

L'IA du juriste — Guide technique V1.0

Christophe Quézel-Ambrunaz

2026-05-19

Contents

1	Avant-propos	3
2	Vue d'ensemble	3
2.1	Présentation	3
2.2	Modèle de menace	3
2.3	Architecture logique	4
3	Prérequis	5
3.1	Matériel	5
3.2	Logiciels	5
4	Trois profils de paramétrage	5
4.1	Profil Confort — Mistral Small 3.2 24B Q5_K_M	5
4.2	Profil Standard — Mistral Small 3.2 24B Q4_K_M	6
4.3	Profil Minimal — Mistral 7B Q4_K_M	6
5	Structure du pack	6
6	Détail des trois modules	7
6.1	Module SAS	7
6.1.1	Architecture interne	7
6.1.2	Intégration Open WebUI	7
6.1.3	Limites de tokenisation	7
6.2	Module Coffre-fort RAG	7
6.2.1	Architecture interne	7
6.2.2	Paramètres critiques de la pipeline RAG	8
6.2.3	Pièges connus	8
6.3	Module MCP supervisé	9
6.3.1	Architecture interne	9
6.3.2	Mécanisme <i>plan-then-execute</i>	9
6.3.3	Compression de payload V0.3.5	9
6.3.4	Audit	9
7	Installation manuelle	10
7.1	Étapes générales	10
7.2	Modèles Ollama — procédure Q5	10
7.3	Variables d'environnement	11
7.4	Profils Docker Compose	12

8 Configuration Open WebUI	12
8.1 Création du compte administrateur	12
8.2 Paramètres globaux	12
8.3 Création des Modèles personnalisés	12
8.4 Tool server MCP	12
9 Mise à jour	13
9.1 Mise à jour du pack lui-même	13
9.2 Mise à jour des corpus	13
10 Diagnostic et journaux	13
10.1 Journaux Docker	13
10.2 Journaux Ollama	13
10.3 Vérifications rapides	14
10.4 Vérification de l'isolement réseau	14
11 Sécurité et confidentialité	14
11.1 Flux réseau	14
11.2 Confidentialité des données utilisateur	14
11.3 Conformité RGPD	15
11.4 Modèle de menace résiduelle	15
12 Licence et redistribution	15
13 Annexes	16
13.1A. Architecture de référence (poste validé)	16
13.2B. Ports utilisés	16
13.3C. Procédure Ollama natif sur Windows + AMD Strix Halo	16
13.4D. Procédure Ollama natif sur macOS Apple Silicon	17
13.5E. Pièges techniques connus (synthèse)	17
13.6F. Procédure de désinstallation complète	17

1 Avant-propos

Ce guide est destiné à un public **technique** : responsable informatique de cabinet, ingénieur, consultant en infrastructure, ou avocat curieux maîtrisant les concepts d'exécution conteneurisée et de modèles de langage. Il décrit l'architecture du pack « LIA du juriste » V1.0, ses prérequis, ses points d'intégration et ses pièges connus.

Pour un usage par un avocat sans appétence technique, se reporter aux manuels par module dans docs/manuel_<module>.md ou au guide docs/manuel_avocat_pour_claude.md (à lire avec un assistant tel que Claude).

Auteur : Christophe Quézel-Ambrunaz. **Licence** : CeCILL-B v1.0 (texte intégral dans LICENSE à la racine du pack). **Contact** : contact@christopheqa.fr.

2 Vue d'ensemble

2.1 Présentation

LIA du juriste est un assistant intelligent destiné aux praticiens du droit, s'exécutant **intégralement sur le poste de travail**. Aucune donnée utilisateur n'est transmise à un service distant sans validation explicite. Trois modules indépendants peuvent être installés à la carte :

- **SAS de pseudonymisation** : substitution locale des données personnelles d'un document avant transmission à un assistant IA distant.
- **Coffre-fort RAG** : assistant juridique fondé sur les codes et la jurisprudence téléchargés localement, sans aucune sortie réseau.
- **MCP supervisé** : interrogation des bases publiques officielles (Légifrance, Judilibre via OpenLegi) avec validation explicite par l'utilisateur de chaque appel sortant.

2.2 Modèle de menace

Le pack est conçu pour fonctionner dans un cabinet d'avocat, où la confidentialité des dossiers est centrale. Les hypothèses suivantes sont explicites :

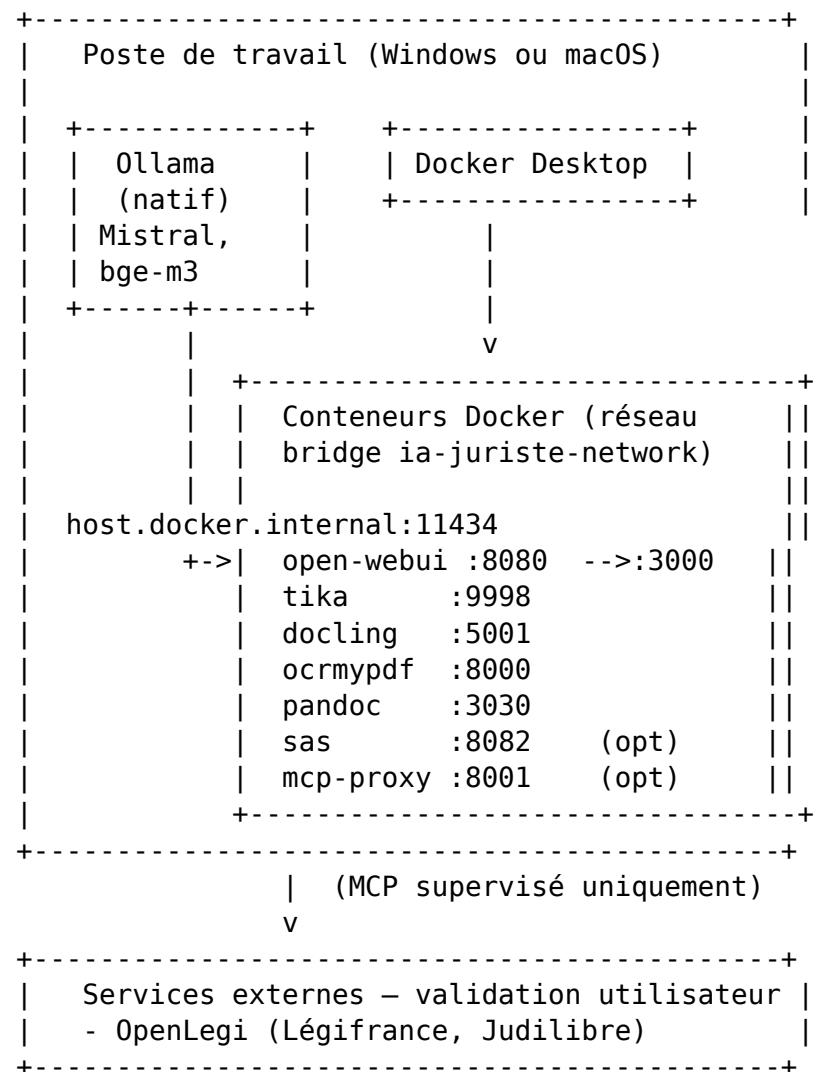
- L'utilisateur est de bonne foi mais n'est pas un expert sécurité.
- Le poste de travail est exposé aux menaces ordinaires (malware, phishing) que les outils standard (antivirus, MDM) ne suffisent pas toujours à prévenir.
- Le réseau du cabinet est supposé non hostile mais non chiffré au niveau interne.
- Les services tiers (OpenLegi, Légifrance via PISTE, Judilibre) sont considérés comme légitimes mais leur disponibilité et leur contenu sont hors du contrôle du pack.

Les mesures de défense en profondeur retenues sont :

1. **Bind 127.0.0.1 uniquement** : tous les services exposent leurs ports sur la boucle locale. Aucun port n'est ouvert sur l'interface réseau du poste.
2. **Validation manuelle de chaque appel MCP** : aucune sortie réseau n'est émise par le module MCP sans clic explicite de l'utilisateur sur la page de validation locale.

3. **Pseudonymisation locale obligatoire** avant tout envoi externe (assistants IA distants) — flux non automatisé : c'est l'utilisateur qui copie-colle le résultat dans l'outil distant.
4. **Volumes Docker isolés** : les données utilisateur (sessions SAS, corpus indexés, audit MCP) résident dans des volumes nommés gérés par Docker, non accessibles directement par les autres applications.

2.3 Architecture logique



Le pack s'appuie sur :

- **Ollama** en natif sur le poste, hébergeant les modèles de langage (Mistral Small 3.2) et d'embedding (bge-m3). Exposé sur 127.0.0.1:11434. Sur Windows, l'exécution native exploite Vulkan (Strix Halo, GPU AMD) ou CUDA (NVIDIA discret). Sur macOS Apple Silicon, Metal natif.
- **Docker Desktop** orchestrant le reste de la stack via Docker Compose. Les conteneurs communiquent entre eux par un réseau bridge privé ia-juriste-network et joignent Ollama via host.docker.internal:11434.
- **Open WebUI** comme interface conversationnelle (port 3000) et orchestrateur de l'embedding RAG (collections Chroma embarquées).
- **Tika, Docling, OCRmyPDF, Pandoc** comme services d'extraction / conver-

sion de documents, sollicités par Open WebUI lors du téléversement.

- **Service SAS** (port 8082) : encodeur Privacy Filter exécuté en Python (FastAPI), avec page de relecture humaine.
- **Proxy MCP de supervision** (port 8001) : intermédiaire FastAPI entre Open WebUI (modèle MCP) et les serveurs MCP externes.

3 Prérequis

3.1 Matériel

Trois profils de paramétrage selon les caractéristiques de la machine. L'installateur effectue la détection automatique et propose le profil correspondant ; l'utilisateur peut surcharger.

Profil	RAM	VRAM / mémoire unifiée	Disque libre	Modèle principal
Confort	≥ 32 Go	≥ 24 Go	≥ 50 Go	Mistral Small 3.2 24B Q5_K_M
Standard	≥ 16 Go	≥ 8 Go	≥ 30 Go	Mistral Small 3.2 24B Q4_K_M
Minimal	≥ 8 Go	(CPU / iGPU)	≥ 20 Go	Mistral 7B Q4_K_M

En deçà de 8 Go de RAM, l'installateur s'arrête.

Plateformes validées :

- AMD Ryzen AI Max+ 395 (« Strix Halo »), Radeon 8060S 32 Go VRAM, Windows 11, backend Vulkan via Ollama natif.
- Mac M-series 32 Go+ unifié, Metal via Ollama natif.
- PC Windows + GPU NVIDIA discret 16 Go+ VRAM (RTX 4080 / 5080), CUDA via Ollama natif.

3.2 Logiciels

- **Docker Desktop** 4.30 ou supérieur (Windows / macOS). Sur Windows, accepter l'option WSL 2 à l'installation.
- **Ollama** 0.4 ou supérieur, installation native (pas dans Docker, sauf cas exceptionnel décrit en annexe).
- **PowerShell 5.1** (intégré à Windows 10/11) ou **bash 3.2** (intégré à macOS) pour l'exécution de l'installateur.

4 Trois profils de paramétrage

4.1 Profil Confort — Mistral Small 3.2 24B Q5_K_M

Pour les postes haut de gamme (32 Go RAM + 24 Go VRAM ou plus). Modèle quantifié en Q5_K_M, soit ~ 17 Go sur disque, exécuté entièrement en mémoire vidéo. La quantification Q5 offre un écart de perplexité d'environ 1 à 2 % par rapport à FP16, contre 3 à 5 % en Q4. La latence est légèrement supérieure (+ 12 %) mais le saut qualitatif est net sur les tâches juridiques fines (citation littérale d'articles, distinction d'identifiants formellement proches, structuration de réponse complexe).

Particularité de distribution : Ollama ne publie pas officiellement les quantifications intermédiaires (Q5_K_M, Q6_K) pour Mistral Small 3.2. Le pack télécharge

le fichier GGUF depuis le dépôt Hugging Face unsloth/Mistral-Small-3.2-24B-Instruct-2506-GGUF (non gated, MIT-compatible) et l'enregistre dans Ollama via une commande `ollama create` accompagnée d'un Modelfile contenant le **TEMPLATE Mistral complet**. Sans ce TEMPLATE, le modèle ne reconnaît pas le format des messages transmis par Open WebUI et reste muet. Le script Modelfile généré est explicité au § [Modèles Ollama — procédure Q5](#).

4.2 Profil Standard — Mistral Small 3.2 24B Q4_K_M

Pour les postes 16-32 Go RAM. Modèle quantifié en Q4_K_M, ~ 15 Go, exécuté en VRAM ou en mémoire unifiée. Distribution officielle Ollama (`ollama pull mistral-small3.2:24b-instruct-2506-q4_K_M`), sans manipulation particulière.

C'est la configuration validée par la recette du 19 mai 2026, à l'exception du Coffre-fort qui était sur Q5. Le Q4 est utilisable pour les trois modules sans dégradation majeure ; les pièges juridiques fins restent un point de vigilance (voir manuel du Coffre-fort).

4.3 Profil Minimal — Mistral 7B Q4_K_M

Pour les postes 8-16 Go sans GPU dédié. Modèle Mistral 7B, ~ 4,5 Go, exécutable en CPU ou en iGPU. Latence sensiblement supérieure (5 à 10 fois celle du 24B Q4) et capacité de raisonnement nettement plus faible. Adapté principalement à la pseudonymisation SAS, où la qualité du modèle de langage importe peu (le travail est fait par Privacy Filter, un modèle distinct).

5 Structure du pack

```
ia-juriste-v1/
├── README.md                      # Page d'accueil
├── LISEZ-MOI-AVANT-INSTALLATION.txt # Procédure Windows et macOS
├── LICENSE                        # CeCILL-B v1.0
├── AUTHORS.md                    # Auteurs et composants tiers
├── CHANGELOG.md                  # Historique des versions
├── version.json                  # Métadonnées
├── installer.ps1                 # Installateur Windows
├── installer.sh                  # Installateur macOS
├── .env.example                  # Modèle de configuration
├── docs/                         # Documentation utilisateur
├── stack/                        # Tout ce qui entre dans Docker
│   ├── docker-compose.yml
│   ├── functions/                # Fonctions Python pour Open WebUI
│   ├── prompts_seed/            # Prompts pré-installés (lecture seule)
│   │   ├── templates/           # Templates DOCX pour Pandoc
│   │   ├── ocrmypdf-server/     # Service OCR
│   │   ├── pandoc-server/       # Service Pandoc
│   │   ├── sas/                 # Module SAS (optionnel)
│   │   ├── proxy/               # Module MCP supervisé (optionnel)
│   │   └── modelfiles/          # Modelfiles Ollama
├── corpus-watcher/              # Pipeline d'ingestion des corpus
└── prompts/                     # Sources de vérité des prompts
```

6 Détail des trois modules

6.1 Module SAS

6.1.1 Architecture interne

Le SAS expose une **API OpenAI Chat Completions** sur le port 8082, ce qui permet à Open WebUI de s'y connecter via la même interface qu'un serveur OpenAI tiers. Deux modèles fictifs y sont exposés :

- `sas-pseudo-v22` : pseudonymisation d'un document.
- `sas-depseudo-v22` : re-substitution des pseudonymes par les valeurs d'origine à partir d'un identifiant de session.

Le service est implémenté en Python avec FastAPI + Uvicorn. Le moteur de pseudonymisation est **OpenAI Privacy Filter** (Apache 2.0, ~ 1,5 milliard de paramètres, encodeur bidirectionnel) chargé via la bibliothèque transformers et exécuté en CPU. Il identifie les entités nommées (noms, dates, adresses, identifiants administratifs et financiers) avec une précision variable selon le motif.

6.1.2 Intégration Open WebUI

L'intégration ne peut **pas** se faire par une simple connexion OpenAI externe : Open WebUI applique sa pipeline RAG en amont (extraction Tika, chunking, embedding), ce qui corrompt le document avant qu'il n'atteigne le SAS. La solution est une **Pipe Function** (`sas_v22_pipe.py`) qui court-circuite la pipeline RAG en lisant directement le contenu du message utilisateur via le mécanisme `__metadata__['user_message']`.

6.1.3 Limites de tokenisation

Cinq motifs récurrents d'erreur identifiés lors de la recette du 19 mai 2026 :

1. Fragmentation des numéros de téléphone en plusieurs pseudonymes, dont un peut être catégorisé à tort comme « date ».
2. Dates partielles : un masquage incomplet (07/) peut laisser 02/2026 lisible.
3. Identifiants partiels : URG-23045-1 peut voir seulement URG- masqué.
4. Toponymes scindés (val masqué, -de-Marne conservé).
5. Médecins abrégés non détectés dans le corps de texte.

Ces motifs **justifient l'obligation de relecture humaine systématique**.

6.2 Module Coffre-fort RAG

6.2.1 Architecture interne

Le Coffre-fort n'est pas un service Docker distinct. C'est une **configuration applicative** :

- Modèle de langage Mistral Small 3.2 (Q5/Q4) servi par Ollama natif.
- Modèle d'embedding bge-m3 également servi par Ollama.
- Reranker BAAI/bge-reranker-v2-m3 servi par Open WebUI (téléchargé à la volée par sentence-transformers).
- Index vectoriel + lexical (ChromaDB + BM25) embarqué dans Open WebUI.

- Prompt système structuré (prompts/coffre-fort.md) imposant un étiquetage [SOURCE RAG : collection/fichier] ou [CONNAISSANCE GÉNÉRALE] à chaque énoncé.
- Pipeline corpus-watcher (Python autonome) pour la mise à jour différentielle des corpus.

6.2.2 Paramètres critiques de la pipeline RAG

Réglages à appliquer dans Admin > Réglages > Documents après création du compte admin Open WebUI. Tableau récapitulatif :

Paramètre	Valeur cible	Justification
Moteur d'extraction	Docling	Meilleur que Tika sur PDF complexes
Ignorer Embedding et Retrieval	OFF	Toggle piégeux — injecte tout le corpus, sature le contexte, fait halluciner du code Python
Découpage par en-têtes markdown	OFF	Sinon découpe les articles de code en chunks aveugles (titre / corps)
Taille des chunks	1000	
Chevauchement	100	
Taille du lot d'embedding	32	Défaut 1 = aberrant, multiplie le temps d'attachement par 6
Traitement asynchrone	ON	Défaut OFF = bloquant
Recherche hybride	ON	BM25 + dense, indispensable pour les références numériques
Reranker	BAAI/bge-reranker-v2-m3	Indispensable — son retrait dégrade fortement la qualité
Top K	8	
Top K Reranker	3	

6.2.3 Pièges connus

- **function_calling = native** sur un Modèle avec Knowledge attachée casse l'auto-injection RAG par Open WebUI. Le modèle hallucine alors. Conserver default.
- **Capacité « Citations »** doit être **ON** sur le Coffre-fort. Désactivation involontaire constatée après plusieurs sauvegardes successives du Modèle.
- **Pathologie d'agrégation des grosses Knowledges** : Open WebUI 0.9.5 agrège imparfaitement les résultats sur les Knowledges de plusieurs milliers de fichiers indexés par script API. Symptôme : 0 source remontée en chat alors que la Knowledge contient l'arrêt pertinent. Remédiation : réindexation via l'écran d'admin, ou ingestion par drag-drop UI au lieu de l'API.

6.3 Module MCP supervisé

6.3.1 Architecture interne

Service `mcp-supervision-proxy` (port 8001) implémenté en FastAPI. Il intercepte les appels MCP émis par Open WebUI et les met en file d'attente jusqu'à validation explicite de l'utilisateur.

Endpoints principaux :

- `POST /plan/submit` — appelé par le LLM via OpenAPI. Soumet un plan d'appels (un ou plusieurs) à valider en bloc.
- `GET /p/<plan_id>` — page web de validation du plan.
- `POST /v/<plan_id>/decision` — réception de la décision utilisateur (autorisé / refusé / sélection partielle).
- `GET /queue` — page web de la file d'attente, avec SSE pour les notifications en temps réel.
- `GET /openapi/openlegi.json` — spec OpenAPI exposée à Open WebUI pour déclarer le tool server.

6.3.2 Mécanisme *plan-then-execute*

Plutôt qu'une validation par appel individuel (qui multipliait les clics pour une question complexe), le LLM annonce un **plan** de K appels dans son texte, le proxy reçoit ce plan, l'utilisateur valide en bloc, le proxy exécute séquentiellement et retourne les résultats au LLM en une seule passe. Quand un seul appel est nécessaire, le comportement est identique à la validation par appel ; quand plusieurs sont nécessaires, on économise des validations.

6.3.3 Compression de payload V0.3.5

Les retours MCP peuvent atteindre plusieurs centaines de Ko (panorama jurisprudentiel triangulé sur 3 appels x N décisions). Sans compression, le contexte de Mistral Q5 (`num_ctx` = 16384) est saturé silencieusement, le modèle reçoit un payload tronqué et produit une prose vide ou incohérente.

Le proxy V0.3.5 applique deux mesures :

1. Suppression du champ `sources` par appel (les URL canoniques sont extraites au niveau racine du plan et conservées une seule fois).
2. Troncature du champ `result.content[*].text` à 5 000 caractères par item, avec une note explicite en queue de troncature (« tronqué — N caractères omis ; consulter `get_decision_*/recherche ciblée` »).

Gain de compression mesuré : 71,5 % sur 3 panoramas (de ~ 16 K à 4,6 K tokens), marge confortable restaurée sous `num_ctx`.

6.3.4 Audit

Chaque appel MCP émis est journalisé dans `mcp_audit` (volume Docker), au format JSON, avec :

- horodatage,
- question d'origine,
- modèle ayant émis le plan,

- détail des appels (serveur, fonction, paramètres),
- décision utilisateur (autorisé / refusé / sélection),
- réponse reçue.

Purge automatique configurable (30 jours par défaut, paramétrable dans `stack/proxy/config.yaml`).

7 Installation manuelle

Cette section est destinée aux opérateurs qui préfèrent ne pas exécuter l'installateur automatique, ou qui souhaitent en comprendre les étapes internes pour adapter le déploiement.

7.1 Étapes générales

1. Décompresser le pack dans un dossier (par exemple `Documents\IA-juriste`).
2. Installer Docker Desktop (cf. [Prérequis logiciels]) et démarrer l'application.
3. Installer Ollama natif (cf. [Prérequis logiciels]).
4. Télécharger les modèles Ollama :
 - `ollama pull bge-m3:latest`
 - `ollama pull mistral-small3.2:24b-instruct-2506-q4_K_M` (Standard)
 - ou suivre la procédure Q5 ci-dessous.
5. Copier `.env.example` en `.env` à la racine du pack et compléter les variables nécessaires (cf. §[Variables d'environnement](#)).
6. Depuis le dossier `stack/`, exécuter :


```
docker compose --env-file ../.env build
docker compose --env-file ../.env up -d
```
7. Ouvrir `http://localhost:3000` dans le navigateur et créer le compte administrateur Open WebUI.
8. Configurer Open WebUI selon §[Configuration Open WebUI](#).

7.2 Modèles Ollama — procédure Q5

```
mkdir -p ~/.ollama-gguf
cd ~/.ollama-gguf
```

1) Téléchargement du GGUF (16-17 Go)

```
curl -L -o Mistral-Small-3.2-24B-Instruct-2506-Q5_K_M.gguf \
  https://huggingface.co/unsloth/Mistral-Small-3.2-24B-Instruct-2506-GGUF/resolve/main/Mistral-Small-3.2-24B-Instruct-2506-Q5_K_M.gguf
```

2) Création d'un Modelfile avec TEMPLATE Mistral

```
cat > Modelfile-q5 <<'EOF'
FROM ./Mistral-Small-3.2-24B-Instruct-2506-Q5_K_M.gguf
```

```
TEMPLATE """{{- range $index, $_ := .Messages }}
{{- if eq .Role "system" }}[SYSTEM_PROMPT]{{ .Content }}[/SYSTEM_PROMPT]
{{- else if eq .Role "user" }}
```

```

{{- if and (le (len (slice $.Messages $index)) 2) $.Tools }}[AVAILABLE_TOOLS]{{ $.To
{{- end }}[INST]{{ .Content }}[/INST]
{{- else if eq .Role "assistant" }}
{{- if .Content }}{{ .Content }}
{{- if not (eq (len (slice $.Messages $index)) 1) }}</s>
{{- end }}
{{- else if .ToolCalls }}
{{- range $i, $_ := .ToolCalls }}[TOOL_CALLS]{{ .Function.Name }}[CALL_ID]{{ $i }}[A
{{- end }}</s>
{{- end }}
{{- else if eq .Role "tool" }}[TOOL_RESULTS]{"content": {{ .Content }}}[/TOOL_RESULT
{{- end }}
{{- end }}""

```

PARAMETER temperature 0.15

E0F

3) Enregistrement dans Ollama

ollama create mistral-small3.2:24b-instruct-2506-q5_K_M -f Modelfile-q5

Erreur classique à éviter : un Modelfile contenant uniquement FROM <fichier.gguf> sans TEMPLATE. Sans le TEMPLATE Mistral complet ([SYSTEM_PROMPT], [INST], etc.), le modèle ne reconnaît pas le format des messages transmis par Open WebUI et reste muet. Constaté lors de la mise au point initiale, persistant tant que le TEMPLATE n'est pas ajouté.

7.3 Variables d'environnement

Variable	Description	Module	Obligatoire
COMPOSE_PROFILES	Profils Docker Compose activés (core, sas, mcp)	tous	oui
EXPORTS_DIR	Dossier hôte des fichiers exportés	tous	non (repli ./exports)
MCP_PROXY_KEY	Clé HTTP partagée Open WebUI ↔ proxy MCP	mcp	oui
OPENLEGI_TOKEN	Jeton d'accès à OpenLegi	mcp	oui (si MCP)
PISTE_CLIENT_ID	Identifiant client PISTE	corpus-watcher	non (mais alors pas de MAJ automatique)
PISTE_CLIENT_SECRET	Secret client PISTE	corpus-watcher	non (idem)
OPEN_WEBUI_API_KEY	API key Open WebUI pour ingestion programmatique	corpus-watcher	non (idem)

7.4 Profils Docker Compose

Le stack/docker-compose.yml utilise trois profils :

- core : socle obligatoire (open-webui, tika, docling, ocrmypdf, pandoc).
- sas : ajoute le conteneur sas.
- mcp : ajoute le conteneur mcp-supervision-proxy.

Combinaisons typiques :

```
COMPOSE_PROFILES=core,sas      # SAS seul
COMPOSE_PROFILES=core,sas,mcp  # SAS + MCP
COMPOSE_PROFILES=core          # Coffre-fort seul (services Docker minimaux)
```

8 Configuration Open WebUI

L'installateur **ne configure pas** automatiquement les Modèles personnalisés Open WebUI. La configuration suppose un compte admin Open WebUI qui n'existe pas avant le premier login, et l'API correspondante est instable sur 0.9.x. La configuration reste manuelle, documentée dans les manuels par module.

8.1 Création du compte administrateur

À la première ouverture de `http://localhost:3000`, Open WebUI demande de créer un compte. Choisir un mot de passe robuste. Le compte est local, les identifiants ne quittent pas le poste.

8.2 Paramètres globaux

À effectuer une fois après la création du compte admin :

Admin > Réglages > Documents — appliquer les valeurs du tableau ci-dessus §[Paramètres critiques de la pipeline RAG](#).

Admin > Réglages > Évaluations — désactiver les Modèles d'arène (brouille le sélecteur de chat).

Admin > Réglages > Modèles — masquer du sélecteur les modèles natifs non conversationnels (bge-m3, anciens alias).

8.3 Création des Modèles personnalisés

Pour chaque module installé, créer un Modèle personnalisé dans Workspace > Modèles > + Nouveau modèle. Détail dans :

- docs/manuel_sas.md § Création du Modèle personnalisé Open WebUI
- docs/manuel_coffre_fort.md § Création du Modèle personnalisé Open WebUI
- docs/manuel_mcp.md § Création du Modèle personnalisé Open WebUI

8.4 Tool server MCP

Pour le module MCP, déclarer le serveur d'outils dans Admin > Réglages > Intégrations > Serveurs d'outils > + Nouveau :

- URL : `http://mcp-supervision-proxy:8001/openapi/openlegi.json`

- Header HTTP custom : X-Proxy-Key = valeur de MCP_PROXY_KEY du .env.

9 Mise à jour

9.1 Mise à jour du pack lui-même

La V1.0 n'a pas de mécanisme automatique. Procédure manuelle :

1. Re-télécharger l'archive depuis le site du projet.
2. Arrêter la stack : `cd stack && docker compose down`.
3. **Sauvegarder** .env à la racine du pack.
4. Extraire la nouvelle archive dans le même dossier, en écrasant. Le .env n'est pas écrasé puisque l'archive ne contient que .env.example.
5. Comparer manuellement le nouveau .env.example avec le .env ; ajouter les éventuelles nouvelles variables.
6. Relancer `installer.ps1 / installer.sh` ; il détecte l'installation préexistante et propose une mise à jour.
7. Les données utilisateur (Open WebUI, SAS, corpus, MCP audit) sont préservées car elles résident dans des volumes Docker nommés.

9.2 Mise à jour des corpus

Le pipeline corpus-watcher (Python autonome) interroge périodiquement les APIs PISTE et Judilibre et applique trois opérations :

- **Phase A** — ajout d'un nouvel arrêt ou article : *upload + indexation*.
- **Phase B** — modification d'un article existant : *replace file*.
- **Politique VIGUEUR → ABROGÉ** : un article abrogé est retiré de la Knowledge et conservé localement avec étiquette [ABROGÉ].

Planification automatique :

- Windows : `corpus-watcher/install_scheduler_windows.ps1`
- macOS : `corpus-watcher/install_scheduler_mac.sh`

Exécution manuelle :

```
cd corpus-watcher/  
py corpus_watcher.py --once --config corpus.yaml --state state/state.json
```

10 Diagnostic et journaux

10.1 Journaux Docker

```
cd stack/  
docker compose logs --tail 200 open-webui  
docker compose logs --tail 200 sas  
docker compose logs --tail 200 mcp-supervision-proxy  
docker compose logs --tail 200 docling
```

10.2 Journaux Ollama

- Windows : `%LOCALAPPDATA%\Ollama\logs\server.log`
- macOS : `~/.ollama/logs/server.log`

10.3 Vérifications rapides

Santé de chaque service

```
curl -s http://localhost:11434/api/tags      # Ollama
curl -s http://localhost:3000              # Open WebUI (HTML)
curl -s http://localhost:9998/version      # Tika
curl -s http://localhost:5001/health       # Docling
curl -s http://localhost:8000/health       # OCRmyPDF
curl -s http://localhost:3030/health       # Pandoc
curl -s http://localhost:8082/health       # SAS (si module activé)
curl -s http://localhost:8001/healthz      # MCP proxy (si module activé)
```

10.4 Vérification de l'isolement réseau

Sous macOS / Linux :

```
lsof -i -P | grep -E '11434|3000|9998|5001|8000|3030|8082|8001'
```

Toutes les adresses doivent être 127.0.0.1. Aucune adresse externe ne doit apparaître pour les conteneurs ia-juriste-*.

Sous Windows : ouvrir le **Moniteur de ressources**, onglet **Réseau**, section **Ports en écoute**. Filtrer sur les ports listés ci-dessus. Seules des adresses 127.0.0.1 doivent figurer.

11 Sécurité et confidentialité

11.1 Flux réseau

Module	Sortie réseau
Socle commun	aucune
SAS	aucune
Coffre-fort	aucune (sauf téléchargements initiaux des modèles et corpus, et MAJ corpus déclenchée par l'utilisateur ou le scheduler)
MCP supervisé	sur validation explicite uniquement

11.2 Confidentialité des données utilisateur

Donnée	Emplacement	Persistance	Chiffrement
Sessions SAS (tables de correspondance)	volume Docker ia-juriste-sas-data	jusqu'à suppression manuelle	non
Documents exportés (DOCX/PDF/MD)	dossier hôte EXPORTS_DIR	persistant	non (au repos, dépend du chiffrement disque)
Compte admin Open WebUI	volume openwebui_data	persistant	mot de passe hashé bcrypt

Donnée	Emplacement	Persistance	Chiffrement
Index vectoriel des corpus	volume openwebui_data	persistant	non
Journal MCP	volume mcp_audit	30 jours (configurable)	non

Le chiffrement des tables de correspondance SAS n’apporterait pas de garantie supplémentaire en pratique : le fichier source coexiste avec le fichier pseudonymisé sur le poste, et le poste lui-même est ou n’est pas chiffré au niveau disque (BitLocker, FileVault) selon les politiques du cabinet.

11.3 Conformité RGPD

Le pack opère un traitement local de données à caractère personnel des clients de l’avocat. Le responsable de traitement est le cabinet utilisateur ; le pack n’effectue aucun traitement pour le compte de l’auteur du logiciel. Recommandations :

- Tenir à jour le registre des traitements du cabinet, en y mentionnant l’usage du pack.
- Documenter le SAS comme mesure technique de pseudonymisation au sens de l’article 4(5) du RGPD.
- Procéder à une analyse d’impact (AIPD) si le traitement le justifie.

11.4 Modèle de menace résiduelle

Le pack ne protège pas contre :

- la compromission du poste lui-même (malware, accès physique non autorisé) — relève du chiffrement disque, MDM, antivirus du cabinet ;
- l’exfiltration volontaire par l’utilisateur lui-même ;
- les vulnérabilités des composants tiers (Open WebUI, Ollama, Mistral, Docker) — relève des mises à jour régulières du pack et de l’OS.

12 Licence et redistribution

Le pack est distribué sous **licence CeCILL-B v1.0**, licence française de logiciel libre compatible avec les licences de type BSD. Le texte intégral est dans LICENSE à la racine du pack.

Points saillants :

- **Modification autorisée** sans demande préalable.
- **Redistribution autorisée**, y compris modifiée, sous réserve :
 - de préserver les mentions de propriété intellectuelle (article 5.3.4) ;
 - d’identifier la version modifiée comme telle ;
 - de citer l’auteur du Logiciel Initial sur le site Web de redistribution.
- **Usage commercial autorisé** (article 5.1).
- **Garantie limitée** : le pack est fourni « en l’état » sans engagement (articles 8 et 9).

Les composants tiers (Ollama, Open WebUI, Mistral, bge-m3, Privacy Filter, etc.) conservent leurs licences propres, listées dans `AUTHORS.md`. La redistribution de ces composants suit leurs conditions respectives, indépendamment de CeCILL-B.

13 Annexes

13.1 A. Architecture de référence (poste validé)

- AMD Ryzen AI Max+ 395 (« Strix Halo »).
- 128 Go RAM unifiée, 32 Go affectables au GPU.
- Radeon 8060S 32 Go VRAM (mémoire partagée).
- Windows 11 Pro 23H2.
- Pilote AMD Adrenalin Edition 25.1+ (livre Ollama et LM Studio en un clic).
- Backend GPU : Vulkan via Ollama natif.

13.2 B. Ports utilisés

Port	Service	Profil
11434	Ollama (natif)	toujours
3000	Open WebUI	core
9998	Apache Tika	core
5001	Docling	core
8000	OCRmyPDF	core
3030	Pandoc	core
8082	SAS	sas
8001	MCP proxy	mcp

Tous bindés sur 127.0.0.1.

13.3 C. Procédure Ollama natif sur Windows + AMD Strix Halo

Particularité de la plateforme AMD Ryzen AI Max+ : **non supportée par ROCm sur Windows** (limitation officielle AMD pour les APU Ryzen AI), mais **pleinement supportée via Vulkan** par Ollama natif.

1. Vérifier le pilote Adrenalin (≥ 25.1 , janvier 2026) :

`Get-WmiObject Win32_VideoController | Select-Object Name, DriverVersion`
Version 32.0.22018.5 ou plus récent.

2. Installer Ollama via AMD Adrenalin (recommandé) ou directement depuis <https://ollama.com/download/windows>.
3. Si installation directe, activer Vulkan :

`[System.Environment]::SetEnvironmentVariable("OLLAMA_VULKAN", "1", "User")`
Quitter Ollama et le relancer.

4. Vérifier :

`curl http://127.0.0.1:11434/api/tags`

Doit retourner {"models":[]}

- Performances mesurées : 15 à 25 tokens/seconde sur Mistral Small 24B Q4 ; 13 à 22 sur Q5.

13.4 D. Procédure Ollama natif sur macOS Apple Silicon

- Télécharger <https://ollama.com/download/mac>.
- Glisser Ollama.app dans Applications.
- Lancer une fois pour démarrer le service en arrière-plan.
- Vérifier :

```
curl http://127.0.0.1:11434/api/tags
```
- Metal est activé par défaut sur Apple Silicon. Performances typiques : 18 à 35 tokens/seconde sur Mistral Small 24B Q4 sur M3 Max 64 Go.

13.5 E. Pièges techniques connus (synthèse)

Piège	Symptôme	Remédiation
Modelfile Q5 sans TEMPLATE	Modèle muet	Fournir le TEMPLATE Mistral complet
function_calling = native avec Knowledge	Hallucinations	Basculer en default
Ignorer Embedding et Retrieval = ON	Contexte saturé, code Python halluciné	Forcer OFF
Reranker désactivé	Hallucinations, réponses non pertinentes	Conserver BAAI/bge-reranker-v2-m3
Plusieurs fichiers joints simultanément	Latence cumulative > 5 min	Règle 1 fichier / conversation
Plan MCP à 4 appels ou plus	Payload tronqué, prose vide	Limiter à 3 appels
Capacité « Citations » se ré-active à OFF sur SAS	Dump du <i>prompt template</i> RAG en sortie	Vérifier la case après chaque sauvegarde
Ingestion par API ne peuple pas l'agrégat de la Knowledge	0 source remontée en chat	Re-ingestion par drag-drop UI ou via corpus-watcher V2.2+

13.6 F. Procédure de désinstallation complète

1. Arrêter et supprimer la stack et les volumes Docker

```
cd stack/
```

```
docker compose --profile core --profile sas --profile mcp down -v
```

2. Supprimer les images locales construites

```
docker image rm ia-juriste/sas:1.0.0 ia-juriste/mcp-supervision-proxy:1.0.0 \
    ia-juriste/ocrmypdf-server:1.0.0 ia-juriste/pandoc-server:1.0.0
```

3. Supprimer les modèles Ollama

```
ollama rm mistral-small3.2:24b-instruct-2506-q5_K_M
```

```
ollama rm mistral-small3.2:24b-instruct-2506-q4_K_M
```

```
ollama rm bge-m3:latest
```

(le cas échéant)

```
ollama rm mistral:7b-instruct-q4_K_M
```

4. Supprimer le dossier d'extraction du pack

```
rm -rf /chemin/vers/ia-juriste-v1
```

*# 5. (optionnel) Désinstaller Docker Desktop et Ollama**# via les outils standard de chaque OS*

Fin du guide technique.