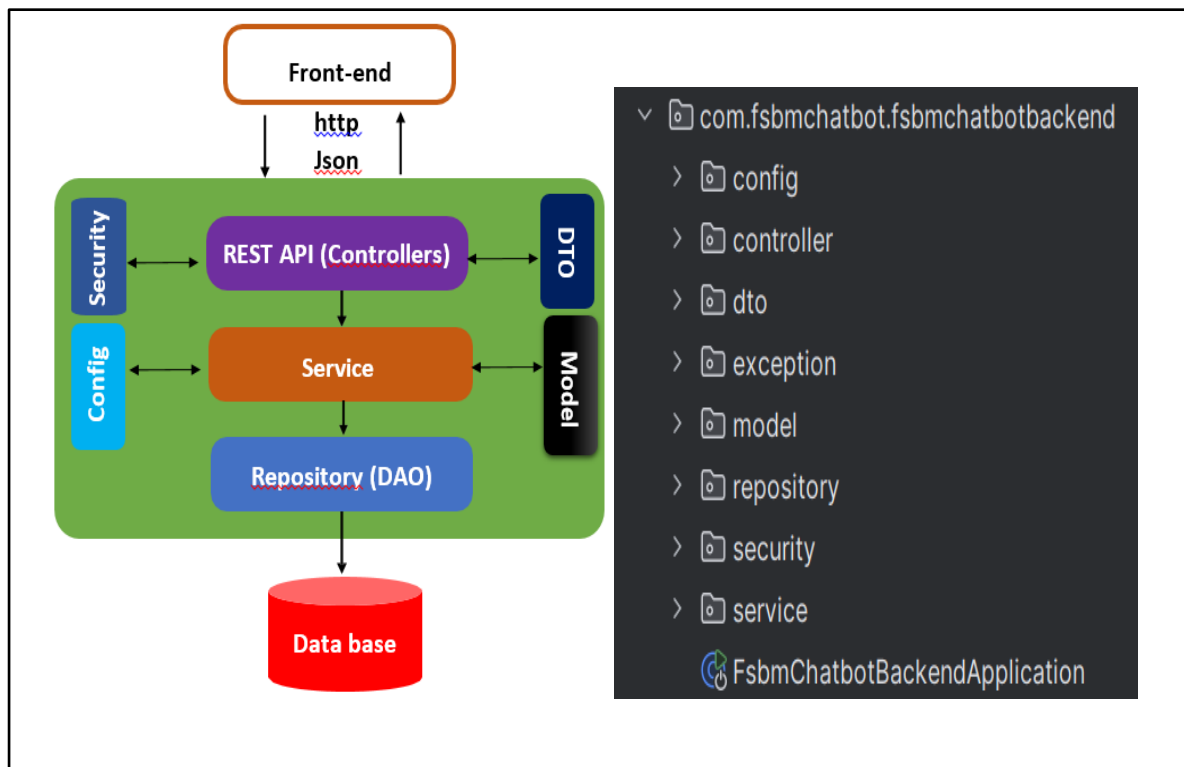


Figure 24 : Schéma de l'Architecture Backend et Structure des Packages

### ✓ *config (Configuration)*

**ChatController.java:** Responsable des interactions de chat. Reçoit les questions des utilisateurs, les transmet à ChatService (qui implémente la logique RAG) et renvoie les réponses générées.

**DocumentController.java:** Gère les opérations CRUD (Create, Read, Update, Delete) sur les documents pédagogiques. Utilisé principalement par les professeurs pour uploader, lister, ou supprimer des documents de cours. Il interagit avec DocumentService.

**FiliereController.java, ModuleController.java:** Gèrent les opérations CRUD pour les filières et les modules, permettant une organisation structurée des documents.

**AdminController.java:** Fournit des endpoints spécifiques aux administrateurs pour la gestion des utilisateurs (activation/désactivation, attribution de rôles).

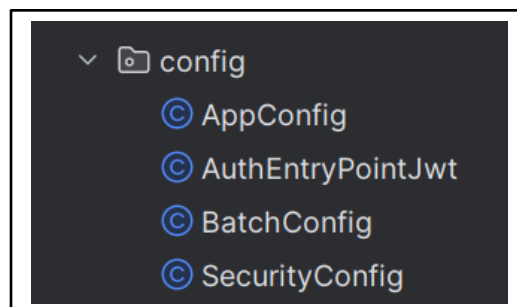


Figure 25 : Schéma de l'architecture configuration

### ✓ **controller (Couche de Contrôle - API REST) :**

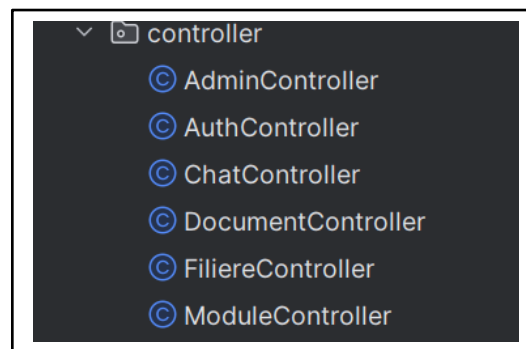
Cette couche expose les points de terminaison (endpoints) de notre API RESTful. Les contrôleurs reçoivent les requêtes HTTP du front-end (développé en Streamlit), les délèguent aux services appropriés pour traitement, et retournent les réponses au client.

**AuthController.java** : Gère les requêtes liées à l'authentification (inscription, connexion, validation d'email).

**ChatController.java** : Traite les requêtes de chat des utilisateurs, orchestrant l'interaction avec la logique RAG pour fournir des réponses contextuelles.

**DocumentController.java** : Permet aux professeurs de gérer les documents pédagogiques (upload, consultation, suppression).

**AdminController.java**, **FiliereController.java**, **ModuleController.java** : Fournissent des fonctionnalités de gestion pour les administrateurs, les filières et les modules respectivement.



*Figure 26 : Schéma de l'architecture Controller*

### ✓ **service (Couche de Service - Logique Métier)**

Cette couche contient la logique métier principale de l'application. Les services sont responsables de la coordination des opérations, de l'application des règles de gestion, et de l'interaction avec la couche de persistance (repositories).

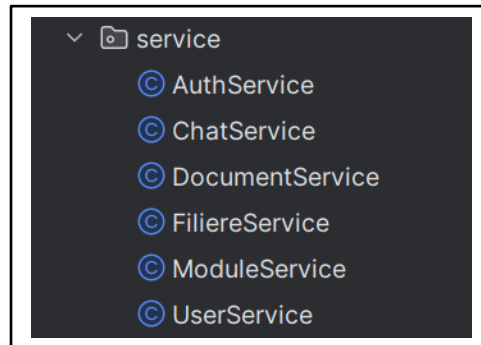
**AuthService.java** : Implémente la logique d'authentification et de gestion des sessions utilisateur.

**ChatService.java** : Représente le cœur de notre système RAG. Il est responsable de :

1. La réception de la question de l'utilisateur.
2. La vectorisation de la question.
3. La recherche des chunks de documents pertinents dans la base de données vectorielle (FAISS).
4. La construction du prompt enrichi pour le LLM.
5. L'appel au LLM externe (LLAMA-4-Scout via OpenRouter).
6. La mise en forme de la réponse et des sources pour le front-end.

**DocumentService.java** : Gère la logique métier liée aux documents, incluant le stockage des fichiers, l'extraction de métadonnées, et potentiellement le déclenchement de leur indexation dans le système RAG.

**UserService.java, FiliereService.java, ModuleService.java** : Implémentent les fonctionnalités spécifiques à la gestion des utilisateurs, des filières et des modules.



*Figure 27 : Schéma de l'architecture Service*

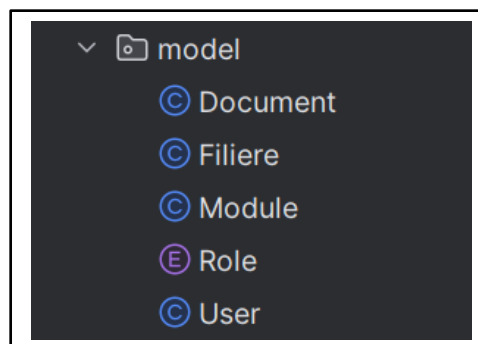
### ✓ **model (Couche Modèle - Entités du Domaine)**

Ce package contient les classes Java (POJOs) qui représentent les entités de notre domaine. Ces classes sont généralement annotées avec JPA pour le mapping objet-relationnel avec la base de données.

**User.java** : Représente les informations d'un utilisateur (étudiant, professeur, admin).

**Document.java** : Décrit les métadonnées d'un document pédagogique.

**Filiere.java, Module.java, Role.java** : Définissent les structures pour les filières, modules et rôles.



*Figure 28 : Schéma de l'architecture Model*

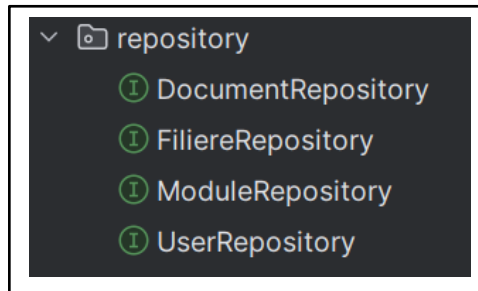
### ✓ **repository (Couche d'Accès aux Données - anciennement DAO)**

Cette couche est responsable de l'interaction avec la base de données relationnelle. Nous utilisons Spring Data JPA, où les interfaces de repository étendent JpaRepository pour fournir des opérations CRUD standardisées et la possibilité de définir des requêtes personnalisées.

**UserRepository.java** : Gère la persistance des entités User.

**DocumentRepository.java** : S'occupe de la persistance des métadonnées des Document.

**FiliereRepository.java, ModuleRepository.java** : Pour les entités Filiere et Module.



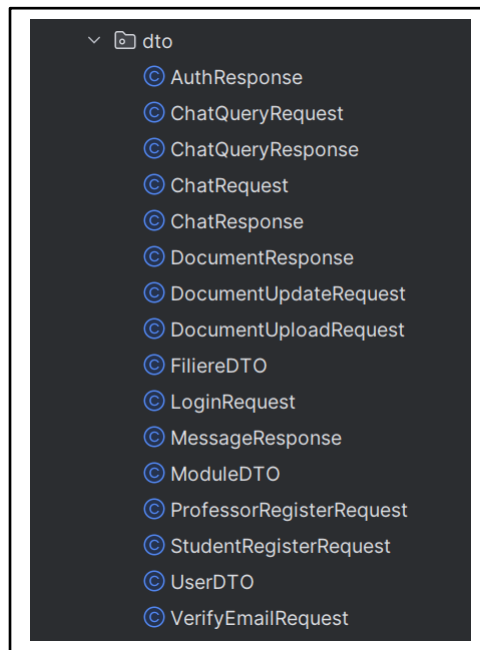
*Figure 29 : Schéma de l'architecture Respositor*

### ***dto (Data Transfer Objects)***

Les DTOs sont utilisés pour encapsuler les données transférées entre les différentes couches de l'application, et notamment entre le client (front-end) et le serveur (back-end) via les API REST. Ils permettent de structurer les requêtes et les réponses de manière claire et découplée des entités du modèle.

Exemples :

**LoginRequest.java, ChatQueryRequest.java, ChatQueryResponse.java, DocumentUploadRequest.java.**

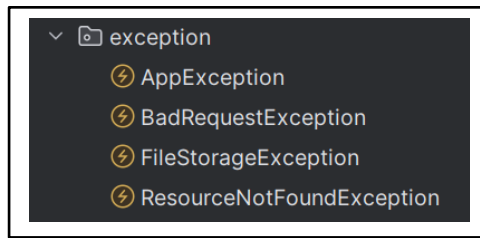


*Figure 30 : Schéma de l'architecture DTO*

### ***✓ exception (Gestion des Exceptions) :***

Ce package contient des classes d'exceptions personnalisées pour gérer les erreurs de manière structurée au sein de l'application. Une gestion globale des exceptions (via @ControllerAdvice) est souvent mise en place pour convertir ces exceptions en réponses HTTP appropriées.

Exemples : ***ResourceNotFoundException.java***, ***BadRequestException.java***



*Figure 31 : Schéma de l'architecture Exception*

### ✓ ***security (Couche de Sécurité)***

Ce package regroupe toutes les classes relatives à la sécurité de l'application, notamment l'implémentation de l'authentification et de l'autorisation basées sur JSON Web Tokens (JWT).

**JwtService.java** : Utilitaires pour la création et la validation des tokens JWT.

**CustomUserDetailsService.java** : Charge les informations spécifiques de l'utilisateur pour Spring Security.

**JwtAuthenticationFilter.java** : Filtre qui intercepte les requêtes HTTP pour valider les tokens JWT et configurer le contexte de sécurité Spring.

**UserDetailsImpl.java** : Implémentation de l'interface UserDetails de Spring Security.



*Figure 32 : Schéma de l'architecture Security*

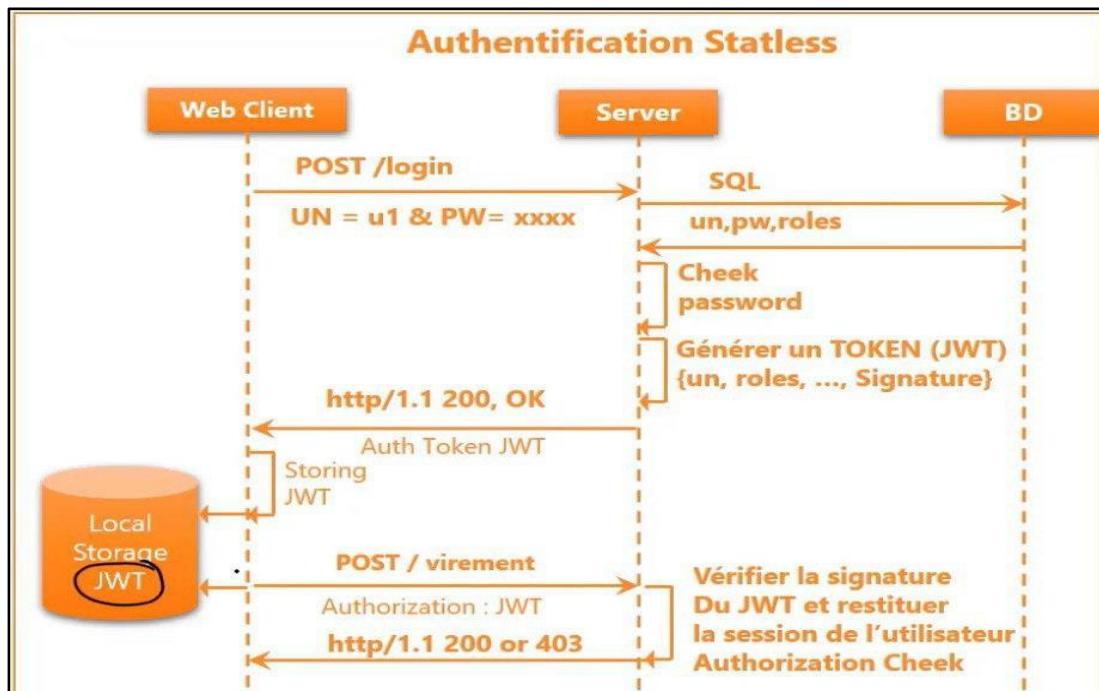


Figure 33 : Authentication Statless

Cette architecture structurée en couches nous a permis de développer un back-end solide et bien organisé pour notre Assistant Académique FSBM, capable de gérer la logique complexe du système RAG tout en assurant la sécurité et la performance nécessaires.

## ii. Développement du Front-end :

### a) Page d'accueil :

C'est la première page que l'utilisateur voit lors de son accès à notre Assistant Académique FSBM. Elle lui permet de se connecter à son compte existant en saisissant son "Email" et son "Mot de passe" dans les champs prévus à cet effet, puis en cliquant sur le bouton "Se connecter". Si l'utilisateur ne possède pas encore de compte, il a la possibilité d'en créer un nouveau en cliquant sur l'un des boutons d'inscription situés en bas : "Inscription Étudiant" ou "Inscription Professeur", selon son profil. Cette page sert donc de portail d'entrée principal à l'application.

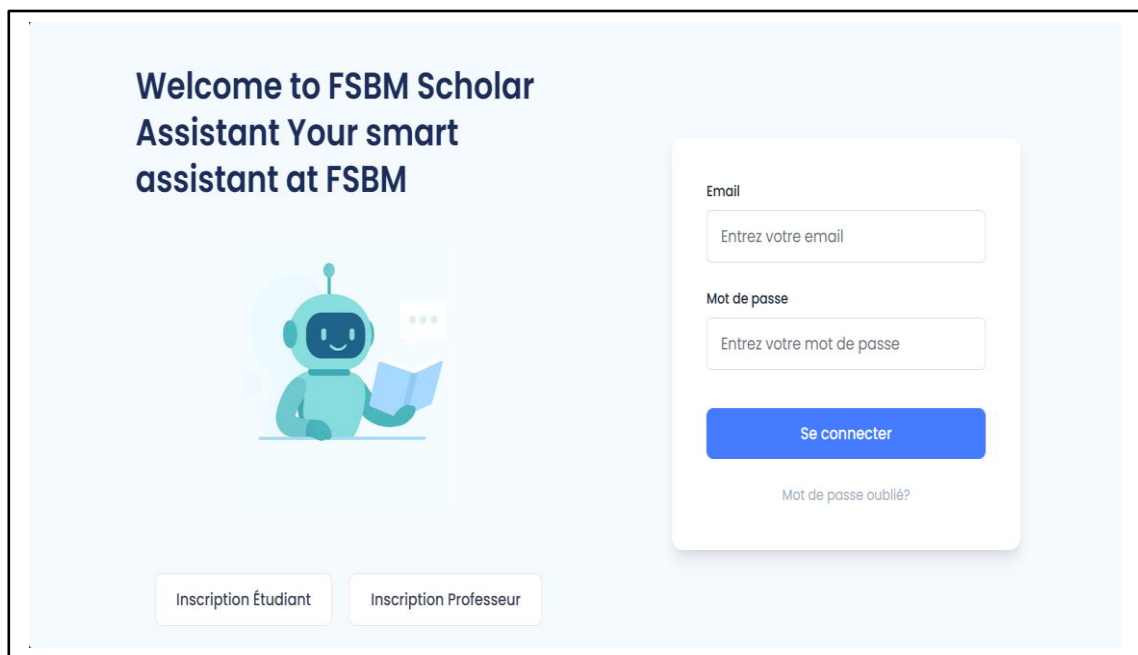


Figure 34 : Landing page

### b) Page d'inscription Étudiant :

Sur la page d'inscription Étudiant, l'utilisateur aspirant à devenir étudiant peut créer un nouveau compte en fournissant son Nom, son Prénom, son Adresse email personnelle, son Email universitaire, sa Filière (sélectionnée depuis une liste déroulante), ainsi qu'un Mot de passe et la confirmation de ce dernier. Il doit respecter les formats spécifiques pour l'adresse email (validité du format) et le mot de passe (par exemple, exigences de longueur ou de complexité). Si l'adresse e-mail (personnelle ou universitaire) existe déjà dans la base de données, ou si le mot de passe ne répond pas aux exigences de format, un message d'erreur approprié apparaîtra. La sélection de la filière est cruciale pour associer l'étudiant aux documents de cours pertinents

Retour

### Inscription Étudiant

Nom

Prénom

Adresse email

Email universitaire

Filière ▼

Mot de passe

Confirmer le mot de passe

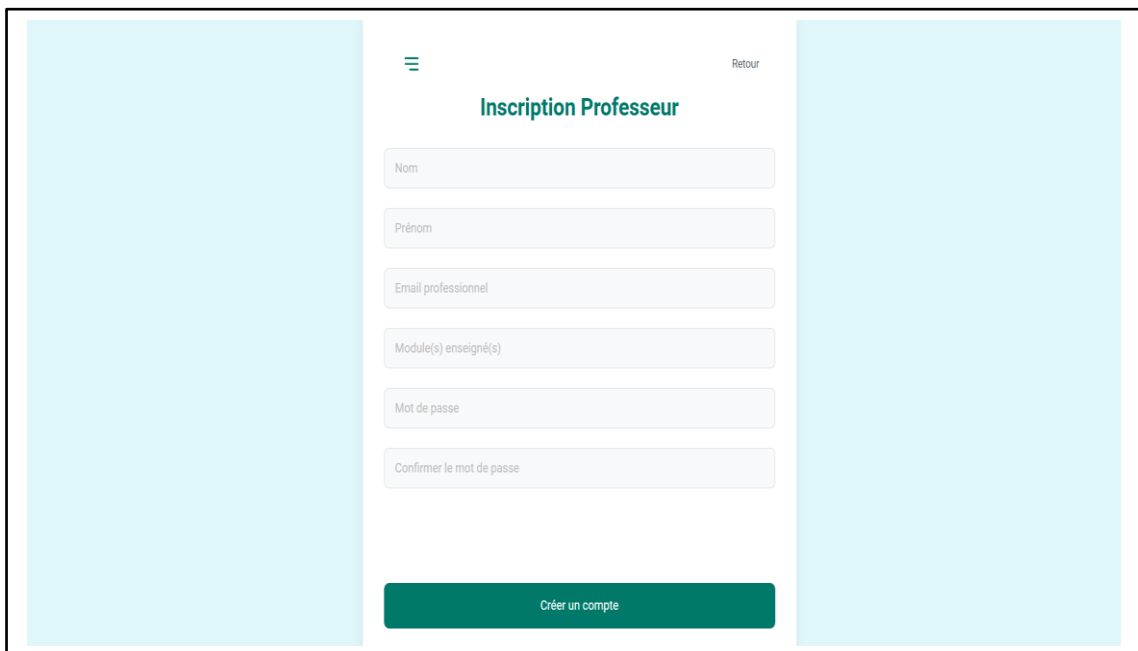
Créer un compte

Figure 35 : Page d'inscription Étudiant



### *c) Page d'inscription Professeur :*

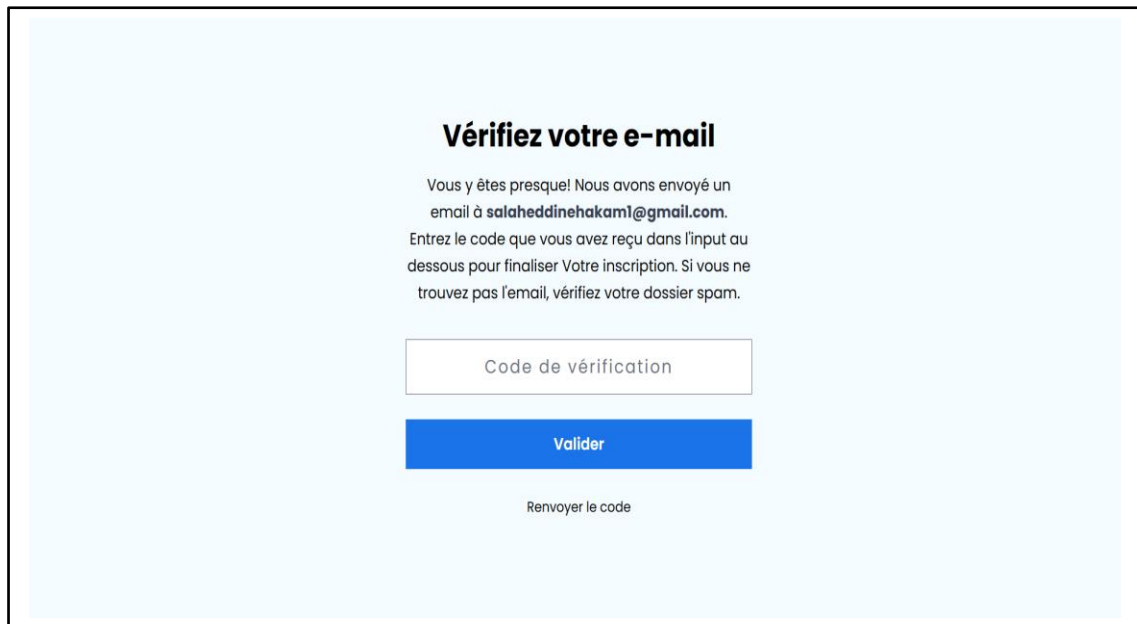
Sur la page d'inscription Professeur, l'utilisateur souhaitant s'enregistrer en tant que professeur peut créer un nouveau compte en fournissant son Nom, son Prénom, son Email professionnel, les Module(s) enseigné(s) (potentiellement un champ texte pour lister les modules ou une sélection multiple si implémentée), ainsi qu'un Mot de passe et la confirmation de celui-ci. Il doit veiller au respect des formats pour l'email professionnel et le mot de passe. Si l'email professionnel est déjà enregistré dans le système ou si le mot de passe ne respecte pas les critères de sécurité définis, un message d'erreur sera affiché. L'indication des modules enseignés est essentielle pour permettre au professeur de gérer les documents pédagogiques associés à ses cours.



*Figure 36 : Page d'inscription Professeur*

### *d) Page de vérification d'e-mail :*

Après que l'utilisateur (étudiant ou professeur) a soumis avec succès son formulaire d'inscription, il est redirigé vers la page de vérification d'e-mail. Sur cette page, il est informé qu'un e-mail contenant un Code de vérification (souvent un OTP - One Time Password) a été envoyé à l'adresse e-mail qu'il a fournie (par exemple, à salaheddinehakam1@gmail.com comme indiqué sur la capture). L'utilisateur doit alors consulter sa messagerie, récupérer ce code, et le saisir dans le champ "Code de vérification" prévu à cet effet sur la page. En cliquant sur le bouton "Valider", le système vérifie si le code entré correspond à celui envoyé. Si le code est correct, l'inscription est finalisée et le compte de l'utilisateur est activé. Si l'utilisateur ne reçoit pas l'e-mail, une option "Renvoyer le code" est généralement disponible pour initier un nouvel envoi. Il est également conseillé à l'utilisateur de vérifier son dossier spam. Un code incorrect ou expiré entraînera un message d'erreur.



*Figure 37 : Page de vérification d'e-mail*

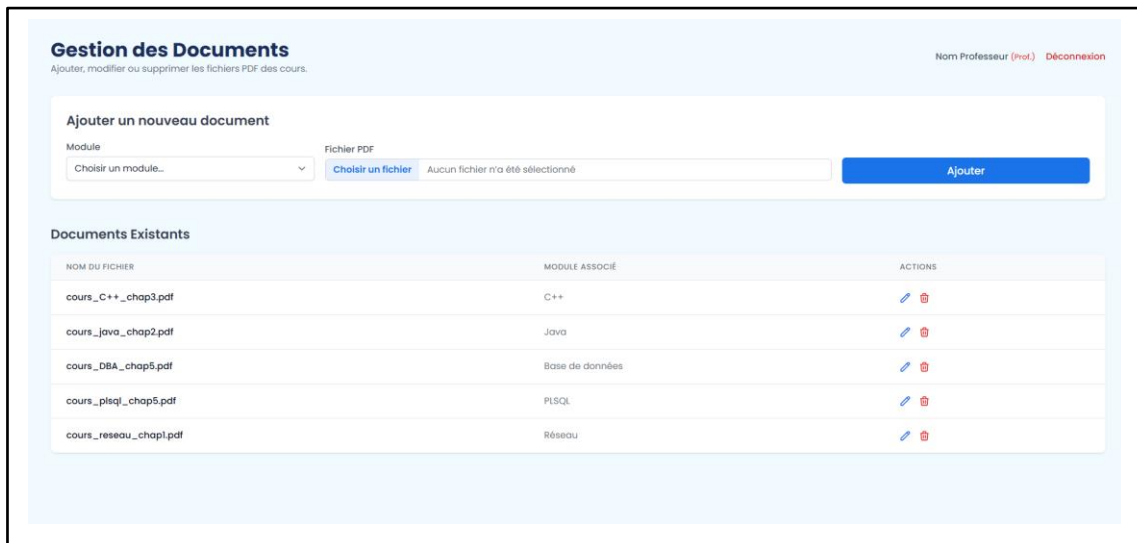
### ***e) Page de Gestion des Documents :***

La Page de Gestion des Documents est l'interface où les professeurs administrent les supports de cours (PDF, etc.) essentiels à la base de connaissances de l'assistant FSBM et à la pertinence de ses réponses RAG.

Ils peuvent Ajouter un document en sélectionnant un module, un fichier, puis en cliquant sur "Ajouter" pour que le système l'indexe.

La section Documents Existants liste les fichiers (nom, module) avec des options pour les Modifier ou Supprimer (ce qui les désindexe du RAG).

L'interface affiche aussi le nom du professeur et un lien de déconnexion, offrant un contrôle centralisé sur les contenus pédagogiques.



*Figure 38 : Page de Gestion des Documents pour le professeur*

#### ***f) Page Principale de Chat :***

Après authentification, l'utilisateur accède à l'interface principale de l'Assistant Académique FSBM, conçue pour une interaction directe et intuitive.

L'interface se structure comme suit :

**Barre Latérale (Gauche) :** Permet de gérer les conversations avec une option "Nouvelle conversation" (icône "+") et un accès à l'historique. Une icône de profil en bas mène aux options du compte et à la déconnexion.

**Zone de Conversation (Centrale) :** Affiche un message d'accueil (ex: "Hi, I'm Your smart Assistant. Comment puis-je vous aider ?") au démarrage. Cet espace se remplit ensuite avec le fil des questions de l'utilisateur et des réponses sourcées de l'assistant.

**Champ de Saisie (En bas) :** Un champ de texte ("Posez une question à Chat FSBM...") avec un bouton d'envoi permet à l'utilisateur de soumettre ses requêtes.

Cette disposition vise à centrer l'expérience sur l'échange conversationnel, en offrant des outils de navigation et de gestion de compte de manière discrète.

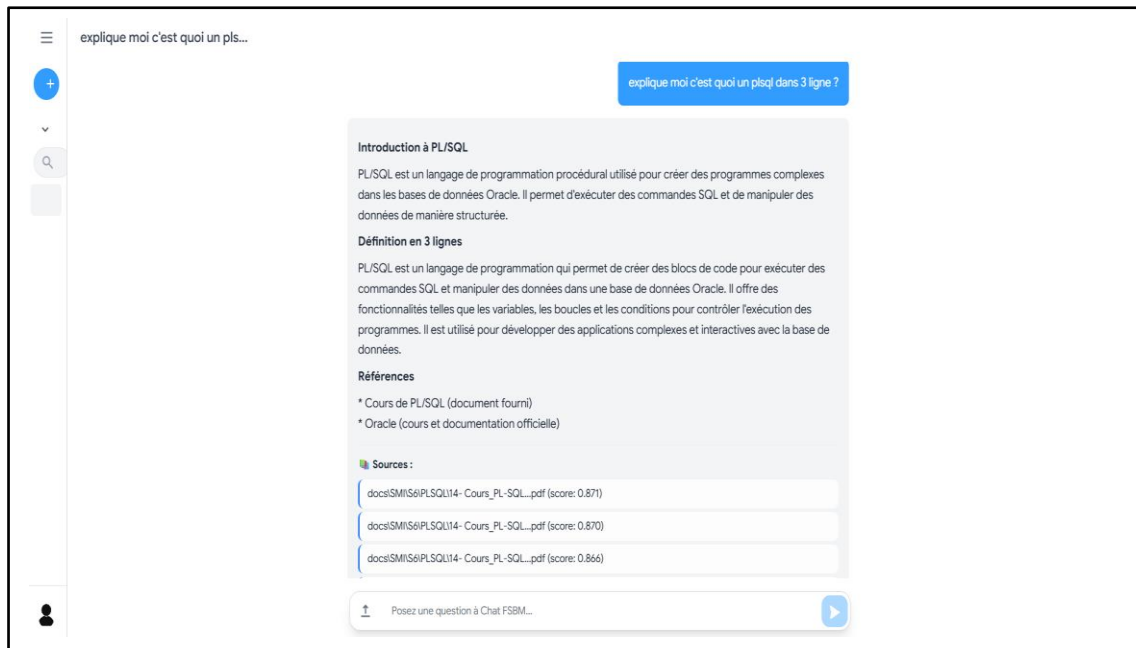


Figure 39 : Page Principale de Chat de l'Assistant

### g) Volet Latéral : Navigation des Conversations et Options du Compte :

Cette barre latérale est un composant crucial, car elle permet à l'utilisateur de gérer ses interactions et d'accéder aux fonctionnalités clés de l'assistant. L'utilisateur peut démarrer un nouvel échange en cliquant sur « Nouvelle Conversation » en haut de cette section. Il a également accès à l'« Historique » de ses conversations précédentes, avec une option pour les rechercher. En bas de cette barre latérale, il trouve les « Paramètres du compte » et l'option de « Déconnexion », ainsi qu'une section « Utilisateur » avec une icône de profil et un menu d'options supplémentaires.

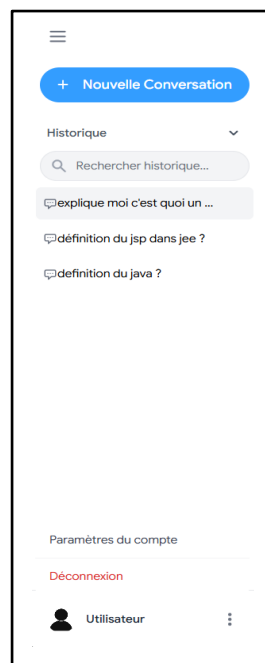


Figure 40 : Barre Latérale de l'Application de Chat

## 6. Conclusion :

Ce chapitre a présenté la mise en œuvre concrète de l'Assistant Académique FSBM. Nous avons détaillé les technologies et outils spécifiques utilisés, tant pour le développement que pour la gestion du projet. Un accent particulier a été mis sur la préparation du dataset, incluant le regroupement des données, leur structuration, leur nettoyage et leur vectorisation via FAISS, ainsi que sur les scripts Python développés pour la gestion de cette mémoire vectorielle. L'évaluation qualitative des réponses du modèle réglé a démontré la pertinence de notre approche RAG. Par la suite, le développement de l'interface utilisateur a été exposé, décrivant l'architecture en couches du back-end Spring Boot, incluant la gestion de la configuration, les contrôleurs REST, les services métier, les entités et les repositories, ainsi que la sécurisation par authentification stateless. Le développement du front-end avec Streamlit a également été présenté, détaillant les différentes pages conçues pour offrir une expérience utilisateur intuitive et fonctionnelle. Cette phase de réalisation a permis de traduire la conception en un prototype fonctionnel, prêt à être évalué plus globalement.

# Conclusion Générale

Ce projet de fin d'études s'est attaqué au défi majeur rencontré par les étudiants de la Faculté des Sciences Ben M'Sick (FSBM) : la difficulté d'accéder rapidement et précisément à l'information pertinente au sein d'une masse croissante de supports de cours numériques. Pour y remédier, nous avons conçu, développé et évalué un assistant académique intelligent, baptisé "FSBM Scholar Assistant" (ou le nom que vous avez choisi), capable de fournir des réponses contextualisées et sourcées aux questions des étudiants, en se basant exclusivement sur les documents pédagogiques fournis.

La solution repose sur l'architecture innovante de Génération Augmentée par Récupération (RAG), combinant la puissance du modèle de langage Llama-4-Scout avec une base de données vectorielle FAISS pour l'indexation et la recherche sémantique des contenus. Une interface web conviviale, développée avec Streamlit, a été mise en place pour permettre une interaction naturelle. Des fonctionnalités distinctes ont été implémentées pour les différents rôles utilisateurs : les étudiants peuvent interroger l'assistant et consulter leur historique, tandis que les professeurs disposent d'un espace dédié pour gérer les documents de cours qui alimentent la base de connaissances, et les administrateurs peuvent gérer les comptes utilisateurs.

La démarche adoptée, s'appuyant sur la méthodologie agile SCRUM et une modélisation UML détaillée, a permis de structurer le développement depuis la collecte et la préparation rigoureuse des données (documents de cours PDF, etc.) jusqu'à l'implémentation d'un backend robuste avec Spring Boot et d'un frontend interactif. Les scripts Python développés pour la création et la mise à jour de la mémoire vectorielle ont constitué un élément clé de la pipeline RAG. L'évaluation du prototype, bien que qualitative, a montré des résultats prometteurs en termes de précision, de pertinence des réponses et de justification par citation des sources.

Ce projet a non seulement permis de démontrer la faisabilité et l'utilité d'un tel assistant dans le contexte académique de la FSBM, mais a également offert une exploration approfondie des technologies d'intelligence artificielle de pointe. Il a souligné l'importance cruciale de la qualité des documents sources et de la pertinence de l'indexation pour l'efficacité d'un système RAG.

Bien que le système actuel soit fonctionnel et réponde aux objectifs initiaux, plusieurs perspectives d'amélioration s'ouvrent pour l'avenir. Celles-ci incluent l'extension à des modules plus spécialisés, l'intégration d'une gestion multilingue, le perfectionnement des capacités de dialogue, et l'ajout d'un système de feedback des utilisateurs pour une amélioration continue.

En conclusion, ce travail représente une contribution significative à l'amélioration des outils pédagogiques numériques au sein de la FSBM. Il offre une base solide pour de futurs développements et illustre le potentiel de l'intelligence artificielle pour transformer l'accès à la connaissance et soutenir l'apprentissage autonome des étudiants.

# Perspectives

## Recommandations

Le prototype développé dans le cadre de ce projet a démontré la faisabilité d'un chatbot pédagogique basé sur la technologie RAG. Toutefois, plusieurs pistes d'amélioration peuvent être envisagées pour renforcer son efficacité et sa portée :

- **Support multilingue** : Actuellement limité au français, le système pourrait être étendu pour comprendre et répondre en arabe et en anglais, répondant ainsi aux besoins des étudiants internationaux.
- **Intégration d'un système de feedback** : Permettre aux utilisateurs d'évaluer la pertinence des réponses aiderait à améliorer la qualité via un apprentissage continu.
- **Déploiement cloud** : Héberger l'application sur des plateformes cloud (AWS, Azure, etc.) garantirait une meilleure accessibilité, fiabilité et scalabilité.
- **Analyse de l'usage** : Intégrer des outils d'analyse (Google Analytics, Matomo) permettrait de mieux comprendre les besoins des étudiants et adapter le système en conséquence.
- **Extension aux formations spécialisées** : Ajouter des modules de Master ou de filières spécifiques permettrait de couvrir un public plus large.

Ces perspectives ouvrent la voie à un assistant académique plus complet, personnalisé et intelligent, capable d'accompagner les étudiants tout au long de leur parcours universitaire.

# Référence

Meta AI, « LLAMA-4: Open Foundation Models » <https://ai.meta.com/llama>.

LangChain Technologies, « Documentation LangChain » <https://docs.langchain.com>.

Facebook AI Research, « FAISS – Facebook AI Similarity Search »  
<https://github.com/facebookresearch/faiss>.

OpenRouter Inc., « Documentation de l'API OpenRouter »  
<https://openrouter.ai/docs>.

Streamlit Inc., « Documentation officielle Streamlit » <https://docs.streamlit.io>.

Tailwind Labs, « Documentation Tailwind CSS » <https://tailwindcss.com/docs>.

Spring Project, « Documentation Spring Boot » <https://spring.io/projects/spring-boot>.

Python Software Foundation, « Documentation Python 3 »  
<https://docs.python.org/3/>.

Hugging Face, « Carte du modèle intfloat/multilingual-e5-small »  
<https://huggingface.co/intfloat/multilingual-e5-small>.

Unstructured IO, « Unstructured – File Loaders » <https://github.com/Unstructured-IO/unstructured>.

Oracle, « Documentation Java SE 8 » <https://docs.oracle.com/javase/8/docs/>.

Équipe TypeScript, « Documentation du langage TypeScript »  
<https://www.typescriptlang.org>.

StarUML, « Site officiel StarUML » <https://staruml.io>.

Postman, « Plateforme API Postman » <https://www.postman.com>.

Oracle, « Documentation développeur MySQL » <https://dev.mysql.com/doc>.

OpenAI, « ChatGPT – Modèle de langage par OpenAI » <https://chat.openai.com>.