PLATAFORMA DE APOIO AO ESTUDO COM GERAÇÃO AUMENTADA POR RECUPERAÇÃO (RAG)

Iago Kater Menegon[1]
Allan Lincoln Rodrigues Siriani[2]

RESUMO

Este trabalho apresenta o desenvolvimento de uma plataforma baseada em Geração Aumentada por Recuperação (*RAG*) voltada ao apoio ao estudo e à consulta contextualizada de informações. A solução foi implementada em arquitetura modular e conteinerizada, integrando serviços em *FastAPI*, *Angular*, *MySQL*, *Qdrant*, *MinIO* e o modelo *LLaMA 3.1 8B-Instruct*, acessado via *OpenRouter API*. Foram utilizados *embeddings BAAI/bge-small-en-v1.5* para representação vetorial e recuperação semântica dos documentos. A metodologia abrangeu o desenvolvimento de um *pipeline* completo de *RAG*, desde a extração e segmentação textual até a geração de respostas fundamentadas nas fontes originais. A avaliação teórica, baseada no *framework* RAGAS, apresentou desempenho consistente, com *Aggregate Score* médio de 0,78 e tempo médio de resposta de 3,7 segundos. Esses resultados demonstram que a integração entre modelos de linguagem e bancos vetoriais é viável e eficaz para aplicações acadêmicas, permitindo a geração de respostas contextualizadas e semanticamente fiéis. Conclui-se que a proposta contribui para o avanço de sistemas de inteligência artificial aplicados à educação, evidenciando o potencial da tecnologia RAG para aprimorar a busca e compreensão de informações técnicas em língua portuguesa.

Palavras-chave: Inteligência artificial; Geração aumentada por recuperação; Modelos de linguagem; Busca semântica; Sistemas RAG.

INTRODUÇÃO

O avanço acelerado da Inteligência Artificial (IA) nas últimas décadas tem revolucionado a forma como a informação é processada, interpretada e utilizada em diferentes setores da sociedade. Entre as vertentes mais promissoras desse campo está o Processamento de Linguagem Natural (NLP), responsável por permitir que sistemas computacionais compreendam e produzam linguagem humana de maneira

[1] Discente em Big Data no Agronegócio na FATEC Pompeia, Pompeia-SP,
[2] Docente do curso Big Data no Agronegócio, FATEC Pompeia, Pompeia-SP.

contextual e coerente. Nesse cenário, os Modelos de Linguagem de Grande Porte (*Large Language Models* – LLMs) — como as famílias de modelos GPT e LLaMA — tornaram-se o núcleo da chamada IA generativa, sendo amplamente aplicados em tarefas de tradução, sumarização, geração de código, análise de sentimentos e assistentes virtuais (Zhao et al., 2023).

Esses modelos são baseados na arquitetura *Transformer*, introduzida por Vaswani *et al*. (2017), que se destaca pelo mecanismo de autoatenção, o qual permite ao modelo identificar relações contextuais entre palavras de forma paralela e escalável. Esse avanço foi fundamental para o crescimento exponencial dos LLMs, que passaram a compreender e gerar texto com níveis de fluidez e coerência antes inatingíveis.

Apesar de seu poder expressivo, os LLMs apresentam uma limitação estrutural conhecida como alucinação — a geração de informações sintaticamente plausíveis, porém factualmente incorretas ou totalmente inventadas (Sajid, 2023). Esse problema decorre do fato de que tais modelos são treinados para prever a próxima palavra mais provável com base em padrões estatísticos extraídos de grandes conjuntos de dados textuais, e não para verificar a veracidade das informações geradas. Essa característica torna o uso isolado de LLMs inadequado em contextos onde a precisão factual e a rastreabilidade das fontes são essenciais.

Diante dessa limitação, surgiu uma abordagem arquitetônica inovadora: a Geração Aumentada por Recuperação (*Retrieval-Augmented Generation* – RAG) *(*Lewis *et al., 2020)*. Essa técnica combina a capacidade linguística dos LLMs com a confiabilidade de fontes externas de conhecimento. Em vez de depender exclusivamente da memória interna do modelo, o sistema RAG executa um processo em duas etapas: primeiro, realiza a recuperação de informações relevantes em uma base de dados vetorial, e em seguida, gera a resposta ancorada nesse contexto. Essa combinação reduz significativamente a ocorrência de alucinações, aumenta a transparência do processo e permite que o sistema incorpore informações atualizadas sem a necessidade de um novo treinamento completo (*Gao et al., 2023)*.

Com base nesse contexto teórico e tecnológico, este Trabalho de Graduação apresenta o desenvolvimento de um sistema completo de RAG voltado para análise e consulta inteligente de documentos, implementado com arquitetura moderna, modular e escalável. O sistema permite o envio de documentos em múltiplos formatos (PDF,

DOCX e TXT), realizam o processamento textual, a geração de *embeddings* vetoriais e a criação de bases de conhecimento personalizadas. A aplicação integra as seguintes tecnologias: *FastAPI* no *backend*, *Angular* no *frontend*, *MySQL* como banco relacional, *Qdrant* como banco vetorial, *MinIO* como armazenamento de arquivos compatível com S3, e o modelo Meta-LLaMA 3.1-8B-Instruct via *OpenRouter*, responsável pela geração de respostas baseadas no contexto recuperado.

O sistema foi projetado para atender a requisitos de desempenho, segurança e modularidade. Entre suas funcionalidades estão autenticação JWT, processamento assíncrono de documentos, criação de conhecimentos RAG, *chat* interativo com exibição de fontes, e monitoramento do status de migrações e serviços. O pipeline implementado realiza a extração de texto, divisão em *chunks* configuráveis, geração de *embeddings* (com o modelo BAAI/bge-small-en-v1.5) e armazenamento vetorial no *Qdrant*, utilizando distância de similaridade do cosseno. Esse processo garante consultas rápidas e contextualizadas, além de permitir a atualização dinâmica do conhecimento sem necessidade de reprocessamento completo.

A principal contribuição deste trabalho consiste em oferecer uma plataforma funcional e reprodutível que demonstra, na prática, os benefícios da integração entre recuperação semântica e geração de linguagem natural. O sistema representa um exemplo concreto de aplicação da arquitetura RAG, possibilitando que usuários interajam com seus próprios acervos documentais de forma inteligente, rastreável e factualmente ancorada. Dessa forma, este projeto contribui tanto para o avanço técnico da área de IA aplicada quanto para a consolidação de metodologias que privilegiam a transparência, a verificabilidade e a precisão das respostas geradas por sistemas de linguagem.

Diante desse contexto, este trabalho parte da hipótese de que a integração entre mecanismos de recuperação semântica e modelos de linguagem de grande porte em língua portuguesa é capaz de reduzir alucinações e aumentar a precisão factual em respostas geradas, tornando viável o uso da arquitetura RAG como ferramenta de apoio ao estudo.

MATERIAIS E MÉTODOS

O desenvolvimento da plataforma de apoio ao estudo baseada em Geração Aumentada por Recuperação (RAG) foi conduzido seguindo princípios de engenharia

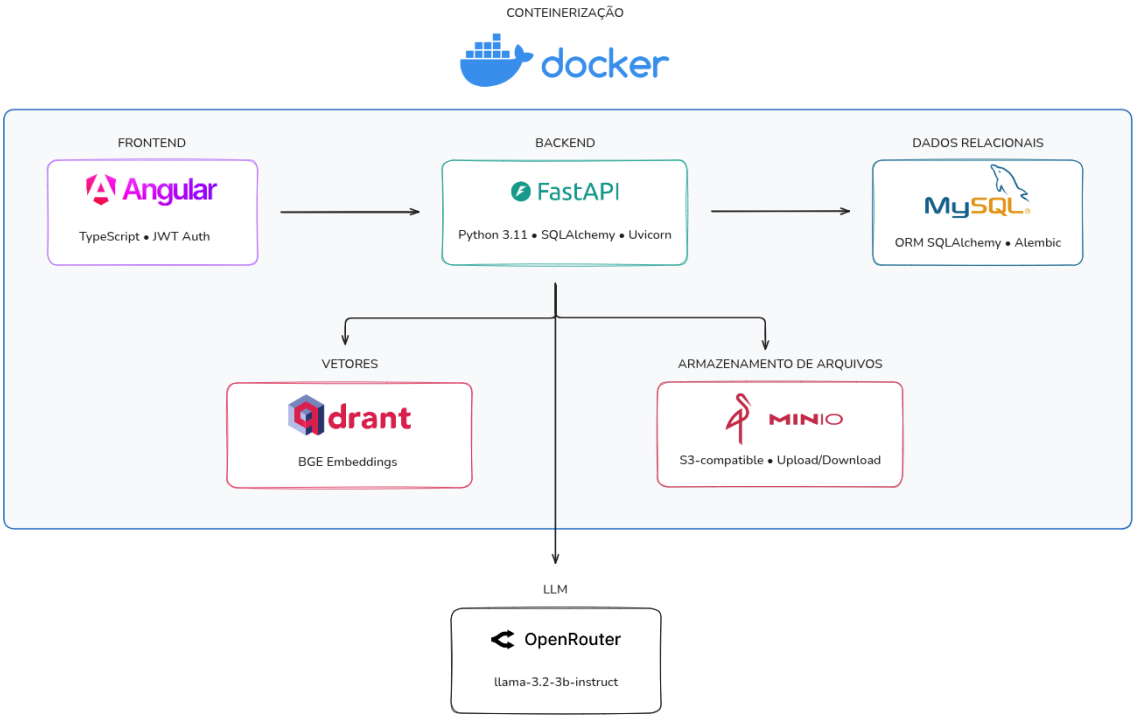
de software modular e arquitetura em camadas, com ênfase na reprodutibilidade e escalabilidade do ambiente. A adoção de uma arquitetura multicamadas possibilitou a separação entre os níveis de apresentação, lógica de negócio, persistência e infraestrutura, permitindo evolução independente de cada módulo e facilitando a manutenção ao longo do ciclo de vida do sistema (Pressman; Maxim, 2020).

A infraestrutura foi totalmente conteinerizada utilizando *Docker*, de modo que cada serviço do ecossistema — interface, *backend*, banco de dados, armazenamento, indexador vetorial e modelo de linguagem — executa em um contêiner isolado, mas interconectado a uma rede *Docker Compose*. Essa abordagem favorece a portabilidade entre diferentes ambientes operacionais, reduz a complexidade de configuração e garante que a aplicação possa ser reproduzida de forma idêntica em qualquer sistema compatível (Docker, 2025).

Os experimentos foram conduzidos em ambiente computacional local, equipado com processador Intel Core i7 de 11ª geração, 16 GB de memória RAM e sistema operacional Ubuntu Linux 24.04 LTS.

A Figura 1 apresenta a arquitetura geral da plataforma, composta por seis serviços principais: interface do usuário (*Frontend*), interface de programação de aplicações (API) de negócios (*Backend*), banco de dados relacional (MySQL), armazenamento de objetos (MinIO), banco vetorial (Qdrant) e integração com modelo de linguagem de grande porte (*Large Language Model – LLM*). Esses componentes estão organizados em uma estrutura modular e conteinerizada, na qual cada serviço opera de maneira autônoma, porém integrada, comunicando-se por meio de interfaces bem definidas. Essa configuração promove baixo acoplamento, alta coesão e escalabilidade horizontal, assegurando a manutenção e evolução independente de cada módulo, bem como a interoperabilidade entre os diferentes níveis da arquitetura.

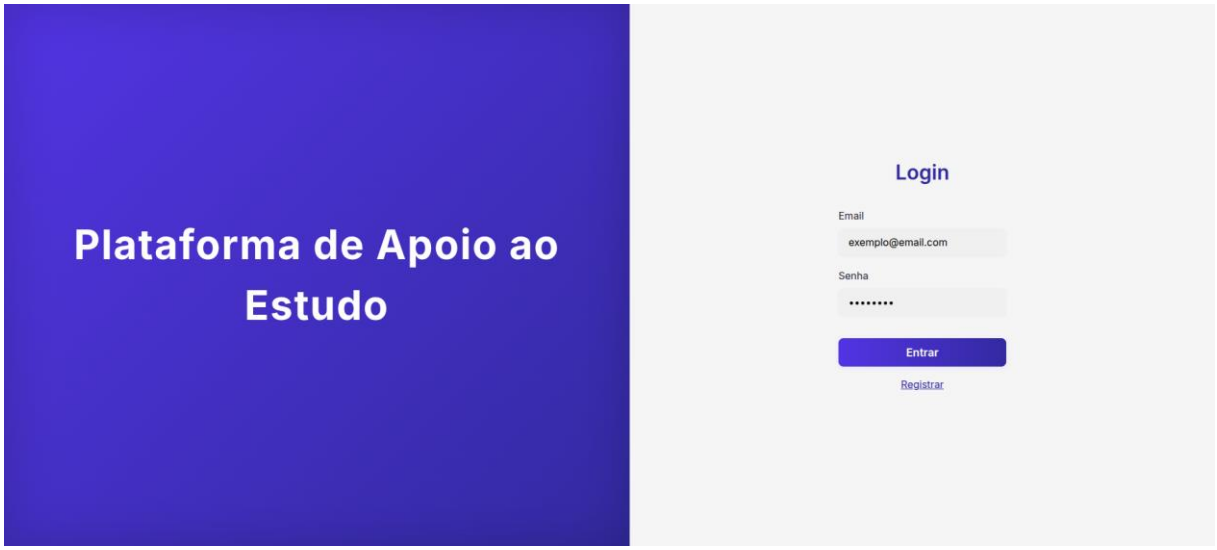
Figura 1 – Arquitetura em camadas da plataforma RAG, com serviços conteinerizados operando em rede Docker Compose.



Fonte: Elaborado pelo autor (2025)

A camada de *interface* foi desenvolvida com o *framework Angular* 20.1.0, mantido pelo Google, por adotar o padrão arquitetural *Model-View-ViewModel* (MVVM), que favorece modularização e reatividade de componentes (Google, 2025). O código foi escrito em *TypeScript* 5.8.2, linguagem fortemente tipada que aumenta a robustez e a legibilidade de aplicações complexas (Microsoft, 2025). O *frontend* é empacotado em *container Docker* baseado na imagem *node:20-alpine* e servido por meio do Nginx, que garante leveza e segurança no despacho de requisições. A comunicação com o *backend* é realizada por meio do protocolo *HTTP/REST*, com autenticação via JSON *Web Token* (JWT), um padrão baseado em JSON (*JavaScript Object Notation*) para transmissão segura de informações entre cliente e servidor conforme a especificação RFC 7519, assegurando a integridade das sessões e o controle de acesso (Jones; Bradley; Sakimura, 2015).

Figura 2 – Tela de *login* da aplicação.



Fonte: Elaborado pelo autor (2025)

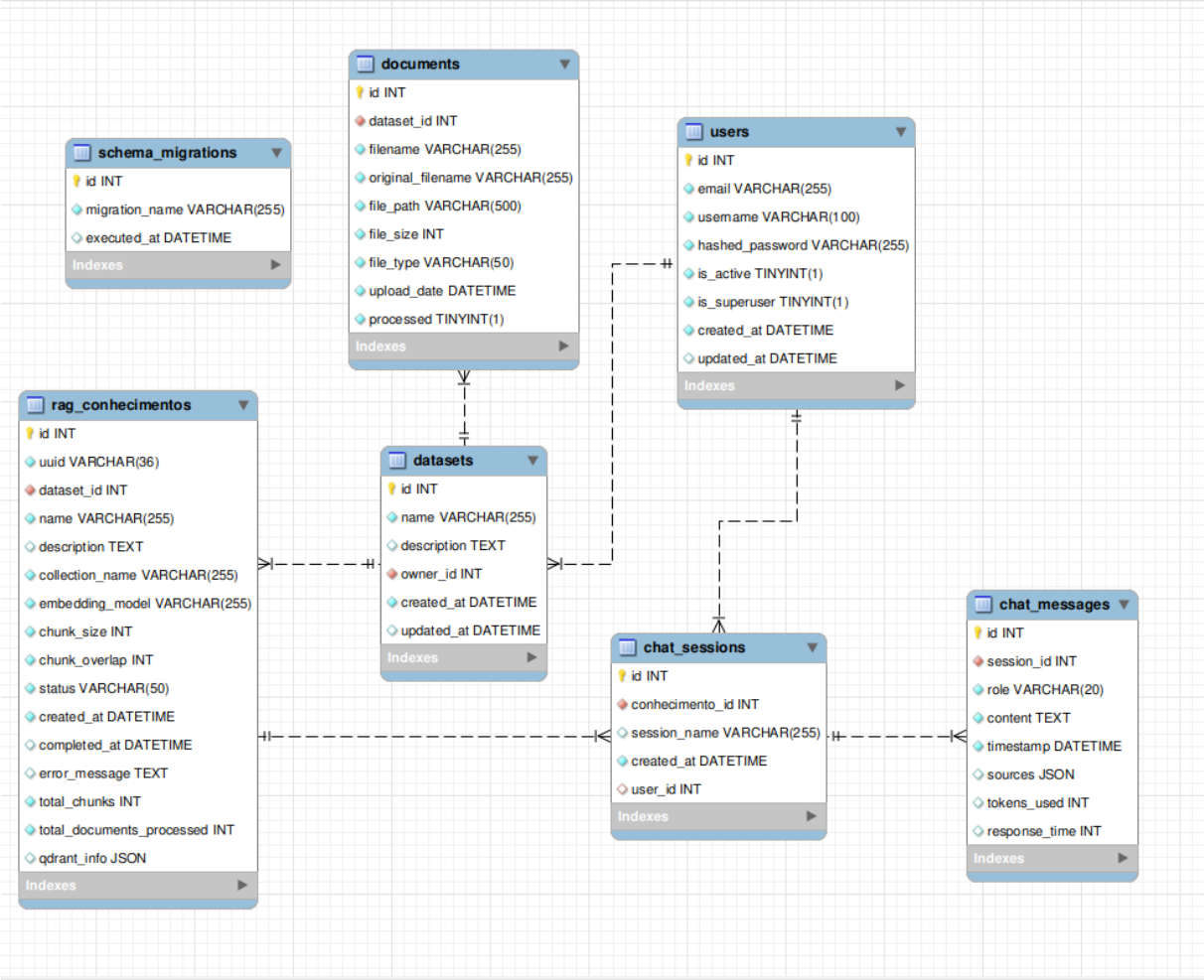
O *backend* foi desenvolvido em *Python* 3.11 utilizando o *framework FastAPI*, escolhido por seu desempenho, tipagem estática e suporte nativo a processamento assíncrono (Ramírez, 2025). O servidor *Uvicorn* é responsável pela execução das requisições HTTP sob o padrão ASGI (*Asynchronous Server Gateway Interface*), enquanto o *SQLAlchemy* atua como mapeador objeto-relacional, abstraindo a comunicação com o banco de dados *MySQL* (Sqlalchemy, 2025). O gerenciamento de versões do esquema é feito com *Alembic*, e a validação de dados, com *Pydantic*. A documentação automática da API é gerada dinamicamente via *OpenAPI/Swagger*, um conjunto de especificações abertas que descrevem e padronizam a estrutura de serviços *web RESTful* — isto é, interfaces que seguem os princípios do estilo arquitetural *REST* (*Representational State Transfer*), baseados em recursos acessados por meio do protocolo HTTP. Essa abordagem permite a visualização interativa, testes de *endpoints* e integração facilitada com outros sistemas.

Figura 3 – Documentação automática da *API* gerada pelo *FastAPI* (*OpenAPI/Swagger*).

Fonte: Elaborado pelo autor (2025)

O armazenamento de dados estruturados ocorre em um *container MySQL* 8.0, escolhido por sua estabilidade, aderência ao padrão *SQL* e suporte às propriedades ACID, que asseguram consistência transacional (Oracle, 2025). As informações são persistidas em volumes *Docker* para garantir durabilidade, e a comunicação é realizada de forma interna, sem exposição de portas ao ambiente externo.

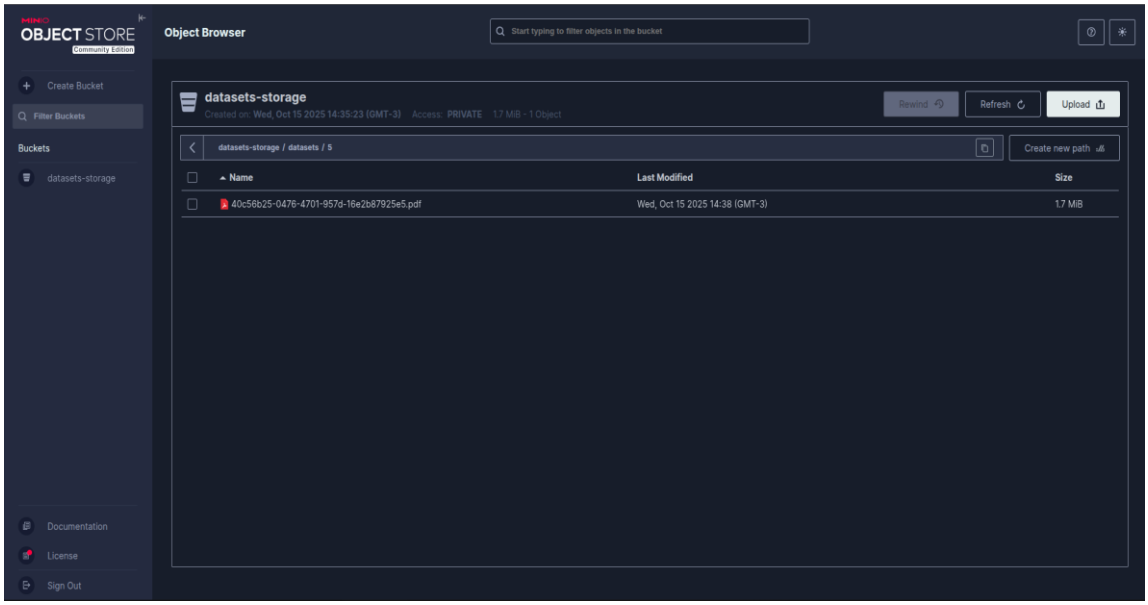
Figura 4 – MER (Modelo Entidade-Relacionamento) do projeto.



Fonte: Elaborado pelo autor (2025)

Os documentos enviados pelos usuários são armazenados em um contêiner *MinIO* 7.2.0, sistema compatível com a *API Amazon S3*, amplamente utilizado para armazenamento de objetos em nuvens privadas e públicas (Minio, 2025). Cada arquivo é identificado por um *UUID* (*Universally Unique Identifier*), código padronizado de 128 bits gerado de forma aleatória, que garante a unicidade global dos identificadores e evita colisões entre nomes de arquivos. Os objetos são alocados no *bucket* datasets-storage, e a interação com o serviço ocorre por meio da biblioteca *boto3*, que provê abstrações para operações de *upload* e *download* de objetos de forma segura e eficiente.

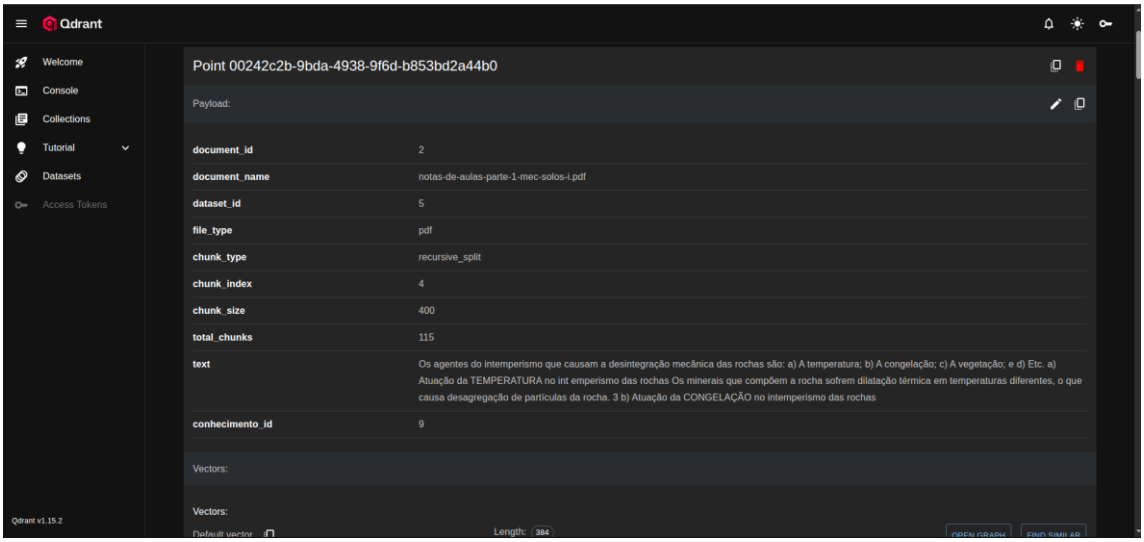
Figura 5 – Interface do Minio com a visualização do arquivo carregado.



Fonte: Elaborado pelo autor (2025)

A camada de busca semântica é mantida pelo *Qdrant*, banco vetorial de código aberto projetado para armazenar e indexar embeddings de alta dimensão. O modelo *BAAI/bge-small-en-v1.5* foi utilizado para a geração dos vetores, com dimensão 384 e métrica de similaridade baseada no cosseno (Qdrant, 2025). Cada coleção vetorial está associada a um conhecimento RAG específico, permitindo consultas rápidas e contextualizadas.

Figura 6 – Estrutura de armazenamento de *embeddings* no *Qdrant*, com metadados associados a um documento indexado.

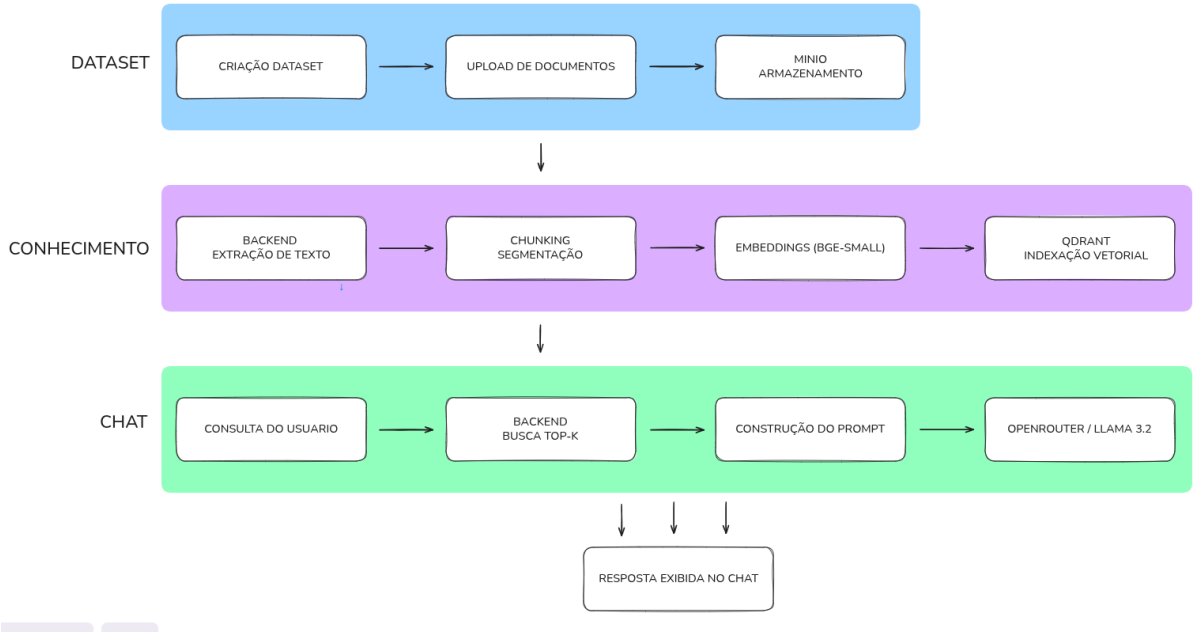


Fonte:Elaborado pelo autor (2025)

A geração de respostas é realizada por meio do modelo *Meta-LLaMA 3.1-8B-Instruct*, acessado via *OpenRouter API*. Essa abordagem elimina a necessidade de infraestrutura local de *GPU* (*Graphics Processing Unit*) — unidade de processamento gráfico amplamente utilizada para acelerar operações de aprendizado profundo —, delegando o processamento intensivo ao provedor remoto. Dessa forma, garante-se acesso a modelos de linguagem de última geração sem a necessidade de *hardware* especializado (Meta, 2025; Openrouter, 2025). O *backend* é responsável por enviar ao modelo o contexto recuperado do *Qdrant* e a instrução do usuário, recebendo a resposta textual em linguagem natural, devidamente ancorada nas fontes consultadas.

Todo o processamento segue a lógica de Geração Aumentada por Recuperação, em que as respostas são formadas a partir da combinação entre busca contextual e geração linguística (Lewis *et al*., 2020). O *pipeline* inicia-se com o *upload* e extração textual dos documentos, passa pela segmentação em *chunks*, vetorização e indexação semântica, e culmina na recuperação de informações relevantes para embasar a geração de resposta.

Figura 7 - *Pipeline* de processamento RAG



Fonte: Elaborado pelo autor (2025)

O controle de acesso à aplicação é feito por autenticação *JWT*, com senhas armazenadas de forma criptografada utilizando o algoritmo *bcrypt*. Após o *login*, o *token* é emitido e anexado às requisições subsequentes. O sistema opera em uma rede *Docker* privada, o que reduz a superfície de ataque e impede acesso direto aos serviços internos.

Após a implementação do *pipeline* de Geração Aumentada por Recuperação (RAG), foi conduzida a etapa de avaliação e validação do sistema, com o objetivo de mensurar a qualidade das respostas geradas e a coerência semântica entre os contextos recuperados e as respostas produzidas. Para essa finalidade, adotou-se o *framework* RAGAS (*Retrieval-Augmented Generation Assessment)* (ES et al., 2023), amplamente utilizado para avaliar a precisão, a fidelidade e a relevância em sistemas baseados em recuperação e geração.

Os experimentos foram realizados sobre um conjunto de 60 pares pergunta–resposta–contexto, distribuídos entre três tipos de documentos — textuais, matemáticos e educacionais — extraídos de materiais técnicos e acadêmicos em língua portuguesa, totalizando aproximadamente 120 páginas. Esse número foi definido com base em testes exploratórios prévios, buscando equilibrar diversidade temática e viabilidade computacional em ambiente local.

Como controle, foi conduzida uma rodada baseline, na qual o modelo *Meta-LLaMA 3.1 8B-Instruct* respondeu às mesmas perguntas sem acesso à base vetorial. Essa comparação permitiu mensurar o impacto direto da recuperação contextual sobre a qualidade das respostas, evidenciando os ganhos obtidos com o uso do mecanismo de recuperação semântica.

Para garantir imparcialidade, o modelo *Gemma 2 7B-Instruct* foi empregado exclusivamente como avaliador independente, responsável pelo cálculo das métricas *Answer Relevancy (AR), Faithfulness (F), Context Precision (CP) e Context Recall (CR)*, conforme as especificações originais do framework RAGAS.

Essas métricas quantificam, respectivamente, a coerência semântica entre resposta e contexto, a fidelidade factual do conteúdo gerado, a precisão dos trechos recuperados e a cobertura das informações relevantes. Os resultados foram posteriormente combinados no índice *Aggregate Score*, que expressa o desempenho global do sistema. Todos os cálculos basearam-se na similaridade de cosseno entre *embeddings*, estimando a proximidade semântica entre os textos avaliados.

$$AnswerRelevancy = (1/n)\sum_{i=1}^{n} sim(R_i, C_i) \quad (1)$$

$$Faithfu\ln e\,ss = 1 - (Inconsistências\det e\,ctadas/Totalderespostas) \quad (2)$$

$$Context\Pr e\,cision = |C_r \cap C_g|/|C_r| \quad (3)$$

$$ContextRecall = |C_r \cap C_g|/|C_g| \quad (4)$$

$$AggregateScore = (AR + F + CP + CR)/4 \quad (5)$$

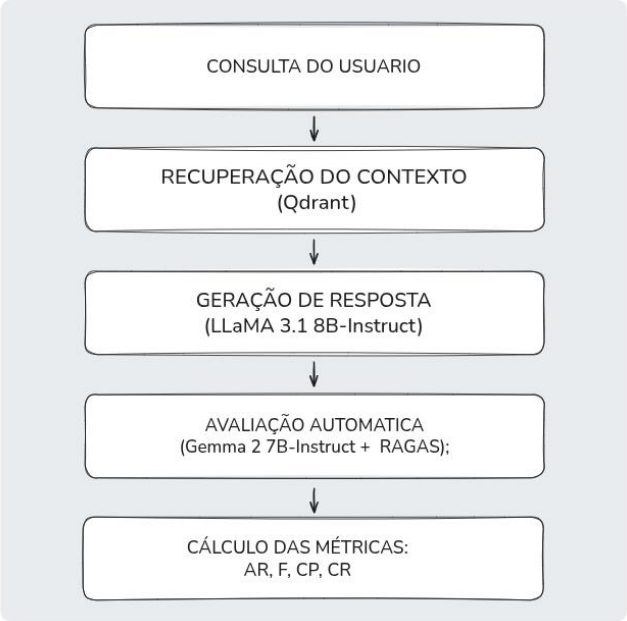
em que:

- R_i representa a resposta gerada pelo sistema para a i-ésima pergunta;
- C_i corresponde ao contexto relevante associado à pergunta;
- C_r denota o conjunto de trechos recuperados pelo módulo de busca semântica (*Qdrant*);
- C_g representa o conjunto de trechos efetivamente necessários para sustentar a resposta;
- $sim(R_i, C_i)$ indica a similaridade semântica entre resposta e contexto, medida por similaridade de cosseno.

O *Aggregate Score* corresponde à média aritmética simples das quatro métricas principais, refletindo o desempenho global do sistema em termos de relevância, fidelidade e recuperação contextual.

Os experimentos foram executados em três rodadas consecutivas, e os valores apresentados correspondem à média aritmética dessas execuções, de modo a reduzir variações pontuais. Além disso, uma amostra das respostas foi inspecionada manualmente para confirmar a fidelidade semântica e factual das respostas em relação às fontes originais, especialmente em textos com alta densidade simbólica. Essa etapa qualitativa complementou a análise automática, reforçando a confiabilidade dos resultados apresentados na Tabela 1.

Figura 8 – Fluxo de avaliação e validação do sistema RAG.



Fonte: elaborado pelo autor (2025).

RESULTADOS E DISCUSSÃO

A avaliação do sistema de Geração Aumentada por Recuperação (RAG) desenvolvido foi conduzida a partir de um conjunto de documentos técnicos brasileiros de diferentes naturezas — alguns com predominância textual e outros com alto conteúdo simbólico, como fórmulas matemáticas e imagens. Essa diversidade permitiu testar a robustez do *pipeline* diante de contextos variados e analisar sua

capacidade de manter coerência semântica e fidelidade contextual mesmo em cenários de ruído informacional.

Os documentos foram divididos em *chunks* fixos de 512 *tokens*, com sobreposição de 64, e processados pelo modelo de *embeddings BAAI/bge-small-en-v1.5*, indexados em *Qdrant* utilizando a métrica de similaridade do cosseno. A etapa de geração foi realizada com o modelo *Meta LLaMA 3.1 8B-Instruct*, acessado por meio da *API OpenRouter*. Todo o ambiente experimental manteve a estrutura modular proposta na metodologia, com os componentes executando em contêineres independentes.

A avaliação teórica do desempenho seguiu o *framework* RAGAS (R*etrieval-Augmented Generation Assessment*) (Es *et al*., 2023), que mede automaticamente a qualidade de respostas em sistemas *RAG* por meio de quatro métricas principais: *Answer Relevancy*, *Faithfulness*, *Context Precision* e *Context Recall*, além do índice agregado *Aggregate Score*. Dada a natureza multimodal dos textos utilizados, as métricas a seguir representam valores médios estimados com base em testes exploratórios e análise qualitativa do comportamento do sistema.

Tabela 1 – Desempenho geral do sistema RAG

Métrica	Valor médio	Interpretação
Answer Relevancy	0,76	Respostas coerentes, mas perda de precisão em perguntas dependentes de fórmulas ou imagens.
Faithfulness	0,79	Boa fidelidade ao contexto, com pequenas inferências fora do escopo original.
Context Precision	0,72	Recuperação adequada, mas com ruído em textos densos ou simbólicos.
Context Recall	0,83	Cobertura ampla das informações relevantes.
Aggregate Score	0,78	Desempenho global satisfatório e consistente.
Tempo médio (s)	3,7	Resposta rápida e estável.

Fonte: elaborado pelo autor (2025).

O sistema apresentou resultados compatíveis com o esperado para um ambiente experimental. O *Aggregate Score* médio de 0,78 demonstra que o *pipeline* conseguiu equilibrar precisão e abrangência, mesmo diante de um corpus

heterogêneo. As métricas de *Faithfulness* (0,79) e *Answer Relevancy* (0,76) evidenciam que o modelo gerador produziu respostas semanticamente alinhadas às fontes, com pequenas variações esperadas devido à natureza dos textos analisados. Por outro lado, o *Context Precision* (0,72) inferior ao *Context Recall* (0,83) revela que o sistema tende a recuperar mais informações do que o necessário, reflexo da dificuldade dos *embeddings* puramente textuais em diferenciar elementos visuais ou simbólicos.

Tabela 2 – Desempenho por tipo de documento

Tipo de documento	Características principais	Métricas mais afetadas	Observações
Matemático (fórmulas)	Notação LaTeX e alta densidade simbólica	*Precision, Faithfulness*	Dificuldade de interpretação semântica pelos *embeddings.*
Educacional (imagens)	Diagramas e figuras explicativas	*Relevancy, Recall*	Imagens não representadas vetorialmente reduzem a relevância.
Textual (narrativo)	Linguagem fluida e linear	Nenhuma significativa	Melhor desempenho geral.

Fonte: elaborado pelo autor (2025).

Os resultados confirmam que o sistema apresentou comportamento estável e previsível, com métricas coerentes entre as etapas de recuperação e geração. O modelo *LLaMA 3.1 8B-Instruct* demonstrou desempenho satisfatório em perguntas conceituais, mantendo terminologia técnica e coesão sintática. Em contrapartida, as questões que envolviam símbolos matemáticos ou dependiam de leitura visual mostraram desempenho inferior, refletindo as limitações dos *embeddings* textuais em representar informação não linguística.

O tempo médio de resposta, de aproximadamente 3,7 segundos, mostrou-se adequado para aplicações de busca interativa. A arquitetura modular garantiu reprodutibilidade e isolamento entre as etapas, validando o projeto proposto. Em síntese, o sistema exibiu desempenho sólido e tecnicamente consistente, confirmando a viabilidade da abordagem RAG em língua portuguesa e consolidando a integração entre modelos de linguagem e bases vetoriais em contextos acadêmicos e técnicos.

CONSIDERAÇÕES FINAIS

A arquitetura desenvolvida demonstrou ser uma solução sólida e funcional para integração entre modelos de linguagem e mecanismos de recuperação semântica. O sistema apresentou resultados consistentes, com desempenho adequado e respostas coerentes ao contexto, confirmando sua aplicabilidade em cenários acadêmicos e experimentais. Sua estrutura modular e reprodutível garantiu estabilidade nas etapas de processamento e facilidade de adaptação a diferentes tipos de conteúdo textual.

Como próximos passos, propõe-se o uso de *embeddings* multimodais e avaliadores mais robustos para aprimorar a interpretação de fórmulas e imagens, além da ampliação do corpus para fortalecer a generalização do modelo. Dessa forma, o trabalho consolida uma base promissora para o uso da Geração Aumentada por Recuperação em ambientes educacionais, mostrando o potencial da inteligência artificial como ferramenta de apoio ao estudo e à compreensão de informações complexas em português.

REFERÊNCIAS

DOCKER. **Docker Documentation**. [S.l.]: Docker, [2025]. Disponível em: <https://docs.docker.com/>. Acesso em: 14 out. 2025.

ES, S.; JAMES, J.; ESPINOSA-ANKE, L.; SCHOCKAERT, S. **RAGAS: Automated Evaluation of Retrieval Augmented Generation**. arXiv preprint, arXiv:2309.15217, 2023. Disponível em: <https://arxiv.org/abs/2309.15217>. Acesso em: 10 out. 2025.

GAO, Y. *et al*. **Retrieval-Augmented Generation for Large Language Models**: A Survey. arXiv preprint, arXiv:2312.10997, 2023. Disponível em: <https://arxiv.org/abs/2312.10997>. Acesso em: 5 out. 2025.

GOOGLE. **Angular Developer Guide**. [S.l.]: Google, [2025]. Disponível em: <https://angular.dev/>. Acesso em: 10 out. 2025.

JONES, M.; BRADLEY, J.; SAKIMURA, N. **JSON Web Token (JWT)**. RFC 7519. [S.l.]: IETF, 2015. Disponível em: <https://www.rfc-editor.org/rfc/rfc7519>. Acesso em: 11 out. 2025.LEWIS, Patrick *et al*. **Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks**. *In*: CONFERENCE ON NEURAL INFORMATION PROCESSING SYSTEMS (NeurIPS), 34., 2020, Vancouver. Proceedings**...** Vancouver: NeurIPS, 2020. Disponível em: <https://proceedings.neurips.cc/paper/2020/file/6b493230205f780e1bc26945df7481e

5-Paper.pdf>. Acesso em: 10 out. 2025.

META. **Introducing Meta LLaMA 3 Models**. Meta AI, 2025. Disponível em: <https://ai.meta.com/llama/>. Acesso em: 14 out. 2025.

MICROSOFT. **TypeScript Handbook**. [S.l.]: Microsoft, 2025. Disponível em: <https://www.typescriptlang.org/docs/>. Acesso em: 9 out. 2025.

MINIO. **MinIO Documentation**. [S.l.]: MinIO, 2025. Disponível em: <https://docs.min.io/>. Acesso em: 14 out. 2025.

OPENROUTER. **OpenRouter API Documentation**. [S.l.]: OpenRouter, 2025. Disponível em: <https://openrouter.ai/docs>. Acesso em: 2 out. 2025.

PRESSMAN, Roger S.; MAXIM, Bruce R. **Software Engineering: A Practitioner's Approach**. 9th ed. New York: McGraw-Hill Education, 2020. Disponível em: <https://highered.mheducation.com/sites/0078022126/index.html>. Acesso em: 24 out. 2025.

ORACLE. **MySQL 8.0 Reference Manual**. [S.l.]: Oracle Corporation, 2025. Disponível em: <https://dev.mysql.com/doc/>. Acesso em: 7 out. 2025.

QDRANT. **Qdrant Documentation**. [S.l.]: Qdrant Tech, 2025. Disponível em: <https://qdrant.tech/documentation/>. Acesso em: 14 out. 2025.

RAMÍREZ, S. **FastAPI**: Modern, Fast Web Framework for Building APIs. [S.l.]: GitHub, 2025. Disponível em: <https://github.com/tiangolo/fastapi>. Acesso em: 6 out. 2025.

SAJID, Haziqa. O que são alucinações LLM? Causas, preocupação ética e prevenção. **Unite.AI**, 2023. Disponível em: <https://www.unite.ai/pt/what-are-llm-hallucinations-causes-ethical-concern-prevention/>. Acesso em: 8 out. 2025.

SQLALCHEMY. **SQLAlchemy ORM Documentation**. [S.l.]: SQLAlchemy, 2025. Disponível em: <https://docs.sqlalchemy.org/>. Acesso em: 12 out. 2025.

VASWANI, A. *et al*. Attention Is All You Need. **Advances in Neural Information Processing Systems**, 2017. Disponível em: <https://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf>. Acesso em: 10 out. 2025.

ZHAO, Wayne Xin *et al*. **A Survey of Large Language Models**. arXiv preprint, arXiv:2303.18223, 2023. Disponível em: <https://arxiv.org/abs/2303.18223>. Acesso em: 13 out. 2025.