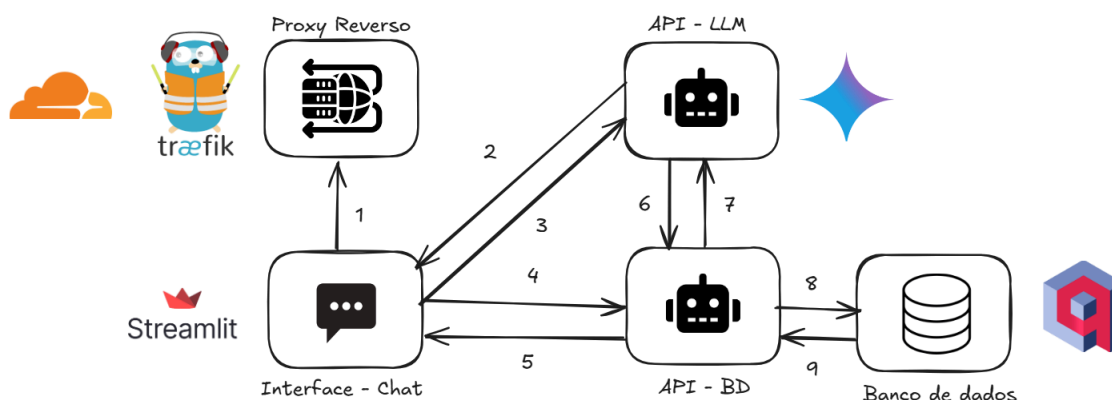


Atividade Prática – DevOps

Aluno: Daniel Hiroyuki Watanabe

RA: 800697

Aplicação escolhida: RAG (Retrieval Augmented Generation)



- Arquivo PDF com uma descrição da aplicação escolhida: visão geral da aplicação, contêineres e/ou tecnologias utilizadas, manual de instalação, etc [Item obrigatório]

Tradução dos números da imagem:

1. A interface do chat vai estar disponível para o usuário através do proxy reverso utilizando o traefik e cloudflare para lidar com isso
2. A LLM retorna para o chat a resposta criada
3. Faz a query para a LLM
4. Quando faz upload de arquivos, ou queries diretas
5. Retorna do banco se conseguiu uploadar com sucesso
6. A LLM retorna os embeddings dos textos
7. BD requisita embedding da LLM
8. Faz a busca vetorial
9. Retorna os 5 textos mais semelhantes semanticamente

Basicamente como funciona essa aplicação, estamos utilizando algumas ferramentas:

- Gemini (LLM)
- Qdrant (Banco de dados de vetor)
- Streamlit (Front em python)

- Traefik (Servirá para proxy reverso)
- CloudFlare (Será utilizado para redirecionar a algum DNS, cuidará de alguns certificados)

Dentro da nossa interface em Streamlit, você pode:

1. Criar uma coleção no Qdrant

- Basta clicar em “Nova Collection” e informar um nome. A collection é onde os dados serão armazenados.

2. Enviar arquivos PDF para a coleção

- Faça o upload dos PDFs diretamente na aba correspondente.
- Cada PDF é dividido em “chunks” (pedaços de texto) para otimizar a busca.

3. Gerar embeddings com a LLM (Gemini)

- Ao fazer o upload, usamos a API da LLM para transformar cada chunk em um vetor num espaço semântico.
- Esses vetores são armazenados no Qdrant e indexados com distância de cosseno, permitindo buscas rápidas e precisas.

4. Fazer consultas em linguagem natural

- Na interface, digite perguntas sobre o conteúdo que você carregou.
- O sistema gera um embedding da sua pergunta, busca os vetores mais relevantes no Qdrant e retorna as respostas com base nos trechos correspondentes.

5. Configuração necessária

- Crie uma conta na plataforma da Gemini e obtenha sua API_KEY.
- Coloque essa chave no arquivo .env do projeto, na variável GEMINI_API_KEY.

6. Infraestrutura de rede

- O Traefik atua como proxy reverso, roteando as requisições para o serviço correto.
- O Cloudflare faz a proteção DDoS e o balanceamento de tráfego na borda.

Fluxo resumido:

1. Cria collection → 2. Faz upload de PDFs → 3. Gera embeddings (Gemini) → 4. Armazena e indexa no Qdrant → 5. Recebe consultas → 6. Gera embedding da query → 7. Busca vetorial → 8. Retorna respostas.

Temos 4 containeres principais para esse trabalho, (5 containers se for contar o do proxy reverso, mas não é o foco pois não tem dockerfile, entretanto é necessário arrumar para conseguir utilizar o programa):

qdrant:

É o próprio banco de dados, persiste os documentos

api-chat-bd:

```
@app.post("/create-collection")
```

```
@app.post("/upload-pdf")
```

```
@app.post("/search", response_model=List[SearchResult])
```

```
@app.get("/health")
```

chat-api-llm

```
@app.post("/ask", response_model=QuestionResponse)
```

```
@app.post("/generate-embedding", response_model=EmbeddingResponse)
```

```
@app.get("/health")
```

chat-interface

Utiliza a query do usuário como parâmetro das APIs expostas acima, para buscar resposta.

```
def create_collection():
```

```
def upload_pdf(file):
```

```
def search_documents(query, limit=5):
```

```
def chat_with_gemini(message, history):
```

Lidando com o .env dos arquivos

Para criar uma senha para o traefik:

htpasswd -nb admin <minhasenha>




output => your_password

Explicando o .env principal:

- GEMINI_API_KEY=your_gemini_key
- CF_DNS_API_TOKEN=your_cloudflare_token
- TRAEFIK_ACME_EMAIL=your_email_acme
- TRAEFIK_DASHBOARD_HOST=traefik2.yourdns.com
- TRAEFIK_AUTH=admin:your_password
- CHAT_DOMAIN=chatdevops.yourdns.com

Nas pastas api_chat_bd

Dentro da cloudflare é importante você redirecionar cada um dos seus tipos para os respectivos IPs:

<input type="checkbox"/>	Tipo ①	Nome ①	Conteúdo ① ▲	Status do proxy ①	TTL ①	Ações
<input type="checkbox"/>	A	chatdevops	137.131.190.118	 Com proxy	Auto	Editar ►
<input type="checkbox"/>	A	portainer2	137.131.190.118	 Com proxy	Auto	Editar ►
<input type="checkbox"/>	A	traefik2	137.131.190.118	 Com proxy	Auto	Editar ►

Na condição atual, ele pode estar com o problema no front quando uploadada, mas no backend ele funciona bem, basta acompanhar os logs do docker compose.