

Voce é um especialista python e langchain implemente nesse projeto a descrição abaixo:

crie um checklist do fluxo faça passo a passo  
use FASTAPI

use como referencia essa pasta C:\projetos\mba-ia-niv-introducao-langchain

Ingestão e Busca Semântica com LangChain e Postgres

Objetivo

Você deve entregar um software capaz de:

Ingestão: Ler um arquivo PDF e salvar suas informações em um banco de dados PostgreSQL com extensão pgVector.

Busca: Permitir que o usuário faça perguntas via linha de comando (CLI) e receba respostas baseadas apenas no conteúdo do PDF.

Exemplo no CLI

Faça sua pergunta:

PERGUNTA: Qual o faturamento da Empresa SuperTechIABrazil?

RESPOSTA: O faturamento foi de 10 milhões de reais.

---

Perguntas fora do contexto:

PERGUNTA: Quantos clientes temos em 2024?

RESPOSTA: Não tenho informações necessárias para responder sua pergunta.

Tecnologias obrigatórias

Linguagem: Python

Framework: LangChain

Banco de dados: PostgreSQL + pgVector

Execução do banco de dados: Docker & Docker Compose (docker-compose fornecido no repositório de exemplo)

Pacotes recomendados

Split: from langchain\_text\_splitters import RecursiveCharacterTextSplitter

Embeddings (OpenAI): from langchain\_openai import OpenAIEmbeddings

Embeddings (Gemini): from langchain\_google\_genai import GoogleGenerativeAIEmbeddings

PDF: from langchain\_community.document\_loaders import PyPDFLoader

Ingestão: from langchain\_postgres import PGVector

Busca: similarity\_search\_with\_score(query, k=10)

OpenAI

Crie uma API Key da OpenAI.

Modelo de embeddings: text-embedding-3-small

Modelo de LLM para responder: gpt-5-nano

Gemini

Crie uma API Key da Google.

Modelo de embeddings: models/embedding-001

---

## Requisitos

### 1. Ingestão do PDF

O PDF deve ser dividido em chunks de 1000 caracteres com overlap de 150.

Cada chunk deve ser convertido em embedding.

Os vetores devem ser armazenados no banco de dados PostgreSQL com pgVector.

### 2. Consulta via CLI

Criar um script Python para simular um chat no terminal.

Passos ao receber uma pergunta:

Vetorizar a pergunta.

Buscar os 10 resultados mais relevantes ( $k=10$ ) no banco vetorial.

Montar o prompt e chamar a LLM.

Retornar a resposta ao usuário.

Prompt a ser utilizado:

CONTEXTO:

{resultados concatenados do banco de dados}

## REGRAS:

- Responda somente com base no CONTEXTO.
- Se a informação não estiver explicitamente no CONTEXTO, responda:  
"Não tenho informações necessárias para responder sua pergunta."
- Nunca invente ou use conhecimento externo.
- Nunca produza opiniões ou interpretações além do que está escrito.

## EXEMPLOS DE PERGUNTAS FORA DO CONTEXTO:

Pergunta: "Qual é a capital da França?"

Resposta: "Não tenho informações necessárias para responder sua pergunta."

Pergunta: "Quantos clientes temos em 2024?"

Resposta: "Não tenho informações necessárias para responder sua pergunta."

Pergunta: "Você acha isso bom ou ruim?"

Resposta: "Não tenho informações necessárias para responder sua pergunta."

## PERGUNTA DO USUÁRIO:

{pergunta do usuário}

## RESPONDA A "PERGUNTA DO USUÁRIO"

Estrutura obrigatória do projeto

Faça um fork do repositório para utilizar a estrutura abaixo: Clique aqui

```
|── docker-compose.yml  
|── requirements.txt # Dependências  
|── .env.example # Template da variável OPENAI_API_KEY  
|── src/  
|   ├── ingest.py # Script de ingestão do PDF  
|   └── search.py # Script de busca
```

```
|   └── chat.py # CLI para interação com usuário  
|   └── document.pdf # PDF para ingestão  
└── README.md # Instruções de execução
```

Repositórios úteis:

Curso de nivelamento com LangChain

Template básico com estrutura do projeto

VirtualEnv para Python

Crie e ative um ambiente virtual antes de instalar dependências:

```
python3 -m venv venv
```

```
source venv/bin/activate
```

Ordem de execução

Subir o banco de dados:

```
docker compose up -d
```

Executar ingestão do PDF:

```
python src/ingest.py
```

Rodar o chat:

```
python src/chat.py
```