Instituto Infnet

GRADUAÇÃO EM BANCO DE DADOS

Projeto de Bloco: Ciência de Dados Aplicada [24E3_5]

WANDERSON RAFAEL MENDONÇA BATISTA

TESTE DE PERFORMANCE – TP3

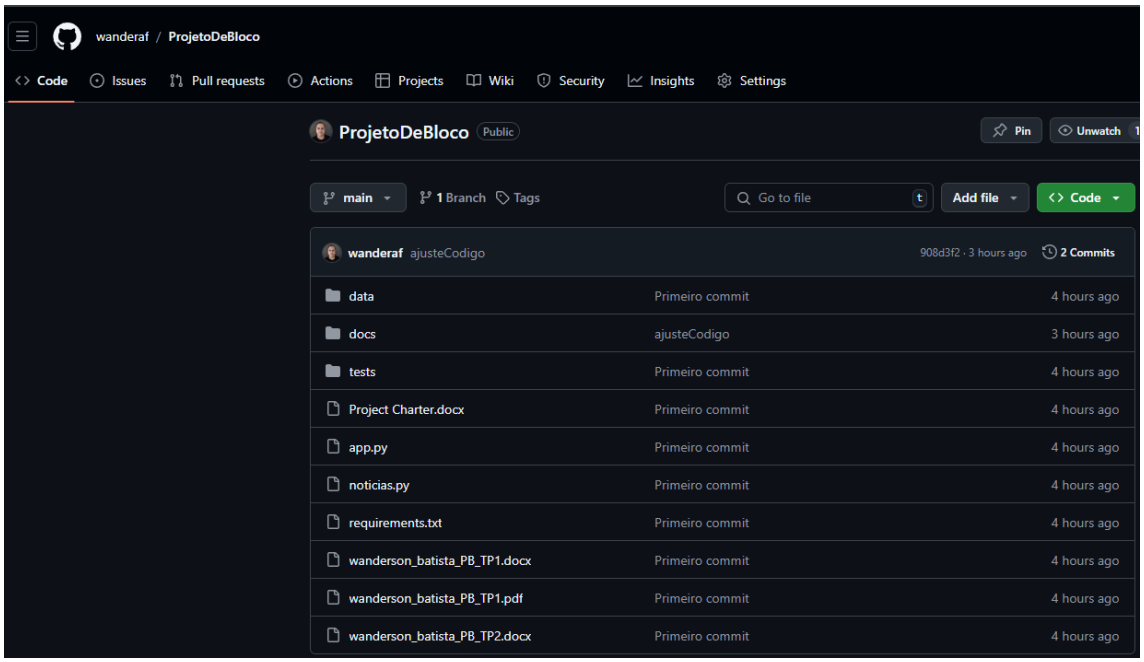
PROF. DIEGO DA SILVA RODRIGUES

RIO DE JANEIRO, 2024

Configuração do Ambiente de Desenvolvimento:

Configure seu ambiente de desenvolvimento, incluindo Git para controle de versão e preparação para deploy. Lembre-se de seguir a estrutura do CRISP-DM para organizar seu projeto de forma eficiente e escalável.

https://github.com/wanderaf/ProjetoDeBloco



1. **Revisão e Atualização da Documentação:**

Revise o Project Charter e o Data Summary Report, atualizando a documentação para refletir as novas funcionalidades e decisões tomadas nesta fase do projeto.
Reavalie o problema de negócio à luz das novas ferramentas (como FastAPI e Selenium) e ajuste suas metas, se necessário.
Atualize a descrição das fontes de dados utilizadas, considerando possíveis novas fontes obtidas com scraping dinâmico.

O Project Chart e Data Summary já estava previsto com os itens implementados nesta etapa

2. **Criação de uma Aplicação com Múltiplas Páginas:**

Evolua a interface da sua aplicação em Streamlit, implementando múltiplas páginas e um menu de navegação que permita ao usuário transitar facilmente entre diferentes seções da aplicação.
Cada página deve representar uma funcionalidade ou análise diferente, como a visualização de dados, gráficos interativos, upload/download de arquivos ou estatísticas geradas a partir dos dados coletados.

Na apresentação do etapa 2 o projeto já contemplava várias páginas

3. **Extração de Dados de Páginas Dinâmicas (Web Scraping):**

Utilize o Selenium para realizar o web scraping de páginas dinâmicas, se necessário.
Caso seu projeto utilize uma fonte de dados que exija interação com elementos dinâmicos (como formulários ou carregamentos assíncronos), o Selenium será essencial.
Observação: Se não houver necessidade de utilizar Selenium, concentre-se no aprimoramento dos dados coletados com Beautiful Soup ou APIs, mantendo a simplicidade quando possível.
Armazene os dados obtidos em arquivos CSV ou TXT, organizando-os no diretório de data/ para uso na aplicação.

Não foi necessário utilizar o Selenium, a utilização do Beutiful Soup já foi realizada ne etapa 2, salvando os arquivos em um text.

4. **Desenvolvimento de APIs com FastAPI:**

Configure o ambiente de desenvolvimento para a criação de APIs com FastAPI.
Crie uma API simples com rotas e endpoints que permitam interagir com os dados da aplicação. Exemplo de funcionalidades da API:
Consulta de dados (GET)
Envio de novos dados (POST)
Implemente ao menos duas rotas em sua API, e documente-as adequadamente, explicando sua função e como elas podem ser usadas para interação com os dados.

A api foi implementada no arquivo api.py, foi implementado o método Get para buscar nomes de unidades por palavras chaves e o método Post para adicionar novos dados no mongo.

Código utilizado:

app.py M api.PY U

api.PY > fetch_all_data

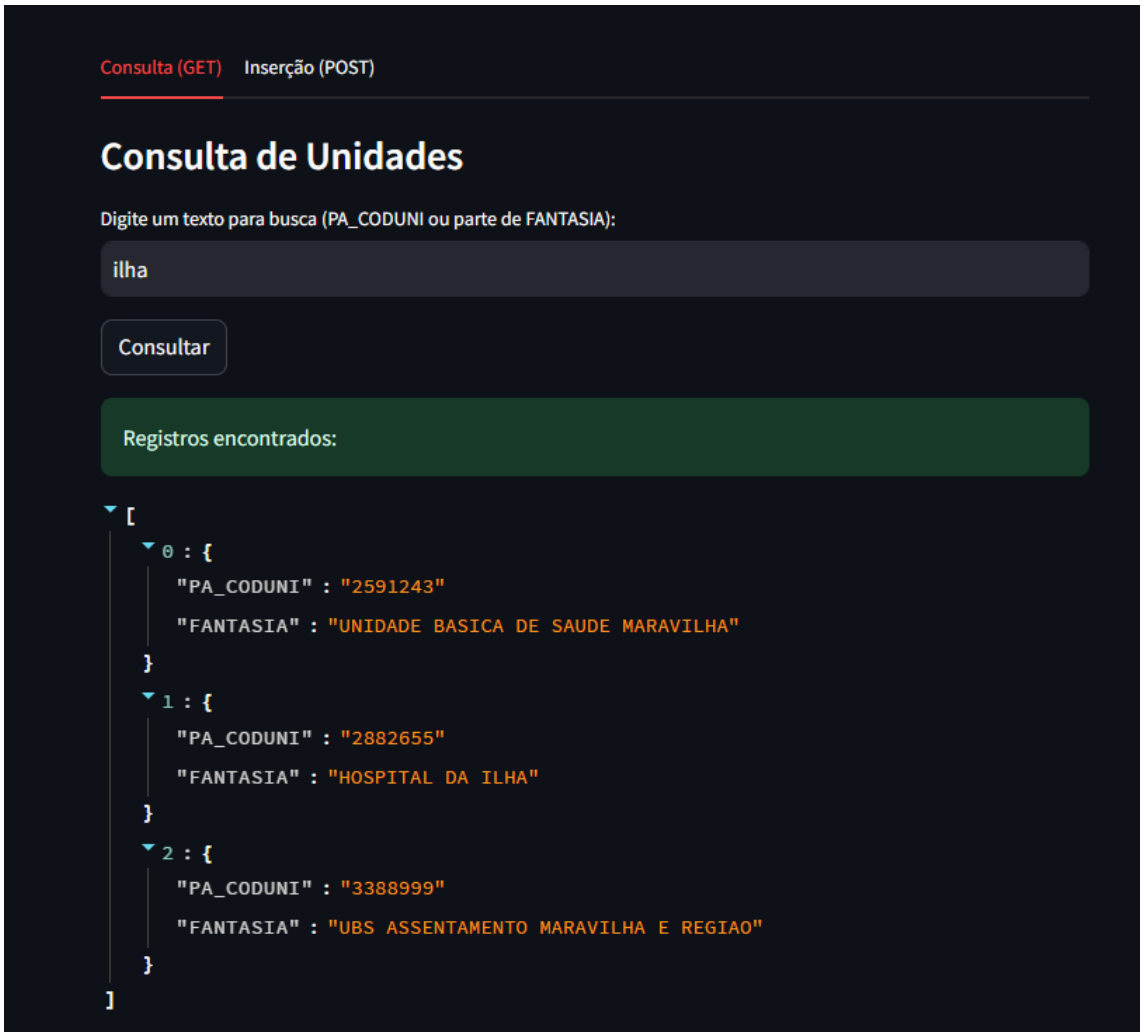
```python
from fastapi import FastAPI, HTTPException, Query
from pymongo import MongoClient
from typing import List
from pydantic import BaseModel

# Função para configurar a conexão com o MongoDB
def fetch_all_data():
    """
    Configura a conexão com o MongoDB e retorna a coleção SIA_ANALISE.
    """
    client = MongoClient("mongodb://localhost:27017/")
    db_analise = client["ANALISE"]
    return db_analise["SIA_ANALISE"]

# Obter a coleção do MongoDB
collection_sia_analise = fetch_all_data()

# Inicializar o FastAPI
app = FastAPI()

class UnidadeData(BaseModel):
    """
    Modelo para validar os dados de entrada no método POST.
    """
    PA_CODUNI: str
    FANTASIA: str
    TOTAL_PA_QTDPRO: int
    TOTAL_PA_QTDAPR: int
    TOTAL_PA_VALPRO: float
    TOTAL_PA_VALAPR: float
    PA_GESTAO: str
    PA_MVMR: str
    PA_CMPEF: str
    PA_PROC_ID: str
    PA_CBOCOD: str
    PA_NAT_JUR: str
    IP_DSCR: str

```

```python
@app.get("/unidades/", response_model=List[dict])
async def consultar_unidades(pa_coduni: str = Query(..., description="Texto para busca (PA_CODUNI ou parte de FANTASIA)")):
    """
    Consulta unidades de saúde na coleção SIA_ANALISE pelo PA_CODUNI ou parte do nome FANTASIA.
    - Se `pa_coduni` corresponder a parte do PA_CODUNI, retorna todos os registros correspondentes.
    - Se `pa_coduni` corresponder a parte do FANTASIA, retorna todos os registros correspondentes.
    """
    try:
        # Construir o filtro de busca dinâmico
        query_filter = {
            "$or": [
                {"PA_CODUNI": {"$regex": pa_coduni, "$options": "i"}},  # Busca parcial no PA_CODUNI
                {"FANTASIA": {"$regex": pa_coduni, "$options": "i"}}    # Busca parcial no FANTASIA
            ]
        }

        # Buscar dados no MongoDB
        resultados = collection_sia_analise.find(query_filter, {"_id": 0, "PA_CODUNI": 1, "FANTASIA": 1})

        # Garantir unicidade dos resultados
        dados_unicos = list({(res["PA_CODUNI"], res["FANTASIA"]) for res in resultados if "PA_CODUNI" in res and "FANTASIA" in res})

        # Verificar se há registros encontrados
        if not dados_unicos:
            return {"message": "Nenhum registro encontrado para a consulta."}

        # Formatar como lista de dicionários
        resposta_formatada = [{"PA_CODUNI": item[0], "FANTASIA": item[1]} for item in dados_unicos]

        return resposta_formatada

    except Exception as e:
        # Log detalhado do erro para depuração
        print(f"Erro ao consultar unidades: {e}")
        raise HTTPException(status_code=500, detail="Erro interno no servidor.")
```

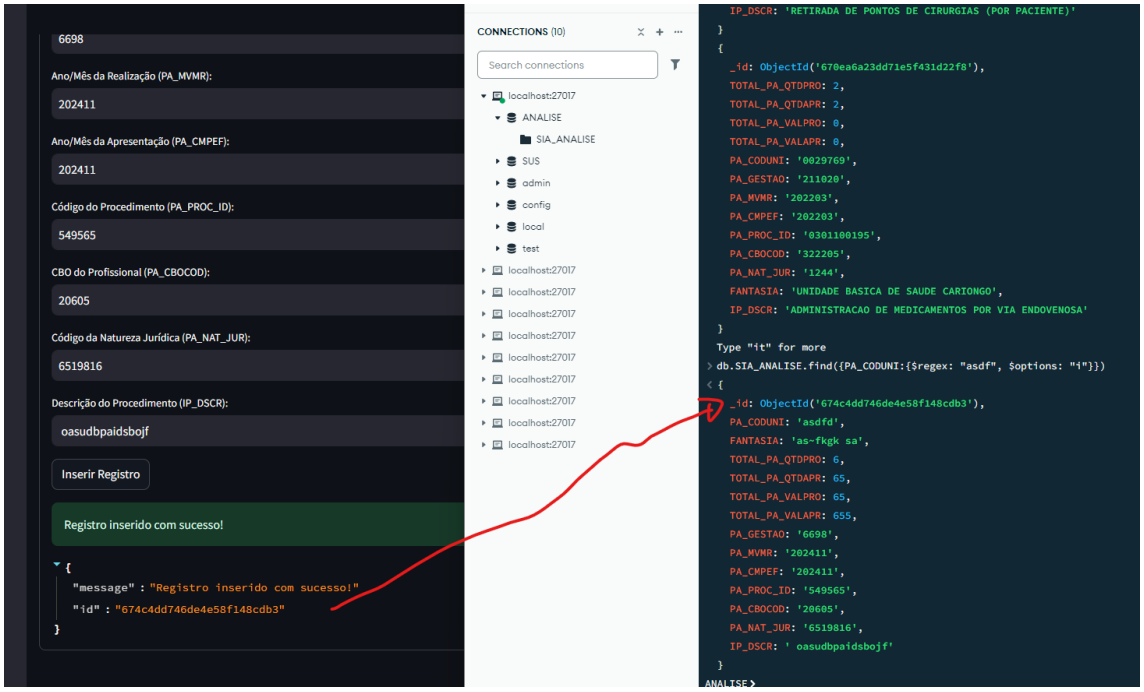
```python
@app.post("/unidades/")
async def adicionar_unidade(unidade: UnidadeData):
    """
    Endpoint para adicionar manualmente um registro na coleção SIA_ANALISE.
    """
    try:
        # Converter o objeto Pydantic para um dicionário
        unidade_dict = unidade.dict()

        # Inserir o registro no MongoDB
        resultado = collection_sia_analise.insert_one(unidade_dict)

        # Verificar se a inserção foi bem-sucedida
        if not resultado.inserted_id:
            raise HTTPException(status_code=500, detail="Erro ao inserir o registro no banco de dados.")

        return {"message": "Registro inserido com sucesso!", "id": str(resultado.inserted_id)}

    except Exception as e:
        print(f"Erro ao adicionar unidade: {e}")
        raise HTTPException(status_code=500, detail="Erro interno no servidor.")
```

Resultados na aplicação em streamlit

Método Get:



Método Post:



5. **Preparação para Uso de Inteligência Artificial com LLMs:**

Nesta etapa, comece a pensar nos dados que você coletou até agora e como eles podem ser utilizados em tarefas baseadas em LLMs nas próximas entregas.

Considere os tipos de dados disponíveis e as possíveis aplicações com LLMs, como:

- **Análise de Texto Gerado: Usar os dados coletados para gerar resumos automáticos de textos longos, facilitando a compreensão e análise de documentos.**
- **Classificação de Sentimentos: Aplicar um modelo de LLM para classificar os sentimentos em textos (positivos, negativos ou neutros) coletados de notícias, redes sociais, ou outras fontes.**
- **Perguntas e Respostas (Q&A): Usar um LLM para construir um sistema de perguntas e respostas a partir dos dados disponíveis, respondendo a perguntas relevantes com base nos conteúdos coletados.**
- **Geração de Texto: Automatizar a criação de relatórios ou insights com base nos dados brutos, utilizando LLMs para gerar textos descritivos.**

Mantenha o foco na preparação do projeto para que, na próxima etapa, as funcionalidades de IA via LLM sejam facilmente integradas e aplicadas a esses cenários.

Com os dados disponíveis pode ser realizado para perguntas e respostas do tipo: “onde consigo realizar a consulta A”, ou ainda “qual unidade existem no município B”