Visão do Produto

Lexi é um chatbot baseado em inteligência artificial especialista no Direito Brasileiro, feito para auxiliar nos estudos para concursos públicos e consultas jurídicas. O objetivo é permitir que estudantes e juristas façam perguntas em linguagem natural e recebam respostas precisas, baseadas em um conjunto de arquivos, composto por documentos legais, jurídicos e materiais de estudo. Permitirá também que os usuários incorporem os seus próprios arquivos para serem revisados antes das respostas pelo modelo, permitindo informações mais específicas ao usuário em questão. Por exemplo, um advogado poderá anexar ao modelo documentos referentes ao caso em que está trabalhando, permitindo que o modelo responda sobre aquele caso usando os documentos jurídicos oficiais que previamente estão incorporados ao modelo.

OBS: os arquivos anexados pelos usuários não serão compartilhados com outros e também não serão incorporados ao modelo geral

Estrutura inicial do repositório

```
project/
├── app/
│   ├── main.py              # API FastAPI com rotas para upload e perguntas
│   ├── qa_pipeline.py       # Pipeline de perguntas com LangChain + OpenRouter
│   ├── document_loader.py   # Carregamento e indexação dos PDFs
│   ├── config.py            # Configurações (chaves, diretórios, etc.)
│   └── vector_store/        # Banco vetorial local (ChromaDB)
├── data/
│   └── pdfs/                # PDFs enviados pelo usuário
├── requirements.txt         # Bibliotecas necessárias
└── README.md                # Documentação do projeto
```

Tecnologias Utilizadas

- LangChain: Framework para construir pipelines com LLMs.
- ChromaDB: Banco vetorial local para armazenar embeddings dos documentos.
- OpenRouter: Intermediário de modelos LLM, usado para acessar o modelo Lhama 3.
- FastAPI: Framework web para criar a API.
- PyMuPDF: Biblioteca para leitura de arquivos PDF.
- Uvicorn: Servidor ASGI para rodar a API.
- Stripe: serviço para gerenciar pagamentos.

Funcionamento Esperado

1. Envio de Documentos:
    - O usuário envia arquivos para a API via endpoint /upload.
    - Esses arquivos processados, extraídos e convertidos em embeddings vetoriais com LangChain + OpenAIEmbeddings.
    - Os vetores serão armazenados localmente no ChromaDB, impedindo o armazenamento em servidores dos arquivos dos usuários, mantendo dados pessoais seguros.
2. Realização de Perguntas:
    - O usuário envia uma pergunta via endpoint.
    - O sistema busca nos documentos os trechos mais relevantes usando recuperação semântica (similaridade vetorial).
    - A pergunta e os documentos relevantes serão enviados ao modelo Llama 4 Maverick via OpenRouter.
    - O modelo responde com base no conteúdo recuperado, retornando uma resposta precisa.

Casos de Uso

- Estudantes e profissionais do Direito que desejam revisar e aprender leis e jurisprudência.
- Candidatos a concursos públicos que queiram reforçar o estudo por temas.
- Leitura assistida de PDFs com possibilidade de perguntas sobre o conteúdo.

Próximos Passos

- Criar interface web amigável (frontend com cores da identidade visual, rosa ou cinza)
- Adicionar autenticação de usuários.
- Suporte para múltiplos arquivos e histórico de perguntas.
- Otimização de desempenho com banco vetorial hospedado ou em memória. - Deploy em servidor (VPS ou serviço em nuvem como Render, Railway etc.

🔴 Backlog

⚙️ Fase 1: Estrutura e fundação do projeto (essencial para começar)

• Estrutura inicial do projeto criada
• Estrutura básica do projeto com FastAPI, LangChain e Chroma
• Organização dos documentos em /data/pdfs
• Configuração de variáveis com .env
• Conectar modelo LLM (Llama 4 Maverick) via OpenRouter
• Integração com embeddings OpenAI e banco vetorial Chroma
• Leitura e indexação automática de PDFs com LangChain
• Rotas de upload e pergunta implementadas

🧱 Fase 2: Testes e validação

• Teste de funcionalidade com API funcionando localmente
• Documento explicativo e versão em PDF criada

💡 Fase 3: Identidade visual e interface

• Identidade visual inicial com logo e interface mockup
• Criar frontend web responsivo, com identidade visual rosa ou cinza

---

🔐 Fase 4: Funcionalidades complementares

• Implementar autenticação de usuários
• Adicionar histórico de perguntas/respostas por usuário
• Adicionar suporte para múltiplos arquivos por sessão
• Adicionar feedback de avaliação nas respostas
• Implementar sistema de logs e métricas de uso
• Criar painel de administração (opcional)

🚀 Fase 5: Finalização e publicação

• Criar landing page com explicação do projeto
• Deploy do sistema (Render, Railway, VPS etc.)

Repositório: https://github.com/DarioAlef/Lexi

Autor: Dário Alef Barros Lima
dario.alef@gmail.com