Livrable — Projet complet (max 2 pages)

Thème : Assistant conversationnel RH/juridique en français, basé sur une architecture RAG (recherche + génération) et un LLM open-source (licence Apache 2.0).

1) Choix du modèle open-source

Modèles cibles (OSI / Apache-2.0) :
• Mistral 7B (Ollama: **mistral**) — bon compromis qualité/latence, efficace en FR.
• Qwen2.5 7B (Ollama: **qwen2.5:7b**) — très bon en raisonnement/JSON, large contexte; la majorité des tailles sont en Apache-2.0.
• Mixtral (Ollama: **mixtral:8x7b**) — MoE, qualité supérieure mais plus lourd.

Pourquoi pas Llama 3.1 ? Licence Meta Llama 3.x: souvent « open-weights » mais pas « open-source » au sens OSI (restrictions d'usage).

2) Stratégie d'adaptation

RAG + prompt engineering (option LoRA si besoin de standardiser la forme). Le RAG réduit les hallucinations, produit des réponses sourcées (citations) et permet de mettre à jour le savoir par réindexation.

3) Pipeline technique (bout-en-bout)

Étape	Entrées	Sorties / contrôles clés
Ingestion	Docs internes (PDF/DOCX/TXT) + métadonnées	Nettoyage, versioning, chunking (taille + overlap)
Indexation	Chunks	Index dense (FAISS) + index lexical (BM25)
Retrieval hybride	Question + filtres (pays, version, droits)	Top-k passages: BM25 + vecteur, fusion pondérée
Génération	Passages + question	Réponse structurée + citations [doc_id]; refus si contexte insuffisant
API & logs	Requêtes	Traçabilité (sans PII), latence, scores retrieval, feedback

4) Plan d'évaluation

• Offline : jeu de test validé RH/juristes, Recall@K (retrieval), exactitude, taux d'affirmations non sourcées, qualité des citations, robustesse prompt-injection.
• Online : feedback (👍/👎), suivi latence, drift documentaire (docs manquants/obsolètes).

5) Risques & mitigations

1) Hallucinations → RAG obligatoire, température basse, refus si pas de preuves, reranking optionnel.
2) Confidentialité (RH/RGPD) → RBAC, filtrage par métadonnées, minimisation des logs (pas de PII), chiffrement.
3) Prompt injection → consignes système fortes, ignorer instructions dans les documents, limiter le contexte.

6) Mise en oeuvre (livrables techniques)

Le projet fournit une implémentation complète (démo locale) : ingestion → indexation → retrieval hybride → génération → API.

Composant	Description
src/ingest.py	Ingestion + chunking + construction index FAISS et BM25.
src/retrieval.py	Retrieval hybride (BM25 + dense) + fusion de scores + filtres metadata.
src/rag.py	Construction du prompt, garde-fous, format de sortie + extraction citations.
src/llm_client.py	Client LLM (Ollama par défaut) + option vLLM API compatible OpenAI.
src/api.py	API FastAPI: endpoints /ask et /health.
eval/run_eval.py	Évaluation retrieval (Recall@K) sur dataset JSONL.

Endpoints : `POST /ask` (question + filtres) → réponse + citations; `GET /health`.

Config : choix du modèle via `llm.model` (ex: mistral, qwen2.5:7b).

Commandes Ollama (exemples)

```
ollama pull mistral
ollama pull qwen2.5:7b
ollama run mistral
ollama run qwen2.5:7b
```