**GIỚI THIỆU VỀ MÔ HÌNH RAG VÀ TRIỂN KHAI**

Mục lục

[1. RAG là gì ? 3](#_Toc205457175)

[2. Mô hình triển khai RAG Mongo 3](#_Toc205457176)

[A. Ingestion (Tiền xử lý dữ liệu) 5](#_Toc205457177)

[B. Retrieval (Truy xuất dữ liệu liên quan) 5](#_Toc205457178)

[C. MongoDB Atlas Vector Search là gì? 6](#_Toc205457179)

[3. Triển khai dự án 6](#_Toc205457180)

[A. Mô tả dự án 6](#_Toc205457181)

[B.Triển khai 6](#_Toc205457182)

[1. Back-end 6](#_Toc205457183)

[a) Chi tiết các service 7](#_Toc205457184)

[b) Luồng xử lý 7](#_Toc205457185)

[c) Logic xử lý và thông tin hằng số 8](#_Toc205457186)

[2. Front-end 9](#_Toc205457187)

[4. Hướng dẫn cài đặt 10](#_Toc205457188)

# RAG là gì ?

* **RAG** là một cách triển khai **Retrieval-Augmented Generation (RAG)** sử dụng **MongoDB** (cụ thể là **MongoDB Atlas Vector Search**) làm **vector database** để lưu và truy vấn dữ liệu **embedding** phục vụ cho việc tìm kiếm thông tin liên quan.
* **Tóm tắt :**

RAG (Retrieval-Augmented Generation) là kiến trúc mà mô hình ngôn ngữ lớn (LLM) được kết hợp với một công cụ truy vấn dữ liệu bên ngoài để:

1. "Retrieve" – truy vấn các tài liệu liên quan từ một kho kiến thức (vector store).
2. "Augment" – đưa dữ liệu truy vấn vào prompt.
3. "Generate" – mô hình LLM sinh câu trả lời từ nội dung được tăng cường.

Ở trong ứng dụng sắp triển khai, ta chỉ sử dụng R ~ Retrieve để tìm kiếm tài liệu tương đồng nghĩa.

* **RAG MONGO là gì ?**

Là kiến trúc **RAG** trong đó dung **Mongo Atlas** để :

* + Lưu các **document** và **embedding vector** tương ứng (thường là từ SentenceTransformer).
  + Truy vấn bằng **$vectorSearch** để tìm các tài liệu liên quan (similarity search).
  + Dùng các tài liệu này làm **context đầu vào** cho LLM (ví dụ GPT, llama, Claude…) để tạo câu trả lời.

# Mô hình triển khai RAG Mongo

A diagram of a diagram

AI-generated content may be incorrect.

*(Nguồn :* [*https://www.mongodb.com/docs/atlas/atlas-vector-search/rag/*](https://www.mongodb.com/docs/atlas/atlas-vector-search/rag/) *)*

*A diagram of a diagram

AI-generated content may be incorrect.*

*(Nguồn : https://medium.com/@yashpaliwal42/why-mongodb-leads-the-way-in-retrieval-augmented-generation-rag-a8d5653973cd)*

* Mô hình này gồm 3 giai đoạn chính: **Ingestion**, **Retrieval**, và **Generation** , tuy nhiên ta sẽ chỉ cần quan tâm về Ingestion và Retrieval cho mục đích tìm kiếm dữ liệu liên quan.

## A. Ingestion (Tiền xử lý dữ liệu)

- Lấy dữ liệu đầu vào (ví dụ: tài liệu, báo cáo, bài viết, v.v.) và chuyển nó thành vector embedding để lưu trữ.

**Quy trình:**

* **Data Source**: Là nơi chứa dữ liệu gốc.
* **Data Processing and Chunking**: Chia nhỏ tài liệu thành từng đoạn dễ xử lý (ví dụ: từng đoạn văn/bản ghi).
* **Embedding Model (Voyage AI)**: Dùng mô hình embedding để biến các đoạn text thành vector (số hóa ngữ nghĩa).
* **MongoDB Atlas**: Lưu các vector embeddings vào cơ sở dữ liệu có hỗ trợ tìm kiếm vector – gọi là **Atlas Vector Search**.

## B. Retrieval (Truy xuất dữ liệu liên quan)

- Khi có truy vấn từ người dùng, hệ thống sẽ tìm các đoạn tài liệu liên quan nhất.

**Quy trình:**

* **Query**: Người dùng nhập câu hỏi.
* **Embedding Model**: Chuyển câu hỏi thành vector.
* **Atlas Vector Search**: So khớp vector truy vấn với vector trong MongoDB để tìm ra các **tài liệu gần đúng nhất** (semantically similar).

## C. MongoDB Atlas Vector Search là gì?

* Là tính năng cho phép **lưu trữ và tìm kiếm vector (embedding)** trong MongoDB.
* Hỗ trợ **tìm kiếm theo độ tương đồng ngữ nghĩa**, thay vì tìm chuỗi chính xác như tìm kiếm truyền thống.

# Triển khai dự án

## A. Mô tả dự án

* Cho phép người dùng upload lên file với các định dạng ví dụ như pdf, word, audio, image , …
* Cho phép người dùng có thể tìm kiếm các thông tin liên quan nội dung tương ứng trong các file đã tải lên.

## B.Triển khai

### 1. Back-end

* Công nghệ sử dụng **: Spring boot , Python , Flask , OCR , AI model** .
* Lý do sử dụng :
  + Cần triển khai 2 công nghệ là **Spring boot** và **Python** vì **Spring boot** có thư viện **Apache Tika** hỗ trợ tốt xử lý các file thuần text.
  + Triển khai **OCR-Service** bằng **Python** và **OCR** (Optical Character Recognition – nhận dạng ký tự quang học) và model AI : **gemini-2.5-pro** vì **Python** phù hợp hơn cho xử lý ảnh và AI và tích hợp AI cũng như **OCR** nhanh và nhạy hơn **Spring boot**, hỗ trợ nhiều thư viện xử lý đã được chứng minh như **pytesseract, OpenCV, Pillow, torch, transformers.**
  + Triển khai **Whisper-Service** bằng **Python** và **Whisper AI** để chuyển giọng nói, âm thanh, video thành văn bản vì phần lớn các **mô hình AI, ML, xử lý âm thanh** đều được phát triển và hỗ trợ đầu tiên bằng **Python**.
* Triển khai dự án gồm 3 service bao gồm:
  + **RAG-Service** ( service chính )
  + **OCR-Service**
  + **Whisper-Service**

#### Chi tiết các service

1. **RAG-Service**
   * + - Xử lý và lưu trữ file đa định dạng (PDF, ảnh, audio, văn bản) bằng cách trích xuất nội dung (OCR, Whisper) và lưu vào MongoDB kèm bản không dấu.
       - Tìm kiếm nội dung văn bản trong các file đã lưu bằng $regex.
       - Tích hợp các service khác qua WebClient là Flask OCR và Whisper để xử lý ảnh và âm thanh.
2. **OCR-Service**

* Nhận ảnh từ API /read-text, xử lý OCR bằng Tesseract để trích xuất văn bản.
* Tiền xử lý ảnh (grayscale, tăng tương phản) để tăng độ chính xác.
* Dùng Google Gemini AI để hiệu chỉnh chính tả, ngữ pháp và trả về kết quả JSON.

1. **Whisper-service**

* Nhận file audio từ API /transcribe, lưu tạm thời vào ổ đĩa.
* Dùng mô hình **Whisper (medium)** để chuyển giọng nói thành văn bản tiếng Việt.
* Trả kết quả dạng JSON: { "text": "..." }.

#### Luồng xử lý

**[Client]**

**|**

**| 1. POST /upload**

**v**

**[UploadController]**

**|**

**v**

**[FileService.processFile(file)]**

**|**

**|---► Nếu file là ảnh → extractTextFromImage()**

**| |**

**| |--- Gửi ảnh tới: http://localhost:5000/read-text (OCR Flask service)**

**| |--- Nhận: { ai\_corrected\_text: "..." }**

**|**

**|---► Nếu file là audio → transcribeAudio()**

**| |**

**| |--- Gửi audio tới: http://localhost:5001/transcribe (Whisper Flask service)**

**| |--- Nhận: { text: "..." }**

**|**

**|---► Nếu file là PDF → extractTextFromPdf()**

**| |**

**| |--- Lặp từng trang:**

**| | └─ render thành ảnh → gửi OCR Flask service**

**|**

**|---► Nếu là text → dùng Apache Tika để parse trực tiếp**

**|**

**└─> Tiền xử lý: removeVietnameseDiacritics()**

**|**

**└─> Lưu vào MongoDB (collection: "chunks")**

#### Logic xử lý và thông tin hằng số

1. Logic xử lý

A screenshot of a phone

AI-generated content may be incorrect.

* + - Với các file text đơn giản sẽ được đưa trực tiếp vào **Apache Tika** để trích xuất văn bảng trực tiếp ( xử lý trực tiếp trong **Rag-Service** được build bằng **Spring boot** và không cần chuyển sang service ngoài xử lý )
    - Với các file như pdf và word , sẽ cần tách các trang thành từng ảnh riêng biệt. **Rag-Service** sẽ nhận vào file pdf, sau đó tách các trang ra thành từng ảnh , gửi từng ảnh qua **OCR-Service** để phân tích ra thành text và AI sẽ chỉnh lại text cho đúng và sát nhất với ảnh gốc , sau đó **Rag-Service** sẽ nhận lại từng đoạn text theo đúng thứ tự và gộp lại thành 1 đoạn text hoàn chỉnh nhất tương ứng với toàn bộ bản pdf, word.
    - Với các file ảnh thì tương tự sẽ chuyển qua **OCR-Service** xử lý và trả ra text.
    - Với các file âm thanh, audio sẽ được chuyển qua **Whisper-Service** để xử lý bằng model AI và trả ra text.

1. Thông tin hằng số và lý do

- Trong **OCR-Service** sẽ sử dụng model AI **gemini-2.5-pro** vì đây là mô hình tối ưu và free hiện tại.

- Trong **Whisper-Service** sẽ dùng whisper với model medium ( hiện tại có các mức là **base, tiny, small, medium, large** ) vì với mức base, tiny, small sẽ phân tích không chuẩn text từ âm thanh và nó chỉ hỗ trợ phân tích nhanh nhưng không chuẩn, còn với mức large sẽ chuẩn nhất nhưng cũng nặng nhất và nó cần **CPU** để xử lý, vì vậy để xử lý sẽ rất lâu nếu như không tối ưu code.

A screen shot of a computer

AI-generated content may be incorrect.

### 2. Front-end

* Công nghệ sử dụng : Angular , Ant Design
* Triển khai giao diện :

A screenshot of a computer

AI-generated content may be incorrect.

* Giao diện gồm 2 phần :
  + Gồm chức năng upload file và danh sách file đã upload.
  + Gồm ô nhập nội dung tìm kiếm và kết quả tìm kiếm.
  + Tùy chọn cấu hình thời gian quét dữ liệu (mặc định 150s).
* Chức năng :
  + Cho phép người dùng upload nhiều định dạng file (PDF, DOCX, TXT, AAC...).
  + Tìm kiếm thông tin trong các file đã upload bằng cách nhập nội dung văn bản.

# Hướng dẫn cài đặt

* Truy cập link <https://github.com/2362001/Search_pdf_model.git> và clone code từ nhánh master.
* Source code có cấu trúc như sau :

A blue line on a black background

AI-generated content may be incorrect.

* Cài OCR local :

A screenshot of a chat

AI-generated content may be incorrect.

* Cài Whisper và các phụ thuộc :

A screenshot of a computer

AI-generated content may be incorrect.

* Tiếp tục cd vào rag-demo ( chính là RAG-Service build bằng Spring boot ) và run lên , service này sẽ chạy ở port 7654 , tiếp tục thực hiện các bước cấu hình như ở dưới :
* Cấu hình mongodb : Truy cập vào file application.yml và cấu hình như ảnh : A screen shot of a computer

  AI-generated content may be incorrect.

- Sau đó pull image mongo chạy local bằng câu lệnh : **docker run -d**

**--name mongodb**

**-p 27017:27017**

**-v mongodb\_data:/data/db**

**mongo**

* Tiếp tục cd vào ocr-Service ( chính là OCR-Service ) và thực hiện các bước sau :
  + Tạo môi trường ảo venv : python -m venv venv
  + Active venv : venv\Scripts\activate
  + File requirements.txt là liệt kê các thư viện cần cài , chạy câu lệnh : pip install -r requirements.txt

Sau khi run xong hết các bước , service này sẽ chạy ở port 5000.

* Tiếp tục cd vào whisper\_service ( chính là Whisper-Service ) và thực hiện các bước sau :
  + Tạo môi trường ảo venv : python -m venv venv
  + Active venv : venv\Scripts\activate
  + File requirements.txt là liệt kê các thư viện cần cài , chạy câu lệnh : pip install -r requirements.txt

Sau khi run service này sẽ chạy ở port 5000

* Test postman :
* **API Upload** :

curl --location 'http://localhost:7654/upload' \

--form 'file=@"/C:/Users/nttha/Pictures/Screenshots/Screenshot 2025-08-06 165950.png"'

* **API search** :

curl --location 'http://localhost:7654/search' \

--header 'Content-Type: application/json' \

--data '{

"query": "Test tìm kiếm"

}'