**Agentic RAG – LLM-Based Multi-Agent System**

# Introduction:

Au sein de l’entreprise IDEO FACTORY, nous travaillons sur IDEO Studio Manager qui est une plateforme équipée d'un assistant IA qui facilite la conception et la gestion de modules e-learning de manière efficace et innovante.

# RAG (Retrieval augmented generation) :

La génération augmentée de récupération (RAG) est une technique qui complète la génération de texte avec des informations provenant de sources de données privées ou propriétaires. Elle combine un modèle de récupération, conçu pour effectuer des recherches dans de grands ensembles de données ou bases de données, à un modèle de génération tel qu'un [grand modèle de langage (LLM)](https://www.elastic.co/fr/what-is/large-language-models), qui extrait des informations et génère une réponse texte lisible.

# Fonctionnement du RAG :

La génération augmentée de récupération est un processus à plusieurs étapes qui commence avec la récupération, pour aller jusqu'à la génération. Voici comment elle fonctionne :

**Step 1 : Récupération**

* Le RAG commence avec une requête d'entrée. Il peut s'agir d'une question d'un utilisateur ou d'une partie de texte nécessitant une réponse détaillée.
* Un modèle de récupération(retrievement) se saisit d'informations pertinentes provenant de bases de connaissances, de bases de données ou de sources externes, voire de plusieurs sources en même temps. L'endroit où le modèle effectue ses recherches dépend de ce que demande la requête d'entrée. Les informations récupérées servent désormais de sources de référence pour les faits et le contexte dont le modèle a besoin.
* Les informations récupérées sont converties en vecteurs dans un espace à haute dimensionnalité. Ces vecteurs de connaissances sont stockés dans une [base de données vectorielle](https://www.elastic.co/fr/elasticsearch/vector-database).
* Le modèle de récupération classe les informations récupérées sur la base de leur pertinence avec la requête d'entrée. Des documents ou des passages ayant le score le plus élevé sont sélectionnés pour un traitement supplémentaire.

**Step 2 : Génération**

* Ensuite, un modèle de génération, comme un LLM, utilise les informations récupérées pour générer des réponses texte.
* Le texte généré peut passer par des étapes post-traitement supplémentaires pour garantir qu'il est grammaticalement correct et cohérent.
* Ces réponses sont globalement plus précises et ont plus de sens en contexte, car elles ont été modelées par les informations supplémentaires fournies par le modèle de récupération. Cette capacité est particulièrement importante dans des domaines spécialisés dans lesquels les données Internet publiques ne sont pas suffisantes.

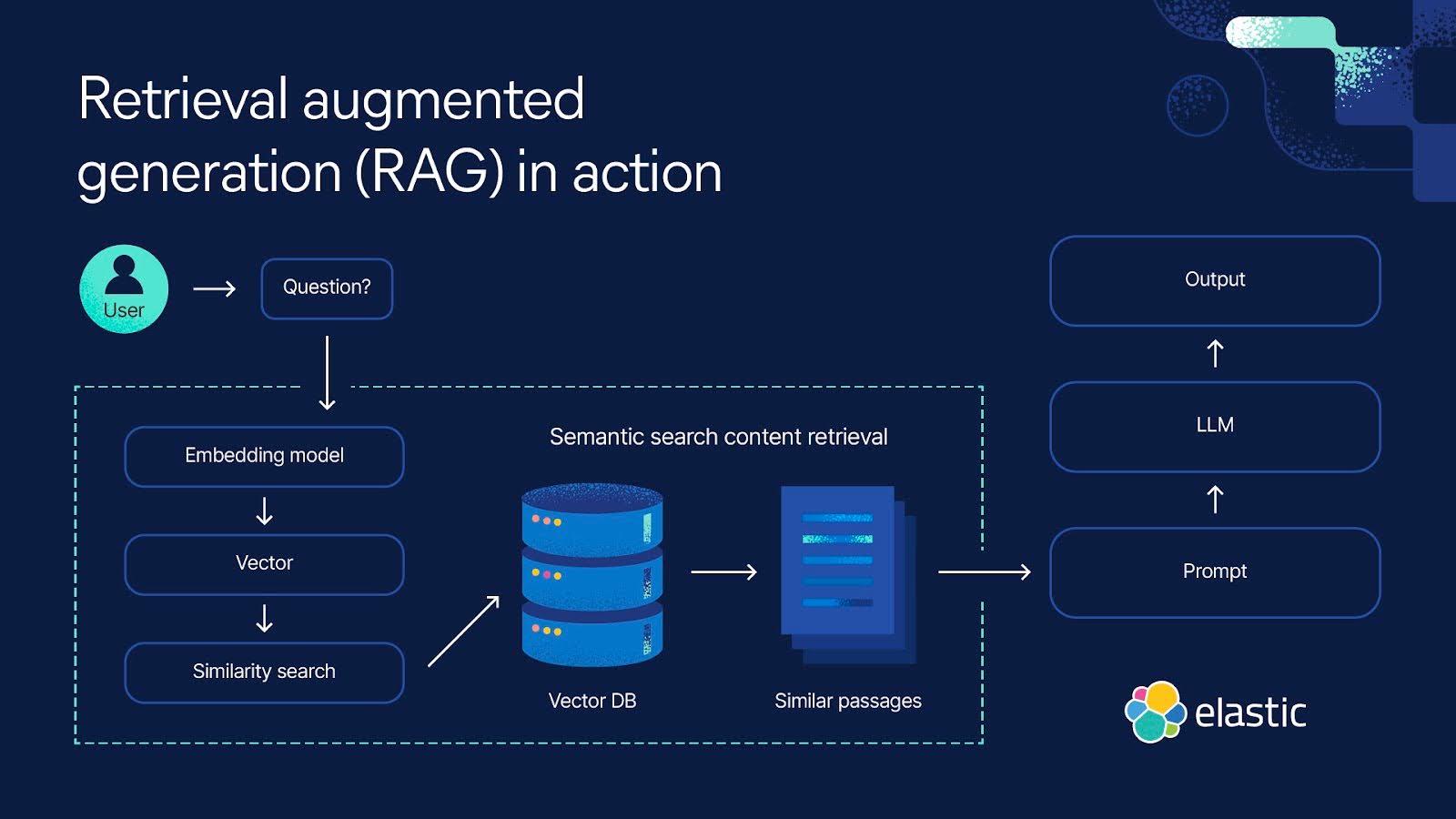


Figure 1 : Architecture du RAG

# Agentic RAG :

Agentic RAG = implémentation RAG basée sur un agent

Agentic RAG transforme notre approche de la réponse aux questions en introduisant un cadre innovant basé sur les agents. Contrairement aux méthodes traditionnelles qui reposent uniquement sur [de grands modèles de langage (LLM)](https://www.leewayhertz.com/better-output-from-your-large-language-model/) , Agentic RAG utilise des agents intelligents pour traiter des questions complexes nécessitant une planification minutieuse, un raisonnement en plusieurs étapes et l'utilisation d'outils externes. Ces agents agissent comme des chercheurs expérimentés, parcourant habilement plusieurs documents, comparant les informations, générant des résumés et fournissant des réponses complètes et précises.

# Fonctionnement du Agentic RAG :

Le système commence par charger plusieurs documents comme sources d’information, qui sont ensuite soumis à une analyse exploratoire (EDA) et un prétraitement pour en extraire les éléments utiles. Ces informations sont transformées en vecteurs via un processus d’embedding, puis stockées dans une base vectorielle afin de permettre une recherche rapide et efficace. Lorsqu’un utilisateur pose une question, un agent intelligent est activé pour orchestrer le processus de réponse en sélectionnant les outils appropriés, tels que le VectorStore Tool pour retrouver les passages pertinents et le Summary Tool pour structurer ou résumer ces informations. Enfin, des modèles de langage avancés comme GPT-4, LLaMA 3 ou Mistral sont utilisés pour générer une réponse cohérente à partir des éléments retrouvés, laquelle est ensuite transmise à l’utilisateur.

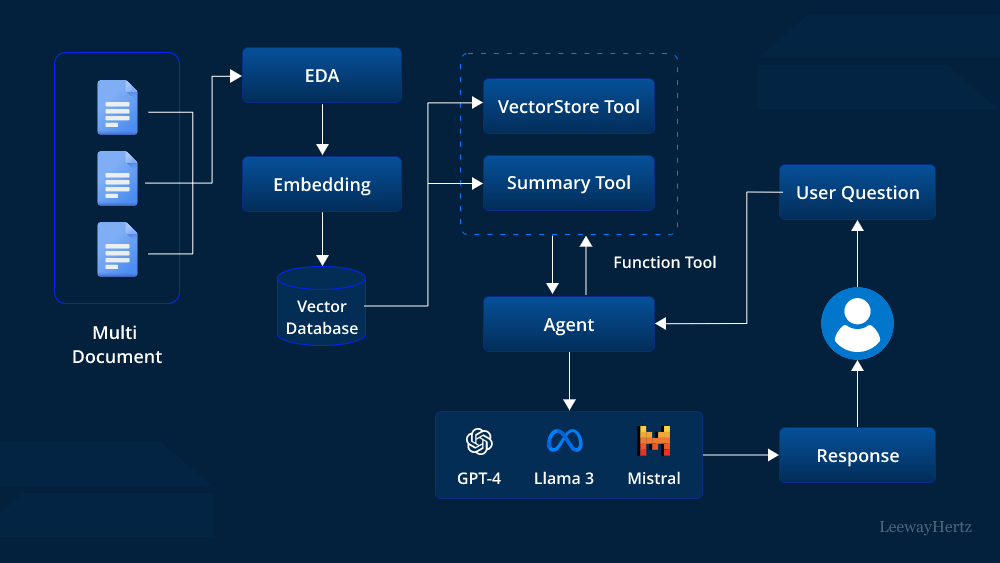


Figure 2: Architecture du Agentic RAG

# Multi-Agents:

Un système multi-agents (SMA) contient plusieurs agents d'intelligence artificielle qui collaborent dans un environnement commun pour atteindre des objectifs spécifiques. Chaque agent d'un système multi-agents peut agir indépendamment, percevoir et interpréter l'environnement, prendre des décisions, puis agir pour atteindre son objectif.

# Plateforme n8n :

Pour avoir un workflow visuel des différentes fonctionnalités de notre agent, nous avons utilisé n8n qui est une plateforme d’automatisation visuelle open-source qui permet de concevoir des workflows complexes sans écrire de code, en connectant des services, API et outils d’intelligence artificielle. Grâce à ses nœuds, triggers et fonctionnalités no-code/low-code, elle offre une grande flexibilité pour orchestrer des processus métier, y compris ceux basés sur l’IA.

Voici un petit exemple classique d’un agent de conseil météorologique :

L’agent LLM tente d'abord de localiser l'utilisateur pour connaître les conditions météorologiques. Il utilise alors l'outil mis à sa disposition pour obtenir cette localisation. Une fois cette localisation obtenue, l'agent utilise l'autre outil pour obtenir les prévisions météo du jour via une API météo. En fonction des prévisions et de celles de la veille, il détermine s'il s'agit d'une météo inattendue. Si oui, il génère des conseils personnalisés en fonction des conditions météorologiques et utilise l'outil pour lui envoyer un e-mail.

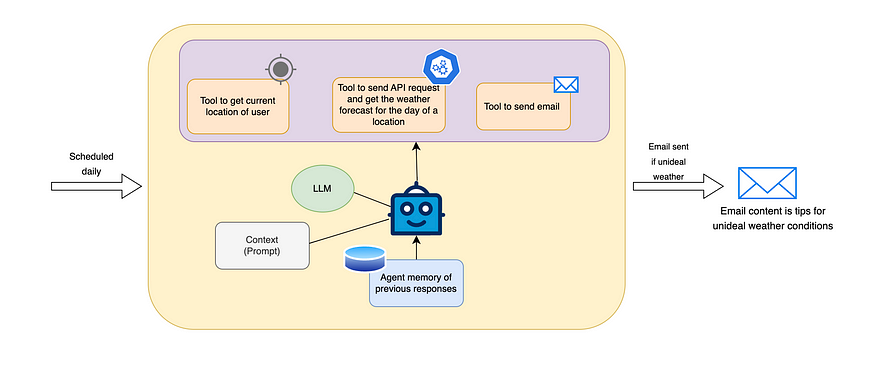


Figure 3 : Architecture de l’agent de conseil météorologique

# Design patterns :

L’utilisation des design patterns est essentielle pour implémenter AI Agents pour bien organiser le code et le réutiliser, pour notre agent on a utilisé 4 patterns ;

* **Tool Use Pattern :**

Le LLM reçoit directement la requête de l’utilisateur, identifie un besoin d’information ou d’action, et appelle un outil interne/externe (exemple : base vectorielle ou API). Une fois les données obtenues, le LLM génère une réponse finale à partir de ces résultats.

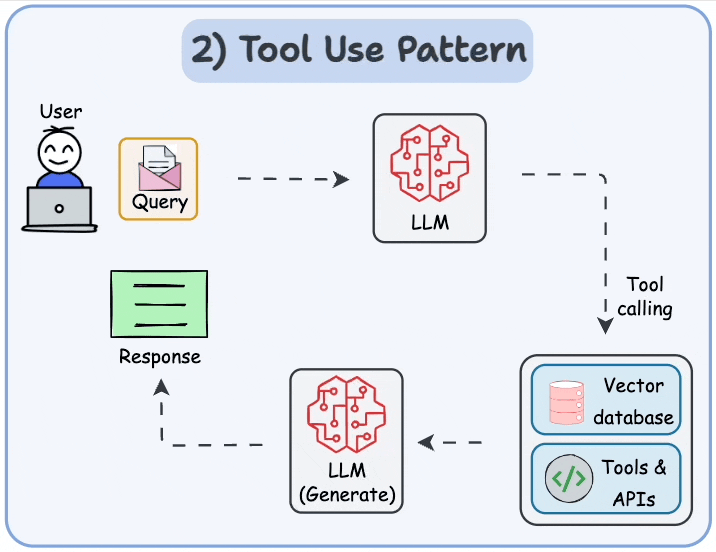


Figure 4 : Architecture du Tool Pattern

* **ReAct (Reason and Act) Pattern:**

Le LLM raisonne étape par étape pour décider quel outil appeler. Il exécute une action (exemple : interroger une API), observe le résultat, puis raisonne à nouveau. Il répète ce cycle jusqu’à pouvoir générer une réponse finale.

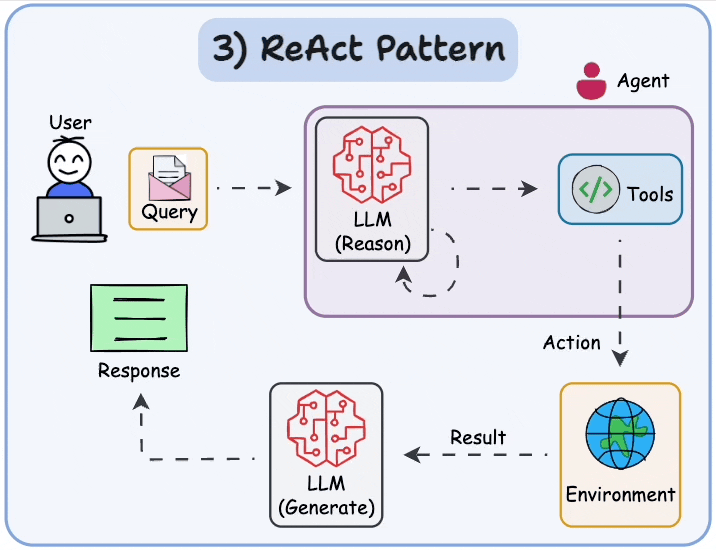


Figure 5 : Architecture du ReAct Pattern

* **Planning Pattern :**

Le système contient un *Planner* qui décompose une requête complexe en plusieurs sous-tâches. Chaque tâche est exécutée par un agent (souvent un ReAct Agent). Le système vérifie après chaque étape si le travail est terminé.

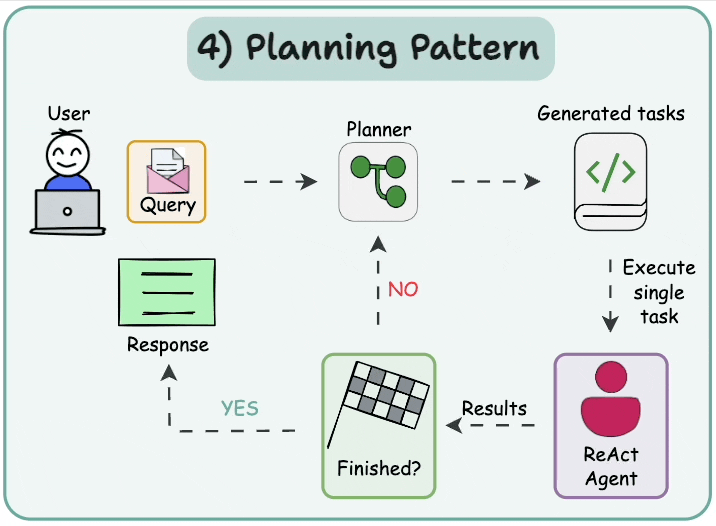


Figure 6 : Architecture du Planning Pattern

* **Multi-Agent Pattern :**

Un agent principal (souvent un PM agent) reçoit la requête et délègue à d’autres agents spécialisés (exemple : DevOps, Tech Lead, etc.). Chaque agent exécute une partie spécifique du travail, puis les résultats sont agrégés pour former une réponse cohérente.

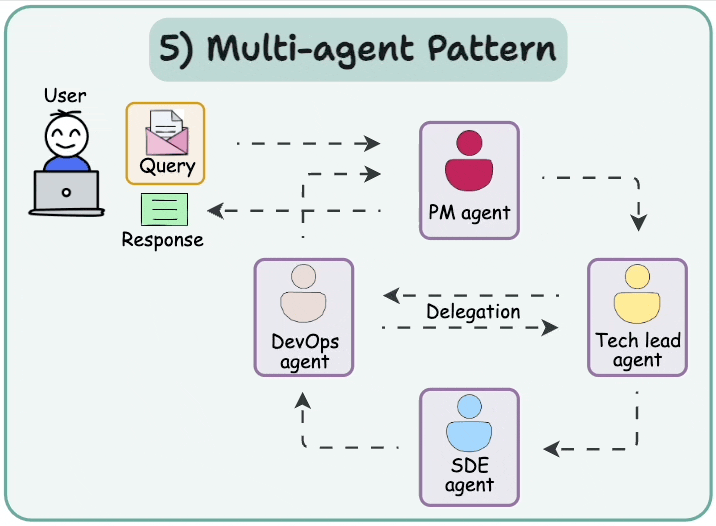
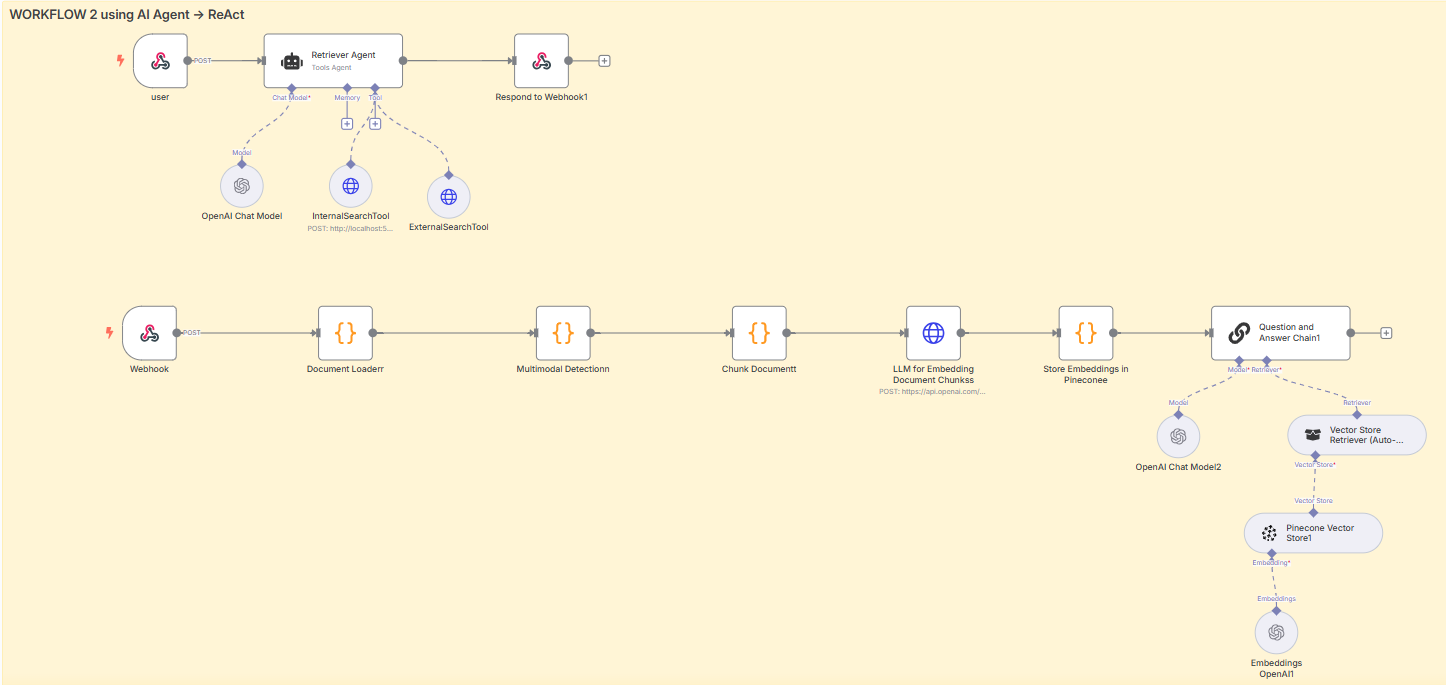


Figure 7 : Architecture du Multi-Agent Pattern

# Etat d’avancement :

Nous avons développé notre Agent en code et dans n8n le **Subject Matter Agent** qui fait l’analyse des différents types de documents. L’utilisateur choisit le type de recherche soit interne ou externe ou les deux ; s’il choisit internal l’agent fait choisit **Internal Search Tool** qui est basée sur des documents privé/interne charger par l’utilisateur dans la plateforme, ce Tool fait lire le fichier, ensuite la phase du **chunking** divise le document en morceaux(chunks) ensuite le model GPT fait l’**embedding** en transformant le texte sous forme des vecteurs numériques, ces vecteurs sont **stocker dans le vector database Pinecone**; ensuite en utilisant Chain Of Thought(COT) qui est **le Question and Answer Chain ;** le query de l’utilisateur est embedded pour ensuite trouver les vecteurs similaire au query dans vector database et la **récupération** fait renvoie le résultat au **LLM** (en utilisant OpenAI API); si l’utilisateur a choisi external alors l’agent fait travailler avec **External Search Tool** est fait une recherche du ressources externes et renvoie le résultat au LLM; si l’utilisateur choisit les deux types de recherche alors les deux Tools sont appeler. Enfin une réponse finale cohérant est générer par le LLM et renvoyer à l’utilisateur.

Figure 8 : Workflow dans n8n

# Prochaines étapes du projet :

* Amélioration du **Subject Matter Agent** et effectuer plusieurs tests en utilisant des métriques d’évaluations
* Commencer à implémenter l’agent suivant **Learning Objective Agent**