



# Imersão em GenAI para Desenvolvedores

## Projeto 5: *Retrieval-Augmented Generation* (RAG)

*Preâmbulo: Neste projeto, você irá implementar um sistema de armazenamento vetorial com ChromaDB para buscas semânticas em currículos, utilizar técnicas de chunking para melhorar a busca em PDFs, e criar uma aplicação web com Streamlit integrando RAG para análise de qualificações de candidatos.*

*Versão: 1.0*

# Sumário

<b>I</b>	<b>Preâmbulo</b>	<b>2</b>
<b>II</b>	<b>Instruções gerais</b>	<b>3</b>
<b>III</b>	<b>Ex00 – Embeddings para otimizar a busca de informações</b>	<b>5</b>
III.1	Exercício . . . . .	5
<b>IV</b>	<b>Ex01 – RAG: Busca semântica e LLMs entram em um bar</b>	<b>8</b>
IV.1	Exercício . . . . .	9
<b>V</b>	<b>Entrega e Avaliação entre pares</b>	<b>11</b>
V.1	Processo de Entrega . . . . .	11
V.2	Avaliação entre pares . . . . .	11
V.3	Dicas para uma avaliação bem-sucedida . . . . .	11

# Capítulo I

## Preâmbulo

*Os avanços recentes em modelos de linguagem têm sido notáveis, mas ainda enfrentam desafios importantes. Lewis e seus colegas (2020) observaram que, apesar de sua capacidade de armazenar conhecimento factual, esses modelos ainda têm dificuldade em acessar e manipular esse conhecimento com precisão em tarefas que exigem informações extensas.*

Para lidar com essa limitação, os autores propuseram uma nova abordagem chamada Retrieval-Augmented Generation (RAG). Embora possa ser traduzida literalmente como "Geração aumentada por recuperação", o termo em inglês é mais comumente usado na literatura. Esse método combina de forma inovadora dois tipos de memória:

- Uma memória paramétrica: um modelo linguístico pré-treinado
- Uma memória não paramétrica: um grande banco de dados de informações (neste caso, a Wikipédia)

O RAG permite que o modelo acesse dinamicamente informações externas durante o processo de geração de texto, resultando em respostas mais precisas e fundamentadas.

Os autores testaram duas variantes do RAG:

1. Uma que usa o mesmo conjunto de informações recuperadas para gerar toda a resposta
2. Outra que pode utilizar diferentes informações para cada parte da resposta

Ao aplicar essa técnica a diversas tarefas de processamento de linguagem natural, os pesquisadores alcançaram resultados impressionantes, superando abordagens anteriores em várias tarefas, incluindo perguntas e respostas de domínio aberto.

*"Para tarefas de geração de linguagem, descobrimos que os modelos RAG geram linguagem mais específica, diversa e factual do que modelos anteriores."*

— Baseado no trabalho de Lewis et al., 2020<sup>1</sup>

---

<sup>1</sup>Retrieval-augmented generation for knowledge-intensive nlp tasks. arXiv preprint arXiv:2005.11401.

# Capítulo II


## Instruções gerais

- Esta página é sua única referência oficial. Não confie em informações não verificadas.
- Os exercícios estão organizados em ordem crescente de complexidade. É essencial dominar cada exercício antes de prosseguir para o próximo.
- Preste atenção às permissões de seus arquivos e pastas.
- Siga rigorosamente o procedimento de entrega para todos os exercícios.
- Seus exercícios serão avaliados por colegas da Imersão.
- Para exercícios em Shell, utilize `/bin/zsh`.
- Mantenha em sua pasta apenas os arquivos explicitamente solicitados nos enunciados.
- Em caso de dúvidas, consulte seus colegas à direita ou à esquerda.
- Utilize recursos como Google, manuais online e a Internet como referência.
- Leia os exemplos com atenção. Eles podem conter requisitos não explicitamente mencionados no enunciado.
- Para exercícios em Python:
  - Use a versão do Python especificada no exercício de configuração do ambiente.
  - Utilize os modelos e provedores sugeridos para garantir tempos de resposta adequados e consistência nos testes.
- Esteja atento a erros em todos os exercícios. Eles raramente são tolerados durante a avaliação.
- **Aviso sobre o uso de ferramentas de AI (como ChatGPT):**
  - O uso de ferramentas como o ChatGPT não deve ser encarado como um substituto para seu próprio esforço e entendimento.
  - O aprendizado efetivo ocorre quando você interage ativamente com o conteúdo: pesquisando, refletindo e aplicando o que aprendeu.
  - Nas avaliações, serão feitas perguntas para avaliar sua compreensão real sobre o assunto.

- E durante as avaliações, seus colegas também avaliarão seu nível de conhecimento.

## Capítulo III

# Ex00 – Embeddings para otimizar a busca de informações

	Exercício : 00
Implementar um sistema de armazenamento vetorial usando ChromaDB para processar e realizar buscas semânticas eficientes em currículos, simulando um assistente de recrutamento baseado em IA	
Pasta de entrega : <i>ex00/</i>	
Arquivos para entregar : <code>resume_analyzer.py</code>	

Você vai criar um sistema de análise de currículos que busca por qualificações. Este sistema permitirá:

- Processar currículos em PDF, convertendo-os em representações vetoriais densas (conhecidos como *embeddings*<sup>1</sup>).
- Armazenar e indexar eficientemente essas representações para busca rápida.
- Realizar consultas em linguagem natural sobre o conteúdo dos currículos.

Com esta ferramenta, você poderá analisar um grande volume de currículos, utilizando buscas semântica<sup>2</sup>. Isso eliminará a necessidade de leitura individual de cada documento, permitindo uma triagem rápida e mais precisa dos candidatos mais adequados às suas necessidades.

### III.1 Exercício

#### Instruções

1. Configurar o ChromaDB com persistência local.

<sup>1</sup><https://www.cloudflare.com/learning/ai/what-are-embeddings/>

<sup>2</sup><https://www.elastic.co/what-is/semantic-search>

2. Implementar a função `process_pdf_directory` para ler e processar arquivos PDF.
3. Adicionar os documentos processados ao ChromaDB.
4. Criar a função `interactive_query_loop` para realizar consultas interativas.

## Código base:

```
def main():
    persist_directory = "./chroma_data"
    pdf_directory = "./pdfs"

    # TODO: Configurar ChromaDB com persistência local
    chroma_client = ...

    # TODO: Criar embedding function
    # Dica: Experimente com diferentes modelos, como "paraphrase-multilingual-MiniLM-L12-v2"
    embedding_function = ...

    # TODO: Criar ou obter uma coleção existente
    collection = ...

    # TODO: Implementar a função process_pdf_directory
    process_pdf_directory(pdf_directory, collection)

    # TODO: Implementar a função interactive_query_loop
    interactive_query_loop(collection)

if __name__ == "__main__":
    main()
```

Dica para a função `interactive_query_loop`:

```
while True:
    query = input("\nConsulta: ")
    if query.lower() == 'sair':
        break
    results = ...
    print("\nResultados:")
    for document, metadata in ...:
        print(f"Documento: {metadata['source']}")
        print(f"Trecho: {document[:200]}..." # Apenas os 200 primeiros caracteres
    print()
```

## Saída esperada:

```
Encontrados 50 arquivos PDF no diretório.
Processando PDF 1/50: curriculo_1.pdf
- Documento curriculo_1.pdf processado e armazenado.
Processando PDF 2/50: curriculo_2.pdf
- Documento curriculo_2.pdf processado e armazenado.
...
Processando PDF 50/50: curriculo_50.pdf
- Documento curriculo_50.pdf processado e armazenado.
Processamento de todos os PDFs concluído.
Processamento concluído. Iniciando modo de consulta.
Digite sua consulta ou 'sair' para encerrar.

Consulta: python

Resultados:
```

```
Documento: curriculo_12.pdf
Trecho: Maria Silva
São Paulo, SP | (11) 9XXXX-XXXX | maria.silva@42sp.org.br
Resumo Profissional
Desenvolvedora de Software Júnior com 2 anos de experiência em Python e frameworks web...

Documento: curriculo_35.pdf
Trecho: João Santos
São Paulo, SP | (11) 9XXXX-XXXX | joao.santos@42sp.org.br
Resumo Profissional
Engenheiro de Machine Learning com forte background em Python e bibliotecas de data science...

Documento: curriculo_7.pdf
Trecho: Ana Oliveira
São Paulo, SP | (11) 9XXXX-XXXX | ana.oliveira@42sp.org.br
Resumo Profissional
Analista de Dados com experiência em manipulação e visualização de dados utilizando Python...
```



Há currículos disponíveis para download na página principal do projeto.




Ainda não estamos usando modelos de linguagem para processamento ou geração de texto. O foco está na criação e consulta de *embeddings*.



## Capítulo IV

### Ex01 – RAG: Busca semântica e LLMs entram em um bar

	Exercício : 03
Criar uma aplicação web usando Streamlit que integre processamento de currículos, busca semântica e um modelo de linguagem para analisar e responder perguntas sobre as qualificações dos candidatos, utilizando <i>Retrieval-Augmented Generation</i> (RAG)	
Pasta de entrega : <i>ex03/</i>	
Arquivos para entregar : <code>resume_analyzer_app.py</code> , <code>project_explanation.md</code>	

*Retrieval-Augmented Generation* (RAG) é uma técnica poderosa que combina a precisão da busca semântica com a flexibilidade dos modelos de linguagem de grande escala (LLMs). Esta abordagem permite gerar respostas mais precisas e contextualizadas, especialmente útil em cenários como análise de currículos, onde informações específicas e detalhadas são cruciais.

#### Como RAG funciona

1. **Recuperação:** Utiliza busca semântica para encontrar informações relevantes em uma base de conhecimento (neste caso, currículos processados).
2. **Contextualização:** Fornece o contexto recuperado ao modelo de linguagem.
3. **Geração:** O LLM gera uma resposta baseada tanto na pergunta quanto no contexto fornecido.

## IV.1 Exercício

### Instruções

1. Configurar o ambiente:
  - Configurar ChromaDB para armazenamento e busca de embeddings.
  - Preparar a integração com o Groq para utilizar o modelo llama3-8b-8192.
2. Implementar o processamento de currículos:
  - Utilizar a API de chunking do exercício anterior para processar os PDFs enviados.
  - Armazenar os chunks processados no ChromaDB.
3. Desenvolver a função de busca semântica utilizando ChromaDB.
4. Criar uma função para gerar respostas com o LLM via Groq, utilizando o contexto recuperado.
5. Construir a interface Streamlit com os seguintes componentes:
  - Opção para upload de múltiplos currículos.
  - Campo de entrada para perguntas sobre os candidatos.
  - Área para exibição das respostas geradas.
6. Gerar documentação automatizada do projeto usando Continue.dev:
  - Utilizar o Continue.dev para analisar o código do projeto e gerar uma documentação do tipo Explanation<sup>1</sup> em um único arquivo Markdown.
  - O arquivo de documentação (project\_explanation.md) deve seguir este formato:

```
# Análise de currículos com RAG

## Visão geral do sistema
[Breve explicação do propósito e funcionamento geral do sistema]

## Componentes principais
1. [Nome do Componente 1]
   - [Breve explicação do componente e seu propósito]

2. [Nome do Componente 2]
```

---

<sup>1</sup><https://docs.divio.com/documentation-system/>

```
- [Breve explicação do componente e seu propósito]

...

## Conceitos principais
- [Conceito 1]: [Breve explicação]
- [Conceito 2]: [Breve explicação]
...

## Fluxo de funcionamento
[Explicação sucinta de como os componentes interagem]
```

- Revisar e ajustar a documentação gerada conforme necessário.

## Código base:

```
import streamlit as st

# TODO: Adicione os imports necessários

# Interface Streamlit
st.title("Análise de currículos")

query = st.text_input("Faça uma pergunta sobre os candidatos:")
if query:
    # TODO: Implementar a lógica de busca e geração de resposta
    pass
```

## Saída esperada:

Uma aplicação Streamlit funcional que permite o upload de currículos, realiza buscas semânticas e responde a perguntas sobre os candidatos usando um modelo linguístico.



Use `st.spinner()` para operações longas



Não deixe a sua chave de API no código submetido. Durante a avaliação, você precisa configurá-la adequadamente.

# Capítulo V

## Entrega e Avaliação entre pares

### V.1 Processo de Entrega

- Submeta seu trabalho no repositório Git gerado na página principal do projeto.
- Certifique-se de que todos os arquivos necessários estejam incluídos e organizados conforme as instruções do projeto.
- Respeite o prazo de entrega estabelecido.

### V.2 Avaliação entre pares

- Seu projeto será avaliado por um dos seus colegas.
- A avaliação focará na qualidade do seu código e na aderência aos requisitos do projeto.
- Critérios de avaliação podem incluir:
  1. Funcionalidade: O código atende a todos os requisitos especificados?
  2. Legibilidade: O código é claro e bem estruturado?
  3. Eficiência: As soluções implementadas são otimizadas e seguem boas práticas?
  4. Organização: Os arquivos e estrutura do projeto estão bem organizados?
- Feedback detalhado é esperado, mas pode variar em extensão e detalhamento.

### V.3 Dicas para uma avaliação bem-sucedida

- Revise seu código antes da submissão final.

- Teste exaustivamente todas as funcionalidades implementadas.
- Se entender necessário para clareza, documente claramente qualquer decisão ou suposição feita.
- Esteja preparado para explicar suas escolhas de implementação.



A avaliação entre pares é uma oportunidade para aprendizado e crescimento pessoal e profissional. Esteja aberto ao feedback recebido e use-o para aprimorar suas habilidades.