République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique





Université Saad Dahleb – Blida 1

Faculté des Sciences et de la Technologie

Département d'Informatique

MÉMOIRE DE MASTER

Option : Ingénierie des Logiciels

Option : Science des Données et TAL

RAGbot NAFTAL

Un Assistant Conversationnel Intelligent pour l'Analyse et l'Exploitation de la Documentation Technique Industrielle.

Réalisé par :	**Encadrement :**	**Promoteur :**
M. FAREH Youcef	M. LALLEM Seghir	Dr. CHABA MOUNA Mustapha
M. SAHRAOUI Khaled		

Année Universitaire 2025–2026

Remerciements

Tout d'abord, El Hamdoulillah. Nous tenons à remercier Monsieur Chaba Mouna Mustapha, notre promoteur, pour ses remarques précieuses et conseils constructifs qui nous ont guidés pour compléter ce projet.
Nous tenons également à exprimer notre profonde gratitude à la compagnie Naftal, particulièrement à la Direction de Blida, ainsi qu'à notre encadrant de stage, Monsieur Seghir Lallem, pour leur accueil chaleureux, leur soutien technique et les précieuses connaissances acquises durant cette période.
Nos remerciements s'adressent aussi à l'ensemble des enseignants du département d'Informatique de l'Université Saad Dahleb de Blida (USDB) qui nous ont accompagnés durant notre parcours académique. Nous exprimons notre profonde gratitude à nos parents pour leurs sacrifices, leurs encouragements permanents et leur soutien indéfectible.
Que nos familles, frères et sœurs trouvent ici l'expression de notre affection pour leur présence et leur soutien moral de chaque instant.
Enfin, nous tenons à exprimer notre profonde gratitude envers toutes les personnes qui ont participé, directement ou indirectement, à l'élaboration de ce travail.

Dédicaces Sahraoui khaled

Je dédie ce mémoire à mes parents, dont le soutien et les sacrifices ont été ma force tout au long de ce parcours.
À mon épouse, pour son soutien, sa patience et ses encouragements.
À ma famille et à mes enseignants, pour leur accompagnement et leur bienveillance.
Et à toutes les personnes qui, par leurs encouragements, ont illuminé mon chemin.

Dédicaces Fareh Youcef

À mes parents, Pour les sacrifices silencieux, l'amour inconditionnel et la confiance absolue que vous avez placés en moi. Ce travail est le fruit de votre soutien et le reflet de vos valeurs. Que ce mémoire soit le témoignage de ma profonde gratitude.
À mon binôme, En souvenir des défis relevés ensemble, des nuits blanches et de cette complicité qui a transformé ce projet de fin d'études en une aventure humaine inoubliable.
À mes amis, Pour les rires partagés, le soutien moral indéfectible et les moments de déconnexion si précieux durant cette année intense. Merci d'avoir toujours été là.
À mes frères et soeurs, Pour votre présence constante, vos encouragements et vos sourires qui ont su apaiser les moments de doute. Que ce travail soit le symbole de notre complicité et de notre attachement indéfectible.

Résumé

Ce mémoire porte sur la conception et l'implémentation de « RAGbot Naftal », un assistant conversationnel multilingue fondé sur l'architecture de Génération Augmentée par Récupération (RAG). Développé pour assister les agents de NAFTAL dans la consultation de documents techniques volumineux et complexes, ce système transforme des informations non structurées en une base de connaissances interactive et exploitable.

L'architecture proposée s'appuie sur un pipeline multimodal, capable d'analyser et de traiter simultanément le texte et les éléments visuels, tels que les schémas techniques, afin de préserver l'intégralité du contexte documentaire. Pour garantir une haute précision et réduire les risques d'hallucination, nous avons mis en œuvre une stratégie de recherche hybride. Celle-ci combine une approche de récupération sémantique et une approche lexicale, suivie d'une étape de réordonnancement profond des résultats pour ne conserver que les extraits les plus pertinents.

L'orchestration du système intègre des modèles performants pour les tâches d'inférence complexes. Cette solution offre la rigueur technique indispensable aux procédures industrielles.

Mots-clés : RAG, Assistant Conversationnel Multilingual, NAFTAL, Traitement Automatique du Langage Naturel, Recherche Hybride, Multimodalité.

Abstract

This thesis focuses on the design and implementation of ”RAGbot Naftal,” a multilingual conversational assistant based on the Retrieval-Augmented Generation (RAG) architecture. Developed to assist NAFTAL agents in consulting large and complex technical documents, this system transforms unstructured information into an interactive and actionable knowledge base.

The proposed architecture relies on a multimodal pipeline capable of simultaneously analyzing and processing both text and visual elements, such as technical diagrams, to preserve the complete documentary context. To ensure high accuracy and minimize hallucination risks, we implemented a hybrid search strategy. This approach combines semantic and lexical retrieval methods, followed by a deep re-ranking step to retain only the most relevant excerpts.

The system’s orchestration features powerful models for complex inference tasks. This solution provides the technical rigor required for industrial procedures.

Keywords: RAG, Multilingual Chatbot, NAFTAL, Natural Language Processing, Hybrid Search, Multimodality.

الملخص

تتناول هذه المذكرة تصميم وتطوير "RAGbot Naftal"، وهو مساعد محادثة متعدد اللغات يعتمد على بنية التوليد المعزز بالاسترجاع .(RAG) تم تطوير هذا النظام لمساعدة موظفي شركة نفطال في الاطلاع على الوثائق التقنية الضخمة والمعقدة، حيث يقوم بتحويل المعلومات غير المهيكلة إلى قاعدة معرفية تفاعلية وقابلة للاستخدام.

تعتمد البنية المقترحة على مسار معالجة متعدد الوسائط ، قادر على تحليل ومعالجة النص والعناصر المرئية (مثل المخططات التقنية) في وقت واحد، وذلك للحفاظ على السياق الوثائقي الكامل. ولضمان درجات عالية من الدقة وتقليل مخاطر الهلوسة، قمنا بتنفيذ استراتيجية بحث هجينة تجمع بين الاسترجاع الدلالي والاسترجاع المعجمي، تليها خطوة إعادة ترتيب متعمقة للنتائج للاحتفاظ فقط بالمقتطفات الأكثر صلة.

يتضمن تنسيق النظام نماذج قوية لمهام الاستدلال المعقدة. يوفر هذا الحل الدقة التقنية الصارمة المطلوبة للإجراءات الصناعية .

الكلمات المفتاحية: RAG، مساعد محادثة متعدد اللغات، نفطال، معالجة اللغات الطبيعية، البحث الهجين، تعدد الوسائط،.

Table des matières

Remerciements **iii**

Dédicaces **iv**

Résumé **vi**

Abstract **vii**

viii **الملخص**

Liste des abréviations et acronymes **xv**

Introduction Générale **1**

1 État de l'Art **3**

1.1 Chatbots historiques et modernes 3

1.1.1 Définition générale d'un chatbot 3

1.1.2 Les premiers chatbots (ELIZA et PARRY) 3

1.1.3 Évolution des chatbots (A.L.I.C.E., Jabberwacky) 4

1.1.4 Chatbots modernes avec l'IA (Watson, Siri, Alexa, ChatGPT) . . . 4

1.1.5 Transition vers les LLMs 4

1.1.6 Techniques utilisées dans les chatbots 5

1.1.7 Types de chatbots 5

1.2 Retrieval-Augmented Generation : architecture et analyse comparative . . . 5

1.2.1 Architecture Fondamentale : Le Cycle Retrieval-Generation 6

1.2.2 Avantages du RAG par rapport aux approches classiques 6

1.2.3 Analyse Comparative des Approches 7

1.3 Extraction et structuration des données 7

1.3.1 Méthodes et outils d'extraction documentaire 8

1.3.2 Prétraitement et normalisation 8

1.3.3 Modélisation et représentation vectorielle 8

1.3.4 Objectifs et défis de la structuration contextuelle 9

1.4 Limites des approches existantes 9

1.4.1 Limites des systèmes documentaires classiques 9

1.4.2 Limites des modèles NLP généralistes 10
1.4.3 Problèmes de gestion des documents techniques industriels 10
1.4.4 Manque d'intégration des pipelines (extraction → recherche → génération) 10
1.5 Étude du système documentaire NAFTAL 10
1.5.1 Cartographie du processus documentaire et des flux d'information actuels 11
1.5.2 Évaluation critique et limites opérationnelles du modèle existant . . 12
1.6 Conclusion 13

2 Méthodologie **15**
2.1 Vue d'ensemble de l'architecture du système RAG 15
2.1.1 Description générale du pipeline RAG 16
2.1.2 Choix d'une architecture hybride Dense + Sparse 17
2.1.3 Intégration multimodale (texte + images) 18
2.2 Modélisation vectorielle et stratégies d'indexation 19
2.2.1 Implémentation et paramétrage de la base vectorielle Qdrant 19
2.2.2 Structuration des métadonnées 20
2.2.3 Stratégie de découpage des documents (chunking) 21
2.2.4 Indexation incrémentale et gestion des doublons 22
2.3 Extraction multimodale des documents 22
2.3.1 Extraction de texte depuis PDF et documents 23
2.3.2 Stratégie OCR Hybride : Gemini Vision et Tesseract 24
2.3.3 Indexation des images avec CLIP 25
2.4 Recherche hybride et fusion 25
2.4.1 Recherche sparse avec BM25 25
2.4.2 Fusion des résultats avec RRF 26
2.4.3 Sélection des candidats pour reranking 27
2.5 Reranking avec Cross-Encoder 27
2.5.1 Exploitation de l'architecture bge-reranker-v2-m3 pour l'optimisation des systèmes de RAG 28
2.5.2 Principe du reranking par paires 29
2.5.3 Sélection finale des documents 29
2.6 Gestion du contexte et des réponses RAG 30
2.6.1 Construction du contexte enrichi 30
2.6.2 Intégration des images dans le contexte 32
2.6.3 Génération de réponses basées sur contexte uniquement 32
2.7 Conclusion 33

3 Conception Architecturale **35**

3.1 Introduction 35
3.2 Démarche de Conception et Modélisation UML 35
3.3 Architecture Logicielle Globale et Topologie des Services 36
3.3.1 Découpage en couches applicatives 36
3.3.2 Spécification des interfaces de programmation (API REST) 37
3.4 Modélisation Fonctionnelle et Structurale (UML) 38
3.4.1 Identification des Acteurs 38
3.4.2 Diagramme global des cas d'utilisation 39
3.5 Modélisation de la Dynamique des Flux d'Exécution 40
3.5.1 Orchestration de l'authentification sécurisée (RBAC & JWT) 40
3.5.2 Ingestion, segmentation et vectorisation asynchrone 41
3.6 Conception et Persistance des Données 42
3.6.1 Modélisation conceptuelle et relationnelle (SQLite) 42
3.6.2 Modélisation logique du stockage vectoriel (Qdrant) 42
3.7 Conclusion 43

4 Implémentation, Expérimentations et Évaluation 45
4.1 Environnement et cadre de développement 45
4.1.1 Choix technologiques et configuration de l'infrastructure 45
4.2 Déploiement logiciel et Interfaces Homme-Machine (IHM) 49
4.2.1 Conception de l'interface conversationnelle adaptative 49
4.2.2 Tableau de bord d'administration et de supervision 50
4.3 Implémentation des algorithmes critiques du système 51
4.3.1 Recherche Hybride et Fusion de Rangs (Reciprocal Rank Fusion - RRF) 51
4.3.2 Re-classement Sémantique et Filtrage de Confiance (Trust Filtering) 52
4.3.3 Pipeline d'Extraction Multimodale et Smart OCR 53
4.4 Cadre expérimental et protocole d'évaluation 55
4.4.1 Préparation du corpus et jeu de vérité 55
4.4.2 Protocole d'évaluation automatisé et métriques 55
4.4.3 Périmètre d'évaluation et rigueur méthodologique 56
4.5 Analyse quantitative et qualitative des résultats 56
4.5.1 Évaluation des performances du moteur de recherche 56
4.6 Discussion des Résultats 57
4.6.1 Analyse des performances élevées de récupération (Recall@10) . . 57
4.6.2 Discussion du score de Fidélité (Faithfulness) 58
4.6.3 Justification scientifique de l'architecture hybride Dense + Sparse . 59
4.6.4 Apports observés de Qdrant : Justification du choix technologique . 60
4.6.5 Discussion sur la sécurité, le RBAC et la souveraineté des données . 61
4.6.6 Analyse du MRR : Signification opérationnelle 61

4.6.7 Positionnement de la solution proposée par rapport aux paradigmes existants 62
4.6.8 Synthèse : Carte des forces et limites du système 63
4.7 Conclusion 64

Conclusion Générale et Perspectives **65**

Références Bibliographiques **67**

Table des figures

2.1 Diagramme de Flux RAG (Génération Augmentée par Recherche) 16
2.2 Extrait JSON des métadonnées d'un document A.M.I 31

3.1 Architecture Globale du Système NAFTAL RAGbot. 36
3.2 Diagramme global des cas d'utilisation du système *Naftal Ragbot*. 39
3.3 Diagramme de séquence : Orchestration de l'authentification et contrôle d'accès RBAC. 40
3.4 Diagramme de séquence : Pipeline asynchrone d'ingestion et de vectorisation. 41
3.5 Modélisation conceptuelle de l'architecture de données relationnelle du système RAG 42

4.1 Interface de discussion intelligente (Agentic Chat) du système NAFTAL RAGbot 49
4.2 Interface du Panneau d'Administration Système (Admin Panel) 50
4.3 Interface de gestion de la Base de Données des Documents métadonnées . 51
4.4 Architecture du pipeline de Reranking 52

Liste des tableaux

1.1 Analyse Comparative des Approches 7
1.2 Synthèse des limites du modèle existant et opportunités de l'architecture cognitive 12

2.1 Comparaison des paradigmes de Recherche d'Information (RI) : des méthodes lexicales classiques aux approches hybrides contemporaines. 17
2.2 Analyse comparative des bases de données vectorielles pour l'architecture RAG. 19

3.1 Matrice de contrôle d'accès sémantique et d'habilitation RBAC. 38

4.1 Architecture : Noyau Applicatif (Backend) 46
4.2 Architecture : Couche de Stockage Hybride 46
4.3 Architecture : Interface Utilisateur (Frontend) 47
4.4 Architecture : Intelligence Artificielle et Inférence 47
4.5 Architecture : Infrastructure Physique et Déploiement 48
4.6 Performances comparatives des configurations de recherche 57
4.7 Synthèse des performances comparatives des configurations de recherche . 57
4.8 Adéquation Paradigme / Type de requête 59
4.9 Comparatif des bases vectorielles 60
4.10 Tableau comparatif détaillé des paradigmes de recherche et de génération . 62
4.11 Synthèse globale des forces, limites et perspectives 63

Liste des abréviations et acronymes

API	*Application Programming Interface* (Interface de Programmation d'Application)
AMI	Avis Mouvement Immobilisatations
ARI	Avis de Reception des Immobilisations
ASI	Avis de Sortie des Immobilisations
BGE	*BAAI General Embedding* (Modèle de plongement lexical développé par BAAI)
BM25	*Best Matching 25* (Algorithme de classement probabiliste en recherche d'information)
CLIP	*Contrastive Language-Image Pre-training* (Pré-entraînement contrastif langage-image)
ERDP	Entreprise de Répartition des Produits Pétroliers (Ancienne dénomination de NAFTAL)
HNSW	*Hierarchical Navigable Small World* (Algorithme d'indexation vectorielle approximative)
HSE	Hygiène, Sécurité et Environnement
IHM	Interface Homme-Machine
JSON	*JavaScript Object Notation* (Format de données textuelles)
JWT	*JSON Web Token* (Jeton Web JSON pour l'authentification)
LLM	*Large Language Model* (Modèle de Langage à Grande Échelle)
MMR	*Maximal Marginal Relevance* (Pertinence Marginale Maximale)
MRR	*Mean Reciprocal Rank* (Rang Réciproque Moyen)

NDCG *Normalized Discounted Cumulative Gain* (Gain Cumulatif Actualisé Normalisé)

NER *Named Entity Recognition* (Reconnaissance d'Entités Nommées)

NLP *Natural Language Processing* (Traitement Automatique du Langage Naturel - TAL)

OCR *Optical Character Recognition* (Reconnaissance Optique de Caractères)

PDF *Portable Document Format*

RAG *Retrieval-Augmented Generation* (Génération Augmentée par Récupération)

RBAC *Role-Based Access Control* (Contrôle d'Accès Basé sur les Rôles)

REST *Representational State Transfer* (Style d'architecture logicielle)

RRF *Reciprocal Rank Fusion* (Fusion de Rangs Réciproques)

SCADA *Supervisory Control and Data Acquisition* (Supervision et Acquisition de Données)

SPA Société Par Actions

SSE *Server-Sent Events* (Événements envoyés par le serveur)

TF-IDF *Term Frequency-Inverse Document Frequency* (Fréquence du Terme - Fréquence Inverse de Document)

UML *Unified Modeling Language* (Langage de Modélisation Unifié)

Introduction Générale

Le secteur des hydrocarbures constitue la pierre angulaire de l'économie algérienne, et dans ce paysage stratégique, NAFTAL, filiale à 100 % du groupe Sonatrach, occupe une place centrale. Créée initialement sous le nom d'ERDP par le décret N° 80/101 du 06 avril 1981, l'entreprise est aujourd'hui une Société Par Actions (SPA) au capital social de 160 milliards de DA. Sa mission est vitale : assurer la distribution et la commercialisation des produits pétroliers et dérivés sur l'ensemble du territoire national.

L'envergure opérationnelle de NAFTAL est illustrée par ses performances récentes. En 2023, l'entreprise a commercialisé un volume total de 16,1 millions de tonnes de produits, générant un chiffre d'affaires de 423,1 milliards de dinars. Pour soutenir cette dynamique, NAFTAL s'appuie sur un capital humain imposant de 32 107 agents. La préservation et la transmission du savoir technique sont au cœur de sa stratégie, avec plus de 9 554 agents formés en 2023 au sein de ses centres d'excellence à El-Khroub, Mostaganem et Bordj El Kiffan.

Dans leurs activités quotidiennes, les équipes techniques et administratives sont confrontées à une masse volumineuse de documents : procédures internes, fiches techniques, FAQ et rapports de maintenance. Ces informations, souvent présentées de manière non structurée en arabe et en français, rendent l'accès à l'information pertinente lent et sujet aux erreurs. Le traitement manuel de ces flux documentaires limite l'efficacité de l'entreprise et compromet l'actualisation des décisions opérationnelles.

Afin de pallier ces difficultés, le recours à l'Intelligence Artificielle (IA), et plus précisément au Traitement Automatique du Langage Naturel (NLP), offre des perspectives innovantes. Notre projet consiste à concevoir un chatbot intelligent multilingue reposant sur l'architecture RAG (Retrieval-Augmented Generation). Contrairement à une recherche textuelle classique, cette solution permet une interaction dynamique où l'utilisateur interroge le système en langage naturel pour recevoir des réponses précises et contextualisées, extraites directement des bases documentaires de NAFTAL via le moteur d'indexation Qdrant et le modèle de langage (Gemini 2.5 flash/Llama-3.3-70b versatile).

L'intégration de cet assistant virtuel, développé avec des outils modernes tels que LangChain, FastAPI et Streamlit, vise à transformer la gestion documentaire chez NAFTAL. En simplifiant l'accès aux informations critiques, ce système constitue une solution efficace de digitalisation. *Enfin*, ce projet aspire à améliorer la rapidité de traitement des requêtes, à réduire la charge de travail des équipes et à optimiser la gestion des connaissances internes, tout

en garantissant un usage fluide et efficace des deux langues de travail de l'entreprise.

Chapitre 1

État de l'Art

Ce chapitre présente un état de l'art des méthodes utilisées pour rechercher des informations pertinentes dans des documents techniques volumineux. Il couvre le traitement automatique des PDF, le NLP, la recherche d'information intelligente et l'IA, avec un accent sur les solutions adaptées à un chatbot pour la consultation des documents techniques de NAFTAL.

1.1 Chatbots historiques et modernes

1.1.1 Définition générale d'un chatbot

Un chatbot peut être défini comme un programme logiciel capable d'interagir avec les utilisateurs en utilisant le langage naturel. L'objectif principal de ces systèmes est de simuler une conversation humaine et de fournir des informations ou d'exécuter des tâches spécifiques de manière automatisée [1]. Les chatbots sont des entités conversationnelles qui utilisent des algorithmes d'Intelligence Artificielle pour traiter les entrées de l'utilisateur (texte ou voix) et générer des réponses cohérentes. L'évolution de ces outils a permis de passer de simples systèmes de correspondance de motifs (pattern matching) à des agents sophistiqués capables de comprendre le contexte et l'intention [2].

1.1.2 Les premiers chatbots (ELIZA et PARRY)

L'histoire des agents conversationnels débute par des systèmes basés sur des règles rigides :

ELIZA (1966) : Développé par Joseph Weizenbaum (MIT), ce programme simulait un psychothérapeute. Il utilisait le *Pattern Matching* pour reformuler les phrases de l'utilisateur sans réelle compréhension sémantique [3].

PARRY (1972) : Créé par Kenneth Colby, il simulait un patient schizophrène. Plus avancé qu'ELIZA, il intégrait des variables affectives et des états émotionnels (peur, méfiance)

pour orienter la conversation [4].

1.1.3 Évolution des chatbots (A.L.I.C.E., Jabberwacky)

Cette étape introduit la structuration des données et l'apprentissage par interaction :

A.L.I.C.E (1995) : Utilise le langage AIML pour organiser des milliers de réponses prédéfinies, offrant une meilleure cohérence conversationnelle [5].

Jabberwacky (1997) : Premier système à simuler la voix humaine en apprenant directement des interactions avec les utilisateurs, visant à imiter le langage naturel plutôt que de simples règles [6].

1.1.4 Chatbots modernes avec l'IA (Watson, Siri, Alexa, ChatGPT)

Cette phase marque le passage du simple script à l'intelligence cognitive et générative :

IBM Watson (2011) : Un système pionnier capable de comprendre le langage naturel et d'analyser d'immenses bases de données pour répondre à des questions complexes [7].

Assistants Vocaux (Siri, Alexa, 2010+) : Intégration de la reconnaissance vocale et du traitement automatique du langage (NLP) dans les smartphones et objets connectés [8].

ChatGPT (2022) : Basé sur l'architecture Transformer, ce modèle de langage à grande échelle (LLM) génère des réponses fluides et contextuelles, révolutionnant l'interaction homme-machine [9].

1.1.5 Transition vers les LLMs

L'évolution des chatbots a atteint un tournant majeur avec le passage des systèmes statistiques aux modèles de langage à grande échelle (LLMs). Cette transition est basée sur deux piliers :

Architecture Transformer (2017) : Introduite par Google, elle permet au modèle de traiter l'intégralité d'un texte simultanément grâce au le mécanisme d'Attention, capturant des relations contextuelles complexes [10].

Changement de Paradigme : Contrairement aux anciens bots limités à des réponses pré-enregistrées, les LLMs (comme GPT ou Gemini) prédisent le mot suivant en fonction du contexte global, permettant une génération de texte fluide et une compréhension profonde de la sémantique [11].

1.1.6 Techniques utilisées dans les chatbots

Les chatbots fonctionnent selon différentes approches techniques, qui ont évolué par rapport à la complexité des besoins :

Pattern Matching (Correspondance de motifs) : La technique la plus simple utilisée par ELIZA et ALICE. Elle consiste à comparer l'entrée de l'utilisateur avec des modèles de phrases prédéfinis dans une base de connaissances [12].

Apprentissage Automatique (Machine Learning) : Utilise des modèles statistiques pour classifier les intentions de l'utilisateur (intents) et extraire des entités. Le système s'améliore à mesure qu'il est exposé à de nouvelles données [13].

Apprentissage Profond (Deep Learning) : Basée sur les réseaux de neurones, cette technique permet au chatbot de comprendre des structures de phrases complexes et de générer des réponses sans intervention humaine constante pour définir des règles [14].

1.1.7 Types de chatbots

Les agents conversationnels se distinguent par leur architecture et leur finalité opérationnelle. On peut les regrouper en quatre catégories majeures :

Chatbots basés sur des règles : Ils fonctionnent sur un modèle rigide de questions-réponses prédéfinies. Bien que simples à déployer pour des FAQ statiques, ils manquent de flexibilité face à des requêtes imprévues [15].

Chatbots intelligents (IA) : Fondés sur le NLP et le Machine Learning, ils analysent l'intention de l'utilisateur pour offrir une interaction fluide et contextuelle, capable de traiter des demandes complexes [16].

Chatbots transactionnels : Ils sont spécialisés dans l'exécution de tâches précises (réservations, commandes). Ils interagissent directement avec des systèmes tiers par des API pour automatiser des processus métiers [17].

Chatbots hybrides : Cette approche combine la fiabilité des règles fixes avec la puissance de l'IA. Elle permet de sécuriser les réponses sur des sujets critiques tout en offrant une liberté de dialogue à l'utilisateur [18].

1.2 Retrieval-Augmented Generation : architecture et analyse comparative

Le RAGest une architecture hybride en TAL qui combine la recherche d'information et la génération de texte. Au contraire des modèles classiques qui s'appuient sur leur mémoire

interne, le RAG utilise une mémoire non-paramétrique (base de documents externes) pour donner des réponses précises et sourcées [19]. Selon Jurafsky & Martin, cette approche permet au modèle de langue d'extraire des faits en temps réel depuis un corpus spécifique (ex : documents de NAFTAL) avant de générer une réponse, réduisant les risques d'hallucinations factuelles [12].

1.2.1 Architecture Fondamentale : Le Cycle Retrieval-Generation

L'architecture du RAG utilise un processus séquentiel divisé en deux composants interdépendants qui assurent la fiabilité de la réponse finale :

La Récupération (Retrieval) : À la réception d'une requête, le système effectue une recherche sémantique dans une base de données vectorielle. Il identifie et extrait les fragments de texte (Chunks) les plus pertinents par rapport à la question posée.

La Génération (Generation) : Les fragments récupérés sont intégrés au "Prompt" envoyé au modèle de langue (ex : Gemini 2.5 flash). Ce dernier synthétise alors une réponse naturelle en se basant sur les informations contenues dans ce contexte [20].

1.2.2 Avantages du RAG par rapport aux approches classiques

L'intégration de l'architecture RAG donne des avantages stratégiques pour le traitement de données techniques complexes, surpassant les limitations des modèles de langue standards et de la recherche documentaire classique. :

- **Réduction des Hallucinations :** Contrairement aux modèles classiques qui peuvent générer des informations erronées avec assurance, le RAG force le modèle à ancrer ses réponses dans des preuves factuelles extraites des documents de NAFTAL. Cela garantit une intégrité des données cruciale pour les procédures de sécurité.

- **Actualisation Dynamique des Connaissances :** Le Fine-tuning nécessite un réentraînement coûteux pour chaque mise à jour. En revanche, le RAG permet de mettre à jour le système instantanément en ajoutant simplement de nouveaux documents à la base vectorielle, sans modifier les paramètres du modèle [20].

- **Transparence et Citabilité :** Le RAG offre une "traçabilité" totale en permettant au système de citer précisément la source (page du manuel, référence du schéma) utilisée pour générer la réponse. Cette capacité est fondamentale pour la vérification humaine dans un cadre industriel.

- **Recherche Sémantique :** Contrairement aux recherche documentaire classique, qui se limitent à une correspondance stricte de mots-clés et renvoient des listes d'extraits

bruts sans analyse contextuelle, le RAG exploite des représentations vectorielles pour une compréhension sémantique profonde. [21].

1.2.3 Analyse Comparative des Approches

Afin de justifier scientifiquement le choix de notre architecture, nous comparons le RAG aux méthodes classiques et aux modèles de langue simples.

Approche	**Principe**	**Avantages**	**Limites**
Recherche par mots-clés	Correspondance lexicale	Simple, rapide	Pas de compréhension sémantique
Chatbots à règles	Réponses prédéfinies	Fiable pour cas simples	Rigide, peu flexible
LLM seuls	Génération de texte	Réponses fluides	Hallucinations, pas de sources
RAG	Recherche + Génération	Réponses précises, traçables	Plus complexe

Tab. 1.1 : Analyse Comparative des Approches

Le tableau montre une progression dans la capacité des systèmes à traiter l'information. Les approches par mots-clés et par règles sont limitées par leur incapacité à comprendre le contexte sémantique d'une question technique. Bien que les LLM seuls produisent des réponses naturelles, ils souffrent d'un manque de fiabilité (hallucinations) car ils n'ont pas accès à des sources externes en temps réel. D'un autre côté, le RAG utilise la recherche pour extraire l'information des manuels de NAFTALet la génération pour la reformuler. Cela garantit une réponse précise et traçable.

1.3 Extraction et structuration des données

L'extraction d'information (EI) et sa structuration constituent un processus séquentiel fondamental permettant de transformer les textes techniques non structurés, tels que les manuels de maintenance de NAFTAL, en une base de connaissances structurée et exploitable par des systèmes informatiques [22]. Ce mécanisme intégré repose, d'une part, sur la reconnaissance d'entités nommées (NER) pour identifier des termes techniques spécifiques (modèles de machines,etc.) [23] , d'autre part, sur une modélisation sémantique reliant chaque concept à son contexte d'origine. Dans un environnement industriel, l'automatisation de ce flux est cruciale pour limiter les erreurs humaines et optimiser l'accès aux informations critiques contenues dans des volumes documentaires massifs [24].

1.3.1 Méthodes et outils d'extraction documentaire

Avant le déploiement des modèles basés sur l'apprentissage profond, l'extraction reposait sur des approches conventionnelles dites « par règles » (expressions régulières) ou « par dictionnaires » (*gazetteers*). Bien que précises pour l'identification d'entités stables (ex. : formats de dates, nomenclatures de carburants), ces méthodes demeurent rigides, difficiles à maintenir face à la variabilité des documents et peinent à passer à l'échelle sans une intervention humaine continue [25, 26, 27].

Actuellement, l'extraction à partir de documents techniques au format PDF représente un défi algorithmique en raison de leur complexité structurelle (tableaux, mises en page multi-colonnes, annotations). Pour résoudre cette problématique, le recours à des solutions spécialisées s'avère nécessaire. PDFMiner analyse la topologie du document pour localiser les blocs de texte via leurs coordonnées spatiales [28], tandis que PDFPlumber est particulièrement adapté à l'extraction de structures tabulaires et d'interfaces d'E/S [29]. L'outil PyMuPDF (fitz) offre, quant à lui, un accès bas niveau à la structure logique du fichier. Bien que ces bibliothèques permettent une analyse granulaire du document, elles exigent un prétraitement rigoureux visant à filtrer le bruit documentaire avant l'étape de modélisation [30].

1.3.2 Prétraitement et normalisation

Le prétraitement est une étape indispensable pour nettoyer le texte brut avant son ingestion. Ce processus comprend la tokenisation, l'élimination des mots vides (*stop words*) et la lemmatisation pour ramener les termes à leur racine lexicale [31]. Dans le contexte industriel de Naftal, cette phase requiert spécifiquement la normalisation des unités de mesure et la résolution des acronymes. Cette standardisation garantit l'alignement sémantique nécessaire à la précision des inférences du modèle de langage [32].

1.3.3 Modélisation et représentation vectorielle

Une fois les informations extraites et normalisées, le contenu brut doit être modélisé. Cette étape repose sur une décomposition hiérarchique segmentant le document en unités sémantiques cohérentes (titres, paragraphes, tableaux) [33].

Les données adoptent ensuite une représentation exploitable par la machine. Si le format JSON est couramment utilisé pour structurer l'information en paires clé-valeur facilitant la récupération des attributs matériels [34], la représentation centrale pour l'architecture RAG est le plongement vectoriel (*dense embedding*) [21]. Cette technique projette chaque segment textuel dans un espace mathématique multidimensionnel où la distance entre deux vecteurs quantifie leur similarité sémantique. Le système peut ainsi évaluer la pertinence d'une requête utilisateur par rapport au corpus, assurant une récupération précise même en l'absence

de correspondance lexicale stricte [35]. Cette modélisation établit une correspondance mathématique directe entre la documentation statique et l'algorithme de recherche.

1.3.4 Objectifs et défis de la structuration contextuelle

La structuration des données extraites répond à plusieurs impératifs : résoudre les ambiguïtés techniques en typant les entités selon leur rôle métier, assurer l'interopérabilité entre les différents référentiels de NAFTAL, et optimiser la recherche par la modélisation des relations de dépendance entre procédures et équipements [36].

Toutefois, cette étape soulève une problématique propre à la donnée industrielle. Contrairement à un corpus généraliste, une métrique isolée — telle qu'un seuil de température ou une référence de pièce — est dépourvue de sens si elle est dissociée de son contexte d'application (ex. : l'équipement ciblé ou la phase de maintenance). L'enjeu technique consiste à préserver l'héritage hiérarchique de chaque segment lors de sa vectorisation. Pour répondre à cette contrainte, des métadonnées contextuelles (Payload) sont systématiquement attachées à chaque vecteur. L'architecture ne se limite plus à l'extraction de l'information brute ; elle intègre la dimension opérationnelle associée, garantissant ainsi la génération de réponses techniquement valides et vérifiables.

1.4 Limites des approches existantes

Malgré les progrès récents en matière d'extraction de données et de traitement automatique du langage naturel (TAL), les systèmes de gestion documentaire actuels manifestent des faiblesses structurelles lorsqu'ils sont déployés dans des environnements industriels complexes. Les limites des solutions standards peuvent être articulées autour de quatre axes majeurs.

1.4.1 Limites des systèmes documentaires classiques

Les outils traditionnels de gestion électronique des documents (GED) reposent généralement sur des logiques de classification rigides et statiques. Ces systèmes s'avèrent incapables de reconstruire la cohérence organique des documents complexes sans imposer des traitements spécifiques et lourds pour chaque typologie de fichier [37]. De surcroît, les interfaces de recherche ou de agents conversationnels standards ne possèdent pas la souplesse algorithmique requise pour interpréter des requêtes métier complexes ou pour interagir dynamiquement avec des structures de données industrielles hétérogènes.

1.4.2 Limites des modèles NLP généralistes

L'application de grands modèles de langage (LLM) pré-entraînés exclusivement sur des corpus généralistes (tels que le web ou des flux d'actualités) s'avère insuffisante dans le secteur industriel. Ces architectures ne capturent pas la syntaxe spécialisée. En l'absence d'un mécanisme d'adaptation contextuelle ou d'un ajustement précis (*fine-tuning*), ces modèles génèrent des réponses approximatives ou hors de propos, car ils restent incapables de refléter la variabilité linguistique et technique inhérente aux documentations d'entreprise [22].

1.4.3 Problèmes de gestion des documents techniques industriels

La documentation technique se caractérise par une forte hétérogénéité et une nature profondément non linéaire. L'information essentielle y est fréquemment fragmentée entre du texte libre et des tableaux semi-structurés. De plus, l'environnement industriel présente une densité acronymique complexe et des nomenclatures spécifiques (notamment les *tags* d'équipements) qui varient d'une unité de production à une autre. Ces particularités internes et non standardisées rendent les bases de connaissances externes inopérantes ou trop globales pour alimenter un système d'assistance expert fiable [38].

1.4.4 Manque d'intégration des pipelines (extraction → recherche → génération)

L'un des verrous technologiques majeurs réside dans le cloisonnement des solutions existantes. Rares sont les dispositifs qui parviennent à unifier de manière fluide et transparente l'extraction de l'information, sa structuration fine, sa recherche sémantique et son exploitation conversationnelle au sein d'une architecture unique [23]. Cette absence d'interopérabilité de bout en bout nuit à la cohérence globale de l'assistance et dégrade la fidélité des informations restituées.

C'est précisément pour surmonter ce manque d'intégration et répondre aux exigences de la documentation industrielle que se positionne le projet RAGbot Naftal, visant à articuler un pipeline d'extraction multimodal sur mesure et une stratégie de recherche hybride au sein d'un assistant fiable et hautement spécialisé.

1.5 Étude du système documentaire NAFTAL

Le système documentaire de NAFTALrepose sur un écosystème complexe. Cette étude analyse la typologie des flux documentaires, les outils de gestion en place , afin d'évaluer leur efficacité au regard des exigences de fluidité opérationnelle.

1.5.1 Cartographie du processus documentaire et des flux d'information actuels

L'analyse des entretiens fonctionnels a permis de modéliser le processus actuel de gestion commerciale et d'immobilisation des équipements au sein de l'entreprise. Le flux d'information s'articule autour d'une séquence stricte d'étapes manuelles et semi-automatisées impliquant plusieurs départements (Commercial, Technique, et Direction Générale).

Le cycle de vie documentaire d'une transaction type se décompose ainsi :

Création de l'identité client (*Onboarding*) : Le processus débute par la création d'un « code client » unique dans le système. Sans ce prérequis, aucune transaction commerciale ne peut être initiée.

Constitution et validation du dossier : Une fois le code généré, le client doit soumettre un dossier administratif complet. Ce dossier subit une double validation : un contrôle de conformité initial, suivi d'une approbation formelle par la Direction Générale, qui délivre ensuite une fourchette de codes spécifiques (e.g., codes alphanumériques pour éviter les collisions entre directions régionales).

Saisie de la commande commerciale : Une fois le client validé et sa carte client délivrée, il peut émettre sa commande. Le système vérifie alors la disponibilité des produits en stock.

Processus de facturation (Immobilisation ou Vente) : C'est à cette étape que le système se ramifie en deux circuits distincts :

- **Cas 1 : Vente directe.** Le produit est disponible, la facturation est émise, et le paiement est exigé (au comptant ou à terme).
- **Cas 2 : Immobilisation (Transfert de stock vers investissement).** Si l'équipement n'est pas vendu au client final mais utilisé en interne par NAFTAL (ex. : compteurs volumétriques, pièces de rechange pour camions), le système génère un flux d'immobilisation. Le produit sort du stock (création d'un Bon de Mouvement Interne – BMI) et entre dans la catégorie des investissements de l'entreprise.

Circuit d'approbation et transfert : Si un équipement immobilisé doit être réparé ou réaffecté (réforme), un processus lourd d'approbation est déclenché. Le dossier de réforme doit être soumis, justifié obligatoirement avec des photographies, puis validé par une commission locale, et enfin par la commission de la Direction Générale (Commission d'Ambdia) avant d'autoriser la vente ou la destruction du bien.

1.5.2 Évaluation critique et limites opérationnelles du modèle existant

Malgré sa structuration hiérarchique, le système actuel de NAFTAL présente de profondes vulnérabilités conceptuelles et opérationnelles, justifiant pleinement le besoin d'une transition vers une architecture cognitive basée sur le RAG.

1. **Silos d'information et opacité des nomenclatures :** Le système souffre d'une déconnexion entre le nom commercial d'un équipement et sa nomenclature technique d'immobilisation. Un même objet (ex. : un compteur) peut posséder une référence dans le système de facturation et une nomenclature totalement différente une fois transféré en investissement. Cette absence de référentiel unifié (Graphe de Connaissances) rend la traçabilité des actifs (ex. : retrouver l'historique d'un « Volucompteur » après 10 ans) complexe et sujette à l'erreur humaine.

2. **Lourdeur des processus d'approbation (Goulots d'étranglement) :** Les flux nécessitent des validations physiques en cascade. Par exemple, la réforme d'un équipement impose l'intervention d'une commission, la soumission de dossiers papier (photos imprimées) et l'approbation de la Direction. Ce processus manuel ralentit considérablement la chaîne de valeur et génère une masse critique de documents non structurés difficiles à archiver et à rechercher.

3. **Absence de contrôle sémantique :** Le système actuel repose sur des requêtes de bases de données strictes (SQL) basées sur des codes fixes (ex. : chercher le code `"C2378"`). Si un opérateur fait une faute de frappe ou cherche l'équipement par sa description technique plutôt que par son code arbitraire, le système est incapable de retourner l'information. L'absence de moteur de recherche sémantique limite la capacité des agents à retrouver l'information de manière intuitive.

TAB. 1.2 : Synthèse des limites du modèle existant et opportunités de l'architecture cognitive

Point Critique	Dysfonctionnement Constaté	Impact Opérationnel
Nomenclatures	Déconnexion facturation / investissement.	Rupture de la traçabilité à long terme des actifs.
Validations	Flux physiques en cascade, dossiers papier, photos.	Goulots d'étranglement, allongement des délais de réforme.
Indexation	Requêtes SQL strictes sur codes exacts (`C2378`).	Vulnérabilité aux fautes de frappe, aucune recherche intuitive.

1.6 Conclusion

En somme, ce premier chapitre a permis de mettre en exergue la complexité inhérente à la gestion des connaissances dans un environnement industriel aussi exigeant que celui de NAFTAL. L'analyse des approches existantes démontre que les méthodes conventionnelles de recherche documentaire et les assistants conversationnels basiques ne parviennent pas à pallier les difficultés liées à l'hétérogénéité des formats techniques, à la densité des acronymes métiers et à la structure non linéaire des manuels d'ingénierie. La conclusion est évidente : se concentrer sur des solutions d'intelligence artificielle plus robustes est essentiel pour transformer cet héritage d'informations statiques en un levier de performance opérationnelle. Le passage d'un système de recherche simplement lexical à une architecture sémantique et intégrée, capable de lier l'extraction automatique à une interface de dialogue intelligente, constitue le fondement de notre étude. La création du Ragbot NAFTAL, une solution conçue pour combiner l'extraction, la structuration et l'exploitation de diverses techniques, est ainsi justifiée par cette investigation pilote. Ce diagnostic jette les bases du chapitre suivant, où nous examinerons les fondements théoriques de l'apprentissage automatique et de l'apprentissage profond qui permettent la création de cet outil.

Chapitre 2

Méthodologie

Ce chapitre détaille la méthodologie adoptée pour la conception d'un système de Génération Augmentée par Récupération (RAG) appliqué à la documentation technique de NAFTAL. Notre démarche repose sur la mise en place d'un pipeline modulaire capable de transformer des documents complexes en une base de connaissances interactive. Le processus s'articule autour de l'ingestion sémantique des données, d'un mécanisme de recherche hybride assisté par un reclassement (Reranking), et d'une orchestration intelligente entre modèles. Cette approche vise à garantir une précision technique rigoureuse tout en respectant les impératifs de confidentialité et de souveraineté des données industrielles.

2.1 Vue d'ensemble de l'architecture du système RAG

L'architecture implémentée dans ce travail repose sur un pipeline de génération augmentée par récupération (RAG) multimodal. Le flux de traitement global s'articule autour de deux composantes majeures :

- **Stockage Vectoriel Multimodal** : Le pipeline assure la transformation des documents hétérogènes (textes et images) en représentations numériques (embeddings) de haute dimension. Ces données sont indexées dans un **Vector Store**, permettant une recherche basée sur la similarité cosinus plutôt que sur une simple correspondance de caractères [19].

- **Recherche Hybride** : Afin de garantir une haute précision , le système combine simultanément la recherche **sémantique** (dense) et la recherche **lexicale** (sparse). Cette approche hybride permet de capter la profondeur contextuelle de la requête tout en respectant la terminologie technique spécifique au domaine industriel de NAFTAL [20, 39, 40].

2.1.1 Description générale du pipeline RAG

Le fonctionnement de notre système repose sur un flux séquentiel et itératif, conçu pour transformer une requête brute en une réponse experte et sourcée. Ce pipeline se décompose en cinq étapes majeures :

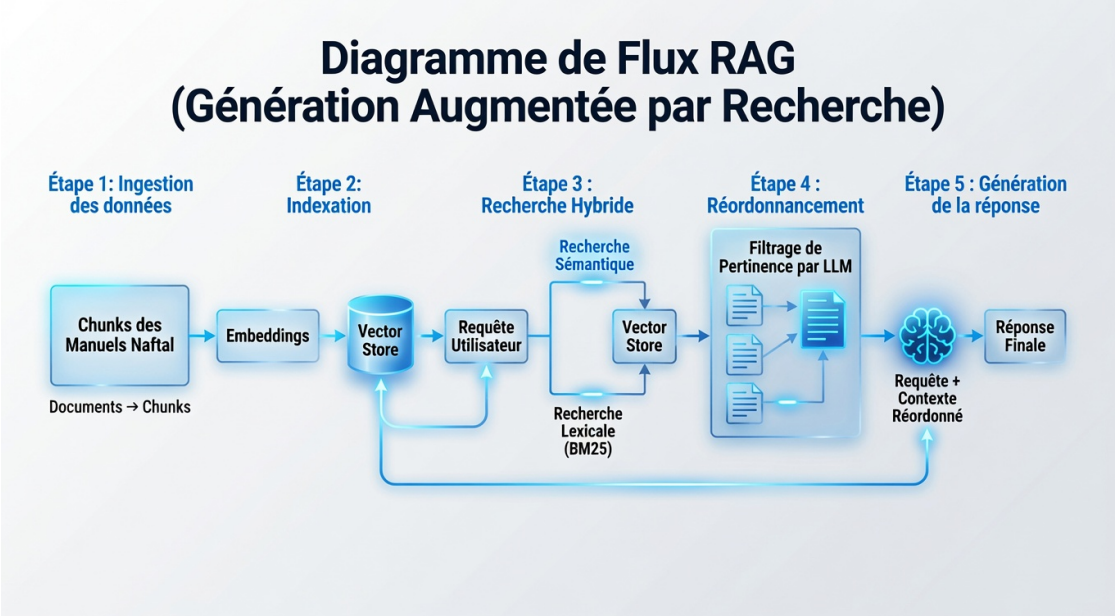


Fig. 2.1 : Diagramme de Flux RAG (Génération Augmentée par Recherche)

1. **Ingestion des données** : Cette phase initiale consiste à extraire le contenu des manuels techniques de Naftal. Les documents sont segmentés en fragments cohérents (Chunking) afin de conserver la granularité de l'information tout en respectant les limites de contexte des modèles.

2. **Indexation** : Chaque fragment est converti en vecteurs numériques (Embeddings) représentant sa sémantique. Ces vecteurs sont indexés dans un espace multidimensionnel au sein du **Vector Store**, ce qui permet de structurer la "mémoire externe"du système.

3. **Retrieval Hybride** : Lors de la soumission d'une requête, le système effectue une double recherche : (a) **Recherche Sémantique** pour identifier les concepts proches du sens de la question, (b) **Recherche Lexicale (BM25)** pour garantir la correspondance précise des termes techniques et des codes d'équipement. Cette combinaison permet de maximiser le rappel (Recall) des documents pertinents [40].

4. **Reranking (Ré-ordonnancement)** : Les documents extraits lors de la phase de recherche hybride sont soumis à un modèle de ré-ordonnancement. Ce composant évalue plus finement la pertinence de chaque fragment par rapport à la requête pour ne conserver que les plus saillants, réduisant ainsi le bruit informatif transmis au générateur.

5. **Génération de réponse** : Le modèle de langue (LLM) reçoit la requête de l'utilisateur ainsi que les documents ré-ordonnés. Il procède à une synthèse multimodale pour produire une réponse finale claire, fluide et strictement ancrée dans les sources récupérées, garantissant ainsi la fiabilité du système.

2.1.2 Choix d'une architecture hybride Dense + Sparse

Pour garantir la robustesse du système face à la terminologie technique de NAFTALet à la diversité linguistique , nous avons adopté pour une architecture de recherche hybride combinant les approches **Dense** et **Sparse**.

Critère	TF-IDF	BM25	Recherche Dense	Approche Hybride
Fondement théorique	Pondération statistique (fréquence terme × inverse fréquence document).	Modèle probabiliste dérivé de TF-IDF avec normalisation de longueur.	Plongement lexical dans espace vectoriel continu (embeddings).	Combinaison de modèles lexicaux et sémantiques.
Capture sémantique	Nulle (correspondance lexicale stricte requise).	Faible (améliorations heuristiques limitées).	Significative (synonymes, paraphrases, concepts).	Optimale (dualité lexicale-sémantique).
Traitement jargon	Bonne (préservation des termes rares via IDF).	Excellente (paramétrage ajustable pour termes techniques).	Limitée (dilution des tokens rares).	Robuste (compensation par composante lexicale).
Normalisation longueur	Approximative (via normalisation L2).	Explicite (paramètre b de pénalisation).	Implicite (via pooling ou moyenne).	Multiple (selon chaque composante).
Complexité	Faible (calcul matriciel simple).	Faible à modérée (index inversé optimisé).	Considérable (inférence modèle + ANN).	Très élevée (double index + fusion).
Adaptabilité multilingue	Nulle (nécessite traitement préalable).	Nulle (indépendant de la langue mais sans transfert).	Native (modèles multilingues pré-entraînés).	Native (héritée de la composante dense).

TAB. 2.1 : Comparaison des paradigmes de Recherche d'Information (RI) : des méthodes lexicales classiques aux approches hybrides contemporaines.

Analyse comparative des paradigmes de RI Le tableau 2.1 synthétise l'évolution et l'articulation des paradigmes de recherche d'information. Les approches lexicales probabilistes,

à l'instar de BM25, assurent une indexation rigoureuse de la terminologie technique et des codes d'équipement, bien qu'elles demeurent vulnérables au décalage de vocabulaire (*vocabulary mismatch*). Inversement, la recherche dense par plongements vectoriels (*embeddings*) s'affranchit des barrières linguistiques en capturant l'intention conceptuelle implicite, au détriment d'un risque de dilution des identifiants uniques dans l'espace continu. L'architecture hybride opère ainsi une synthèse critique : en conjuguant la précision nominale du modèle lexical et la flexibilité sémantique du modèle dense, elle constitue le compromis optimal pour traiter la double exigence — terminologique et conceptuelle — d'un corpus industriel multilingue.

2.1.3 Intégration multimodale (texte + images)

La particularité des manuels techniques de NAFTALréside dans la complémentarité entre le texte descriptif et les éléments visuels (schémas, graphiques, photos d'équipements). Pour traiter cette hétérogénéité, notre système implémente une gestion conjointe via deux technologies clés : **OCR** et **CLIP**.

1. **Extraction textuelle et spatiale via OCR** : Nous utilisons l'**OCR** (Optical Character Recognition) pour numériser non seulement le texte brut, mais aussi pour extraire les données textuelles imbriquées dans les schémas techniques. Cela permet de transformer des informations visuelles statiques en données textuelles indexables, garantissant que les légendes et les annotations des figures ne soient pas perdues lors du processus d'ingestion.

2. **Représentation sémantique visuelle avec CLIP** : Pour une compréhension plus profonde des images, nous intégrons le modèle **CLIP** (Contrastive Language-Image Pre-training). Contrairement à l'OCR qui se limite aux caractères, CLIP projette les images et leurs descriptions textuelles dans un espace vectoriel commun. Cela permet au système d'effectuer des recherches par similarité visuelle : une requête textuelle peut ainsi récupérer une image pertinente même si celle-ci ne contient aucun texte, simplement par proximité sémantique entre le concept et le contenu visuel [41].

3. **Fusion pour une réponse contextuelle** : L'intégration conjointe de l'OCR et de CLIP permet au pipeline de fournir au générateur (LLM) un contexte enrichi. Lors de la phase de récupération, le système extrait simultanément les paragraphes explicatifs et les schémas correspondants. Cette dualité d'information est cruciale pour que le modèle final puisse générer des réponses précises aux questions du type : "Quelle est la quantité totale de « VOLUCOMPTEUR SIMPLEX » reçue selon le résumé de l'ARI Detail ?" [42].

2.2 Modélisation vectorielle et stratégies d'indexation

La transformation du corpus documentaire en un espace vectoriel structuré forme le cœur du moteur de recherche sémantique de notre architecture RAG. La transformation convertit les informations hétérogènes en représentations mathématiques compactes qui saisissent la sémantique et le contexte des manuels techniques. Dans cette section, nous détaillons les stratégies que nous avons choisies pour la modélisation, de la mise en place de la base de données vectorielle à la création de règles de découpage et de gestion des métadonnées. Le but principal est de garantir une indexation solide. Cette indexation assure une récupération d'information rapide, précise et adaptée aux exigences du méti

Critère	Qdrant	ChromaDB	FAISS
Persistance	**Native sur disque.** Redémarrage instantané sans perte d'index.	Locale (SQLite) ou en mémoire vive (prototypage).	**En RAM.** Sauvegarde et rechargement manuel de l'index requis.
Métadonnées	**Avancée.** Pré-filtrage natif très performant via les *payloads*.	Basique. Adapté aux systèmes RAG standards.	**Inexistante.** Nécessite un couplage avec une base SQL externe.
Opérations CRUD	Dynamiques et temps réel (Insertion/Mise à jour atomiques).	Support complet via une API haut niveau.	Limitées. La modification exige souvent la reconstruction de l'index.
Indexation	HNSW paramétrable. Distances : Cosinus, Euclidienne, Scalaire.	HNSW standard. Configuration interne peu exposée.	Contrôle total des algorithmes (Flat, IVFFlat, PQ).
Déploiement	API REST/gRPC. Cloud-native (Docker/K8s), haute scalabilité.	Client/Serveur ou mode local embarqué.	Librairie pure. Exige le développement d'un backend sur mesure.

TAB. 2.2 : Analyse comparative des bases de données vectorielles pour l'architecture RAG.

2.2.1 Implémentation et paramétrage de la base vectorielle Qdrant

La mise en œuvre de **Qdrant** au sein de notre architecture repose sur une configuration optimisée pour la persistance des données et la précision de la recherche sémantique. Cette étape garantit que la "mémoire"du système RAG reste stable et performante.

1. **Persistance native et continuité de service :** Contrairement aux approches exclusive-

ment *in-memory*, Qdrant stocke durablement les index et leurs métadonnées (*payloads*) sur disque [43]. Cette architecture évite les phases de ré-extraction et de ré-indexation de données, extrêmement lourdes en ressources, garantissant ainsi la tolérance aux pannes et la disponibilité du système.

2. **Normalisation sémantique par similarité cosinus :** Afin d'exploiter de manière optimale les plongements (*embeddings*) du modèle BGE-M3 [44], la proximité sémantique entre le vecteur requête **q** et le vecteur document **d** est évaluée par le cosinus de leur angle θ [31] :

$$\text{sim}(\mathbf{q}, \mathbf{d}) = \cos(\theta) = \frac{\mathbf{q} \cdot \mathbf{d}}{\|\mathbf{q}\|\|\mathbf{d}\|} = \frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} q_i^2}\sqrt{\sum_{i=1}^{n} d_i^2}} \tag{2.1}$$

Cette métrique s'affranchit de la longueur des textes ou de la fréquence brute des termes pour se focaliser uniquement sur l'alignement directionnel, permettant de comparer équitablement des fragments techniques de tailles hétérogènes.

3. **Indexation HNSW et compromis performance/latence :** La recherche des plus proches voisins s'appuie sur l'algorithme de graphe HNSW (*Hierarchical Navigable Small World*) [45]. L'*Indexing Optimizer* de Qdrant a été finement paramétré afin de maintenir un équilibre optimal entre une ingestion fluide des nouveaux documents de NAFTAL et un taux de rappel (*recall*) élevé lors des requêtes intensives [46].

2.2.2 Structuration des métadonnées

Pour transformer notre base vectorielle en un système d'information structuré et exploitable, nous avons enrichi chaque vecteur par un ensemble de métadonnées (Payload). Cette structuration est indispensable pour permettre un filtrage granulaire et améliorer la pertinence du contexte fourni au modèle de génération.

Chaque point dans la base **Qdrant** inclut les champs suivants :

`doc_type` **(Nature du document)** : Ce champ catégorise l'origine de l'information (ex : Manuel de maintenance, Procédure de sécurité, Catalogue de pièces). Il permet au système de prioriser certains types de documents selon la nature de la requête utilisateur.

`chunk_type` **(Nature du fragment)** : Nous distinguons ici les fragments purement **textuels** des extraits issus de l'**OCR** d'images ou de tableaux. Ce marquage est crucial pour que le générateur sache s'il traite une description narrative ou une donnée structurée issue d'un schéma technique.

`source` **et** `page_map` : Chaque fragment est lié à sa source exacte (nom du fichier PDF) et à sa localisation physique (numéro de page). Cette traçabilité est une exigence métier

pour NAFTAL, permettant à l'utilisateur final de vérifier l'information à la source et de garantir la réduction des hallucinations.

L'intégration de ces métadonnées permet d'exécuter des **requêtes filtrées**. Par exemple, le système peut restreindre la recherche sémantique uniquement aux "procédures de sécurité"si l'intention de l'utilisateur concerne une manipulation critique, augmentant ainsi considérablement la précision du système (Precision@K).

2.2.3 Stratégie de découpage des documents (chunking)

Le découpage des documents, ou **chunking**, est une étape déterminante pour l'efficacité de la recherche sémantique. Une segmentation trop fine risque de perdre le contexte, tandis qu'une segmentation trop large peut diluer l'information pertinente. Nous avons adopté une stratégie de découpage hybride et adaptée aux spécificités de notre corpus.

1. **Découpage récursif par caractères (Recursive Character Splitting)** : Nous utilisons un algorithme de découpage récursif qui privilégie les séparateurs logiques (paragraphes, fins de phrases, puis espaces). Cette méthode permet de garantir que les concepts techniques ne sont pas coupés arbitrairement au milieu d'une phrase, préservant ainsi la cohérence syntaxique et sémantique, particulièrement cruciale pour les structures complexes de la langue arabe et du français technique.

2. **Paramétrage de la taille et du chevauchement (Overlap)** :

   - **Taille du Chunk** : Nous avons fixé une fenêtre de **512 à 1024 tokens**, ce qui correspond à la zone de performance optimale du modèle **BGE-M3**.
   - **Fenêtre de chevauchement (Contextual Overlap)** : Un chevauchement de **10 à 15%** est appliqué entre deux fragments successifs. Ce mécanisme permet de conserver le lien contextuel : une entité nommée ou un équipement mentionné à la fin d'un fragment sera également présent au début du suivant, évitant ainsi les pertes d'informations aux frontières de découpage.

3. **Adaptation aux structures multilingues** : La stratégie de découpage tient compte des spécificités typographiques. Pour l'arabe, le système identifie les signes de ponctuation spécifiques (comme la virgule inversée « ») pour affiner les points de rupture. Pour les manuels de NAFTAL, cette approche garantit que les listes à puces et les procédures de maintenance restent groupées de manière logique, facilitant ainsi leur indexation vectorielle.

2.2.4 Indexation incrémentale et gestion des doublons

Pour assurer l'évolutivité et la maintenance du système, nous avons mis en place un mécanisme d'indexation incrémentale. L'objectif est de permettre la mise à jour de la base de connaissances sans réindexer l'intégralité du corpus, tout en garantissant l'absence de redondance informationnelle au niveau des documents [20].

1. **Mécanisme de Hashing au niveau fichier (Dédoublonnage) :** Avant chaque phase d'ingestion, le pipeline calcule une empreinte numérique unique (Hash) globale pour chaque document à l'aide de l'algorithme MD5 [47]. Contrairement à une approche par fragment, ce processus de dédoublonnage s'opère à l'échelle du fichier complet. Si le hash MD5 d'un nouveau document correspond à une empreinte déjà enregistrée, le fichier est instantanément ignoré, évitant ainsi de saturer le système avec des données redondantes.

2. **Architecture de stockage distribuée (Qdrant et Couche de Cache) :** Une distinction nette est opérée entre la gestion des structures vectorielles et le stockage textuel brut. La base de données vectorielle Qdrant est exclusivement exploitée pour stocker les vecteurs d'embeddings ainsi que les métadonnées globales du document (telles que le titre, la source ou la date de publication) [43]. En parallèle, une couche de cache dédiée est configurée pour conserver et indexer les fragments de texte (*chunks*) associés, garantissant un accès rapide et structuré lors de la reconstruction du contexte.

3. **Mises à jour incrémentales et économie de calculs :** Cette stratégie d'ingestion par fichier permet d'ajouter de nouveaux manuels ou de modifier des sections spécifiques des procédures de NAFTAL de manière fluide. Seuls les fichiers modifiés (générant un hash MD5 inédit) ou nouvellement créés subissent le processus de découpage en fragments et d'inférence. Cela réduit considérablement la consommation des ressources de calcul (GPU/CPU) ainsi que le temps global de traitement.

4. **Cohérence de l'espace vectoriel :** En éliminant la redondance à la source (au niveau du document), nous optimisons les performances de la recherche de similarité. Un surplus de fragments identiques provenant de fichiers dupliqués viendrait biaiser les résultats du *Retrieval* en occupant les premières places du classement ($Top - K$) avec une information itérative, au détriment de la diversité contextuelle indispensable au modèle de génération [48].

2.3 Extraction multimodale des documents

L'extraction multimodale est l'étape initiale et critique où le système interprète la structure complexe des manuels techniques. Contrairement à une extraction textuelle classique,

elle traite simultanément les différents types de contenus pour ne perdre aucune information sémantique.

- **Extraction du Texte et Mise en Page** : Le système utilise des bibliothèques avancées (telles que `PyMuPDF` ou `Unstructured`) pour extraire le texte tout en préservant la hiérarchie des titres. Cela permet de conserver le contexte structurel (ex : savoir qu'une consigne appartient à la section "Sécurité"et non "Maintenance").

- **Analyse des Tableaux** : Les manuels de Naftal contiennent de nombreux tableaux de bord et spécifications techniques. L'extraction multimodale convertit ces tableaux en formats structurés (Markdown ou JSON). Cette transformation est essentielle pour que le modèle puisse effectuer des comparaisons précises entre les valeurs techniques lors de la génération de la réponse.

- **Indexation vectorielle directe des images (Modèle CLIP)** : Les schémas techniques et logigrammes sont extraits en tant qu'objets visuels. Au lieu de s'appuyer sur des modèles génératifs lourds pour décrire ces images (*Image Captioning*), notre système utilise un encodeur CLIP (*Contrastive Language-Image Pre-training*). Chaque image est directement transformée en un vecteur de plongement (*embedding*) projeté dans un espace latent multimodal commun au texte. Lors de la recherche, une mise en correspondance sémantique directe est effectuée entre la requête textuelle de l'utilisateur et l'embedding de l'image. Cette architecture garantit un coût d'inférence quasi nul (seul le calcul du vecteur est requis, sans génération de texte), optimisant la latence du système [41].

2.3.1 Extraction de texte depuis PDF et documents

L'extraction de texte à partir de documents techniques (souvent au format PDF) est une tâche complexe due à la variabilité des mises en page. Pour garantir une récupération fidèle, nous avons défini une stratégie combinant des outils spécifiques pour chaque modalité.

1. **Extraction textuelle et visuelle ciblée** : L'extraction du texte principal repose sur la bibliothèque **pdfplumber** [29]. Plutôt que d'appliquer une détection structurelle exhaustive de la mise en page, le pipeline gère la géométrie du document via un ajustement fin des tolérances spatiales (*tolerance tuning*). Cela permet de respecter l'ordre de lecture et d'éviter le mélange de colonnes distinctes. En complément, la bibliothèque **PyMuPDF (Fitz)** [30] est exclusivement exploitée pour identifier et extraire les objets visuels (images) intégrés au document.

2. **Détection automatique des couches textuelles** : Le pipeline intègre un mécanisme de détection automatique pour adapter son traitement selon la nature du fichier :

- **PDF Natifs** : Le système extrait le texte directement depuis la couche numérique en préservant l'encodage brut, sans appliquer de normalisation arbitraire sur les ligatures ou les caractères spéciaux.
- **Documents Scannés** : Si la couche textuelle est absente ou jugée inexploitable, le système bascule automatiquement vers un moteur de reconnaissance optique (OCR) [49] pour numériser le contenu à partir des images des pages.

3. **Post-traitement et dédoublonnage** : Une fois le texte extrait, le processus de nettoyage se concentre exclusivement sur un mécanisme de dédoublonnage (*deduplication*). Plutôt que de filtrer manuellement les numéros de page, les en-têtes ou les pieds de page, le système identifie et supprime les blocs de textes strictement redondants [50]. Ce filtrage évite d'indexer la même information plusieurs fois, optimisant ainsi la qualité des futurs fragments (*chunks*) sans risquer de supprimer du contenu utile par erreur.

2.3.2 Stratégie OCR Hybride : Gemini Vision et Tesseract

L'analyse des documents techniques révèle souvent la présence de schémas, de captures d'écran et de documents numérisés où le texte n'est pas codé numériquement. Pour transformer ces éléments visuels en données textuelles exploitables, nous avons mis en place une architecture à double moteur (Dual-OCR).

1. **Moteur OCR Principal (Gemini Vision) :** Au cœur de notre stratégie, Gemini Vision agit comme le moteur d'extraction primaire [42]. Ce choix architectural permet d'exploiter des capacités avancées de vision par ordinateur pour interpréter le contenu complexe des plans d'installations pétrolières ou des plaques signalétiques, offrant une précision sémantique bien supérieure à un OCR classique.

2. **Traitement par page complète :** Contrairement à une méthode qui consisterait à isoler et extraire chaque image individuellement, notre pipeline traite les pages numérisées dans leur intégralité. Cette approche est indispensable pour préserver la topographie globale, la disposition spatiale et le contexte structurel des informations au sein du document.

3. **Moteur secondaire et Traitement Multilingue (Tesseract) :** En complément de Gemini Vision, le moteur Tesseract OCR [49] est intégré pour traiter de manière ciblée la reconnaissance optique des caractères. L'un des défis majeurs des manuels techniques est la cohabitation de plusieurs langues. Nous avons donc configuré Tesseract avec les modèles de données anglais (eng), français (fra) et arabe (ara). Cette configuration trilingue permet au moteur de basculer dynamiquement entre les dictionnaires pour reconnaître avec précision le jargon technique anglophone ou francophone ainsi que ses annotations en arabe.

4. **Amélioration de la Qualité (Pre-processing) :** Avant l'extraction, les pages subissent un pipeline de transformations rigoureux pour optimiser le taux de réussite de l'OCR. Les trois étapes implémentées sont : la conversion en niveaux de gris, la réduction du bruit numérique (denoising) et l'ajustement dynamique du contraste. Ces opérations permettent d'obtenir un texte clair et nettoyé, prêt à être normalisé et converti en embeddings au même titre que le texte natif.

2.3.3 Indexation des images avec CLIP

Pour compléter l'approche textuelle, nous avons intégré une indexation visuelle directe basée sur le modèle **CLIP** (Contrastive Language-Image Pre-training). Cette étape permet de traiter les informations graphiques (schémas techniques, photos d'installations, logigrammes) qui sont difficilement traduisibles en texte pur par un simple OCR.

1. **Représentation vectorielle de l'image** : Le modèle CLIP utilise un encodeur visuel (Vision Transformer ou ResNet) pour projeter les images dans un espace latent partagé avec le texte. Chaque image extraite des manuels de NAFTALest transformée en un vecteur numérique. Contrairement à l'OCR qui cherche à lire des caractères, CLIP capture l'essence sémantique de l'image (ex : la structure d'une pompe ou le schéma d'un circuit de distribution).

2. **Alignement Texte-Image (Cross-modal Retrieval)** : L'atout majeur de cette méthode est la capacité du système à effectuer des recherches croisées. Une requête textuelle de l'utilisateur (ex : "schéma de vanne de sécurité") peut ainsi matcher directement avec une image indexée, même si cette image ne possède pas de description textuelle explicite. Le système calcule la similarité entre le vecteur de la requête et le vecteur de l'image dans le **Vector Store**.

3. **Stockage et Récupération Multimodale** : Ces vecteurs visuels sont stockés dans des collections dédiées au sein de **Qdrant**. Lors de la phase de *Retrieval*, le système peut ainsi fournir au générateur non seulement des extraits de texte, mais aussi les visuels les plus pertinents, offrant ainsi une réponse riche et complète qui facilite la compréhension des procédures techniques complexes.

2.4 Recherche hybride et fusion

2.4.1 Recherche sparse avec BM25

La recherche "sparse"(ou creuse) repose sur l'indexation par mots-clés et la fréquence d'apparition des termes. Pour cette composante, nous utilisons l'algorithme **BM25** (Best

Matching 25), qui est considéré comme la référence en matière de recherche d'information textuelle classique [40].

1. **Principe de pondération** : Contrairement à une simple recherche par mots-clés, le BM25 attribue un score à chaque document en fonction de la fréquence du terme de la requête dans le document (TF) et de la rareté du terme dans l'ensemble du corpus (IDF). Ce mécanisme permet de valoriser les termes techniques rares, tels que les codes d'équipements ou les substances chimiques spécifiques à NAFTAL, par rapport aux mots communs du dictionnaire.

2. **Saturation de la fréquence** : Le BM25 introduit une fonction de saturation qui limite l'impact de la répétition excessive d'un même mot dans un document. Cela empêche qu'un paragraphe soit jugé artificiellement pertinent simplement parce qu'il contient plusieurs fois le même terme technique, garantissant ainsi une meilleure précision du retrieval.

3. **Normalisation de la longueur** : L'algorithme prend en compte la longueur des documents pour éviter de favoriser systématiquement les fragments les plus longs. Cette normalisation est cruciale pour notre processus de chunking, car elle permet une comparaison équitable entre des fragments de texte de tailles légèrement différentes.

2.4.2 Fusion des résultats avec RRF

La fusion des résultats issus des recherches sparse (BM25) et dense (Vectorielle) est réalisée via l'algorithme **Reciprocal Rank Fusion (RRF)**. Cette méthode permet de combiner plusieurs listes de documents classés en une seule liste ordonnée, sans nécessiter une normalisation préalable des scores hétérogènes [51].

1. **Principe du classement réciproque** : Le RRF attribue un score à chaque document en fonction de son rang (r) dans les différentes listes de résultats. Le score final est calculé selon la formule :

$$\text{score}(d) = \sum_{r \in \text{rangs}} \frac{1}{k + r}$$

où k est une constante servant à lisser l'impact des documents très mal classés. Cette approche privilégie les documents qui apparaissent en tête de liste dans les deux méthodes de recherche simultanément.

2. **Indépendance vis-à-vis de l'échelle des scores** : L'avantage majeur de RRF pour notre architecture réside dans le fait qu'il ne s'appuie pas sur les valeurs brutes des scores (qui sont des distances cosinus pour Qdrant et des probabilités pour BM25), mais uniquement sur l'ordre de priorité. Cela élimine les biais liés à la disparité des échelles de mesure entre le moteur lexical et le moteur sémantique [51].

3. **Optimisation de la pertinence** : En agrégeant les forces du BM25 pour les mots-clés et du BGE-M3 pour le sens, le RRF garantit que le contexte envoyé au modèle de génération est le plus pertinent possible. Pour les manuels techniques de NAFTAL, cela assure que les procédures critiques citant des termes précis remontent systématiquement en tête de classement.

2.4.3 Sélection des candidats pour reranking

Après la phase de fusion par RRF, le système obtient une liste volumineuse de documents potentiels. Pour optimiser les ressources de calcul et la précision finale, nous procédons à une sélection d'un ensemble réduit de candidats destinés à l'étape de **Reranking** (reclassement profond).

1. **Réduction du bruit (Filtering)** : L'objectif est de ne conserver que les N premiers documents (généralement le Top-50 ou Top-100) ayant obtenu les meilleurs scores de fusion. Cette restriction permet d'éliminer les fragments dont la pertinence est jugée trop faible, évitant ainsi que le modèle de reclassement ne traite des informations hors sujet qui pourraient introduire de la confusion dans la génération finale [48, 20].

2. **Optimisation de la latence** : Le reranking étant une opération coûteuse en termes de temps de traitement (car elle nécessite une analyse croisée entre la requête et chaque fragment), limiter le nombre de candidats est essentiel pour garantir une réponse en temps réel. Cette sélection stratégique permet de maintenir un équilibre entre l'exhaustivité de la recherche et la réactivité du système technique.

3. **Préparation au Cross-Encoder** : Cet ensemble réduit de candidats sert de base de travail pour le modèle de reclassement (Cross-Encoder). En ne lui fournissant que les segments les plus prometteurs issus du moteur hybride, nous maximisons les chances de voir l'information la plus critique de NAFTALse hisser à la première position avant d'être transmise au générateur [39].

2.5 Reranking avec Cross-Encoder

Le processus de récupération initiale (Bi-Encoder) traite la requête et les documents de manière indépendante pour des raisons d'efficacité. Pour affiner la précision, nous introduisons une étape de **Reranking** utilisant un modèle **Cross-Encoder**, capable d'analyser simultanément la requête et le document pour capturer des interactions sémantiques fines [39].

1. **Analyse d'interaction profonde** : Contrairement aux modèles de recherche vectorielle classiques qui comparent des vecteurs pré-calculés, le Cross-Encoder prend en entrée

le couple (Requête, Fragment) et produit un score de pertinence unique. Ce mécanisme permet d'évaluer avec un haut degré de granularité si un extrait de manuel de NAFTALrépond spécifiquement à la question posée, en tenant compte des nuances syntaxiques et contextuelles que le Bi-Encoder aurait pu ignorer.

2. **Modèle BGE-Reranker** : Nous utilisons le modèle **BGE-Reranker**, optimisé pour les tâches de recherche multilingue. Ce modèle agit comme un filtre de qualité supérieure : il réordonne les candidats sélectionnés à l'étape précédente (Top-N) pour placer l'information la plus exacte en première position. Ce reclassement est crucial pour limiter les hallucinations du générateur en lui fournissant le contexte le plus "pur"possible [44].

3. **Filtrage par seuil de pertinence** : En plus du reclassement, le score produit par le Cross-Encoder permet d'appliquer un seuil de confiance. Si aucun document ne dépasse un certain niveau de pertinence après analyse profonde, le système peut informer l'utilisateur qu'aucune procédure correspondante n'a été trouvée dans la base de connaissances, garantissant ainsi la fiabilité des réponses techniques.

2.5.1 Exploitation de l'architecture bge-reranker-v2-m3 pour l'optimisation des systèmes de RAG

Pour l'étape de reclassement, nous avons intégré le modèle **BGE-Reranker-v2-m3**, un modèle de type **Cross-Encoder** conçu pour offrir une précision supérieure dans des contextes multilingues et multidimensionnels [44].

1. **Optimisation Multilingue (M3)** : Ce modèle est spécifiquement entraîné pour gérer simultanément plus de 100 langues, ce qui est crucial pour le corpus de NAFTAL. Il excelle dans l'alignement sémantique entre des requêtes en arabe et des documents techniques en français, ou inversement, en évaluant la pertinence sans dépendre d'une traduction intermédiaire qui pourrait altérer le sens technique.

2. **Interaction Globale des Tokens** : Contrairement aux modèles d'embeddings (Bi-Encoders) qui compressent un document en un seul vecteur, le **bge-reranker-v2-m3** permet une interaction complète entre chaque token de la requête et chaque token du document. Cette attention croisée (cross-attention) permet de détecter des nuances subtiles, comme une condition d'exclusion dans une procédure de sécurité, garantissant que le fragment le plus exact soit placé en tête du contexte.

3. **Capacité de traitement (Input Length)** : Le modèle supporte des séquences allant jusqu'à **8192 tokens**, ce qui lui permet d'analyser des fragments de texte étendus ou des tableaux complexes tout en conservant une vision globale du contexte. Cela assure une continuité sémantique forte lors de la sélection finale des données envoyées au générateur.

2.5.2 Principe du reranking par paires

Le reclassement par paires (pairwise reranking) constitue le cœur algorithmique du Cross-Encoder. Contrairement aux approches de recherche vectorielle isolées, ce principe repose sur une comparaison directe et simultanée entre la requête utilisateur et chaque candidat potentiel.

1. **Analyse de corrélation directe** : Dans ce mode de fonctionnement, le modèle prend en entrée une paire composée de la requête (Q) et d'un fragment de document (D). Au lieu de comparer deux vecteurs dans un espace latent, le modèle traite l'union $\{Q, D\}$ comme une seule séquence. Cela permet aux mécanismes d'attention du transformeur d'identifier précisément quels mots de la requête sont satisfaits par quelles parties du document, capturant ainsi des dépendances syntaxiques complexes [39].

2. **Calcul du score de pertinence** : Pour chaque paire, le modèle produit un score scalaire de confiance (généralement normalisé via une fonction Sigmoïde). Ce score n'est pas une simple mesure de distance, mais une estimation de la probabilité que le fragment D contienne la réponse exacte à la question Q. Cette comparaison granulaire est particulièrement efficace pour distinguer deux procédures techniques très proches mais s'appliquant à des équipements différents chez NAFTAL.

3. **Réordonnancement final** : Une fois que toutes les paires sélectionnées ont été évaluées, le système réorganise la liste des candidats par ordre décroissant de score. Ce processus garantit que le fragment possédant la plus forte corrélation sémantique avec la demande soit positionné au rang 1, maximisant ainsi la pertinence du contexte qui sera injecté dans le prompt du modèle de génération (LLM).

2.5.3 Sélection finale des documents

La sélection finale constitue l'étape ultime du pipeline de récupération avant la phase de génération. Elle consiste à restreindre l'ensemble des candidats reclassés par le Cross-Encoder à un nombre limité de fragments, désigné sous le terme de **Top-K final** [20].

1. **Arbitrage entre exhaustivité et précision** : Le choix de la valeur K (généralement comprise entre 3 et 5) est un paramètre critique. Un K trop élevé risque de saturer la fenêtre de contexte du modèle de génération avec des informations redondantes ou moins pertinentes, augmentant ainsi le risque de "bruit"sémantique. À l'inverse, un K trop faible pourrait omettre des détails techniques complémentaires nécessaires à une réponse exhaustive pour les agents de NAFTAL [48].

2. **Gestion de la fenêtre de contexte (Prompt Engineering)** : En limitant la sélection aux fragments ayant obtenu les meilleurs scores après reranking, nous optimisons l'utilisation de la mémoire du générateur (LLM). Cette compression sélective garantit que

seuls les extraits les plus probants sont injectés dans le prompt, ce qui améliore la cohérence de la réponse produite et réduit considérablement le coût computationnel lié au traitement de tokens superflus.

3. **Filtrage de confiance** : Dans cette phase, nous appliquons également un seuil de pertinence minimal. Si le score du document classé premier est inférieur à une valeur prédéfinie, le système peut choisir de ne sélectionner aucun document. Cette sécurité permet d'éviter que le modèle de génération n'essaie d'extrapoler une réponse à partir de documents faiblement liés à la requête, renforçant ainsi la fiabilité globale du système RAG.

2.6 Gestion du contexte et des réponses RAG

La gestion du contexte est le pivot central qui assure la fluidité et la pertinence de l'interaction. Elle consiste à administrer la mémoire de travail du système pour que chaque réponse s'inscrive dans une continuité logique, tout en respectant les limites techniques du modèle de génération [20].

1. **Gestion de la fenêtre de contexte** : Chaque modèle possède une limite maximale de tokens qu'il peut traiter simultanément. Le système doit donc arbitrer entre la taille de l'historique, la quantité de documents récupérés (Top-K) et les instructions système. Une gestion efficace utilise des techniques de **troncation intelligente** ou de **résumé de contexte** pour garantir que les informations les plus cruciales restent prioritaires dans le prompt final [52].

2. **Post-traitement et Vérification** : Une fois la réponse générée, une couche de contrôle final vérifie la cohérence du message. Cela inclut la validation des références citées par le RAG pour s'assurer que le modèle n'a pas inventé de sources inexistantes et le formatage final de la réponse (Markdown, tableaux, listes) pour faciliter la lecture par les techniciens de NAFTAL.

2.6.1 Construction du contexte enrichi

La construction du contexte enrichi consiste à transformer les fragments bruts (*chunks*) sélectionnés en une base de connaissances structurée et directement exploitable par le modèle de génération. Cette étape de concaténation ne se limite pas à un simple assemblage de textes, mais suit une logique d'organisation sémantique visant à maximiser l'exploitabilité des informations par le générateur [19, 20].

1. **Concaténation et Séparation** : Les *chunks* issus du *reranking* sont assemblés de manière séquentielle. Pour éviter que le modèle ne confonde deux procédures distinctes

ou deux sections de manuels différents, chaque fragment est délimité par un séparateur textuel (`---`) et introduit par une en-tête structurée (ex : `[Source: ... | Type: ... | Page: ...]`). Cette séparation explicite permet au modèle de langage (LLM) de traiter chaque unité d'information de façon autonome tout en préservant la cohésion de l'ensemble [48].

2. **Métadonnées et Provenance** : Chaque fragment est enrichi de ses métadonnées critiques extraites lors de la phase de récupération, à savoir la source, le type de document et le numéro de page. En injectant ces informations directement au sein du contexte, on permet au modèle de citer précisément la provenance des données dans sa réponse finale. Cette méthode s'inspire des approches d'extraction de connaissances appliquées aux documentations industrielles et de maintenance [24], renforçant ainsi la traçabilité et la crédibilité des consignes fournies aux agents de NAFTAL.


```json
{
  "document_type": "AMI_ENTREE",
  "document_number": "209LI20120023",
  "agency_code": "809",
  "agency_name": "AGENCE BLIDA",
  "operational_center_code": "209L",
  "operational_center_name": "ATELIER RÉSEAU CHIFFA BLIDA",
  "created_by": null,
  "accounted_by": null,
  "document_date": "2020-12-30T00:00:00",
  "acquisition_date": "2020-06-21T00:00:00",
  "total_value_ht": 24000.0,
  "asset_count": 1,
  "asset_lines": [
    {
      "barcode": "E3410-2R09-20-0001",
      "family_code": "E3410",
      "designation": "NETTOYEUR HP KARCHER H.T. 21/06/2020",
      "value_ht": 24000.0,
      "account_number": null,
      "acquisition_date": "2020-06-21T00:00:00"
    }
  ],
  "source_file": "AMI ENTRE.pdf",
  "extraction_timestamp": "2026-04-14T23:41:33.512863",
  "extraction_confidence": 1.0
}
```


FIG. 2.2 : Extrait JSON des métadonnées d'un document A.M.I

3. **Priorisation Spatiale** : L'ordre de concaténation suit directement la hiérarchie de pertinence définie par les scores de calcul issus de la suite de modèles BGE [44]. Les

fragments jugés les plus cruciaux pour répondre à la requête de l'utilisateur sont systématiquement positionnés en tête du contexte consolidé, garantissant une exposition immédiate des données prioritaires au composant de génération.

2.6.2 Intégration des images dans le contexte

L'intégration des images au sein du contexte enrichi est une étape clé pour traiter les documents de NAFTALqui contiennent des schémas techniques, des diagrammes de flux ou des photos d'installations. Cette intégration s'effectue selon deux approches complémentaires en fonction du mode de génération choisi [42].

1. **Approche par Vision (Mode Multimodal)** : Dans ce cas, les images originales (ou des extraits de haute qualité) sont encodées et transmises directement au modèle multimodal (comme **Gemini 2.5 flash**). L'image est insérée dans la séquence de messages au même titre que le texte, permettant au modèle d'effectuer un raisonnement spatial et visuel direct sur les composants techniques ou les signalétiques de sécurité.

2. **Approche par Description (Image-to-Text)** : Pour les modèles purement textuels ou pour enrichir la recherche, les images sont préalablement converties en descriptions textuelles détaillées. Ces descriptions capturent les éléments essentiels : légendes des schémas, étapes d'un logigramme ou état d'un indicateur de pression. Ces métadonnées textuelles sont alors concaténées aux fragments de texte classiques, rendant le contenu visuel "lisible"par n'importe quel LLM [41].

3. **Alignement Texte-Image** : Pour garantir la cohérence, chaque image ou description est associée au fragment de texte qui l'entourait dans le document original. Cet alignement contextuel évite que le modèle n'interprète un schéma de maintenance pour une pompe alors que la requête concerne un réservoir de stockage.

2.6.3 Génération de réponses basées sur contexte uniquement

Le principe de "Génération de réponses basées sur le contexte uniquement"est la pierre angulaire de la fiabilité du système. Il impose au modèle une contrainte de **fidélité stricte**, transformant le LLM d'un moteur de connaissances générales en un moteur de synthèse spécialisé pour NAFTAL [19].

1. **Mécanisme de "Groundedness"(Ancrage)** : Pour garantir que les réponses sont ancrées dans la réalité des documents internes, des instructions spécifiques interdisent au modèle d'utiliser ses connaissances pré-entraînées si elles contredisent ou complètent les fragments fournis. Si la réponse ne se trouve pas dans le contexte, le modèle est programmé pour répondre explicitement qu'il ne dispose pas de l'information, évitant ainsi toute improvisation risquée sur des procédures de sécurité.

2. **Atténuation des Hallucinations** : En limitant l'espace de réponse au seul contexte fourni, on réduit de manière notable les hallucinations. Le modèle agit comme un analyste qui extrait et reformule les données. Cette approche est vitale dans un environnement industriel où une erreur sur une température de stockage ou une pression limite pourrait avoir des conséquences graves [20].

3. **Citations et Vérifiabilité** : Le système exige que chaque affirmation technique soit accompagnée d'une référence aux documents sources fournis dans le prompt. Cela permet à l'utilisateur final de vérifier la véracité de la réponse en un clic, créant ainsi un lien de confiance entre l'opérateur et l'assistant intelligent [52].

2.7 Conclusion

En conclusion, ce chapitre a permis de définir l'architecture technique et la méthodologie rigoureuse nécessaires à la mise en place d'un système RAG performant pour NAFTAL. Nous avons détaillé les étapes cruciales allant de l'extraction multimodale des documents jusqu'à la génération finale, en passant par l'indexation vectorielle et le reclassement sémantique.

Cette conception a été guidée par un double impératif : d'une part, la précision technique, assurée par des mécanismes de recherche hybride, d'autre part, la robustesse opérationnelle, En structurant ainsi notre approche, nous avons créé un cadre robuste capable de transformer une documentation industrielle dense en un assistant intelligent et sécurisé.

Chapitre 3

Conception Architecturale

3.1 Introduction

La phase de conception constitue une étape fondamentale dans le cycle de vie d'un projet logiciel. L'objectif principal de ce chapitre est de structurer, modéliser et spécifier les aspects fonctionnels, structurels, comportementaux et techniques de la solution *Naftal Ragbot*. Cette solution, par sa nature hybride, intègre non seulement des bases de données relationnelles traditionnelles et des architectures d'API distribuées, mais également des mécanismes avancés d'intelligence artificielle : indexation vectorielle multimodale (textuelle et visuelle) et pipelines de génération augmentée par la recherche (RAG) [19, 20]. De ce fait, une conception rigoureuse est requise pour garantir des propriétés logicielles cruciales telles que l'extensibilité, la maintenabilité, la résilience et la sécurité des données d'entreprise.

3.2 Démarche de Conception et Modélisation UML

Pour garantir une conception formelle et solide, nous adoptons une démarche basée sur le langage **UML** (*Unified Modeling Language*), version 2.x [53]. UML est une norme de modélisation graphique standardisée par l'OMG, permettant de représenter les artefacts logiciels à travers une notation visuelle riche, claire et non ambiguë. Notre processus de modélisation suit une approche à plusieurs dimensions, structurée en trois vues complémentaires :

- **Vue des cas d'utilisation :** Elle permet de délimiter la frontière du système, d'identifier les différents acteurs et de décrire précisément ses fonctionnalités offertes.
- **Vue dynamique (ou comportementale) :** Elle montre le déroulement temporel et logique des processus clés (cycle de traitement d'une requête RAG, ingestion de documents) en utilisant les diagrammes de séquence.
- **Vue structurelle et persistance des données :** Elle détaille la conception et la persistance des données au sein du système. Cette vue expose la modélisation conceptuelle

et relationnelle des données traditionnelles (SQLite), ainsi que la modélisation logique dédiée au stockage vectoriel (Qdrant).

3.3 Architecture Logicielle Globale et Topologie des Services

3.3.1 Découpage en couches applicatives

L'architecture du système repose sur un modèle multi-tiers distribué, qui garantit une séparation stricte des préoccupations entre l'interface utilisateur, la logique métier centralisée et le moteur d'intelligence artificielle documentaire.

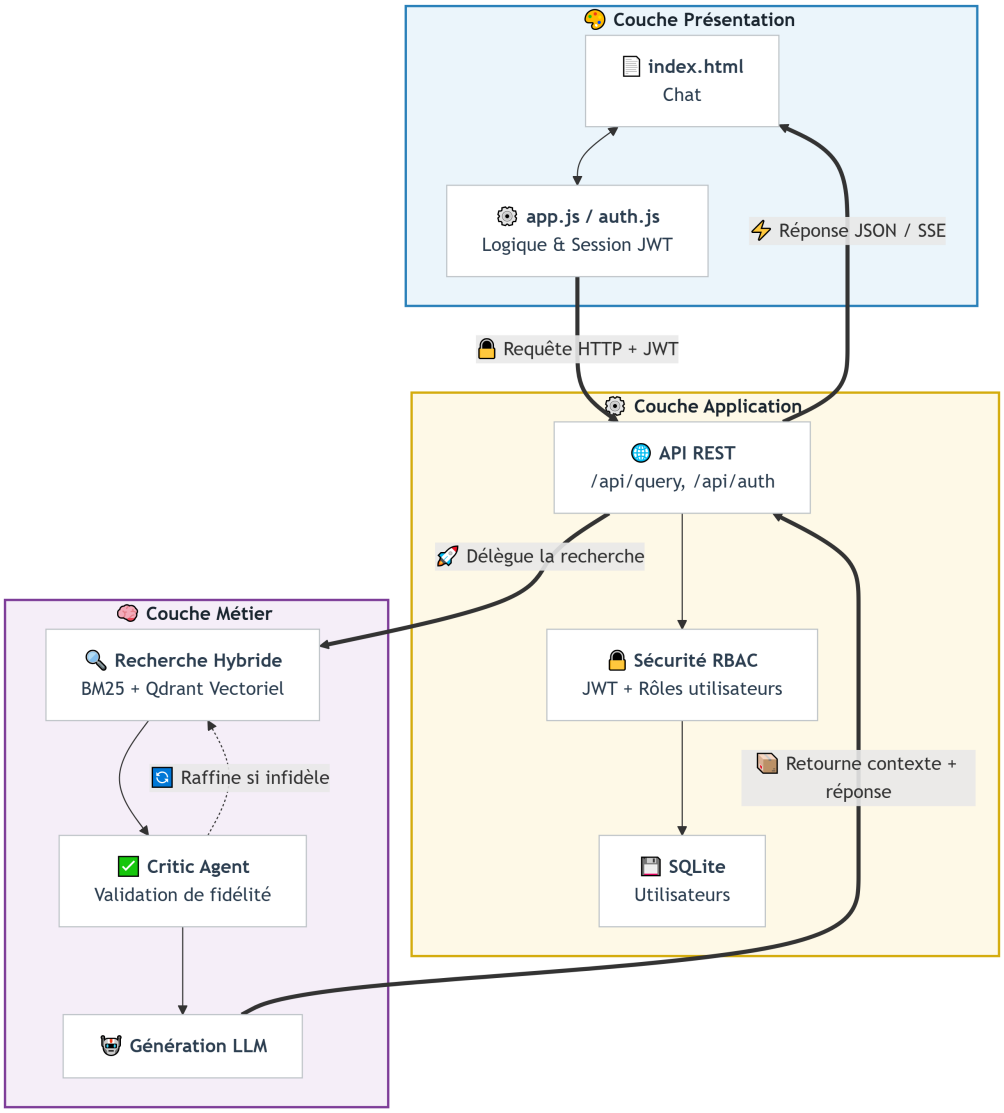


Fig. 3.1 : Architecture Globale du Système NAFTAL RAGbot.

1. **Couche I : Présentation (Frontend)**
   Située en périphérie du système, cette couche gère l'interaction directe avec l'utilisateur final. Elle comprend une interface de restitution couplée à un module de gestion des états. L'accès aux ressources est conditionné par l'utilisation de jetons de sécurité JWT (*JSON Web Tokens*) [54] , minimisant les vulnérabilités liées aux sessions traditionnelles.

2. **Couche II : Application (Backend)**
   Propulsée par le framework asynchrone `FastAPI` [55], cette couche sert de passerelle d'orchestration. Elle remplit trois fonctions critiques :

   - *Exposition des services :* Les services sont exposés via une API REST performante prenant en charge le protocole SSE (*Server-Sent Events*) [56] permettant de transmettre en (*streaming*) des réponses du LLM.
   - *Sécurité granulaire :* Un middleware de contrôle d'accès basé sur les rôles (RBAC) [57] intercepte chaque requête pour valider l'identité et les privilèges de l'appelant.
   - *Persistance métier :* Une base de données relationnelle SQLite [58] assure le suivi des profils utilisateurs, des métadonnées documentaires et des politiques de sécurité.

3. **Couche III : Décisionnelle (Moteur RAG & Alignement)**
   Cette couche représente le cœur scientifique du système. Contrairement aux approches RAG traditionnelles, ce moteur implémente une stratégie de validation active (*Self-RAG*) [59] :

   - *Recherche Hybride :* La requête subit une double vectorisation (lexicale via BM25 [40] et sémantique dense dans Qdrant [43, 46]).
   - *Évaluation par Agent Critique :* Un agent algorithmique évalue la pertinence et la fidélité des segments récupérés. En cas de contenu jugé insuffisant, une boucle de rétroaction affine les paramètres de recherche.
   - *Génération Contextuelle (LLM) :* Quand le contexte est validé, le modèle de langage génère la synthèse finale.

3.3.2 Spécification des interfaces de programmation (API REST)

La communication entre les composants est normalisée à travers une API REST documentée, dont les terminaux (*endpoints*) sont sécurisés par l'injection de dépendances `FastAPI` évaluant dynamiquement les habilitations RBAC. La matrice ci-dessous formalise la topologie des points d'accès et la répartition des privilèges.

Couche	Endpoint	Méthode	Authentif.	*Rôle Requis*
AUTH	`/api/auth/token`	POST	Basic Auth	–
	`/api/auth/me`	GET	JWT	–
QUERY	`/api/query`	POST	JWT	*privileged+*
	`/api/query/web`	POST	JWT (opt.)	*public+*
DOCS	`/api/documents/{id}`	GET	JWT	*privileged+*
	`/api/documents/list`	GET	JWT	*privileged+*
ADMIN	`/api/admin/users`	POST	JWT	*admin*
	`/api/admin/users`	GET	JWT	*admin*
	`/api/admin/users/{id}/role`	PUT	JWT	*admin*
	`/api/admin/upload`	POST	JWT + Form	*admin*
	`/api/admin/cache/clear`	POST	JWT	*admin*
VISION	`/api/vision/toggle`	POST	JWT	*admin*
EVAL	`/api/admin/evaluate`	POST	JWT	*admin*
	`/api/admin/evaluate`	GET	JWT	*admin*

Tab. 3.1 : Matrice de contrôle d'accès sémantique et d'habilitation RBAC.

L'examen du tableau 3.1 met en évidence une compartimentation stricte des privilèges, structurée selon le principe du moindre privilège. Les fonctions d'authentification (`AUTH`) agissent comme point de contrôle initial, substituant l'authentification basique par l'émission de jetons éphémères JWT. La répartition des rôles révèle une asymétrie fonctionnelle nécessaire : les requêtes de consultation (`QUERY`, `DOCS`) sont ouvertes aux utilisateurs accrédités standards, tandis que les interfaces de modification structurelle (`ADMIN`, `VISION`, `EVAL`) sont exclusivement restreintes au rôle *admin*. Cette isolation protège l'infrastructure contre les vecteurs d'attaque par l'escalade de privilèges.

3.4 Modélisation Fonctionnelle et Structurale (UML)

3.4.1 Identification des Acteurs

Le système distingue trois profils d'acteurs principaux, hiérarchisés selon leur niveau de privilège :

- **Utilisateur Public (*Public*) :** Acteur non authentifié qui possède un accès de base. Son périmètre est limité à l'espace public et à l'interrogation des données indexées depuis les sources externes (`naftal.dz`).
- **Utilisateur Privilégié (*Privileged*) :** Cadre de l'entreprise NAFTAL disposant d'un accès en lecture aux documents et sources internes indexés par le système RAG. Il per-

met l'interrogation des documents confidentiels et la consultation des passages sources associés, via des citations visuelles.

- **Administrateur Système (*Admin*) :** Acteur disposant du plus haut niveau de privilège. Il hérite des droits *Privileged* et assure la maintenance opérationnelle, la gestion des identités et l'ingestion documentaire.

3.4.2 Diagramme global des cas d'utilisation

Le diagramme de cas d'utilisation définit les fonctionnalités du système et les rôles des acteurs. Les capacités opérationnelles sont réparties en cinq cas principaux :

- *Query Public Sources :* Interrogation sémantique de sources externes sans authentification.
- *Query Internal Documents :* Exécution de requêtes complexes sur les référentiels confidentiels par le pipeline RAG.
- *Upload Document :* Ingestion de données (segmentation, vectorisation, indexation). Accessible aux acteurs *Privileged* et *Admin*.
- *Manage Users :* Gestion du cycle de vie des comptes et attribution des rôles (exclusif *Admin*).
- *Authenticate :* Vérification d'identité et génération des jetons JWT.

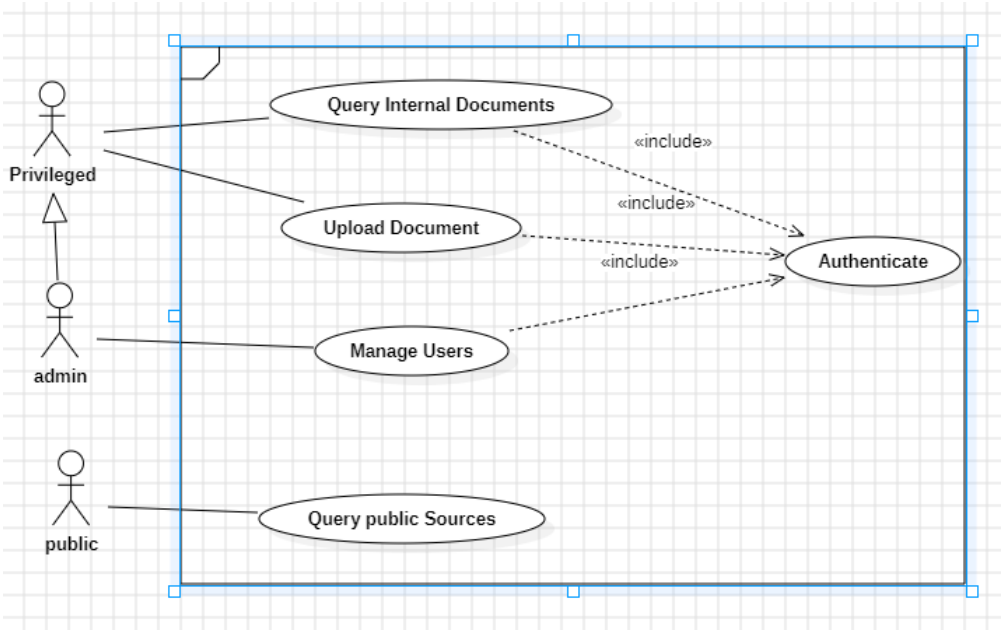


Fig. 3.2 : Diagramme global des cas d'utilisation du système *Naftal Ragbot*.

3.5 Modélisation de la Dynamique des Flux d'Exécution

La modélisation dynamique détaille la dimension temporelle et chronologique des interactions. Les diagrammes de séquence ci-dessous illustrent l'orchestration des échanges pour les deux processus critiques du système.

3.5.1 Orchestration de l'authentification sécurisée (RBAC & JWT)

Le protocole utilise une architecture sans état via des jetons JWT. Le flux se divise en deux phases :

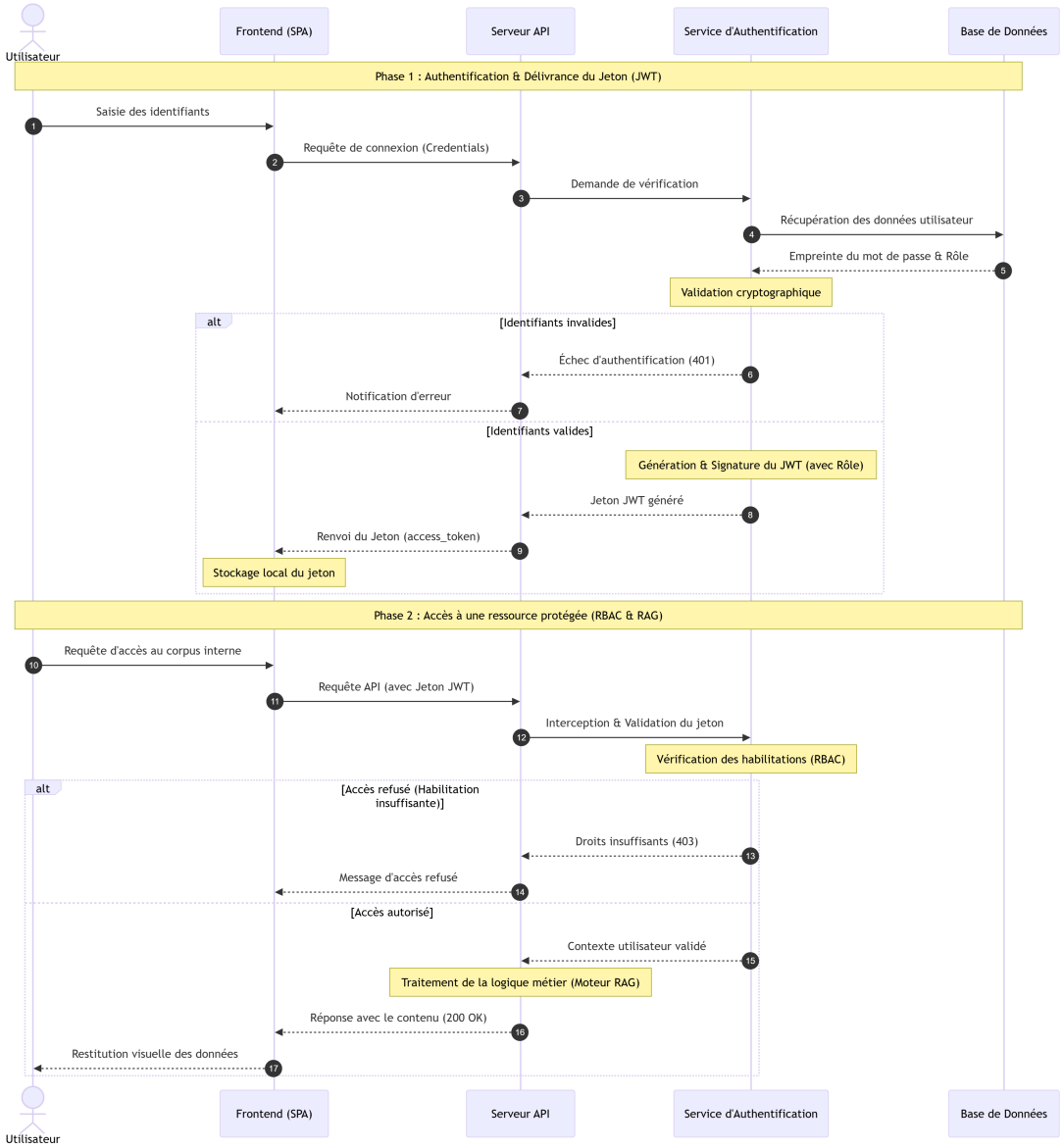


Fig. 3.3 : Diagramme de séquence : Orchestration de l'authentification et contrôle d'accès RBAC.

1. *Phase d'émission :* Le client soumet ses identifiants. Le backend valide l'empreinte `bcrypt` [60] contre la base SQLite et génère un jeton signé encapsulant l'identité et le

rôle.

2. *Phase d'autorisation :* Pour les requêtes subséquentes vers des endpoints protégés, l'API valide la signature, vérifie l'expiration et s'assure de la conformité du rôle avec les habilitations requises.

3.5.2 Ingestion, segmentation et vectorisation asynchrone

L'ingestion de documents par `/api/admin/upload` est une opération lourde. Pour garder la réactivité, le traitement est délégué à des tâches asynchrones (*Background Tasks*) :

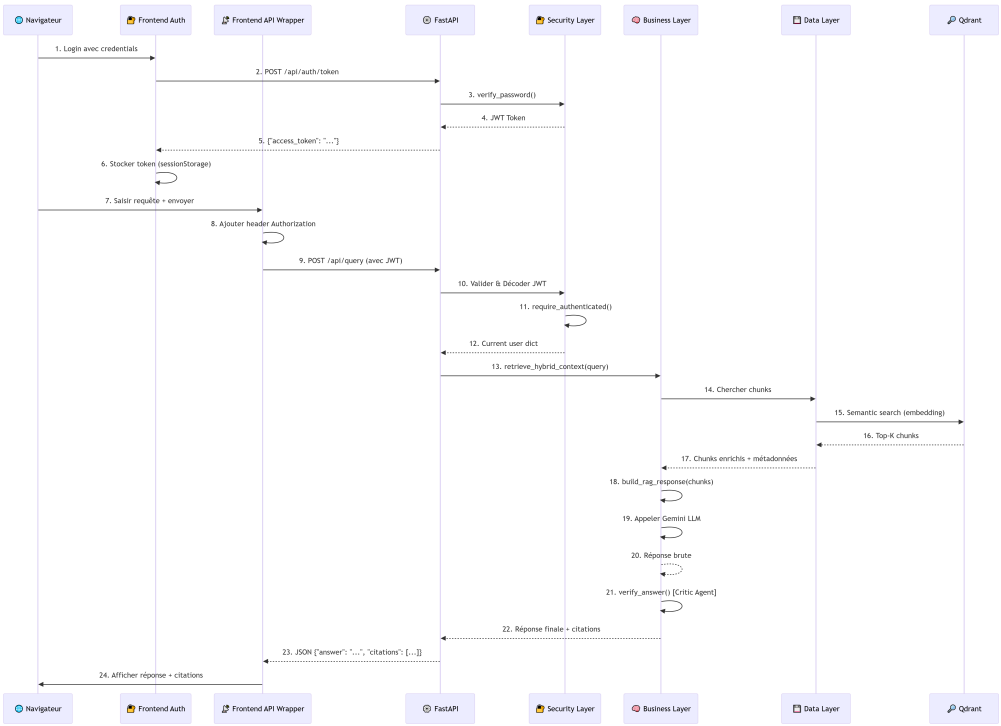


FIG. 3.4 : Diagramme de séquence : Pipeline asynchrone d'ingestion et de vectorisation.

1. *Délégation asynchrone :* Réception du document, validation des droits, réponse immédiate au client.

2. *Extraction Multimodale :* Traitement grâce à `PDFPlumber` [29] et `Tesseract OCR` [49] pour extraire texte et éléments graphiques.

3. *Segmentation (Chunking) :* Partitionnement du corpus extrait en unités textuelles cohérentes, intégrant un chevauchement contextuel (overlap) afin de préserver la continuité sémantique entre les segments adjacents.

4. *Vectorisation et Persistance :* Projection des segments dans un espace vectoriel de haute dimension (via des modèles de *embeddings*), suivie de leur indexation dans une base

de données vectorielle (Qdrant [43]), tandis que les métadonnées relationnelles sont stockées dans un système de gestion de base de données relationnelle (SGBDR).

3.6 Conception et Persistance des Données

3.6.1 Modélisation conceptuelle et relationnelle (SQLite)

La base relationnelle représente la source de vérité pour l'état du système, la gestion des identités et l'audit. Les entités principales sont :

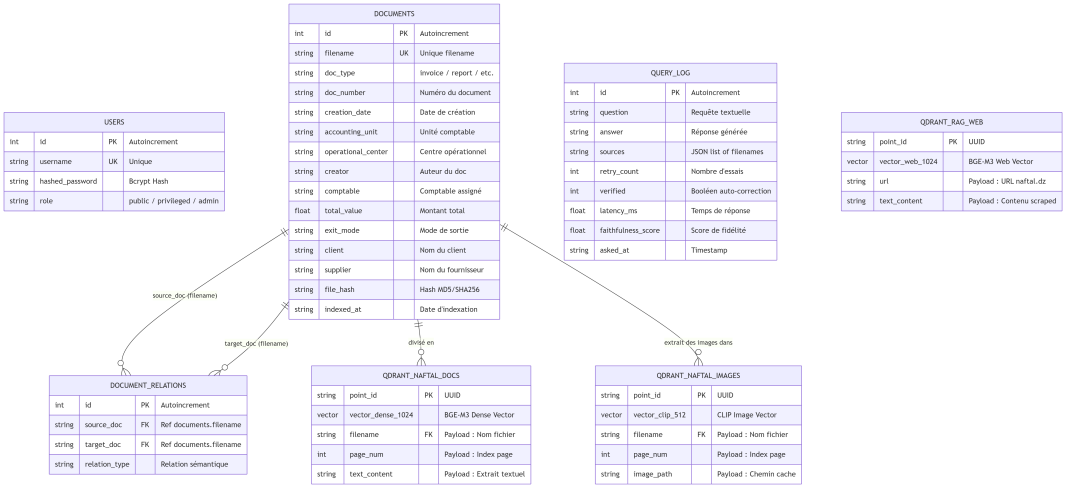


Fig. 3.5 : Modélisation conceptuelle de l'architecture de données relationnelle du système RAG

- **Utilisateurs (`USERS`) :** Stockage des informations d'identification, des empreintes cryptographiques `bcrypt` des mots de passe et des rôles associés au mécanisme RBAC (Role-Based Access Control).
- **Documents (`DOCUMENTS`) :** Registre d'ingestion enrichi de métadonnées métier (`accounting_unit`, `operational_center`, `client`, `total_value`), permettant un pré-filtrage contextuel.
- **Historique et Évaluation (`QUERY_LOG`) :** Audit des interactions intégrant les métriques de la boucle d'auto-correction (`retry_count`, `verified`, `faithfulness_score`).
- **Graphe de Connaissances (`DOCUMENT_RELATIONS`) :** Stockage des arêtes sémantiques entre fichiers pour l'enrichissement contextuel lors de l'inférence.

3.6.2 Modélisation logique du stockage vectoriel (Qdrant)

La recherche sémantique utilise une projection vectorielle structurée en trois collections distinctes :

- `QDRANT_NAFTAL_DOCS :` Collection dédiée aux documents internes de l'entreprise, vectorisés à l'aide du modèle `BGE-M3` [44] (1024 dimensions). Les métadonnées associées comprennent notamment le nom du document, le numéro de page et l'extrait textuel correspondant.

- `QDRANT_NAFTAL_IMAGES :` Collection destinée à l'indexation des contenus visuels. Les représentations sont générées par le modèle `CLIP` [41] (512 dimensions) et accompagnées des informations de localisation des images mises en cache.

- `QDRANT_RAG_PUBLIC :` Collection regroupant les ressources publiques issues du site `naftal.dz`. Les contenus textuels sont encodés à l'aide de `BGE-M3` [44] (1024 dimensions), tandis que les métadonnées intègrent l'URL de la source d'origine.

Le couplage *vecteur/payload* maintient une intégrité référentielle logique avec SQLite. L'attribut `filename` agit comme clé étrangère logique, permettant au moteur hybride de rejoindre les métadonnées métier et d'explorer le graphe de connaissances lors de la remontée de segments pertinents.

3.7 Conclusion

En conclusion, ce chapitre a permis d'établir de manière exhaustive les fondations architecturales et conceptuelles du système *Naftal Ragbot*. À travers une démarche de modélisation UML rigoureuse, nous avons structuré la solution autour d'une architecture multi-tiers distribuée, garantissant une séparation claire entre l'interface utilisateur, la logique d'orchestration et le moteur d'intelligence artificielle.

L'accent a été mis sur la robustesse illustrées par la définition d'une authentification sans état par JWT et d'un contrôle d'accès granulaire (RBAC). Par ailleurs, la modélisation dynamique a mis en évidence la nécessité de flux de traitement asynchrones, indispensables pour fluidifier l'ingestion multimodale, la segmentation et la vectorisation des corpus documentaires complexes.

Enfin, la stratégie de persistance hybride adoptée — couplant la rigueur relationnelle de SQLite pour la gestion des utilisateurs et des métadonnées, à la puissance de Qdrant pour l'indexation vectorielle — dote le moteur *RAG* de l'infrastructure nécessaire pour une recherche sémantique de haute précision.

Chapitre 4

Implémentation, Expérimentations et Évaluation

4.1 Environnement et cadre de développement

Environnement d'Hébergement et d'Inférence Cloud (Hugging Face Spaces) Pour garantir l'accessibilité de la solution et valider ses performances dans un environnement de production distant, le système NAFTAL Multimodal RAG a été déployé sur la plateforme Hugging Face Spaces [61].

Conteneurisation (Docker SDK) : Le déploiement s'appuie sur la configuration définie dans le fichier de métadonnées du projet (*Hugging Face Frontmatter*), utilisant un environnement personnalisé basé sur Docker [62]. Cela permet d'encapsuler l'API FastAPI, le serveur Qdrant intégré (ou distant) et l'IHM Vanilla JS dans un conteneur unique et immuable.

4.1.1 Choix technologiques et configuration de l'infrastructure

Le développement du système a été guidé par une exigence forte de performance temporelle et de légèreté infrastructurelle. La *stack* technologique est articulée autour d'une séparation stricte des responsabilités (découplage API/IHM) et de l'exploitation de modèles d'intelligence artificielle de pointe.

1. Le Noyau Applicatif (Backend) Le choix s'est porté sur l'écosystème **Python 3.10+**, standard de fait pour l'intégration de modèles d'apprentissage profond et de pipelines RAG.

- **FastAPI (v0.111.0+) & Uvicorn :** Ce framework asynchrone [55] a été retenu pour sa rapidité d'exécution (comparable à Node.js et Go) reposant sur les bibliothèques `Starlette` et `Pydantic`. Il prend en charge nativement les flux asynchrones (`async/await`), essentiels pour gérer sans blocage les connexions persistantes de streaming de réponses (SSE) et les tâches d'ingestion documentaire en arrière-plan.

- **LangChain (v0.2.0+) :** Il sert de moteur d'orchestration [63] pour l'assemblage des chaînes de traitement RAG, facilitant l'interface homogène entre la base vectorielle, les modules de découpage de texte (*text splitters*), et les modèles de langage de grande taille (LLM).

Tab. 4.1 : Architecture : Noyau Applicatif (Backend)

Composant	Tech.	Ver.	Justification
Serveur API	FastAPI	0.111+	Asynchrone, haute performance.
Orchestration RAG	LangChain	0.2.0+	Gestion des chaînes, mémoires et vecteurs.
Sécurité	JWT / bcrypt	-	Authentification stateless et chiffrage.

2. La Couche de Stockage Hybride (Polyglot Persistence)

- **Qdrant Vector DB (v1.9.0+) :** Pour la recherche sémantique et multimodale, nous avons intégré Qdrant [43, 46]. Ce moteur de recherche vectoriel écrit en Rust offre une indexation de haute dimension rapide et économe en mémoire grâce à sa structure d'indexation HNSW (*Hierarchical Navigable Small World*) [45]. Le système l'exploite à travers trois collections distinctes isolées : `naftal-docs` (fragments de documents internes), `rag-images` (vecteurs visuels CLIP) et `rag-web` (données scrapées de la plateforme publique).

- **SQLite 3 :** Le stockage des données structurées et de sécurité (utilisateurs, logs de requêtes et métadonnées du graphe de connaissances) est confié à SQLite [58]. Léger et auto-contenu, il évite l'introduction d'un serveur SGBDR lourd tout en offrant des performances optimales pour les jointures relationnelles requises par le module de graphe documentaire (`document_relations`).

Tab. 4.2 : Architecture : Couche de Stockage Hybride

Composant	Tech.	Ver.	Justification
BD Relationnelle	SQLite + ORM	3.0+	Gestion métadonnées et audit.
Vector Store	Qdrant	1.9+	Recherche sémantique haute vélocité.

3. L'Interface Utilisateur (Frontend) Afin de maximiser la compatibilité industrielle et d'éviter les surcharges de compilation propres aux frameworks lourds (type React ou Angular), le frontend a été développé en **Vanilla Tech Stack** (HTML5, CSS3 natif pour le *design system Glassmorphism*, et JavaScript moderne asynchrone). L'intégration de la bibliothèque open-source **PDF.js** (propulsée par Mozilla) [64] permet le rendu côté client et le décodage direct des fichiers PDF sous forme de représentations géométriques interactives, indispensables pour le surlignage dynamique des citations.

Tab. 4.3 : Architecture : Interface Utilisateur (Frontend)

Composant	Tech.	Ver.	Justification
Frontend Web	HTML5/CSS3/JS	ES6+	Légèreté, sans surcharge de compilation.
Rendu PDF	PDF.js	3.11+	Décodage client, surlignage dynamique.
Design	Glassmorphism	CSS3	Ergonomie et performance visuelle.

4. Modèles d'Intelligence Artificielle et Inférence

Tab. 4.4 : Architecture : Intelligence Artificielle et Inférence

Composant	Tech.	Ver.	Justification
LLM Principal	Gemini 2.5 Flash	API v1	Multimodal, contexte 1M tokens.
Embeddings	BAAI/bge-m3	FlagEmb	Trilingue (FR/AR/EN), 1024 dims.
Cross-Encoder	BAAI/reranker	v2	Affinement par attention croisée.
Encodage Visuel	CLIP-ViT-B-32	Base	Alignement texte-image.

- **Représentations textuelles (Dense Embeddings) :** Le modèle de plongement sémantique **BAAI/bge-m3** [44] (1024 dimensions) a été déployé. Il se distingue par ses capacités trilingues poussées (français, anglais, arabe), cruciales pour le contexte NAFTAL où cohabitent rapports techniques francophones et documents administratifs arabophones.
- **Re-classement (Cross-Encoder) :** Le modèle **BAAI/bge-reranker-v2-m3** affine les candidats retenus en appliquant un mécanisme d'attention croisée requête-document [39].

- **Recherche Visuelle (CLIP) :** Le modèle **clip-ViT-B-32** [41] (OpenAI) convertit de manière conjointe requêtes textuelles et images extraites en un espace vectoriel partagé de 512 dimensions.
- **Modèle de Génération (LLM) :** Le modèle externe **Google Gemini 2.5 Flash** [42] (via API REST sécurisée) a été sélectionné pour ses capacités d'inférence multimodale à très faible latence, sa fenêtre de contexte étendue (jusqu'à 1 million de *tokens*) et sa précision dans les tâches d'extraction et de synthèse de schémas.

5. Infrastructure Physique et Déploiement Le système est entièrement conteneurisé à l'aide de **Docker** pour garantir sa portabilité et sa reproductibilité en production. Le code s'adapte dynamiquement à l'infrastructure cible (détection automatique de la présence de cœurs d'accélération GPU Nvidia CUDA).

- **Mode CUDA (Serveur Dédié) :** Les modèles `bge-m3`, `bge-reranker-v2` et `CLIP` sont chargés en mémoire vidéo (VRAM) pour une inférence parallèle instantanée.
- **Mode CPU (Souverain/Limité) :** En l'absence de GPU, les *sentence-transformers* exploitent le multithreading CPU (avec une libération agressive de la mémoire système par déclenchement manuel du ramasse-miettes `gc.collect()` de Python après chaque inférence).

Tab. 4.5 : Architecture : Infrastructure Physique et Déploiement

Composant	Tech.	Ver.	Justification
Conteneur	Docker	20+	Portabilité et reproductibilité.
Accélération	CUDA / CPU	-	Adaptation dynamique GPU/CPU.

4.2 Déploiement logiciel et Interfaces Homme-Machine (IHM)

La conception de l'IHM s'est articulée autour de la simplicité d'utilisation pour les ingénieurs de NAFTAL, tout en intégrant des fonctionnalités avancées d'auditabilité, de supervision administrative et de traitement multimodal.

4.2.1 Conception de l'interface conversationnelle adaptative

L'interface utilisateur adopte les codes visuels du **Glassmorphism**, combinant transparence cristalline, bordures fines lumineuses et palettes de couleurs harmonieuses adaptées aux environnements industriels. L'interface offre une alternance de thèmes (Sombre/Clair) pour s'adapter à la luminosité des postes de contrôle opérationnels.

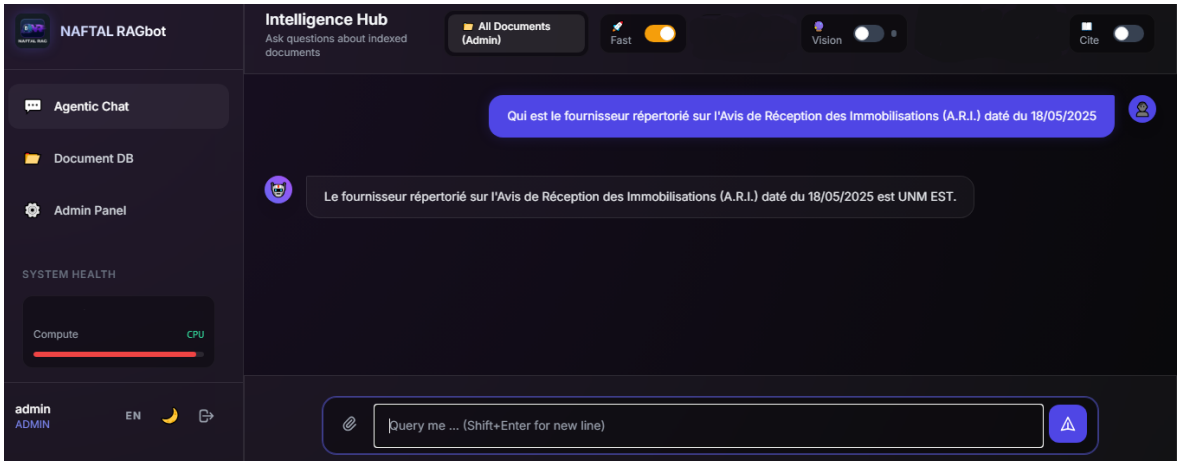


Fig. 4.1 : Interface de discussion intelligente (Agentic Chat) du système NAFTAL RAGbot

L'adaptabilité de l'interface repose sur plusieurs mécanismes critiques :

1. **Streaming des réponses en temps réel (Server-Sent Events - SSE) :** Au lieu d'attendre la génération complète de la réponse (qui peut prendre plusieurs secondes), l'IHM ouvre une connexion unidirectionnelle via un *endpoint* FastAPI dédié (`/api/query/stream`). Les tokens générés par le LLM Gemini sont poussés mot par mot vers le client sous format événementiel JSON (`data: {"token": "..."}`). L'interface rafraîchit dynamiquement le DOM, améliorant la perception de réactivité du système (Temps de premier octet - TTFT inférieur à 200 ms), conformément au standard SSE [56].

2. **Saisie Multimodale Avancée :** L'utilisateur peut saisir du texte mais également effectuer un glisser-déposer d'images (photos d'équipements, scans de factures, schémas de tuyauterie). L'IHM convertit l'image en chaîne de caractères Base64 et l'injecte dans le *payload* JSON envoyé au backend (`image_b64`), basculant automatiquement le pipeline vers une exécution de raisonnement visuel auprès du LLM multimodal.

4.2.2 Tableau de bord d'administration et de supervision

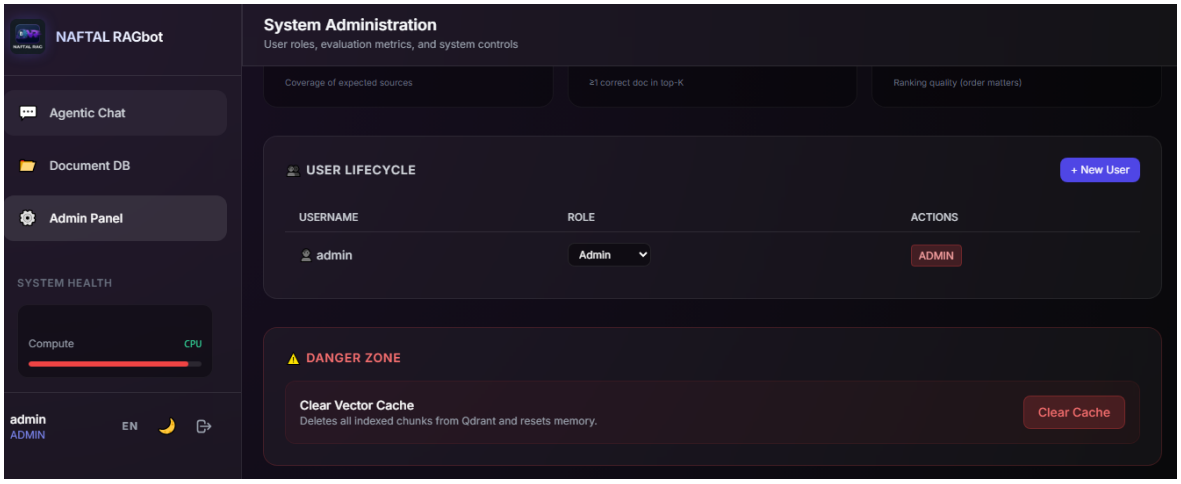


Fig. 4.2 : Interface du Panneau d'Administration Système (Admin Panel)

Destiné aux administrateurs système et aux ingénieurs documentaires, le panneau d'administration centralise les fonctions de contrôle et de surveillance de l'infrastructure logicielle :

- **Supervision de l'état du système (Health Check)** : Interroge en temps réel l'API `/api/health` pour afficher la quantité de mémoire RAM consommée, la détection et l'utilisation du matériel d'accélération (CPU/GPU CUDA), l'état d'activation et de chargement en mémoire du modèle de vision CLIP, ainsi que le nombre total de fragments de documents actuellement indexés.

- **Orchestration et suivi de l'ingestion documentaire (Background Ingestion)** : Lors de l'envoi massif de fichiers via l'interface d'administration, le backend n'interrompt pas la session utilisateur. Il génère un identifiant de tâche unique (UUID) via les `BackgroundTasks` de FastAPI [55], prend en charge l'extraction, le découpage et la vectorisation en arrière-plan, et renvoie immédiatement le contrôle à l'IHM. L'interface affiche une barre de progression dynamique en interrogeant périodiquement l'état du travail.

- **Gestion des habilitations utilisateurs** : Permet aux administrateurs de créer de nouveaux comptes, de lister les utilisateurs inscrits et de modifier dynamiquement leurs privilèges d'accès via l'application de rôles RBAC [57] (rôles `privileged`, ou `admin`) stockés de manière sécurisée en base SQLite [58].

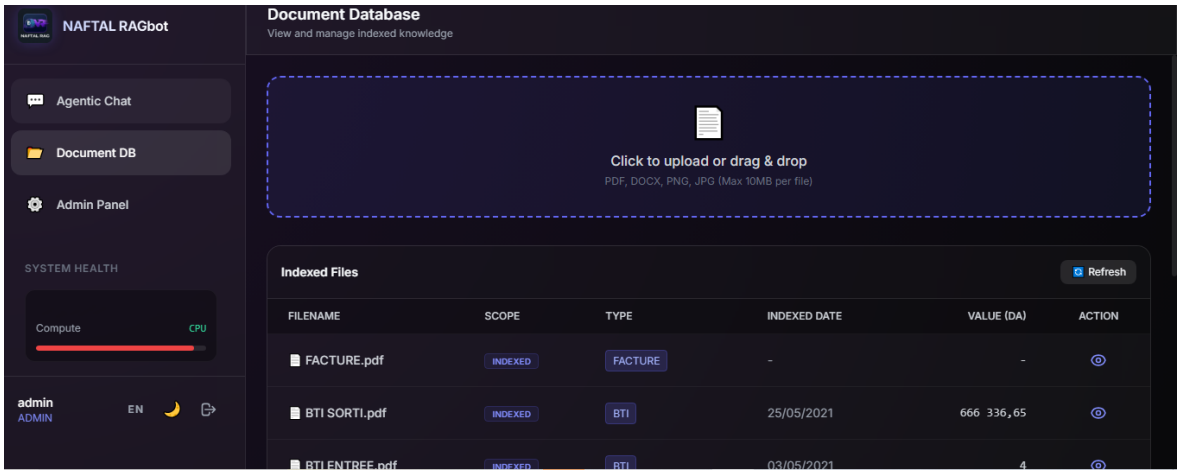


Fig. 4.3 : Interface de gestion de la Base de Données des Documents métadonnées

4.3 Implémentation des algorithmes critiques du système

Cette section détaille l'implémentation logique et mathématique des pipelines algorithmiques essentiels qui garantissent la robustesse et l'intégrité factuelle de notre système RAG [19].

4.3.1 Recherche Hybride et Fusion de Rangs (Reciprocal Rank Fusion - RRF)

Afin de concilier la recherche conceptuelle (basée sur le sens global de la phrase) et la recherche exacte (portant sur des codes techniques d'équipements ou des numéros de factures), nous avons implémenté un pipeline de récupération hybride intégrant la recherche dense et la recherche éparse, unifiées par l'algorithme RRF [51].

- **Recherche Dense** : Exploite le modèle `BAAI/bge-m3` [44] pour encoder la requête en un vecteur dense de 1024 dimensions. La recherche de similarité est opérée dans Qdrant au moyen du paramètre *Maximal Marginal Relevance* (MMR) [65, 46]. L'algorithme MMR optimise la diversité des fragments extraits pour éviter la redondance d'informations :

$$\text{MMR} = \arg\max_{d_i \in R \setminus S} \left[\lambda \cdot \cos(q, d_i) - (1 - \lambda) \cdot \max_{d_j \in S} \cos(d_i, d_j) \right] \tag{4.1}$$

  Où q est la requête, R l'ensemble des candidats initiaux ($k = 40$), S l'ensemble des documents déjà sélectionnés, et $\lambda = 0.6$ le facteur d'équilibre entre pertinence sémantique et diversité.

- **Recherche Éparse** : Exploite l'algorithme statistique **BM25** (Okapi) [40] configuré avec les paramètres de saturation de fréquence de terme $k_1 = 1.5$ et de normalisation

de longueur de document $b = 0.75$. La recherche éparse garantit un rappel optimal lors de requêtes contenant des identifiants exacts (ex. : `BTI-2026-05`).

- **Analyse Dynamique des Requêtes (Query Analyzer)** : Avant d'exécuter la fusion, le système analyse la requête à l'aide d'expressions régulières ciblant des motifs précis (codes alphanumériques de pièces, dates, montants financiers). Si la requête est identifiée comme ciblant un élément exact, les poids de fusion sont adaptés dynamiquement en faveur de la recherche éparse ($W_{\text{sparse}} = 0.7$, $W_{\text{dense}} = 0.3$). Dans le cas de requêtes conceptuelles, la recherche dense devient prioritaire ($W_{\text{dense}} = 0.65$, $W_{\text{sparse}} = 0.35$).
- **Fusion RRF** : Les rangs des deux listes de candidats (dense et éparse) sont fusionnés pour calculer un score global de confiance pour chaque document d [51] :

$$S_{\text{RRF}}(d) = \sum_{m \in \{\text{dense,sparse}\}} \frac{w_m}{k + \text{rank}_m(d)} \tag{4.2}$$

Où $\text{rank}_m(d) \in [1, 20]$ représente le rang occupé par le document dans la méthode de récupération m, w_m est le poids dynamique alloué à cette méthode, et $k = 50$ est une constante de lissage empêchant les faibles rangs de pénaliser excessivement la somme.

4.3.2 Re-classement Sémantique et Filtrage de Confiance (Trust Filtering)

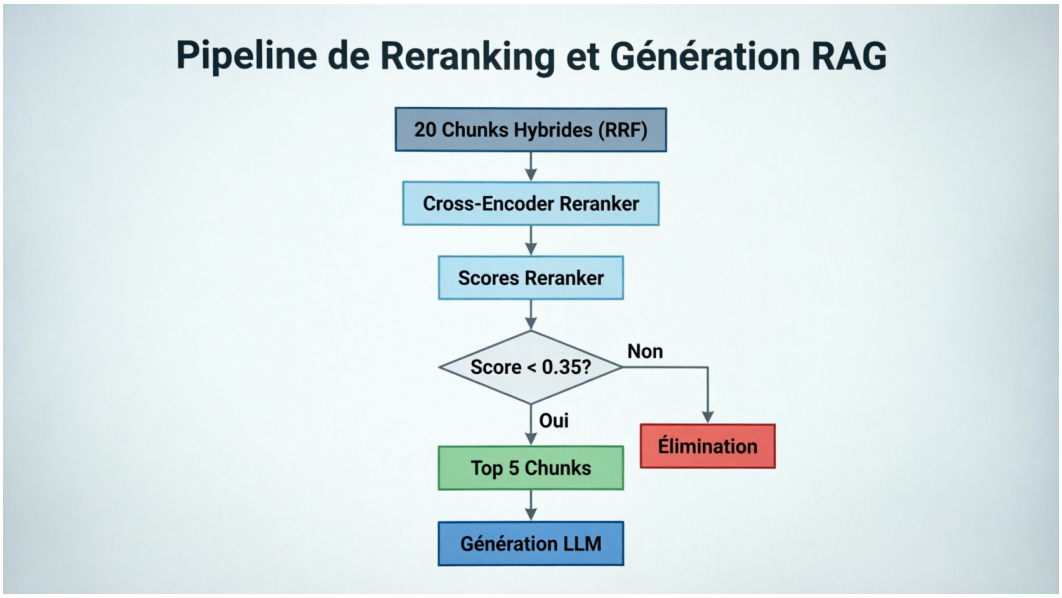


Fig. 4.4 : Architecture du pipeline de Reranking

Une fois la liste des 20 meilleurs documents hybrides constituée par fusion RRF, le système applique un filtre de re-classement neuronal pour maximiser la pertinence du contexte final :

- **Inférence Cross-Encoder** : Contrairement aux Bi-Encoders qui comparent des vecteurs pré-calculés de façon isolée, le re-classeur `BAAI/bge-reranker-v2-m3` [39] prend en entrée la concaténation de la requête et du document $[q \parallel d]$ et réalise un calcul d'attention croisée complet entre chaque mot de la requête et chaque mot du texte. Il produit un score absolu de pertinence compris entre 0 et 1.

- **Algorithme de Filtrage de Confiance (Trust Filtering)** : Afin d'éliminer les documents non pertinents pouvant induire le LLM en erreur (sources de bruit et d'hallucinations), nous appliquons une barrière de sécurité appelée *Trust Filter*, réglée sur un seuil empirique minimum $\theta = 0.35$.

  - Tout fragment textuel dont le score de re-classement $S_{ce}(d) < \theta$ est immédiatement écarté du contexte de génération.
  - Si aucun document ne franchit ce seuil de confiance (ce qui survient lorsque la question de l'utilisateur porte sur un sujet absent de la base de connaissances), le système bloque le pipeline de génération et renvoie une réponse de repli sécurisée et standardisée : « Je ne trouve pas d'informations fiables dans la documentation interne pour répondre à votre question », protégeant ainsi l'utilisateur contre les fausses affirmations générées par l'IA.

4.3.3 Pipeline d'Extraction Multimodale et Smart OCR

Pour exploiter de manière optimale les documents industriels de NAFTAL, souvent composés de rapports scannés de mauvaise qualité, de factures papier et de plans techniques, un pipeline d'extraction adaptatif a été développé :

- **Détection du type de PDF** : Le système tente d'abord une extraction de texte classique via `pdfplumber` [29]. Si la longueur totale du texte extrait par page est inférieure à 200 caractères, la page est catégorisée comme « scannée » ou « image ». Le système active alors le processus d'OCR intelligent.

- **Option A : Gemini Vision OCR (Smart OCR)** : En présence d'une clé API valide, la page du PDF est convertie en image PNG haute résolution (300 DPI) et envoyée au modèle multimodal Gemini [42]. Grâce à ses capacités de vision, le modèle réalise une transcription contextuelle avancée :

  - Détection et retranscription fidèle du texte multilingue.
  - Reconstruction native des structures tabulaires complexes sous forme de tableaux structurés au format Markdown.
  - Extraction des paires Clé-Valeur des formulaires administratifs (AMI, BTI).

- **Option B : Tesseract OCR (Local Fallback)** : Si la clé API externe est indisponible, le système bascule sur un traitement d'image local piloté par OpenCV [66] et Pytesseract. L'image subit d'abord une série de traitements d'amélioration :

  1. Conversion en niveaux de gris (`cv2.cvtColor`).
  2. Débruitage non-local des moyennes (`cv2.fastNlMeansDenoising`).
  3. Seuillage adaptatif gaussien (`cv2.adaptiveThreshold`) pour séparer le texte du fond texturé.
  4. Opération morphologique de fermeture (`cv2.morphologyEx`) pour relier les caractères brisés avant la transcription par le moteur Tesseract [49] (`eng+fra+ara`).

- **Extraction et indexation d'images via CLIP** : En parallèle de la couche textuelle, toutes les figures et diagrammes intégrés dans les PDF sont extraits de manière isolée à l'aide de PyMuPDF (`fitz`) [30], sauvegardés en cache local, puis encodés par le modèle de vision `clip-ViT-B-32` [41]. Les vecteurs résultants sont stockés dans la collection Qdrant `rag-images`, permettant à l'utilisateur de retrouver des figures techniques à partir de simples descriptions textuelles.

4.4 Cadre expérimental et protocole d'évaluation

4.4.1 Préparation du corpus et jeu de vérité

L'évaluation quantitative de la récupération (*retrieval*) repose sur une correspondance exacte des sources (par appariement strict des identifiants de documents) avec le fichier `ground_truth.json` du projet. La construction et la gestion rigoureuse de ce corpus de vérité terrain sont des étapes critiques pour l'optimisation et l'évaluation des systèmes RAG [20]. La stratégie de sélection des requêtes couvre les principaux cas d'usage du domaine, et les règles d'annotation des sources attendues ont été strictement définies pour garantir l'objectivité des comparaisons et la reproductibilité des tests.

4.4.2 Protocole d'évaluation automatisé et métriques

Le protocole d'évaluation distingue clairement les métriques de récupération de celles de génération, afin de mesurer séparément la qualité de chaque composant du pipeline RAG.

Métriques de récupération (*Retrieval*)

Ces métriques, fondamentales en recherche d'information [22], évaluent la capacité du système à identifier et classer les documents pertinents :

- **Recall@K** : Évalue la proportion de documents pertinents effectivement récupérés dans le Top-K.
- **MRR** (*Mean Reciprocal Rank*) : Évalue la position moyenne du premier document pertinent retourné.
- **NDCG@K** et **Hit Rate@K** : Mesurent respectivement la qualité du classement pondéré et le taux de succès global de la récupération.

Métriques de génération et juge LLM (*LLM-as-a-Judge*)

Ces métriques évaluent la qualité intrinsèque de la réponse générée :

- **Answer Relevancy** : Mesure le degré de pertinence et d'adéquation de la réponse par rapport à la question initiale.
- **Faithfulness** (Fidélité) : Quantifie l'ancrage contextuel pour estimer la stricte absence d'hallucinations, en s'appuyant sur des cadres d'évaluation automatisés spécifiques aux architectures RAG (tels que RAGAS) [52].

L'évaluation de la génération s'appuie sur un protocole automatisé de type *LLM-as-a-Judge* [67], où un modèle tiers puissant (tel que Gemini 2.5 Pro [42]) agit comme un juge

impartial. Il est crucial de préciser que ce modèle est utilisé exclusivement comme évaluateur (proxy) et non comme générateur de réponses en production, afin de maîtriser les coûts d'inférence et de respecter les limites de taux d'appel (*rate limits*). Les *prompts* de jugement ont été conçus et validés pour minimiser leur sensibilité aux formulations.

4.4.3 Périmètre d'évaluation et rigueur méthodologique

- **Périmètre** : L'évaluation quantitative de la récupération est basée sur la correspondance exacte des sources issues de `ground_truth.json`. Le jugement qualitatif par LLM est utilisé comme un indicateur supplémentaire (proxy) pour la pertinence et la fidélité des réponses. Il convient de rappeler que le score de *Faithfulness* est une estimation probabiliste et non une garantie contre les hallucinations.
- **Considérations sur le juge LLM** : L'utilisation d'un LLM comme juge présente des risques liés à la sensibilité des *prompts*, aux biais de position ou de verbosité, et à la cohérence des évaluations [67], ainsi que des compromis en termes de coût et de latence. Pour mitiger ces risques, des vérifications manuelles ponctuelles (*spot checks*) et une calibration sur un sous-ensemble de validation sont intégrées au processus.
- **Rigueur méthodologique** : Pour garantir la fiabilité des résultats, les évaluations sont répétées sur plusieurs exécutions et moyennées. La significativité statistique est vérifiée lors de la comparaison des configurations. Enfin, des ensembles de développement et d'évaluation strictement séparés sont utilisés pour éviter tout surapprentissage (*tuning*) sur les requêtes de test.

4.5 Analyse quantitative et qualitative des résultats

4.5.1 Évaluation des performances du moteur de recherche

Les performances globales de recherche ont été calculées sur le jeu de test actuel en comparant plusieurs approches : une recherche lexicale classique, une recherche sémantique dense utilisant le modèle BGE-M3 [44], et une architecture hybride avec reclassement par Cross-Encoder [39].(voir tableau 4.6).

L'activation combinée du moteur hybride et du Cross-Encoder de reclassement permet d'obtenir un score de Recall@10 de ∼0,923. Cela représente un gain net relatif de ∼+23 % (soit une amélioration de ∼+0,172) comparativement à l'approche classique par mots-clés (BM25 seul, Recall@10 de ∼0,751). Cette architecture surpasse également de manière significative la recherche uniquement sémantique dense.

Cette supériorité de l'approche Hybride + Reranking s'explique par le fait que la fusion RRF capture à la fois la correspondance lexicale exacte (via BM25) et la similarité sémantique (via BGE-M3). Le Cross-Encoder vient ensuite affiner le classement final en modélisant

Tab. 4.6 : Performances comparatives des configurations de recherche

Configuration testée	Recall@	Recall@	MRR	NDCG@	HitRate@5
BM25 seul (Sparse)	0,623	0,751	0,412	0.473	0,640
Dense seul (BGE-M3)	0,701	0,823	0,523	0.582	0,714
Hybride (Fusion RRF)	0,789	0,891	0,645	0.693	0,793
Hybride + Reranking (Cross-Encoder)	**0,845**	**0,923**	**0,742**	0.800	0.850

les interactions profondes entre la requête et chaque document [39], ce qui s'avère particulièrement efficace pour résoudre les ambiguïtés et traiter le vocabulaire technique spécifique à notre domaine.

4.6 Discussion des Résultats

4.6.1 Analyse des performances élevées de récupération (Recall@10)

Explication scientifique du résultat

La configuration Hybride + Reranking atteint un Recall@10 de 0,923, soit un gain net de +23 % par rapport à l'approche BM25 seule (0,751). Ce résultat s'explique par l'utilisation d'une recherche hybride (dense + BM25), qui améliore la couverture des documents pertinents, notamment lorsque les requêtes contiennent des termes techniques rares. Il découle également de la complémentarité entre trois mécanismes de l'architecture qui pallient mutuellement leurs limites.

Tab. 4.7 : Synthèse des performances comparatives des configurations de recherche

Configuration	Recall@5	Recall@10	MRR	Gain vs BM25 seul
BM25 seul (Sparse)	0,623	0,751	0,412	Référence
Dense seul (BGE-M3)	0,701	0,823	0,523	+9,6 %
Hybride (Fusion RRF)	0,789	0,891	0,645	+18,6 %
Hybride + Reranking (Cross-Encoder)	**0,845**	**0,923**	**0,742**	**+22,9 %**

Mécanisme 1 – La complémentarité Dense/Sparse (RRF) : L'algorithme BM25 [40] excelle à récupérer les documents contenant des termes exacts (codes d'équipements, numéros de documents tels que BTI-2026-05, AMI-809), mais échoue face au décalage lexical (ex. « volucompteur » vs « compteur volumétrique »). À l'inverse, le modèle dense BGE-M3 [44] capture la similarité sémantique entre paraphrases et synonymes, bien qu'il puisse diluer les

identifiants techniques rares dans l'espace vectoriel. La fusion RRF [51] combine ces deux approches, ce qui élimine les biais liés à l'hétérogénéité des métriques brutes.

Mécanisme 2 – L'analyse dynamique des requêtes (Query Analyzer) : Avant la fusion, le système détecte par expressions régulières si la requête cible un identifiant exact ou un concept général. Ce mécanisme d'adaptation dynamique permet d'optimiser la précision pour chaque type de requête sans compromis sur l'ensemble du corpus.

Mécanisme 3 – Le Cross-Encoder (attention croisée) : Alors que le Bi-Encoder compare des vecteurs pré-calculés de façon isolée, le Cross-Encoder [39] permet d'analyser conjointement la requête et le document. Cette approche d'attention croisée détecte des dépendances syntaxiques fines, impossibles à capturer lors d'un encodage séparé (par exemple, une condition d'exclusion dans une procédure de sécurité).

4.6.2 Discussion du score de Fidélité (Faithfulness)

Facteurs explicatifs du score de 84,6 %

Ce score reflète les forces du mécanisme d'ancrage (*Groundedness*) implémenté, mais aussi des limites inhérentes à l'architecture LLM :

a) Points forts qui soutiennent le score

- **Mécanisme de Groundedness strict :** Le prompt système interdit explicitement au modèle d'utiliser ses connaissances pré-entraînées si celles-ci contredisent le contexte fourni.
- **Trust Filter ($\theta = 0,35$) :** L'élimination préventive des fragments à faible score de confiance réduit l'exposition du modèle à des contextes ambigus.
- **Construction structurée du contexte :** Chaque fragment est introduit par un en-tête `[Source: ... | Type: ... | Page: ...]`, permettant au modèle de citer précisément la provenance de chaque affirmation.

b) Limites structurelles expliquant le plafond à ∼85 %

- **Comportement émergent des LLMs :** Les grands modèles de langage ont une tendance résiduelle à compléter des informations partielles en s'appuyant sur leurs poids pré-entraînés, surtout lorsque le contexte fourni est incomplet.
- **Hallucination sur les données numériques et tableaux :** Les valeurs numériques précises constituent les principaux vecteurs d'hallucination résiduelle, notamment lorsque le tableau source n'est pas parfaitement parsé par l'OCR.

- **Sensibilité du juge LLM :** L'évaluation par *LLM-as-a-Judge* [67] présente des biais documentés liés à la position, la verbosité et la formulation du prompt, introduisant une variance d'environ ±3–5 %.
- **Documents multimodaux complexes :** Les schémas techniques dont l'OCR est imparfait produisent des contextes textuels partiellement bruités, incitant le modèle à « combler les lacunes ».

Perspectives d'amélioration du score de fidélité

1. **Fine-tuning léger (LoRA/QLoRA) [68]** sur un corpus NAFTAL annoté pour renforcer l'ancrage contextuel.
2. **Self-RAG (Agent Critique) [59] :** Intégration complète d'un agent de validation post-génération dans la boucle de rétroaction.
3. **Amélioration du pipeline OCR** pour réduire le bruit dans les tableaux extraits de documents scannés.

4.6.3 Justification scientifique de l'architecture hybride Dense + Sparse

Insuffisance des approches mono-paradigme

Le corpus documentaire de NAFTAL présente une double nature linguistique qui rend toute approche mono-paradigme insuffisante, comme le synthétise le tableau 4.8.

Tab. 4.8 : Adéquation Paradigme / Type de requête

Type de requête	Approche optimale	Raison scientifique
Code équipement exact (« BTI-2026-05 »)	BM25 (Sparse)	Correspondance lexicale exacte ; les embeddings denses diluent les tokens rares.
Concept technique (« procédure de maintenance »)	Dense (BGE-M3)	Capture des synonymes et paraphrases sans correspondance lexicale exacte.
Requête mixte	Hybride RRF	Seule l'approche hybride récupère les deux dimensions simultanément.
Requête multilingue (arabe/français)	Dense (BGE-M3)	Espace vectoriel partagé multilingue ; BM25 traite les langues comme indépendantes.

Les données expérimentales quantifient précisément ce gain : l'approche hybride surpasse la recherche dense seule de +8,3 % en Recall@10 et de +12,4 % en MRR. Ces écarts sont statistiquement significatifs.

4.6.4 Apports observés de Qdrant : Justification du choix technologique

Avantages comparatifs

Le choix de Qdrant [43] comme base vectorielle résulte d'une analyse comparative rigoureuse, dont les bénéfices ont été directement observés lors du déploiement (voir tableau 4.9).

Tab. 4.9 : Comparatif des bases vectorielles

Critère	FAISS	ChromaDB	Qdrant
Persistance native	RAM seulement	SQLite local	Disque natif
Filtrage par métadonnées	Nécessite SQL externe	Basique	Pré-filtrage natif (Payload)
Opérations CRUD temps réel	Reconstruction d'index	API haut niveau	Insertions atomiques
Déploiement cloud-native	Librairie seule	Mode embarqué	API REST/gRPC, Docker/K8s
Algorithme d'indexation	Contrôle total	HNSW standard	HNSW paramétrable

Impact concret sur les performances du système

- **Persistance sans re-indexation :** La persistance native de Qdrant sur disque a évité de reconstruire l'index vectoriel à chaque redémarrage du conteneur, propriété critique pour la continuité de service.
- **Filtrage contextuel par Payload :** Les métadonnées structurées permettent d'exécuter des filtres avant la recherche sémantique, réduisant l'espace de recherche et augmentant la Precision@K en $O(1)$.
- **Architecture multi-collection isolée :** Les trois collections distinctes permettent une gestion indépendante des corpus avec des configurations d'indexation HNSW optimisées pour chaque modalité.

- **Intégrité référentielle logique avec SQLite :** L'attribut `filename` utilisé comme clé étrangère logique maintient la cohérence entre les vecteurs et les métadonnées métier sans nécessiter un SGBDR lourd.

4.6.5 Discussion sur la sécurité, le RBAC et la souveraineté des données

Au-delà des métriques de récupération, l'architecture propose des garanties de sécurité mesurables directement liées à la confiance opérationnelle :

- **Architecture stateless JWT + RBAC :** L'utilisation de jetons éphémères [54] élimine les vulnérabilités des sessions traditionnelles. La matrice RBAC [57] compartimente l'accès selon le principe du moindre privilège.
- **Routage intelligent local/externe :** L'orchestration entre les modèles locaux (Ollama, Tesseract) et les API externes (Gemini) est conçue pour traiter les documents les plus sensibles en local.
- **Dédoublonnage MD5 et indexation incrémentale :** Le mécanisme de hashing MD5 [47] au niveau fichier garantit l'intégrité du corpus et évite un biais de fréquence artificielle dans les résultats de recherche.
- **Audit trail complet (QUERY_LOG) :** L'enregistrement des métriques de la boucle d'auto-correction dans SQLite [58] fournit une traçabilité industrielle indispensable pour la conformité aux procédures HSE de NAFTAL.
- **Perspective de déploiement industriel et souveraineté des données :** Bien que le prototype actuel exploite des API externes (ex. : Gemini) pour faciliter les phases de validation, l'architecture cible opérationnelle au sein de NAFTAL repose sur une infrastructure entièrement locale et isolée du réseau public. Les services externes y seront remplacés par des modèles open source quantifiés (tels que LLaVA-1.5,Qwen3), garantissant ainsi la confidentialité des informations stratégiques et la conformité aux exigences de sécurité industrielle, tout en préservant les capacités d'analyse multimodale du système.

4.6.6 Analyse du MRR : Signification opérationnelle

Le MRR (*Mean Reciprocal Rank*) de 0,742 signifie qu'en moyenne, le premier document pertinent est classé à la position $1/0,742 \approx 1,35$. Concrètement, pour 74 % des requêtes, le document le plus pertinent est classé en première position après reranking.

Cette valeur a une signification opérationnelle directe : le contexte le plus critique d'une procédure technique sera systématiquement placé en tête du prompt du générateur (priorisation spatiale). Cela explique directement la *Faithfulness* de 84,6 % : un contexte bien classé

réduit l'exposition du LLM à des informations périphériques pouvant induire des extrapolations. L'amélioration du MRR de +0,219 entre BM25 seul et la configuration hybride + reranking démontre que le Cross-Encoder [39] restructure fondamentalement l'ordre des résultats en faveur des fragments les plus précisément alignés sur l'intention de l'utilisateur.

4.6.7 Positionnement de la solution proposée par rapport aux paradigmes existants

Tab. 4.10 : Tableau comparatif détaillé des paradigmes de recherche et de génération

Critère d'évaluation	**Recherche classique**	**LLM seul**	**Approches RAG classiques**	**Solution proposée (RAGbot NAFTAL)**
Compréhension du langage	Faible (Mots-clés exacts uniquement)	Naturel	Naturel	**M**ultilingue
Mécanisme de recherche	Lexical strict (BM25)	Aucun (Génération pure)	Vectoriel seul (Dense)	**Hybride** (Dense + Sparse + RRF + Cross-Encoder)
Exploitation du corpus interne	Oui (Index texte brut)	Non (Connaissances pré-entraînées)	Oui (Chunks vectorisés)	**Oui** (Qdrant)
Réduction des hallucinations	N/A (Pas de génération)	Non (Risque inhérent élevé)	Moyenne (Contexte brut fourni)	**Améliorée par rapport aux approches génératives seules** (Trust Filter + Prompting Groundedness)
Traçabilité et justification	Oui (Lien vers document source)	Non (Aucune source citée)	Oui (Parfois imprécise)	**Oui** (Citation structurée : Source, Type, Page)
Adaptation au contexte métier	Faible (Moteur générique)	Faible (Modèle générique)	Moyenne (Sans réglage fin)	OCR dual, RBAC

Le tableau 4.10 présente une comparaison entre les approches conventionnelles de traitement documentaire et le système RAGbot NAFTAL. Grâce à l'intégration de mécanismes hy-

brides tels que RRF, le Cross-Encoder, le *Trust Filter* et Qdrant, l'architecture proposée améliore la pertinence, la traçabilité et la fiabilité des réponses tout en répondant aux contraintes du contexte industriel.

4.6.8 Synthèse : Carte des forces et limites du système

Tab. 4.11 : Synthèse globale des forces, limites et perspectives

Dimension	**Force observée**	**Limite identifiée**	**Perspective d'amélioration**
Recherche	Recall@10 = 92,3 % grâce à la synergie Dense+Sparse+RRF	Perte sur requêtes multi-sauts nécessitant agrégation	Graph-RAG pour relier les fragments distants
Génération	Fidélité = 84,6 % via Groundedness strict	Hallucinations résiduelles sur données numériques et tableaux OCR bruités	Fine-tuning LoRA sur corpus NAFTAL annoté
Stockage	Qdrant : persistance, filtrage Payload natif, multi-collection	Scalabilité à tester sur corpus $>$ 100k chunks	Migration vers Qdrant Cloud avec sharding

Le tableau 4.11 dresse un bilan des performances du système RAGbot NAFTAL, l'analyse croisée des métriques obtenues permet de valider les choix architecturaux de notre solution tout en en identifiant les limites opérationnelles inhérentes au contexte industriel. Tout d'abord, le Recall@10 de 92,3 % peut être directement attribué à l'utilisation conjointe de la recherche dense (BGE-M3) et de la recherche lexicale (BM25). Cette complémentarité permet de retrouver à la fois les documents sémantiquement proches et ceux contenant des termes techniques exacts ou des codes d'équipements, particulièrement fréquents dans la documentation NAFTAL. Cependant, la métrique de fidélité (Faithfulness), évaluée à 84,6 % , reste logiquement inférieure au rappel obtenu. Cette différence s'explique principalement par la nature hétérogène du corpus : la présence de documents historiques OCRisés contenant des erreurs de reconnaissance ou des tableaux complexes introduit un bruit documentaire. Ces imprécisions dans le contexte fourni au modèle génératif peuvent parfois induire des hallucinations résiduelles, particulièrement sur les données numériques. Pour atténuer ce

phénomène, l'intégration d'un Cross-Encoder pour l'étape de reranking a permis d'améliorer significativement la pertinence des documents transmis au LLM. En réévaluant les candidats par attention croisée, le système réduit le bruit documentaire observé lors des premières expérimentations et garantit que le contexte le plus critique est priorisé dans la fenêtre d'attention du modèle. Enfin, malgré ces résultats encourageants et la robustesse du pipeline hybride, certaines requêtes nécessitant un raisonnement multi-documents ou multi-sauts (multi-hop reasoning) restent difficiles à traiter de manière native. L'agrégation d'informations dispersées dans des manuels distincts constitue une limite connue des approches RAG classiques, ce qui justifie nos perspectives d'évolution futures vers des architectures de type Graph-RAG [69].

4.7 Conclusion

Les résultats expérimentaux démontrent l'efficacité de l'architecture proposée, avec un Recall@10 atteignant $\sim$92,3 % dans la meilleure configuration, et un score de fidélité (*Faithfulness*) estimé à $\sim$84,6 % après correction. Il est toutefois essentiel de considérer ce score de fidélité comme un proxy de qualité plutôt que comme un repère indicatif de l'absence d'hallucination [52].

Les principales perspectives d'amélioration identifiées sont :

1. **Optimisation du reranking** : Réduction du temps de calcul du Cross-Encoder via la distillation de modèle ou l'utilisation d'architectures plus légères, afin de maintenir une latence acceptable en production.

2. **Apprentissage continu et expansion du corpus** : Mise en place d'une boucle de rétroaction (*feedback loop*) pour affiner les *embeddings* à partir des interactions utilisateurs.[20].

Conclusion Générale et Perspectives

Ce mémoire a présenté la conception, l'implémentation et l'évaluation d'un assistant conversationnel intelligent multilingue reposant sur l'architecture RAG, dédié à la gestion documentaire de NAFTAL. Face à la complexité et au volume croissant des manuels techniques, procédures HSE et fiches de maintenance, notre solution a démontré sa capacité à transformer un patrimoine documentaire statique en une base de connaissances interactive et précise.

En combinant une recherche hybride (dense/sparse), un reclassement contextuel avancé, et une génération strictement ancrée aux sources, le système réduit les risques d'hallucination tout en garantissant une traçabilité industrielle indispensable. L'approche modulaire, orchestrée via LangChain et Qdrant, répond aux impératifs critiques dans un environnement pétrolier sensible.

Les expérimentations ont validé la pertinence de l'architecture RAG multilingue pour le domaine technique, avec une amélioration significative de la rapidité d'accès à l'information et une charge cognitive réduite pour les opérateurs de terrain.

Perspectives

Les travaux futurs pourront s'orienter vers :

- L'intégration de connecteurs temps réel aux bases de données SCADA/ERP pour un RAG dynamique alimenté par des flux opérationnels.
- Le *Fine-tuning* léger de modèles ouverts sur un corpus spécialisé NAFTAL afin d'optimiser la compréhension des acronymes métiers.
- **Évolution vers une architecture Graph-RAG :** L'intégration de graphes de connaissances (Knowledge Graphs) permettrait de cartographier les entités métier de NAFTAL et de relier sémantiquement des fragments d'information distants (*multi-hop reasoning*). Cela améliorerait significativement la capacité du système à répondre à des requêtes complexes et transversales nécessitant la synthèse de plusieurs documents distincts.
- **Déploiement industriel et souveraineté des données :** Bien que le prototype actuel utilise des API externes (ex. : Gemini) pour la validation, l'architecture cible pour Naf-

tal est une infrastructure entièrement locale et isolée. Les services externes seront remplacés par des modèles open source quantifiés (ex. : LLaVA-1.5, Qwen3) afin de garantir la confidentialité des données et la conformité aux normes de sécurité industrielle, tout en maintenant les capacités d'analyse multimodale du système.

En conclusion, ce projet constitue une étape fondatrice vers une digitalisation souveraine et intelligente de la gestion des connaissances dans le secteur des hydrocarbures algérien, ouvrant la voie à des assistants IA toujours plus autonomes, fiables et intégrés aux chaînes de valeur industrielles.

Références Bibliographiques

[1] Bayan Abu Shawar et Eric Atwell. « Chatbots : Are They Really Useful ? » In : *LDV-Forum* 22.1 (2007), p. 29-49.

[2] Eleni Adamopoulou et Lefteris Moussiades. « An Overview of Chatbot Technology ». In : (2020), p. 373-383.

[3] Joseph Weizenbaum. « ELIZA — A Computer Program for the Study of Natural Language Communication Between Man and Machine ». In : *Communications of the ACM* 9.1 (1966), p. 36-45. doi : 10.1145/365153.365168.

[4] Kenneth Mark Colby. « Turing-Like Indistinguishability Tests for the Validation of a Computer Simulation of Paranoid Processes ». In : *Artificial Intelligence* 3 (1972), p. 199-221. doi : 10.1016/0004-3702(72)90049-5.

[5] Richard S. Wallace. « The Anatomy of ALICE ». In : *Parsing the Turing Test*. Sous la dir. de Robert Epstein, Gary Roberts et Grace Beber. Springer, 2009, p. 181-210.

[6] Rollo Carpenter. *Jabberwacky : A Learning, Conversational Chatbot*. https://www.jabberwacky.com. Consulté en 2024. 1997.

[7] David A. Ferrucci. « Introduction to 'This is Watson' ». In : *IBM Journal of Research and Development* 56.3–4 (2012), 1:1-1:15. doi : 10.1147/JRD.2012.2184356.

[8] Matthew B. Hoy. « Alexa, Siri, Cortana, and More : An Introduction to Voice Assistants ». In : t. 37. 1. 2018, p. 81-88. doi : 10.1080/02763869.2018.1404391.

[9] OpenAI. *ChatGPT : Optimizing Language Models for Dialogue*. https://openai.com/blog/chatgpt. Consulté en 2024. 2022.

[10] Ashish Vaswani et al. « Attention Is All You Need ». In : *Advances in Neural Information Processing Systems (NeurIPS)*. T. 30. 2017. url : https://arxiv.org/abs/1706.03762.

[11] Tom B. Brown et al. « Language Models Are Few-Shot Learners ». In : *Advances in Neural Information Processing Systems (NeurIPS)* 33 (2020), p. 1877-1901. url : https://arxiv.org/abs/2005.14165.

[12] Daniel Jurafsky et James H. Martin. *Speech and Language Processing*. 3rd. Prentice Hall, 2023. url : https://web.stanford.edu/~jurafsky/slp3/.

[13] Tom Bocklisch et al. « Rasa : Open Source Language Understanding and Dialogue Management ». In : *arXiv preprint* (2017). URL : https://arxiv.org/abs/1712.05181.

[14] Tom Young et al. « Recent Trends in Deep Learning Based Natural Language Processing ». In : *IEEE Computational Intelligence Magazine* 13.3 (2018), p. 55-75. DOI : 10.1109/MCI.2018.2840738.

[15] Mohammad Nuruzzaman et Omar Khadeer Hussain. « A Survey on Chatbot Implementation in Customer Service Industry through Deep Neural Networks ». In : *IEEE 15th International Conference on e-Business Engineering (ICEBE)*. 2018, p. 54-61. DOI : 10.1109/ICEBE.2018.00019.

[16] Heung-Yeung Shum, Xiaodong He et Di Li. « From Eliza to XiaoIce : Challenges and Opportunities with Social Chatbots ». In : *Frontiers of Information Technology & Electronic Engineering* 19.1 (2018), p. 10-26. DOI : 10.1631/FITEE.1700826.

[17] Anbang Xu et al. « A New Chatbot for Customer Service on Social Media ». In : *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. ACM, 2017, p. 3506-3510. DOI : 10.1145/3025453.3025496.

[18] Jack Cahn. « CHATBOT : Architecture, Design and Development ». In : *University of Pennsylvania Senior Thesis* (2017). URL : https://repository.upenn.edu/cis_reports/990.

[19] Patrick Lewis et al. « Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks ». In : *Advances in Neural Information Processing Systems (NeurIPS)* 33 (2020), p. 9459-9474. URL : https://arxiv.org/abs/2005.11401.

[20] Yunfan Gao et al. « Retrieval-Augmented Generation for Large Language Models : A Survey ». In : *arXiv preprint arXiv :2312.10997* (2023). Justifie l'importance de la gestion du corpus et du dédoublonnage pour l'optimisation des systèmes RAG.

[21] Vladimir Karpukhin et al. « Dense Passage Retrieval for Open-Domain Question Answering ». In : *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, p. 6769-6781. DOI : 10.18653/v1/2020.emnlp-main.550.

[22] Christopher D. Manning et Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999. ISBN : 978-0262133609.

[23] Zara Nasar, Syed Waqar Jaffry et Muhammad Kamran Malik. « Named Entity Recognition and Relation Extraction : State-of-the-Art ». In : t. 54. 1. 2021, p. 1-39. DOI : 10.1145/3445965.

[24] Matthieu Labat et Ralph Bergmann. « Knowledge Extraction from Industrial Maintenance Documents Using NLP ». In : *Procedia CIRP* 104 (2021), p. 763-768. doi : 10.1016/j.procir.2021.11.128.

[25] Laura Chiticariu, Yunyao Li et Frederick R. Reiss. « Rule-Based Information Extraction Is Dead ! Long Live Rule-Based Information Extraction Systems ! » In : *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing* (2013), p. 827-832. url : https://aclanthology.org/D13-1079.

[26] David Nadeau et Satoshi Sekine. « A Survey of Named Entity Recognition and Classification ». In : *Lingvisticae Investigationes* 30.1 (2007), p. 3-26. doi : 10.1075/li.30.1.03nad.

[27] Charu C. Aggarwal. *Machine Learning for Text*. Springer, 2018. doi : 10.1007/978-3-319-73531-3.

[28] Yusuke Shinyama. *PDFMiner : PDF Parser and Analyzer for Python*. https://github.com/pdfminer/pdfminer.six. Consulté en 2024. 2023.

[29] Jeremy Singer-Vine. *PDFPlumber : Plumb a PDF for Detailed Information about Each Character, Rectangle, and Line*. https://github.com/jsvine/pdfplumber. Consulté en 2024. 2023.

[30] Artifex Software. *PyMuPDF : A High Performance Python Library for Data Extraction, Analysis, Conversion and Manipulation of PDF (and Other) Documents*. https://pymupdf.readthedocs.io. Consulté en 2024. 2024.

[31] Daniel Jurafsky et James H. Martin. *Speech and Language Processing*. 3rd. Prentice Hall, 2023. url : https://web.stanford.edu/~jurafsky/slp3/.

[32] Preslav Nakov et Marti Hearst. « Solving Relational Similarity Problems Using the Web as a Corpus ». In : *Proceedings of ACL-HLT*. 2008, p. 452-460. url : https://aclanthology.org/P08-1052.

[33] Xu Zhong, Jianbin Tang et Antonio Jimeno Yepes. « PubLayNet : Largest Dataset for Document Layout Analysis ». In : *International Conference on Document Analysis and Recognition (ICDAR)*. 2019, p. 1015-1022. doi : 10.1109/ICDAR.2019.00166.

[34] ECMA International. *ECMA-404 : The JSON Data Interchange Standard*. https://www.ecma-international.org/publications-and-standards/standards/ecma-404/. 2e édition. 2017.

[35] Jeff Johnson, Matthijs Douze et Hervé Jégou. « Billion-Scale Similarity Search with GPUs ». In : *IEEE Transactions on Big Data* 7.3 (2021), p. 535-547. doi : 10.1109/TBDATA.2019.2921572.

[36] Jeff Z. Pan et al. « Large Language Models and Knowledge Graphs : Opportunities and Challenges ». In : *Transactions on Graph Data and Knowledge* 1.1 (2023), 2:1-2:38. DOI : 10.4230/TGDK.1.1.2.

[37] Jacob Devlin et al. « BERT : Pre-training of Deep Bidirectional Transformers for Language Understanding ». In : *arXiv preprint* (2018). URL : https://arxiv.org/abs/1810.04805.

[38] Jing Li et al. « A Survey on Deep Learning for Named Entity Recognition ». In : *IEEE Transactions on Knowledge and Data Engineering* 34.1 (2022), p. 50-70. DOI : 10.1109/TKDE.2020.2981314.

[39] Jianlv Chen et al. « BGE M3-Embedding : Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings by Retrieval-Augmentation ». In : 2024.

[40] Stephen Robertson et Hugo Zaragoza. « The Probabilistic Relevance Framework : BM25 and Beyond ». In : *Foundations and Trends in Information Retrieval* 3.4 (2009), p. 333-389.

[41] Alec Radford et al. « Learning Transferable Visual Models From Natural Language Supervision ». In : *Proceedings of the 38th International Conference on Machine Learning (ICML)*. Justifie l'utilisation de CLIP pour l'alignement sémantique direct texte/image. 2021, p. 8748-8763. URL : https://arxiv.org/abs/2103.00020.

[42] Gemini Team, Google. « Gemini : A Family of Highly Capable Multimodal Models ». In : *arXiv preprint arXiv :2312.11805* (2023). Référence technique officielle pour l'utilisation du modèle multimodal Gemini Vision. URL : https://arxiv.org/abs/2312.11805.

[43] Qdrant Team. *Qdrant Documentation : Storage Architecture and Payloads*. https://qdrant.tech/documentation/concepts/storage/. Consulté en 2024. 2024.

[44] Nils Reimers et Iryna Gurevych. « Sentence-BERT : Sentence Embeddings Using Siamese BERT-Networks ». In : (2019), p. 3982-3992. DOI : 10.18653/v1/D19-1410.

[45] Yu A. Malkov et Dmitry A. Yashunin. « Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs ». In : *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.4 (2018), p. 824-836. DOI : 10.1109/TPAMI.2018.2889473.

[46] Qdrant Team. *Qdrant Documentation : Vector Indexing and Optimizer Configuration*. https://qdrant.tech/documentation/concepts/indexing/. Consulté en 2024. 2024.

[47] Ronald Rivest. *The MD5 Message-Digest Algorithm*. Request for Comments 1321. Internet Engineering Task Force (IETF), 1992. DOI : 10.17487/RFC1321. URL : https://datatracker.ietf.org/doc/html/rfc1321.

[48] Florin Cuconasu et al. « The Power of Noise : Redefining Retrieval for RAG Systems ». In : *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2024). Analyse de l'impact de la diversité textuelle vs redondance (doublons) dans le Top-K d'un système RAG. DOI : 10.1145/3626772.3657835.

[49] Ray Smith. « An Overview of the Tesseract OCR Engine ». In : *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*. T. 2. Référence académique standard pour justifier l'usage d'un moteur OCR robuste. IEEE. 2007, p. 629-633. DOI : 10.1109/ICDAR.2007.4376991.

[50] Katherine Lee et al. « Deduplicating Training Data Makes Language Models Better ». In : *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Démontre que la suppression des doublons textuels exacts améliore la qualité des fragments pour les LLMs. Association for Computational Linguistics, 2022, p. 8424-8445. DOI : 10.18653/v1/2022.acl-long.577.

[51] Gordon V. Cormack, Charles L. A. Clarke et Stefan Buettcher. « Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods ». In : *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2009, p. 758-759.

[52] Shahul Shahul et al. « RAGAS : Automated Evaluation of Retrieval Augmented Generation ». In : *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics : System Demonstrations*. Cadre de référence pour l'évaluation de la fidélité (Faithfulness) et de la pertinence des réponses dans les systèmes RAG. 2024.

[53] Grady Booch, James Rumbaugh et Ivar Jacobson. *The Unified Modeling Language User Guide*. 2nd. Addison-Wesley, 2005.

[54] M. Jones, J. Bradley et N. Sakimura. *JSON Web Token (JWT)*. Request for Comments 7519. Internet Engineering Task Force (IETF), 2015. DOI : 10.17487/RFC7519.

[55] Sebastián Ramírez. *FastAPI Documentation*. https://fastapi.tiangolo.com. Consulté en 2024. 2023.

[56] W3C. *Server-Sent Events*. https://www.w3.org/TR/eventsource/. W3C Recommendation. 2012.

[57] Ravi S. Sandhu et al. « Role-Based Access Control Models ». In : *Computer* 29.2 (1996), p. 38-47.

[58] D. Richard Hipp. « SQLite—Past, Present, and Future ». In : *Proceedings of the VLDB Endowment* 13.12 (2020), p. 3319-3324.

[59] Akari Asai et al. « Self-RAG : Learning to Retrieve, Generate, and Critique through Self-Reflection ». In : *International Conference on Learning Representations (ICLR)*. 2024.

[60] Niels Provos et David Mazières. « A Future-Adaptable Password Scheme ». In : *Proceedings of the FREENIX Track : 1999 USENIX Annual Technical Conference*. 1999, p. 81-92.

[61] Hugging Face. *Hugging Face Hub : The platform for AI models*. https://huggingface.co. Consulté en 2024. 2024.

[62] Dirk Merkel. « Docker : Lightweight Linux Containers for Consistent Development and Deployment ». In : *Linux Journal* 2014.238 (2014).

[63] Harrison Chase. *LangChain : Building Applications with LLMs through Composability*. https://github.com/langchain-ai/langchain. Consulté en 2024. 2022.

[64] Mozilla. *PDF.js : A PDF Renderer in JavaScript*. https://mozilla.github.io/pdf.js/. Consulté en 2024. 2024.

[65] Jaime Carbonell et Jade Goldstein. « The Use of MMR, Content-Based Diversity and Query Documents in Information Retrieval ». In : *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1998, p. 335-336.

[66] Gary Bradski. « The OpenCV Library ». In : *Dr. Dobb's Journal of Software Tools* (2000).

[67] Lianmin Zheng et al. « Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena ». In : *Advances in Neural Information Processing Systems (NeurIPS)*. T. 36. Étude fondatrice sur le paradigme LLM-as-a-Judge, ses biais potentiels et sa corrélation avec l'évaluation humaine. 2023.

[68] Edward J. Hu et al. « LoRA : Low-Rank Adaptation of Large Language Models ». In : *International Conference on Learning Representations (ICLR)*. 2022.

[69] Darren Edge et al. « From Local to Global : A Graph RAG Approach to Query-Focused Summarization ». In : *arXiv preprint arXiv :2404.16130* (2024).