Aplicativo Flask com IA

2º AD e 3° Tech

Instituto J&F
Centro de Educação para os Negócios

Atividade Prática

LLM + juízes de IA + Flask

- Integrar recursos de RAG e juiz de IA ao aplicativo de chat atendente/usuário em tempo real, implementado em Flask
- Mudar a aplicação de consulta simples sem memória para consulta tipo chat com memória
- Assunto a ser usado para os arquivos de referência para o RAG: escolhido pelos grupos de trabalho

Sequência de funcionamento

1. O usuário do chat faz uma pergunta – deverá terminar em ponto de interrogação (como já está na aplicação)
2. O cliente **usuário** utiliza recursos de RAG e consulta ao Gemini (usando recursos de memória da conversa) para gerar uma resposta. **O recurso de RAG deverá consultar os arquivos com informações levantadas pelo grupo.**

Sequência de funcionamento

3. A resposta obtida do agente de consulta com RAG é validada por um juiz de IA
4. A resposta é registrada no log do chat atendente/usuário (como já está implementado na aplicação)

Aplicação Chat em Flask

- Aplicação de chat com comunicação assíncrona entre processos e consumo da API do Gemini

https://github.com/astromyrna/TECH_flask_chat/tree/gemini

Juízes de IA e RAG

- Códigos funcionais - consumo da API Gemini com RAG e juiz de IA
- Podem ser usados como alternativa ao código fornecido na apostila “Alunos_Construindo um Sistema Inteligente.pdf”

https://github.com/astromyrna/modelos-juiz-rag

Modificações no código-fonte

1. Insira o código no arquivo `modelo.py` (é a biblioteca de rotinas de IA deste app)
2. Essas rotinas serão chamadas em `routes.py` (acrescente os nomes das novas rotinas no `import`)

routes.py

app > routes.py

```
1  from flask import Blueprint, render_template, request, session
2  from datetime import datetime
3  from app import socketio
4  import os
5  from app.gemini.modelo import responder_pergunta
```

Modificações no código-fonte

2. Modifique o trecho realçado no arquivo `routes.py`

- Passará a usar RAG e juiz

gemini TECH_flask_chat / flask_chat / app / routes.py ↑ Top

Code Blame 80 lines (69 loc) · 3.29 KB Raw

```
41  @bp.route("/usuario", methods=["GET", "POST"])
42  def usuario():
43      if "chat_id" not in session:
44          session["chat_id"] = datetime.now().strftime("%Y%m%d-%H%M%S")
45          registrar_log("SISTEMA", f"=== Início da Sessão {session['chat_id']} ===", session["chat_id"])
46
47      if request.method == "POST":
48          if "enviar" in request.form:
49              msg = request.form["mensagem"]
50              registrar_log("USUÁRIO", msg, session["chat_id"])
51
52              # Se for uma pergunta, consulta o Gemini
53              if msg.strip().endswith("?"):
54                  resposta = responder_pergunta(msg)
55
56                  # seu código de juízes de IA aqui
57
58                  # após verificar se ocorreu alucinação, gravar a resposta no log da sessão
59                  registrar_log("GEMINI", resposta, session["chat_id"])
60          elif "encerrar" in request.form:
61              registrar_log("SISTEMA", f"=== Fim da Sessão {session['chat_id']} ===", session["chat_id"])
62              session.pop("chat_id", None)
```

Entregas por grupo

- Link do GitHub (público) para a aplicação funcional
- Demonstração da aplicação em classe
- Opcionais:
  - humor ☺
  - resposta por voz
  - etc

Instituto J&F

Centro de Educação para os Negócios