Interface gráfica do usuário

Descrição gerada automaticamente com confiança baixa

GRADUAÇÃO EM BANCO DE DADOS

DESENVOLVIMENTO DE DATA-DRIVEN APPS COM PYTHON [24E4\_3]

WANDERSON RAFAEL MENDONÇA BATISTA

TESTE DE PERFORMANCE – TP3

PROF. FERNANDO GUIMARÃES FERREIRA

RIO DE JANEIRO, 2024

Os códigos estão disponíveis em <https://github.com/wanderaf/wanderaf-datadrive_tp02>

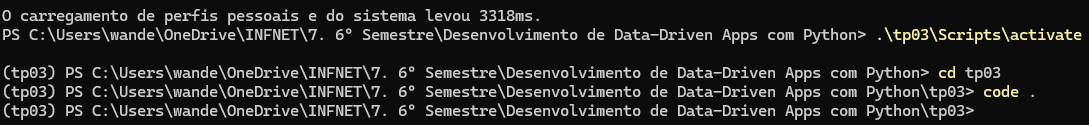
1. **Defina o Problema**: Escolha um problema que possa ser resolvido com o uso de agentes inteligentes. Descreva o objetivo da aplicação, o público-alvo e as principais funcionalidades planejadas. Detalhe os casos de uso que serão testados para validar a solução.

**Problema**: Buscar e resumir informações sobre proposições legislativas no Brasil usando dados abertos da Câmara dos Deputados.

* **Objetivo**: Criar um agente inteligente que consulta, analisa e resume informações sobre projetos de lei com base em palavras-chave, intervalos de datas e textos fornecidos pelo usuário.
* **Público-alvo**: Pesquisadores, estudantes e cidadãos interessados em legislação brasileira.
* **Funcionalidades principais**:
  1. Consultar projetos de lei na API da Câmara dos Deputados.
  2. Gerar resumos e palavras-chave a partir de textos fornecidos.
  3. Memória conversacional para melhorar a interatividade.
* **Casos de uso**:
  + Buscar projetos de lei com palavras-chave específicas.
  + Resumir textos legislativos longos.
  + Usar histórico de interações para refinar pesquisas.

1. **Configure o Framework LangChain:** Configure o ambiente de desenvolvimento para utilizar LangChain, escolhendo um LLM apropriado (como Gemini ou GPT-4) para sua aplicação.

Implementação do LLM com GPT



Texto

Descrição gerada automaticamente

1. **Implemente Ferramentas:** Crie e configure no mínimo duas ferramentas externas (tools). Cada ferramenta deve ter nome, descrição e função claramente definidos, e ser integrada ao LangChain.

Criada uma ferramenta para consulta de projetos de lei e outra para resumo e extração de palavras chaves

Texto

Descrição gerada automaticamente

1. **Utilize o Framework ReAct**: Implemente o raciocínio interativo usando o framework ReAct, garantindo que o agente possa alternar entre raciocinar, tomar ações e observar os resultados para alcançar os objetivos.

Texto

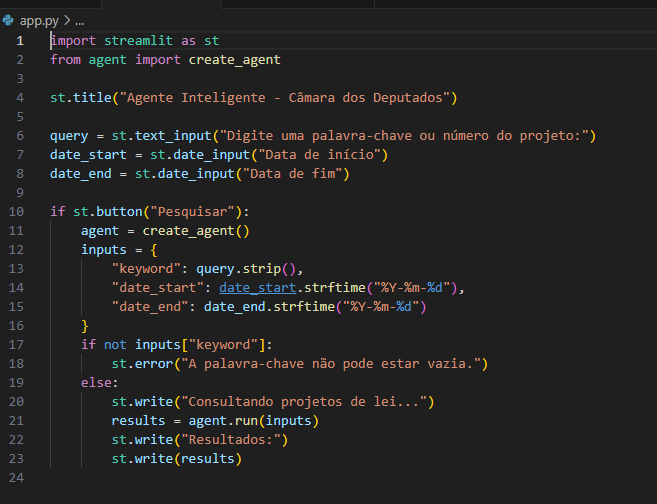
Descrição gerada automaticamente

1. **Adicione Memória Conversacional**: Configure um sistema de memória conversacional, permitindo que o agente lembre o histórico de interações e utilize esse contexto para gerar respostas mais coerentes.

A memória foi utilizada na linha 25 da imagem anterior:

Memory = ConversationBufferMemory

1. **Desenvolva a Lógica do Agente**: Estruture a lógica de funcionamento do agente, integrando ferramentas, memória e raciocínio interativo para resolver o problema proposto.
2. **Projete a Interface do Usuário**: Crie uma interface funcional (streamlit ou linha de comando) para que o agente possa ser utilizado. Não é necessário que seja sofisticada, mas deve ser clara e intuitiva.



1. **Teste a Solução**: Realize testes práticos utilizando os casos de uso definidos. Inclua uma explicação clara sobre como o agente facilita ou possibilita o trabalho e o que seria necessário fazer sem o uso desse recurso.

A solução retornou erro, houve várias tentativas, mas não funcionou junto através dos agentes.