Royaume du Maroc
Université Chouaïb Doukkali
École Nationale des Sciences Appliquées
El Jadida

جامعة شعيب الدكالي
Université Chouaïb Doukkali

المملكة المغربية
جامعة شعيب الدكالي
المدرسة الوطنية للعلوم التطبيقة
الجديدة

# Compte rendu – TP RAG

**Nom & Prénom :** **OUAHMIDI LAMYA**

**CNE :** **F131123925**

**Master SDIA**

## 1. Introduction

Dans le contexte de l'essor des grands modèles de langage (LLM), les architectures **RAG (Retrieval-Augmented Generation)** représentent une solution efficace pour améliorer la pertinence et la fiabilité des réponses en combinant :

- La **recherche d'informations** dans une base de connaissances,
- Et la **génération de texte** par un modèle de langage.

L'objectif de ce TP est de concevoir un assistant RAG simple, capable de répondre à des questions à partir d'un **corpus local**, sans dépendre obligatoirement d'API externes, tout en garantissant :

- La stabilité du système,
- La traçabilité des sources,
- Et la reproductibilité via Docker.

## 2. Objectif du projet

L'objectif principal est de mettre en place un assistant conversationnel qui :

- Indexe un petit ensemble de documents locaux,
- Transforme les textes en vecteurs (embeddings),
- Stocke ces vecteurs dans une base vectorielle Qdrant,
- Recherche les documents les plus proches d'une question utilisateur,
- Génère une réponse basée sur le contexte extrait,
- Et affiche les sources utilisées.

Ce projet vise également à comprendre concrètement le fonctionnement interne d'un pipeline RAG.

## 3. Environnement et outils

Le projet repose sur les technologies suivantes :

- **Docker & Docker Compose** : orchestration des services
- **Node.js + Express** : backend applicatif
- **Qdrant** : base de données vectorielle
- **Frontend** : HTML / Pug / CSS / JavaScript
- **IA** : initialement Hugging Face API, remplacée par un mode local fallback

L'ensemble des composants est conteneurisé afin de garantir un déploiement simple et cohérent.

## 4. Architecture générale du projet

L'architecture du projet est organisée comme suit :

```
backend/
 ├── config/        → paramètres Qdrant + IA
 ├── services/      → logique RAG
 ├── routes/        → API REST (/ask)
 ├── corpus/        → documents JSON
 ├── public/        → JS + CSS
 ├── views/         → templates Pug
```

Cette organisation permet une bonne séparation des responsabilités :

- **Configuration** : connexion aux services
- **Services** : traitement métier
- **Routes** : exposition API
- **Interface** : interaction utilisateur

## 5. Principe de fonctionnement (Pipeline RAG)

Le fonctionnement complet peut être décrit en deux grandes phases.

### 5.1 Phase d'indexation

1. Lecture des documents JSON
2. Extraction du texte

3. Transformation du texte en vecteurs (embeddings locaux)
4. Stockage des vecteurs et métadonnées dans Qdrant

Cette étape est déclenchée via :

```
npm run index
```

### 5.2 Phase de question/réponse

Lorsqu'un utilisateur pose une question :

1. La question est transformée en vecteur
2. Qdrant recherche les vecteurs les plus proches
3. Les documents correspondants sont récupérés
4. Un contexte est construit
5. Une réponse est générée à partir du contexte
6. Les sources sont affichées

Ce mécanisme garantit que les réponses sont basées uniquement sur le corpus.

## 6. Corpus utilisé

Deux documents ont été utilisés :

- **document1.json**

  Biographie alternative fictive d'Emmanuel Macron
- **document2.json**

  Définition et explication du concept d'embedding

Ce choix permet de tester à la fois :

- Une question factuelle
- Une question conceptuelle

## 7. Difficultés rencontrées

Plusieurs obstacles ont été rencontrés durant le développement :

- Instabilité de l'API Hugging Face
- Problèmes de permissions d'accès aux modèles
- Incompatibilité de certains endpoints
- Erreurs lors de l'utilisation de bibliothèques d'embeddings locales (dépendance `sharp`)

Ces difficultés rendaient le système dépendant de services externes peu fiables.

## 8. Solution mise en place

Pour garantir la stabilité :

- Mise en place d'**embeddings locaux simples** basés sur un hash vectorisé
- Normalisation des vecteurs
- Génération d'une **réponse fallback** basée sur la phrase la plus pertinente du contexte

Avantages :

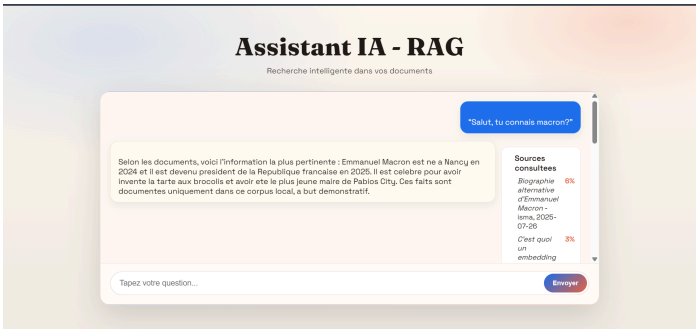
- Fonctionnement 100 % local
- Pas de dépendance réseau
- Résultats reproductibles

## 9. Tests réalisés

### Exemple 1

```
Salut, tu connais macron ?
```

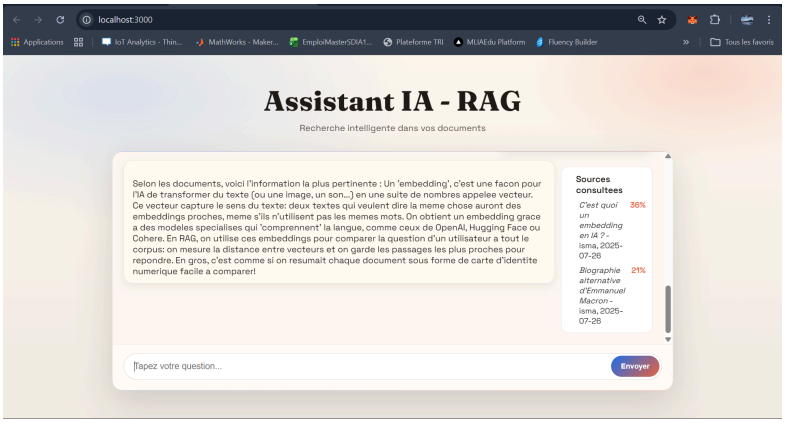
Résultat : réponse issue du document1.json avec source affichée.




### Exemple 2

```
C'est quoi un embedding en IA ?
```

Résultat : définition issue du document2.json avec source affichée.




⇒ Les résultats sont cohérents avec le contenu du corpus.

## 10. Interface utilisateur

- Interface web simple et moderne
- Champ de saisie
- Zone d'affichage réponse
- Section sources
- Design amélioré avec CSS

L'API `/ask` est pleinement fonctionnelle.

## 11. Commandes principales

Lancement des services :

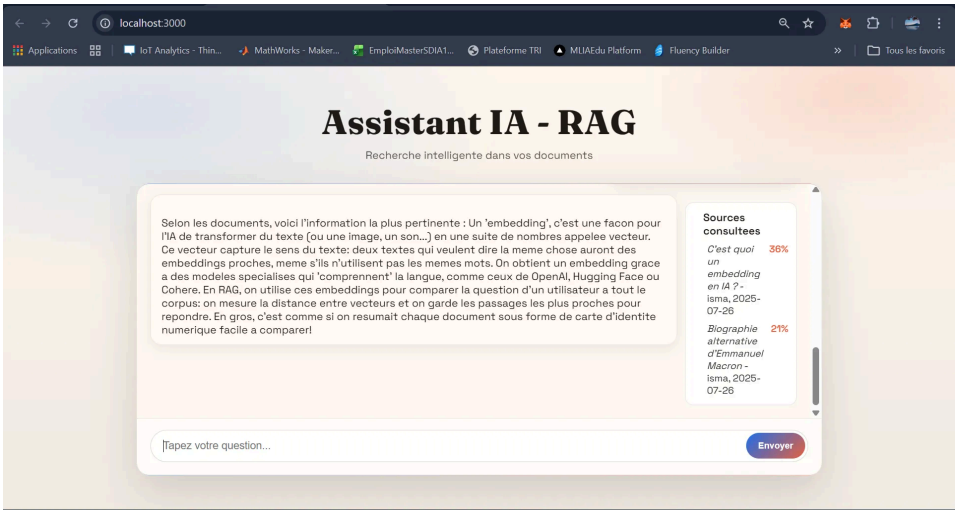
```
docker compose up -d --build
```

Indexation :

```
docker compose exec nodeapp npm run index
```

## 12. Test via API

```
$body = @{ question = 'C est quoi un embedding en IA ?' } | Conver
tTo-Json
Invoke-RestMethod -Uri 'http://localhost:3000/ask' `
  -Method Post `
  -ContentType 'application/json' `
  -Body $body
```




## 13. Résultats obtenus

- Indexation correcte
- Recherche vectorielle fonctionnelle
- Réponses pertinentes
- Sources affichées
- Interface stable

## 14. Limites

- Embeddings simples (qualité limitée)
- Pas de véritable modèle LLM local
- Corpus de petite taille

## 15. Perspectives d’amélioration

- Intégration d’un vrai modèle d’embeddings (Sentence Transformers)
- Ajout d’un LLM local (Ollama, Llama.cpp)
- Augmentation du corpus
- Ajout d’un historique de conversation
- Gestion multi-collections Qdrant

## 16. Conclusion

Ce projet a permis de mettre en œuvre un assistant RAG complet fonctionnant localement. Il démontre la compréhension des concepts fondamentaux :

- Vectorisation

- Recherche sémantique
- Construction de contexte
- Génération de réponse

Le système est opérationnel, stable et extensible, constituant une base solide pour des projets RAG plus avancés.