Chatbot RAG pour l'Assistance Académique – FSO

Ce projet implémente un chatbot intelligent basé sur l'architecture RAG (Retrieval-Augmented Generation) pour répondre aux questions des étudiants souhaitant intégrer la Faculté des Sciences d'Oujda (FSO) en se basant sur les documents officiels.

Architecture

Le projet est composé de deux parties principales :

1. **Backend (Python/Flask) :** Un serveur qui expose une API pour le moteur RAG. Il gère l'extraction de texte, la génération d'embeddings, la recherche de similarité et la génération de réponses via l'API d'OpenAI.
2. **Frontend (React/Vite) :** Une interface utilisateur web moderne qui permet aux utilisateurs d'interagir avec le chatbot.

🚀 Lancement du Projet

Pour lancer l'application, vous devez démarrer le backend et le frontend séparément.

1. Démarrage du Backend (Serveur API)

Le backend est un serveur Flask qui écoute sur le port `5000`.

1. **Ouvrez un terminal** à la racine du projet.
2. **Créez et activez un environnement virtuel** (recommandé) :

```
# Créer l'environnement
python -m venv venv

# Activer sur Windows (PowerShell)
.\venv\Scripts\Activate.ps1

# Activer sur macOS/Linux
source venv/bin/activate
```

3. **Installez les dépendances Python :**

```
pip install -r requirements.txt
```

4. **Lancez le serveur Flask :**

```
cd backend
python api.py
```

Le serveur est maintenant en cours d'exécution sur `http://127.0.0.1:5000`.

2. Démarrage du Frontend (Interface React)

Le frontend est une application React qui s'exécute généralement sur le port `5173`.

1. **Ouvrez un second terminal.**
2. **Naviguez dans le dossier `frontend` :**

```
cd frontend
```

3. **Installez les dépendances Node.js :**

```
npm install
```

4. **Lancez le serveur de développement React :**

```
npm run dev
```

L'application est maintenant accessible dans votre navigateur à l'adresse `http://localhost:5173` (ou un autre port si celui-ci est déjà utilisé).

🎨 Interface Utilisateur Améliorée

Le frontend a été entièrement redessiné avec une interface moderne et intuitive :

Caractéristiques principales :

- **Écran de bienvenue** : Accueil avec actions rapides pour les questions courantes
- **Bulles de messages** : Distinction claire entre les messages utilisateur (bleu) et bot (beige) avec timestamps
- **Indicateur de saisie** : Animation avec des points animés quand le bot traite la requête
- **Actions rapides** : Boutons pour les questions prédéfinies, disponibles toujours
- **Conception réactive** : Fonctionnalité optimale sur desktop, tablette et mobile
- **Scrollbar personnalisée** : Thématisée aux couleurs FSO (#003366)
- **Persistance du chat** : Les messages sont sauvegardés dans le localStorage
- **Gestion d'erreurs améliorée** : Messages d'erreur clairs avec options de réessai
- **États de chargement** : Feedback visuel pendant les appels API

Composants réutilisables :

- `ChatMessage.jsx` : Affichage des messages avec timestamps
- `TypingIndicator.jsx` : Animation de saisie avec points animés
- `QuickActionButton.jsx` : Boutons d'actions rapides

🛠️ Structure du Projet

```
.
├── api.py              # Backend : Serveur API Flask
├── app.py              # Ancienne interface Streamlit (archivée)
├── main.py             # Backend : Cœur du pipeline RAG et logique du chatbot
├── requirements.txt    # Dépendances Python
├── frontend/           # Dossier contenant l'application React
│   ├── src/
│   │   ├── App.jsx     # Composant principal de l'interface de chat
│   │   └── App.css     # Styles de l'interface
│   └── package.json    # Dépendances Node.js
├── data/
│   └── embeddings.json # Base de données vectorielle (générée)
└── pdfs/               # Dossier contenant les documents sources
```