

## **Relatório 14 - Prática: Projeto Final**

<Paulo Victor dos Santos>

### **Descrição do Projeto**

Um laudo médico pode ser gerado após uma série de exames clínicos ou após a realização de um diagnóstico por exame de imagem. Como por exemplo, um simples raio-X, uma tomografia mais complexa ou uma ressonância magnética bem detalhada.

Os laudos são elaborados por médicos especialistas na área do exame, eles descrevem em termos técnicos os possíveis achados e as possíveis conclusões. Os laudos possuem uma estrutura básica e seguem algum padrão de dados, sendo eles: dados do paciente, dados do solicitante, datas, justificativas, conduta médica, hipótese diagnóstica, informações adicionais e a assinatura do médico laudador.

### **Problema**

Em suma, os laudos médicos existem para formalizar os possíveis resultados de exames, ou seja, há uma relação de composição. Além disso os laudos facilitam a comunicação entre outros médicos, entre os pacientes e outras partes interessadas. Por serem documentos textuais, há uma dificuldade na recuperação dos dados e das informações dos laudos, que nem sempre estão acompanhados dos exames de origem.

Imagine um grande hospital, com fluxo de centenas de exames e laudos diários. Imagine que neste mesmo hospital, uma equipe de cientista de dados deseja montar uma base de dados de exames de tomografia de crânio com anormalidades diversas. O caminho mais eficiente para criação dessa base é a partir dos laudos médicos e que estes laudos possam levar aos devidos exames de imagem.

Contudo, a simples busca textual conhecida como “full text search” (consulta através de banco relacional), não é eficiente devido à complexidade da linguagem natural envolvida nesses laudos.

### **Oportunidade**

Cada laudo médico, mesmo com a complexidade da área médica envolvida, possui informações relevantes que podem ser extraídas e padronizadas, de forma a facilitar na recuperação dos laudos.

Imagine um grande hospital que tenha um sistema de classificação das especialidades e das modalidades médicas a partir dos laudos e que cada laudo tivesse um cabeçalho com informações adicionais que poderiam ser utilizadas para possíveis agrupamentos e possíveis recuperações.

No cenário hipotético da criação da base de dados de tomografia de crânio com anormalidades, uma possível busca semântica nos laudos já seria suficiente para separar os exames de imagiologia, ou uma simples busca textual em dados pré-classificados também seria uma solução viável para a recuperação desses dados e para alcançar o objetivo proposto. Visto que para os médicos especialistas fazerem a anotação e a classificação desses laudos, é uma tarefa trabalhosa e dispendiosa, com custos elevados.

### **Objetivos – Escopo**

A proposta do sistema visa contemplar o escopo principal de classificação dos laudos médicos através de agentes inteligentes. Onde o agente possa simular o trabalho do médico

especialista e de forma sucinta, classificar a especialidade e a modalidade médica do laudo através de uma tabela de domínio já existente.

Além da classificação para facilitar na recuperação e agrupamento, outro objetivo do sistema é na sumarização dos laudos, de forma a facilitar na interpretação por parte dos pacientes. Suponha um laudo extenso e complexo, com linguagem específica da área médica. Esse laudo se torna um empecilho para o paciente, que por sua vez precisa de um profissional para realizar a interpretação dele.


Os objetivos são: classificar os laudos pela especialidade e modalidade e gerar sumarização dos laudos que serão fornecidos aos pacientes.


## Detalhamento Técnico


### Dataset


A base de dados utilizada para o desenvolvimento do MVP (produto mínimo viável), trata-se de uma base de dados privada desenvolvida no Hospital Israelita Albert Einstein. Essa base utiliza dados de pacientes fictícios e com descrições fidedignas, ou seja, os laudos possuem conteúdos reais, porém os pacientes e os médicos laudadores não existem. Todas as informações sensíveis presentes nos laudos foram substituídas por dados fictícios.


Uma amostra da base de dados está presente no código do sistema, esta amostra foi utilizada nas etapas de desenvolvimento e testes. As imagens abaixo exibem partes da base de dados e de como o laudo é estruturado.


 PID\_563032.ACN\_8960684-428.pdf


 PID\_701601.ACN\_9632341-15.pdf


 PID\_721449.ACN\_9650156-302.pdf

 PID\_774079.ACN\_9677116-14.pdf

 PID\_781361.ACN\_9686661-13.pdf

 PID\_781361.ACN\_9686661-40.pdf

 PID\_802562.ACN\_9720704-13.pdf

 PID\_813347.ACN\_9693995-15.pdf

O laudo possui dados do paciente, título informativo e descritivo sobre a análise clínica, observações importantes com medições e finalmente, a conclusão com a opinião do médico laudador a respeito dos achados que foram evidenciados no exame que gerou o laudo, conforme a imagem abaixo.

## ULTRASSONOGRAFIA DA TIREOIDE COM DOPPLER COLORIDO

Glândula tireoide tóxica, com morfologia normal e contornos regulares.  
Ecogenidade preservada e ecotextura heterogênea.  
Diminuta imagem hipocogênica medindo 0,3 cm no lobo esquerdo, não se podendo diferenciar entre um nódulo e um cisto.

### Observam – se:

#### TIRADS 2

Formação nodular à direita, de componente misto, bem delimitada, apresentando focos hiperecóticos, com reverberação posterior, de aspecto esponjiforme, compatível com padrão colóide, medindo 0,7 x 0,5 cm.  
Padrão II de Chammas.

#### TIRADS 4B

Nódulo hipocóico, hipervascularizado, com fino halo periférico, localizado no terço inferior do lobo esquerdo e medindo 2,7 x 1,4 cm.  
Padrão III de Chammas.

### Avaliação Dopplervelocimétrica ( Classificação de Chammas et al ):

- Padrão I:** ausência de vascularização;
- Padrão II:** apenas vascularização periférica;
- Padrão III:** vascularização periférica maior ou igual a central;
- Padrão IV:** vascularização central maior que periférica;
- Padrão V:** apenas vascularização central.

Estudo de 177 nódulos com modo B,doppler,PAAF. Otolaryngol Head Neck Surg. 2005 Jun;132(6):874-82.

### Biometria da glândula tireoide:

- **Lobo direito** medindo 2,3 x 1,6 x 5,0 cm.
- **Lobo esquerdo** medindo 2,2 x 1,8 x 5,2 cm.
- O **istmo tireoidiano** mede 0,4 cm de espessura.
- Volume **tireoidiano** estimado 20,3 cm<sup>3</sup>.
- Ausência de linfonodomegalias cervicais regionais

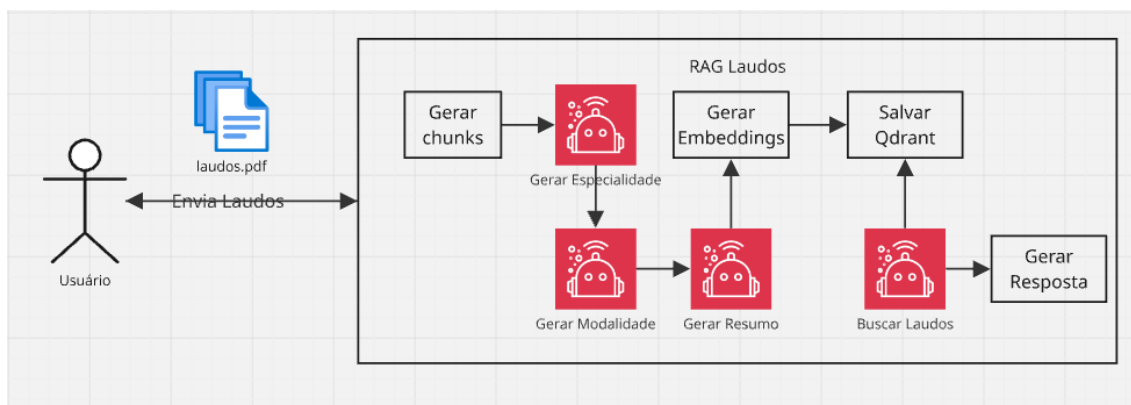
### CONCLUSÃO:

- **Nódulos tireoideanos conforme descritos.**
- **Categoria TIRADS 4B.**

## Arquitetura – Diagrama

A arquitetura do sistema proposto, segue o fluxo principal conforme o diagrama abaixo. O usuário envia um laudo para o sistema, o sistema processa o laudo em pequenas partes. O agente de inteligência artificial gera a especialidade médica, a modalidade e o resumo do laudo. Esses dados são vetorizados e salvos em uma base de dados vetorial, denominado Qdrant.

Após a indexação dos laudos, o usuário poderá interagir com os laudos utilizando a tela de chat do sistema. Ao solicitar ou perguntar alguma informação referente aos laudos, um agente de inteligência artificial se conecta na base vetorial, busca e gera uma resposta humanizada para o usuário.



## Dados de Entrada

Os laudos enviados para o sistema, são transformados em objetos de negócio. Esses objetos utilizam a ferramenta pydantic, que ajuda na criação e validação das informações fornecidas.

A partir do arquivo PDF do laudo, são extraídas pequenas partes do laudo, essas partes posteriormente são transformadas em vetores e armazenados na base de dados. A busca por similaridade utiliza cálculos matemáticos a partir desses vetores para gerar contexto semântico entre as palavras. O pydantic tem papel fundamental nesse processo, pois facilita a conversão dos dados, o tráfego das informações dentro do sistema e a qualidade dessas informações que serão armazenadas.

A imagem abaixo representa um objeto escrito na linguagem de programação python, com as regras de validação da ferramenta pydantic. O laudo em PDF, é transformado nesse objeto, contendo identificação única, nome do arquivo PDF, texto quebrado, especialidade médica, modalidade médica, texto completo do laudo, resumo e os vetores que serão armazenados na base de dados.

```
id: UUID
name: str
texto: str
especialidade: Optional[str] = None
modalidade: Optional[str] = None
page_content: str
sumarizacao: Optional[str] = None
embedding: List
```

## Fluxo Langflow

A Solução proposta não utiliza a ferramenta Langflow, trata-se de um sistema on-primese, ou seja, um sistema que não necessita de servidores e provedores para sua execução.

A escolha da utilização de agentes inteligentes da ferramenta CrewAI se mostrou eficiente para a proposta em questão, por sua facilidade, versatilidade e independência. Os agentes inteligentes com CrewAI, fazem parte da arquitetura do sistema, não são serviços publicados em servidores externos.

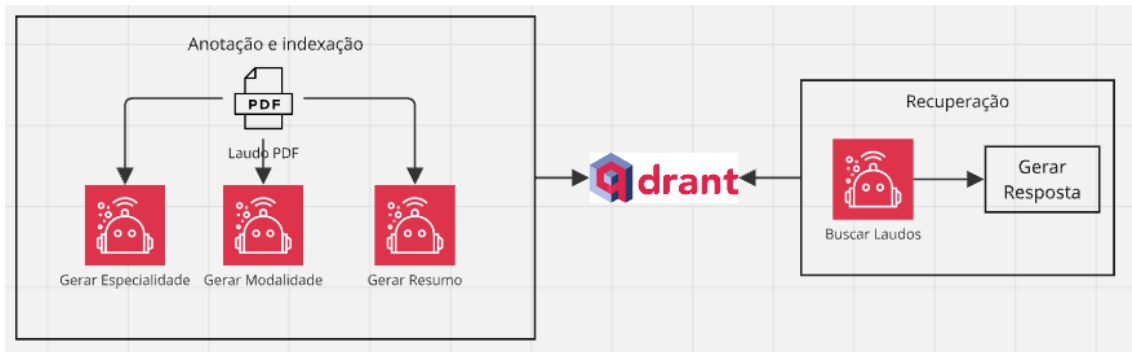
## Embeddings

Para a geração de embeddings, o modelo "all-MiniLM-L6-v2" foi utilizado como gerador de vetores. Este modelo é leve trazendo portabilidade e eficiência para a proposta.

A geração de vetores é utilizada nas partes menores extraídas dos laudos (chunks), esses pequenos textos são submetidos ao modelo, que por sua vez, gera os vetores para que possam ser salvos na base de dados. A utilização de um modelo padrão, faz com que vetores gerados de textos similares, também possuem similaridade matemática. Essa similaridade dá expertise para que um agente de inteligência artificial consiga buscar dados similares ou sinônimos, e a partir dessa similaridade possa gerar respostas humanizadas com base no contexto inicial.

## Orquestração De Multiagentes

Foram utilizados quatro agentes inteligente no sistema. Três agentes no primeiro fluxo do sistema, que corresponde a etapa de anotação e indexação do laudo na base de dados. E um agente para gerar a busca e a respostas para o usuário, caso o usuário interaja com o sistema.



A imagem acima representa a orquestração dos agentes com a ferramenta CrewAI. Na etapa de indexação os agentes e tarefas são orquestrados de forma sequencial, cada um gerando sua resposta, essa estratégia trás velocidade para o sistema, visto que tarefas menores são executadas mais rápidas do que tarefas maiores e mais complexas.

Na segunda etapa do projeto, na fase de “retriever”, apenas um agente é orquestrado pela ferramenta CrewAI, esse agente é munido de uma ferramenta capaz de se conectar na base de dados vetorial, realizar a busca semântica e entregar os dados similares ao agente, que por sua vez trata os textos brutos e gera uma resposta humanizada a ser enviada ao usuário.

## Interface Com Usuário

Streamlit é uma biblioteca utilizada na linguagem de desenvolvimento python, essa biblioteca facilita a criação de aplicativos web de forma interativa através da própria linguagem de programação, ou seja, não necessita de um especialista em designer para a programação da interface do sistema.

A partir dessa biblioteca foi possível criar o protótipo da interface da proposta, conforme a imagem abaixo.



O menu lateral esquerdo, o usuário consegue enviar laudos médicos para o sistema e posteriormente interagir com esses laudos. Ainda no menu, é exibido a quantidade de laudos

já existentes na base de dados do sistema. A quantidade de laudos auxilia na contagem e na verificação da inclusão de novos laudos.

Na parte central do protótipo, está localizado o título “RAG De Laudos Médicos”, abaixo está localizado o campo de entrada de texto que o usuário poderá utilizar para interagir com o sistema. As interações podem ser perguntas qualitativas ou quantitativas a respeito dos laudos já indexados no sistema. Através dos agentes de inteligência artificial as interações e respostas são humanizadas, ou seja, os textos são simplificados para que haja fácil entendimento por parte do usuário.

### **Integração com Comunicação**

A proposta não utiliza comunicação direta com usuário a partir de ferramentas como WhatsApp. O escopo bem definido com base de dados reais, não proporciona interação com ferramentas de mensagens instantâneas.

Caso houvesse dados telefônicos nos laudos, poderia ter o requisito de envio de mensagens instantâneas para o paciente ou médico. Contudo, entende-se que o sistema proposto estará inserido em um ambiente de contexto médico e que a atuação e interação direta com o sistema será restrito a determinados perfis.

A parte de busca, simula um paciente externo interagindo com a inteligência artificial questionando e pedindo informações simples a respeito de um determinado laudo. Porém, como se trata de um MVP, não foi implementado perfis de acesso, cadastros de usuários ou quaisquer outros dados que possam ser úteis para interação ou comunicação com o usuário.

### **Desenvolvimento e Testes**

O desenvolvimento do MVP foi realizado em linguagem de programação python. Foram aplicados conceitos de arquitetura de software, como orientação a objetos, organização e responsabilidades, baixo acoplamento e alta coesão. Todos esses detalhes são úteis e garantem que o fluxo principal do sistema esteja funcional e que possa se recuperar de possíveis falhas, caso aconteçam.

É de responsabilidade da arquitetura de software, garantir que o sistema possa ser evoluído ou utilizado em larga escala. A partir de componentes bem definidos é possível realizar a integração de bibliotecas, ferramentas e outros conceitos para atendimento do escopo principal.

A imagem abaixo demonstra como a solução está organizada e responsabilidade de cada componente do sistema. Esta organização é essencial para manutenibilidade e evolução da solução. O fluxo principal proposto no escopo, trata-se de uma solução para classificação de laudos médicos utilizando agentes inteligentes e a recuperação dos dados desses laudos de forma amigável também por agentes inteligentes.

```
├─ data/
│   └─ Protocolo.py           # Classe de modelagem de negócio
├─ docs/                     # Dataset de laudos anonimizados e privados
├─ src/
│   └─ Util.py               # Classe utilitária para suporte ao processamento de textos e PDFs
│   └─ Enumeradores.py       # Classe de domínio da área médica
│   └─ Agentes.py            # Classe responsável por manter os agentes inteligentes
│   └─ QdrantConection.py     # Integrações com o banco de dados Qdrant para contagem de dados
├─ app.py                    # Arquivo principal contendo a lógica do Streamlit
├─ run_debug.py              # Facilita o desenvolvimento no Streamlit, possibilitando depurar
├─ testar_fluxo.py           # Simula o fluxo principal sem a necessidade do Streamlit
├─ requirements.txt          # Lista de dependências para rodar o projeto
└─ README.md                 # Documentação do projeto
```

O escopo principal da solução é a interação do usuário com laudos médicos, seja indexando os laudos no sistema, ou seja, interagindo com laudos já indexados. As imagens abaixo, exibem um teste de cobertura, onde um laudo específico foi selecionado e perguntas referentes a este laudo foram submetidas ao sistema.

<b>Nome</b>	: Luiz Fernando L F Pereira	<b>Leito</b>	: SAI15 SAI15
<b>Idade</b>	: 3A 7M	<b>Prontuário</b>	: 272727
<b>Sexo</b>	: Masculino	<b>Data</b>	: 31 maio 2013

**TOMOGRAFIA COMPUTADORIZADA DA FACE**

**TÉCNICA:**  
**Método:** Helicoidal – Multislice. **Colimação:** 0,5mm, reconstruções multiplanares. **Contraste:** Não.

**ACHADOS:**  
Ossos nasais e processos frontais da maxila íntegros.  
Septo nasal centrado.  
Velamento do seio maxilar esquerdo e espessamento do revestimento mucoso do seio maxilar direito, seios esfenoidais e de algumas células etmoidais.  
Obliteração das vias de drenagem sinusais por componente mucoso dos infundíbulos e dos recessos esenoetmoidais.  
Ossos temporais sem particularidades.  
Estruturas orbitárias preservadas.  
Glândulas parótidas e submandibulares sem alterações. Ausência de dilatações ductais ou cálculos.  
Maxila e mandíbula com aspecto normal. Canais dos nervos alveolares inferiores e forames mandibulares preservados.  
Rino e orofaringe livres.  
Estruturas vasculares conservadas ao estudo sem contraste.  
Linfonodos cervicais aumentados em número medindo até 1,3 cm no seu maior eixo axial, nos níveis I, II, III, IV e V bilaterais.  
Achados do crânio descritos em estudo específico da mesma data.

**Relatado por:** Dr. Ricardo Bastos - CRM 32322  
**Revisado e Assinado Eletronicamente por:** Dr. Ricardo Bastos - CRM 32322

O resultado do teste pode ser observado na imagem abaixo. Acima, está o laudo, onde relata-se achados com contexto de septo nasal centrado. O teste de cobertura é eficiente pois

evidencia a partir de uma entrada conhecida a resposta esperada para tal entrada, conforme a imagem abaixo.

## RAG De Laudos Médicos

Faça uma pergunta sobre os laudos médicos

há algum laudo sobre septo nasal no sistema?

O laudo da tomografia computadorizada da face de Luiz Fernando L F Pereira, 3 anos e 7 meses, informa que o "septo nasal está centrado". O laudo também descreve velamento do seio maxilar esquerdo e espessamento do revestimento mucoso do seio maxilar direito, obstrução das vias de drenagem sinusais e aumento de linfonodos cervicais. O score de similaridade da consulta é 0.45811374133661203. Os metadados do laudo incluem a especialidade "Otorrinolaringologia" e a modalidade "Tomografia Computadorizada". O laudo foi relatado por Dr. Ricardo Bastos - CRM 32322.

Destaca-se a resposta humanizada gerada pelo agente de inteligência artificial, que a partir de um laudo complexo e específico o sistema consegue resumir de forma clara e concisa. Além de evidenciar assertivamente a especialidade e a modalidade médica com base no laudo fornecido.

### Resultados Obtidos

Concluo este desafio com um enorme aprendizado de uma solução simples e muito eficiente, ao estudar sobre a ferramenta de leitura de CSV, aprendi que também existem ferramentas que já fazem leitura de dados diretamente em bancos relacionais, o que pode ser muito útil para desenvolvimento de sistemas com agentes integrados diretamente a base de dados de outros sistemas.

### Instruções de Uso

Concluo

Para a utilização deste MVP é necessário realizar a clonagem do repositório dos códigos, que estão disponíveis no repositório: <https://github.com/duvictor/fastcamp agentes projeto final.git> na branch "Master".

```
git clone https://github.com/duvictor/fastcamp agentes projeto final.git
cd chat-laudo
```

Após a clonagem do código fonte, basta configurar o ambiente virtual python. É necessário que haja o python na versão 3.11 ou superior instalado na máquina.

Para criar o ambiente virtual, basta executar o código abaixo na IDE de preferência ou no terminal do computador.

```
python -m venv venv
source venv/bin/activate          # macOS/Linux
venv\Scripts\activate
```



Após a criação do ambiente virtual, é necessário criar o arquivo de variáveis de ambiente `.env` conforme a imagem abaixo. Este arquivo deve estar na raiz do código fonte e seu conteúdo deve conter a chave de acesso aos modelos de LLM da OpenAI e o nome da coleção de dados vetoriais que será criado localmente na máquina.

```
OPENAI_API_KEY=12345  
COLLECTION_NAME=laudos_medicos
```

Instale as dependências através do comando abaixo.

```
pip install -r requirements.txt
```

E finalmente, execute o comando abaixo para que o sistema se inicialize. Uma janela do navegador será aberta com a interface do sistema.

```
streamlit run app.py
```

Em caso de dúvidas, no link do repositório contém mais informações técnicas a respeito da solução.

## Referencias

<https://levelup.gitconnected.com/building-a-rag-application-using-streamlit-langchain-and-deepseek-r1-7e7225e598ae>

<https://medium.com/@habbema/estruturando-projetos-em-python-692502641c05>

<https://eskelsen.medium.com/estruturando-projetos-em-python-um-modelo-de-sistema-d0652d289bc>

<https://medium.com/@vikrambhat2/agentic-rag-mastering-document-retrieval-with-crewai-deepseek-and-streamlit-21cb3886bbbf>

<https://ai.gopubby.com/multi-agent-system-for-research-summarization-and-reporting-with-crew-ai-c5e41a712c25>

<https://www.datacamp.com/tutorial/agentic-rag-tutorial>

<https://medium.com/the-ai-forum/build-a-local-reliable-rag-agent-using-crewai-and-groq-013e5d557bcd>

<https://github.com/benitomartin/crewai-rag-langchain-qdrant>

<https://medium.aiplanet.com/retrieval-augmented-generation-using-qdrant-huggingface-embeddings-and-langchain-and-evaluate-the-3c7e3b1e4976>

<https://medium.com/@erickcalderin/classic-rag-with-huggingface-qdrant-and-streamlit-4a0918b20fa7>

<https://qdrant.tech/blog/webinar-crewai-qdrant-obsidian/>

<https://qdrant.tech/documentation/agentic-rag-crewai-zoom/>

<https://github.com/qdrant/examples/blob/master/rag-with-qdrant-deepseek/deepseek-qdrant.ipynb>

<https://qdrant.tech/documentation/agent-ic-rag-crewai-zoom/>

<https://docs.crewai.com/tools/qdrantvectorsearchtool>

<https://huggingface.co/blog/Andyrasika/qdrant-transformers>

<https://discuss.streamlit.io/t/cannot-debug-streamlit-in-pycharm-2023-3-3/61581>

<https://python.langchain.com/docs/integrations/vectorstores/qdrant/>

<https://qdrant.tech/documentation/data-ingestion-beginners/#:~:text=To%20start%20visualizing%20your%20data,and%20select%20Access%20the%20database.&text=The%20first%20query%20retrieves%20all,third%20performs%20a%20sample%20query.>

<https://docs.crewai.com/tools/qdrantvectorsearchtool>