1 Base vectorielle

Le repository utilise Qdrant comme base vectorielle, lancé via Docker Compose, pour avoir un stockage persistant et un service séparé de l'API, comme on le ferait en conditions réelles. Ce choix permet de stocker les vecteurs, mais aussi un payload complet (texte du chunk et métadonnées comme la source et la page), ce qui est indispensable pour filtrer, tracer l'origine des informations et produire des citations propres. FAISS en local aurait été plus rapide à mettre en place, mais moins proche d'un cas d'usage entreprise car il faut gérer soi-même la persistance et l'exploitation.Des alternatives comme pgvector, Milvus ou Weaviate auraient aussi convenu, mais Qdrant est un bon compromis entre simplicité, maturité et intégration facile dans une stack Docker de démo.

2 Stratégie d'embeddings

Pour les embeddings, le choix est une approche locale avec SentenceTransformers et un modèle BGE. Ça évite les coûts à la requête et la dépendance à une API externe, ce qui rend la démo plus simple à exécuter et plus prévisible en latence. Le code sépare clairement la génération de texte, qui peut passer par OpenAI ou par un backend de type Ollama, et la production d'embeddings, qui reste dans la librairie RAG. Cette séparation rend le pipeline plus stable et plus facile à tester, et elle permet de remplacer le modèle d'embeddings plus tard sans toucher à l'orchestration.

3 Stratégie de chunking

Le repository découpe les documents en chunks de taille contrôlée avec un découpage récursif et un léger chevauchement. Les PDF sont traités page par page, puis découpés, et les fichiers texte et Markdown sont chargés puis découpés de la même façon. Ce choix est volontairement simple, car il fonctionne bien sur des formats variés et il limite la perte d'information aux frontières grâce au chevauchement. La limite est connue: ce découpage ne suit pas toujours la structure logique du contenu, donc il peut couper des tableaux, du code ou des sections très structurées à des endroits peu naturels.

4 Méthode de retrieval

La méthode de retrieval est hybride: elle combine une recherche dense par similarité vectorielle et une recherche lexicale sur le texte stocké dans Qdrant, puis elle fusionne les résultats et

applique un reranking léger. Le dense récupère bien les passages pertinents sur le fond, tandis que le lexical aide quand la question contient des termes exacts, comme des acronymes, des identifiants ou des intitulés. La fusion déduplique les chunks et évite d'envoyer plusieurs fois le même contenu à l'étape de raisonnement. Le reranking actuel est volontairement déterministe et peu coûteux, ce qui aide la reproductibilité et les tests, mais il pourra être remplacé par un reranker plus puissant si la précision devient prioritaire.

5 Framework d'agents

Le framework d'agents choisi est LangGraph, car il permet de décrire l'orchestration sous forme de graphe d'états explicite, avec une structure claire et des étapes bien séparées. Chaque nœud a un rôle précis, par exemple planification, retrieval, raisonnement, citations et mémoire, ce qui rend le système plus lisible et plus simple à faire évoluer. Ce choix favorise aussi le test et le débogage, car le passage d'un état à l'autre est explicite au lieu d'être caché dans une boucle d'agent. D'autres frameworks auraient pu fonctionner, mais LangGraph correspond bien à l'objectif de garder un routage contrôlé et un comportement prévisible dans une démo.

6 Modèle d'orchestration

Le modèle d'orchestration retenu repose sur un superviseur léger qui choisit le bon chemin, puis sur un pipeline RAG séquentiel pour produire la réponse. Le superviseur envoie la requête vers une réponse directe pour les cas simples, vers une demande de clarification si la question est incomplète, ou vers le chemin RAG quand une réponse doit être fondée sur les documents. Ce superviseur reste volontairement simple pour limiter la latence et éviter d'utiliser un appel LLM uniquement pour décider d'un routage évident. Le pipeline est conçu pour pouvoir évoluer, par exemple en améliorant les prompts, en ajoutant des règles de décision plus fines, ou en renforçant la qualité du retrieval et du reranking.