**Projeto 01 - Novo Estudo de caso: Leitor de PDF para armazenar dados em banco vetorial e LLMs para conversar com os documentos.**

Aqui está um exemplo completo de uma aplicação RAG (Retrieval-Augmented Generation) usando Python, banco de dados vetorial (FAISS), leitura de documentos PDF com PyMuPDF e interface web com Streamlit. A aplicação realiza:

* Leitura de arquivos PDF.
* Segmentação e vetorização dos textos com OpenAIEmbeddings.
* Armazenamento e busca de vetores usando FAISS.
* Integração com OpenAI para gerar respostas com base nos trechos recuperados.
* Um chatbot via Streamlit para interação com o usuário.

**📦 Requisitos**

pip install streamlit langchain openai faiss-cpu pymupdf

**📁 Estrutura de pastas**

rag\_chatbot/

├── app.py

├── .env

└── docs/

└── exemplo.pdf

**📄 .env**

OPENAI\_API\_KEY=your\_openai\_key

**✅ Melhores práticas implementadas:**

* Uso de **FAISS** para armazenamento vetorial local e rápido.
* Separação de texto em chunks com sobreposição para **melhor contexto**.
* Modelo da OpenAI com controle de temperatura para respostas mais exatas.
* Upload e leitura de arquivos diretamente no navegador com Streamlit.
* .env para segurança da chave da API.

**🚀 Para rodar:**

streamlit run app.py

Se quiser, posso ajudar com:

* Armazenar múltiplos PDFs e construir um índice geral.
* Persistência do FAISS para uso offline.
* Uso do LangChain com PGVector.
* Estilo visual melhorado com componentes interativos no Streamlit.