ENSIAS

المديرية العامة للضرائب
ⵜⴰⴷⵓⵔⵉⵜ ⵜⴰⵎⴰⵡⴰⵙⵜ ⵏ ⵜⵣⵔⴼⵜ
Direction Générale des Impôts

National School of Computer Science and Systems Analysis - Rabat

Second-year internship report

Advanced Multilingual Chatbot Using Retrieval-Augmented Generation (RAG) for Moroccan Fiscal Documents

Presented By :
MICHAL Yassine
BOUTANFIT Salma

Supervised By :
Mr. El MAHBOUBI Oussama
Mr. MOUDNI Ahmed

Jury :
Pr. BELLABDAOUI Adil
Pr. BENBRAHIM Houda

Academic year 2025-2026

Acknowledgements

Praise be to Allah alone. May His blessings be upon our Lord and Master Muhammad and upon his family .

At the conclusion of this internship, we would like to express our deepest gratitude to all those who contributed to the success of this work.

First and foremost, we sincerely thank the **Moroccan Tax Administration (DGI)**, and in particular the **Risk Analysis and Programming Service (SARP)**, for welcoming us and giving us the opportunity to work on such an innovative and meaningful project.

We are especially grateful to our supervisors at the DGI, **Mr. Oussama El Mahboubi** and **Mr. MOUDNI Ahmed**, for their continuous guidance, availability, and constructive advice throughout the project. their expertise and support have been instrumental in the successful completion of our work.

We would also like to acknowledge the entire SARP team for their collaboration, insightful discussions, and the stimulating professional environment that allowed us to learn and grow during this internship.

We extend our sincere thanks to the members of the **jury**, **Mrs. Houda Benbrahim** and **Mr. Adil Bellabdaoui**, whose evaluation, feedback, and critical insights have been a valuable contribution to our academic journey.

Finally, we wish to express our heartfelt gratitude to our families and friends, whose encouragement, patience, and unwavering support have been invaluable throughout this experience.

List of abbreviations

Abbreviation	**Meaning**
AI	Artificial intelligence
API	Application Programming Interface
ASGI	Asynchronous Server Gateway Interface
ATAF	African Tax Administration Forum
BLEU	Bilingual Evaluation Understudy (text evaluation metric)
CGI	General Tax Code (*Code Général des Impôts*)
DGI	Direction Générale Des Impôts
DH	Moroccan dirham
IR	Individual income tax (*Impôt sur le Revenu*)
IS	Corporate income tax (*Impôt sur les Sociétés*)
JSON	JavaScript Object Notation
LF	Finance Law (*Loi de Finances*)
LLM	Large Language Model
MRR	Mean Reciprocal Rank (search metric)
NC	Circular note (*Note Circulaire*)
OCR	Optical Character Recognition
OECD	Organisation for Economic Cooperation and Development
PDF	Portable Document Format
RAG	Retrieval-Augmented Generation
SARP	Risk Analysis and Programming Service
SIMPL	SIMPL portal for electronic tax filing
TVA	Value added tax (*Taxe sur la Valeur Ajoutée*)
UI	User Interface

Abstract

This report presents the development of an intelligent assistant based on Retrieval-Augmented Generation (RAG) for the Moroccan Tax Administration (DGI). The project addresses the major challenges posed by the size, heterogeneity, and multilingual nature of Moroccan fiscal documentation, including the General Tax Code and annual circular notes published between 2011 and 2025. The proposed system integrates document preprocessing, OCR for scanned files, hierarchical chunking adapted to variable structures, semantic search with vector embeddings, and controlled text generation through OpenRouter APIs. The solution enables DGI staff to query large volumes of fiscal texts in natural language and obtain accurate, cited answers grounded in official documents. By combining semantic retrieval with generative AI, the system significantly reduces search time, enhances information accessibility, and improves legal reliability. The report details the technical architecture, implementation challenges, and design decisions, while highlighting the system's potential to modernize knowledge management DGI.

Résumé

Ce rapport présente le développement d'un assistant intelligent basé sur la technique du Retrieval-Augmented Generation (RAG) pour la Direction Générale des Impôts (DGI) du Maroc. Le projet répond aux défis liés au volume, à l'hétérogénéité et au multilinguisme de la documentation fiscale marocaine, comprenant le Code Général des Impôts et les notes circulaires annuelles publiées entre 2011 et 2025. La solution conçue intègre un pipeline complet allant de la numérisation des documents (OCR) et du découpage hiérarchique (chunking) à l'indexation vectorielle (ChromaDB) et à la génération contrôlée de réponses via les API OpenRouter. L'assistant permet aux agents de la DGI d'interroger la documentation en langage naturel et d'obtenir des réponses précises, accompagnées de références vérifiables. Cette approche réduit considérablement le temps de recherche, améliore l'accessibilité de l'information et garantit une meilleure fiabilité juridique. Le rapport décrit en détail l'architecture technique, les choix méthodologiques et les défis rencontrés, tout en mettant en avant le potentiel de ce système pour moderniser la gestion des connaissances dans la DGI.

Introduction

The digital transformation of public administrations is fundamentally changing the way governments manage, process, and deliver information. In this context, the **Moroccan Tax Administration (DGI)** of Morocco plays a central role in ensuring the proper implementation of fiscal laws and the collection of revenues essential to national development.

Every year, the DGI publishesa large volume of legal and regulatory documentation, including the *General Tax Code* (CGI) and annual circular notes. These documents are essential references for both tax officials and practitioners. However, their considerable volume, heterogeneous structure, and frequent updates make them particularly difficult to access and exploit in daily operations. Employees often spend valuable time manually searching for specific information across multiple sources, which hinders efficiency and increases the risk of errors in the interpretation and application of fiscal law.

Recent advances in **artificial intelligence** and, in particular, the emergence of **Large Language Models (LLMs)** combined with **Retrieval-Augmented Generation (RAG)** techniques offer new opportunities for improving access to complex legal and fiscal knowledge. By leveraging these technologies, it becomes possible to create intelligent assistants capable of understanding natural language queries, retrieving relevant excerpts from large corpora of documents, and generating reliable, cited answers grounded in authoritative sources.

The objective of this internship project is to design and implement such an intelligent assistant for the Moroccan DGI. More specifically, the project focuses on developing a RAG-based chatbot capable of processing the CGI and annual circular notes, handling multilingual content (French and Arabic), and ensuring transparency through systematic citation of sources.

This report is structured as follows:

- **Chapter 1** introduces the institutional context of the DGI, the challenges posed by fiscal documentation, and the objectives of the project.
- **Chapter 2** presents the state of the art on Retrieval-Augmented Generation and its relevance to legal and fiscal contexts.
- **Chapter 3** details the design and implementation of the proposed system, from document preprocessing to the architecture of the RAG pipeline.
- **Chapter 4** presents the practical realization of the system, detailing the technologies, implementation process, and reports a quantitative evaluation of the RAG pipeline.
- Finally, the general conclusion summarizes the main achievements, discusses the system's real-world impact, and outlines future work directions.

Through this project, we aim to demonstrate how advanced AI methods can directly support public administration by making fiscal knowledge more accessible, reliable, and efficient.

Contents

List of Figures

Chapter 1

General Context of the Project

This chapter provides a comprehensive overview of the internship's setting. It begins with a presentation of the host organization, Moroccan Tax Administration (DGI) of Morocco, detailing its missions and structure. Subsequently, it delves into the specifics of the internship project, outlining its objectives and the challenges it addresses. Finally, it describes the project management methodology adopted to ensure its successful completion.

1.1 Presentation of the Host Organization

Moroccan Tax Administration (DGI) is a key institution within the Moroccan Ministry of Economy and Finance. It is entrusted with the administration of the national tax system, playing a vital role in securing the state's revenue, which is essential for funding public services and infrastructure. [1]

1.1.1 The Host Company

Moroccan Tax Administration (DGI) is a central directorate within the Moroccan **Ministry of Economy and Finance**. Its mandate is to manage the national tax system, ensure compliance with fiscal legislation, and collect revenues that fund public services and infrastructure.

Figure 1.1 illustrates the organizational structure of the Ministry of Economy and Finance, showing the DGI as one of its key directorates alongside the Treasury, the Budget Directorate, and the Customs and Excise Administration.

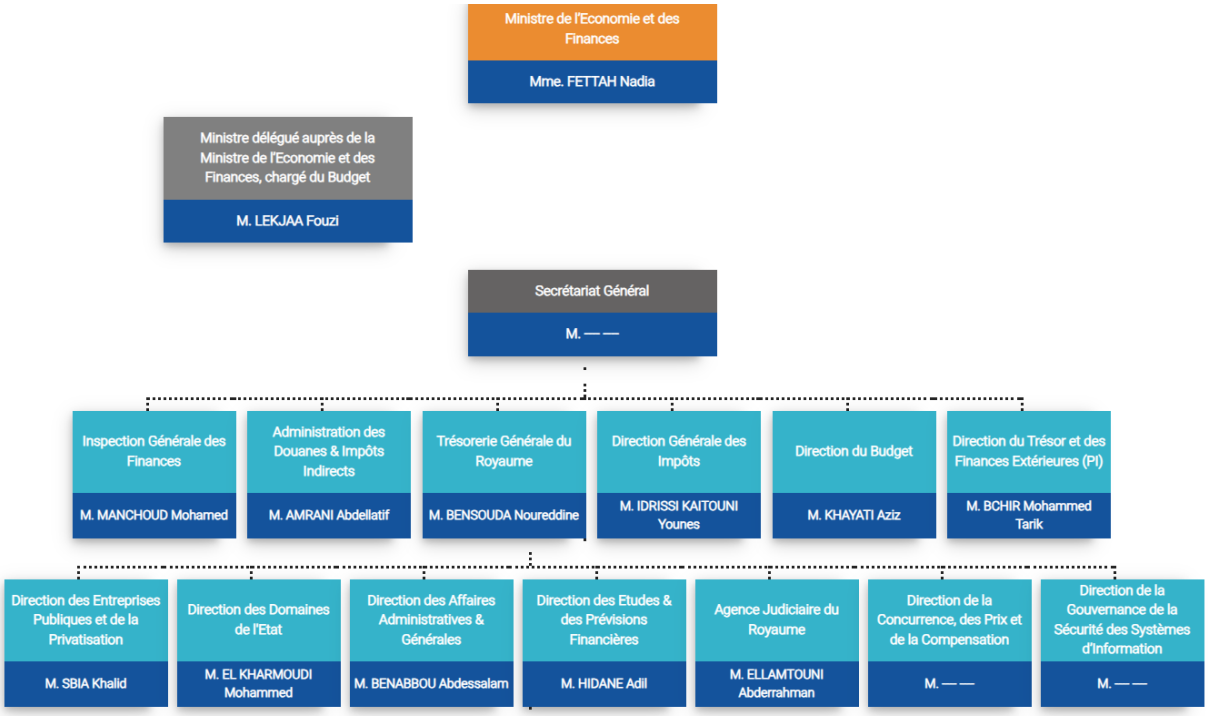


Figure 1.1: Organizational chart of the Ministry of Economy and Finance, highlighting the position of the DGI.

Within the DGI, the organizational structure is composed of several directorates and divisions covering legislation, audit, taxpayer services, risk management, and information systems. Figure 1.2 provides a detailed view of the internal structure of the DGI.

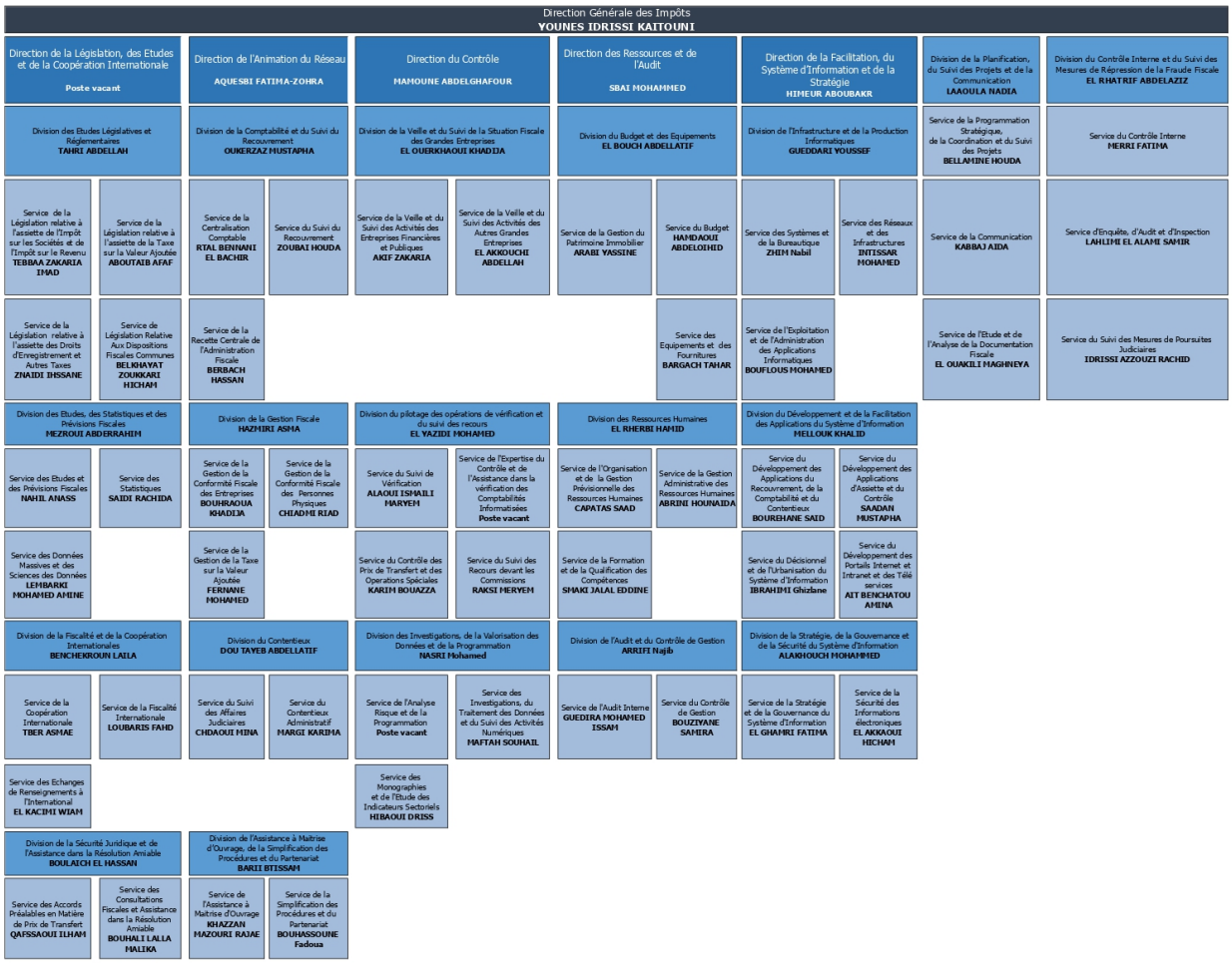


Figure 1.2: Detailed organizational structure of the DGI.

More specifically, our internship was carried out within the **Division of Investigations, Data Valorization and Programming**, and more precisely in the **Service d'Analyse des Risques et de Programmation (SARP)**. This service is responsible for analyzing fiscal risks, developing predictive programming strategies, and supporting the DGI in data-driven decision-making. The strategic role of the SARP provides a direct link between risk analysis, operational efficiency, and the modernization of tax administration through innovative tools such as artificial intelligence.

1.1.2 Activities and Organizations

The DGI's activities are multifaceted and can be categorized as follows:

- **Tax Assessment and Collection:** This is the core function of the DGI, involving the determination of the tax base, the calculation of the amount of tax due, and the collection of these taxes. The organization processes thousands of tax returns annually from individual taxpayers, corporations, and other legal entities.
- **Tax Audit and Control:** The DGI is responsible for verifying the accuracy of tax returns and combating tax fraud and evasion. This is an increasingly important

mission, with the DGI leveraging data analytics and artificial intelligence to improve the effectiveness of its controls and identify potential compliance issues.

- **Taxpayer Services:** The DGI provides information and assistance to taxpayers to help them comply with their tax obligations. This includes online services through the SIMPL portal, which allows for the electronic filing of tax returns and payment of taxes, significantly modernizing the taxpayer experience.
- **Legal and Legislative Affairs:** The DGI is actively involved in the development of tax legislation and regulations. It also manages tax-related disputes and litigation, ensuring proper interpretation and application of fiscal laws.

The DGI operates through a hierarchical structure with central directorates in Rabat and an extensive network of regional and local offices throughout the country. This organizational model enables the DGI to maintain close proximity to taxpayers while ensuring consistent application of tax policies at the national level. The organization employs thousands of civil servants across various specialties, from tax inspectors and auditors to IT specialists and legal experts.

1.1.3 Partners and Customers

The DGI's main "customers" are the taxpayers, which include individuals, companies, and other legal entities subject to Moroccan tax jurisdiction. The organization serves a diverse clientele ranging from small individual taxpayers to large multinational corporations operating in Morocco.

The DGI's partners include other government agencies such as customs authorities, social security institutions, and various ministries. Professional partnerships exist with organizations like the Order of Chartered Accountants (Ordre des Experts Comptables), legal professionals, and tax advisors who assist taxpayers in meeting their obligations.

On the international level, the DGI collaborates with international organizations such as the OECD, African Tax Administration Forum (ATAF), and maintains bilateral relationships with tax administrations worldwide for information exchange and capacity building purposes.

1.2 General Context of the Project

This chapter provides a comprehensive overview of the internship's setting within the Moroccan Tax Administration (DGI). It establishes the institutional context, identifies the operational challenges that motivated this project, and outlines the objectives and methodology adopted to develop an intelligent RAG-based chatbot for fiscal document processing.

1.2.1 Project Presentation

Moroccan Tax Administration (DGI) plays a central role in Morocco's fiscal system, responsible for administering tax laws, collecting revenues, and providing services to taxpayers. **Each year, the institution publishes extensive legal and regulatory**

documentation, including the General Tax Code (CGI) and annual circular notes that explain the provisions introduced by finance laws.

While these documents constitute the primary reference for DGI employees, their length, complexity, and non-standardized structure make them difficult to use in daily operations. Staff often need to locate precise information across thousands of pages, a task that is both time-consuming and prone to errors. This problem directly impacts productivity and the quality of services delivered to taxpayers.

This internship project was therefore dedicated to the design and development of an intelligent assistant based on Retrieval-Augmented Generation (RAG). The system is intended to make fiscal knowledge more accessible, allowing DGI employees to query large volumes of legal text in natural language and receive accurate, cited answers drawn directly from official documents.

1.2.2 Problem Statement

The operational challenges faced by DGI staff arise mainly from the nature of fiscal documentation:

Excessive volume: The 2011 circular note alone spans 1,240 pages across three volumes. From 2011 to 2025, fifteen additional circular notes were published, each averaging 60 pages. Together, this represents more than 2,100 pages of legal guidance.

Non-standardized format: Many documents exist only as scanned images, which are not directly searchable. This prevents straightforward keyword search and forces employees to manually scan entire sections.

Information retrieval difficulties: The mixture of long textual explanations, tables, and heterogeneous structures complicates navigation. Employees often spend significant time reviewing multiple documents to resolve a single query.

Impact on efficiency: The inability to quickly access reliable, up-to-date information leads to delays, lower productivity, and risks of misinterpretation when advising taxpayers or applying fiscal law.

These factors highlight the necessity of an innovative solution capable of transforming how employees interact with fiscal knowledge resources.

1.2.3 Project Objectives

The project was designed to deliver a practical and high-value solution for DGI employees. Its objectives include:

- **Developing a RAG-based chatbot:** Building an intelligent assistant that combines information retrieval with generative AI to provide precise, contextually relevant answers.
- **Efficient indexing of fiscal documents:** Creating robust preprocessing and chunking strategies tailored to the structure of the CGI and circular notes, enabling accurate search and retrieval.
- **Reliable source attribution:** Ensuring that every answer is supported by explicit references (article number, circular note, page), thus guaranteeing transparency and legal compliance.

- **Improved information accessibility:** Allowing staff to save considerable time by replacing manual searches with natural language queries.
- **Handling document heterogeneity:** Using OCR and intelligent text extraction to process scanned and poorly structured files effectively.
- **Supporting decision-making:** Providing employees with a reliable tool that reduces uncertainty and helps enforce tax regulations consistently.

1.3 Project Planning

The development process was divided into five phases over eight weeks:

1. **Data acquisition and preprocessing:** Collection of fiscal documents, OCR of scanned files, and initial cleaning.
2. **Chunking strategy development:** Experimentation with segmentation approaches to capture both text and tabular data.
3. **RAG model integration:** Selection and configuration of the retrieval and generation pipeline.
4. **Chatbot and interface development:** Implementation of the user-facing interface with citation functionality.
5. **Testing and evaluation:** Accuracy assessments, performance measurements, and user acceptance testing.

A Gantt chart was used to track deliverables and ensure timely completion of milestones.

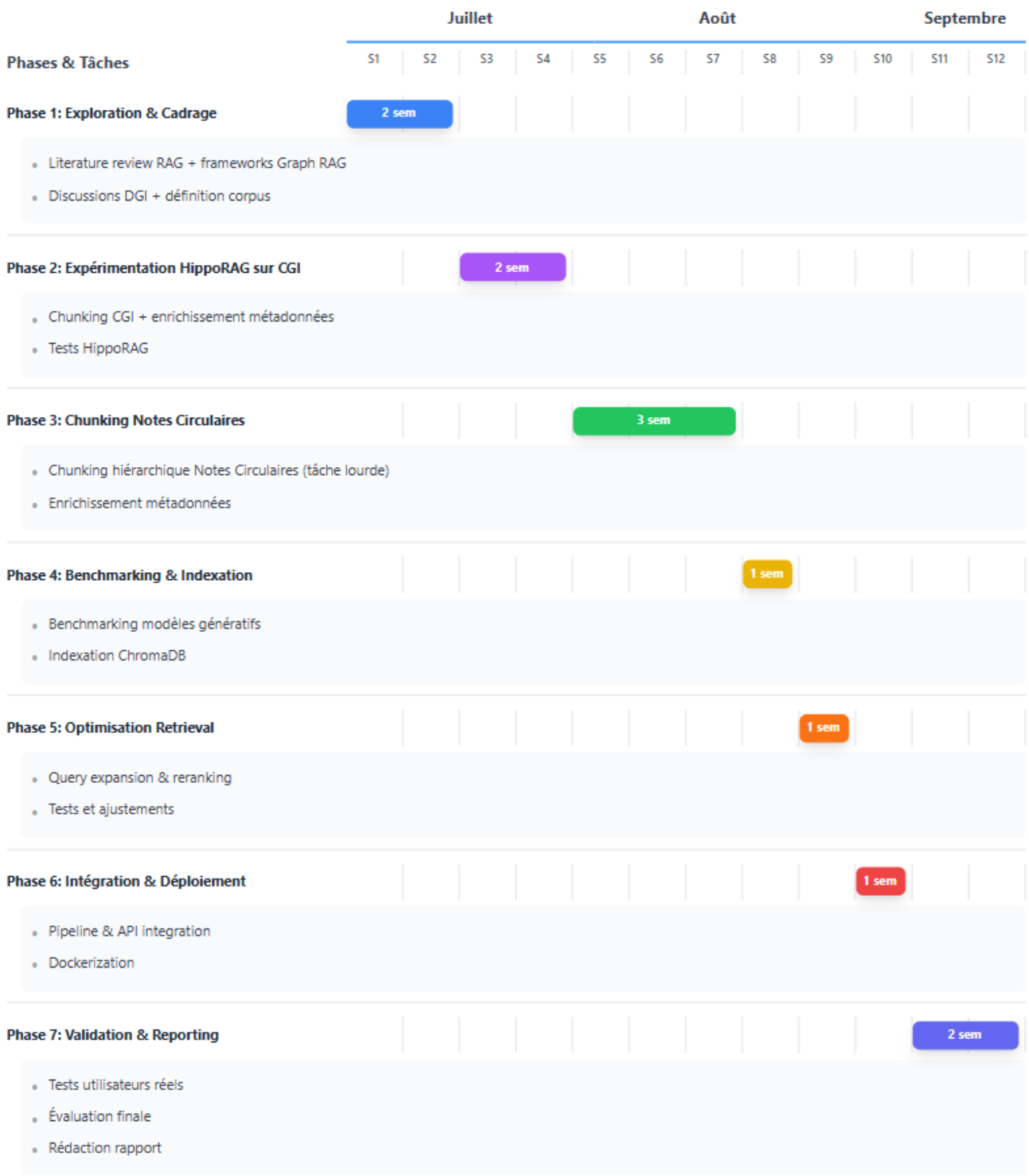


Figure 1.3: gantt chart

1.4 Conclusion

This chapter has outlined the institutional context of the DGI, the challenges created by the scale and structure of fiscal documentation, and the objectives guiding this project. By addressing concrete issues of information accessibility, reliability, and efficiency, the project aims to demonstrate how advanced AI techniques specifically Retrieval-Augmented Generation can improve public administration practices.

The following chapters will build upon this foundation, presenting the technical analysis, design, and implementation details of the proposed solution.

Chapter 2

System Study and Analysis

Introduction

Moroccan Tax Administration (DGI) manages an extensive corpus of fiscal documents that are essential for daily operations but challenging to navigate efficiently. This chapter analyzes the specific characteristics of Moroccan fiscal documentation, examines the technical challenges they present, justifies the selection of Retrieval-Augmented Generation (RAG) as the architectural approach, and establishes the functional and non-functional requirements that guided system development.

2.1 Structural Heterogeneity

A critical characteristic of the DGI corpus is the **lack of structural consistency** across annual circulars. While 'Note Circulaire' NC 717 (2011) maintains a stable nine-level hierarchy, the organizational patterns of amendment documents evolved significantly between 2011 and 2025.

Foundational Structure: NC 717 (2011)

The 2011 commentary note (**Note Circulaire 717**) introduces a *multi-level book structure*, which follows classical legal drafting traditions and serves as the baseline reference for subsequent fiscal texts:

```
LIVRE PREMIER (Level 1)
   PREMIERE PARTIE (Level 2)
       TITRE PREMIER (Level 3 - Roman)
           CHAPITRE PREMIER (Level 4 - Text)
               I- Section (Level 5 - Roman + dash)
                   5- Subsection (Level 6 - Arabic numeral)
                       a- Sub-subsection (Level 7 - lowercase)
```

This hierarchical depth makes NC 717 structurally rich but also particularly complex to parse computationally.

Pre-2021 Documents: Multiple Structural Variants

Between 2011 and 2020, annual circulars exhibit at least four distinct organizational patterns:

- **Pattern A: Inverted Hierarchy (e.g., NC 2014, 2016, 2018)** Structure begins with Roman numerals, followed by capital letters and then Arabic numerals, inverting the order compared to NC 717.

```
I. SECTION PRINCIPALE
   A- Sous-section
       1- Mesure
           a- Détail
```

- **Pattern B: Narrative-Example Structure (unique in NC 2019)** The 2019 circular (NC 729) is split into two parts with heterogeneous formats:
  - *Partie 1* follows the conventional hierarchical style (Roman numerals, Arabic numerals, capital letters).
  - *Partie 2* abandons legal structuring and shifts entirely to illustrative case studies, each labeled sequentially:

    ```
    Exemple n°1: Cas d'une société X
    Exemple n°2: Cas d'une société Y
    Exemple n°3: Cas d'une société Z
    ...
    ```

    This dual-format document introduces a hybrid nature: normative commentary in Part 1, and pedagogical simulations in Part 2. For automated chunking, NC 2019 is challenging because examples mix narrative text, numeric calculations, and tabular summaries.
- **Pattern C: Hybrid Split Structure (unique to NC 2020)** The 2020 circular is exceptional because it is divided into two separate PDFs:
  - **Part 1** preserves a hierarchical organization similar to earlier circulars, with nested numbering and subdivisions.
  - **Parts 2 & 3**, in contrast, abandon hierarchical nesting and follow a *flat enumeration*, where fiscal measures are simply listed as:

    ```
    1- Mesure fiscale n°1
    2- Mesure fiscale n°2
    3- Mesure fiscale n°3
    ```

  This duality (structured Part 1 vs. flat Parts 2 & 3) makes NC 2020 particularly challenging for automated chunking, as different parsing rules must be applied within the same year.
- **Pattern D: Post-2021 Standardization**

  Starting with NC 730 (2021), the DGI adopted a **unified structure** applied consistently until NC 736 (2025). This standardized format uses a six-level hierarchy with Roman numerals, Arabic numerals, and capital letters, e.g.:

```
I. MESURES SPÉCIFIQUES
   1- Mesure détaillée
       A- Sous-mesure
           2- Point spécifique
               b) Détail
                    Énumération
```

This marks a turning point toward normalization, facilitating both human consultation and automated processing.

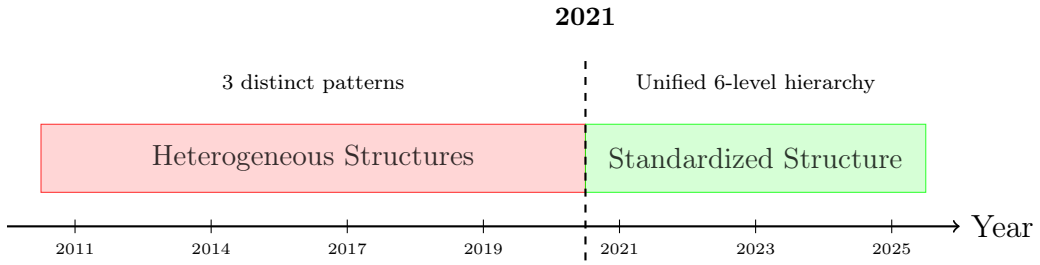


Figure 2.1: Evolution of document structure in DGI annual circulars (2011–2025).

2.2 Technical and Domain-Specific Challenges

The development of an intelligent assistant for the DGI requires addressing challenges that are both technical and inherently tied to the fiscal domain.

2.2.1 Scanned Documents and OCR Requirements

A substantial portion of documents from earlier years exist only as scanned images without searchable text layers. Moreover, **all circulars** include signature pages in scanned form with official signatures, administrative stamps, and legal attestations. This creates two distinct OCR requirements:

- **Full-document OCR:** For older circulars available only as scanned images
- **Selective OCR:** For signature pages across all documents

OCR is complicated by low scan quality, handwritten annotations in margins, mixed French-Arabic bidirectional text, and physical document degradation (faded text, stains).

2.2.2 Multilingual Content

Circulars consistently mix French and Arabic, though not in uniform ways:

- Legal articles are predominantly in French
- Administrative procedures and annotations appear in both languages
- Article references may combine Arabic numerals with French text or vice versa

This heterogeneity complicates text extraction, language detection at the chunk level, semantic embedding across languages, and appropriate response generation depending on the query language.

2.2.3 Complex Hierarchical Structures

The corpus exhibits significant structural variance:

- Early circulars include deeply nested hierarchies (up to very fine-grained levels)
- Pre-2020 annual circulars follow heterogeneous organizational patterns
- Circulars from 2021 onward adopt a standardized multi-level hierarchy

Parsing systems must therefore handle **multiple structural paradigms simultaneously** rather than relying on fixed rules. Structural markers are also context-dependent: for example, "I-" may denote a Level 1 section in recent circulars, a subsection in older ones, or simply a legal reference ("article I-52 du CGI").

2.2.4 Tabular Data Heterogeneity

Circulars contain numerous tables presenting tax rates, thresholds, and exemption schedules. These tables vary in formatting (with or without borders, merged cells, nested headers), header positioning, and data types (numeric values, textual descriptions, formulas). Some are even embedded as images rather than native PDF objects.

To be interpretable, tables must be semantically linked to surrounding text, as isolated tables lack sufficient context for accurate retrieval.

2.3 RAG Principles and Architecture

2.3.1 Core RAG Concept

Retrieval-Augmented Generation combines two complementary components:

1. A **retriever** that searches an indexed knowledge base to find relevant document passages
2. A **generator** (LLM) that formulates coherent, natural language answers by synthesizing the user query and retrieved context

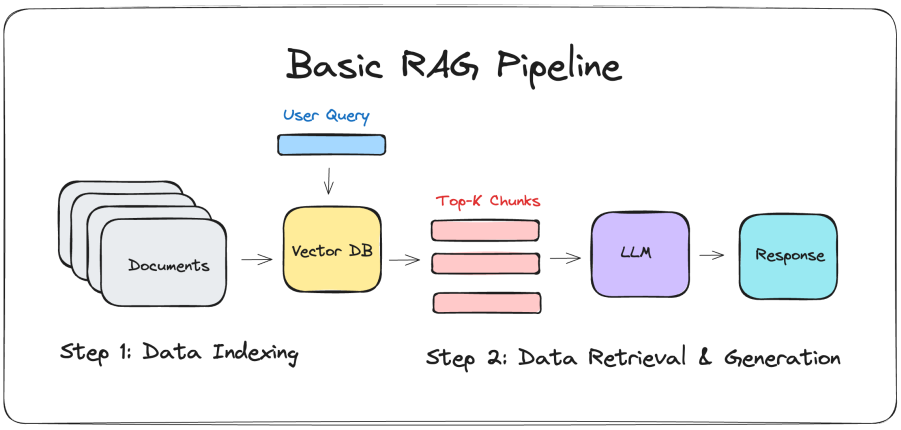


Figure 2.2: Basic RAG pipeline

2.3.2 Comparison: Pure LLM versus RAG

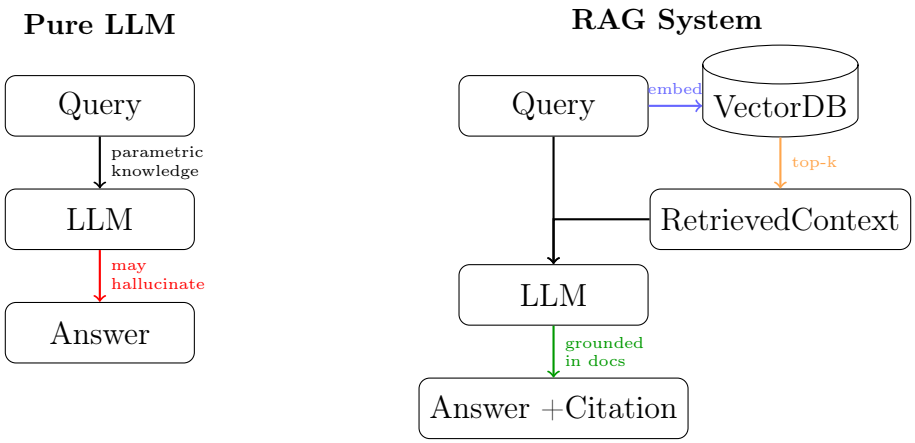


Figure 2.3: Comparison of pure LLM versus RAG architecture. RAG grounds generation in retrieved documents, reducing hallucinations and enabling source attribution.

This architectural approach is crucial for fiscal administration, where accuracy and traceability are non-negotiable requirements.

2.3.3 High-Level System Architecture

The implemented system follows a modular two-phase architecture separating offline indexing from online query processing.

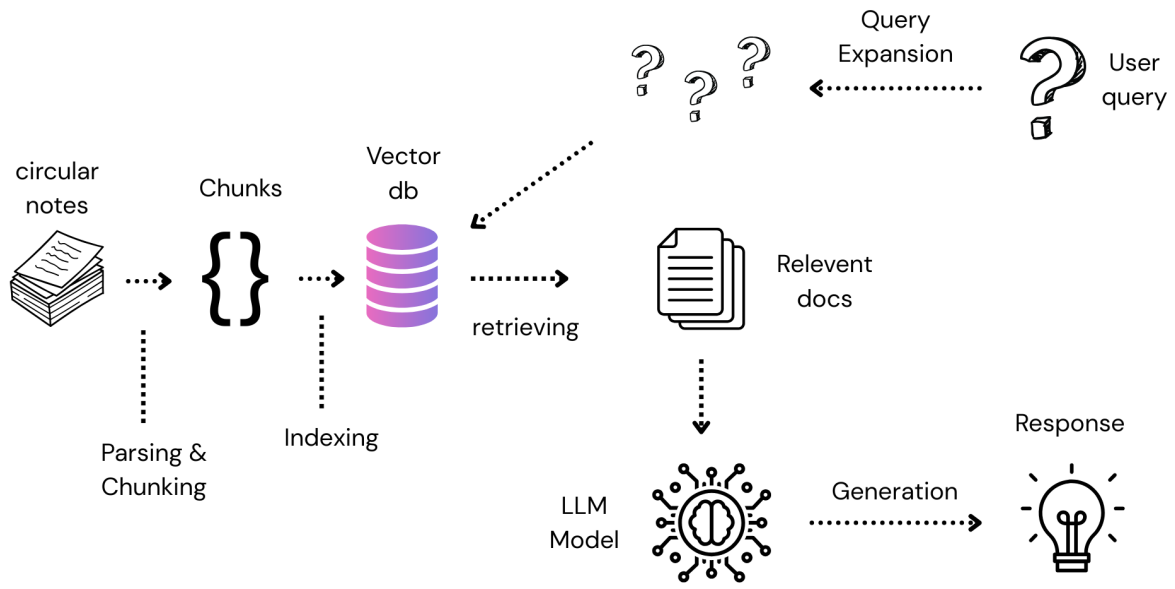


Figure 2.4: High-level system architecture

Indexing Phase (Offline)

The indexing phase transforms raw PDF documents into searchable vector representations:

1. **Text and table extraction:** Content is extracted using multiple specialized libraries to handle native PDFs, scanned images, and tabular data
2. **Hierarchical chunking:** Documents are segmented according to their structural patterns while preserving legal hierarchy
3. **Embedding generation:** Text chunks are encoded into dense vector representations using multilingual semantic models
4. **Vector storage:** Embeddings, original text, and metadata are persisted in a vector database for efficient similarity search

Query Phase (Online)

The query phase processes user requests in real-time:

1. **Query expansion:** Original queries are augmented with semantic variations to improve retrieval recall
2. **Semantic retrieval:** Expanded queries are embedded and matched against the vector database using similarity search
3. **Re-ranking:** Retrieved candidates are re-scored using cross-attention models to prioritize true relevance

4. **Answer generation:** Top-ranked chunks are formatted as context and sent to an LLM with strict citation instructions

5. **Response formatting:** The final answer includes explicit references to source documents with verifiable identifiers

2.4 Requirements Specification

2.4.1 Functional Requirements

Table 2.1: Functional requirements with operational justifications.

ID	Requirement	Justification
FR1	Process and index complete fiscal corpus (CGI commentary and annual circulars 2011–2025)	Employees need access to complete historical legislation for comparative analysis and understanding regulatory evolution.
FR2	Provide accurate answers with precise source citations (year, section, page)	DGI accountability standards require traceability. Employees must verify information before advising taxpayers or making decisions.
FR3	Support queries in both French and Arabic	Significant portion of DGI staff prefer Arabic interface. Documents contain both languages, requiring multilingual understanding.
FR4	Extract and present relevant tabular data	A large percentage of queries concern numeric thresholds and tax rates stored in tables. Tables must be searchable and contextualized.
FR5	Preserve hierarchical document structure	Legal reasoning depends on understanding article context within broader sections, chapters, and titles.
FR6	Handle heterogeneous document structures	Documents from 2011–2020 follow different organizational patterns than 2021–2025. System must adapt dynamically.
FR7	Process both native PDF text and scanned images	Significant portion of corpus is fully scanned; all documents have scanned final pages. OCR capability is mandatory.

2.4.2 Non-Functional Requirements

Table 2.2: Non-functional requirements addressing performance, compliance, and maintainability.

ID	Requirement	Justification
NFR1	Query response time under 10 seconds in normal load conditions	Response time exceeding 10 seconds disrupts workflow. Target: 95th percentile under 8 seconds.
NFR2	Scalability to support concurrent users (target: 50+ simultaneous queries)	DGI has hundreds of potential users. Peak usage occurs during tax filing periods.
NFR3	Compliance with Moroccan data sovereignty and protection regulations	Fiscal data must not leave national territory. System must be deployable on-premises.
NFR4	Modular, containerized deployment	Ensures reproducibility across development, staging, and production environments. Facilitates updates and rollbacks.
NFR5	Explainability and debuggability	System must log retrieval scores, selected chunks, and model responses for quality assurance and error analysis.
NFR6	Extensibility for future enhancements	Architecture must accommodate new embedding models, LLMs, and document types without major refactoring.

Conclusion

This chapter analyzed the technical and domain-specific challenges of Moroccan fiscal documentation and justified the selection of Retrieval-Augmented Generation as the architectural approach. Key findings include:

- The DGI corpus exhibits significant structural heterogeneity, with at least four distinct organizational patterns across 16 documents spanning 3,690 pages
- Traditional keyword search and pure LLM approaches are insufficient for this domain, necessitating a hybrid RAG solution combining semantic retrieval with controlled generation
- The multilingual nature of documents (French-Arabic), combined with OCR requirements and tabular data, creates unique processing challenges
- A modular two-phase architecture (offline indexing, online querying) provides the necessary separation of concerns for efficient scaling
- Both functional and non-functional requirements emphasize accuracy, traceability, and performance, which are critical factors for effective fiscal administration.

The following chapter will detail the technical implementation of each component, including document processing pipelines, embedding strategies, vector indexing, retrieval mechanisms, and API design.

Chapter 3

System Architecture and Implementation

Introduction

This chapter outlines the system architecture and implementation of the Retrieval Augmented Generation (RAG) system for Moroccan fiscal documents. It presents the modular architecture, describes the configuration and environment management, details the adaptive document-processing pipeline (including OCR enhancement and table normalization), introduces the hierarchical chunking strategy, and covers the vector embedding, retrieval and generation components. It also describes the API layer and containerized deployment. The aim is to provide a reproducible, scalable and maintainable technical foundation for the system.

3.1 System Architecture Overview

The system follows a modular architecture composed of seven core components, each corresponding to a specific stage of the RAG pipeline.

Table 3.1: System modules and their main responsibilities.

Module	**Primary Function**	**Key Technologies**
`config.py`	Configuration management	Python pathlib, environment variables
`chunker.py`	Document processing & segmentation	PyMuPDF, pdfplumber, Camelot, EasyOCR, OpenCV
`data_loader.py`	Data normalization & serialization	JSON, regex metadata extraction
`indexer.py`	Vector embedding & persistence	SentenceTransformers, ChromaDB
`retriever.py`	Semantic search & reranking	ChromaDB query API, CrossEncoder
`generator.py`	Context formatting & LLM query generation	LangChain, OpenRouter API, Ollama
`main_api.py`	Web API orchestration	FastAPI, OpenAI-compatible endpoints

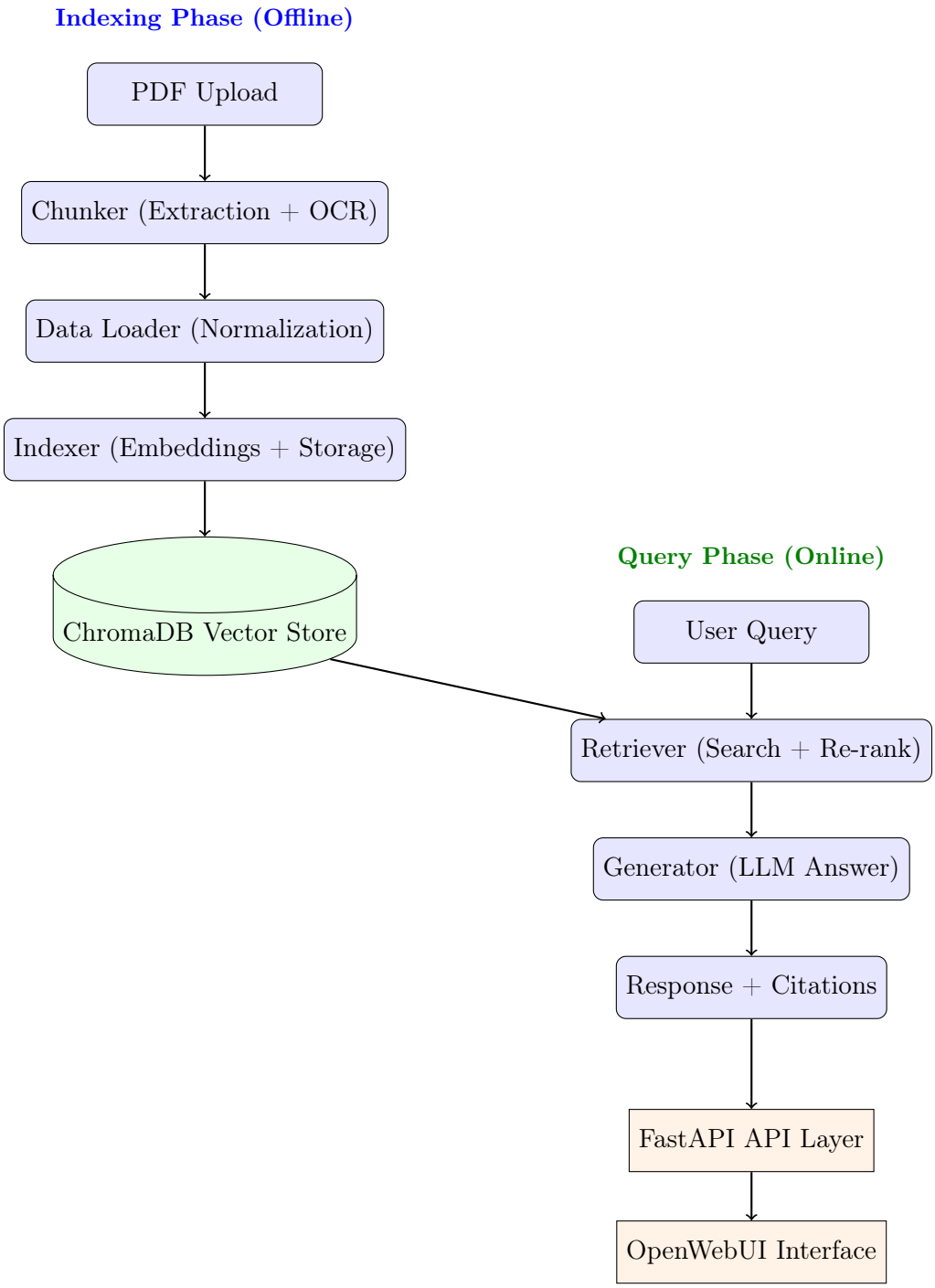


Figure 3.1: Global RAG architecture showing offline indexing and online query phases.

The architecture separates offline document indexing from online user queries, allowing independent scaling and optimized performance for both pipelines.

3.2 Configuration Management

The `config.py` module centralizes system configuration to guarantee consistency across environments. It defines all directory paths, embedding model names, and vector database

settings. Sensitive credentials such as API keys are never hardcoded but are loaded from a `.env` file using the `python-dotenv` library.

3.2.1 Environment Variables and Directory Layout

Key variables include:

- `OPENROUTER_API_KEY` – authentication token for remote inference
- `COLLECTION_NAME` – name of the ChromaDB vector collection
- `CHROMA_DB_PATH` – storage directory for the vector store
- `EMBEDDING_MODEL` – embedding model used by SentenceTransformers

This separation ensures reproducibility and isolation between ingestion, preprocessing, and vector storage phases.

3.2.2 Embedding Model Selection

The system uses `paraphrase-multilingual-mpnet-base-v2`, which produces 768-dimensional vectors and supports French, Arabic, and English natively. This model offers strong semantic performance for multilingual paraphrasing while maintaining manageable resource usage.

3.3 Document Processing Pipeline

The `chunker.py` module performs adaptive text extraction, OCR recovery, and hierarchical segmentation.

3.3.1 Multi-Source Text Extraction

Extraction follows a cascading approach:

1. `pdfplumber` [11] for standard digital PDFs.
2. `PyMuPDF` [10] as a fallback for embedded or irregular fonts.
3. `EasyOCR` [8] for scanned pages after OpenCV preprocessing.

3.3.2 OCR Enhancement Pipeline

The OCR preprocessing pipeline enhances recognition by zooming the page image and applying adaptive thresholding before passing it to EasyOCR. The strategy differs depending on whether the document is fully scanned or digitally generated:

- If the PDF is detected as *scanned* (i.e., all pages are images without embedded text), OCR is applied to **all pages**.

- Otherwise, OCR is applied **only to the last page**, since it typically contains the director's signature and is always scanned even in digitally generated documents.

Each page processed via OCR is rendered using PyMuPDF with a 3× zoom factor (fitz.Matrix(3.0, 3.0)). This approach improves the quality of the OCR processing by enhancing text clarity for the recognition engine.

The OCR pipeline follows these steps:

1. Render the page as an image (pixmap) using PyMuPDF with 3× zoom.
2. Convert the image to RGB, then to grayscale via `cv2.cvtColor`.
3. Apply adaptive thresholding (`cv2.adaptiveThreshold`) to enhance text contrast.
4. Run EasyOCR on the thresholded image; if recognition fails, fall back to *pdfplumber.extract_text*()

3.3.3 Table Extraction and Normalization

Camelot [9] is used in both "lattice" and "stream" modes depending on the presence of borders. Extracted tables are flattened into a textual format (`Header: Value | ...`) to make them searchable in the embedding space.

3.3.4 Adaptive Hierarchical Chunking

The chunking algorithm identifies structural patterns such as "LIVRE PREMIER" or "1- Section" to detect hierarchy levels. Each chunk is assigned a hierarchical ID like `2025.I.1.A.p45`, preserving the logical context across queries.

3.4 Vector Indexing

The `indexer.py` module converts textual chunks into dense vectors and stores them persistently using **ChromaDB**.

3.4.1 Vector Store Structure

ChromaDB is initialized locally via:

```
client = chromadb.PersistentClient($path=db\_path$)
collection = client.get_or_create_collection(name=collection_name)
```

Each entry contains text, metadata, and a precomputed 768-dimensional vector generated by the `paraphrase-multilingual-mpnet-base-v2` model.

```
{
  "chunk_id": "2025.I.1.A.p45",
  "content": "...",
  "year": "2025",
  "page": 45,
  "source": "IS_2025.pdf"
}
```

3.4.2 Persistence and Scalability

The index is stored persistently under `/app/chroma/` and survives container restarts. When `force_reindex=True`, the collection is rebuilt from scratch, useful during schema changes or document updates.

3.4.3 Indexing Performance

Document indexing is performed in batches to balance throughput and memory consumption. Batch size (`batch_size`) can be adjusted, allowing the system to encode multiple chunks in parallel before inserting them into the Chroma database.

No precise benchmarking has been carried out in this version. The real throughput (in terms of chunks encoded or inserted per second) depends on the hardware (CPU/GPU) and execution environment. For production deployment, performance should be measured empirically and the batch size tuned accordingly.

3.5 Document Retrieval

The `retriever.py` module implements semantic retrieval enhanced by query expansion and optional re-ranking.

3.5.1 Query Expansion: Definition and Implementation

Principle. *Query expansion* consists of generating multiple semantic variants of a single query to increase recall during dense retrieval. By capturing equivalent formulations (synonyms, paraphrases, expanded acronyms), the system reduces "zero-results" scenarios and retrieves relevant passages even when the exact wording does not match the user's formulation [15].

In our system. Query expansion is implemented in `retriever.py` via the `SimpleQueryExpander` class. For each query q, three variants are produced: (1) q as-is, (2) an answer-oriented paraphrase (*"La réponse à la question ... est :"*), (3) a topic-oriented variant (*"Concernant ..."*). Common fiscal acronyms are expanded before encoding (`IS` → *Impôt sur les Sociétés*, `IR` → *Impôt sur le Revenu*, `TVA` → *Taxe sur la Valeur Ajoutée*).

Example:

```
User input: "Quel est le taux proportionnel de l'IS après la LF 2022 ?"

Generated variants:
1) "Quel est le taux proportionnel de l'Impôt sur les Sociétés
    après la Loi de Finances 2022 ?"
2) "La réponse à la question 'Quel est le taux proportionnel
    de l'IS après la LF 2022 ?' est :"
3) "Concernant le taux proportionnel de l'IS après la LF 2022"
```

Each variant is encoded (bi-encoder embeddings), queries the vector database (*top-k* per variant), then results are merged and deduplicated by `chunk_id`, keeping the best score. Optionally, a year filter (`year_filter="2022"`) can restrict the search via metadata.

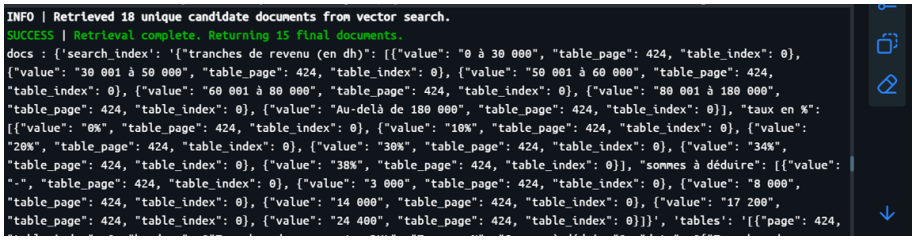


Figure 3.2: Retrieval logs

Remark. Query expansion improves recall but may introduce noise. The *re-ranking* step (next section) serves to promote the best candidates.

3.5.2 Re-ranking: Definition and Implementation

Principle. *Re-ranking* reorders a small list of candidates (*top-N*) from the semantic search using a more precise model (a CrossEncoder). While the bi-encoder relies on cosine similarity of separate embeddings, the CrossEncoder jointly considers the query and each passage to produce a relevance score $s_{\text{CE}}(q, d)$. [14, 13].

Formulation. For a set of candidates $D = \{d_1, \ldots, d_N\}$ obtained by cosine similarity $s_{\cos}(q, d)$, re-ranking applies a model f_θ (here `ms-marco-MiniLM-L-6-v2`) such that:

$$s_{\text{CE}}(q, d_i) = f_\theta([q; d_i]) \quad \Rightarrow \quad \text{sort by } s_{\text{CE}} \text{ (descending) and select the top-}K\,.$$

In `retriever.py`, this step is optional (controlled by `use_reranker`). The final number of documents returned is set by `top_n_final`, whose default value is 15.

Why is this useful in fiscal contexts?

- *Fine disambiguation*: differentiate between close concepts (e.g., "proportional rate" vs. "progressive scale").
- *Sensitivity to numbers*: favour passages that mention exactly the requested rate or threshold.
- *Robustness to terminology variants*: handle different expressions of fiscal terms (IS vs Corporate Tax, etc.).

Pipeline summary:

1. (Bi-encoder) Embedding of each query variant and of the chunks → vector search (*top-k* per variant).
2. Merge deduplicate by `chunk_id`, keeping the best initial score.
3. (CrossEncoder) Optional re-ranking of the N aggregated candidates → retain the top-K results for the LLM.

Control and costs. Re-ranking improves top-K precision but adds latency (CrossEncoder inference on N `(q, d)` pairs). In our API, it can be disabled via `use_reranker=False` for ablation tests or when very low latency is required.

Retrieval logs showing aggregated candidates and final documents. After query expansion, the system merges and deduplicates the retrieved chunks, The system then retrieves 18 candidate chunks and returns 15 final documents from the vector search with detailed metadata (page number, section hierarchy, source file).

3.6 Response Generation

The `generator.py` module formats the retrieved context and interacts with LLMs via `llm_loader.py`.

3.6.1 Prompt Structure and Context Assembly

Each retrieved chunk is annotated with its metadata (year, page, section) and concatenated into a structured prompt:

```
SYSTEM: You are a fiscal expert on Moroccan taxation.
Answer using only the provided documents and cite sources.
USER: What is the IS proportional rate for 2025?
CONTEXT:
[IS_2025.pdf - I.1.A - Page 45]
The proportional corporate tax rate for 2025 is 20%...
```

This approach ensures evidence-based, non-hallucinatory answers.

3.6.2 Language Detection

Regex-based heuristics detect Arabic script to select the corresponding system prompt. Each chunk in the retrieved context is annotated with metadata to allow contextual grounding and citation by the language model.

3.6.3 LLM Invocation

The generator can invoke:

- **OpenRouter API** (for remote models like DeepSeek, Mixtral) [2]
- **Ollama** (for local inference when offline) [7]

The model choice is defined in the `.env` file. Both OpenRouter-hosted and locally deployed Ollama models are supported, though switching is manual (no automatic failover).

3.6.4 Post-Processing

After the relevant passages are retrieved, the final answer is generated by the LLM module (`generator.py`). This module builds a single prompt containing the context (list of chunks) and the user's question, then sends it either to an OpenRouter model or to a local Ollama model. The output is a *single text string* answering the question based on the provided context.

The current pipeline does not return a structured JSON object with `answer`, `sources`, or `metadata` fields. Citations are implicitly included in the prompt sent to the LLM but are not separately exposed in the response.

3.7 API Integration

The `main_api.py` module exposes a RESTful interface built with **FastAPI**.

3.7.1 Endpoints

- `POST /v1/chat/completions` – generates answers from queries (OpenAI-compatible endpoint)
- `POST /v1/documents/upload` – dynamically indexes new PDFs

File uploads trigger the full chunking and indexing pipeline asynchronously.

3.7.2 Component Initialization

Heavy models (embeddings, reranker, generator) use **lazy initialization** to minimize startup latency. The `pipeline.py` module enforces the **singleton** pattern to ensure only one instance of each model exists per process.

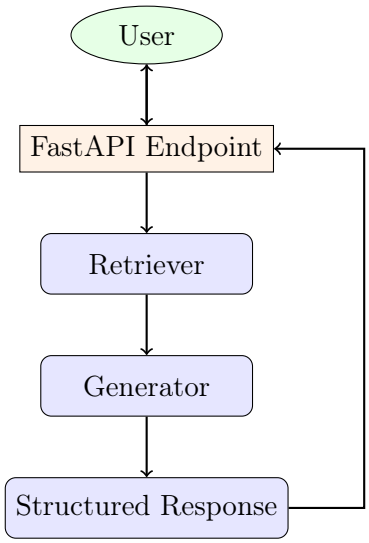


Figure 3.3: Simplified API call flow.

The backend creates the VectorIndexer, loads the multilingual embedding model, connects to the persistent ChromaDB collection, and initializes the chunker, retriever, re-ranking model and generator. Once these components are ready, the RAG pipeline can handle incoming queries.

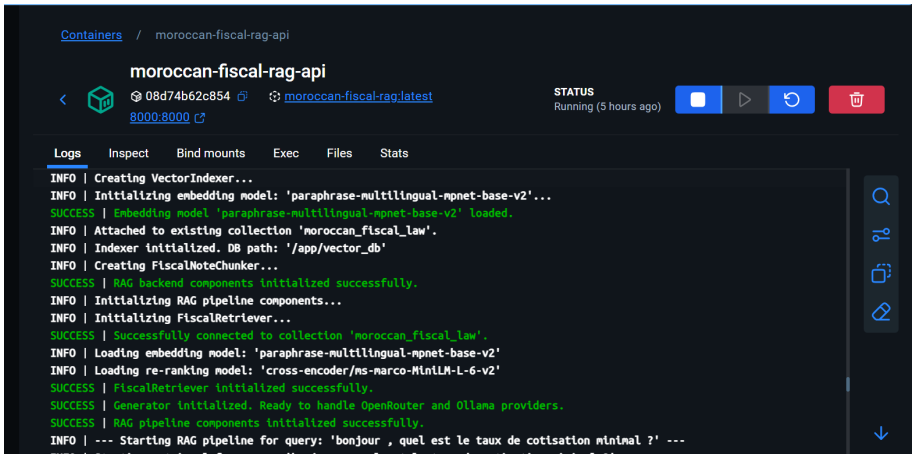


Figure 3.4: API start-up logs

3.8 Containerization and Deployment

The system is containerized using **Docker** [6] to ensure reproducibility and ease of deployment.

3.8.1 Docker Structure

- `Dockerfile` – defines the RAG API image
- `docker-compose.yml` – orchestrates API and ChromaDB volume
- `.env` – configuration variables (keys, ports, model names)
- `build.ps1` and `fresh_deployment_reset.ps1` – automate build/reset on Windows

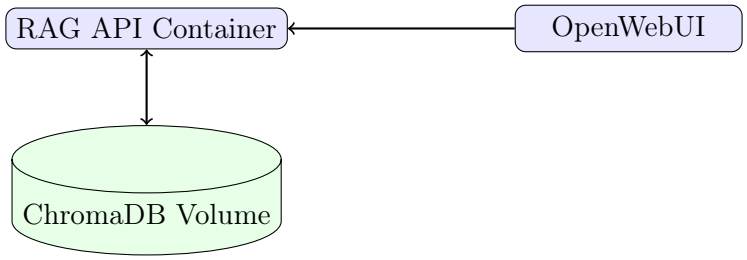


Figure 3.5: Dockerized architecture with persistent ChromaDB volume.

3.8.2 Build and Runtime

```
docker-compose build
docker-compose up -d
```

The container runs a FastAPI server linked to a persistent ChromaDB volume under `/app/chroma/`. Dependencies are installed from `requirements.txt` ensuring deterministic builds. Models are cached locally under `/root/.cache/torch/` to avoid repeated downloads.

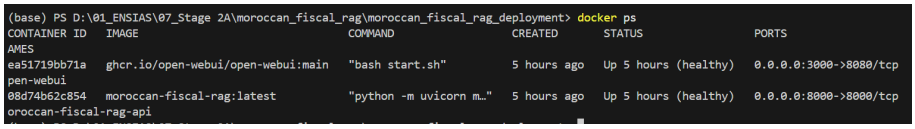


Figure 3.6: Docker status

This containerized setup allows immediate redeployment without manual configuration and can be hosted on any Linux or Windows server supporting Docker Desktop.

Figure 3.6 Docker status showing the two containers in operation. The moroccan-fiscal-rag-api and open-webui containers are both running and healthy, exposing ports 8000 and 3000 respectively.

Conclusion

This chapter described the complete technical implementation of the fiscal RAG system – from document ingestion to containerized deployment. The design emphasizes modularity, reproducibility, and scalability. Core technical achievements include:

- Multi-source text extraction with adaptive OCR
- Hierarchical chunking preserving document structure
- Persistent vector storage with ChromaDB
- Query expansion and re-ranking for high retrieval accuracy
- Multilingual generation with grounded, cited responses
- FastAPI-based API packaged in Docker for reproducible deployment

This implementation provides a robust and extensible baseline for advancing the system toward adaptive retrieval, domain-specific embedding optimization, and enterprise-grade orchestration with secure multi-tenant access control.

Chapter 4

Results and Evaluation

Introduction

This chapter focuses on the results and user-facing aspects of the RAG system. It enumerates the programming languages, frameworks, libraries and infrastructure used, details how the backend API was integrated into the OpenWebUI [3] interface, and illustrates the user experience through screenshots. It also demonstrates the system's capabilities on real fiscal queries, including OCR on scanned documents, multilingual conversations, and complex queries involving tables and formulas. The goal is to show how the technical system translates into a usable tool for DGI staff.

4.1 Technologies Used

This section presents the complete set of technologies, libraries, and environments used throughout the development of the fiscal RAG system. Each tool was selected for its robustness, maturity, and suitability for large-scale document processing and retrieval tasks.

Python 3.13.3 was used as the main programming language, running inside a dedicated Anaconda environment . Python's mature ecosystem for backend development, combined with native support for asynchronous programming via `asyncio`, makes it a natural fit for the FastAPI-based architecture. Version 3.13 introduces performance improvements in the CPython interpreter and enhanced async task scheduling, which benefit concurrent API workloads.

Figure 4.1: Python logo.

4.1.1 Frameworks and Servers

- **FastAPI** – used to build the RESTful API that exposes the RAG system. It offers asynchronous request handling, built-in OpenAPI documentation, and exceptional performance through the Starlette core.
- **Uvicorn** – lightweight ASGI server for running FastAPI applications efficiently in production environments.

Figure 4.2: FastAPI logo

Figure 4.3: Uvicorn logo

4.1.2 Core Libraries

Document Processing and OCR

- **pdfplumber** – extraction of textual content from native digital PDFs.
- **PyMuPDF (fitz)** – fallback library for complex or embedded text layouts.
- **Camelot** – detection and extraction of tabular data.
- **EasyOCR** – multilingual OCR engine used for scanned fiscal pages.
- **OpenCV** – preprocessing of scanned documents (thresholding, denoising).

Vectorization and Retrieval

- **SentenceTransformers** – encodes text chunks into multilingual semantic vectors.
- **ChromaDB** – lightweight vector database enabling persistent and efficient similarity search.
- **CrossEncoder (ms-marco-MiniLM-L-6-v2)** – used for semantic re-ranking to improve retrieval precision.

Utility and Orchestration

- **LangChain** – framework for constructing and managing LLM prompts and contexts.
- **requests** – for API calls to OpenRouter and other web services.
- **loguru** – structured logging and runtime monitoring.
- **tqdm** – progress bar visualization during batch operations.
- **python-dotenv** – environment variable management.
- **pathlib** – cross-platform filesystem handling.
- **re** (regex) – pattern-based parsing of fiscal document structure.
- **json** – serialization and storage of intermediate data.

4.1.3 LLM Providers and Models

The system supports both cloud-hosted and local inference backends:

- **OpenRouter** – provides access to high-performance open LLMs such as DeepSeek, Mixtral, and QwQ.
- **Ollama** – allows local inference using lightweight models for offline or private environments.

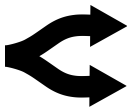


Figure 4.4: LLM provider: OpenRouter

Figure 4.5: LLM provider: Ollama (local).

4.1.4 Infrastructure and Deployment Environment

- **Docker** – containerization of the FastAPI backend and ChromaDB to ensure reproducibility and easy deployment.

Figure 4.6: Docker logo

4.1.5 Interfaces and External Tools

OpenWebUI – web interface used to interact with the chatbot in real time, supporting multilingual queries.

Figure 4.7: Openwebui logo

4.1.6 Development Environments

Development was initially performed in **Google Colab** for rapid prototyping of document parsing and embeddings. Once the core logic stabilized, the project migrated to **Visual Studio Code** for structured local development and integration with Docker.

Figure 4.8: Development environment: Google Colab

Figure 4.9: Development environment: Visual Studio Code

4.1.7 Summary

The combination of these technologies enables a modular, scalable, and maintainable Retrieval-Augmented Generation system capable of:

- Extracting and normalizing heterogeneous fiscal PDFs,
- Persistently storing semantic vectors for retrieval,
- Performing multilingual question answering through LLMs,
- Deploying seamlessly across environments via Docker containers.

4.2 User Interface Integration

The system's user interface was implemented using **OpenWebUI**, an open-source front-end designed for interacting with large language models (LLMs). Instead of developing a custom web client from scratch, the backend Retrieval-Augmented Generation (RAG) API was integrated directly into OpenWebUI through an `OpenAI-compatible` endpoint. This approach allowed rapid deployment, seamless testing, and access to advanced features such as model selection, chat history, and prompt customization without additional front-end engineering effort.

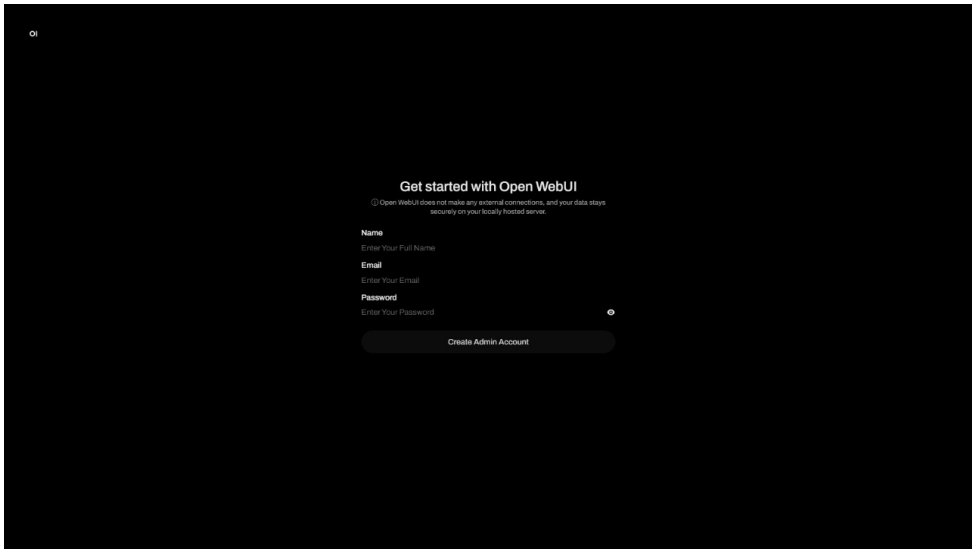


Figure 4.10: OpenWebUI login page for creating or accessing an admin account.

Figure 4.10 illustrates the OpenWebUI authentication interface. The platform provides an optional built-in authentication mechanism, allowing users to create an administrator account locally. This ensures that data and interactions remain securely hosted on the internal network, without external connections.

After authentication, users access the main workspace illustrated in Figure 4.11. The interface offers a minimalist design and an intuitive layout that includes essential interaction components.

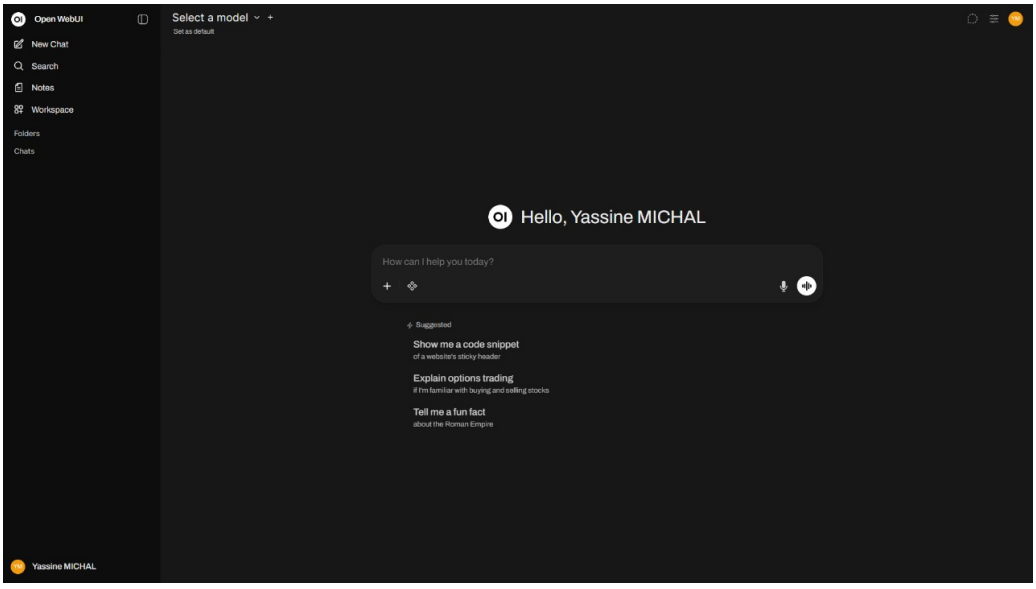


Figure 4.11: OpenWebUI main interface connected to the fiscal RAG backend.

Through this interface, users can submit natural language queries in French or Arabic, visualize generated responses, and inspect the cited fiscal document excerpts retrieved from the database. The sidebar includes several main sections:

- **New Chat**: Start a new query session.

- **Search**: Browse or filter previous conversations.
- **Notes & Workspace**: Organize saved results or annotations.
- **Settings**: Configure model preferences, response format, and language.

The decision to use OpenWebUI offers several practical advantages:

- **Rapid Integration:** The interface natively supports OpenAI-style APIs, making backend connection immediate.
- **Security and Isolation:** All data remains within the local network, ensuring fiscal document confidentiality.
- **User Experience:** Provides chat history, model switching, and prompt editing out of the box.
- **Extensibility:** Allows the addition of plugins or visualization modules for richer context inspection.

Authentication and user management were handled entirely by OpenWebUI, while this project focused on the design and integration of the backend RAG pipeline rather than developing new front-end components.

4.3 Model Selection and Query Interface

One of the key features of the system is its flexibility in model selection, allowing users to choose between different language models based on their specific needs regarding response quality, speed, computational resources, and data privacy requirements.

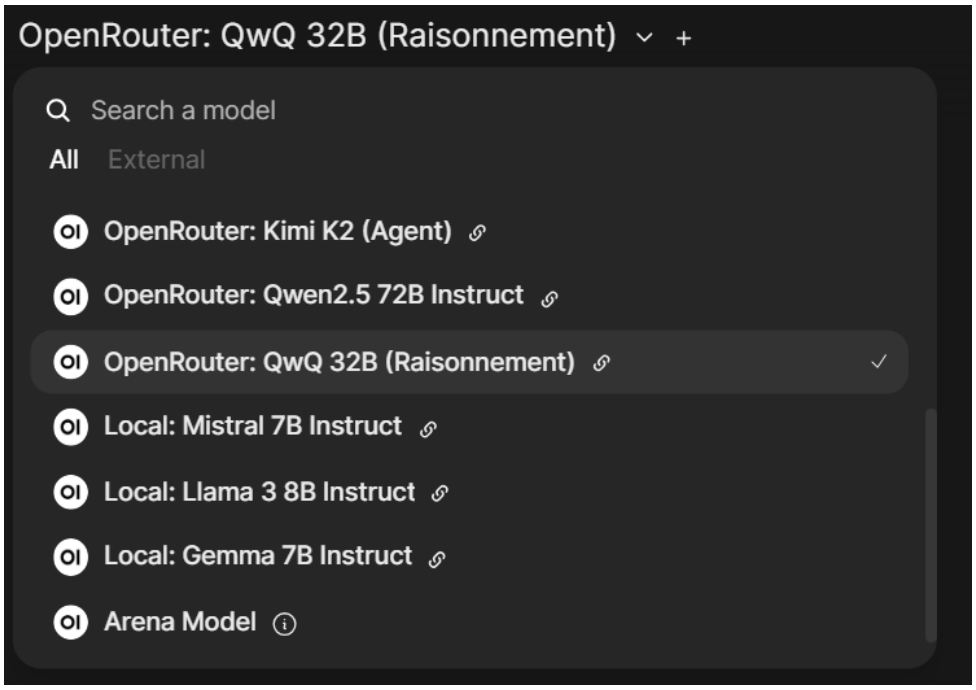


Figure 4.12: Model selection interface showing available providers (OpenRouter and Ollama).

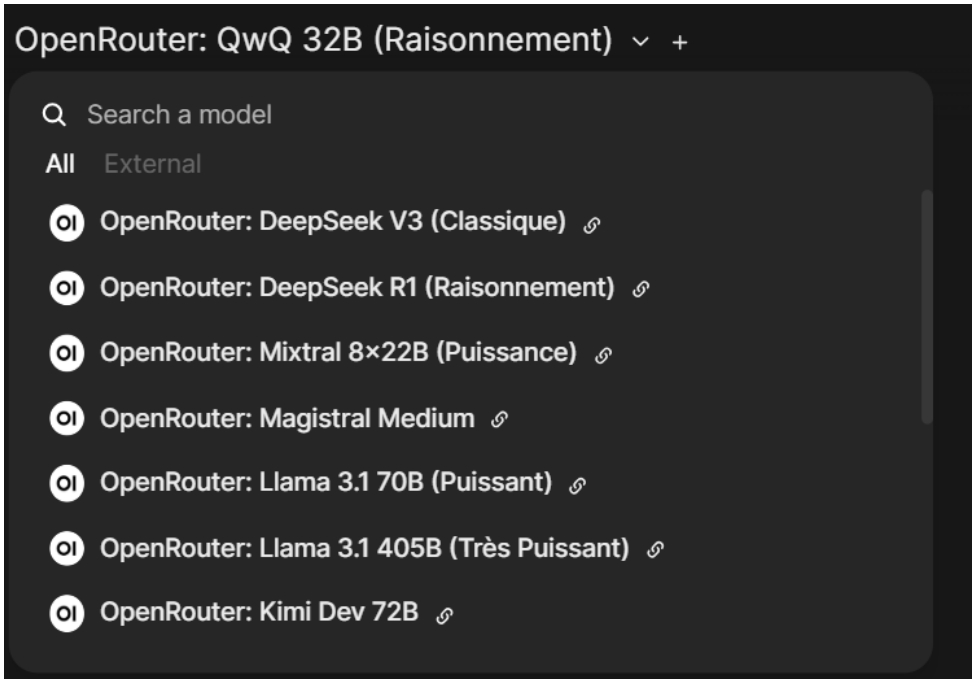


Figure 4.13: Expanded model options displaying specific models available through each provider.

Figures 4.12 and 4.13 illustrate the model selection interface. The system supports two primary providers, each offering distinct advantages. This flexibility allows users to balance between response quality, cost, latency, and privacy requirements depending on the nature and sensitivity of their fiscal queries.

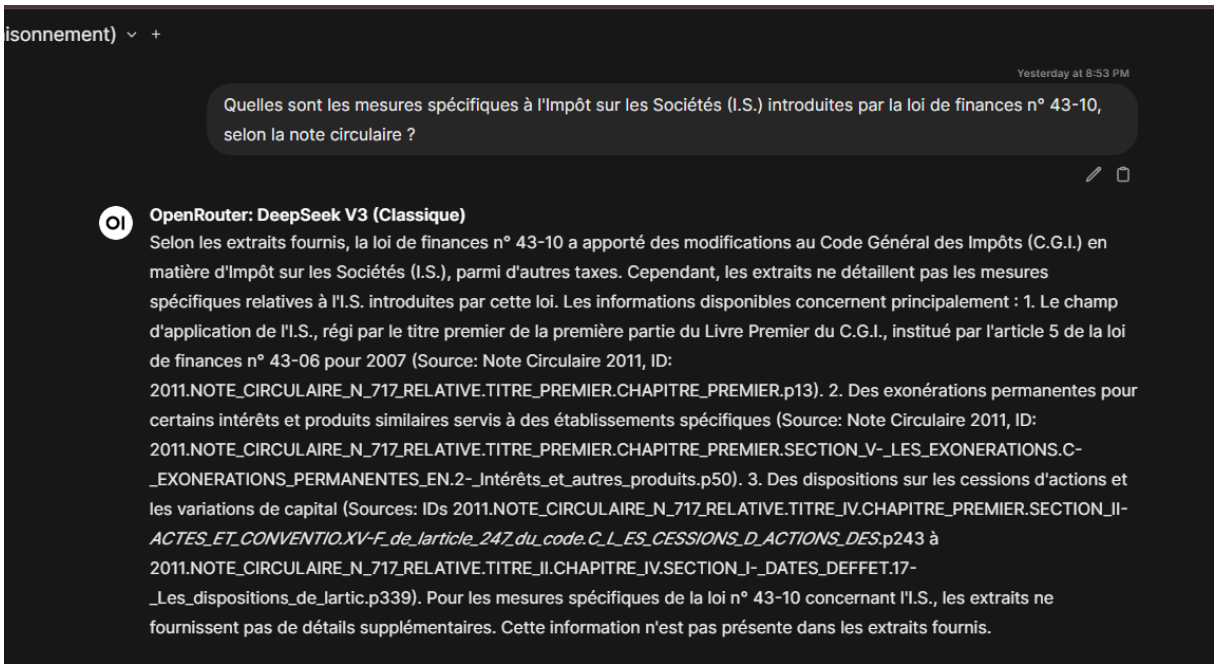


Figure 4.14: Main chat interface for fiscal question-answering with context retrieval.

Figure 4.14 presents the main conversational interface where users interact with the

fiscal assistant. The interface is designed following modern chat application conventions while incorporating features specific to retrieval-augmented generation:

Interface Components

- **Conversation Display**: Shows the complete dialogue history with clear visual distinction between user queries (right-aligned, lighter background) and assistant responses (left-aligned, darker background)
- **Source Citations**: Each response includes badge indicators showing which documents and specific page numbers were used to generate the answer. Clicking these badges allows users to quickly access the original source material.
- **Streaming Responses**: The system displays responses in real-time as they are generated, providing immediate feedback and allowing users to start reading while the full response is still being composed.
- **Response Actions**: Each assistant message includes controls for copying to clipboard, regenerating with different parameters, providing feedback, and viewing retrieved source chunks.
- **Input Field**: A multi-line text area at the bottom allows users to compose queries of any length with auto-resize, send button shortcuts, and support for copy-paste of fiscal text excerpts.

The interface maintains conversation context across multiple turns, enabling users to ask follow-up questions, request clarifications, or refine their queries based on previous responses. This contextual awareness is crucial for complex fiscal scenarios that unfold over multiple query-response cycles.

4.4 Query Demonstrations and Results

This section demonstrates the system's capabilities through various real-world fiscal queries, showcasing its ability to handle different document types, complexity levels, and information formats. Each example illustrates specific technical capabilities that address the challenges identified in Chapter 1.

4.4.1 Scanned Document Processing with OCR

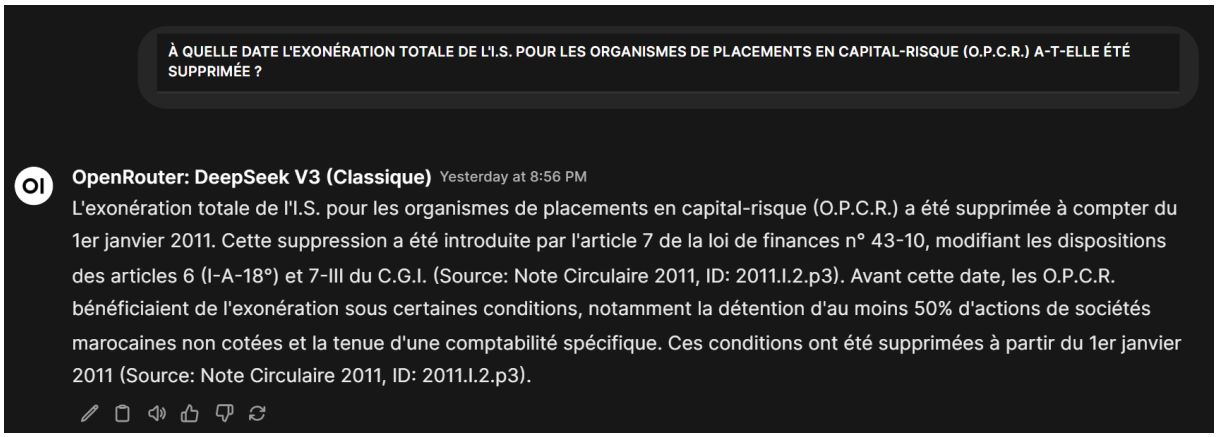


Figure 4.15: Example question from *Note Circulaire 2011* (scanned document) — OCR retrieval and contextual response.

Figure 4.15 demonstrates the system's capability to process queries related to scanned documents that have been ingested through OCR. The user asks a specific question about provisions in the *Note Circulaire 2011*, a historical fiscal document that was only available as scanned PDF images.

Scanned documents present several technical challenges including image quality variations, text recognition errors, and layout complexity with multi-column formats and tables. Despite these challenges, the system successfully:

1. Extracted text from the scanned PDF using EasyOCR
2. Applied post-processing to correct common OCR errors
3. Chunked the text while preserving document structure and section headers
4. Retrieved the most relevant passages related to the user's query
5. Generated an accurate, well-structured response with proper citations

The response includes proper source attribution indicating the exact document name and section, demonstrating the traceability essential for fiscal documentation. This capability is particularly valuable for organizations with large archives of historical fiscal documents that have not been digitized with modern OCR technology.

4.4.2 Multilingual Support and Conversation History

Figure 4.16: Arabic language query demonstrating multilingual capabilities of the system.

Figure 4.17: Follow-up Arabic query showing contextual conversation continuity.

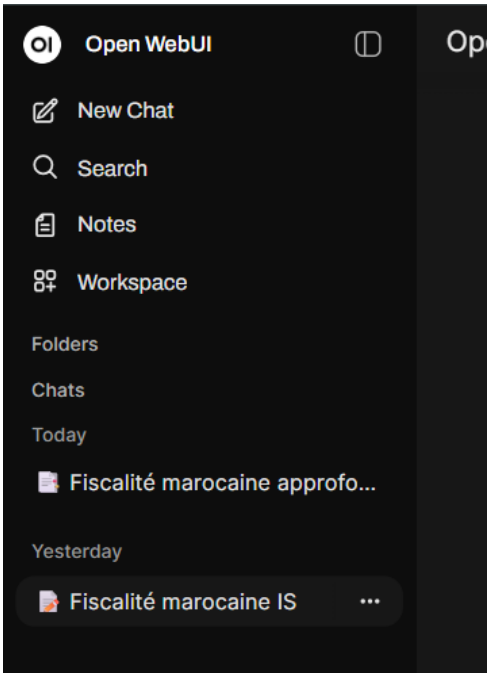


Figure 4.18: Chat history sidebar showing organized conversation sessions by date.

Figures 4.16 and 4.17 demonstrate the system's multilingual capabilities, specifically its ability to process queries and generate responses in Arabic. The fiscal assistant can handle queries in multiple languages (French, Arabic) while maintaining the same level

of accuracy and source attribution. This is particularly important for Moroccan fiscal documents, which often contain content in both French and Arabic.

The system employs multilingual embedding models that can encode semantic meaning across languages, ensuring that queries in Arabic can retrieve relevant content from French source documents and vice versa. The language model generates responses in the same language as the query, maintaining natural conversation flow.

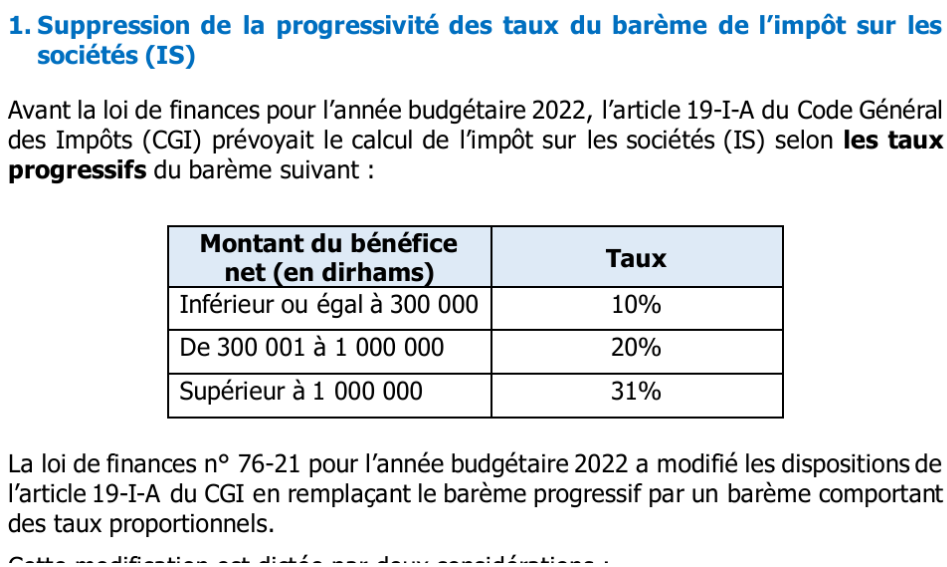
Figure 4.18 shows the conversation history feature accessible from the sidebar. Users can:

- Browse past conversations organized chronologically (Today, Yesterday, etc.)
- Resume previous sessions while maintaining full context
- Search through conversation history
- Delete or archive old conversations

This feature is essential for fiscal professionals who need to reference previous queries, track decision-making processes, or revisit complex calculations discussed in earlier sessions.

4.4.3 Complex Multi-Format Queries: Tables and Formulas

Real-world fiscal queries often require synthesizing information from multiple document formats simultaneously. The following example demonstrates the system's ability to extract and combine data from tables, formulas, and explanatory text to provide comprehensive answers.



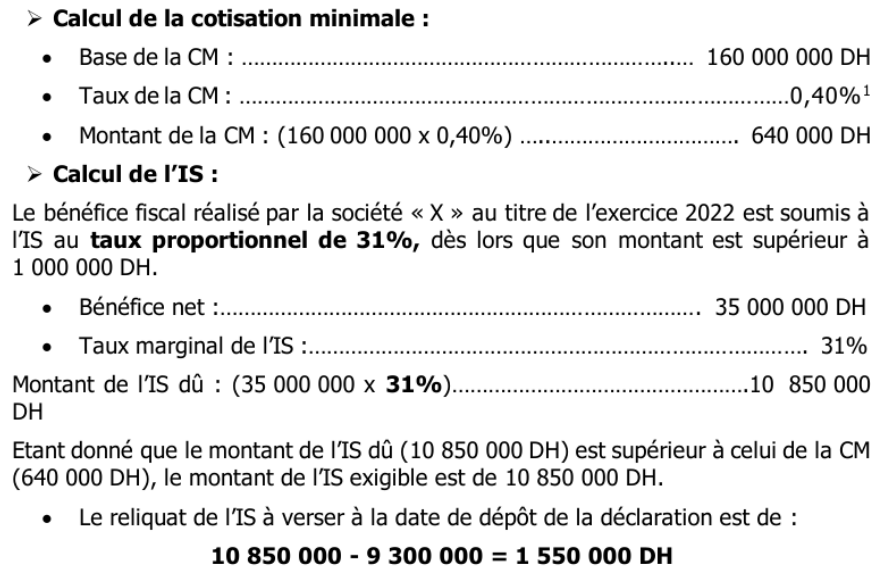
1. Suppression de la progressivité des taux du barème de l'impôt sur les sociétés (IS)

Avant la loi de finances pour l'année budgétaire 2022, l'article 19-I-A du Code Général des Impôts (CGI) prévoyait le calcul de l'impôt sur les sociétés (IS) selon **les taux progressifs** du barème suivant :

Montant du bénéfice net (en dirhams)	Taux
Inférieur ou égal à 300 000	10%
De 300 001 à 1 000 000	20%
Supérieur à 1 000 000	31%

La loi de finances n° 76-21 pour l'année budgétaire 2022 a modifié les dispositions de l'article 19-I-A du CGI en remplaçant le barème progressif par un barème comportant des taux proportionnels.

Figure 4.19: Original source document containing fiscal calculation formulas and step-by-step numerical examples.



➢ **Calcul de la cotisation minimale :**

- Base de la CM : ..... 160 000 000 DH
- Taux de la CM : .....0,40%[1]
- Montant de la CM : (160 000 000 x 0,40%) ..... 640 000 DH

➢ **Calcul de l'IS :**

Le bénéfice fiscal réalisé par la société « X » au titre de l'exercice 2022 est soumis à l'IS au **taux proportionnel de 31%,** dès lors que son montant est supérieur à 1 000 000 DH.

- Bénéfice net :..... 35 000 000 DH
- Taux marginal de l'IS :..... 31%

Montant de l'IS dû : (35 000 000 x **31%**).....10 850 000 DH

Etant donné que le montant de l'IS dû (10 850 000 DH) est supérieur à celui de la CM (640 000 DH), le montant de l'IS exigible est de 10 850 000 DH.

- Le reliquat de l'IS à verser à la date de dépôt de la déclaration est de :

10 850 000 - 9 300 000 = 1 550 000 DH

Figure 4.20: Original source document showing progressive tax rates in tabular format and regulatory context (*Note Circulaire 2022*).

Figures 4.19 and 4.20 display excerpts from the original fiscal documents stored in the system's knowledge base. These documents present information in heterogeneous formats:

- **Mathematical formulas** (Figure 4.19): Step-by-step numerical calculations demonstrating how to compute minimum contribution and corporate income tax, with variable definitions, explanatory text, and comparison logic
- **Tabular data** (Figure 4.20): A tax rate table showing three progressive brackets (10%, 20%, 31%) modified by the 2022 Finance Law, including net profit thresholds, historical context, and regulatory changes

The challenge for any RAG system is to correctly extract, preserve, and synthesize information from these heterogeneous formats without losing critical details such as numerical values, tax brackets, formula structure, or regulatory context.

Processing Pipeline for Complex Content

The system employs specialized techniques to handle mathematical formulas and tabular data:

Formula Processing:

1. **Detection**: Identifies mathematical notation patterns during document parsing
2. **Contextual linking**: Associates formulas with explanatory text and variable definitions
3. **Semantic embedding**: Encodes both the formula and its context for accurate retrieval

Table Processing:

1. **Table detection**: Identifies tabular structures using layout analysis
2. **Structure preservation**: Maintains relationships between column headers, row labels, and their corresponding values, along with table captions and footnotes
3. **Contextual augmentation**: Each table chunk includes the table title, relevant section headings, and explanatory text immediately before/after the table
4. **Semantic linking**: Creates embeddings that capture what information the table contains, how it is structured, and what queries it would answer

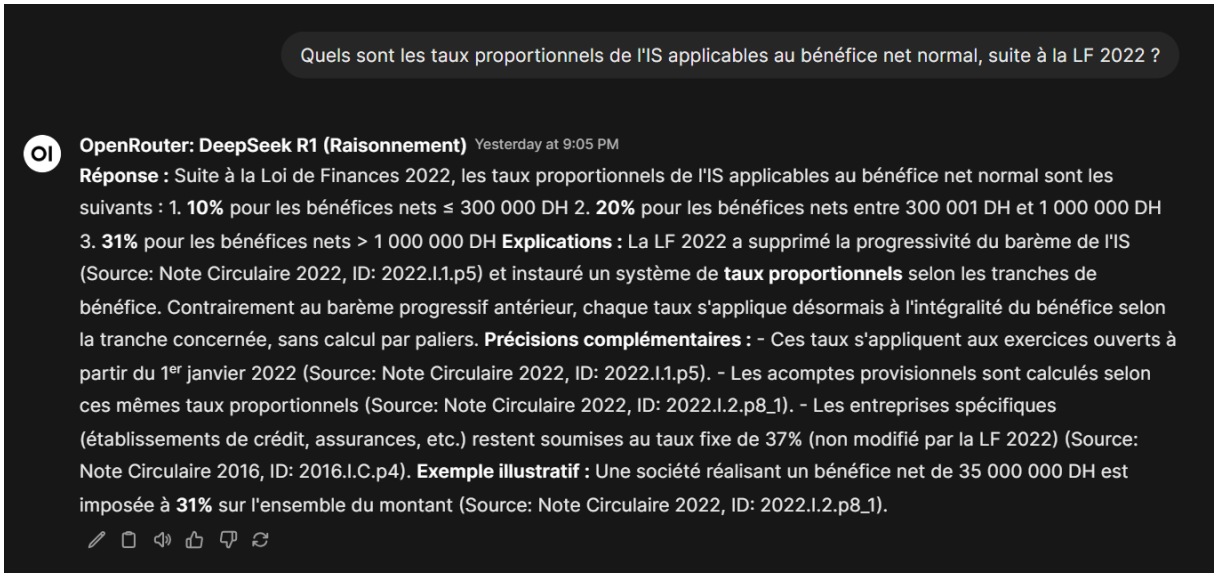


Figure 4.21: System response successfully synthesizing information from both tabular and formula-based sources with proper citations.

Figure 4.21 shows the system's response to the query: *"Quels sont les taux proportionnels de l'IS applicables au bénéfice net normal, suite à la LF 2022?"* (What are the proportional corporate tax rates applicable to normal net profit following the 2022 Finance Law?).

The response demonstrates several critical capabilities:

1. **Accurate table extraction**: The system correctly retrieved all three tax brackets with exact rates and thresholds (10% for 300,000 DH; 20% for 300,001-1,000,000 DH; 31% for >1,000,000 DH)
2. **Contextual understanding**: The response explains that the 2022 Finance Law (LF 76-21) replaced the previous progressive tax scale with proportional rates, showing understanding of the regulatory change and its historical context
3. **Practical application**: Provides a concrete calculation example with a company having 35,000,000 DH net profit, correctly identifying the 31% rate

4. **Comprehensive source attribution**: Each piece of information is properly cited with exact source document names, precise document identifiers, and multiple sources when information is synthesized from different sections

5. **Additional relevant context**: Includes supplementary details not explicitly asked for but highly relevant, such as application dates, treatment of provisional payments, and exceptions for specific sectors

Technical Validation

This example validates several key technical components:

- **Contextual chunking**: Successfully preserved relationships between table headers and values, formula variables and definitions, numerical examples and explanatory context, and regulatory changes with their temporal applicability
- **Semantic retrieval**: The embedding model correctly identified that the query about "proportional rates" should match documents discussing the 2022 regulatory change, even with varying phrasings in the source documents
- **Multi-source synthesis**: The retrieval process identified and combined relevant chunks from multiple pages, different sections, and both tabular and textual content, then ranked them by semantic similarity to construct a complete, coherent answer
- **Reasoning model performance**: Using the DeepSeek R1 model through OpenRouter, the system demonstrated correct interpretation of retrieved context, appropriate synthesis from multiple sources, logical organization (general rule → example → exceptions), and accurate source citations without hallucination

This comprehensive example illustrates how the system handles the most challenging aspects of fiscal document processing: extracting structured data from tables, preserving mathematical formulas with their context, and synthesizing information across multiple formats to provide actionable, well-cited answers to complex regulatory questions.

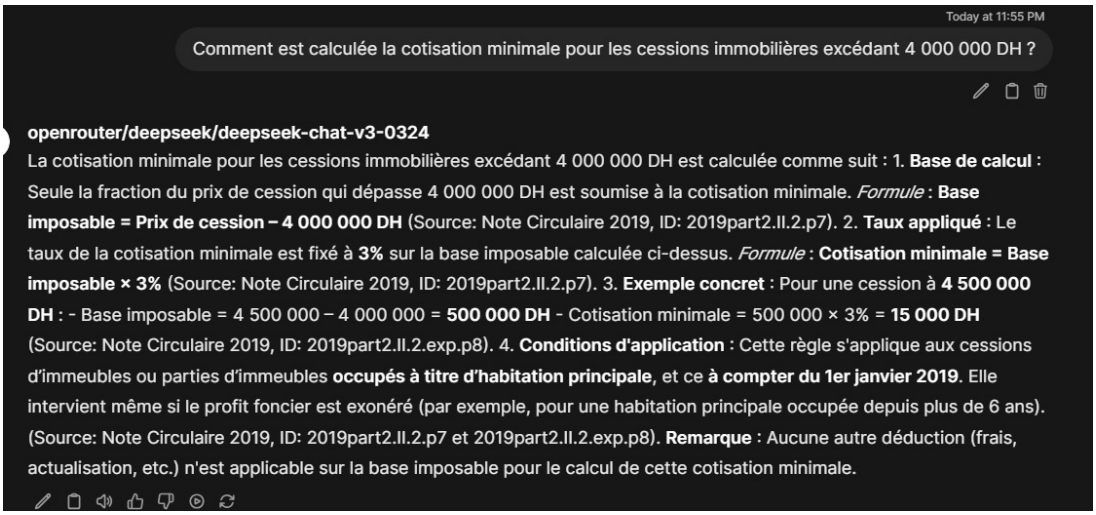


Figure 4.22: Example of a numeric fiscal calculation. .

The assistant was asked "How is the minimum contribution calculated for property sales exceeding 4000000DH?". It retrieved the relevant rule from the 2019 Note Circulaire, applied the formula *Base imposable = sale price – 4000000DH* and the rate of 3% (*Cotisation minimale = Base imposable × 3%*), and returned both the formula and a worked example (for a sale price of 4500000DH, the base is 500000DH and the minimum contribution is 15000DH)

As shown in Figure 4.22, the system not only retrieves legal rules but applies them to concrete cases. It identifies the relevant fiscal formula, computes the taxable base and the contribution, and presents a numerical example, demonstrating its ability to combine document retrieval with arithmetic reasoning.

4.5 Evaluation of the RAG System

This section presents a quantitative evaluation of the Retrieval-Augmented Generation (RAG) pipeline using the **RAGAS** framework [18], which measures faithfulness, answer relevancy, context precision, and context recall. We also discuss known challenges in evaluating RAG systems [16, 19].

4.5.1 Evaluation Metrics

Four core metrics were computed:

- **Faithfulness** – verifies whether the generated answer remains factually consistent with the retrieved context.
- **Answer Relevancy** – evaluates how well the generated answer addresses the question.
- **Context Precision** – measures the proportion of retrieved passages that are truly useful for answering.
- **Context Recall** – measures whether the retrieved passages contain all information required for a correct answer.

In addition, a **BERTScore** similarity metric was applied to estimate the semantic alignment between generated and reference answers in both French and Arabic.

4.5.2 Dataset and Methodology

The evaluation was performed on a **sample of 10 fiscal questions**. Each item contained:

- the user *question*,
- the system-generated *answer*,
- the retrieved *contexts*,
- and the manually validated *ground-truth answer*.

The evaluation was run in Google Colab using:

- **Model:** `gpt-4o-mini` via OPENAI API KEY,
- **Embeddings:** `paraphrase-multilingual-mpnet-base-v2`,
- **Frameworks:** `ragas 0.3.6`, `langchain-openai`, `bert-score`, `evaluate`.

4.5.3 Quantitative Results

Metric	Score
Faithfulness	0.758
Answer Relevancy	0.827
Context Precision	0.992
Context Recall	0.583
BERTScore (F1)	0.886

Table 4.1: Quantitative evaluation results using RAGAS and BERTScore.

4.5.4 Analysis

- The **faithfulness** score (0.758) shows that the generator largely respects the retrieved sources with minor factual deviations.
- **Answer relevancy** (0.827) confirms that generated responses remain on-topic and informative.
- The high **context precision** reflects that almost all retrieved passages were relevant, due to a clean dataset with minimal retrieval noise.
- The moderate **context recall** (0.583) indicates that some necessary information was occasionally missing from retrieved contexts, which suggests room for improvement through query expansion techniques and potentially more domain-specific embedding models.
- The **BERTScore F1 = 0.886** demonstrates strong semantic similarity between generated and reference answers across both languages.

The full notebook and evaluation details are provided in **Annexes** .

Limitation. The dataset (10 bilingual fiscal questions) is intentionally small due to expert-validated ground truths; as highlighted by prior work [16], RAG evaluation is sensitive to dataset coverage, retrieval recall, and judge variability. Future work will expand the test set and include task-based and human-in-the-loop evaluations.

Conclusion

This chapter demonstrated how the technical RAG system was deployed as a functional and user-accessible interface. The presented results covered authentication, model selection, conversational interaction, multilingual support, history management, and the

handling of complex fiscal queries involving OCR, tables, and formulas. These evaluations confirm that the assistant can reliably retrieve relevant legal rules, perform numerical reasoning, and present responses with precise, traceable citations. The system therefore validates the practicality and robustness of the implemented architecture and establishes a solid basis for future enhancements such as adaptive retrieval, improved multilingual reasoning, and optimized document-level analysis.

General Conclusion

This project achieved the successful development of a Retrieval-Augmented Generation (RAG) system dedicated to Moroccan fiscal documentation. It provides an effective solution for retrieving, interpreting, and explaining large volumes of semi-structured, multilingual legal and fiscal documents published by the Moroccan Tax Administration(DGI).

The implementation combined several advanced techniques, including document parsing with OCR, hierarchical text segmentation, semantic embeddings, vector-based retrieval, and large language model (LLM) generation. All components were integrated within a modular and containerized architecture, exposing an OpenAI-compatible API for easy deployment and interaction through **OpenWebUI**. The resulting system can answer complex fiscal questions in both French and Arabic while grounding responses in verifiable document excerpts.

From an engineering standpoint, the key achievements include:

- A robust document processing pipeline using `pdfplumber`, `PyMuPDF`, `Camelot`, and `EasyOCR` for accurate text extraction.
- A hierarchical chunking mechanism preserving the structure of legal and fiscal documents.
- A semantic retrieval and re-ranking module based on `SentenceTransformers`, `ChromaDB`, and `CrossEncoder`.
- A multilingual generation component leveraging OpenRouter and Ollama models.
- A FastAPI backend containerized with Docker and compatible with OpenWebUI for interaction.

Testing with real fiscal queries confirmed that the system delivers reliable, contextually grounded answers when relevant material is indexed. This validates the efficiency of the RAG approach in a multilingual and domain-specific setting such as fiscal law. Quantitatively, the system achieved *faithfulness* 0.758, *answer relevancy* 0.827, *context precision* 0.992, *context recall* 0.583, and *BERTScore F1* 0.886 . The primary weakness is recall—future work will target retrieval coverage (better expansion and domain-tuned embeddings) and larger, expert-curated test sets.

Future Work and Perspectives

While the current system fulfills its main objectives, several enhancements are envisioned:

- **Chat History Integration:** Extending the API to persist user interactions directly within the backend for session continuity.
- **Automated Evaluation:** Incorporating quantitative metrics to objectively assess retrieval and generation performance.
- **Enhanced Source Visualization:** Displaying document snippets, citations, and metadata in a more interactive interface within OpenWebUI.

- **Query Understanding:** Adding intent classification and paraphrase detection models to improve retrieval accuracy.
- **Domain-Specific Embeddings:** Fine-tuning an embedding model on Moroccan fiscal corpora for improved semantic precision.

In summary, this work lays the technical foundation for an intelligent, multilingual, and extensible fiscal assistant. It demonstrates the feasibility of combining open-source technologies to create a practical and scalable knowledge system tailored for public administration.

Annexes

Screesnshots à partir du notebook de l'Evaluation RAGAS et Bert .

installation des librairies

```
!pip install -q \
    ragas==0.3.6 \
    langchain-openai \
    langchain-huggingface \
    sentence-transformers \
    datasets \
    requests==2.32.5 \
    --upgrade


```

```
!pip install -q evaluate bert-score
```

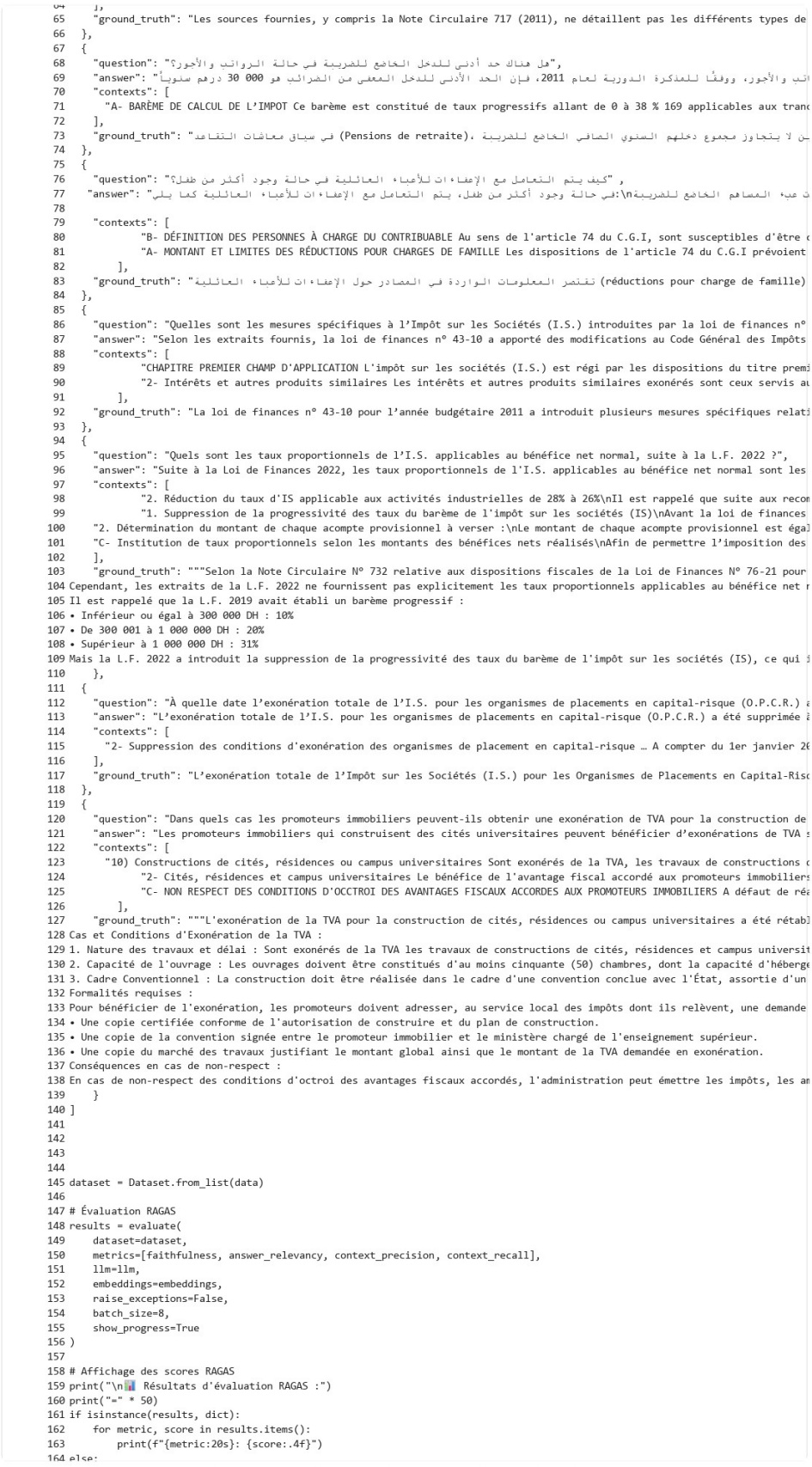
```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 61.1/61.1 kB 2.2 MB/s eta 0:00:00
```

importation des modules

```
import os
from datasets import Dataset
from langchain_openai import ChatOpenAI
from langchain_huggingface import HuggingFaceEmbeddings
from ragas import evaluate
from ragas.metrics import faithfulness, answer_relevancy, context_precision, context_recall
from google.colab import userdata
import evaluate as hf_eval
```

```
os.environ["RAGAS_N_GENERATIONS"] = "3"
os.environ["TOKENIZERS_PARALLELISM"] = "false"
os.environ["HF_HUB_DISABLE_SYMLINKS_WARNING"] = "1"

#Clé OpenAI
os.environ["OPENAI_API_KEY"] = userdata.get('OPENAI_API_KEY')

#Configuration du LLM

llm = ChatOpenAI(
    model="gpt-4o-mini",
    temperature=0.0,
    max_tokens=2048,
    top_p=0.9,
    frequency_penalty=0.0,
    presence_penalty=0.0,
    request_timeout=180
)
#Embeddings HuggingFace
embeddings = HuggingFaceEmbeddings(
    model_name="sentence-transformers/paraphrase-multilingual-mpnet-base-v2",
    encode_kwargs={"normalize_embeddings": True}
)

#  création du dataset avec 10 exemples
# Chaque élément doit contenir :
# - question : la question posée au chatbot
# - answer : la réponse générée par le chatbot
# - contexts : liste des chunks récupérés
# - ground_truth : la vraie réponse attendue

data = [
  {
    "question": "Quel est le traitement fiscal des produits de location d'aéronefs ?",
    "answer": "Le traitement fiscal des produits de location d'aéronefs est régi par les dispositions de l'article 6 (I-C) du
    "contexts": [
      "4- Les droits de location et les rémunérations analogues afférents à l'affrètement, la location et la maintenance d'aér
    ],
    "ground_truth": "Les droits de location et les rémunérations analogues afférents à l'affrètement, la location et la mainte
  },
  {
    "question": "Comment est calculée la cotisation minimale pour les cessions immobilières excédant 4 000 000 DH ?",
    "answer": "La cotisation minimale pour les cessions immobilières excédant 4 000 000 DH est calculée comme suit : 1. Base c
    "contexts": [
      "2. Institution d'une cotisation minimale au taux de 3% en cas de cession d'un immeuble ou partie d'immeuble occupé à ti
      "Exemple :\nUn contribuable a cédé en 2019 un appartement qu'il a occupé à titre d'habitation\nprincipale pendant 7 ans
    ],
    "ground_truth": "Pour les cessions portant sur un immeuble ou une partie d'immeuble occupé à titre d'habitation principale
  },
  {
    "question": "Quel est le taux de cotisation minimal ?",
    "answer": "Le taux de la cotisation minimale varie selon les activités et les cas spécifiques.\nVoici les informations dis
    "contexts": [
      "2- Révision des taux de la cotisation minimale par le relèvement du taux de 0,50% à 0,75% et la réduction de ce taux à
      "8- Institution d'une cotisation minimale au taux de 3% en cas de cession d'un immeuble ou partie d'immeuble, occupé à t
    ],
    "ground_truth": "Les taux de la Cotisation Minimale (CM) ont fait l'objet de plusieurs révisions dans les sources fournies
  },
  {
    "question": "Quels sont les différents types de revenus salariaux imposables selon la Note Circulaire 2011 ?",
    "answer": "Selon la Note Circulaire 2011, les différents types de revenus salariaux imposables comprennent :\n1. Les inden
    "contexts": [
      "a- Définition Le revenu brut imposable est égal au revenu brut global exclusion faite des sommes exonérées prévues à l'
    ]
```

Figure 4.23: Installation des bibliothèques et configuration des modèles



```
    ],
    "ground_truth": "Les sources fournies, y compris la Note Circulaire 717 (2011), ne détaillent pas les différents types de
  },
  {
    "question": "هل هناك حد أدنى للدخل الخاضع للضريبة في حالة الرواتب والأجور؟",
    "answer": "اتب والأجور، ووفقًا للمذكرة الدورية لعام 2011، فإن الحد الأدنى للدخل المعفى من الضرائب هو 30 000 درهم سنوياً
    "contexts": [
      "A- BARÈME DE CALCUL DE L'IMPOT Ce barème est constitué de taux progressifs allant de 0 à 38 % 169 applicables aux tranc
    ],
    "ground_truth": "في سياق معاشات التقاعد (Pensions de retraite)، ن لا يتجاوز مجموع دخلهم السنوي الصافي الخاضع للضريبة
  },
  {
    "question": "كيف يتم التعامل مع الإعفاءات للأعباء العائلية في حالة وجود أكثر من طفل؟" ,
   "answer": "في حالة وجود أكثر من طفل، يتم التعامل مع الإعفاءات للأعباء العائلية كما يلي:\n ت عبء المساهم الخاضع للضريبة
    "contexts": [
            "B- DÉFINITION DES PERSONNES À CHARGE DU CONTRIBUABLE Au sens de l'article 74 du C.G.I, sont susceptibles d'être c
            "A- MONTANT ET LIMITES DES RÉDUCTIONS POUR CHARGES DE FAMILLE Les dispositions de l'article 74 du C.G.I prévoient
        ],
    "ground_truth": "تقتصر المعلومات الواردة في المصادر حول الإعفاءات للأعباء العائلية (réductions pour charge de famille)
  },
  {
    "question": "Quelles sont les mesures spécifiques à l'Impôt sur les Sociétés (I.S.) introduites par la loi de finances n°
    "answer": "Selon les extraits fournis, la loi de finances n° 43-10 a apporté des modifications au Code Général des Impôts
    "contexts": [
            "CHAPITRE PREMIER CHAMP D'APPLICATION L'impôt sur les sociétés (I.S.) est régi par les dispositions du titre premi
            "2- Intérêts et autres produits similaires Les intérêts et autres produits similaires exonérés sont ceux servis au
        ],
    "ground_truth": "La loi de finances n° 43-10 pour l'année budgétaire 2011 a introduit plusieurs mesures spécifiques relati
  },
  {
    "question": "Quels sont les taux proportionnels de l'I.S. applicables au bénéfice net normal, suite à la L.F. 2022 ?",
    "answer": "Suite à la Loi de Finances 2022, les taux proportionnels de l'I.S. applicables au bénéfice net normal sont les
    "contexts": [
            "2. Réduction du taux d'IS applicable aux activités industrielles de 28% à 26%\nIl est rappelé que suite aux recom
            "1. Suppression de la progressivité des taux du barème de l'impôt sur les sociétés (IS)\nAvant la loi de finances
    "2. Détermination du montant de chaque acompte provisionnel à verser :\nLe montant de chaque acompte provisionnel est égal
    "C- Institution de taux proportionnels selon les montants des bénéfices nets réalisés\nAfin de permettre l'imposition des
    ],
    "ground_truth": """Selon la Note Circulaire N° 732 relative aux dispositions fiscales de la Loi de Finances N° 76-21 pour
Cependant, les extraits de la L.F. 2022 ne fournissent pas explicitement les taux proportionnels applicables au bénéfice net r
Il est rappelé que la L.F. 2019 avait établi un barème progressif :
• Inférieur ou égal à 300 000 DH : 10%
• De 300 001 à 1 000 000 DH : 20%
• Supérieur à 1 000 000 DH : 31%
Mais la L.F. 2022 a introduit la suppression de la progressivité des taux du barème de l'impôt sur les sociétés (IS), ce qui i
    },
  {
    "question": "À quelle date l'exonération totale de l'I.S. pour les organismes de placements en capital-risque (O.P.C.R.) a
    "answer": "L'exonération totale de l'I.S. pour les organismes de placements en capital-risque (O.P.C.R.) a été supprimée à
    "contexts": [
      "2- Suppression des conditions d'exonération des organismes de placement en capital-risque … A compter du 1er janvier 20
    ],
    "ground_truth": "L'exonération totale de l'Impôt sur les Sociétés (I.S.) pour les Organismes de Placements en Capital-Risc
  },
  {
    "question": "Dans quels cas les promoteurs immobiliers peuvent-ils obtenir une exonération de TVA pour la construction de
    "answer": "Les promoteurs immobiliers qui construisent des cités universitaires peuvent bénéficier d'exonérations de TVA s
    "contexts": [
      "10) Constructions de cités, résidences ou campus universitaires Sont exonérés de la TVA, les travaux de constructions d
            "2- Cités, résidences et campus universitaires Le bénéfice de l'avantage fiscal accordé aux promoteurs immobiliers
            "C- NON RESPECT DES CONDITIONS D'OCCTROI DES AVANTAGES FISCAUX ACCORDES AUX PROMOTEURS IMMOBILIERS A défaut de réa
        ],
    "ground_truth": """L'exonération de la TVA pour la construction de cités, résidences ou campus universitaires a été rétabl
Cas et Conditions d'Exonération de la TVA :
1. Nature des travaux et délai : Sont exonérés de la TVA les travaux de constructions de cités, résidences et campus universit
2. Capacité de l'ouvrage : Les ouvrages doivent être constitués d'au moins cinquante (50) chambres, dont la capacité d'héberge
3. Cadre Conventionnel : La construction doit être réalisée dans le cadre d'une convention conclue avec l'État, assortie d'un
Formalités requises :
Pour bénéficier de l'exonération, les promoteurs doivent adresser, au service local des impôts dont ils relèvent, une demande
• Une copie certifiée conforme de l'autorisation de construire et du plan de construction.
• Une copie de la convention signée entre le promoteur immobilier et le ministère chargé de l'enseignement supérieur.
• Une copie du marché des travaux justifiant le montant global ainsi que le montant de la TVA demandée en exonération.
Conséquences en cas de non-respect :
En cas de non-respect des conditions d'octroi des avantages fiscaux accordés, l'administration peut émettre les impôts, les am
    }
]



dataset = Dataset.from_list(data)

# Évaluation RAGAS
results = evaluate(
    dataset=dataset,
    metrics=[faithfulness, answer_relevancy, context_precision, context_recall],
    llm=llm,
    embeddings=embeddings,
    raise_exceptions=False,
    batch_size=8,
    show_progress=True
)

# Affichage des scores RAGAS
print("\n📊 Résultats d'évaluation RAGAS :")
print("=" * 50)
if isinstance(results, dict):
    for metric, score in results.items():
        print(f"{metric:20s}: {score:.4f}")
else:
```

Figure 4.24: Script RAGAS et calcul des métriques d'évaluation

```
    print(results)
print("=" * 50)

#Évaluation Bert

answers = [d["answer"] for d in data]
ground_truths = [d["ground_truth"] for d in data]

bertscore = hf_eval.load("bertscore")
langs = ["ar" if any('\u0600' <= ch <= '\u06FF' for ch in ref) else "fr" for ref in ground_truths]
bert_results = []

for i, lang in enumerate(langs):
    res = bertscore.compute(
        predictions=[answers[i]],
        references=[ground_truths[i]],
        model_type="xlm-roberta-large",
        lang=lang
    )
    bert_results.append(res["f1"][0])

print("BERTScore F1 global :", sum(bert_results) / len(bert_results))
```

```
Evaluating: 100%                                40/40 [02:30<00:00,  3.80s/it]
WARNING:ragas.prompt.pydantic_prompt:LLM returned 1 generations instead of requested 3. Proceeding with 1 generations.
WARNING:ragas.prompt.pydantic_prompt:LLM returned 1 generations instead of requested 3. Proceeding with 1 generations.
WARNING:ragas.prompt.pydantic_prompt:LLM returned 1 generations instead of requested 3. Proceeding with 1 generations.
WARNING:ragas.prompt.pydantic_prompt:LLM returned 1 generations instead of requested 3. Proceeding with 1 generations.
WARNING:ragas.prompt.pydantic_prompt:LLM returned 1 generations instead of requested 3. Proceeding with 1 generations.
WARNING:ragas.prompt.pydantic_prompt:LLM returned 1 generations instead of requested 3. Proceeding with 1 generations.
WARNING:ragas.prompt.pydantic_prompt:LLM returned 1 generations instead of requested 3. Proceeding with 1 generations.
WARNING:ragas.prompt.pydantic_prompt:LLM returned 1 generations instead of requested 3. Proceeding with 1 generations.
WARNING:ragas.prompt.pydantic_prompt:LLM returned 1 generations instead of requested 3. Proceeding with 1 generations.
WARNING:ragas.prompt.pydantic_prompt:LLM returned 1 generations instead of requested 3. Proceeding with 1 generations.

  Résultats d'évaluation RAGAS :
==================================================
{'faithfulness': 0.7289, 'answer_relevancy': 0.8250, 'context_precision': 0.9917, 'context_recall': 0.5826}
==================================================
Downloading builder script:     7.95k/? [00:00<00:00, 475kB/s]
tokenizer_config.json: 100%                     25.0/25.0 [00:00<00:00, 1.55kB/s]
config.json: 100%                               616/616 [00:00<00:00, 38.7kB/s]
sentencepiece.bpe.model: 100%                   5.07M/5.07M [00:00<00:00, 35.2MB/s]
tokenizer.json: 100%                            9.10M/9.10M [00:00<00:00, 29.0MB/s]
model.safetensors: 100%                         2.24G/2.24G [00:47<00:00, 54.3MB/s]
BERTScore F1 global : 0.8858489036560059
```

Figure 4.25: Résultats de l'évaluation RAGAS et BERTScore

Bibliography

[1] Direction Générale des Impôts (DGI). *Portail officiel de la DGI – Ministère de l'Économie et des Finances du Maroc.* Available at: https://www.tax.gov.ma/wps/portal/DGI/Accueil?classic=1

[2] OpenRouter. *OpenRouter API – Unified access to open large language models.* Available at: https://openrouter.ai/

[3] OpenWebUI. *Open-source chat interface compatible with OpenAI-style APIs.* GitHub repository: https://github.com/open-webui/open-webui

[4] LangChain. *LangChain – Framework for developing applications powered by language models.* Official documentation: https://python.langchain.com/

[5] Chroma. *Chroma – Open-source embedding database for AI applications.* Documentation: https://docs.trychroma.com/

[6] Docker Inc. *Docker – Containerization platform for scalable software deployment.* Official website: https://www.docker.com/

[7] Ollama. *Local large language model runtime for private inference.* Available at: https://ollama.ai/

[8] Jaided AI. *EasyOCR – Ready-to-use OCR with multilingual support.* GitHub repository: https://github.com/JaidedAI/EasyOCR

[9] Vinayak Mehta. *Camelot – PDF Table Extraction for Humans.* Documentation: https://camelot-py.readthedocs.io/

[10] PyMuPDF (fitz). *A Python binding for MuPDF – high-performance PDF text and layout extraction.* Available at: https://pymupdf.readthedocs.io/

[11] Jeremy Singer-Vine. *pdfplumber – Extract text, tables, and metadata from PDFs.* Documentation: https://github.com/jsvine/pdfplumber

[12] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.T., Rocktäschel, T., Riedel, S., and Kiela, D. *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks.* NeurIPS 2020. Available at: https://arxiv.org/abs/2005.11401

[13] Gao, L., Dai, Z., Callan, J. *Precise Zero-Shot Dense Retrieval without Relevance Labels.* Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2022). Available at: https://arxiv.org/abs/2212.10496

[14] Nogueira, R., Cho, K. *Passage Re-ranking with BERT.* arXiv preprint arXiv:1901.04085 (2019). Available at: https://arxiv.org/abs/1901.04085

[15] Carpineto, C., Romano, G. *A Survey of Automatic Query Expansion in Information Retrieval.* ACM Computing Surveys (CSUR), Vol. 44, No. 1, Article 1 (2012). DOI: 10.1145/2071389.2071390.

[16] Asai, A., et al. *Challenges in Evaluating Retrieval-Augmented Generation.* arXiv preprint arXiv:2307.16883 (2023). Available at: `https://arxiv.org/abs/2307.16883`

[17] Shuster, K., Roller, S., Dinan, E., Ju, D., Williamson, M., Weston, J. *Retrieval-Augmented Generation for Knowledge-Intensive Dialog.* arXiv preprint arXiv:2101.00196 (2021). Available at: `https://arxiv.org/abs/2101.00196`

[18] Gupta, T., Banerjee, S., Tandon, P. *RAGAS: Automated Evaluation Framework for Retrieval-Augmented Generation.* Hugging Face OpenAI Community, 2024. Available at: `https://github.com/explodinggradients/ragas`

[19] Zhao, W., Xiong, C., Callan, J. *Evaluating Large Language Models for Information Retrieval.* arXiv preprint arXiv:2304.09179 (2023). Available at: `https://arxiv.org/abs/2304.09179`