Teste para vaga Fhir

Segue estrutura do projeto teste

> Docs: Algumas documentações para rodar o projeto e Recursos resultados de testes.
> Hapi-fhir-jpa: Projeto HapiFhir JPA com interface web, utilizado como nosso datastore Fhir.
> Service: Scripts e CSV com dados do paciente.

Scripts

> fhir_condition_builder.py : Cria recurso condition para Observações: Hipertensão, Diabetes e Gestação.
> fhir_observation_builder.py : Cria recurso observation para Observações: Hipertensão, Diabetes e Gestação.
> fhir_patient_builder.py: Cria recurso patient para Paciente.
> post_on_fhir.py : Serviço que posta no Fhir (Hapi-fhir-jpa).
> verify_cpf.py: Script que verifica dados do CPF do paciente.

Arquivos

> Service/data : CSV com os dados do paciente fornecido pelo enunciado do teste.

Ferramentas utilizadas na camada Service

> Spark & PySpark (https://github.com/apache/spark)
> chardet (https://pypi.org/project/chardet/)
> datetime (built-in)
> requests (https://pypi.org/project/requests/)
> json (built-in)
> unidecode (https://pypi.org/project/Unidecode/)
> Python 3.11.4 (https://www.python.org/downloads/release/python-3114/)

Rodando o teste

- Na pasta do hapi-fhir-jpa, siga as instruções no readme.pdf ou readme.md
- Na pasta service, no terminal, pode rodar sem aspas > “pip install -r requirements.txt” para que as dependências dos scripts sejam instaladas. Obs: Precisamos do spark para que tudo funcione corretamente.
- Depois do servidor HapiFhir rodando no docker na porta 8080 (Confira se não há nenhum serviço utilizando essa porta)
- Entre na pasta Service e digite o comando > py .\main.py
- Aguarde até que a carga de dados é postada no Fhir.
- Para conferir os dados, pode utilizar as buscas no Postman, Insomnia ou outro app de teste de api desejado
- Buscas que pode realizar para conferir os dados:
- - http://localhost:8080/fhir/Patient?_count=60 - Aqui buscamos todos os pacientes com um limite de 60 paciente na busca (Fhir por padrão só retorna 20)
- - http://localhost:8080/fhir/Condition?_count=60 - Aqui buscamos todos os Conditions criadas, no corpo do recurso fhir temos o atributo “subject” que aponta para o paciente que está com está condição.
- - http://localhost:8080/fhir/Observation?_count=60 - Mesmo principio do Condition, também temos o atributo “subject” que refere ao paciente. Obs: Observation sempre gera um historico.

Porque Hapi-fhir-jpa

> Com o hapi-fhir conseguimos subir de maneira mais simples um servidor que suporte o padrão Fhir.
> Tem código aberto
> Tem interface gráfica web para consultas e validações.

Porque Condition e Observation

> No Fhir tudo é uma estrutura, como uma arvore e sua ramificações, nos nossos dados, temos a flag observações que no diz um detalhe sobre paciente, nele temos as seguintes informações: Gestante, Hipertenso e Diabético. Para que faça sentido para o Fhir, precisamos desses dois recursos para conseguirmos realizar um pequeno composition do prontuario do mesmo, onde, Condition diz realmente o que paciente tem (Ex: Hipertenso, Diabético e etc.) já o Observation nos diz o que esse paciente “sofreu” e ou vai “sofrer” por conta desses detalhes. Exemplo: Condition de paciente X é Hipertenso, logo esse paciente segue em observação pela equipe, a primeira observação que a equipe tem é que sua pressão sistólica / diastólica está com alteração, essa informação e registrada como Observation para o Fhir que em sua ramificação aceita por que o paciente em questão tem a Condition hipertenso. Sendo assim existe vários outros recursos que devem acompanhar esses casos, porém esses dois são os mais “prioritários” para ramificação.

Prazo estipulado

> O prazo do teste foi de 4 dias
> Solução em questão e para fins de estudo e entendimento do Fhir.