

PASSO A PASSO DA ATIVAÇÃO DO AMBIENTE

1)cd C:\Users\Mario\Desktop\projeto\_app

2)novo\_ambiente\Scripts\activate ou (conda activate empenhos2\_env)

3)pip install -r requirements.txt

4)pip install psycopg2-binary

5)streamlit run app10.py (streamlit run app15.py)

**Contexto Geral**

O projeto integra **4 componentes principais**:

1.**Banco de Dados PostgreSQL** (Dockerizado)

2.**Interface de Admin (pgAdmin)**

3.**Aplicação Streamlit** (Frontend)

4.**LLM Groq + LangChain** (Geração de SQL)

**1. Componentes e Papéis**

**a) Docker + PostgreSQL**

* **Objetivo**: Isolar o banco de dados em um container para portabilidade e consistência.
* **Detalhes**:
  + Imagem: postgres:17 (versão 17 do PostgreSQL).
  + Volume: postgres\_data para persistência dos dados.
  + Configuração: docker-compose.yml define rede, portas (5432) e variáveis de ambiente (usuário/senha).
  + **Método de autenticação**: Inicialmente configurado como trust (sem senha) via pg\_hba.conf.

**b) pgAdmin**

* **Objetivo**: Gerenciar o PostgreSQL via interface gráfica.
* **Detalhes**:
  + Conecta-se ao container Docker (host: localhost, porta: 5432).
  + Usado para:
    - Visualizar tabelas (ex: despesas).
    - Executar backups/restores (ex: pg\_restore).
    - Validar schemas e dados.

**c) Streamlit App**

* **Objetivo**: Interface web para interação com o banco via linguagem natural.
* **Detalhes**:
  + **Frontend**:
    - Campo de input para perguntas (ex: "Traga gastos de dezembro de 2024").
    - Exibição de resultados em tabela formatada.
  + **Backend**:
    - Conexão com PostgreSQL via psycopg2 + SQLAlchemy.
    - Formatação de valores monetários para o padrão brasileiro (ex: 1.234,56).

**d) LangChain + Groq**

* **Objetivo**: Gerar queries SQL a partir de perguntas em linguagem natural.
* **Detalhes**:
  + **LangChain**:
    - Orquestra a pipeline de prompts e LLMs.
    - Componentes usados:
      * SQLDatabase: Conecta-se ao PostgreSQL para extrair schema.
      * RunnablePassthrough: Encadeia etapas (extrair schema → gerar prompt → enviar para LLM).
      * ChatPromptTemplate: Define o template de pergunta (schema + query).
  + **Groq**:
    - Fornece o LLM llama-3.3-70b-versatile via API.
    - Chave de API: GROQ\_API\_KEY (armazenada em .env).

**2. Fluxo de Dados**

1.**Usuário** faz uma pergunta no Streamlit (ex: "Gastos de dezembro de 2024").

2.**Streamlit** invoca o sql\_chain (LangChain):

* + **Passo 1**: Extrai o schema da tabela despesas via get\_schema().
  + **Passo 2**: Gera um prompt combinando schema e pergunta.
  + **Passo 3**: Envia o prompt para o LLM (Groq) → Retorna uma query SQL.

3.**Query SQL** é executada no PostgreSQL via run\_query().

4.**Resultados** são formatados (valores monetários) e exibidos no Streamlit.

**3. Tecnologias e Dependências**

| Componente | Tecnologias/Pacotes |
| --- | --- |
| **Containerização** | Docker, docker-compose |
| **Banco de Dados** | PostgreSQL, pgAdmin |
| **Backend** | Python, Streamlit, pandas, psycopg2, SQLAlchemy |
| **LLM/Geração SQL** | LangChain, Groq API, LLaMA 3.3 70B |
| **Ambiente** | Conda (ambiente empenhos\_env), .env |