1. Réaliser une étude comparative des types de RAG, classés par type d'architecture, selon les critères suivants :

- Mode d'implémentation
- Mode de génération
- Mode de récupération du contexte
- Organisation du workflow
- Stockage et persistance
- Domaine d'application

Étude comparative des architectures RAG

Le RAG combine des modèles de récupération d'information et de génération de texte afin de permettre aux LLM d'accéder dynamiquement à des connaissances spécialisées et actualisées. Cette approche permet de produire des résultats précis, pertinents et fondés sur des faits, adaptés à de nombreux cas d'utilisation.

Critères de comparaison

- **Mode d'implémentation** (architecture, technologies, complexité)
- **Mode de génération** (type de générateur, séquence-à-séquence, multi-agent, etc.)
- **Mode de récupération du contexte** (vectorielle, lexicale, hybride, graphe, etc.)
- **Organisation du workflow** (séquentiel, parallèle, adaptatif, etc.)
- **Stockage et persistance** (mémoire, cache, base de données, etc.)
- **Domaine d'application** (utilisations privilégiées)

---

1. RAG traditionnel (Standard RAG)

Mode d'implémentation : Pipeline séquentiel :

récupération vectorielle (embedding dense) => génération séquence-à-séquence (T5, BART…)

Mode de génération : Générateur séquence-à-séquence, prompt augmenté

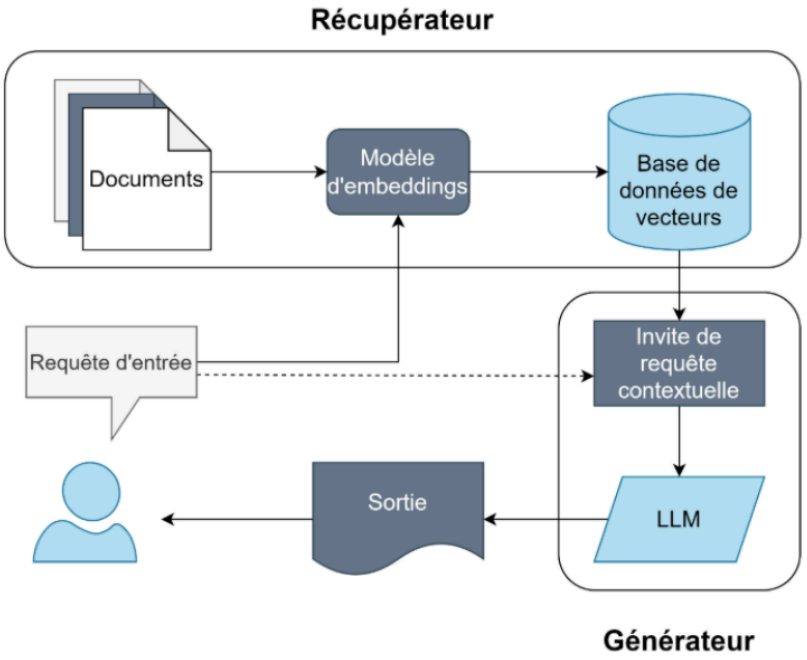
Mode de récupération du contexte : Vectorielle (embedding dense, Word2Vec...)

Organisation du workflow : Séquentiel :

récupération → augmentation du prompt → génération

Stockage et persistance : Index vectoriel (type FAISS, Pinecone…), éventuellement cache de prompts

Domaine d'application : Q&A open-domain, résumé, chatbots, recherche documentaire sur données non structurées



---

2. GraphRAG

Mode d'implémentation : Ajout d'une couche graphe de connaissances pour structurer l'information ; récupération et génération orientées graphe

Mode de génération : Générateur tenant compte de la structure du graphe (relations, entités…)

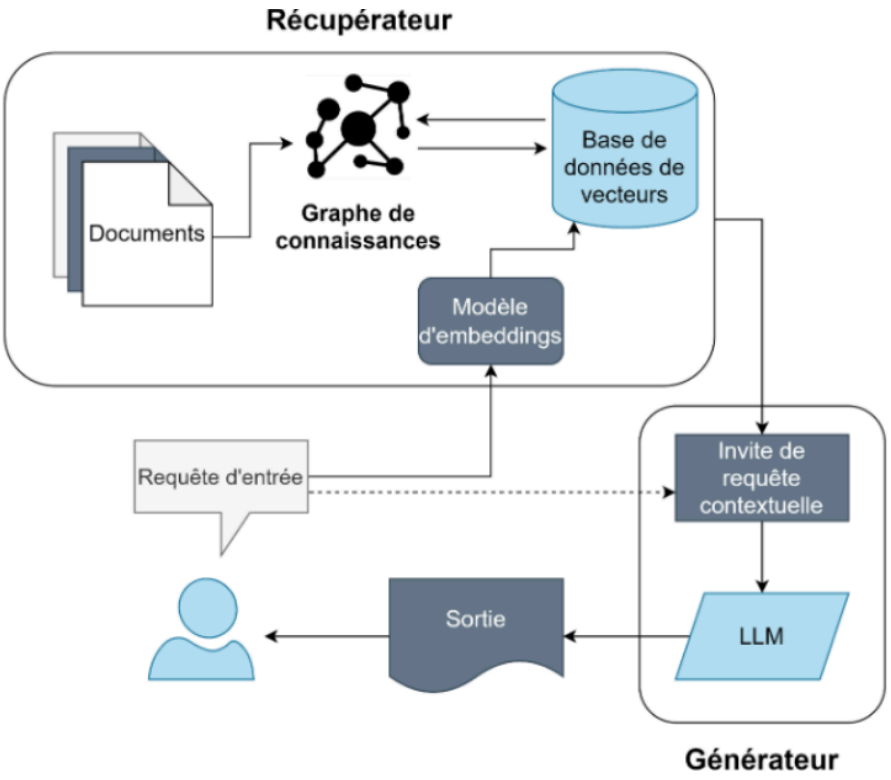
Mode de récupération du contexte : Parcours de graphe, requêtes sur graphes de connaissances, embeddings de graphes

Organisation du workflow : Séquentiel ou hybride :

requête → récupération sur graphe → génération

Stockage et persistance : Base de données graphe (Neo4j, RDF, etc.)

Domaine d’application : Données interconnectées (détection de fraude, recherche scientifique, systèmes de recommandation)



---

3. Simple RAG with Memory

Mode d’implémentation : Ajout d’un module mémoire pour conserver le contexte des interactions précédentes

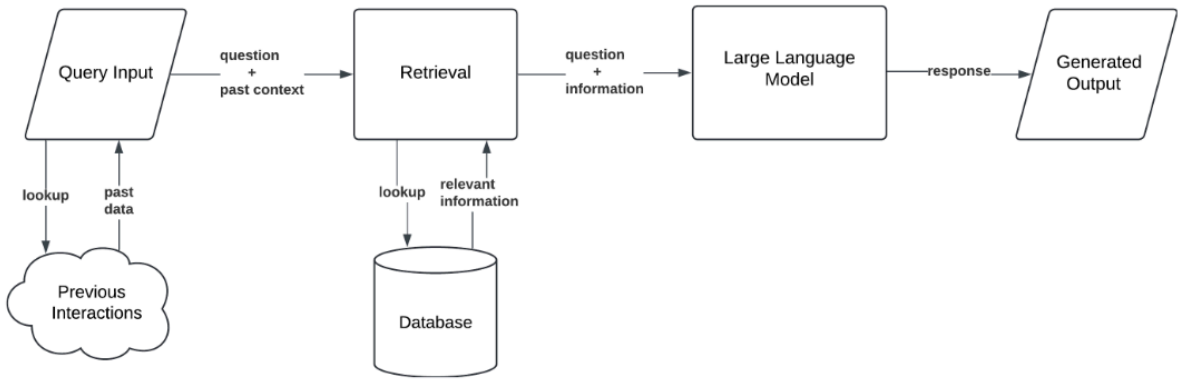
Mode de génération : Générateur prenant en compte mémoire + contexte récupéré

Mode de récupération du contexte : Vectorielle classique, enrichie par contexte stocké

Organisation du workflow : Séquentiel avec accès mémoire

Stockage et persistance : Mémoire conversationnelle, cache de prompts, persistance possible via base de données

Domaine d’application : Chatbots avec suivi contextuel, recommandations personnalisées



Simple RAG with Memory

---

4. Branched RAG

Mode d'implémentation : Sélection dynamique de la source de récupération la plus pertinente

Mode de génération : Générateur standard

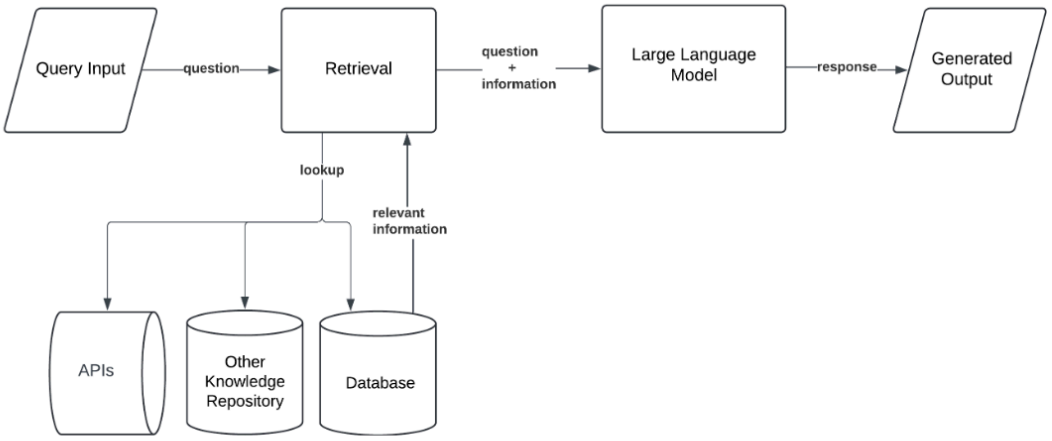
Mode de récupération du contexte : Vectorielle/lexicale/graphique selon branche

Organisation du workflow : Branche :

évaluation de la requête → sélection de la source → récupération → génération

Stockage et persistance : Index multiples (vectoriel, lexical, graphe…)

Domaine d'application : des requêtes complexes qui exigent des connaissances spécialisés (légal, médical)



Branched RAG

---

5. HyDe (Hypothetical Document Embedding)

Mode d'implémentation : Génération d'un « document hypothétique » (embedding du document idéal), puis récupération

// HyDe commence par créer une représentation vectorielle d'un document idéal, en se basant sur la requête. Ensuite, il utilise ce document hypothétique pour guider la recherche des documents.

Mode de génération : Générateur séquence-à-séquence, influencé par l'embedding hypothétique

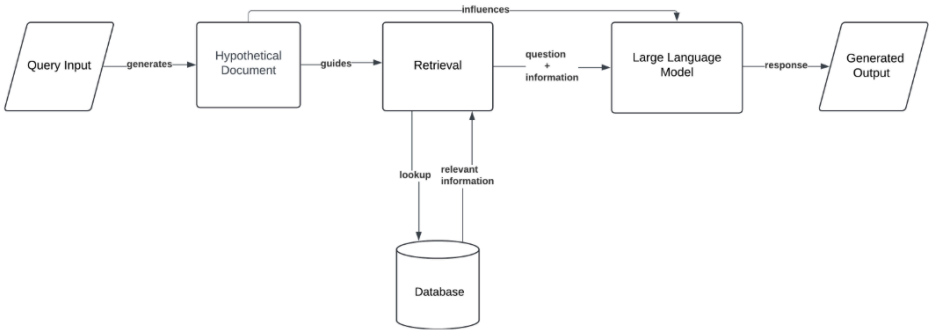
Mode de récupération du contexte : Vectorielle, guidée par l'embedding généré

Organisation du workflow :

Génération hypothétique → récupération → génération finale

Stockage et persistance : Index vectoriel classique

Domaine d'application : Recherche et développement, génération créative, réponses à requêtes vagues



HyDe (Hypothetical Document Embedding)

---

6. Adaptive RAG

Mode d'implémentation : Récupération adaptative selon la complexité de la requête

// Pour des requêtes simples, il peut récupérer des documents depuis une seule source, tandis que pour des requêtes plus complexes, il peut accéder à plusieurs sources de données ou utiliser des techniques de recherche plus avancées.

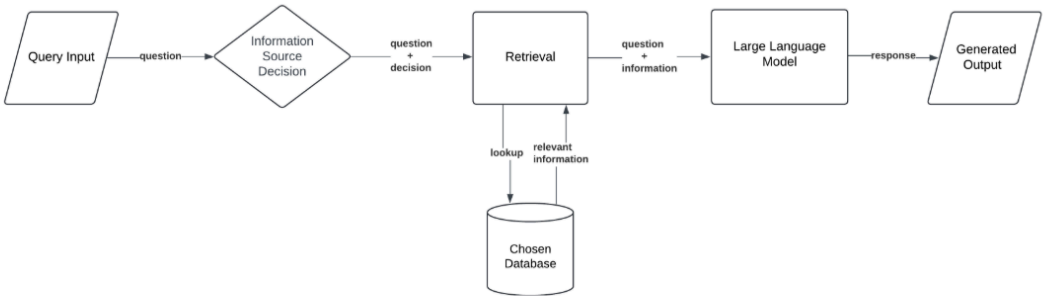
Mode de génération : Générateur adaptatif, capable de fusionner diverses sources

Mode de récupération du contexte : Vectorielle, lexicale, hybride, choix dynamique

Organisation du workflow : Adaptatif (chemin déterminé en temps réel)

Stockage et persistance : Index multiples, logique de fusion

Domaine d'application : Recherche d'entreprise(la nature des requêtes varie significativement), gestion de requêtes hétérogènes



Adaptive RAG

---

7. Corrective RAG (CRAG)

Mode d’implémentation : Mécanisme de self-grading (auto-évaluation des documents récupérés)

// CRAG évalue de manière critique la qualité des informations récupérées avant de passer à la phase de génération: Le système découpe les documents récupérés en "fragments de connaissance" et attribue une note à chaque fragment selon sa pertinence.

Si la récupération initiale ne dépasse pas un certain seuil de pertinence, CRAG lance des étapes de récupération supplémentaires, comme des recherches web, afin de s'assurer qu'il dispose des meilleures informations possibles pour générer la réponse.

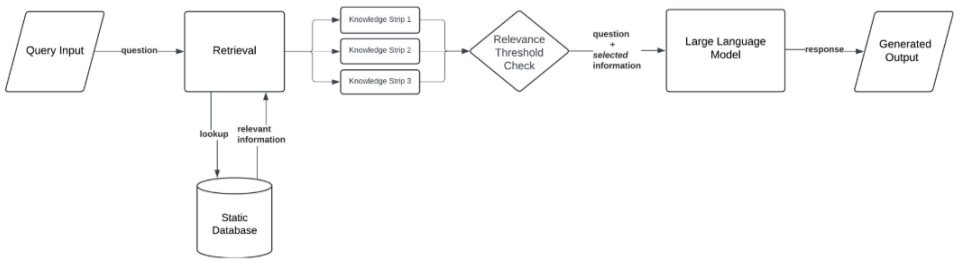
Mode de génération : Générateur standard, alimenté par les « knowledge strips » les plus pertinents

Mode de récupération du contexte : Vectorielle, enrichie par étapes de filtrage/réévaluation

Organisation du workflow : Récupération → découpage en strips → évaluation → complétion si nécessaire → génération

Stockage et persistance : Index vectoriel, base de données pour knowledge strips

Domaine d’application : Domaines sensibles : juridique, médical, financier (exige high accuracy)



Corrective RAG (CRAG)

---

8. Self-RAG

Mode d’implémentation : Boucle auto-réflexive : le modèle génère de nouvelles requêtes de récupération pendant la génération

// Self-RAG introduit un mécanisme d’auto-récupération, permettant au modèle de générer lui-même des requêtes de recherche pendant le processus de génération.

Self-RAG peut affiner ses requêtes de recherche de manière itérative au fur et à mesure qu’il génère du contenu.

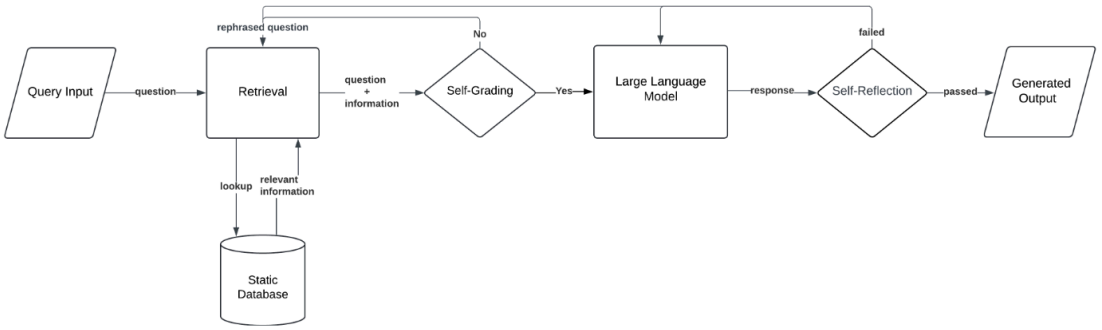
Mode de génération : Générateur itératif, amélioration continue du contexte

Mode de récupération du contexte : Vectorielle, guidée par le modèle

Organisation du workflow : Boucle : génération → détection des lacunes → nouvelle récupération → génération, ...

Stockage et persistance : Index vectoriel, mémoire temporaire

Domaine d’application : Recherche exploratoire (Le modèle doit récupérer davantage d’informations de manière dynamique au fur et à mesure que la réponse se développe), génération de contenu long



Self-RAG

---

9. Agentic RAG

If **Self-RAG** is **introspective**, **Agentic RAG** is **strategic**.

Instead of just fixing bad outputs, Agentic RAG thinks like a planner or a researcher. It breaks queries into subtasks, dynamically decides how to search, when to stop, and how to synthesize the final response.
In practice:

- Analysing the user’s intent
- Splitting the query if needed
- Running separate retrievals
- Asking follow-up questions

1. **Deep Query Understanding**: It starts by truly understanding what the user wants.

2. **Smart Path Selection**: Based on the query, it decides whether to search the web, use personal data, or go straight to the LLM.
3. **Multiple Data Sources**: Instead of pulling info from just one place, it fetches high-quality data from various trusted sources.
4. **Refinement Loops**: Like Self-RAG, it checks and improves the data — but it goes further by rewriting queries in smarter ways to get even better results.

Mode d'implémentation : Orchestration multi-agent : décomposition de la requête, agents de récupération parallèles, consolidation

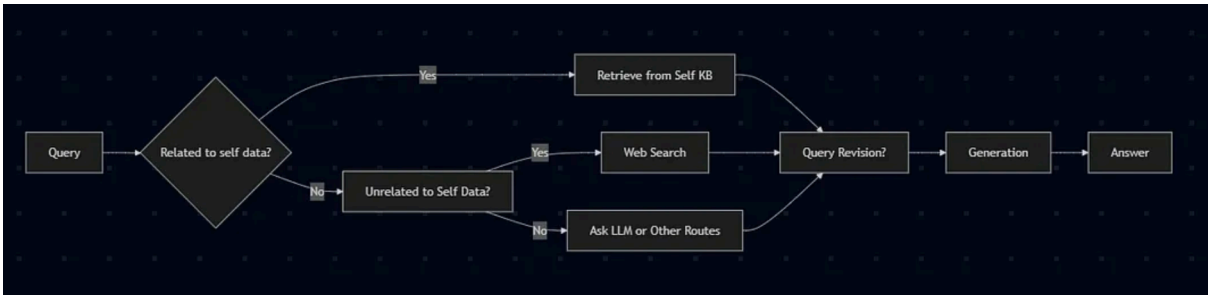
Mode de génération : Générateur orchestré, reranker LLM

Mode de récupération du contexte : Multi-source (web, base interne, API…), requêtes parallélisées

Organisation du workflow : Orchestration, parallélisme, boucle de raffinement et de reranking

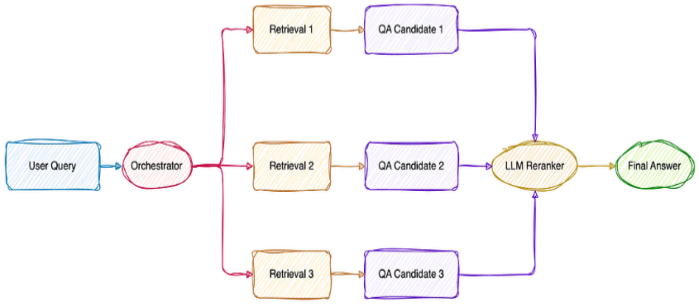
Stockage et persistance : Multiples index, mémoire d'orchestration

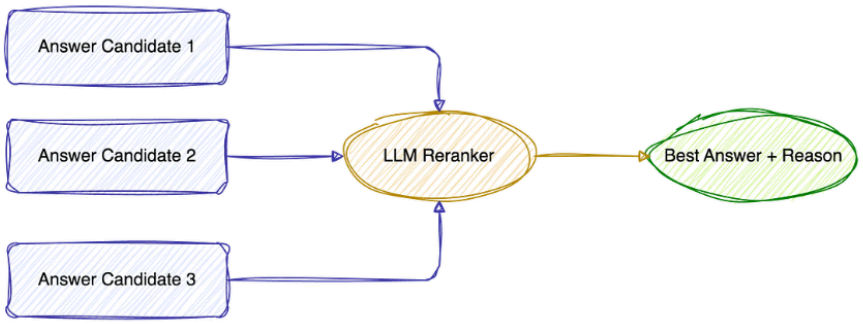
Domaine d'application : Recherche automatisée, agrégation multi-source, support décisionnel



L'orchestrateur permet le parallélisme.
À partir de la requête de l'utilisateur, il lance plusieurs agents de récupération (retrieval agents) **en parallèle**.
Chaque contexte récupéré est ensuite traité dans une étape de questions-réponses (QA) séparée, ce qui produit **plusieurs réponses candidates**.
À la fin, un **LLM Reranker** (modèle de classement) analyse et consolide toutes les réponses candidates, **sélectionne la meilleure**, et fournit une **réponse finale**.

Cette approche est **robuste face au bruit** dans la récupération d'informatio





---

10. Hybrid RAG (Lexical + Vector)

Mode d’implémentation : Double pipeline : vectoriel + lexical (BM25, etc.), fusion des résultats

Mode de génération : Générateur séquence-à-séquence

Mode de récupération du contexte : Vectorielle ET lexicale, résultats fusionnés/dédoublonnés

Organisation du workflow : Parallélisme, fusion avant génération

Stockage et persistance : Index vectoriel et lexical

Domaine d’application : Bases hétérogènes, requêtes ambiguës

---

11. RAG Multimodal

Mode d’implémentation : Capacité à récupérer et traiter divers types de données (texte, image, audio, code…)

Mode de génération : Générateur multimodal

Mode de récupération du contexte : Multimodale (vectorielle, lexicale, embeddings d'images, etc.)

Organisation du workflow : Parallélisme par modalité, fusion des résultats

Stockage et persistance : Index multimodal (vectoriel, image, audio…)

Domaine d'application : Recherche scientifique, assistants médicaux, code assistants

---

12. Streaming/Real-time RAG

Mode d'implémentation : Intégration de flux de données temps réel (news, logs, réseaux sociaux…)

Mode de génération : Générateur standard, prompt mis à jour dynamiquement

Mode de récupération du contexte : Vectorielle sur index mis à jour en continu

Organisation du workflow : Récupération temps réel, génération à la demande

Stockage et persistance : Index en écriture continue, base temps réel

Domaine d'application : Analyse financière, veille, monitoring

---

13. Re-ranking/Feedback-Augmented RAG

Mode d'implémentation : Application d'un reranker (LLM ou classifieur) après récupération

Mode de génération : Générateur séquence-à-séquence alimenté par les documents rerankés

Mode de récupération du contexte : Vectorielle ou hybride, reranking avant génération

Organisation du workflow : Récupération → reranking → génération

Stockage et persistance : Index vectoriel, logs de feedback

Domaine d'application : Domaines où la précision est critique (médical, juridique)

---

14. Multi-granularity/Passage-level RAG

Mode d’implémentation : Indexation à plusieurs granularités (phrase, paragraphe, document)

Mode de génération : Générateur s’adaptant à la granularité récupérée

Mode de récupération du contexte : Vectorielle, granularité dynamique

Organisation du workflow : Sélection de la granularité selon la requête

Stockage et persistance : Index multi-niveau

Domaine d’application : Documents volumineux, manuels techniques

---

Autres familles notables

RAG Chain-of-Thought : Génération étape par étape (reasoning chain) et récupération intermédiaire à chaque étape.

RAG Fine-tuned : Générateur ou récupérateur spécifiquement fine-tuné pour un domaine ou une tâche.

RAG Tool-augmented et RAG with External Tools/API : Capacité à interroger dynamiquement des API externes/outils métier pendant la génération.

Architecture	Implémentation / Architecture	Mode de génération	Mode de récupération du contexte	Organisation du workflow	Stockage et persistance	Domaines d’application
RAG traditionnel (Standard RAG)	Pipeline séquentiel récupération/ génération	Seq2Seq (T5, BART...)	Vectorielle (embeddings denses)	Séquentiel	Index vectoriel (FAISS, Pinecone...), cache de prompts	Q&A open-domain, résumé, chatbot, recherche documentaire

GraphRAG	Ajout d’un graphe de connaissances	Générateur graphe-aware	Parcours de graphe, embeddings de graphes	Séquentiel ou hybride	Base graphe (Neo4j, RDF, ...)	Données interconnectées, détection de fraude, recherche scientifique, reco
Simple RAG with Memory	Ajout module mémoire	Générateur tenant compte de la mémoire	Vectorielle + accès mémoire	Séquentiel avec mémoire	Mémoire conversationnelle, cache, base de données	Chatbots contextuels, recommandation personnalisée
Branched RAG	Sélection dynamique de la source	Générateur standard	Vectorielle/lexicale/graphique selon branche	Branche (sélection source dynamique)	Index multiples (vectoriel, lexical, graphe)	Multidisciplinaire, outils spécialisés
HyDe	Génération d’un document hypothétique	Générateur influencé par embedding hypothétique	Vectorielle guidée par l’embedding généré	Hypothèse → récupération → génération	Index vectoriel classique	Recherche, génération créative, requêtes vagues
Adaptive RAG	Récupération adaptative selon la complexité	Générateur adaptatif	Vectorielle, lexicale, hybride, dynamique	Adaptatif (workflow dynamique)	Index multiples, logique de fusion	Recherche entreprise, requêtes hétérogènes
Corrective RAG (CRAG)	Self-grading, auto-évaluation des documents	Générateur standard, knowledge strips	Vectorielle, filtrage/évaluation	Récupération → strips → évaluation →	Index vectoriel, base de données strips	Juridique, médical, finance (domaines sensibles)

		pertinents		raffinement		
Self-RAG	Boucle auto-réflexive	Générateur itératif	Vectorielle, guidée par modèle	Boucle génération/récupération	Index vectoriel, mémoire temporaire	Recherche exploratoire, génération longue
Agentic RAG	Orchestration multi-agent, décomposition requête	Générateur orchestré, reranker LLM	Multi-source (web, interne, API), requêtes parallèles	Parallélisme, raffinement, consolidation	Multiples index, mémoire d'orchestration	Recherche auto, agrégation multi-source, support décision
Hybrid RAG (Lexical + Vector)	Double pipeline vectoriel + lexical, fusion résultats	Générateur séquence-à-séquence	Vectorielle ET lexicale, fusion/dédoublonnage	Parallélisme, fusion	Index vectoriel et lexical	Bases hétérogènes, requêtes ambiguës
RAG Multimodal	Support multi-modalité (texte, image, audio, code...)	Générateur multimodal	Récupération multimodale (texte, image, etc.)	Parallélisme par modalité, fusion	Index multimodal (vectoriel, image, audio, ...)	Recherche scientifique, médical, code assistant
Streaming/ Real-time RAG	Intégration flux temps réel (news, logs, ...)	Générateur standard	Vectorielle sur index en temps réel	Temps réel, mise à jour continue	Index écriture continue, base temps réel	Analyse financière, veille, monitoring
Re-ranking /Feedback-Augmented RAG	Ajout d'un reranker après récupération	Générateur alimenté	Vectorielle ou hybride, reranking	Récupération → reranking	Index vectoriel, logs de feedback	Précision critique (médical, juridique)

		par docs rerankés		→ génération		
Multi-granularity/Passage-level RAG	Indexation multi-niveau (phrase, paragraphe, doc)	Générateur adaptatif selon granularité	Vectorielle, granularité dynamique	Sélection de granularité	Index multi-niveau	Documents volumineux, manuels techniques
RAG Chain-of-Thought	Génération étape par étape (reasoning chain)	Générateur étape par étape, récupération intermédiaire	Vectorielle ou hybride, récupération multi-étapes	Raisonnement en chaîne	Index vectoriel, mémoire de raisonnement	Raisonnement complexe, explications
RAG Fine-tuned	Modèles fine-tunés pour un domaine	Générateur ou récupérateur fine-tuné	Vectorielle/lexicale adaptée	Personnalisé	Index spécialisé, modèles fine-tunés	Domaine spécifique (santé, droit, industrie)
RAG Tool-augmented/External Tools	Interrogation d'APIs/outils externes dynamiquement	Générateur orchestrant outils et données	Multi-source + interrogation dynamique d'APIs			

Conclusion

Le choix de l'architecture RAG dépend du cas d'usage, du type de données à traiter, des exigences de précision et de rapidité, et de la complexité des requêtes.

- **RAG standard** : pour la simplicité et la rapidité.
- **Self-RAG/CRAG** : pour la précision et la robustesse.

- **Agentic/Branched/Adaptive RAG** : pour les tâches complexes et multi-source.
- **RAG Multimodal/Streamin**g : pour les cas avec données variées ou en temps réel.

a) Synthèse des besoins spécifiques du projet

- **Volume et hétérogénéité des données** : tendances marché, documents concurrents, signaux faibles sectoriels, documents internes.
- **Cas d'usage variés** : veille automatisée, benchmark, synthèse, génération d'offres personnalisées, adaptation au secteur et à la cible.
- Exigence de pertinence et différenciation des réponses.
- **Scalabilité et adaptabilité** : le système doit pouvoir évoluer avec de nouveaux types de sources ou de formats (PPT, PDF, web scraping…).

b) Architecture RAG recommandée

Architecture recommandée : RAG Adaptive et Multi-source, orchestrée

- Pourquoi ?
    - **Adaptive** : permet d'ajuster dynamiquement la stratégie de récupération (ex : vectorielle pour les tendances, lexicale pour les concurrents, API pour les données de veille).
    - **Multi-source** : indispensable car tu dois agréger plusieurs types de connaissances (veille, benchmark, doc interne, signaux externes).
    - **Orchestration** : pour permettre l'enchaînement de tâches complexes (veille → synthèse → génération d'offre → design automatique).
    - Possibilité d'intégrer une mémoire conversationnelle pour le suivi de contexte lors de sessions d'élaboration d'offres.

Avantages :

- Meilleure pertinence et adaptabilité selon la requête.
- Capacité à mixer, hiérarchiser et filtrer différentes sources pour produire des propositions différenciantes.
- Facilité à ajouter des modules spécialisés (analyse concurrentielle, scoring, design de propales…).

c) Technologies et frameworks recommandés

1. Pour le cœur RAG (Orchestration & retrieval)

- LangChain (Python, open-source)
    - Points forts : très modulaire, supporte nativement le multi-source, l'orchestration de pipelines, les agents, la mémoire conversationnelle, l'intégration de modules custom (scraping, parsers, etc.), et tous les principaux vector stores (FAISS, Pinecone…).
    - Supporte la génération en plusieurs étapes (reasoning chain, multi-prompt), la connexion à des APIs externes, la gestion de workflows adaptatifs.
- LlamaIndex (ex-GPT Index)
    - Points forts : performant pour l'indexation multi-granularité, la fusion de sources, la récupération intelligente, l'intégration facile avec LangChain et les LLMs du marché.

2. Pour le stockage et la recherche

- Vector store : Pinecone, Weaviate, Chroma, ou FAISS (open-source)
    - Choix selon la volumétrie et les contraintes de déploiement (cloud ou on-premise).
    - Pour un POC ou une version cloud, Pinecone est très simple à utiliser, scalable, et bien documenté.
- Base de données : PostgreSQL (pour logs, persistance structurée, mémoire conversationnelle si besoin).

3. Pour la génération de documents

- LLM : OpenAI (GPT-4o/3.5-turbo), Mistral, Llama 3 (selon contraintes RGPD/coût)
    - Possibilité d'intégrer des modèles spécialisés via API.
- Module de génération de slides/propales :
    - Intégration avec PowerPoint/Google Slides via API, ou génération automatique de PDF via Python (reportlab, pptx, etc.)
    - Export en Notion via API si besoin.

4. Pour la veille et le scraping

- Modules Python custom (scrapy, beautifulsoup, selenium)
    - Automatisation de la collecte de signaux (offres, actualités, incidents, etc.)
- Intégration possible d'APIs externes (alertes cybersécurité, bases appels d'offres, etc.)

5. Pour le développement rapide d'un assistant interne

- LangChain Agents : pour orchestrer la récupération, l’analyse, la génération, et le design des offres en workflow.
- UI : Streamlit, Gradio, ou intégration dans Notion pour une interface simple.