

PRESENTATION DE L'AGENT CONVERSATIONNEL JURIDIQUE

Nous avons choisi la méthode **CRAG (Correction-RAG)** parce qu'elle correspond parfaitement aux exigences d'un projet dans le domaine juridique. Ici, il ne suffit pas de répondre : il faut répondre juste, avec des sources fiables, adaptées au niveau de compréhension de chaque utilisateur. CRAG permet précisément cela : elle vérifie les sources, recoupe les informations, et ajuste les réponses selon le contexte de la conversation. Dans un sujet aussi délicat que le droit, c'est un vrai avantage. Cette approche réduit aussi le risque d'erreurs ou de réponses inventées, ce qui est essentiel quand on construit un assistant censé guider les gens sur des sujets sérieux.

1. Editeur , langage et environnement utilisés

Pour que le projet puisse voir le jour nous avons opter pour :

- **Python version 3** : Comme langage de programmation principale , car il est simple, accessible et parfaitement adapté aux projets en intelligence artificielle.
- **Jupyter Notebook** : comme environnement de travail ,qui un outil très pratique qui permet de coder bloc par bloc, de tester en direct et de commenter chaque étape du projet. Cela nous a permis d'avancer progressivement, tout en gardant une vue claire sur le fonctionnement du système.

L'ensemble du projet a été exécuté en local, sur un ordinateur .

2. Bibliothèques utilisées

Notre assistant juridique repose sur une combinaison intelligente de technologie telles que :









- **LangChain** : C'est le chef d'orchestre de notre système. Il organise le flux de travail, fait communiquer tous les composants entre eux et s'assure que chaque étape de la lecture des documents à la génération des réponses se déroule parfaitement .
- **FAISS** : Elle nous permet de retrouver rapidement les informations pertinentes dans un grand volume de données juridiques stockées sous forme vectorielle. Elle agit comme notre mémoire interne, capable d'identifier en un instant les textes ou extraits les plus proches d'une question.
- **Sentence-Transformers** : Charger de convertir les phrases juridiques en vecteurs numériques compréhensibles par la machine, facilitant ainsi la recherche sémantique et la comparaison de sens entre les textes.
- **PyMuPDF** : Charger de lire minutieusement chaque document PDF, même les plus complexes, et en extrait les informations utiles sans se tromper de ligne ni de paragraphe.
- **Google-GenerativeAI** : Grâce à cette bibliothèque, notre assistant est capable de comprendre les questions, raisonner à partir des textes juridiques et générer des réponses claires et adaptées. C'est l'intelligence derrière la conversation.
- **Streamlit** : C'est l'interface. Elle rend notre solution accessible via une application web simple, fluide et réactive, pour une interaction directe avec l'utilisateur, sans friction.
- **Logging** enregistre toutes les actions du système pour faciliter le dépannage et le suivi des performances.

- **Concurrent Futures** accélère le traitement en lisant plusieurs fichiers PDF en parallèle.

Toutes ces bibliothèques ont été installées via la commande suivante :

```
"pip install langchain langchain-community faiss-cpu sentence-transformers pymupdf google-generativeai streamlit"
```

3. Description des fichiers et dossiers

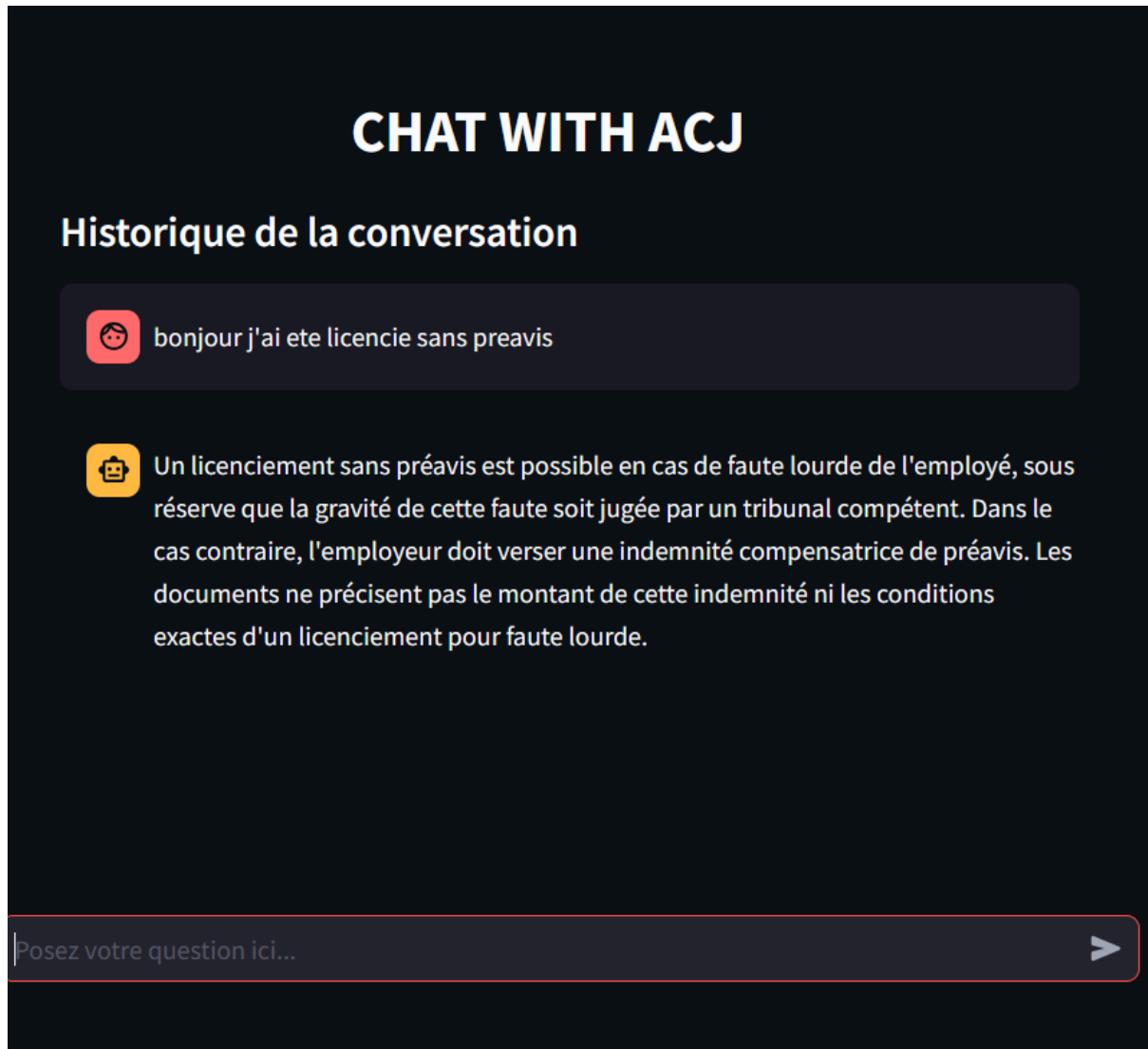
 Untitled	24/07/2025 22:58	Fichier source Jupy...	3 Ko
 moteur_recherche	09/08/2025 14:12	Fichier source Pyth...	3 Ko
 interface	04/08/2025 01:42	Fichier source Pyth...	2 Ko
 generer_reponse	09/08/2025 14:15	Fichier source Pyth...	3 Ko
 exigences	24/07/2025 19:35	Text Document	1 Ko
 evaluateur	09/08/2025 14:43	Fichier source Pyth...	6 Ko
 creer_vecteurs	09/08/2025 14:11	Fichier source Pyth...	3 Ko
 donnees	24/07/2025 19:26	Dossier de fichiers	
 base_vecteurs	24/07/2025 20:14	Dossier de fichiers	

Voici le rôle de chaque fichier et dossier composant le projet :

- **exigences.txt** : Fichier texte contenant la liste des bibliothèques nécessaires pour exécuter le projet
- **donnees/** : Dossier contenant les documents textes juridiques à indexer
- **base_vecteurs/** : dossier qui stocke la base vectorielle générée à partir des documents
- **creer_vecteurs.py** : script pour créer les vecteurs à partir des fichiers dans donnees/
- **moteur_recherche.py** : Fichier contenant les fonctions de recherche de documents les plus pertinents
- **generer_reponse.py** : script principal qui interroge le modèle pour produire une réponse
- **interface.py** : permet de lancer une interface utilisateur pour tester le chatbot , via la commande "**streamlit run [interface.py](#)**" dans le terminal pour interagir directement avec le chatbot, poser des questions et visualiser les réponses en temps réel.
- **notebook_test.ipynb** : utilisé pour tester le pipeline par blocs (lecture, vecteurs, réponse)

- **evaluateur.py** analyse si les documents trouvés sont vraiment utiles pour répondre à la question et peut reformuler la question si nécessaire.

4. Démonstration Visuelle : Dialogue avec l'Agent Juridique



Pour mieux comprendre comment fonctionne notre assistant, imaginons une situation concrète.

Un utilisateur se connecte à l'application et pose une question simple, comme :
« Bonjour j'ai été licencié sans préavis ? »

À partir de là, tout se met en marche :

- L'agent comprend la question grâce à son moteur de traitement du langage.
- Il parcourt les textes juridiques qu'on lui a fournis.
- Il identifie les passages les plus pertinents.

- Il vérifie la cohérence de sa réponse.
- Et enfin, il formule une réponse claire, concise et adaptée.

Et notre agent réponds :

"Un licenciement sans préavis est possible en cas de faute lourde de l'employé, sous réserve que la gravité de cette faute soit jugée par un tribunal compétent. Dans le cas contraire, l'employeur doit verser une indemnité compensatrice de préavis. Les documents ne précisent pas le montant de cette indemnité ni les conditions exactes d'un licenciement pour faute lourde."

Le tout, en citant la bonne loi et en expliquant les choses dans un langage accessible à tous.

5. L'impact concret de notre solution

Le droit, c'est sérieux. Et pourtant, pour beaucoup, il reste difficile à comprendre ou à consulter.

Avec ce projet, on voulait justement changer ça. On voulait :

- Aller à l'essentiel, sans perdre l'utilisateur dans des centaines de pages.
- Répondre vite, même quand la question est compliquée.
- Proposer une interface simple et facile à l'utiliser.

Ce n'est pas un agent qui débite des articles de loi. C'est un outil pensé pour accompagner, expliquer et éclairer.

Et demain, on peut aller encore plus loin : se connecter à des bases officielles, suivre les mises à jour des textes, ou même adapter le ton de la réponse à chaque profil d'utilisateur.

EN RÉSUMÉ

Ce projet, c'est notre façon de rendre le droit un peu plus humain. On ne remplace pas les juristes, on ne joue pas au juge. On crée juste un assistant intelligent, qui aide à mieux comprendre les règles, à trouver des réponses fiables, et à se sentir moins seul face à une question juridique. Grâce à des outils puissants comme LangChain, FAISS ou les modèles génératifs de Google, on a réussi à bâtir un système solide, mais surtout utile. Et ça, c'était le vrai but du projet.