Passo a passo para a execução do MVP

Passo 1: Configuração do Ambiente

1. **Instale as dependências necessárias:**
    - Linguagem de programação Python (versão >= 3.8).
    - Biblioteca para interação com o Telegram: python-telegram-bot.
    - Integração com Pinecone: pinecone-client.
    - Modelos de linguagem (como OpenAI): openai ou similar.

- Outras dependências:
  pip install langchain
  pip install pydantic
  pip install requests

2. **Crie contas nas plataformas necessárias:**
    - **Telegram:** Configure um bot através do BotFather.
    - **Pinecone:** Crie um índice em Pinecone para armazenar vetores.
    - **OpenAI:** Gere uma chave de API para acessar os embeddings do GPT.
3. **Configure variáveis de ambiente para segurança:**

Defina as variáveis no arquivo .env:

TELEGRAM_BOT_TOKEN=<seu_token_telegram>
PINECONE_API_KEY=<sua_chave_api_pinecone>
OPENAI_API_KEY=<sua_chave_api_openai>

Passo 2: Configuração do Banco de Dados Vetorial

1. **Configuração do Pinecone:**
    - Crie um índice no painel do Pinecone.
    - Nomeie o índice, defina a dimensão para corresponder aos embeddings gerados (ex.: 1536 para GPT-3.5).
    - Configure o tipo de índice para cosine ou dot product.

Inicialize o índice no código:

```
import pinecone

pinecone.init(api_key="YOUR_API_KEY", environment="us-west1-gcp")
index = pinecone.Index("nome_do_indice")
```

Passo 3: Desenvolvimento do Fluxo "Load Data into Database"

1. **Receber Arquivo pelo Telegram:**
    - Use o webhook do Telegram para receber mensagens e arquivos enviados pelo usuário.
2. **Processar o Arquivo PDF:**
    - Extraia o texto do PDF usando bibliotecas como PyPDF2 ou pdfplumber.

```
import pdfplumber

with pdfplumber.open("arquivo.pdf") as pdf:

    text = ""

    for page in pdf.pages:

        text += page.extract_text()
```

3. **Dividir Texto em Chunks:**
    - Utilize uma ferramenta de splitting, como o CharacterTextSplitter da biblioteca LangChain.

```
from langchain.text_splitter import CharacterTextSplitter

splitter = CharacterTextSplitter(chunk_size=500, chunk_overlap=50)
chunks = splitter.split_text(text)
```

4. **Gerar Embeddings:**
    - Use a API da OpenAI para gerar embeddings de cada chunk.

```
import openai

embeddings = [

    openai.Embedding.create(input=chunk,
engine="text-embedding-ada-002")["data"][0]["embedding"]

    for chunk in chunks

]
```

5. **Salvar no Pinecone:**
    - Envie os embeddings e IDs únicos para o índice.

```
for i, embedding in enumerate(embeddings):

    index.upsert(vectors=[(f"doc_{i}", embedding)])
```

Passo 4: Desenvolvimento do Fluxo "Chat with Database"

1. **Receber Perguntas pelo Telegram:**
    - Configure o bot para escutar mensagens de texto enviadas pelos usuários.
2. **Recuperar Dados do Pinecone:**
    - Use o índice do Pinecone para buscar os chunks mais relevantes.

```
query_embedding = openai.Embedding.create(
    input="pergunta_do_usuario", engine="text-embedding-ada-002"
)["data"][0]["embedding"]

result = index.query(query_embedding, top_k=3, include_metadata=True)
```

3. **Formular Resposta:**
    - Passe os textos recuperados para o modelo (GPT) gerar uma resposta final.

```
context = " ".join([match["metadata"]["text"] for match in
result["matches"]])
response = openai.Completion.create(
    engine="text-davinci-003", prompt=f"Com base no contexto:
{context}, responda: {user_query}", max_tokens=200
)
```

4. **Enviar Resposta pelo Telegram:**
    - Use o método sendMessage da API Telegram para enviar a resposta ao usuário.

Passo 5: Testes e Validação

1. Teste o fluxo "Load Data into Database":
    - Envie um arquivo PDF pelo Telegram.
    - Verifique no painel do Pinecone se os vetores foram salvos corretamente.
2. Teste o fluxo "Chat with Database":
    - Envie perguntas relacionadas ao conteúdo do PDF.
    - Confirme se o bot responde com informações precisas e relevantes.

Passo 6: Documentação e Implantação

1. Documente todo o processo no README do repositório.
2. Implemente logs para monitorar erros e desempenho.
3. Faça o deploy em um servidor (como AWS ou Heroku) para manter o bot funcionando 24/7.