Universidade Federal do Estado do Rio de Janeiro
Centro de Ciências Exatas e Tecnológicas
Escola de Informática Aplicada

Retrieval Augmented Generation Aplicada à Bibliotecas

Breno Costa da Silva Filgueiras

Rio de Janeiro, RJ – Brasil
Dezembro, 2024

Retrieval Augmented Generation Aplicada à Bibliotecas

Breno Costa da Silva Filgueiras

Projeto de graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) como cumprimento de requerimento parcial para obtenção título de Bacharel em Sistemas de Informação.

Approved by:

Supervisor, D.Sc. – UNIRIO

Supervisor 2, D.Sc. – UNIRIO

Supervisor 3, D.Sc. – XXXX

Rio de Janeiro, RJ – Brasil

Dezembro, 2024

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Resumo

Em uma parceria entre Seagate e a International Data Corporation (IDC) foi realizado o estudo “The Digitization of the World From Edge to Core”, nele a IDC fala sobre diversos aspectos referentes aos dados presentes no mundo digital e um dos tópicos abordados no estudo é “Mankind is on a quest to digitize the world” e neste mesmo tópico eles explicam que os dados que geramos no dia a dia está em constante crescimento, ou seja, estamos gradualmente produzindo mais dados.

Com um volume cada vez maior de dados, uma busca por informação otimizada é essencial, dado que são necessárias ferramentas que nos garantam confiança e precisão da informação adquirida. Com isso em mente, este trabalho visa o desenvolvimento de um sistema capaz de ler, processar e armazenar documentos diversos de determinada biblioteca (conjunto de documentos) para que possamos utilizar um Large Language Model (LLM) para responder perguntas que os usuários possam ter acerca dos documentos.

A ideia é conseguir processar documentos de diferentes épocas, temas, formatos e conseguir responder o maior número possível de perguntas dos usuários com a melhor confiança possível.

Palavras-chave: retrieval, augmented, generation, inteligência, artificial.

Abstract

In a partnership between Seagate and the International Data Corporation (IDC), the study "The Digitization of the World From Edge to Core" was conducted. In it, IDC discusses various aspects related to data present in the digital world and one of the topics covered in the study is "Mankind is on a quest to digitize the world". In this same topic, they explain that the data we generate on a daily basis is constantly growing, that is, we are gradually producing more data.

With an ever-increasing volume of data, an optimized search for information is essential, given that tools are needed that guarantee reliability and accuracy of the information acquired. With this in mind, this work aims to develop a system capable of reading, processing and storing various documents from a given library (set of documents) so that we can use a Large Language Model (LLM) to answer questions that users may have about the documents.

The idea is to be able to process documents from different periods, themes and formats and to be able to answer as many user questions as possible with the greatest possible confidence.

Keywords: retrieval, augmented, generation, artifical, inteligence.

Conteúdo

Lista de Figuras

Lista de Tabelas

List of Algorithms

1 Introdução

1.1 Motivação

Recentemente precisei implementar uma solução de onboarding para funcionários de uma determinada empresa. Por ser uma empresa grande, diversas regras e normas estavam distribuidas em inúmeros documentos (documentos de formatos distintos e sem uma padronização específica), o que gerava uma dor para funcionários recém contratados, que nem sempre sabiam qual documento consultar.

Para implementar esta solução, foi escolhida uma abordagem que utilizasse o conceito de Retrieval Augmented Generation (RAG) para trazer aos funcionários a informação buscada no menor tempo possível e com confiança de que a informação é válida. Ao longo da implementação da solução, lembrei de todas as vezes que precisei ler um artigo, livro ou até slides só por conta de um determinado tópico ou assunto e o tempo que gastei procurando uma informação que nem sempre seria útil, seja por falta de referência ou a própria informação.

Com isso em mente, busquei implementar um projeto que seria capaz de processar uma biblioteca de documentos e implementar o conceito de RAG para que um usuário seja capaz de fazer perguntas em uma interface de chat simples e com base nos dados processados da biblioteca de documentos, um Large Language Model (LLM) irá retornar uma resposta humanizada contendo uma resposta e a referência, de qual documento veio a resposta, ao usuário.

O tema de busca por informação é importante, pois na internet ainda encontramos diversos dados sem referência ou representados de maneiras distintas (como em um gráfico e em um texto descritivo, ambos com a mesma informação), alguns dados também podem estar alocados em aplicações pouco intuitivas o que acaba aumentando o tempo de busca por informação, seja um site difícil de navegar ou um portal com o mecanismo de busca ruim. Todo esse tempo investido na busca por uma informação, nem sempre é vantajoso para estudantes e pesquisadores, o que pode dificultar pesquisas e trabalhos a longo prazo.

1.2 Objetivos

O principal objetivo deste trabalho é provar que é possível implementar uma solução RAG para bibliotecas de documentos específicos, neste caso os Trabalhos de Conclusão de Curso (TCCs), disponíveis na biblioteca de publicações do curso de Bacharelado em Sistemas de Informação (BSI) da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) [tccs_unirio<empty citation>].

Mais especificamente, estarei usando como biblioteca todos os TCCs de 2023 disponíveis na biblioteca do curso. Embora sejam do mesmo curso e ano, nem todos os trabalhos possuem a mesma estrutura, os alunos são livres para escrever seus trabalhos de diversas formas e não há como garantir uma padronização comum à eles, tornando o processamento dos documentos mais complexo. No entanto, todos os documentos disponíveis na bibliotecas são do tipo Portable Document Format (PDF) com extensão .pdf, isso vai contribuir durante a etapa de implementação dos extratores mais a frente no projeto.

1.3 Metodologia

Neste projeto irei utilizar uma abordagem de Design Science Research (DSR) para que ao final do projeto o artefato modelado esteja implementado e funcionando como planejado.

O DSR tem suas raízes na engenharia e nas ciências do artificial [simon_1996<empty citation>
É uma metodologia fundamental para a resolução de problemas, buscando aprimorar o conhecimento humano com a criação de artefatos inovadores e a geração de conhecimento de design por meio de soluções para problemas do mundo real [design_science<empty citation>].

Deste modo, ao utilizar o DSR, ao final do projeto haverá um artefato que foi produzido com base na aplicação de tudo estudado e discutido nas próximas seções deste trabalho.

2 Fundamentação

2.1 Conceitos Fundamentais

No livro RAG-Driven Generative AI, do Denis Rothman, ele diz que "Mesmo o modelo mais avançado de Inteligência Artificial (IA) generativa é limitado a responder sobre dados nos quais ela foi treinada." rothman<empty citation> o que nos chama a atenção a um probelma em especial, como fazer para que uma IA saiba responder perguntas referentes a um conjunto específico de dados?

De fato, uma IA não tem como saber o que ela não sabe, não existe conhecimento além dos dados nos quais ela foi treinada. Perguntas fora do contexto do treinamento de uma IA geralmente levam a halucinações, viés e respostas sem sentido. Para isso, foi implementado um framework, ou estrutura, que combina abordagens baseadas em recuperação com modelos generativos, esta estrutura é a Retrieval Augmented Generation (RAG).

A RAG recupera dados relevantes de fontes externas em tempo real e usa esses dados para gerar respostas contextualmente relevantes. Uma de suas principais vantagens é a adaptabiolidade, tendo em vista que a estrutura pode ser aplicada independente do tipo de dado abordado na solução, seja texto, imagens, áudios ou documentos diversos.

2.2 O que é RAG?

Quando um modelo de IA generativa não sabe responder determinada pergunta com precisão, diz-se que ele está alucinando ou apresentando viés, mas, na prática, está apenas gerando respostas sem sentido. Isso ocorre porque o modelo não foi treinado com as informações solicitadas ou por conta de limitações em sua configuração, resultando em sequências prováveis, mas não precisas necessariamente.

A RAG começa onde a IA generativa termina, fornecendo informações que um modelo de LLM não possui para responder com precisão. A RAG otimiza tarefas de recuperação de informações e adiciona os dados recuperados durante a entrada (seja consulta do usuário ou um prompt automatizado), gerando uma saída melhorada e

mais amigável ao usuário. O funcionamento geral do RAG pode ser resumido na figura a seguir:

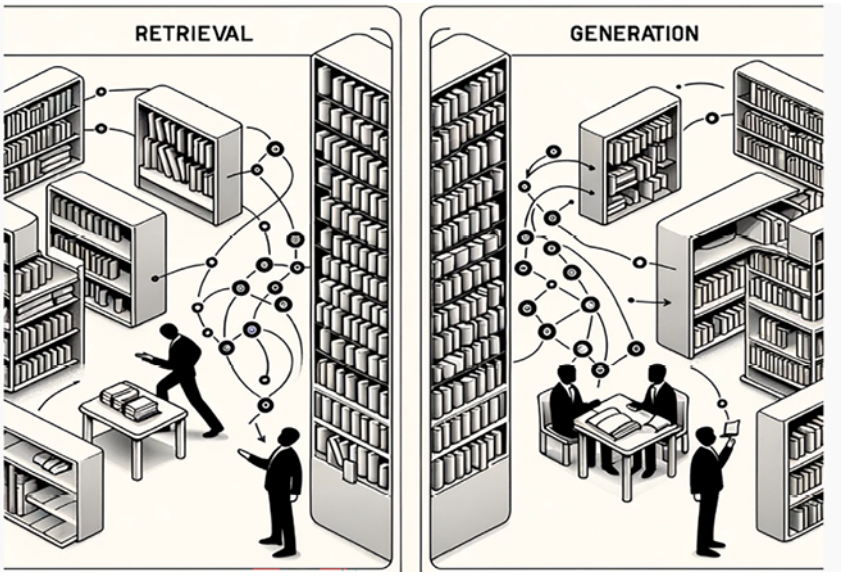

Figura 1: Funcionamento geral de uma estrutura RAG

Imagine um estudante em uma biblioteca, com a tarefa de escrever uma dissertação sobre RAG. Assim como o ChatGPT ou outros copilotos de IA, o estudante sabe ler e escrever. Como qualquer LLM, ele é treinado para compreender informações avançadas, resumir e criar conteúdo. No entanto, como qualquer IA avançada, seja do Hugging Face, Vertex AI ou OpenAI, há muitas informações que este estudante ainda desconhece.

Na fase de recuperação, ele busca por livros sobre o tema necessário (lado esquerdo da figura 1) na biblioteca. Em seguida, ele retorna ao seu lugar, realiza a tarefa de recuperação sozinho ou com a ajuda de um colega, extraindo as informações relevantes dos livros adquiridos. Na fase de geração (lado direito da figura 1), o estudante começa a escrever sua dissertação. Assim, funciona como um agente humano guiado por RAG, de maneira semelhante a um framework de IA generativa baseado em RAG.

Enquanto escreve o seu ensaio sobre RAG, o estudante encontra tópicos difíceis com os quais não tem tempo para consultar todas as informações disponíveis. Como um agente humano generativo, ele fica travado, assim como um modelo de IA generativa. Ele até pode tentar escrever algo, mas, como a IA, não saberá se o conteúdo está correto até que alguém corrija a dissertação e lhe avalie de alguma

maneira.

Neste ponto, ele já atingiu seu limite e decide recorrer a um copiloto de IA generativa com RAG para obter respostas corretas. No entanto, existe uma variedade tão grande de modelos LLM e configurações RAG disponíveis que o estudante acaba ficando confuso. Antes de prosseguir, é necessário entender os recursos disponíveis e como o RAG está organizado.

2.3 Ecossistema RAG

A IA generativa baseada em RAG é um framework que pode ser implementado com diversas configurações, funcionando dentro de um ecossistema amplo (Figura 2). Independentemente da quantidade de frameworks de recuperação e geração disponíveis, tudo se resume a quatro domínios principais e suas respectivas questões:

- Dados: De onde vêm os dados? São confiáveis e suficientes? Há questões de direitos autorais, privacidade ou segurança?
- Armazenamento: Como os dados serão armazenados antes ou depois do processamento? Qual será o volume armazenado?
- Recuperação: Como os dados corretos serão recuperados para complementar o input do usuário? Qual tipo de framework RAG será mais adequado ao projeto?
- Geração: Qual modelo de IA generativa melhor se adapta ao framework RAG escolhido?

Esses domínios dependem do tipo de framework RAG utilizado. Antes de escolher, é essencial avaliar a proporção de conhecimento paramétrico e não paramétrico no ecossistema implementado. A Figura 1.3 ilustra os principais componentes do framework RAG, independentemente do tipo implementado.

- Retriever (D): Responsável pela coleta, processamento, armazenamento e recuperação de dados.

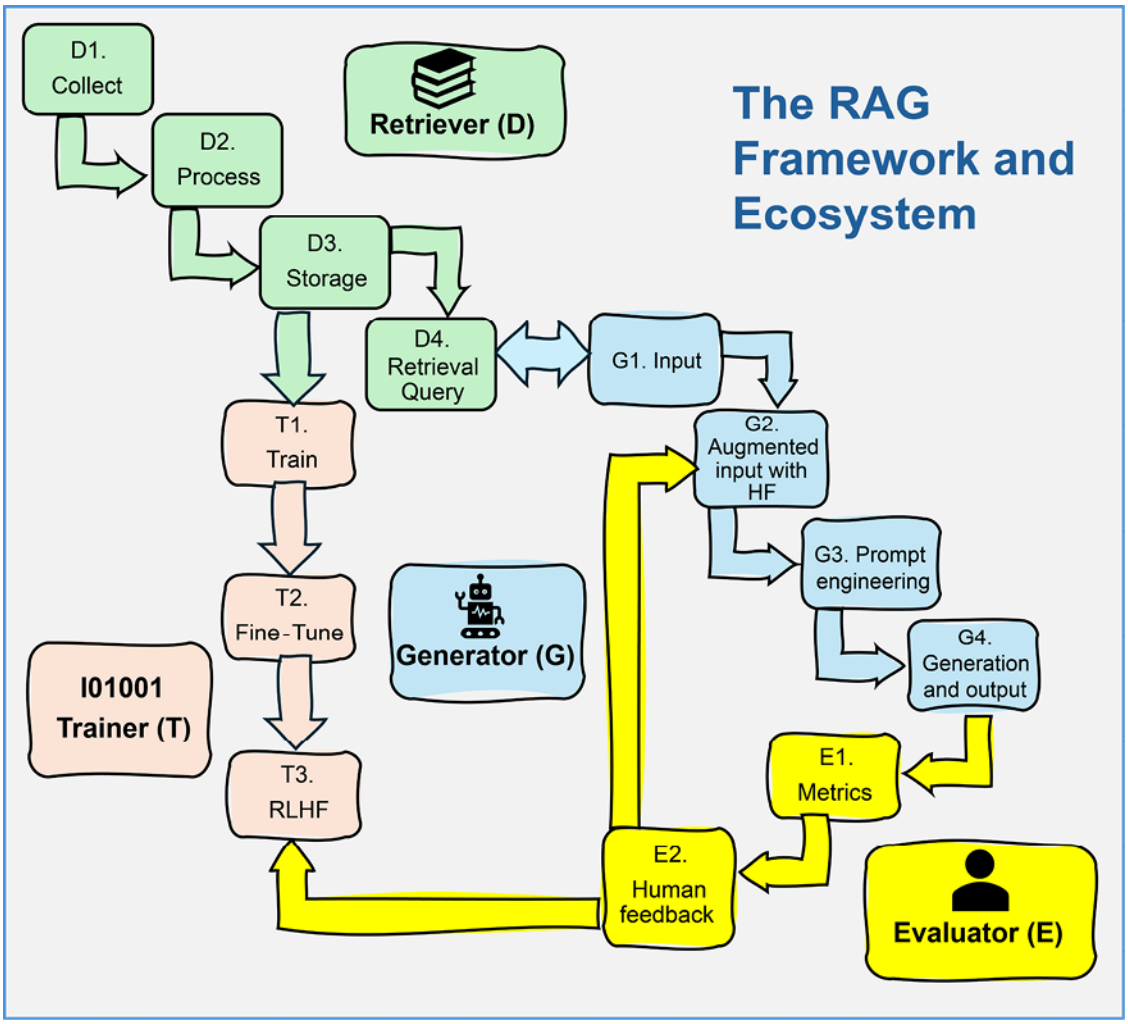

Figura 2: Arquitetura da estrutura RAG

- Generator (G): Cuida da complementação do input, engenharia de prompts e geração de respostas.
- Evaluator (E): Avalia o desempenho usando métricas matemáticas, feedback humano e outras formas de validação.
- Trainer (T): Gerencia o modelo pré-treinado inicial e sua posterior fine-tuning.

Esses quatro componentes dependem de seus respectivos ecossistemas, formando o pipeline de IA generativa baseada em RAG. Nas seções a seguir, usaremos as siglas D, G, E e T para representar respectivamente Retriever, Generator, Evaluator e Trainer. Começando pelo Retriever (D).

2.3.1 Retriever (D)

O componente retriever de um ecossistema RAG coleta, processa, armazena e recupera dados. O ponto de partida de um ecossistema RAG é, portanto, um processo de ingestão de dados, cujo primeiro passo é a coleta de dados.

1. Coleta de Dados (D1):

   Atualmente dados são extremamente diversos, podendo ser textos, arquivos de mídia (como músicas ou vídeos em mp4) ou arquivos estruturados e não estruturados (PDFs, JSONs e páginas web). Além disso, grande parte desses dados é não estruturada e pode ser encontrada de maneiras imprevisíveis e complexas. Felizmente, várias plataformas, como Pinecone, OpenAI, Chroma e Activeloop, oferecem ferramentas prontas para processar e armazenar essa vasta quantidade de dados.

2. Processamento de Dados (D2):

   Na fase de coleta de dados (D1) no processamento de dados multimodais, diferentes tipos de dados, como texto, imagens e vídeos, podem ser extraídos de websites utilizando técnicas de web scraping ou outras fontes de informação. Esses objetos de dados são então transformados para criar representações uniformes. Alguns exemplos dessas transformações incluem: chunking, embedding e indexação. Essas técnicas serão discutidas mais adiante.

3. Armazenamento de Dados (D3):

   Neste estágio do pipeline, já coletamos e começamos a processar uma grande quantidade de dados diversos. Mas para fazermos com que esses dados para sejam úteis, vamos fazer uso de vetor stores (armazenamento de vetores), como Elastic Search. Ele não apenas armazena os dados, mas os convertem em entidades matemáticas, representadas como vetores, permitindo realizar cálculos poderosos. Esses sistemas também utilizam técnicas de indexação e outras abordagens para garantir acesso rápido e eficiente aos dados. Em vez de manter os dados em arquivos estáticos, transformamos tudo em um sistema dinâmico e pesquisável, pronto para alimentar chatbots, motores de busca e outras aplicações.

4. Consulta de Recuperação (D4):

   O processo de recuperação é acionado pelo input do usuário ou input automatizado (G1). Para recuperar dados rapidamente, carregamos os dados nos vetor stores e datasets após transformá-los para um formato adequado. Em seguida, utilizamos uma combinação de pesquisas por palavras-chave, embeddings inteligentes e indexação para recuperar os dados de forma eficiente. A similaridade cosseno, por exemplo, encontra itens que estão intimamente relacionados, garantindo que os resultados da busca não sejam apenas rápidos, mas também altamente relevantes. Após a recuperação dos dados, o próximo passo é aumentar o input, ou seja, adicionar as informações recuperadas para enriquecer a resposta gerada.

2.3.2 Generator (G)

No ecossistema RAG, as linhas entre a entrada e a recuperação não são tão nítidas, como mostrado na figura 2, que representa o framework e ecossistema RAG. O input do usuário (G1), seja automatizado ou humano, interage com a consulta de recuperação (D4) para complementar o input antes de enviá-lo ao modelo generativo. O fluxo gerativo começa com o input, que é aprimorado com dados recuperados antes de ser processado pelo modelo de IA generativa.

1. Entrada (G1):

   O input pode ser uma série de tarefas automatizadas (como o processamento de e-mails, por exemplo) ou prompts humanos por meio de uma Interface de Usuário (UI). Essa flexibilidade permite integrar a IA de forma fluida em diversos ambientes profissionais, aprimorando a produtividade em diferentes setores.

2. Entrada Aumentada com Feedback Humano (G2):

   O feedback humano (HF, de Human Feedback) pode ser adicionado ao input, conforme descrito na seção Feedback Humano (E2), sob o componente Evaluator (E). O feedback humano torna o ecossistema RAG consideravelmente mais adaptável, permitindo total controle sobre a recuperação de dados e os inputs para a IA generativa. Na seção Building Hybrid Adaptive RAG in Python

deste capítulo, construiremos o input aumentado com feedback humano, melhorando a precisão e a relevância das respostas geradas.

3. Engenharia de Prompts (G3):
   Tanto o retriever (D) quanto o generator (G) dependem fortemente da engenharia de prompts para preparar a mensagem padrão e aumentada que o modelo de IA generativa deverá processar. A engenharia de prompts combina a saída do retriever com o input do usuário, garantindo que o modelo receba uma entrada bem estruturada e relevante para gerar a resposta desejada.

4. Geração e Saída (G4):
   A escolha de um modelo de IA generativa depende dos objetivos do projeto. Modelos como Llama, Gemini, GPT e outros podem atender a diferentes requisitos. No entanto, o prompt precisa estar alinhado com as especificações de cada modelo.

2.3.3 Evaluator (E)

Frequentemente, dependemos de métricas matemáticas para avaliar o desempenho de um modelo de IA generativa. No entanto, essas métricas fornecem apenas uma parte do todo. É importante lembrar que o teste final da eficácia de uma IA depende da avaliação humana, que garante uma compreensão mais completa da qualidade e relevância dos resultados gerados.

1. Métricas (E1):
   Um modelo não pode ser avaliado sem métricas matemáticas, como a similaridade cosseno, assim como em qualquer sistema de IA. Essas métricas garantem que os dados recuperados sejam relevantes e precisos. Ao quantificar as relações e a relevância dos pontos de dados, elas fornecem uma base sólida para avaliar o desempenho e a confiabilidade do modelo.

2. Feedback Humano (E2):
   Em um sistema de IA generativa, seja ele baseado em RAG ou não, e independentemente de as métricas matemáticas parecerem suficientes, o feedback humano é essencial. A avaliação humana é o fator decisivo que determina se

um sistema projetado para usuários humanos será aceito ou rejeitado, elogiado ou criticado. O RAG adaptativo introduz o feedback humano, pragmático e da vida real, o que melhora o ecossistema de IA generativa baseado em RAG.

2.3.4 Trainer (T)

Um modelo de IA generativa padrão é pré-treinado com uma grande quantidade de dados de uso geral. Em seguida, podemos ajustar (T2) o modelo com dados específicos de um domínio.

3 Modelagem

Com os conceitos de um ecossistema RAG em mente, é possível explicar de forma objetiva a modelagem da solução proposta para este trabalho. Antes de abordar as tecnologias específicas, tema reservado para um capítulo posterior, será apresentada uma visão abstrata da modelagem da solução.

De maneira objetiva, o projeto pode ser descrito como um sistema composto por três artefatos, cada um responsável por uma parte do ecossistema proposto. Esses artefatos implementam sistemas ou serviços que se comunicam entre si ao longo da solução, podendo implementar uma LLM, uma API ou uma instância de banco vetorial. Apesar de estarem conectados, cada artefato é independente e funciona de forma autônoma.

Juntos, esses três artefatos formam a solução proposta. Proporcionando uma abordagem modular e integrada para alcançar os objetivos propostos neste trabalho. Dentre os artefatos temos respectivamente um pipeline de processamento, uma interface de programação de aplicações (API) e uma interface gráfica de usuário.

3.0.1 Pipeline de Processamento

Artefato responsável por gerenciar fluxos de análise e transformação de documentos para a solução. Neste contexto, ele desempenha a função de lidar com bibliotecas de documentos, analisando cada documento e garantindo que cada um seja processado corretamente. Esse processo envolve etapas como leitura, análise e transformação do conteúdo dos documentos, com o objetivo de prepará-los para demais etapas do processo.

Ao término do processamento de cada documento na biblioteca, os dados processados são inseridos em um banco vetorial. Assim, o pipeline converte os dados para um formatos propícios para utilização no ecossistema em questão. Essa abordagem garante escalabilidade e precisão no gerenciamento e utilização dos dados processados.

3.0.2 Interface de Programação de Aplicações (API)

Artefato central no ecossistema, sendo responsável por mediar a comunicação entre os diferentes componentes do sistema. Sua principal função é receber e processar as requisições feitas pelos usuário na GUI, garantindo que as operações sejam realizadas de forma eficiente e segura.

Além disso, a API desempenha um papel crítico na segurança do sistema ao restringir o acesso direto ao banco vetorial por parte dos outros artefatos. Essa camada de abstração impede interações não autorizadas ou inadequadas com o banco de dados, garantindo que apenas as operações de usuários autenticados sejam realizadas. Com isso, a API promove uma integração controlada e otimizada entre os artefatos, além de assegurar a integridade dos dados e a consistência do sistema como um todo.

3.0.3 Interface Gráfica de Usuário (GUI)

Artefato responsável por receber e guiar o usuário durante suas interações com o sistema, assim como também é responsável em garantir a comunicação com o artefato da API.

Artefato dedicado a mediar a interação entre o sistema e usuário de forma intuitiva. Sua principal função é receber as entradas do usuário e orientá-lo durante a navegação pelo sistema, oferecendo elementos visuais para simplificar o uso das funcionalidades disponíveis.

Além de interagir diretamente com o usuário, a GUI desempenha um papel importante na comunicação com a API, garantindo que as solicitações realizadas pelo usuário sejam traduzidas em requisições apropriadas para o sistema. Essa integração possibilita que somente ações como consultas de dados sejam realizadas pelo usuário, de forma transparente e sincronizada com o restante dos artefatos.

3.1 Modelo Conceitual

Analisar uma biblioteca de documentos pode ser complexo, pois cada biblioteca pode comportar diversos tipos de documentos, em formatos distintos. Este projeto foi construído visando a análise de qualquer biblioteca de documentos, ou seja, o processamento dos dados da biblioteca será realizado mesmo que ela tenha arquivos em formatos diferentes. O projeto foi contruido com base nos três artefatos previamente explicados e funciona de acordo com a seguinte visão arquitetural:

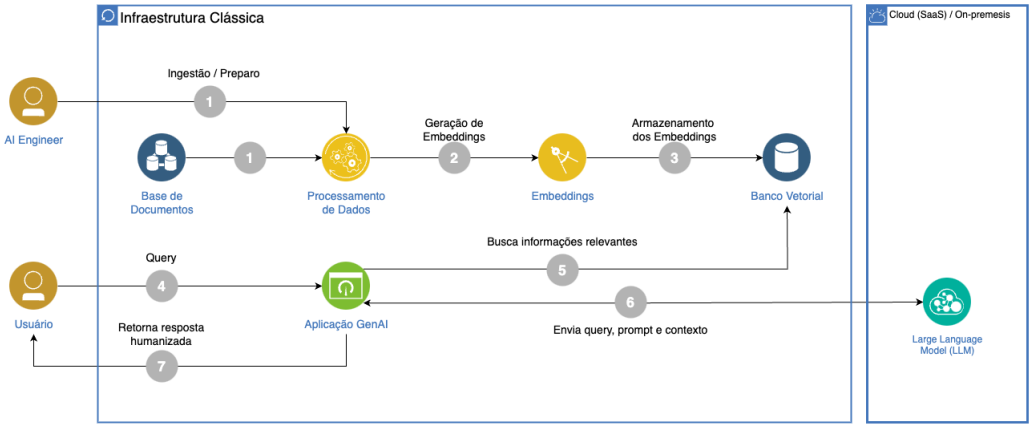

Figura 3: Visão arquitetural da solução RAG proposta para o trabalho

3.1.1 Ingestão e Preparo

Etapa onde a biblioteca é analisada, seus documentos são processados e transformados de maneira sequencial para obtermos um determinado modelo com as informações necessárias de cada documento. Esta etapa é dividida em duas partes, uma de ingestão e outra de preparo.

- Ingestão: Nesta parte os documentos são lidos e sua extensão é análisada, determinando a forma como o documento terá seus dados extraídos. Uma vez analisados, o documento é divididoo em chunks, um bloco de texto com os dados do documento.
- Preparo: Para cada chunk obtido na ingestão, é instanciado um modelo contendo os dados gerais do documento, assim como os dados do próprio chunk.

Este modelo, em específico, visa uma inserção no banco vetorial que será feita na etapa de armazenamento dos embeddings.

3.1.2 Geração de Embeddings

Antes de armazenar os modelos obtidos na etapa anterior, a informação de cada chunk precisa estar em um formato específico para o banco vetorial, este formato é obtido quando geramos os embeddings, que neste contexto, são uma representação numérica de baixa dimensão dos dados de um chunk expecífico.

3.1.3 Armazenamento dos Embeddings

Uma vez gerados, os embeddings de cada chunk estão presentes no modelo previamente adquirido na etapa de Ingestão e Preparo. Nesta etapa ocorre o armazenamento dos modelos na instância do banco vetorial do projeto. O armazenamento é feito de forma sequencial, porém em batches ou grupos com tamanho baseado na quantidade de chunks obtidos. Isso ocorre pois reduz o tempo da etapa de inserção ao banco, tendo em vista que uma vez que um documento é particionado em chunks, dependendo do tamanho do documento, podemos obter mais de 100 chunks, ou seja, 100 inserções.

3.1.4 Query

Aqui ocorre a interação do usuário com o sistema, nesta etapa o usuário digita uma pergunta dentro do espaço de input ou entrada da interface gráfica, que neste contexto é composta por um chat.

3.1.5 Busca por informações relevantes

No momento em que o usuário realiza a query, o sistema realiza uma busca por contexto ou informações relevantes como a biblioteca que o usuário está consultando (caso hajam múltiplas bibliotecas), uma determinada data, autor ou demais filtros que o usuário possa vir a fazer. Com as informações relevantes analisadas, é gerado um objeto de contexto que contém todas as informações relevantes

3.1.6 Envio de Query, Prompt e Contexto

Com o contexto e a query em mãos, o sistema prossegue com o fluxo principal e envia uma solicitação a API contendo a query que o usuário realizou, o prompt a ser seguido pela LLM e o contexto.

3.1.7 Retorno de Resposta Humanizada

Ao final da requisição feita pelo usuário, a API responde com uma lista contendo os objetos presentes no banco vetorial cujo embedding contém informações relevantes para a query do usuário. Esta lista contém os modelos presentes no banco vetorial, definidos na etapa de Preparação, e a partir dessa resposta passamos a lista para uma LLM que irá analisar primeiramente um prompt focado em entregar respostas humanizadas para o usuário. Neste prompt, há todo um cuidado para que qualquer desenvolvedor que deseje implementar este projeto possa utilizar do prompt engineering a fim de realizar as afinações necessárias para seu próprio contexto.

3.2 Tecnologias

3.2.1 Python e Ambientes Virtuais

O Python 3.11 é uma das versões mais recentes da linguagem de programação Python, lançada com melhorias significativas em desempenho, novas funcionalidades e correções de bugs. Uma das principais inovações dessa versão é a otimização do tempo de execução, com um aumento significativo na velocidade em relação às versões anteriores, devido a melhorias no interpretador e na execução de código. Além disso, a versão 3.11 introduziu aprimoramentos na sintaxe, como a simplificação de anotações de tipos e novos recursos que facilitam o desenvolvimento de aplicativos mais robustos e eficientes. A linguagem continua a ser uma das mais populares, especialmente em áreas como desenvolvimento web, ciência de dados e inteligência artificial.

Ambientes virtuais são espaços isolados onde é possível instalar e gerenciar dependências específicas de um projeto, sem afetar o sistema global ou outros pro-

jetos. Em Python, esses ambientes são criados com o módulo venv, permitindo que cada projeto tenha suas próprias versões de pacotes e bibliotecas. Isso evita conflitos entre dependências e facilita o gerenciamento de diferentes versões de pacotes para cada projeto.

As principais vantagens dos ambientes virtuais incluem a capacidade de isolar dependências, garantindo que cada projeto utilize a versão correta de pacotes, a facilidade de replicar e compartilhar ambientes de desenvolvimento e a redução de problemas de compatibilidade entre pacotes ou versões do Python. Eles oferecem um controle mais preciso sobre o ambiente de execução, tornando o desenvolvimento mais seguro e eficiente.

3.2.2 Docling

Uma biblioteca Python projetada para facilitar a extração e organização de documentos. Ela permite que desenvolvedores processem automaticamente documentos de diferentes formatos, como PDFs, textos e arquivos de Word, para extrair informações estruturadas e relevantes. A biblioteca é útil para automatizar tarefas de leitura e processamento de grandes volumes de dados, organizando-os de forma a facilitar a análise posterior.

A principal vantagem do Docling é a sua capacidade de simplificar o processo de extração de dados, tornando-o mais rápido e eficiente, sem a necessidade de escrever código complexo para lidar com diferentes tipos de documentos. Isso a torna uma ferramenta valiosa para projetos que requerem análise de texto ou a criação de bancos de dados a partir de documentos não estruturados.

3.2.3 Llama Index

Uma biblioteca Python que oferece ferramentas para criar índices eficientes, que podem ser usados em conjunto com modelos de linguagem, como o GPT, para realizar tarefas de busca e processamento de texto com alta performance.

A principal vantagem do LlamaIndex é sua capacidade de organizar dados complexos de forma que seja fácil e rápido realizar consultas sobre eles, integrando-

-se bem com outras ferramentas de IA e processamento de linguagem natural. Isso a torna útil para criar soluções que necessitam de pesquisa e análise de grandes quantidades de informações, como em assistentes virtuais, sistemas de recomendação e análise de documentos.

3.2.4 Elastic Search

O ElasticSearch é um serviço de banco de dados baseado em busca de texto completo e análise de grandes volumes de dados. Ele é amplamente utilizado para armazenar, pesquisar e analisar dados de forma rápida e escalável. Baseado no motor de busca Apache Lucene, o ElasticSearch permite que dados sejam indexados e consultados com alta performance, suportando operações como pesquisa por palavras-chave, agregações e análises em tempo real.

Em um banco vetorial, como o ElasticSearch, os dados são representados como vetores, o que permite consultas de similaridade, ou seja, encontrar documentos que sejam semanticamente semelhantes aos dados consultados. Isso é especialmente útil em casos de busca por texto, recomendação de conteúdo ou sistemas de recuperação de informações avançados. O ElasticSearch organiza e indexa esses vetores para proporcionar consultas eficientes e rápidas, sendo um componente essencial em sistemas que exigem grandes volumes de dados e análises complexas.

3.2.5 Ollama

A LLM do Ollama é uma ferramenta de inteligência artificial focada em fornecer modelos de linguagem avançados para tarefas como processamento de texto, geração de conteúdo, tradução, resumo e respostas a perguntas. Desenvolvido para ser altamente eficiente e acessível, o Ollama oferece modelos de linguagem que são otimizados para uma variedade de aplicações, desde automação de processos até assistência personalizada em diferentes domínios.

A principal vantagem da LLM do Ollama é sua facilidade de integração em soluções de software, além de seu desempenho rápido e preciso. A plataforma é projetada para ser simples de usar, com uma interface intuitiva e suporte a diferentes

linguagens de programação, tornando a tecnologia de IA mais acessível para desenvolvedores e empresas. Com um foco em flexibilidade e escalabilidade, o Ollama facilita a implementação de modelos de linguagem em larga escala sem comprometer a eficiência.

3.2.6 Fast API

Um framework moderno e de alto desempenho para construção de APIs em Python. Ele é baseado em padrões como Python type hints e pydantic, permitindo que desenvolvedores escrevam código de forma rápida e eficiente, com uma validação de dados robusta. O FastAPI se destaca pela sua velocidade, sendo capaz de processar requisições de maneira extremamente rápida, o que o torna ideal para aplicações que exigem alta performance.

As principais vantagens do FastAPI incluem sua facilidade de uso, a validação automática de dados, a documentação interativa gerada com o Swagger UI, e sua performance comparável a frameworks como Node.js e Go. Além disso, o FastAPI aproveita os recursos de tipagem do Python, o que resulta em um código mais claro e com menos erros, além de ser altamente escalável e fácil de integrar com outras ferramentas.

3.2.7 Streamlit

Uma biblioteca em Python usada para criar interfaces web interativas de forma rápida e simples, sem a necessidade de conhecimentos avançados em desenvolvimento front-end. Ele permite que desenvolvedores criem dashboards, visualizações de dados e aplicativos interativos com poucos comandos, utilizando apenas Python. O Streamlit é ideal para protótipos rápidos e aplicativos que precisam integrar visualizações com análises de dados.

As principais vantagens do Streamlit incluem sua simplicidade de uso, já que permite criar interfaces com poucas linhas de código, e sua integração direta com bibliotecas como Pandas, Matplotlib e Plotly, facilitando a visualização e análise de dados. Além disso, ele permite a criação de aplicativos interativos com atualizações

em tempo real, sem a necessidade de configurar complexos sistemas de front-end. Isso torna o Streamlit uma opção excelente para cientistas de dados e desenvolvedores que desejam criar interfaces intuitivas sem precisar aprender HTML, CSS ou JavaScript.

4 Solução Desenvolvida

Com uma visão geral de como funciona um ecossistema RAG e quais artefatos são necessários para implementá-lo, prosseguiremos para o desenvolvimento da solução deste projeto. Levando em consideração as tecnologias apresentadas anteriormente, os próximos tópicos irão abordar o desenvolvimento concreto da solução buscada para este projeto.

Para a implementação da solução descrita, a biblioteca utilizada foi de todos os trabalhos de conclusão de curso do ano de 2023 publicados pela secretaria do Bacharelado em Sistemas de Informação da Universidade Federal do Estado do Rio de Janeiro, UNIRIO.

4.1 Pipeline de Ingestão

Tendo em vista que uma biblioteca de documentos contém diversos documentos e cada documento possui sua própria extensão, é necessário desenvolver uma solução capaz de processar diversos tipos de documentos, podendo atender diversas bibliotecas distintas.

Este processamento foi implementado seguindo um Design Pattern, mais especificamente o factory design pattern. O factory design pattern implementa uma interface para a criação de objetos que herdam de uma superclasse, permitindo que as classes filhas possam alterar o tipo de objeto criado. Neste contexto, foi implementada uma fábrica de extratores, todo extrator possui uma instancia dos clientes provedores de serviços do LlamaIndex e Docling, assim como variáveis pré definidas e os métodos de extração.

No entanto, cada extrator implementa o método de extração a sua própria maneira, assim é possível implementar a extração de um arquivo .pdf ou .docx mudando somente a implementação de um único método. Isso contribui para uma melhor manutenção do código e atribui a responsabilidade de extração para um único extrator específico de cada documento.

Após extraídos os dados dos documentos, é necessário separar os dados re-

levantes de cada documento e particionar o mesmo em chunks. Para isso, antes de particionar os documentos é realizada uma busca por dados relevantes do documento, neste caso, por se tratar de uma biblioteca de TCCs os dados relevantes são o resumo (abstract), nome, biblioteca a qual ele pertence, caminho (filepath), extensão e mídias do arquivo.

As mídias são separadas em dois tipos, imagens e tabelas, durante a primeira leitura do documento a biblioteca Docling identifica as mídias dos documentos e as salva em um diretório específico dentro do código do artefato do pipeline, posteriormente este diretório será injetado no código do artefato da API.

As informações relevantes devem então ser extraídas no começo da extração do documento, uma vez que a busca pelo resumo do trabalho pode tomar tempo de processamento, seria imprudente realizar uma busca após a partição dos documentos. Uma vez extraídas, é gerado um breve resumo do abstract por uma LLM, neste caso foi utilizado o Ollama com um prompt que enfatiza o resumo de no máximo 2 paragrafos contendo 250 caracteres. Essas informações nos geram o seguinte modelo:

```
{
    "abstract": "resumo extraído do documento",
    "summary": "um breve resumo do abstract",
    "filename": "nome do arquivo sem extensão",
    "database": "biblioteca a qual o arquivo
       pertence",
    "filepath": "caminho onde o arquivo se encontra
       ",
    "medias": {
        "images": "lista com o caminho das imagens
           achadas no arquivo",
        "tables": "lista com o caminhos das tabelas
            achadas no arquivo"
    },
    "extension": "extensão do arquivo"
}
```

Com as informações gerais acima, podemos dar continuidade a partição dos documentos em chunks. Neste projeto, cada chunk possui 2048 tokens. Cada chunk gerado será lido e terá o embedding gerado para a inserção no banco vetorial, neste projeto o modelo de embedding é parametrizado mas optei por seguir esta implementação utilizando o modelo impira/layoutlm-document-qa pois é um modelo especifico para RAG e já treinado para melhor atender nosso caso de uso. Assim, obtemos o seguinte modelo para os chunks:

```
{
    "_id": "chave única do chunk",
    "chunk": "objeto imutável obtido durante
       partição",
    "sequence": "sequência do chunk em questão",
    "pages": "lista de páginas com a informação do
       chunk em questão",
    "tables": "lista de caminhos das tabelas
       achadas no arquivo",
    "figures": "lista de caminhos das imagens
       achadas no arquivo",
    "vector": "embeddings gerado para o chunk"
}
```

Este processo de leitura e processamento irá se repetir diversas vezes até que o último documento seja devidamente processado. Uma vez processados, os documentos estarão no formato de chunk seguindo o modelo acima e estarão prontos para inserção no banco vetorial. A inserção no banco irá selecionar cem chunks e realizar uma inserção em batch dos mesmos, devido ao tamanho da biblioteca em questão cem foi um número razoável de chunks, mas dependendo da biblioteca será necessário aumentar este número.

4.2 Interface de Programação de Aplicações

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

4.3 Interface Gráfica de Usuário

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie

non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

5 Conclusão

5.1 Considerações Finais

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

5.2 Limitações

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus.

Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

5.3 Trabalhos Futuros

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Referências