

# PROJET POLYCHAT

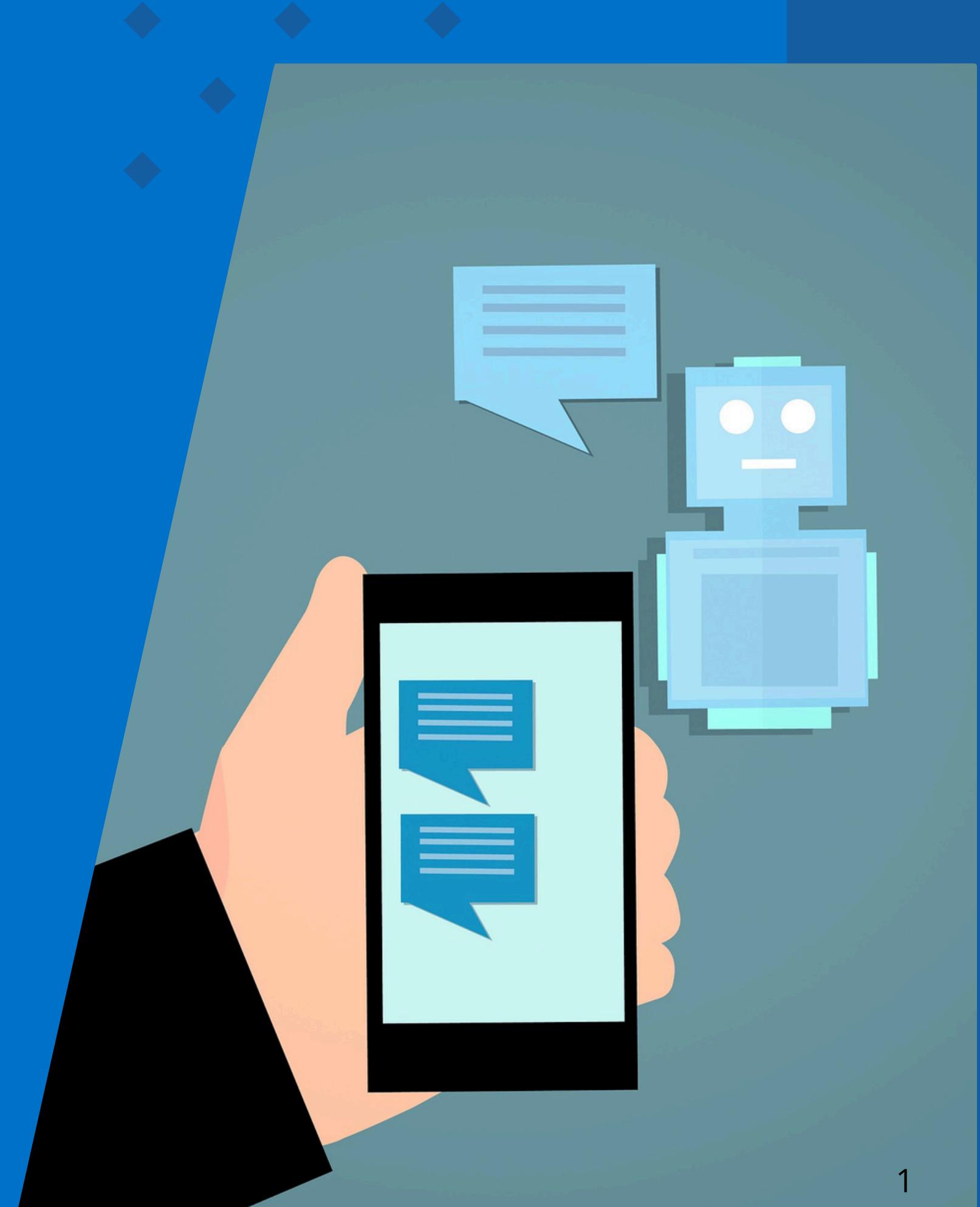
*Samuel AVAH BELOBO-OHANA*

*Tristan CHENAILLE,*

*Amalia SEYDOU*

*Yannick ZHANG*

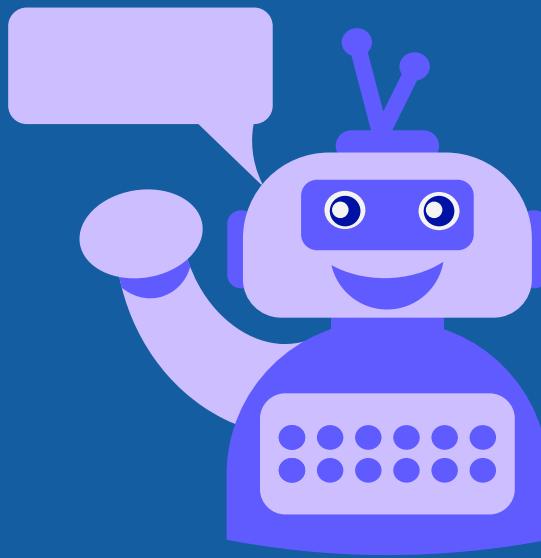
*Projet encadré par M.TANNIER*



# INTRODUCTION

## CONTEXTE

IMPLÉMENTATION D'UN  
CHATBOT POUR LE SITE DE  
POLYTECH SORBONNE



## OBJECTIFS

- Réponses correctes
- Réactivité rapide
- Interface friendly

## TECHNOLOGIES UTILISÉES



OpenAI

Chroma

Render

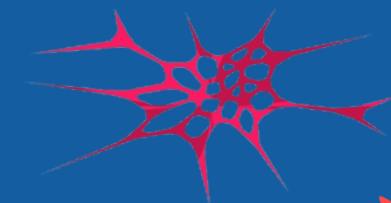
OVH



LangChain



NOMIC



Streamlit

# SOMMAIRE

01

Structure du code

02

Web Scraping

03

Vector Base

04

RAG

05

Déploiement

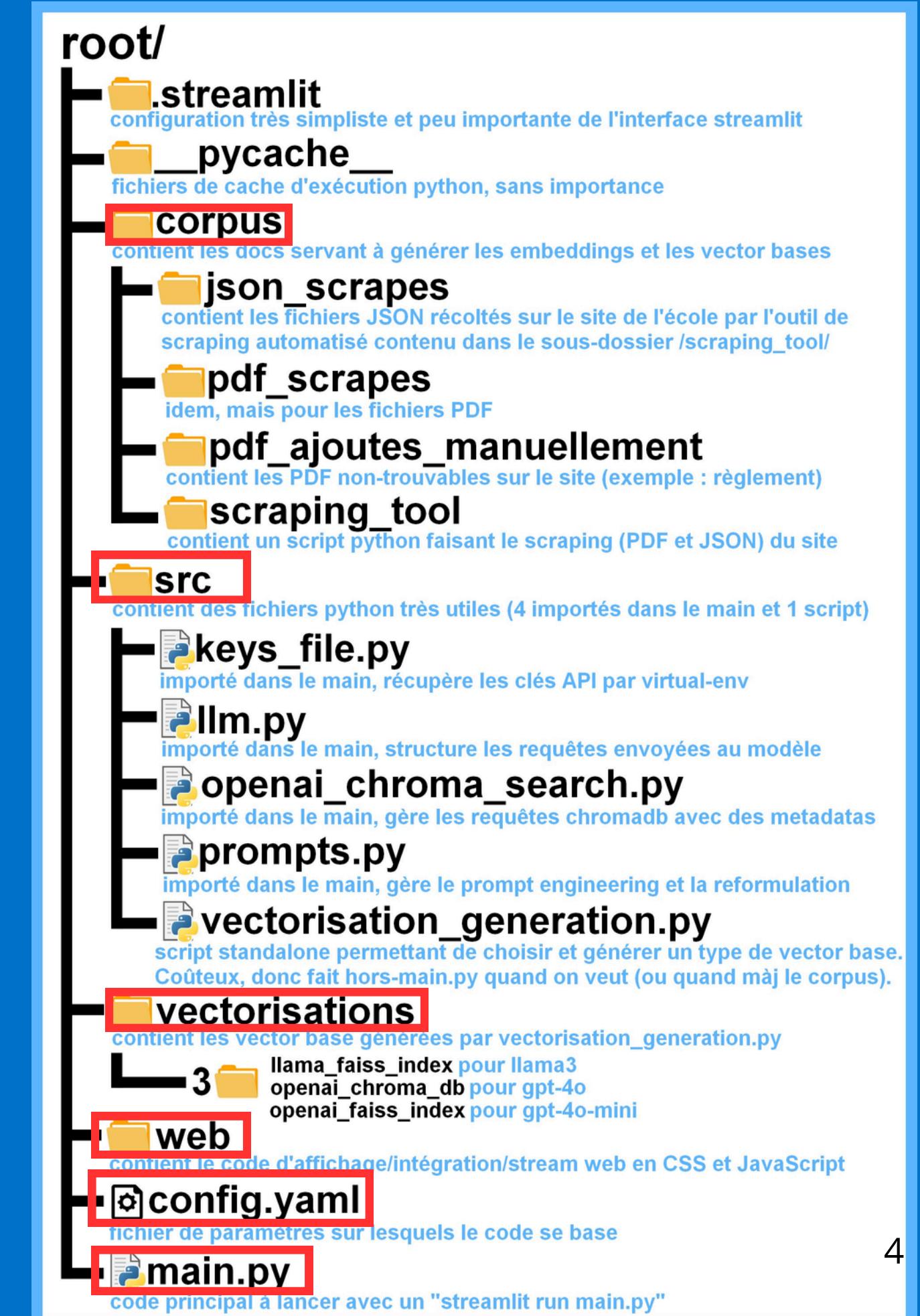
06

Conclusion

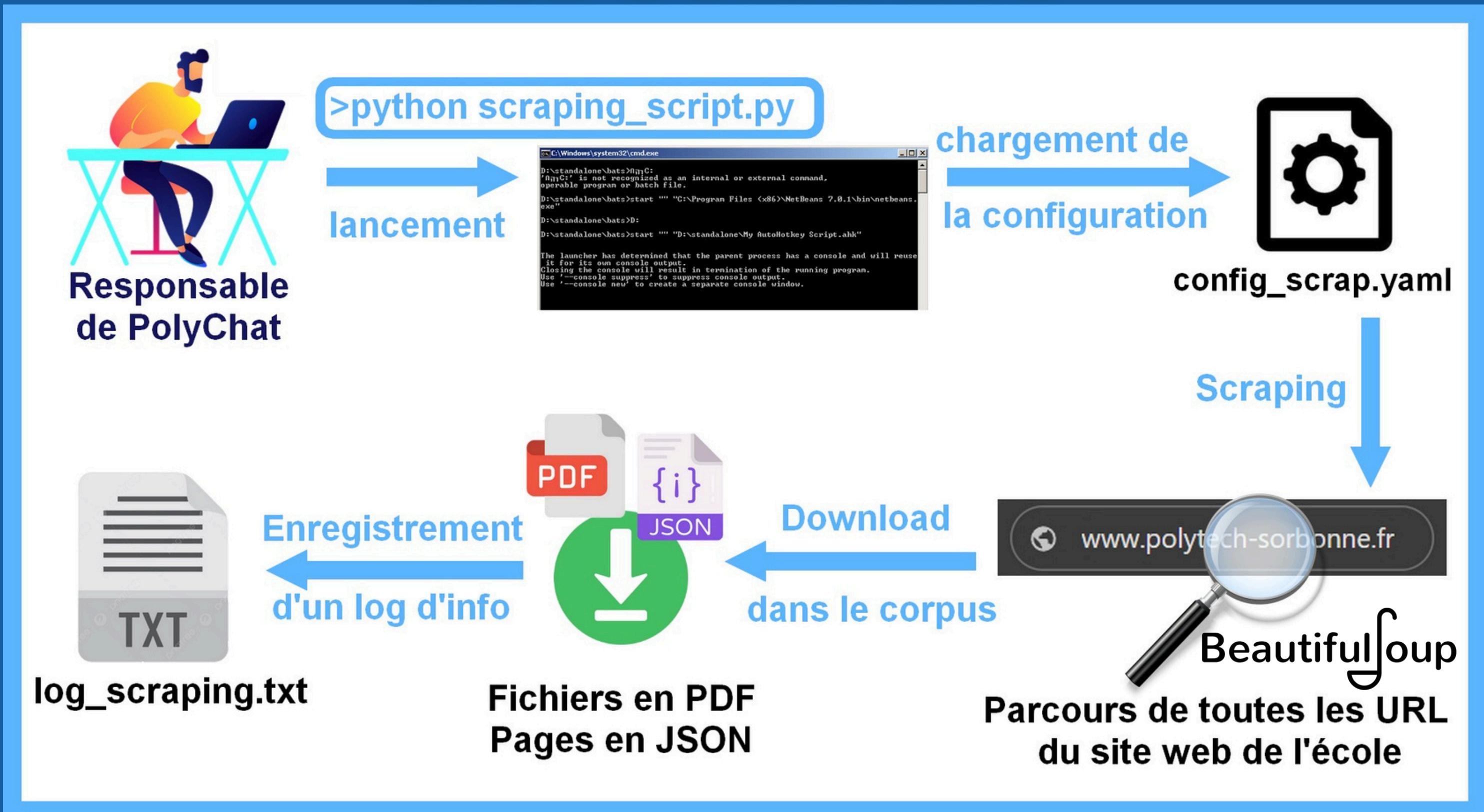


# ARCHITECTURE GLOBALE DU CODE

P I corpus/  
A M  
R P src/  
T O  
I R vectorisations/  
E T web/  
S A main.py + config.yaml



# WEB SCRAPING



# LOGS DU WEB SCRAPING

## Log d'un premier scrap (corpus vide)

```
=====
Le dernier scraping en date a été effectué le 18/02/2025 à 22:12
Nombre de PDF téléchargés ou mis à jour : 42
Nombre de JSON téléchargés ou mis à jour : 47
Attention : parfois, les fichiers téléchargés/mis à jour le sont
par précaution et non pas parce qu'ils sont nouveaux.
=====
```

## Log d'un deuxième scrap (corpus déjà rempli)

```
=====
Le dernier scraping en date a été effectué le 18/02/2025 à 22:17
Nombre de PDF téléchargés ou mis à jour : 2
Nombre de JSON téléchargés ou mis à jour : 9
Attention : parfois, les fichiers téléchargés/mis à jour le sont
par précaution et non pas parce qu'ils sont nouveaux.
=====
```

# GENERATION DES VECTOR BASES



script : src/vectorisation\_generation.py

## FAISS

- gpt-4o mini
- Hermes-3-8B



## CHROMADB

- gpt-4o



# FAISS

gpt-4o mini et Hermes-3-8B



## ✓ Ce qu'on a fait :

- Récupération des pdf dans le corpus
- Génération d'embeddings à partir de documents PDF
  - OpenAIEmbedding pour gpt-4o mini
  - nomic-embed-text-v1-5-vxq pour Hermes-3-8B
- Intégration avec GPT-4o mini et Hermes-3-8B pour améliorer les réponses.

## Pourquoi FAISS ?

- Permet une recherche rapide et efficace sur un grand volume de données.
- Optimisé pour les recherches par similarité.

## Inconvénients :

- ✗ Stockage volumineux
- ✗ Dépendance aux embeddings et moins efficace pour des vecteurs de mauvaises qualités
- ✗ Pas de gestion native des métadonnées et ne supporte donc pas de recherche avancée comme les filtres textuels (contrairement à ChromaDB).



+

1 – Récupération des fichiers json et pdf dans le corpus.

2 – Conversion des fichiers en vecteurs avec ajout des méta-données

Home/ ... / corpus/(json ou pdf) / (spécialité) / (FILE)

Le(s) vecteur(s) associé(s) au fichier FILE va avoir pour méta données :

Status : (publique ou privé)

Speciality : (spécialité)

URL : (l'URL renseigné dans FILE si c'est un json)

ID : un identifiant

3 – Ajout dans le store :

Dans l'interface du script vectorisation\_generation.py, lorsqu'on sélectionne gpt-4o, la chromadb est automatiquement générée et prête à être utilisée.

---

NB : Attribut Privé/Publique :

Faute de documents privés, le corpus actuel n'intègre pas de dossier /privé/.

Néanmoins le code le supporte parfaitement, et cette fonctionnalité est donc bien prête à l'emploi. Elle pourra être utilisée lorsque cela sera nécessaire.

# LOGS DE GENERATION DES VECTOR BASES

Exemple de log (même logique que pour le scrap)

```
=====
Cette vectorisation a été générée le 19/02/2025 à 20:23

Fichiers JSON utilisés :

- acces_polytech_sorbonne.json
- agroalimentaire.json
- bdac_polytech_sorbonne.json
- genie_mecanique.json
- qui_contacter_polytech_sorbonne.json

Fichiers PDF utilisés :

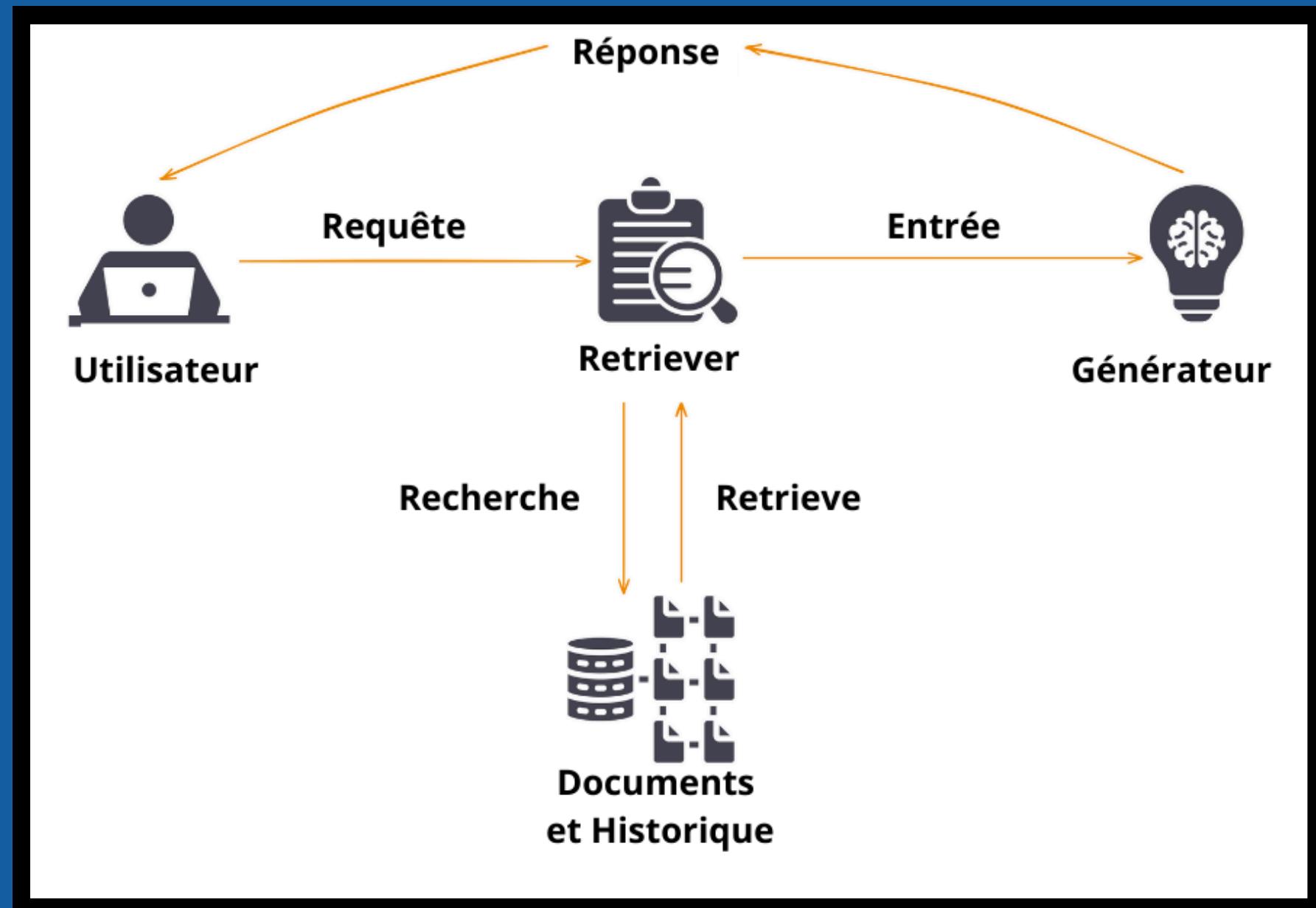
- 2022_Enquete_emploi.pdf
- Enquete_emploi_2024_version_diffusable_v4_0.pdf
- Maquette_MTX_2024_2025_avril2024_web.pdf
- Maquette_ST_2022-23_officielle_modif_3.pdf
- Poster_du_projet_-capgemini.pdf
- reseauPolytech_GuideAdmissions.pdf
- admissions_etudiants_etrangers.pdf
- programme_detaille_filiere_MAIN.pdf
- reglement_des_etudes.pdf

=====
```

(simple exemple, en réalité le corpus est davantage fourni)

# RAG

## Retrieval Augmented Generation



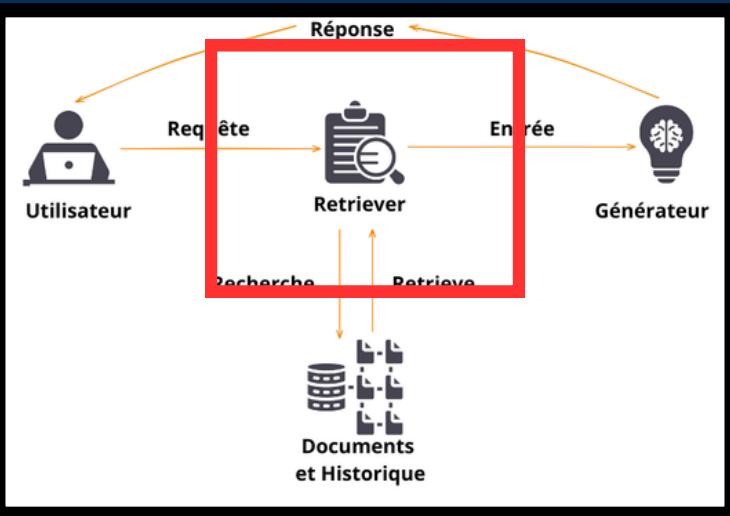
# RAG

## Prompts

- system\_prompt
- qa\_prompt
- contextualize\_q\_prompt

## Châînes

- create\_history\_aware\_retriever
- create\_stuff\_documents\_chain
- create\_retrieval\_chain



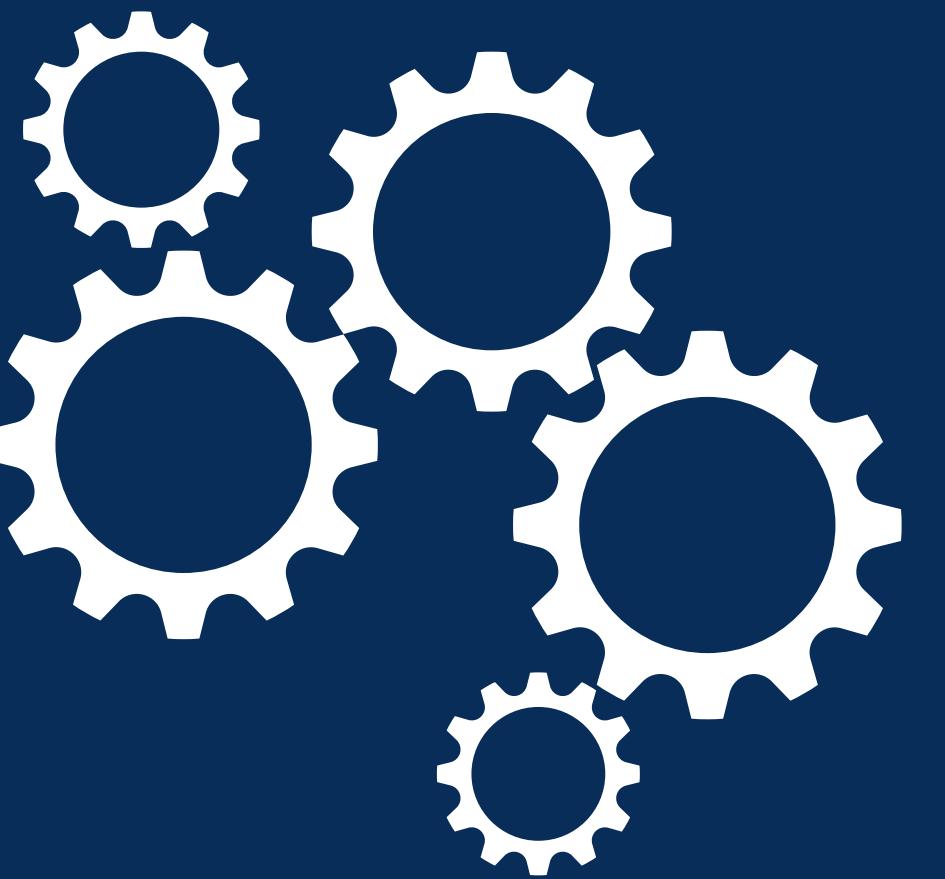
## Options LLM

- Gpt4o
- Gpt4o-mini
- Hermes-3-8B

# RAG

## Prompts

- system\_prompt



Comportement global

# RAG

## Prompts

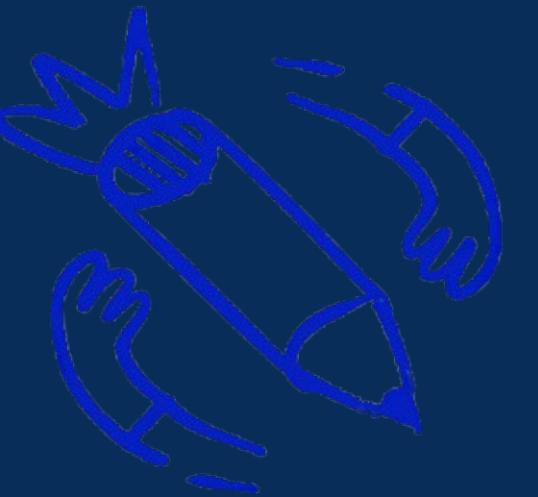
- qa\_prompt

```
qa_prompt = ChatPromptTemplate.from_messages(  
    [  
        ("system", system_prompt),  
        MessagesPlaceholder("chat_history"),  
        ("human", "{input}"),  
    ]  
)
```

# RAG

## Prompts

- contextualize\_q\_prompt



## Reformuler

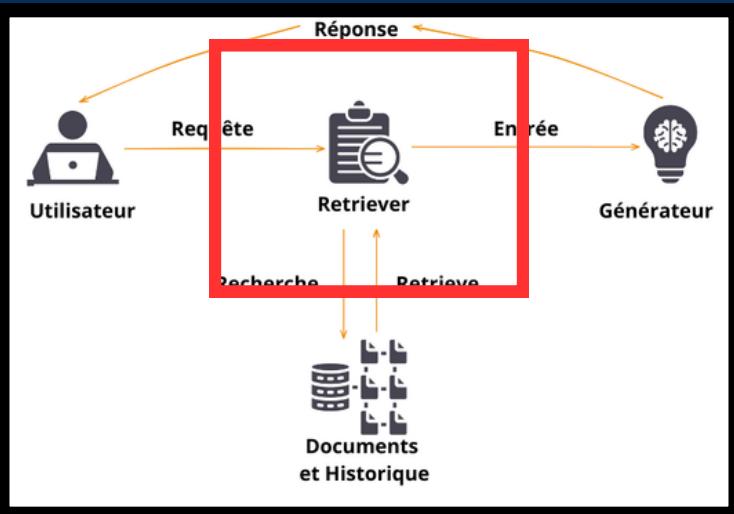
# RAG

## Prompts

- system\_prompt
- qa\_prompt
- contextualize\_q\_prompt

## Châînes

- create\_history\_aware\_retriever
- create\_stuff\_documents\_chain
- create\_retrieval\_chain



## Options LLM

- Gpt4o
- Gpt4o-mini
- Hermes-3-8B

# DÉPLOIEMENT

L'interface Streamlit est personnalisée grâce à l'intégration de styles CSS dans le code Python du main

Un script JS met instantanément à jour l'iframe pour y inclure un user\_id permettant une gestion de l'historique

Ce script gère aussi le behavior onclick



# DÉMONSTRATION EN DIRECT

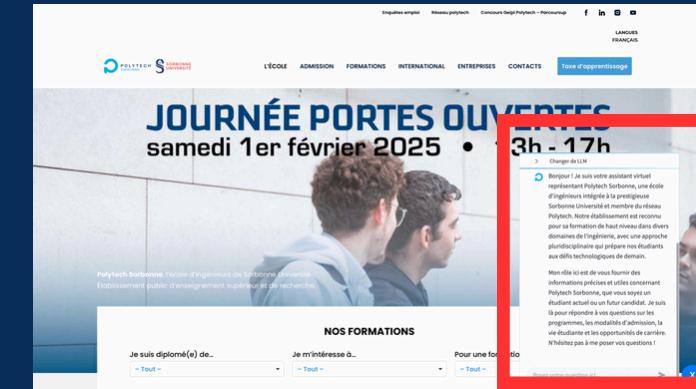


Voir le partage d'écran pour la démo

# CONCLUSION

## Résultats

- Chatbot opérationnel
- Interface plaisante
- Prototype interactif et paramétrable
- Compatibilité multi-modèles



## Limites

- Coûts computationnels et énergétiques élevés pour des LLM LLaMA
- Recours (et donc dépendance) à des serveurs externes
- Le déploiement web n'est pas automatique

## Perspectives

- Évaluation poussée pour sélectionner une logique de RAG (FAISS sans metadata ou ChromaDB avec metadata)
- Évaluation du coût effectif d'utilisation d'un modèle OpenAI
- Amélioration de l'interface utilisateur en responsive
- Amélioration, nettoyage et mise à jour du site web
- Déploiement sur le site web de l'école
- Stockage et data-analyse des requêtes des utilisateurs

MERCI POUR  
VOTRE ATTENTION