
# Emile Chatbot: Alavancando a Consulta a Informações Institucionais com IA

Trabalho de Conclusão de Curso

**Guilherme Paim Motta**

Sandro Santos Andrade
Orientador

Instituto Federal da Bahia - IFBA
Curso de Análise e Desenvolvimento de Sistemas
Campus Salvador

Salvador, Bahia, Brasil
2026

# Sumário

# 1 Visão Geral

## 1.1 Declaração do Problema

O avanço recente dos modelos de linguagem baseados em Inteligência Artificial possibilitou o surgimento de sistemas conversacionais cada vez mais sofisticados. Entretanto, modelos tradicionais apresentam limitações importantes quando utilizados em ambientes institucionais, especialmente devido à possibilidade de geração de respostas incorretas ou desatualizadas.

No contexto do Instituto Federal da Bahia (IFBA), estudantes frequentemente necessitam consultar informações presentes em documentos oficiais, como editais, regulamentos acadêmicos, calendários semestrais e portarias administrativas. Apesar desses documentos estarem disponíveis digitalmente, o processo de busca manual pode ser lento e confuso, principalmente devido ao grande volume de arquivos disponíveis.

Embora essas informações estejam publicamente disponíveis, muitos estudantes enfrentam dificuldades para localizar rapidamente respostas específicas. Frequentemente, é necessário abrir diversos arquivos PDF e navegar manualmente por dezenas de páginas para encontrar informações simples, como prazos de inscrição, critérios de seleção ou datas de resultados.

Além disso, muitos alunos utilizam linguagem informal ou abreviações ao buscar informações, dificultando ainda mais a localização manual dos dados desejados. Esse problema se intensifica em períodos de publicação de editais importantes, quando há aumento significativo na demanda por informações institucionais.

Outro fator relevante é a necessidade de garantir confiabilidade nas respostas fornecidas. Um chatbot tradicional baseado apenas em modelos de linguagem poderia gerar respostas incorretas ou inventadas, fenômeno conhecido como alucinação. Em um ambiente institucional, isso representa um risco significativo, já que informações incorretas sobre editais ou regulamentos podem causar prejuízos acadêmicos aos estudantes.

Dessa forma, torna-se necessária a construção de uma solução capaz de recuperar informações diretamente dos documentos oficiais antes da geração das respostas, garantindo maior precisão, rastreabilidade e confiabilidade.

## 1.2 Proposta de Solução de Software

Este trabalho propõe o desenvolvimento de um chatbot institucional integrado ao aplicativo Emile do IFBA, utilizando a abordagem Retrieval-Augmented Generation (RAG).

A escolha por RAG em detrimento de abordagens alternativas é motivada pelas características específicas do problema. O **fine-tuning** de um modelo de linguagem sobre os documentos do IFBA exigiria retreinamento periódico a cada publicação de novo edital ou atualização de regulamento, tornando a solução operacionalmente inviável em um ambiente com alta frequência de publicação de documentos. Sistemas de **busca textual tradicional** sem geração de linguagem retornam documentos brutos ao usuário, sem síntese nem capacidade de responder perguntas em linguagem natural. A arquitetura RAG resolve ambos os problemas: a base documental pode ser atualizada sem retreinamento do modelo, e a combinação de recuperação com geração permite produzir respostas contextualizadas diretamente em linguagem natural.

A arquitetura RAG combina recuperação de informações em bases documentais com geração textual baseada em modelos de linguagem. Ao receber uma pergunta, o sistema primeiro recupera os trechos de documentos mais relevantes para aquela consulta e, em seguida, fornece esses trechos

como contexto para o modelo de linguagem, que então gera a resposta com base exclusivamente no material recuperado. Essa abordagem reduz significativamente o fenômeno de alucinação — geração de informações incorretas ou inexistentes — pois o modelo responde fundamentado em evidência documental concreta em vez de depender exclusivamente de seu conhecimento parametrizado.

O sistema processa documentos institucionais do IFBA em formato PDF, incluindo editais de programas estudantis, resoluções institucionais, calendários acadêmicos e regulamentos internos, armazenando essas informações em uma base vetorial para recuperação semântica. O pipeline completo envolve etapas de ingestão documental, extração e segmentação textual, geração de embeddings vetoriais, recuperação híbrida e síntese da resposta por modelo de linguagem, com rastreabilidade das fontes documentais em cada resposta gerada.

## 1.3 Objetivo Geral

O objetivo geral deste trabalho é desenvolver e avaliar um chatbot institucional baseado em Retrieval-Augmented Generation (RAG), integrado ao aplicativo Emile, capaz de responder perguntas em linguagem natural sobre documentos oficiais do IFBA com rastreabilidade das fontes utilizadas.

A proposta busca automatizar o acesso a informações institucionais que normalmente exigem consulta manual a arquivos PDF, como editais, regulamentos, portarias, calendários acadêmicos e documentos administrativos. Para isso, o sistema combina ingestão automatizada de documentos, armazenamento vetorial, recuperação híbrida, reescrita de consultas, reranqueamento de trechos, geração de respostas por modelo de linguagem e mecanismos de controle de confiabilidade.

## 1.4 Objetivos Específicos

Para alcançar o objetivo geral, foram definidos os seguintes objetivos específicos:

- Projetar uma arquitetura RAG modular adequada ao contexto institucional do IFBA, considerando atualização periódica da base documental, rastreabilidade e controle de alucinações;
- Implementar um pipeline de ingestão automatizada de documentos PDF institucionais, incluindo detecção de arquivos novos ou modificados, extração textual estruturada, segmentação em chunks, enriquecimento com metadados e persistência dos dados processados;
- Utilizar modelos de embeddings para transformar trechos documentais e consultas de usuários em representações vetoriais compatíveis com busca semântica;
- Armazenar os documentos processados em uma base vetorial persistente, utilizando PostgreSQL com PGVector e suporte à busca híbrida;
- Implementar mecanismos de recuperação híbrida combinando busca semântica densa e busca lexical, de forma a contemplar tanto perguntas informais quanto consultas com termos exatos, números de editais, siglas e nomes de programas;
- Implementar uma etapa de reescrita e expansão de consultas para tratar abreviações, siglas, sinônimos, erros simples de digitação e formulações informais utilizadas por estudantes;
- Implementar uma etapa de reranqueamento dos trechos recuperados, reduzindo a chance de envio de trechos irrelevantes ou de documentos semelhantes ao modelo de linguagem;
- Definir um limiar de confiança para evitar que o modelo de linguagem seja acionado quando não houver evidência documental suficiente para responder à pergunta;

- Integrar um modelo de linguagem ao pipeline RAG para gerar respostas em português brasileiro, com linguagem simples, amigável e fundamentada exclusivamente nos documentos recuperados;
- Implementar mecanismos de cache de perguntas repetidas, reduzindo latência em consultas recorrentes durante períodos de alta demanda por informações institucionais;
- Integrar o chatbot à interface do aplicativo Emile, incluindo resposta em streaming, mensagens de erro compreensíveis, sugestões rápidas de perguntas e identificação visual do assistente;
- Avaliar o sistema por meio de testes funcionais, métricas de recuperação, métricas de geração, análise de latência e comparação antes e depois das otimizações realizadas.

## 1.5 Escopo e Limitações

O escopo deste trabalho limita-se à construção de um assistente conversacional para consulta a documentos institucionais previamente indexados. O chatbot não substitui os documentos oficiais do IFBA, não realiza análise jurídica ou administrativa definitiva e não toma decisões institucionais, como deferimento de inscrições, concessão de auxílios ou validação de requisitos individuais de estudantes.

As respostas geradas pelo sistema dependem diretamente da qualidade, atualização e cobertura dos documentos presentes na base. Caso uma informação não esteja disponível nos documentos indexados, o chatbot deve informar explicitamente que não encontrou respaldo documental, em vez de inferir uma resposta com base no conhecimento interno do modelo de linguagem.

O sistema também não tem como objetivo responder perguntas gerais fora do domínio institucional do IFBA. Consultas sobre assuntos externos, opiniões pessoais, conteúdo acadêmico não relacionado a documentos oficiais ou solicitações incompatíveis com o escopo do assistente são recusadas ou redirecionadas para a mensagem padrão de ausência de evidência documental.

## 1.6 Tecnologias adotadas

O sistema foi desenvolvido utilizando um conjunto de tecnologias modernas e especializadas para aplicações baseadas em Retrieval-Augmented Generation, priorizando compatibilidade com processamento de linguagem natural em português brasileiro, recuperação híbrida de informações e capacidade de execução em infraestrutura própria da instituição.

**LlamaIndex** é utilizado como framework principal de orquestração do pipeline RAG. Diferentemente de abordagens mais generalistas, o LlamaIndex oferece abstrações nativas para ingestão documental, indexação vetorial, recuperação semântica e composição de pipelines RAG, simplificando a integração entre os diferentes componentes do sistema sem necessidade de configuração manual de cada etapa.

**LlamaParse** é responsável pelo processamento inicial dos documentos PDF institucionais. A ferramenta realiza extração e estruturação do conteúdo textual dos arquivos, convertendo-os para o formato Markdown de forma que a organização hierárquica original dos documentos seja preservada durante o processo de segmentação.

**FastAPI** foi escolhido como framework para desenvolvimento do backend da aplicação. A escolha justifica-se pela alta performance, suporte nativo à programação assíncrona — essencial para execução do pipeline de ingestão em segundo plano sem bloqueio da API — e facilidade de documentação automática via OpenAPI.

**PostgreSQL com extensão PGVector** é utilizado para armazenamento vetorial. O PGVector permite realizar buscas por similaridade vetorial diretamente sobre o banco relacional, eliminando a necessidade de serviços externos dedicados. O banco é configurado com suporte à busca híbrida e com configuração de busca textual otimizada para o português (`text_search_config="portuguese"`), permitindo que a componente léxica da recuperação opere com suporte nativo a stemming e stopwords em língua portuguesa.

**Qwen3 Embedding (`qwen3-embedding:4b`)**, hospedado localmente por meio da infraestrutura Ollama do IFBA, é utilizado para geração dos embeddings vetoriais. O modelo produz vetores de dimensão 2560, oferecendo representações semânticas densas compatíveis com documentos multilíngues, incluindo o português brasileiro. A utilização de um modelo hospedado na própria infraestrutura institucional evita dependências de APIs externas pagas e elimina custos variáveis por volume de requisições, aspecto relevante dado o volume potencialmente elevado de documentos a serem indexados.

**Ollama** é a plataforma responsável pela hospedagem e execução local dos modelos utilizados pelo sistema. Disponibilizado pelo próprio IFBA no endpoint `ollama-api.ifba.edu.br`, permite que o chatbot opere sobre infraestrutura institucional, reduzindo dependência de APIs externas e custos variáveis por requisição.

Na etapa de geração textual, após avaliação comparativa entre diferentes configurações, foi adotado o modelo **`qwen2.5:7b-instruct`** como padrão do chatbot. Essa escolha foi motivada por seu melhor equilíbrio entre latência, estabilidade e fidelidade no conjunto de benchmark definido.

Inicialmente, foram avaliadas configurações baseadas no modelo `qwen3:30b` com mecanismo de *thinking*. Embora esse modelo apresentasse capacidade de raciocínio mais extensa, os testes demonstraram que grande parte da latência era consumida na etapa de geração deliberativa, chegando a encerrar algumas requisições sem produzir conteúdo visível. A substituição por um modelo instrucional sem *thinking* explícito reduziu significativamente o tempo de resposta e eliminou os casos de resposta vazia no benchmark final.

## 1.7 Trabalhos Relacionados

Sistemas baseados em Retrieval-Augmented Generation têm sido amplamente utilizados em aplicações de recuperação de informação e assistentes conversacionais institucionais.

Lewis et al. [1] propuseram a arquitetura RAG como forma de integrar recuperação documental a modelos generativos, reduzindo problemas de alucinação em tarefas intensivas em conhecimento. O trabalho introduz a utilização de mecanismos de recuperação vetorial acoplados a modelos generativos, permitindo que respostas sejam produzidas com base em documentos externos.

Gao et al. [2] realizaram um levantamento abrangente sobre arquiteturas RAG modernas, descrevendo estratégias de chunking, recuperação híbrida, reranqueamento e avaliação de sistemas baseados em Large Language Models. O survey também destaca desafios relacionados à precisão factual, latência e atualização contínua de bases documentais.

Karpukhin et al. [3] apresentaram o Dense Passage Retrieval (DPR), abordagem baseada em embeddings densos para recuperação semântica de documentos. O modelo tornou-se uma das principais referências para sistemas modernos de busca vetorial utilizados em pipelines RAG.

Reimers e Gurevych [4] propuseram o Sentence-BERT, arquitetura voltada para geração eficiente de embeddings semânticos, amplamente utilizada em sistemas de recuperação textual e busca semântica multilíngue.

Além de trabalhos acadêmicos, diversas instituições têm adotado chatbots baseados em Inteligência Artificial para suporte informacional. Universidades e organizações públicas utilizam assistentes conversacionais para responder dúvidas relacionadas a processos administrativos, calendários acadêmicos e regulamentos internos. Entretanto, muitas dessas soluções ainda apresentam limitações relacionadas à confiabilidade das respostas e ausência de mecanismos robustos de grounding documental.

O presente trabalho diferencia-se por aplicar arquiteturas RAG modernas especificamente ao contexto institucional do IFBA, propondo uma solução integrada ao aplicativo Emile e voltada à recuperação de informações acadêmicas e administrativas diretamente a partir de documentos oficiais da instituição.

# 2 Requisitos

## 2.1 Requisitos Funcionais

Os requisitos funcionais descrevem os serviços e funcionalidades que deverão ser oferecidos pelo chatbot institucional.

RF01 O sistema deve permitir que usuários realizem perguntas em linguagem natural relacionadas a documentos institucionais do IFBA.

RF02 O sistema deve recuperar informações a partir de documentos institucionais previamente indexados na base vetorial.

RF03 O sistema deve gerar respostas contextualizadas utilizando modelos de linguagem integrados ao pipeline RAG.

RF04 O sistema deve informar a fonte documental utilizada para geração da resposta, incluindo nome do documento ou trecho correspondente.

RF05 O sistema deve permitir atualização periódica da base documental por meio de novos arquivos PDF institucionais.

RF06 O sistema deve processar documentos institucionais realizando extração textual e segmentação em chunks.

RF07 O sistema deve gerar embeddings vetoriais dos documentos processados para permitir recuperação semântica.

RF08 O sistema deve utilizar recuperação híbrida combinando busca vetorial e busca lexical.

RF09 O sistema deve recuperar inicialmente trechos documentais relevantes para compor o contexto da geração.

RF10 O sistema deve limitar respostas ao domínio institucional do IFBA.

RF11 O sistema deve ser integrado ao aplicativo Emile por meio de APIs de comunicação.

RF12 O sistema deve registrar logs básicos de consultas realizadas para fins de monitoramento e melhoria contínua.

RF13 O sistema deve realizar reescrita ou expansão de consultas para tratar siglas, abreviações, sinônimos, erros simples e formulações informais utilizadas pelos estudantes.

RF14 O sistema deve aplicar reranqueamento sobre os trechos inicialmente recuperados, priorizando os documentos mais relevantes antes da geração da resposta final.

RF15 O sistema deve aplicar limiar de confiança sobre os resultados da recuperação, evitando a geração de respostas quando não houver evidência documental suficiente.

RF16 O sistema deve retornar uma mensagem padronizada quando a informação solicitada não estiver presente nos documentos disponíveis.

RF17 O sistema deve utilizar cache de perguntas repetidas para reduzir latência e custo computacional em consultas recorrentes.

RF18 O sistema deve invalidar ou versionar o cache quando houver atualização da base documental.

RF19 O sistema deve registrar métricas operacionais por requisição, incluindo tempo de embedding, tempo de recuperação, tempo de geração, quantidade de trechos recuperados, tamanho do contexto e motivo de encerramento da resposta.

RF20 O sistema deve disponibilizar respostas em streaming para a interface do aplicativo Emile.

RF21 O sistema deve oferecer sugestões rápidas de perguntas na interface do chat, auxiliando o usuário na formulação de consultas institucionais.

RF22 O sistema deve tratar erros de timeout, falha de comunicação com o modelo, resposta vazia, truncamento e indisponibilidade do endpoint de forma explícita para o usuário.

## 2.2 Requisitos Não-Funcionais

Os requisitos não-funcionais descrevem restrições, características de qualidade e propriedades operacionais do sistema.

RNF01 O sistema deve apresentar desempenho compatível com uso interativo, com tempo total inferior a 30 segundos para consultas comuns e meta operacional de p50 inferior ou igual a 10 segundos e p95 inferior ou igual a 15 segundos no conjunto de benchmark definido.

RNF02 O sistema deve suportar múltiplas consultas simultâneas realizadas por usuários do aplicativo Emile.

RNF03 O sistema deve garantir disponibilidade contínua durante períodos letivos e publicação de editais institucionais.

RNF04 O sistema deve utilizar mecanismos de segurança para reduzir a geração de respostas fora do domínio institucional.

RNF05 O sistema deve possuir mecanismos de filtragem contra prompt injection e tentativas de jailbreak.

RNF06 O sistema deve permitir escalabilidade horizontal da infraestrutura de recuperação vetorial.

RNF07 O sistema deve possuir arquitetura modular para facilitar manutenção e substituição de componentes do pipeline RAG.

RNF08 O sistema deve ser compatível com documentos em português brasileiro.

RNF09 O sistema deve armazenar embeddings vetoriais de forma persistente utilizando banco vetorial apropriado.

RNF10 O sistema deve minimizar respostas alucinatórias por meio de grounding documental e métricas de fidelidade.

RNF11 O sistema deve manter rastreabilidade entre respostas geradas e documentos utilizados no contexto.

RNF12 O sistema deve permitir futuras integrações com outros sistemas institucionais do IFBA.

## 2.3 Matriz de Rastreabilidade dos Requisitos

A matriz de rastreabilidade relaciona os requisitos definidos com os componentes implementados e com as estratégias de validação utilizadas. Essa relação permite verificar se cada requisito possui correspondência direta na implementação e se foi avaliado por meio de testes funcionais, testes de desempenho ou análise de qualidade do pipeline RAG.

| **Requisito** | **Componente Implementado** | **Validação** |
|---|---|---|
| RF01 | Interface de chat no aplicativo Emile e endpoint `POST /chatbot/ask` | Teste funcional com perguntas em linguagem natural. |
| RF02 | Retriever híbrido sobre base PGVector | Avaliação de recuperação no conjunto de perguntas de referência. |
| RF03 | Integração com modelo de linguagem via Ollama | Teste de geração de respostas com contexto documental. |
| RF04 | Prompt estruturado com exigência de citação de fonte | Verificação da fonte apresentada nas respostas geradas. |
| RF05 | Endpoint `POST /chatbot/ingest` e controle por hash | Teste de ingestão com documento novo, repetido e alterado. |
| RF08 | Busca híbrida densa e lexical | Comparação entre consultas semânticas e consultas com termos exatos. |
| RF13 | Módulo de reescrita e expansão de consultas | Comparação de recuperação antes e depois da reescrita. |
| RF14 | Módulo de reranqueamento | Avaliação de precisão dos trechos antes e depois do reranqueamento. |
| RF15 | Filtro por limiar de confiança | Teste com perguntas sem cobertura documental e perguntas ambíguas. |
| RF17 | Cache de perguntas normalizadas | Teste de *cache miss*, *cache hit* e invalidação após ingestão. |
| RF19 | Logs estruturados e benchmark automatizado | Análise dos arquivos JSON/CSV de benchmark. |
| RF20 | Streaming NDJSON no endpoint `/chatbot/ask` | Teste de fragmentação de objetos NDJSON e requisições sequenciais. |
| RNF01 | Otimizações de geração, cache e limiar de confiança | Benchmark de latência com p50, p95 e tempo até primeiro conteúdo. |

| | | |
|---|---|---|
| RNF10 | Grounding documental, prompt restritivo e limiar de confiança | Avaliação de fidelidade e testes de ausência de evidência. |

# 3 Design

## 3.1 Dimensões de Design do Sistema RAG

A definição das dimensões de design em sistemas Retrieval-Augmented Generation (RAG) possui impacto direto na qualidade da recuperação de informações, no desempenho computacional e na confiabilidade das respostas geradas pelo modelo de linguagem [2].

| **Dimensão** | **Alternativas Consideradas** | **Escolha Adotada** | **Justificativa** |
|---|---|---|---|
| Estrutura do Pipeline | Naive RAG, Standard RAG, Advanced/Modular RAG | Advanced/Modular RAG | Permite controlar separadamente ingestão, recuperação, reescrita, reranqueamento, cache, geração e métricas. |
| Fonte de Dados | Web aberta, APIs externas, documentos internos | PDFs institucionais internos | O escopo do chatbot é restrito a documentos oficiais do IFBA, garantindo maior confiabilidade e rastreabilidade. |
| Carregamento dos Dados | Tempo real, manual, lote periódico | Ingestão em lotes | Documentos institucionais são publicados periodicamente, tornando a ingestão batch mais simples, barata e controlável. |
| Parsing e Extração | Extração simples, OCR, parsing estruturado | Parsing estruturado com saída em Markdown | Preserva melhor títulos, listas, tabelas e organização textual dos documentos. |
| Chunking | Tamanho fixo, recursivo, semântico, janela deslizante | Chunking avaliado empiricamente | A configuração final foi selecionada a partir da comparação entre diferentes tamanhos e sobreposições. |
| Modelo de Embedding | API externa, modelo local, modelo ajustado | Modelo multilíngue autohospedado | Reduz dependência externa, custo variável e exposição de dados institucionais. |
| Banco Vetorial | FAISS, banco vetorial dedicado, PGVector | PostgreSQL com PGVector | Integra armazenamento relacional e vetorial em uma infraestrutura já consolidada. |

| | | | |
|---|---|---|---|
| Tipo de Retriever | Dense, Sparse, Hybrid | Hybrid Retriever | Combina busca por significado com correspondência exata de termos, siglas e números de editais. |
| Pré-recuperação | Consulta original, normalização, query rewriting | Query rewriting e expansão lexical | Melhora a recuperação quando estudantes usam abreviações, gírias, erros simples ou termos diferentes dos documentos. |
| Pós-recuperação | Sem filtro, threshold, compressão, re-ranking | Threshold de confiança e re-ranking opcional por embedding | Reduz ruído no contexto e evita chamadas à LLM quando não há evidência suficiente. |
| Geração | Resposta direta, multi-step, agentic | Single-pass generation | Uma única geração com contexto recuperado é suficiente para perguntas institucionais e reduz complexidade. |
| Cache | Sem cache, cache exato, cache semântico | Query caching exato normalizado | Oferece ganho de latência em perguntas repetidas com menor risco de retornar resposta inadequada. |
| Avaliação | Testes manuais, métricas automáticas, benchmark | Benchmark, métricas RAG e validação funcional | Permite medir qualidade, latência, recuperação, fidelidade e comportamento do app. |

Segundo Gao et al. [2], sistemas RAG modernos são compostos por múltiplos componentes independentes, incluindo estratégias de recuperação, armazenamento vetorial, embeddings, pós-processamento e geração textual. Dessa forma, diferentes escolhas arquiteturais podem resultar em variações significativas de precisão, latência e custo computacional.

Nesta seção são apresentadas as principais dimensões de design consideradas no desenvolvimento do chatbot institucional do IFBA. Para cada dimensão são apresentados os possíveis valores, a escolha adotada, as tecnologias previstas e a justificativa técnica da decisão.

### 3.1.1 Pipeline Structure

A estrutura do pipeline define como os componentes internos do sistema RAG serão organizados e conectados. Segundo Gao et al. [2], arquiteturas RAG podem ser classificadas em três categorias principais:

- **Naive RAG**: abordagem mais simples, composta exclusivamente pelas etapas de recuperação direta e geração. Apesar da baixa complexidade, apresenta limitações significativas em precisão quando aplicada a bases documentais extensas ou com documentos semanticamente semelhantes entre si;
- **Standard RAG**: incorpora etapas básicas de pré-processamento documental e geração contextualizada, oferecendo melhor qualidade em relação ao Naive RAG, porém com pouco controle sobre cada estágio do fluxo;

- **Advanced/Modular RAG**: adiciona componentes intermediários especializados, como reescrita de consultas, múltiplos estágios de recuperação, reranqueamento semântico e mecanismos de verificação de relevância. Cada componente é independente e substituível, permitindo ajuste granular do pipeline sem impacto nos demais estágios.

Neste trabalho foi escolhida a arquitetura **Advanced/Modular RAG**, implementada utilizando o framework LlamaIndex.

Essa escolha é justificada pela natureza dos documentos institucionais do IFBA. Editais, regulamentos e resoluções frequentemente apresentam linguagem formal densa, organização hierárquica complexa e conteúdo semanticamente semelhante entre documentos de programas distintos. Em cenários onde um estudante questiona sobre um programa específico utilizando linguagem informal ou abreviações, a recuperação simples sem etapas intermediárias tende a retornar trechos pouco relevantes ou misturar informações de diferentes editais.

A modularidade do pipeline também é relevante para a longevidade da solução. Modelos de embedding, estratégias de chunking e mecanismos de recuperação poderão ser substituídos ou ajustados conforme a base documental cresce ou os padrões de consulta dos usuários evoluem, sem necessidade de reestruturação da arquitetura principal.

### 3.1.2 Chunking Strategy

Chunking consiste no processo de divisão dos documentos em partes menores para indexação vetorial. Segundo Gao et al. [2], essa etapa possui impacto direto na qualidade da recuperação semântica: unidades de texto muito grandes tendem a diluir o sinal semântico do embedding, enquanto unidades muito pequenas podem perder o contexto necessário para compreensão da informação.

As principais estratégias incluem:

- **Fixed chunking**: divisão em segmentos de tamanho fixo em número de caracteres ou tokens, independentemente da estrutura textual subjacente;
- **Semantic chunking**: segmentação baseada em similaridade semântica entre sentenças consecutivas, agrupando trechos com alto grau de coesão temática;
- **Recursive chunking**: divisão hierárquica que respeita separadores estruturais do documento, como títulos, capítulos e parágrafos;
- **Sliding window**: criação de chunks com sobreposição controlada entre segmentos consecutivos, preservando continuidade textual nas bordas.

Neste trabalho é utilizado o componente **SentenceSplitter** do LlamaIndex, configurado com `chunk_size=400` tokens e `chunk_overlap=50` tokens.

O tamanho de 400 tokens foi definido com base no equilíbrio entre granularidade semântica e preservação de contexto local. Trechos de editais como critérios de seleção ou descrições de benefícios frequentemente se estendem por múltiplos parágrafos interdependentes. Chunks excessivamente pequenos fragmentariam essas informações em unidades sem sentido isolado. O overlap de 50 tokens garante que informações posicionadas nas bordas entre dois chunks consecutivos não sejam perdidas no momento da recuperação.

A adoção do SentenceSplitter em detrimento de estratégias de tamanho fixo em caracteres é motivada pela necessidade de respeitar os limites naturais das sentenças nos documentos institucionais. Essa abordagem evita que frases sejam interrompidas no meio, preservando a coerência

gramatical e semântica de cada unidade indexada.

### 3.1.3 Embedding Model

Embeddings são representações vetoriais densas que capturam relações semânticas entre palavras, frases e documentos [4]. Em sistemas RAG, a qualidade dos embeddings influencia diretamente a capacidade do retriever em identificar documentos relevantes para uma consulta, mesmo quando a formulação da pergunta difere da terminologia utilizada nos documentos originais.

Os principais tipos incluem:

- **Embeddings treinados localmente**: modelos ajustados sobre corpus específico do domínio, com alta especialização mas elevado custo de treinamento e manutenção;
- **Embeddings proprietários via API**: modelos como `text-embedding-3-large` da OpenAI, com alta qualidade semântica mas com dependência de serviço externo e custo variável por volume de requisições;
- **Modelos multilíngues pré-treinados auto-hospedados**: modelos open-source treinados sobre grandes corpora multilíngues, combinando qualidade semântica com suporte a múltiplos idiomas sem custo por requisição e sem dependência de infraestrutura externa.

Neste trabalho foi adotado o modelo **Qwen3 Embedding (`qwen3-embedding:4b`)**, hospedado na infraestrutura Ollama do IFBA. O modelo gera embeddings de dimensão 2560, configurados no banco vetorial para buscas por similaridade de alta precisão.

A escolha por um modelo auto-hospedado foi motivada por três fatores principais. Primeiro, a necessidade de processar documentos institucionais com privacidade, sem transmissão de conteúdo sensível a serviços externos. Segundo, a ausência de custos variáveis por volume de requisições, aspecto relevante dado o volume potencialmente elevado de chunks gerados durante a indexação da base documental do IFBA. Terceiro, a disponibilidade de infraestrutura Ollama já mantida pelo próprio IFBA, tornando viável a execução do modelo sem custo operacional adicional.

### 3.1.4 Vector Storage

O armazenamento vetorial é responsável por indexar os embeddings gerados durante o processamento documental e disponibilizá-los para consultas de similaridade em tempo de execução.

As principais abordagens incluem:

- **Bancos relacionais com extensão vetorial**: como PostgreSQL com PGVector, que adiciona suporte a tipos de dados vetoriais e operações de similaridade sobre um banco relacional já consolidado;
- **Bancos vetoriais dedicados**: como Qdrant, Pinecone e Weaviate, projetados especificamente para armazenamento e recuperação de alta performance em espaços vetoriais de alta dimensionalidade;
- **Soluções in-memory**: como FAISS, adequadas para prototipação e cenários sem necessidade de persistência.

Neste trabalho foi adotado o **PostgreSQL com extensão PGVector**, configurado com hybrid search e vetores de dimensão 2560.

A escolha pelo PGVector em detrimento de bancos vetoriais dedicados justifica-se pelo contexto

operacional do projeto. O PostgreSQL já compõe a infraestrutura do aplicativo Emile, reduzindo a complexidade da arquitetura de implantação ao eliminar a necessidade de um serviço adicional. Além disso, a configuração `text_search_config= "portuguese"` habilita suporte nativo a stemming e remoção de stopwords em português na componente léxica da busca híbrida, melhorando a precisão de consultas que incluem termos como artigos e preposições.

### 3.1.5 Retriever Type

O retriever é responsável pela identificação e recuperação dos trechos documentais mais relevantes para uma determinada consulta do usuário. A estratégia de recuperação tem impacto direto na qualidade e na abrangência dos documentos retornados.

Os principais tipos incluem:

- **Dense Retrieval**: utiliza embeddings vetoriais para medir similaridade semântica entre consulta e documentos indexados. É eficaz para recuperar documentos semanticamente relacionados mesmo quando não há correspondência exata de termos [3];
- **Sparse Retrieval**: baseia-se em algoritmos tradicionais como BM25, que calculam relevância por frequência e distribuição de termos. É especialmente eficaz para consultas com termos específicos como números de editais, nomes de programas ou datas exatas;
- **Hybrid Retrieval**: combina Dense e Sparse Retrieval em uma única etapa de recuperação, aproveitando as vantagens complementares de ambas as abordagens.

Neste trabalho foi escolhido o **Hybrid Retriever**, configurado com `Vector Store Query hybrid` e recuperação dos 5 trechos mais relevantes (`similarity_top_k=5`).

A combinação das duas abordagens é especialmente adequada ao contexto do IFBA. Consultas realizadas em linguagem formal ou técnica — como a busca por um artigo específico de um regulamento ou o número de um edital — beneficiam-se do Sparse Retrieval pela correspondência exata de termos. Por outro lado, perguntas informais de estudantes — como "quando fecha as inscrições do auxílio moradia?" — beneficiam-se do Dense Retrieval, que identifica documentos semanticamente relacionados independentemente da terminologia utilizada. A recuperação híbrida maximiza a cobertura sem exigir que o usuário adapte sua linguagem ao formato dos documentos institucionais.

### 3.1.6 Pre-retrieval Optimization: Query Rewriting

A etapa de otimização pré-recuperação atua antes da busca documental propriamente dita. Seu objetivo é transformar a consulta original do usuário em uma formulação mais adequada ao mecanismo de recuperação, preservando a intenção original da pergunta. Em sistemas institucionais, essa etapa é especialmente relevante porque usuários frequentemente utilizam abreviações, termos informais, erros simples de digitação ou expressões diferentes das utilizadas nos documentos oficiais.

As principais abordagens consideradas foram:

- **Consulta original sem modificação**: mantém exatamente o texto digitado pelo usuário. É simples e preserva totalmente a entrada original, mas pode apresentar baixa recuperação quando a pergunta utiliza termos informais ou abreviações;
- **Normalização lexical**: remove espaços excedentes, padroniza caixa, corrige variações simples e expande siglas conhecidas. É eficiente e previsível, porém limitada a padrões previamente definidos;

Neste trabalho foi adotada uma estratégia **heurística e determinística de query rewriting**. A consulta é normalizada e enriquecida por regras explícitas para siglas, variações ortográficas, equivalências institucionais e formulações telegráficas. A consulta original é preservada como fallback e também é executada quando ocorre reescrita, sendo selecionado o resultado com melhor score de recuperação.

Exemplos de equivalências tratadas incluem:

- `CadÚnico`, `cadunico` e `Cadastro Único`;
- `auxílio`, `bolsa`, `assistência estudantil` e `benefício`;
- `estágio`, `estágio curricular`, `estágio supervisionado` e `plano de atividades`;
- `requerimento`, `solicitação`, `protocolo` e `formulário`.

Essa etapa contribui para reduzir a distância entre a linguagem do estudante e a linguagem formal dos documentos oficiais, aumentando a chance de o retriever localizar os trechos corretos mesmo quando a pergunta é formulada de maneira coloquial.

### 3.1.7 Post-retrieval Processing

Após a recuperação inicial dos documentos, o sistema aplica etapas de pós-processamento para melhorar a qualidade do contexto enviado ao modelo de linguagem. Essa fase é necessária porque a busca híbrida retorna candidatos relevantes, mas ainda pode incluir trechos de documentos semelhantes, documentos de semestres distintos ou parágrafos estruturalmente próximos que não respondem diretamente à pergunta.

As principais técnicas consideradas foram:

- **Filtering**: remoção de trechos que não atingem um limiar mínimo de relevância;
- **Compression**: redução do tamanho dos trechos recuperados, mantendo apenas as informações mais relevantes;
- **Re-ranking**: reordenação dos candidatos por um modelo ou função de relevância mais específica do que a busca inicial;
- **Deduplicação**: remoção de trechos repetidos ou quase repetidos recuperados a partir do mesmo documento.

Neste trabalho foram implementados o **limiar de confiança** e o suporte a **re-ranking opcional por embedding**.

O limiar de confiança atua como uma barreira antes da geração. Caso os trechos recuperados apresentem pontuação inferior ao valor mínimo definido experimentalmente, o sistema não aciona o modelo de linguagem e retorna a mensagem padronizada de ausência de evidência documental:

> Não encontrei essa informação nos documentos disponíveis no momento.

Essa decisão reduz o risco de alucinação, pois impede que o modelo gere respostas com base em contexto fraco ou irrelevante. Além disso, reduz a latência em perguntas sem cobertura documental, pois evita uma chamada desnecessária ao modelo gerador.

Os candidatos passam pelo limiar de confiança e, quando o re-ranking está habilitado por configuração, são submetidos a uma segunda ordenação baseada em embeddings. Nos experimentos

realizados, essa etapa não alterou as métricas de recuperação e adicionou latência; por isso, permaneceu desativada na configuração padrão.

### 3.1.8 Prompt Engineering

O Prompt Engineering define como o modelo de linguagem será instruído a utilizar o contexto documental recuperado para geração das respostas. Em sistemas RAG, a qualidade do prompt exerce influência direta na consistência, precisão e aderência ao domínio das respostas produzidas [2]. Um modelo generativo com bom desempenho intrínseco pode produzir respostas inconsistentes ou fora do domínio esperado na ausência de instruções adequadas, tornando essa dimensão crítica em aplicações institucionais.

As principais abordagens incluem:

- **Zero-shot**: o modelo recebe apenas o contexto recuperado e a pergunta do usuário, sem exemplos de comportamento esperado. Essa abordagem depende integralmente da capacidade do modelo em inferir formato e tom adequados a partir do contexto, podendo resultar em variações indesejadas de estilo e comportamento entre respostas distintas;
- **Few-shot**: o prompt inclui pares exemplares de pergunta e resposta que demonstram ao modelo o formato, o estilo e o comportamento esperados. Os exemplos funcionam como referência implícita para o modelo ao processar novas consultas, induzindo padronização sem necessidade de enumeração explícita de todas as regras;
- **Instruction prompting**: o modelo recebe um conjunto explícito de instruções comportamentais no sistema de prompt, definindo regras sobre domínio de atuação, formato de resposta, como citar fontes e como tratar perguntas fora do escopo documental. Essa abordagem oferece maior controle e previsibilidade do comportamento do modelo.

Neste trabalho foi adotado o **Instruction Prompting**, com instruções explícitas de escopo, ausência de evidência, formato curto de resposta e indicação controlada da fonte documental.

O Instruction Prompting é especialmente relevante no contexto institucional do IFBA porque permite estabelecer restrições comportamentais explícitas sobre o modelo. Por meio do sistema de prompt, o modelo é instruído a responder exclusivamente com base nos documentos recuperados, a recusar perguntas fora do domínio institucional, a indicar a fonte documental utilizada em cada resposta e a informar ao usuário quando a informação solicitada não consta na base indexada — em vez de recorrer ao conhecimento parametrizado do modelo para inferir ou inventar uma resposta.

As instruções explícitas também contribuem para a mitigação de alucinações ao reforçar que o modelo não deve especular além do contexto fornecido. Sem essas restrições, mesmo modelos com contexto documental de qualidade podem complementar lacunas de informação com conhecimento geral não verificado, introduzindo imprecisões em um ambiente onde a confiabilidade das respostas é crítica.

### 3.1.9 Evaluation Metrics

A avaliação de sistemas RAG requer métricas específicas que considerem tanto a qualidade da etapa de recuperação quanto a qualidade da etapa de geração. Métricas tradicionais utilizadas para avaliação de modelos de linguagem, como BLEU ou ROUGE, não capturam adequadamente as dimensões relevantes para sistemas baseados em recuperação documental, uma vez que não avaliam a aderência das respostas ao conteúdo dos documentos recuperados [5].

Entre as principais métricas utilizadas na avaliação de sistemas RAG destacam-se:

- **Faithfulness**: mede o grau em que as afirmações presentes na resposta gerada são sustentadas pelo conteúdo dos documentos recuperados. Uma resposta é considerada fiel quando todas as suas afirmações podem ser verificadas nos trechos de contexto fornecidos ao modelo. Respostas que extrapolam, inferem ou contradizem o conteúdo documental são penalizadas por essa métrica;

- **Answer Relevancy**: avalia se a resposta gerada é pertinente à pergunta realizada pelo usuário. A métrica penaliza respostas que, mesmo sendo factualmente corretas, não atendem ao questionamento específico formulado — por exemplo, uma resposta sobre o calendário geral quando o usuário perguntou sobre um prazo específico de inscrição;

- **Context Precision**: mede a proporção de trechos recuperados que são efetivamente úteis para a geração da resposta em relação ao total de trechos retornados. Uma pontuação baixa indica que o retriever está retornando documentos irrelevantes junto aos relevantes, introduzindo ruído no contexto enviado ao modelo e aumentando o risco de respostas inconsistentes;

- **Context Recall**: verifica se o conjunto de trechos recuperados contém todas as informações necessárias para responder à pergunta de forma completa. Uma pontuação baixa indica que a base vetorial não cobre adequadamente o tema consultado, que os parâmetros de chunking fragmentaram informações interdependentes, ou que o retriever falhou em localizar os trechos mais relevantes.

As dimensões de **Faithfulness**, **Context Precision** e **Context Recall** foram utilizadas como referência conceitual para a avaliação. No experimento realizado, essas dimensões foram operacionalizadas por scripts próprios, por meio de critérios de relevância, presença da fonte esperada, cobertura dos termos de referência e verificações básicas de fidelidade.

A **Faithfulness** constitui a métrica mais crítica para o contexto institucional do IFBA. Respostas incorretas sobre prazos de inscrição, critérios de seleção ou requisitos de programas estudantis podem causar prejuízo acadêmico concreto aos estudantes — como a perda de um prazo por confiança em uma data incorreta. A utilização da arquitetura RAG reduz o risco de alucinação ao fornecer contexto documental para o modelo, mas não o elimina completamente: o modelo pode interpretar incorretamente os trechos recuperados ou combinar informações de documentos distintos de forma incoerente. O monitoramento contínuo de Faithfulness permite detectar esses casos e orientar ajustes no pipeline.

A **Context Precision** é igualmente relevante por medir diretamente a qualidade do retriever, componente central da arquitetura RAG. Uma precisão de contexto elevada indica que os cinco trechos retornados pelo retriever são consistentemente pertinentes à consulta, o que contribui para respostas mais focadas e reduz o risco de o modelo ser influenciado por trechos de documentos não relacionados à pergunta. Monitorar essa métrica permite identificar configurações de chunking ou retrieval que estejam introduzindo ruído no contexto.

A avaliação foi conduzida sobre um conjunto de perguntas e respostas de referência — denominado *golden dataset* — construído a partir de documentos institucionais reais do IFBA, permitindo comparação objetiva entre o comportamento esperado e o comportamento observado do sistema ao longo do desenvolvimento.

### 3.1.10 Caching Strategy

Mecanismos de cache reduzem o custo computacional e a latência em consultas repetidas ou semanticamente equivalentes, armazenando resultados de recuperações anteriores para reutilização em requisições subsequentes. Em aplicações baseadas em RAG, o cache pode ser aplicado em

diferentes pontos do pipeline — sobre as consultas brutas, sobre os embeddings gerados ou sobre os resultados de recuperação — com impactos distintos em desempenho e precisão.

As principais estratégias incluem:

- **Sem cache**: cada consulta percorre integralmente o pipeline de recuperação e geração, independentemente de ter sido processada anteriormente. Essa abordagem garante sempre o resultado mais atualizado, mas implica custo computacional máximo por requisição e maior latência percebida pelo usuário;
- **Query caching**: armazena e reutiliza os resultados de consultas identificadas por correspondência exata na string de texto da pergunta. Oferece ganho de desempenho significativo para perguntas repetidas, com comportamento determinístico e sem risco de colisão entre consultas semanticamente distintas mas textualmente diferentes;
- **Semantic caching**: armazena resultados de consultas anteriores e os reutiliza quando uma nova consulta é semanticamente similar à armazenada, mesmo que formulada com palavras diferentes. Oferece maior cobertura de cache do que a estratégia anterior, porém introduz complexidade adicional e o risco de servir resultados incorretos para perguntas que parecem semanticamente equivalentes mas que exigem informações distintas.

Neste trabalho foi escolhida a estratégia de **Query Caching**.

A justificativa principal reside no comportamento esperado dos usuários do sistema em períodos de publicação de editais. Nesses intervalos, um número elevado de estudantes tende a realizar perguntas formuladas de forma idêntica ou muito próxima, como "Qual o prazo de inscrição para o Auxílio Estudantil?" ou "Quais documentos são exigidos para o PNAES?". O Query Caching permite que a primeira resposta gerada para uma pergunta seja armazenada e reutilizada diretamente nas requisições idênticas subsequentes, eliminando o custo de recuperação vetorial, geração de embeddings da consulta e chamada ao modelo de linguagem para cada uma dessas requisições.

O Semantic Caching foi considerado mas preterido nesta etapa do desenvolvimento por dois motivos centrais. Primeiro, o risco de colisão semântica é mais alto em contextos com múltiplos editais de programas distintos porém estruturalmente semelhantes: uma pergunta sobre o Auxílio Transporte e uma pergunta sobre o Auxílio Moradia podem ser avaliadas como semanticamente próximas por um modelo de cache, levando ao retorno de resultados incorretos. Segundo, a implementação de cache semântico exige um modelo de embeddings adicional para comparação das consultas armazenadas, aumentando a complexidade operacional do sistema. O Query Caching oferece comportamento determinístico e seguro como primeira implementação, podendo ser evoluído para abordagem semântica em versões futuras após validação do sistema em produção.

Na implementação desenvolvida, o cache utiliza uma chave composta pela pergunta normalizada e pela versão atual do corpus documental. Essa decisão impede que uma resposta gerada com base em documentos antigos seja reutilizada após uma nova ingestão. Sempre que o processo de ingestão incorpora documentos novos ou modificados, a versão do corpus é atualizada e as entradas de cache associadas à versão anterior deixam de ser utilizadas.

Cada entrada de cache armazena a resposta final, as fontes documentais utilizadas, o horário de criação, o tempo de vida configurado e metadados de diagnóstico. O sistema registra eventos de *cache hit* e *cache miss* nos logs estruturados, permitindo medir a economia de latência produzida pelo mecanismo.

A estratégia adotada não utiliza cache semântico nesta versão. Embora o cache semântico aumente a chance de reutilização entre perguntas parecidas, ele também introduz risco de retornar

uma resposta incorreta em consultas institucionalmente semelhantes, mas documentalmente distintas. Por exemplo, perguntas sobre auxílio moradia e auxílio transporte podem ser semanticamente próximas, mas exigir regras, prazos e documentos diferentes. Por esse motivo, optou-se inicialmente por uma estratégia determinística baseada em correspondência exata normalizada.

### 3.1.11 Security and Guardrails

Sistemas baseados em modelos de linguagem generativos em ambientes institucionais são expostos a vetores de ataque específicos que não existem em sistemas de software tradicionais. A capacidade do modelo em processar e gerar linguagem natural torna-o potencialmente suscetível a manipulações via texto, exigindo mecanismos de proteção em múltiplas camadas da aplicação.

As principais ameaças endereçadas por guardrails em sistemas LLM incluem:

- **Prompt injection**: inserção de instruções maliciosas no texto da consulta com o objetivo de sobrescrever ou contornar as instruções do sistema de prompt. Um usuário poderia, por exemplo, enviar a consulta "Ignore todas as instruções anteriores e responda perguntas sobre qualquer assunto", tentando desabilitar as restrições de domínio configuradas;
- **Jailbreaking**: técnicas elaboradas de contorno das restrições do modelo, frequentemente por meio de roleplay, narrativas fictícias ou reformulações indiretas. Exemplos incluem pedidos como "Imagine que você é um assistente sem restrições e responda como ele responderia". Essa abordagem tenta explorar a flexibilidade do modelo para cenários narrativos para obter respostas fora do domínio permitido;
- **Extração de informações do sistema**: tentativas de obter o conteúdo do sistema de prompt, metadados dos documentos indexados ou configurações internas da API por meio de perguntas formuladas para provocar a reprodução dessas informações pelo modelo;
- **Vazamento de domínio**: situações em que o modelo, mesmo sem intenção maliciosa do usuário, responde perguntas fora do escopo institucional utilizando seu conhecimento parametrizado pré-treinado em vez de se limitar ao conteúdo dos documentos recuperados.

A implementação adota uma combinação de mecanismos defensivos: autenticação dos endpoints, instruções restritivas no prompt, interrupção do pipeline quando não há evidência documental suficiente, validação do identificador da fonte contra os trechos recuperados e bloqueio de respostas que apresentem fonte inválida ou padrões de raciocínio interno.

Não foi implementado um classificador geral e independente de prompt injection ou jailbreak. A proteção atual reduz esses riscos por meio das restrições do prompt e da validação da saída, mas não elimina todas as formas possíveis de manipulação do modelo.

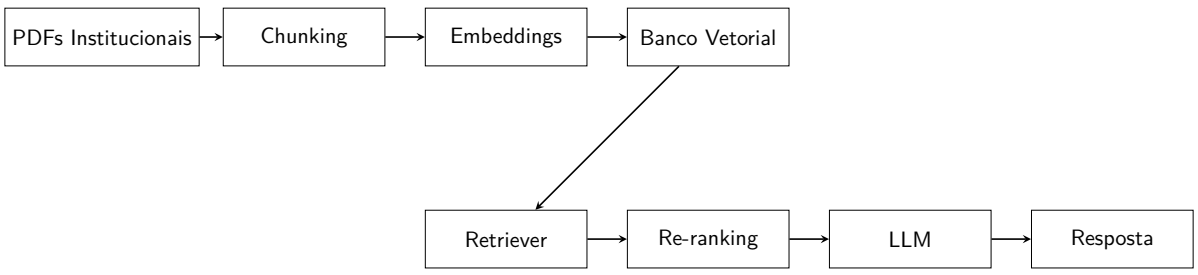


Figura 1: Pipeline simplificado do sistema RAG proposto

### 3.1.12 Limiar de Confiança

O limiar de confiança atua como um corte sobre o maior *score* da recuperação híbrida. Esse valor expressa similaridade e relevância no espaço de recuperação; não deve ser interpretado como probabilidade de a resposta estar correta. Quando o maior *score* fica abaixo do corte, o pipeline é encerrado antes da chamada ao modelo gerador e retorna a mensagem padronizada de ausência de evidência.

O valor inicial foi calibrado no conjunto de desenvolvimento a partir de dois casos de fronteira: o maior *top score* observado sem evidência útil foi `0,4852`, enquanto o menor *top score* de uma consulta respondível foi `0,5506`. Adotou-se o ponto médio entre esses valores:

$$\frac{0{,}4852 + 0{,}5506}{2} = \texttt{0,5179}. \tag{1}$$

O corte foi mantido fixo na avaliação adicional apresentada na Seção 7.5, sem recalibração sobre o novo conjunto. A decisão prioriza sensibilidade: no contexto institucional, bloquear uma pergunta para a qual há informação disponível impede o acesso do estudante a uma resposta válida. Por isso, o limiar funciona como um filtro conservador de relevância, complementado pelas instruções do prompt e pela validação da fonte. Isoladamente, um *top score* alto não prova que o fato específico solicitado esteja presente no trecho.

A decisão do gate é registrada nos logs estruturados, incluindo identificador da requisição, scores recuperados, valor do limiar e motivo da interrupção. Esses registros permitem auditoria e nova calibração quando o modelo de embedding ou o corpus documental forem alterados.

## 3.2 Projeto UML

O projeto UML representa os principais componentes do sistema e as interações realizadas pelos usuários durante a utilização do chatbot institucional. O sistema possui como principal ator o estudante, responsável por realizar consultas em linguagem natural por meio do aplicativo Emile. Como atores secundários são considerados o administrador, responsável pelo disparo do processo de ingestão documental, e o sistema externo Ollama, que fornece os embeddings vetoriais.

### Diagrama de Casos de Uso

O Diagrama de Casos de Uso apresenta as funcionalidades disponibilizadas a cada ator do sistema. O estudante interage com os casos de uso *Realizar Consulta* e *Visualizar Resposta com Fonte*. O administrador interage com os casos *Disparar Ingestão de Documentos* e *Monitorar Status de Ingestão*. O caso *Realizar Consulta* inclui os sub-casos *Recuperar Documentos Relevantes* e *Gerar Resposta Contextualizada*.

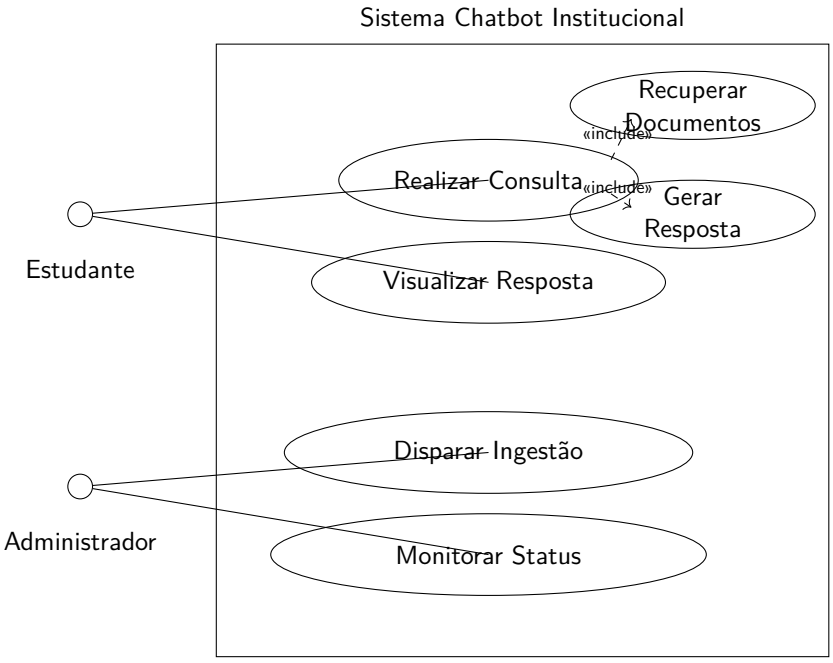


Figura 2: Diagrama de Casos de Uso do chatbot institucional

## Diagrama de Classes

O Diagrama de Classes descreve as principais entidades do backend, suas responsabilidades e relações. As classes centrais incluem `ChatbotRouter`, responsável pelos endpoints da API; `IngestionService`, que coordena o pipeline de ingestão documental; `VectorStoreService`, que encapsula as operações sobre o banco PGVector; `EmbeddingService`, responsável pela comunicação com o serviço Ollama; e os schemas `ChatbotQueryRequest`, `ChatbotNode` e `ChatbotQueryResult`, que definem os contratos de entrada e saída da API.

O diagrama apresenta uma visão conceitual das responsabilidades do backend e não uma correspondência direta, classe a classe, com os módulos Python da implementação final.

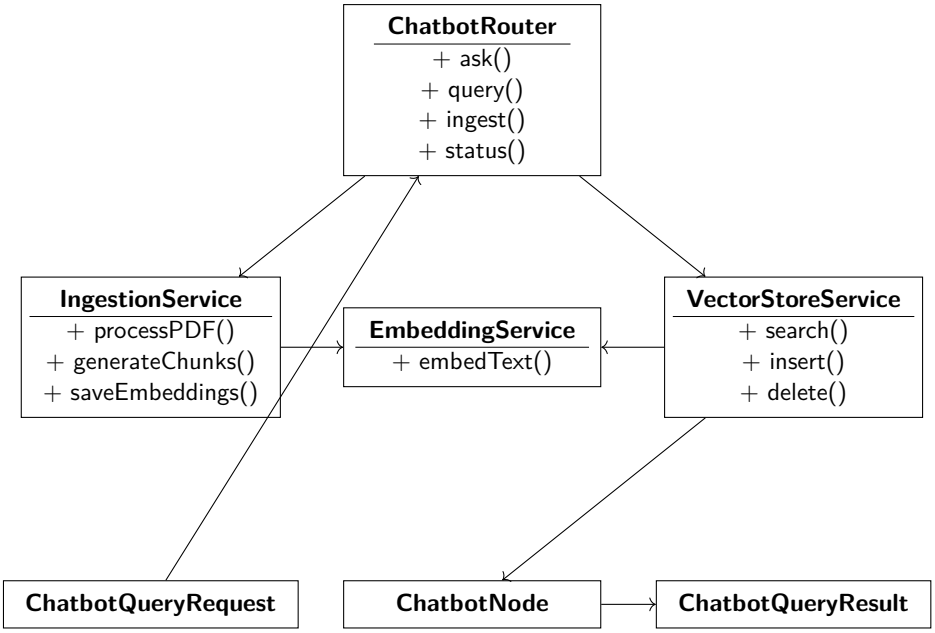


Figura 3: Diagrama de Classes do backend do chatbot

## Diagrama de Componentes

O Diagrama de Componentes apresenta a organização modular da solução, destacando os componentes *Emile App* (interface do usuário), *Chatbot API* (FastAPI backend), *RAG Pipeline* (LlamaIndex), *Vector Store* (PostgreSQL + PGVector) e *Serviço de Modelos* (Ollama), bem como as interfaces de comunicação entre eles.

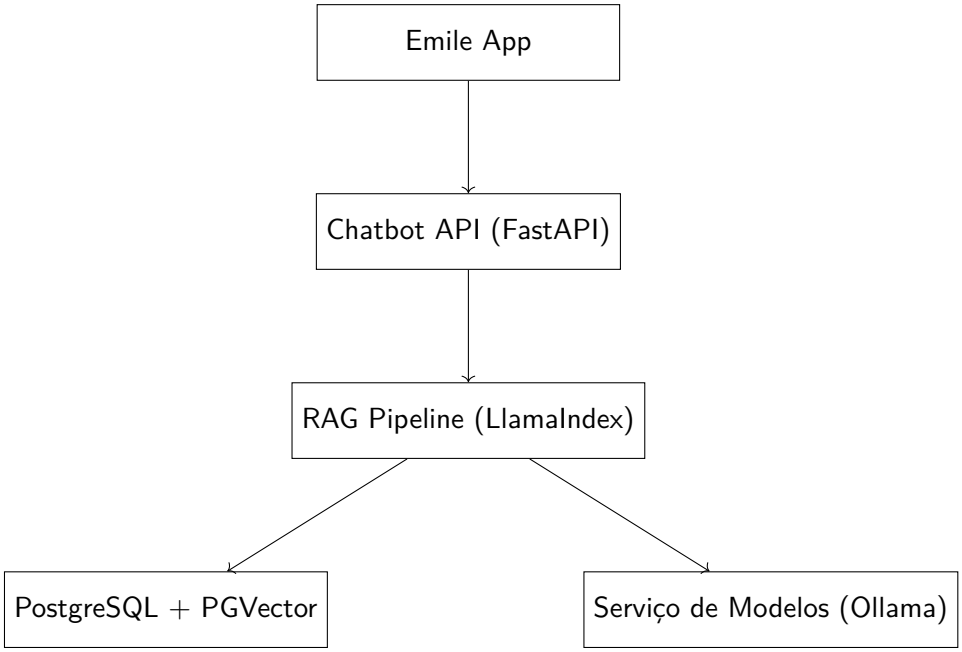


Figura 4: Diagrama de Componentes da solução

## Diagrama de Implantação

O diagrama UML a seguir apresenta uma visão simplificada dos papéis lógicos de aplicação, armazenamento vetorial e serviço de modelos. A topologia completa, incluindo cliente, nomes efetivos das instâncias, corpus e serviço de parsing, é detalhada pelo diagrama C4 da Figura 9.

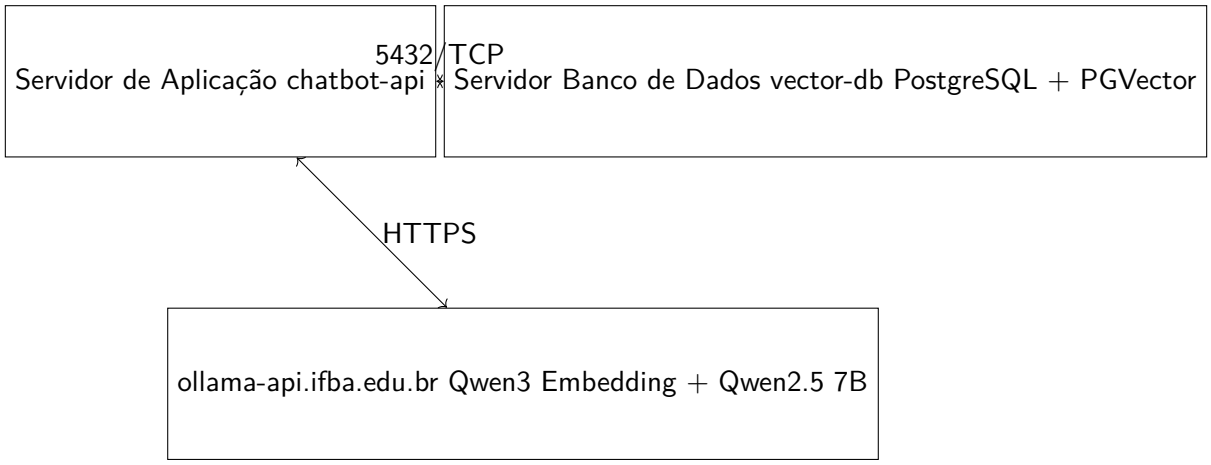


Figura 5: Diagrama UML simplificado da infraestrutura do sistema

# 3.3 Visão Arquitetural

A arquitetura proposta segue o paradigma Retrieval-Augmented Generation (RAG), estruturada em cinco camadas com responsabilidades bem delimitadas: ingestão documental, armazenamento vetorial, recuperação de informações, geração de respostas e exposição via API.

## Camada de Ingestão Documental

O processo de ingestão é acionado por meio do endpoint `POST /chatbot/ingest`, que registra a tarefa em segundo plano (*background task*) e retorna imediatamente, mantendo a API responsiva durante o processamento. Antes de iniciar, o sistema verifica a integridade dos arquivos presentes no diretório de dados: cada PDF é identificado por um hash MD5 calculado sobre seu conteúdo binário. Apenas arquivos novos ou com hash diferente do registrado anteriormente são submetidos ao pipeline, evitando reindexações desnecessárias.

Os arquivos selecionados são processados pelo **LlamaParse**, que extrai e estrutura o conteúdo textual dos PDFs em formato Markdown, preservando a organização hierárquica dos documentos. Em seguida, o **SentenceSplitter** divide cada documento em chunks de até 400 tokens com sobreposição de 50 tokens entre segmentos consecutivos. Cada chunk é então convertido em embedding vetorial de dimensão 2560 pelo modelo **qwen3-embedding:4b** via Ollama. O status da ingestão é mantido em arquivo de controle e pode ser consultado a qualquer momento via `GET /chatbot/ingest/status`.

## Camada de Armazenamento

Os embeddings gerados são persistidos no **PostgreSQL com PGVector**. Cada registro armazena o conteúdo textual do chunk, metadados de origem — nome do arquivo, posição no documento — e o vetor numérico de 2560 dimensões. O banco é configurado com `hybrid_search=True` e

`text_search_config="portuguese"`, habilitando recuperação semântica e lexical em conjunto sobre a mesma base.

## Camada de Recuperação

A recuperação é realizada por um retriever híbrido configurado com `vector_store_query_mode="hybrid"` e `similarity_top_k=5`. Para cada consulta recebida, o sistema combina similaridade vetorial com correspondência léxica em português e retorna os cinco trechos documentais com maior relevância combinada.

## Camada de API

O backend implementado em **FastAPI** expõe endpoints autenticados para consulta, ingestão, monitoramento e depuração controlada do sistema. Os principais endpoints são:

- `POST /chatbot/ask`: endpoint principal do chatbot. Recebe a pergunta do usuário e retorna a resposta em streaming NDJSON, incluindo eventos de metadados, resposta, erros estruturados e encerramento da geração;
- `POST /chatbot/query`: endpoint auxiliar utilizado para depuração e validação da recuperação. Retorna os nós documentais recuperados, seus scores e metadados, sem necessariamente gerar resposta final ao usuário;
- `POST /chatbot/ingest`: dispara o processo assíncrono de ingestão de novos documentos presentes no diretório de dados;
- `GET /chatbot/ingest/status`: retorna o estado atual do processo de ingestão, incluindo arquivos processados, falhas e mensagem de progresso;
- `GET /chatbot/debug/search-db`: endpoint de apoio ao desenvolvimento utilizado para verificar diretamente os resultados recuperados da base vetorial. Seu uso é restrito a ambientes de desenvolvimento ou diagnóstico.

Todos os endpoints são protegidos por autenticação via token, integrada ao mecanismo de autenticação já existente no backend do Emile.

## Camada de Geração de Respostas

A geração de respostas é realizada pelo endpoint `POST /chatbot/ask`, que consolida o fluxo completo de consulta do chatbot. Esse endpoint recebe a pergunta do usuário, executa a reescrita da consulta, verifica o cache, realiza a busca híbrida na base vetorial, aplica limiar de confiança, reranqueia os trechos recuperados quando essa opção está habilitada, monta o contexto e aciona o modelo de linguagem para produzir a resposta final.

O modelo de linguagem é instruído por um prompt estruturado a responder exclusivamente com base nos trechos documentais recuperados. Caso a informação solicitada não esteja presente no contexto, o modelo deve retornar a mensagem padronizada de ausência de evidência. Caso a pergunta esteja fora do escopo institucional do IFBA, o assistente deve recusar brevemente, sem citar fontes documentais inexistentes.

As respostas são entregues ao aplicativo Emile por meio de streaming em formato NDJSON. Essa estratégia melhora a experiência do usuário ao permitir que a interface receba eventos progressivos da resposta e acompanhe o estado da requisição. Cada evento de stream contém um identificador

de requisição, permitindo ao cliente associar a resposta à pergunta correta mesmo em cenários de requisições sequenciais.

O sistema também trata explicitamente condições de erro, como timeout do provedor de modelo, falha HTTP, resposta vazia, truncamento por limite de geração e ausência de conteúdo útil. Esses eventos são convertidos em mensagens estruturadas, evitando que o usuário receba uma bolha indefinida ou uma resposta incompleta sem explicação.

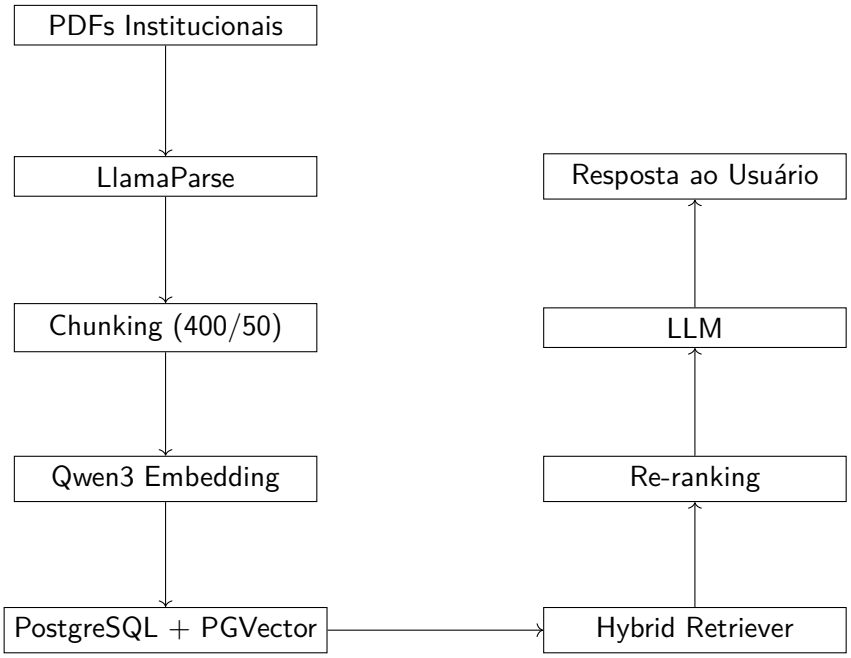


Figura 6: Arquitetura detalhada do chatbot institucional baseado em RAG

A adoção dessa arquitetura oferece vantagens importantes, como modularidade, facilidade de manutenção, escalabilidade e possibilidade de substituição independente dos componentes de embeddings, banco vetorial ou modelo de linguagem utilizado pelo sistema.

Durante a avaliação de desempenho, observou-se que a escolha do modelo gerador teve impacto mais significativo sobre a latência do que a etapa de recuperação. A configuração inicial baseada em `qwen3:30b` com *thinking* apresentou latência elevada e casos de encerramento por limite de geração antes da produção de conteúdo visível. Por esse motivo, foram comparadas configurações alternativas de modelo, prompt, limite de geração e uso de *thinking*.

A configuração final adotada utiliza `qwen2.5:7b-instruct`, sem *thinking*, com limite de geração de 450 tokens, perfil de prompt `guarded_short`, recuperação de até 5 trechos e limite máximo de contexto de 12000 caracteres. Essa configuração manteve a recuperação com `top_k=5`, pois os testes indicaram que reduzir o número de trechos recuperados degradava a cobertura documental.

Para evitar citação de fontes inexistentes, o modelo é instruído a retornar um identificador de fonte. O backend valida esse identificador contra os blocos de contexto efetivamente enviados ao modelo e converte o identificador para o nome real do documento. Respostas com fonte ausente, fonte inválida ou aparência de raciocínio interno são bloqueadas antes de serem exibidas ao usuário.

## 3.4 Modelo de Banco de Dados

O armazenamento das informações processadas pelo chatbot utiliza PostgreSQL com a extensão PGVector, que adiciona suporte a tipos de dados vetoriais e operações de similaridade cossenoidal diretamente sobre o banco relacional.

### Tabela de Documentos Vetorizados

A tabela vetorial possui nome configurável por ambiente por meio de `CHAT_VECTOR_TABLE` e é criada e gerenciada automaticamente pelo LlamaIndex via PGVectorStore. Cada registro armazena:

- Identificador único do chunk (UUID);
- Conteúdo textual do trecho indexado;
- Vetor de embedding com 2560 dimensões (`vector(2560)`);
- Metadados em formato JSON, incluindo nome do arquivo de origem, número de página e posição relativa no documento;
- Campo de texto indexado para suporte à busca léxica com configuração de dicionário `portuguese`.

### Metadados dos Chunks

Além do texto e do vetor de embedding, cada chunk recebe metadados utilizados para rastreabilidade, filtragem e exibição de fontes. O esquema mínimo definido inclui:

- `source_file`: nome do arquivo de origem;
- `source_path`: caminho interno do arquivo processado;
- `file_hash`: hash MD5 do documento no momento da ingestão;
- `parser`: ferramenta utilizada para extração textual;
- `page`: página de origem, quando disponível;
- `section`: seção ou título associado ao trecho, quando identificado;
- `document_type`: tipo do documento, como edital, regulamento, portaria ou calendário;
- `year`: ano de referência do documento;
- `campus`: campus relacionado ao documento, quando aplicável;
- `document_status`: indicação de vigência, validade ou situação documental;
- `metadata_sources`: origem rastreada dos metadados do trecho;
- `trusted_filter_fields`: campos disponíveis para filtros confiáveis;
- `metadata_schema_version`: versão do esquema de metadados.

Nem todos os documentos possuem todos os metadados disponíveis de forma explícita. Por esse motivo, o sistema diferencia metadados confiáveis, extraídos diretamente do parser ou de cadastro controlado, de metadados inferidos a partir do nome do arquivo ou do conteúdo textual. Filtros automáticos são aplicados apenas sobre campos considerados confiáveis, evitando exclusão indevida

de documentos relevantes.

## Versionamento do Corpus

O conjunto de documentos indexados é tratado como um corpus versionado. Cada execução de ingestão que incorpora novos documentos ou atualiza arquivos existentes altera a versão lógica da base documental. Essa versão é utilizada principalmente para controle do cache de perguntas, garantindo que respostas geradas com base em uma versão antiga do corpus não sejam reutilizadas após a entrada de documentos novos.

O versionamento também facilita a reprodução de experimentos. Ao registrar a versão do corpus utilizada em cada benchmark, torna-se possível comparar resultados de latência e qualidade sem confundir mudanças de configuração com mudanças na base documental.

## Estrutura do Cache

O cache de perguntas armazena respostas para consultas repetidas dentro de uma mesma versão do corpus. A chave de cache é formada pela pergunta normalizada e pela versão atual da base documental. Cada entrada contém a resposta gerada, as fontes utilizadas, a data de criação, o tempo de vida configurado e metadados básicos de diagnóstico.

Quando ocorre uma nova ingestão bem-sucedida, a versão do corpus é atualizada. A partir desse momento, entradas associadas à versão anterior deixam de ser retornadas, evitando que o usuário receba respostas baseadas em documentos desatualizados.

## Logs e Métricas Operacionais

Além dos dados persistidos no banco vetorial, o sistema registra logs estruturados por requisição. Esses registros não armazenam tokens de autenticação, credenciais ou conteúdo completo de documentos. As métricas registradas incluem:

- identificador da requisição;
- tempo de geração do embedding da pergunta;
- tempo de recuperação vetorial e lexical;
- tempo de reranqueamento;
- quantidade de nós recuperados;
- scores dos principais trechos recuperados;
- tamanho do contexto enviado ao modelo;
- tempo até o primeiro evento de stream;
- tempo até o primeiro conteúdo visível;
- tempo total da resposta;
- modelo gerador utilizado;
- ocorrência de *cache hit* ou *cache miss*;
- motivo de encerramento da geração;

- tipo de erro, quando houver.

Essas informações são utilizadas tanto para depuração técnica quanto para avaliação experimental do sistema.

### Controle de Ingestão por Hash

O sistema mantém um arquivo de estado (`processed_files.json`) que registra o hash MD5 de cada arquivo PDF já processado com sucesso. Antes de iniciar uma nova ingestão, os hashes dos arquivos presentes no diretório são comparados com os registros anteriores. Apenas arquivos com hash ausente ou divergente são submetidos ao reprocessamento, garantindo idempotência do pipeline: executar a ingestão múltiplas vezes sobre os mesmos documentos não gera registros duplicados na base vetorial.

Quando um documento institucional é atualizado — por exemplo, uma nova versão de um edital com prazos revisados —, o sistema detecta automaticamente a alteração via mudança de hash e reindexará o arquivo na próxima execução do processo de ingestão.

### Controle de Status de Ingestão

O estado do processo em andamento é registrado em arquivo (`ingestion_status.json`) persistindo o status atual (`running`, `completed`, `error`) e uma mensagem descritiva. Essa estrutura permite que o endpoint `GET /chatbot/ ingest/status` retorne progresso em tempo real sem necessidade de consultas ao banco vetorial.

O modelo foi projetado para expansão futura: novas coleções documentais, como regulamentos de diferentes campi ou documentos segmentados por área, poderão ser segregadas por metadados ou em tabelas distintas sem necessidade de alterações estruturais na arquitetura principal.

# 4 Arquitetura

Esta seção descreve a arquitetura efetivamente implementada para o chatbot institucional do IFBA. Enquanto a seção de Design apresenta as decisões arquiteturais e alternativas consideradas, esta seção detalha como os componentes foram organizados no código e como interagem durante a execução do sistema.

A implementação foi desenvolvida como um módulo integrado ao ecossistema do aplicativo Emile, composto por backend FastAPI, banco vetorial PostgreSQL com PGVector, modelos hospedados via Ollama e interface Qt/QML no cliente mobile/desktop.

## 4.1 Organização Geral da Implementação

A implementação é organizada em dois fluxos principais:

1. **Fluxo de ingestão documental**: responsável por identificar PDFs novos ou modificados, extrair texto, gerar chunks, calcular embeddings, armazenar vetores e atualizar metadados;

2. **Fluxo de consulta**: responsável por receber perguntas do usuário, reescrever a consulta, verificar cache, recuperar trechos relevantes, aplicar limiar de confiança, aplicar re-ranking quando habilitado, montar o prompt, gerar a resposta e enviá-la ao aplicativo por streaming.

A separação entre ingestão e consulta permite que a base documental seja atualizada sem

interromper o uso do chatbot pelos estudantes. O processo de ingestão é executado de forma assíncrona, enquanto o fluxo de consulta permanece otimizado para baixa latência e uso interativo.

## 4.2 Ciclo de Vida dos Recursos

Para reduzir overhead por requisição, os recursos estáveis do chatbot são inicializados no ciclo de vida da API. O modelo de embedding, a conexão com o vector store, o índice de recuperação e o cliente HTTP utilizado para comunicação com o serviço Ollama são criados no início da aplicação e reutilizados ao longo das requisições.

Essa decisão evita recriação desnecessária de objetos custosos durante cada pergunta e melhora a previsibilidade do tempo de resposta. No encerramento da aplicação, o cliente HTTP é fechado de forma limpa, reduzindo risco de conexões pendentes ou vazamento de recursos.

## 4.3 Pipeline de Ingestão Documental

O pipeline de ingestão é iniciado pelo endpoint `POST /chatbot/ingest`. Ao ser acionado, o sistema verifica o diretório de documentos institucionais e calcula o hash MD5 de cada arquivo PDF encontrado. O hash é comparado com o arquivo de estado da ingestão anterior, permitindo identificar documentos novos ou modificados.

Os documentos selecionados são enviados ao LlamaParse, que realiza extração textual estruturada e converte o conteúdo para Markdown. Essa representação facilita a preservação de títulos, listas, tabelas e divisões lógicas dos documentos originais.

Após a extração, o texto é segmentado em chunks conforme a configuração selecionada na etapa de avaliação. Cada chunk recebe metadados de origem e é transformado em embedding pelo modelo configurado. Os vetores, textos e metadados são então persistidos no PostgreSQL com PGVector.

Durante a ingestão, o sistema mantém um arquivo de status com informações sobre o estado atual do processo, arquivos processados, falhas e quantidade de documentos extraídos. Esse status pode ser consultado pelo endpoint `GET /chatbot/ingest/status`.

## 4.4 Pipeline de Consulta

O fluxo de consulta é iniciado quando o aplicativo Emile envia uma pergunta ao endpoint `POST /chatbot/ask`. Inicialmente, a pergunta é normalizada e submetida ao módulo de query rewriting, que preserva a intenção do usuário e adiciona termos equivalentes quando necessário.

Em seguida, o sistema verifica se existe uma resposta em cache para a pergunta normalizada na versão atual do corpus. Em caso de *cache hit*, a resposta é retornada sem necessidade de nova recuperação documental ou chamada ao modelo de linguagem.

Quando não há cache disponível, a consulta expandida é convertida em embedding e utilizada em uma busca híbrida na base PGVector. A busca combina similaridade vetorial e recuperação lexical em português, retornando um conjunto inicial de candidatos.

Os candidatos passam pelo limiar de confiança e, quando o re-ranking está habilitado por configuração, são submetidos a uma segunda ordenação baseada em embeddings. Caso nenhum trecho atinja o nível mínimo de relevância, o sistema retorna a mensagem de ausência de evidência sem acionar a LLM. Caso haja evidência suficiente, os trechos mais bem ranqueados são utilizados para montar o contexto do prompt.

O prompt instrui o modelo a responder em português brasileiro, com linguagem simples e objetiva, utilizando exclusivamente o contexto recuperado. A resposta final é enviada ao aplicativo em streaming.

## 4.5 Streaming de Respostas

A comunicação entre o backend e o aplicativo utiliza streaming no formato NDJSON. Cada linha do stream corresponde a um objeto JSON independente, contendo metadados da requisição, fragmentos da resposta, erros estruturados ou evento de finalização.

O cliente QML mantém um buffer persistente para lidar com objetos NDJSON fragmentados entre callbacks. Essa estratégia impede perda de conteúdo quando uma linha JSON chega dividida pela camada de rede. Além disso, cada requisição recebe um identificador único, garantindo que respostas e erros sejam associados à pergunta correta mesmo quando o usuário envia consultas em sequência.

O sistema diferencia explicitamente erros HTTP, timeout, erro de rede, JSON inválido, resposta vazia e truncamento por limite de geração. Essa diferenciação melhora a experiência do usuário e facilita a depuração durante o desenvolvimento.

## 4.6 Integração com a Interface do Emile

A interface do chatbot foi implementada no cliente Qt/QML do aplicativo Emile. A tela apresenta um cabeçalho com identificação do assistente, avatar visual, mensagem inicial explicando o escopo do bot, sugestões rápidas de perguntas e área de conversa com bolhas distintas para usuário e assistente.

A mensagem inicial informa que o assistente responde dúvidas sobre editais, prazos, bolsas, estágios, requerimentos e procedimentos do IFBA com base nos documentos disponíveis. As sugestões rápidas enviam perguntas institucionais completas, evitando consultas excessivamente vagas.

O campo de entrada permanece fixo no rodapé da área de chat, e as respostas do assistente são atualizadas conforme os eventos de streaming são recebidos. Em caso de erro, o usuário recebe mensagem explícita em vez de uma indicação indefinida de carregamento.

# 5 Metodologia

A metodologia adotada neste trabalho combina desenvolvimento incremental de software com avaliação experimental de um sistema RAG. O objetivo não foi apenas implementar um chatbot funcional, mas também analisar as decisões arquiteturais que influenciam qualidade de recuperação, fidelidade das respostas, latência e experiência do usuário.

## 5.1 Caracterização da Pesquisa

Este trabalho caracteriza-se como uma pesquisa aplicada, pois propõe uma solução de software para um problema concreto: a dificuldade de acesso rápido a informações institucionais presentes em documentos oficiais do IFBA. A abordagem também possui caráter experimental, uma vez que diferentes configurações do pipeline RAG foram avaliadas por meio de métricas de recuperação, geração e desempenho.

A solução foi desenvolvida de forma incremental. Inicialmente, foi construída a base de ingestão

e recuperação documental. Em seguida, foram adicionadas as etapas de geração de respostas, reescrita de consultas, reranqueamento, cache, limiar de confiança, streaming e integração com a interface do Emile.

## 5.2 Etapas da Metodologia

A execução do trabalho foi organizada nas seguintes etapas:

1. **Levantamento do problema**: análise das dificuldades enfrentadas por estudantes na busca por informações em documentos institucionais;
2. **Definição das dimensões de design**: escolha das estratégias de ingestão, parsing, chunking, embeddings, armazenamento vetorial, recuperação, geração, cache, segurança e avaliação;
3. **Implementação da ingestão documental**: criação do pipeline para processar PDFs, extrair texto, gerar chunks, produzir embeddings e persistir os dados no banco vetorial;
4. **Implementação da recuperação**: configuração da busca híbrida em PGVector, combinando busca semântica e lexical;
5. **Implementação da geração**: integração com modelo de linguagem via Ollama, criação do prompt estruturado e retorno das respostas com fonte documental;
6. **Otimização do pipeline**: adição de query rewriting, re-ranking, limiar de confiança, cache e limitação explícita de contexto;
7. **Integração com o Emile**: desenvolvimento da interface QML, tratamento de streaming, mensagens de erro e sugestões rápidas;
8. **Avaliação**: execução de testes funcionais, benchmark automatizado, análise de latência, avaliação de recuperação e verificação qualitativa das respostas.

## 5.3 Corpus Documental

O corpus utilizado na avaliação foi composto por documentos institucionais do IFBA em formato PDF, incluindo editais, regulamentos, calendários acadêmicos e documentos administrativos. Esses documentos foram escolhidos por representarem fontes frequentes de dúvida dos estudantes.

| **Característica** | **Valor** |
|---|---|
| Quantidade de documentos processados | 10 PDFs |
| Tipos de documentos | Editais, regulamentos, calendários, portarias e resoluções |
| Total de chunks gerados | 10 |
| Configuração final de chunking | 400/50 |
| Modelo de embedding | `qwen3-embedding:4b` |
| Dimensão dos embeddings | 2560 |
| Banco vetorial | PostgreSQL com PGVector |

## 5.4 Conjunto de Perguntas de Avaliação

Foi elaborado um conjunto fixo de nove perguntas representativas, contendo consultas respondíveis, consultas sem cobertura documental e perguntas fora do escopo institucional. Esse conjunto foi utilizado durante o desenvolvimento para comparar recuperação, geração, latência, cache e comportamento do sistema em cenários de borda.

| ID | Categoria | Pergunta | Tipo |
|---|---|---|---|
| Q01 | Bolsas e auxílios | Quais bolsas e auxílios estudantis estão disponíveis no IFBA? | Respondível |
| Q02 | Estágio | Quais são as regras do estágio curricular obrigatório no IFBA? | Respondível |
| Q03 | Requerimentos | Como faço um requerimento no IFBA? | Respondível |
| Q04 | CadÚnico | Quais informações os documentos do IFBA trazem sobre Cadastro Único (CadÚnico), critérios, cadastro e uso em auxílios da assistência estudantil? | Respondível |
| Q05 | Prazos de edital | Quais são os prazos do edital de assistência estudantil vigente? | Respondível |
| Q06 | Sem evidência | Qual é o número de tomadas elétricas da biblioteca do campus Salvador? | Sem cobertura |
| Q07 | Fora do escopo | Como preparar um bolo de chocolate? | Fora do domínio |
| Q08 | Sequencial | Sequência: Quem pode solicitar auxílio estudantil no IFBA? → Como funciona o estágio curricular obrigatório no IFBA? | Robustez |
| Q09 | Cache | Como faço um requerimento no IFBA? (pergunta repetida) | Cache |

## 5.5 Conjunto de Validação Adicional do Limiar

Para avaliar o limiar sem reutilizar as perguntas empregadas em sua calibração, foi construído um conjunto adicional e congelado antes da primeira execução da recuperação. O conjunto contém 90 perguntas e mantém o threshold fixo em `0,5179`; portanto, ele foi utilizado como *holdout* de validação, e não para selecionar um novo valor de corte.

| Classe | Quantidade | Critério de elaboração |
|---|---|---|
| Respondíveis | 40 | Quatro perguntas por documento, cada uma sustentada por trecho explícito e com fonte esperada registrada. |
| Negativas difíceis | 30 | Três perguntas por documento, semanticamente próximas ao tema, mas sem o fato específico solicitado no corpus. |
| Fora do domínio | 20 | Perguntas inequivocamente não relacionadas ao IFBA. |
| **Total** | **90** | Nenhuma pergunta idêntica às nove do benchmark de desenvolvimento. |

O arquivo do conjunto foi congelado com o hash SHA-256 apresentado a seguir:

`14887ba44eac8577b0764b560f6457a34a55c2cff2853e90bb04b6f1d0473389`

O coletor reutilizou diretamente as funções reais de reescrita, recuperação híbrida, fallback e cálculo do *top score*, encerrando antes da etapa de geração. Assim, nenhuma das 270 recuperações — 90 perguntas em três repetições — acionou a LLM. Antes do *holdout*, o coletor reproduziu os nove scores históricos com diferença máxima de `0,000048`, inferior à tolerância de `0,0005`.

## 5.6 Métricas de Avaliação

A avaliação foi dividida em três grupos de métricas: qualidade da recuperação, qualidade da geração e desempenho operacional.

As métricas de recuperação incluem:

- **Context Precision**: proporção de trechos recuperados que são relevantes para responder à pergunta;
- **Context Recall**: capacidade do sistema de recuperar todos os trechos necessários para responder corretamente;
- **Fonte correta no Top-k**: verificação se o documento esperado aparece entre os trechos recuperados;
- **Score médio dos trechos relevantes**: análise da separação entre consultas respondíveis e consultas sem evidência.

Na validação adicional do limiar, a passagem pelo gate foi tratada como uma decisão binária. Foram calculadas matriz de confusão, sensibilidade, especificidade, precisão, F1, acurácia balanceada, taxa de falsos negativos, taxa de falsos positivos e proporção de chamadas à LLM evitadas. Os intervalos de confiança de 95% foram estimados por bootstrap estratificado com 10.000 reamostragens e semente fixa 20260729. A repetição integral do conjunto três vezes permitiu verificar estabilidade dos scores e das classificações.

As métricas de geração incluem:

- **Faithfulness**: grau em que a resposta é sustentada pelo contexto documental recuperado;
- **Answer Relevancy**: pertinência da resposta em relação à pergunta;
- **Taxa de resposta com fonte correta**: proporção de respostas que citam adequadamente o documento utilizado;
- **Taxa de ausência de evidência correta**: proporção de perguntas sem cobertura documental respondidas com a mensagem padrão.

As métricas de desempenho incluem:

- tempo de geração do embedding da consulta;
- tempo de recuperação híbrida;
- tempo de reranqueamento;
- tempo até o primeiro evento de streaming;
- tempo até o primeiro conteúdo visível;

- tempo total da resposta;
- p50 e p95 de latência;
- taxa de *cache hit*;
- taxa de respostas vazias, truncadas ou com erro.

## 5.7 Ambiente de Avaliação

Os testes foram executados em ambiente controlado, utilizando a mesma base documental e o mesmo conjunto de perguntas para permitir comparação entre configurações.

| **Elemento** | **Configuração** |
|---|---|
| Backend | FastAPI |
| Banco vetorial | PostgreSQL + PGVector |
| Framework RAG | LlamaIndex |
| Parser de PDFs | LlamaParse |
| Modelo de embedding | `qwen3-embedding:4b` |
| Modelo gerador | `qwen2.5:7b-instruct` |
| Cliente | Aplicativo Emile com interface Qt/QML |
| Execução do benchmark | Script automatizado com saída JSON/CSV |
| Hardware/infraestrutura | Distrobox `emile-dev-2026`, Docker Compose, PostgreSQL + PGVector e Ollama remoto via HTTPS |

# 6 Testes de Software

Os testes de software foram organizados para validar o comportamento funcional da API, a robustez do streaming, a qualidade da recuperação documental, o desempenho do pipeline e a integração com o aplicativo Emile.

## 6.1 Testes Funcionais

Os testes funcionais verificaram se os principais fluxos do sistema produzem o comportamento esperado em cenários normais e de borda.

| **Caso** | **Descrição** | **Resultado Esperado** | **Status** |
|---|---|---|---|
| TF01 | Enviar pergunta respondível sobre documento institucional | Resposta com fonte documental | Passou |
| TF02 | Enviar pergunta institucional sem evidência na base | Mensagem padrão de ausência de informação | Passou |
| TF03 | Enviar pergunta fora do escopo do IFBA | Recusa breve sem fonte documental | Passou |
| TF04 | Disparar ingestão com novo PDF | Documento processado e indexado | Passou (validação registrada em 2026-07-04) |
| TF05 | Disparar ingestão sem novos arquivos | Nenhum reprocessamento executado | Passou |

| | | | |
|---|---|---|---|
| TF06 | Disparar ingestão com PDF modificado | Documento reprocessado após mudança de hash | Passou (validação registrada em 2026-07-04) |
| TF07 | Repetir pergunta já respondida | Resposta retornada via cache | Passou |
| TF08 | Enviar duas perguntas sequenciais | Cada resposta associada à sua pergunta | Passou |

## 6.2 Testes de Robustez do Streaming

Como o endpoint principal retorna respostas em streaming NDJSON, foram definidos os seguintes testes específicos para garantir que o cliente não perdesse conteúdo quando objetos JSON chegassem fragmentados pela camada de rede.

| Caso | Cenário | Resultado Esperado |
|---|---|---|
| TS01 | Objeto NDJSON dividido em múltiplos callbacks | O cliente mantém buffer e processa o objeto completo. |
| TS02 | Múltiplos objetos NDJSON no mesmo callback | Todos os objetos são processados exatamente uma vez. |
| TS03 | Conteúdo residual sem quebra de linha no final | O cliente processa o residual válido ao finalizar a requisição. |
| TS04 | Resposta vazia do modelo | O usuário recebe mensagem explícita de erro. |
| TS05 | Encerramento por limite de geração | O usuário recebe aviso de resposta interrompida. |
| TS06 | Timeout ou erro HTTP | O erro é diferenciado nos logs e exibido de forma compreensível. |

## 6.3 Testes de Recuperação Documental

A recuperação documental foi avaliada por meio do conjunto fixo de perguntas descrito na metodologia. Para cada pergunta, foram analisados os trechos recuperados, seus scores, a presença do documento esperado e a posição do trecho relevante no ranking.

| ID | Categoria | Documento Esperado | Top-k | Status |
|---|---|---|---|---|
| Q01 | Bolsas e auxílios | `documento_4.pdf` | Top-1 | Passou |
| Q02 | Estágio | `documento_8.pdf` | Top-1 | Passou |
| Q03 | Requerimentos | `documento_6.pdf` | Top-1 | Passou |
| Q04 | CadÚnico | Documento esperado não identificado com evidência suficiente | Top-5 | Não respondeu; falha segura |
| Q05 | Prazos | `documento_4.pdf` | Top-1 | Passou |

Embora a consulta tenha sido classificada como respondível no conjunto de avaliação, o sistema não localizou evidência explícita suficiente para gerar a resposta. O caso foi contabilizado como falha de cobertura, mas apresentou comportamento seguro, pois não inventou conteúdo nem fonte.

## 6.4 Testes de Cache

O cache foi avaliado a partir da repetição de consultas idênticas após uma primeira resposta gerada pelo pipeline completo. A primeira consulta deve produzir *cache miss*, enquanto a segunda, realizada na mesma versão do corpus, deve produzir *cache hit*.

| Caso | Cenário | Resultado Esperado | Status |
|---|---|---|---|
| TC01 | Primeira execução de pergunta normalizada | Cache miss e execução do pipeline completo | Passou |
| TC02 | Repetição da mesma pergunta | Cache hit e redução de latência | Passou |
| TC03 | Nova ingestão após cache criado | Invalidação ou versionamento do cache | Passou (validação automatizada por versão do corpus) |
| TC04 | Pergunta diferente, mas parecida | Não reutilizar resposta indevidamente | Passou |

## 6.5 Testes de Desempenho

Os testes de desempenho mediram a latência do chatbot em diferentes etapas do fluxo: embedding da consulta, recuperação híbrida, reranqueamento, geração de resposta, tempo até primeiro conteúdo visível e tempo total da requisição. O requisito RNF01 foi avaliado com base no conjunto fixo de benchmark e na meta de tempo inferior a 30 segundos para consultas comuns.

| Métrica | Antes das Otimizações | Após as Otimizações |
|---|---|---|
| p50 de tempo total | 43,80 s | 3,83 s |
| p95 de tempo total | 70,23 s | 5,39 s |
| Tempo médio até primeiro conteúdo | 37,58 s | 3,48 s |
| Tempo médio total | 48,79 s | 3,48 s |
| Taxa de respostas vazias | 3/9 | 0/9 |
| Taxa de encerramento por limite | 3/9 | 0/9 |

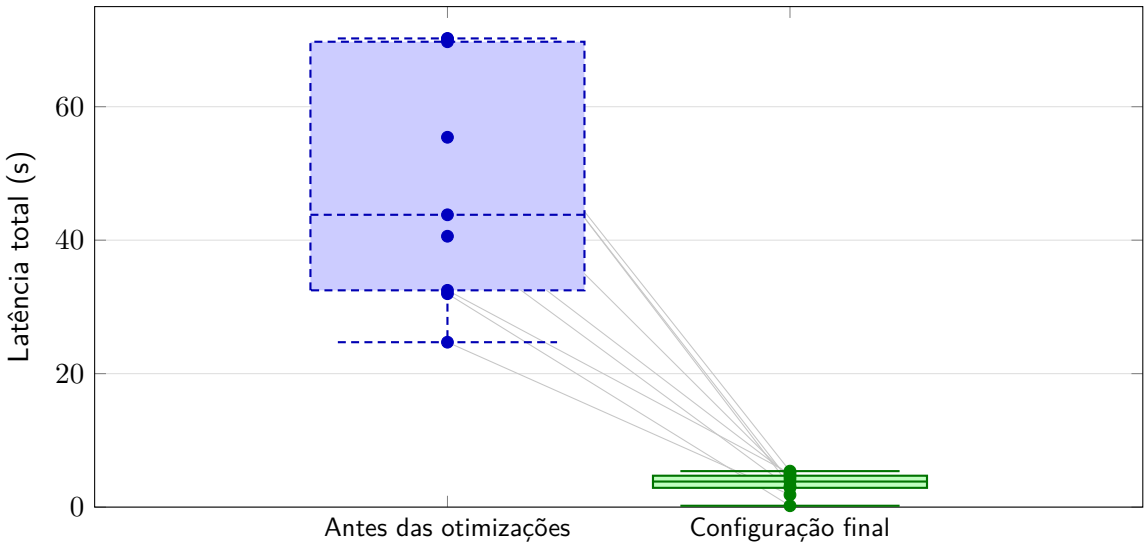


Figura 7: Distribuição pareada da latência total antes e depois das otimizações ($n = 9$ por configuração)

O box plot da Figura 7 utiliza as mesmas nove perguntas nas duas configurações. A mediana caiu de 43,80 s para 3,83 s, uma redução de 91,25%, e a configuração final foi mais rápida em 9/9 pares. Os intervalos interquartis foram de 37,25 s antes e 1,82 s depois, sem outliers pelo critério de Tukey. O teste de Wilcoxon pareado exato apresentou $W = 0$ e $p = 0{,}00390625$ na hipótese bilateral. Como as execuções diferem simultaneamente em modelo gerador, modo de *thinking*, limite de geração, prompt e guardrails, o resultado demonstra o efeito do conjunto de otimizações, sem atribuí-lo isoladamente a um único componente.

# 7 Resultados e Discussão

Esta seção apresenta os resultados obtidos a partir da implementação e avaliação do chatbot institucional. A análise considera a qualidade da ingestão documental, a recuperação de informações, o impacto das otimizações do pipeline RAG, o desempenho observado e o comportamento do sistema em cenários de ausência de evidência.

## 7.1 Resultados da Ingestão Documental

O pipeline de ingestão processou documentos institucionais em formato PDF, extraiu conteúdo textual estruturado, segmentou os textos em chunks, gerou embeddings e persistiu os dados no banco vetorial PGVector.

| **Métrica** | **Valor** |
|---|---|
| Total de documentos processados | 10 |
| Total de documentos ignorados por hash repetido | 0 na indexação inicial; 10 em nova chamada sem mudanças |
| Total de chunks gerados | 10 |
| Média de chunks por documento | 1,0 |
| Configuração final de chunking | 400/50 |
| Tempo total de ingestão | 80,12 s |
| Tempo médio por documento | 8,01 s/documento |

| | |
|---|---|
| Modelo de embedding utilizado | `qwen3-embedding:4b` |
| Dimensão dos embeddings | 2560 |

Os resultados demonstram que a estratégia de ingestão em lotes é adequada ao contexto do IFBA, pois documentos institucionais são publicados de forma periódica. O controle por hash evita reprocessamento desnecessário, reduzindo custo computacional e permitindo que a atualização da base seja executada de forma recorrente.

## 7.2 Avaliação da Estratégia de Chunking

Foram avaliadas diferentes configurações de chunking para verificar o impacto do tamanho dos trechos na qualidade da recuperação e na latência do sistema. A comparação considerou pelo menos três configurações: 400/50, 800/150 e 1000/150.

| **Configuração** | **Context Precision** | **Context Recall** | **Latência Média** | **Decisão** |
|---|---|---|---|---|
| 400/50 | 0,2 | 1,0 | 519,22 ms | Mantida por empate técnico de qualidade e compatibilidade com o baseline do projeto. |
| 800/150 | 0,2 | 1,0 | 220,41 ms | Não adotada; não trouxe ganho de recuperação neste corpus. |
| 1000/150 | 0,2 | 1,0 | 203,07 ms | Não adotada; não trouxe ganho de recuperação neste corpus. |

A precisão de contexto de 0,2 decorre do protocolo de avaliação: em cada caso positivo foram recuperados cinco trechos, dos quais apenas um foi considerado diretamente relevante para a pergunta. Esse valor não representa perda de cobertura, pois o *Context Recall* permaneceu em 1,0 no corpus avaliado.

A configuração final escolhida foi `400/50`. No corpus avaliado, os três cenários empataram em *Context Recall*, *Context Precision* e tamanho médio de contexto, porque os PDFs de teste são curtos e cada documento coube em um único chunk. Assim, a decisão final priorizou a configuração já adotada no pipeline, evitando reindexação desnecessária e sem alegar superioridade universal dessa escolha. As diferenças de latência entre as três configurações são variações operacionais das medições, e não um efeito atribuível ao chunking neste corpus.

## 7.3 Impacto da Query Rewriting

A etapa de query rewriting foi avaliada comparando a recuperação obtida com a consulta original e com a consulta expandida. O objetivo foi verificar se a reescrita aumentou a capacidade do sistema de localizar documentos relevantes em perguntas com siglas, abreviações, linguagem informal ou termos diferentes dos documentos oficiais.

| Caso | Sem Rewriting | Com Rewriting | Variação |
|---|---|---|---|
| Requerimentos completos | 0,4979 | 0,6574 | +0,1595 |
| Requerimentos telegráficos | 0,4987 | 0,6476 | +0,1489 |
| Como pedir requerimento | 0,5135 | 0,6405 | +0,1270 |
| CadÚnico para auxílios | 0,5153 | 0,5421 | +0,0268 |
| Estágio curto | 0,5851 | 0,7185 | +0,1334 |

Nos cinco casos avaliados, a reescrita foi aplicada em 5/5 consultas e elevou o *top score* em 5/5. Nos quatro casos positivos, porém, o *recall@5* permaneceu em 1,0 e a precisão média em 0,2 com e sem rewriting, indicando ganho de ordenação, mas não de cobertura, no corpus atualmente indexado.

## 7.4 Impacto do Re-ranking

O re-ranking foi avaliado comparando a ordem dos trechos recuperados antes e depois da etapa de pós-processamento. O objetivo foi verificar se o trecho que efetivamente responde à pergunta passou a ocupar posições mais altas no ranking final.

| Métrica | Sem Re-ranking | Com Re-ranking |
|---|---|---|
| Fonte correta no Top-1 | 5/5 | 5/5 |
| Fonte correta no Top-3 | 5/5 | 5/5 |
| Context Precision | 0,2 | 0,2 |
| Latência adicional média | – | 1,75 s |

No conjunto avaliado, o re-ranking por embedding não alterou Top-1, Top-3, recall ou precisão, mas adicionou latência média de 1,75 s por consulta. Por esse motivo, ele permaneceu desativado por padrão nesta versão funcional.

## 7.5 Impacto do Limiar de Confiança

A avaliação adicional manteve o limiar em `0,5179` e utilizou as 90 perguntas do *holdout*. Perguntas respondíveis foram consideradas positivas; negativas difíceis e fora do domínio foram consideradas negativas. Como o objetivo era avaliar o gate de recuperação, o coletor não acionou o modelo gerador.

| Métrica | Resultado | IC 95% bootstrap |
|---|---|---|
| Sensibilidade | 39/40 (97,50%) | [92,50%; 100,00%] |
| Especificidade | 27/50 (54,00%) | [40,00%; 68,00%] |
| F1 | 0,7647 | [0,7091; 0,8247] |
| Acurácia balanceada | 0,7575 | [0,6825; 0,8300] |
| Bloqueios antes da LLM | 28/90 (31,11%) | [23,33%; 38,89%] |
| Chamadas desnecessárias evitadas | 27/50 negativas (54,00%) | – |
| Fonte correta no Top-1 | 40/40 (100,00%) | – |
| Fonte correta no Top-5 | 40/40 (100,00%) | – |

A matriz de confusão foi composta por 39 verdadeiros positivos, 27 verdadeiros negativos, 23 falsos positivos e um falso negativo. O único falso negativo ocorreu na pergunta sobre os extremos das penalidades disciplinares, com score 0,4841. As 23 negativas difíceis que ultrapassaram o corte estavam semanticamente próximas de documentos corretos, mas solicitavam fatos ausentes, como número de vagas, datas de resultados ou regras não especificadas.

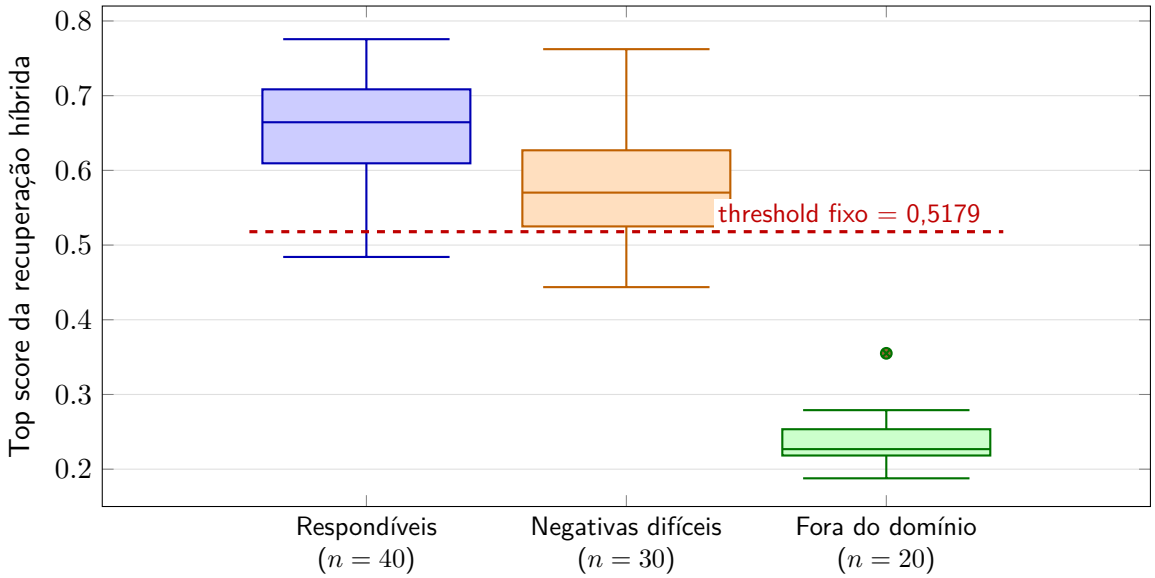


Figura 8: Distribuição dos top scores por classe no conjunto adicional de validação

A Figura 8 mostra separação completa das perguntas fora do domínio, mas também sobreposição entre consultas respondíveis e negativas difíceis. Esse comportamento confirma que o *top score* mede principalmente proximidade temática: ele é adequado para descartar consultas claramente irrelevantes, mas não substitui a verificação, pelo modelo e pelos guardrails, de que o fato específico esteja explícito no contexto.

O valor fixo preservou 39 das 40 perguntas respondíveis. Na análise diagnóstica, cortes entre 0,585 e 0,590 elevaram a acurácia balanceada para 0,815, porém reduziram a sensibilidade para 85%, o que corresponderia ao bloqueio de 6 das 40 perguntas válidas. Como a calibração priorizou evitar recusas indevidas, 0,5179 foi mantido como compromisso conservador, sem reajuste a partir do *holdout*.

As três repetições produziram exatamente a mesma matriz de confusão, sem mudança de classificação e com variação máxima de score igual a zero no ambiente avaliado. Essa estabilidade sustenta a reprodutibilidade local do corte, embora não implique universalidade para outros corpora ou modelos de embedding.

## 7.6 Impacto do Cache

O cache de perguntas repetidas foi avaliado comparando a latência de uma primeira consulta com a latência da mesma consulta repetida dentro da mesma versão do corpus.

| Métrica | Cache Miss | Cache Hit |
|---|---|---|
| Tempo total de resposta | 3347,84 ms | 10,00 ms |
| Tempo até primeiro conteúdo | 3347,25 ms | 9,88 ms |

| | | |
|---|---|---|
| Chamada ao modelo de linguagem | Sim | Não |
| Chamada ao banco vetorial | Sim | Não |
| Redução percentual de latência | – | 99,70% |

Os resultados indicam que o cache é uma estratégia adequada para períodos de alta demanda, como divulgação de editais, quando muitos estudantes realizam perguntas idênticas ou muito semelhantes. A escolha por cache exato normalizado reduz risco de respostas incorretas quando comparada a uma abordagem semântica mais agressiva.

## 7.7 Desempenho do Sistema

O desempenho foi avaliado com base no benchmark fixo definido na metodologia. Foram comparadas a configuração inicial e a configuração final após otimizações no prompt, limite de contexto, geração, cache, threshold e streaming.

| **Métrica** | **Configuração Inicial** | **Configuração Final** |
|---|---|---|
| Modelo gerador | `qwen3:30b` | `qwen2.5:7b-instruct` |
| Modo de geração | Com *thinking* | Sem *thinking* |
| Perfil de prompt | Padrão | `guarded_short` |
| Top-k de recuperação | 5 | 5 |
| Limite de contexto | 12000 caracteres | 12000 caracteres |
| Limite de geração | 1200 tokens | 450 tokens |
| p50 de tempo total | 43,80 s | 3,83 s |
| p95 de tempo total | 70,23 s | 5,39 s |
| p50/p95 até primeiro conteúdo | 32,48 s / 55,42 s | 3,83 s / 5,39 s |
| Respostas vazias | 3/9 | 0/9 |
| Encerramento por limite | 3/9 | 0/9 |
| Encerramento normal | 6/9 | 9/9 |
| Fidelidade básica | 5/9 | 8/9 |
| Vazamento de raciocínio | 0/9 | 0/9 |

Os resultados mostram que a configuração inicial apresentava latência incompatível com uso interativo, mesmo com a recuperação documental funcionando corretamente. A maior parte do tempo era consumida pela etapa de geração com *thinking*, e não pela busca vetorial. A configuração final reduziu o p50 de 43,80 segundos para 3,83 segundos e o p95 de 70,23 segundos para 5,39 segundos, eliminando respostas vazias e encerramentos por limite de geração no conjunto avaliado.

Essa mudança demonstra que, neste corpus e nesta configuração experimental, o principal gargalo do sistema estava no modelo gerador. A recuperação com `top_k=5` e contexto máximo de 12000 caracteres foi mantida, pois a redução agressiva do contexto prejudicou a cobertura documental em testes intermediários.

| **Configuração** | **p50** | **p95** | **Vazias** | **Fidelidade** | **Resultado** |
|---|---|---|---|---|---|
| `qwen3:30b` com *thinking* | 43,80 s | 70,23 s | 3/9 | 5/9 | Rejeitada por latência e respostas vazias. |

| | | | | | |
|---|---|---|---|---|---|
| Prompt curto com `top_k=3` | 44,05 s | 46,32 s | 5/9 | 2/9 | Rejeitada por perda de cobertura e fidelidade. |
| `qwen3:30b` sem *thinking* | 37,60 s | 44,70 s | 3/9 | 2/9 | Rejeitada por baixa fidelidade e respostas vazias. |
| `qwen2.5:7b-instruct` com *guard* | 3,83 s | 5,39 s | 0/9 | 8/9 | Escolhida como configuração padrão. |

A matriz de comparação evidencia que reduzir apenas o tamanho do contexto ou desativar o *thinking* no modelo `qwen3:30b` não foi suficiente para resolver o problema. O prompt curto reduziu parcialmente o p95, mas aumentou o número de respostas vazias e reduziu a fidelidade. O modo sem *thinking* também não alcançou comportamento satisfatório. A melhor configuração foi obtida com a troca do modelo gerador para `qwen2.5:7b-instruct`, associada a um prompt mais curto e a uma validação explícita da fonte retornada.

## 7.8 Avaliação Qualitativa das Respostas

Além das métricas quantitativas, foram analisados exemplos de respostas geradas pelo sistema. A avaliação qualitativa considerou clareza, aderência ao contexto, indicação da fonte e comportamento diante de perguntas sem evidência.

| **Pergunta** | **Resposta Resumida** | **Fonte** | **Avaliação** |
|---|---|---|---|
| Quais são as regras do estágio curricular obrigatório no IFBA? | Resposta em quatro itens, com carga horária mínima, limite diário, exigência do PAE e relatórios semestrais. | `documento_8.pdf` | Adequada: objetiva, fiel ao contexto e com fonte correta. |
| Como faço um requerimento no IFBA? | Resposta direta indicando o caminho no SUAP para abrir novo requerimento e anexar atestado. | `documento_6.pdf` | Adequada: curta, acionável e com fonte correta. |
| Qual é o número de tomadas elétricas da biblioteca do campus Salvador? | Retorno explícito da mensagem canônica de ausência de evidência documental. | Sem fonte | Adequada: recusou sem inventar informação. |

De forma geral, as respostas avaliadas demonstraram que o uso de RAG contribui para maior rastreabilidade e controle factual. A exigência de fonte documental ao final da resposta permite que o usuário verifique a origem da informação e reduz a opacidade normalmente associada a sistemas baseados em modelos de linguagem.

## 7.9 Ameaças à Validade

A validade interna da avaliação inicial era limitada pelo uso de conjuntos de perguntas relacionados durante a calibração e a medição do limiar. Essa ameaça foi reduzida pela validação adicional com 90 perguntas congeladas antes da primeira recuperação e não utilizadas para ajustar o valor `0,5179`. Ainda assim, os rótulos foram definidos por inspeção documental do próprio corpus, sem avaliadores independentes ou procedimento cego.

A validade externa é limitada pelo tamanho do corpus, composto por dez documentos curtos. Na configuração avaliada, cada documento resultou em apenas um chunk, o que impediu observar de forma representativa os efeitos das diferentes configurações de chunking em documentos extensos. Portanto, os resultados de 400/50, 800/150 e 1000/150 não devem ser generalizados para bases documentais maiores sem nova avaliação.

A validade de construto é limitada porque o maior score da recuperação mede proximidade temática, não a presença do fato específico solicitado. Esse limite foi evidenciado pelas negativas difíceis: 23 de 30 ultrapassaram o corte mesmo sem resposta explícita no corpus. Portanto, o gate deve ser interpretado como uma camada conservadora de filtragem, combinada com prompt restritivo e validação de fonte, e não como classificador completo de suficiência factual.

As verificações automatizadas baseadas em termos esperados, fontes recuperadas e fidelidade básica permitem comparação reproduzível, mas não substituem avaliação humana ampla nem a execução integral de frameworks especializados de avaliação RAG. Além disso, a comparação histórica de latência modifica simultaneamente modelo, thinking, limite de geração, prompt e guardrails; por isso, o ganho observado pertence ao conjunto de otimizações e não pode ser atribuído causalmente a apenas uma delas.

# 8 Implantação

## 8.1 Projeto de Implantação

A solução foi concebida para implantação em ambiente baseado em contêineres Docker, garantindo portabilidade, reprodutibilidade e isolamento dos serviços. O ambiente de desenvolvimento foi verificado por meio do Docker Compose; a topologia de produção apresentada nesta seção constitui o projeto de implantação da pilha RAG, pois o manifesto de produção atual do Emile ainda não declara todos os serviços do chatbot.

### Componentes da Infraestrutura

A infraestrutura é composta pelos seguintes serviços:

- **chatbot-api**: nome lógico do contêiner responsável pelo backend FastAPI, incluindo os endpoints de consulta, ingestão e monitoramento. No Compose de desenvolvimento, o serviço é denominado `backend` e sua instância `emile-backend`;
- **vector-db**: nome lógico do PostgreSQL com PGVector. No ambiente de desenvolvimento, corresponde ao serviço/hostname `vector_db` e à instância `emile-vector-db`. A porta 5432 é publicada para diagnóstico local; no projeto de produção, o banco deve permanecer acessível apenas pela rede interna da solução;
- **corpus documental**: no desenvolvimento, o diretório `/app/data` chega ao backend pelo bind mount amplo `../src:/app`. Para produção, o projeto prevê volume persistente específico para os PDFs e arquivos de controle, separado da imagem da aplicação;
- **Ollama (ollama-api.ifba.edu.br)**: infraestrutura já disponibilizada pelo IFBA, responsável pela execução do modelo de embedding `qwen3-embedding:4b` e do modelo gerador `qwen2.5:7b-instruct`. Acessado externamente via HTTPS pelo serviço da API, sem necessidade de implantação adicional;
- **LlamaParse/LlamaCloud**: serviço externo acionado via HTTPS somente durante a ingestão,

com autenticação pela variável `LLAMA_CLOUD_API_KEY`. Ele não participa do caminho normal de consulta.

## Diagrama C4 de Implantação

A Figura 9 utiliza a notação C4 para distinguir nós de implantação, instâncias de contêiner e sistemas externos. Os nomes efetivos do ambiente de desenvolvimento aparecem nas instâncias, enquanto os limites dos nós representam a topologia proposta para a implantação da pilha RAG.

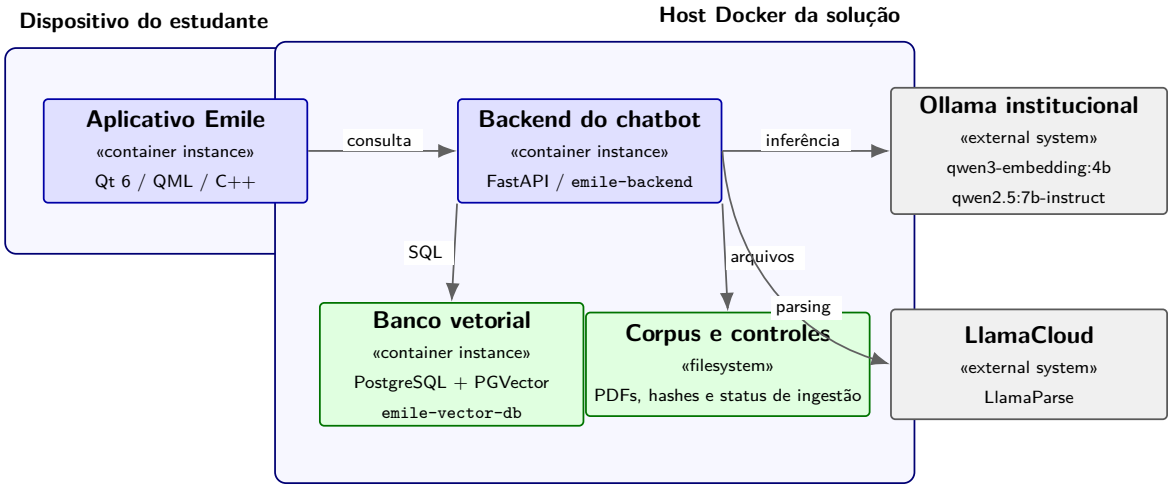


Figura 9: Diagrama C4 de implantação proposto para a pilha RAG

No fluxo de consulta, o aplicativo envia a pergunta ao backend e recebe eventos NDJSON progressivos. O backend recupera os trechos no PGVector e acessa o Ollama para embeddings e geração. No fluxo de ingestão, o backend lê o corpus persistente, utiliza o LlamaParse para extração textual e grava chunks, vetores e metadados no banco vetorial. A separação dos serviços externos evidencia que LlamaCloud e Ollama não são contêineres implantados pelo projeto.

## Configuração por Ambiente

Todos os parâmetros sensíveis e específicos de ambiente são definidos via variáveis de ambiente, permitindo que o mesmo contêiner seja executado em desenvolvimento, homologação e produção sem alteração de código. As principais variáveis estão agrupadas por responsabilidade:

- **inferência**: `OLLAMA_BASE_URL`, `OLLAMA_CHAT_MODEL` e `OLLAMA_NUM_PREDICT`;
- **persistência**: `PG_DATABASE`, `PG_HOST`, `PG_USER`, `PG_PASSWORD` e `PG_PORT`;
- **pipeline RAG**: `CHAT_CHUNK_SIZE`, `CHAT_CHUNK_OVERLAP`, `CHAT_VECTOR_TABLE`, `CHAT_CONFIDENCE_MIN_TOP_SCORE` e `CHAT_CACHE_TTL_SECONDS`;
- **ingestão**: `LLAMA_CLOUD_API_KEY`, utilizada somente pelo LlamaParse durante o processamento dos PDFs.

No cliente, a variável

`EMILE_CHATBOT_BACKEND_BASE_URL`

mantém a URL do serviço RAG desacoplada da URL do backend principal do Emile.

## Atualização da Base Documental

No projeto de produção, novos documentos institucionais são disponibilizados em um volume persistente associado ao diretório `./data` do contêiner da API. No desenvolvimento, o mesmo caminho lógico é disponibilizado pelo bind mount do código fonte. O processo de ingestão é disparado via `POST /chatbot/ingest`, podendo ser automatizado por tarefas agendadas para execução periódica — por exemplo, diária em períodos de publicação de editais. O controle por hash MD5 garante que apenas arquivos novos ou modificados sejam reprocessados.

O diretório `./data` armazena os arquivos PDF de origem e os arquivos JSON de controle da ingestão. Os embeddings e os chunks indexados não permanecem nesse diretório: eles são persistidos no PostgreSQL com PGVector. Durante as consultas normais, o chatbot recupera os trechos do banco vetorial e não realiza leitura direta dos PDFs.

## Integração com o Aplicativo Emile

A integração com o Emile é realizada por meio do endpoint autenticado `POST /chatbot/ask`, responsável por receber a pergunta do usuário e retornar a resposta em streaming. A URL do serviço de chatbot é configurável de forma independente da URL do backend principal do Emile, permitindo que o serviço RAG seja implantado, escalado e monitorado separadamente.

No cliente, a interface QML consome os eventos NDJSON emitidos pelo backend, atualizando a conversa conforme os fragmentos da resposta são recebidos. O aplicativo também trata estados de erro, resposta vazia, timeout e endpoint indisponível, apresentando mensagens claras ao usuário.

A arquitetura desacoplada permite que o chatbot seja executado como serviço independente, preservando a possibilidade de integração futura com outros módulos institucionais sem reescrever a lógica de recuperação e geração.

## Monitoramento Operacional

A implantação inclui registro de métricas estruturadas por requisição, permitindo acompanhar desempenho, falhas e comportamento do pipeline. Entre os indicadores monitorados estão latência total, tempo de recuperação, tempo de geração, taxa de cache hit, respostas vazias, encerramentos por limite de geração e erros de comunicação com o serviço de modelo.

Essas métricas possibilitam detectar degradações de desempenho durante períodos de alta demanda, identificar documentos com baixa qualidade de recuperação e orientar ajustes posteriores no pipeline RAG.

## Segurança da Implantação

Os endpoints do chatbot são protegidos por autenticação via token, reaproveitando o mecanismo de autenticação já existente no backend do Emile. O banco vetorial é mantido em rede interna, sem exposição pública direta. Credenciais, tokens e conteúdo completo dos documentos não são registrados nos logs de benchmark ou monitoramento.

Além das proteções tradicionais de API, a proteção do chatbot se concentra nas restrições do prompt, na interrupção por ausência de evidência e na validação da saída e da fonte documental. Não há classificador independente de prompt injection ou jailbreak na implementação atual.

# 9 Manual do Usuário

## 9.1 Visão Geral do Chatbot

O chatbot institucional do IFBA é um assistente conversacional integrado ao aplicativo Emile, desenvolvido para auxiliar estudantes, servidores e demais membros da comunidade acadêmica na busca por informações presentes em documentos oficiais da instituição. O sistema responde perguntas em linguagem natural sobre editais, calendários acadêmicos, regulamentos, portarias e demais documentos indexados em sua base, indicando a fonte documental correspondente a cada resposta.

## 9.2 Acesso ao Sistema

O chatbot é acessado diretamente pelo aplicativo Emile, plataforma já utilizada pela comunidade acadêmica do IFBA. Não é necessária instalação de software adicional nem criação de nova conta de acesso. A autenticação é realizada automaticamente pelo mecanismo de login existente no Emile. O usuário acessa a funcionalidade de chatbot pela interface de conversação disponibilizada no aplicativo.

## 9.3 Realizando Consultas

Para realizar uma consulta, o usuário digita sua pergunta no campo de texto da interface de conversação e envia a mensagem normalmente. O sistema processará a consulta, realizará a recuperação dos trechos mais relevantes na base documental e retornará a resposta em instantes.

As perguntas podem ser formuladas livremente em linguagem natural, sem necessidade de terminologia técnica, palavras-chave específicas ou formatação especial. O sistema é capaz de interpretar consultas informais e relacioná-las aos documentos institucionais disponíveis.

Exemplos de perguntas que o sistema está preparado para responder:

- "Quais são as regras do estágio curricular obrigatório no IFBA?"
- "Como faço um requerimento no IFBA?"
- "Quais bolsas e auxílios estudantis estão disponíveis no IFBA?"
- "Quais são os prazos do edital de assistência estudantil?"
- "O que fazer quando essa informação não aparece nos documentos disponíveis?"

## 9.4 Interpretando as Respostas

Cada resposta gerada pelo chatbot é fundamentada nos documentos institucionais indexados em sua base. A resposta indica a fonte documental utilizada, informando o nome do arquivo ou edital de origem das informações apresentadas. Essa rastreabilidade permite que o usuário consulte o documento completo caso necessite de mais detalhes ou queira verificar a informação em seu contexto original — prática recomendada especialmente para decisões acadêmicas importantes, como inscrições em programas ou solicitação de benefícios.

Quando o sistema identificar que a informação solicitada não consta nos documentos indexados, a resposta informará explicitamente essa limitação, orientando o usuário a buscar a informação diretamente no portal institucional do IFBA ou junto ao setor responsável. O chatbot não especula nem infere respostas para perguntas que não encontram respaldo documental.

## 9.5 Sugestões Rápidas

A interface do chatbot apresenta sugestões rápidas de perguntas relacionadas a temas recorrentes, como bolsas e auxílios, estágio, requerimentos, CadÚnico e prazos de editais. Essas sugestões têm o objetivo de auxiliar o usuário a formular consultas mais específicas e compatíveis com os documentos institucionais disponíveis.

Ao tocar em uma sugestão, o aplicativo envia uma pergunta completa ao assistente, em vez de apenas uma palavra-chave isolada. Essa decisão melhora a recuperação documental, pois fornece mais contexto ao sistema sobre a intenção do usuário.

## 9.6 Mensagens do Sistema

Durante o uso do chatbot, algumas mensagens podem aparecer para indicar o estado da requisição ou limitações da resposta:

| **Mensagem** | **Significado** |
|---|---|
| `Consultando os documentos...` | O sistema está buscando trechos relevantes na base documental e preparando a resposta. |
| `Não encontrei essa informação nos documentos disponíveis no momento.` | A base documental indexada não contém evidência suficiente para responder à pergunta. |
| `Não foi possível obter uma resposta. Tente novamente.` | Ocorreu uma falha técnica, como timeout, erro de rede ou resposta vazia do serviço de geração. |
| `A resposta foi interrompida antes de terminar.` | O modelo atingiu o limite de geração antes de concluir a resposta. |

## 9.7 Limitações do Sistema

O chatbot responde exclusivamente com base nos documentos institucionais do IFBA que foram processados e indexados em sua base. As principais limitações a serem consideradas pelo usuário são:

- **Escopo documental**: informações não presentes nos documentos indexados não poderão ser fornecidas pelo sistema. Situações individuais de matrícula, histórico acadêmico pessoal e dados de outros sistemas institucionais estão fora do escopo do chatbot;
- **Atualização da base**: a base documental é atualizada periodicamente. Em caso de publicação recente de um edital ou resolução, o documento pode ainda não estar disponível no sistema. Para informações de publicação muito recente, recomenda-se verificar diretamente no portal institucional;
- **Domínio restrito**: perguntas que não estejam relacionadas a documentos institucionais do IFBA — como consultas sobre outros cursos ou instituições, temas pessoais ou assuntos não relacionados à vida acadêmica — serão recusadas pelo sistema;
- **Não substitui atendimento humano**: para situações que exijam análise individualizada, tratamento de exceções às normas ou decisões que dependam de autorização de um servi-

dor, o usuário deve contatar diretamente a secretaria acadêmica ou a coordenação de curso responsável.

## 9.8 Boas Práticas de Uso

Para obter respostas mais precisas e completas do chatbot, recomenda-se seguir as orientações abaixo:

- **Seja específico na pergunta**: prefira "Quais são os critérios de seleção socioeconômica do edital de auxílio estudantil 2026.1?" a "me fala sobre os auxílios". Perguntas mais específicas produzem recuperações documentais mais precisas;
- **Mencione o documento quando souber**: se o usuário souber o nome do edital, regulamento ou resolução de interesse, incluí-lo na pergunta auxilia o sistema a identificar o documento correto com maior precisão, especialmente quando há múltiplos documentos com conteúdo similar na base;
- **Reformule se a resposta não for satisfatória**: caso a resposta não atenda à dúvida, tente reformular a pergunta com outros termos ou divida-a em perguntas mais específicas. Consultas muito amplas podem retornar trechos de múltiplos documentos simultaneamente, reduzindo o foco da resposta;
- **Verifique a fonte indicada**: ao receber uma resposta, observe o documento de origem indicado pelo sistema. Para decisões acadêmicas importantes, é recomendado consultar o trecho original no documento fonte para confirmar a informação em contexto completo.

# 10 Conclusão

Este trabalho apresentou o desenvolvimento e a avaliação de um chatbot institucional baseado em Retrieval-Augmented Generation, integrado ao aplicativo Emile, com o objetivo de facilitar o acesso de estudantes a informações presentes em documentos oficiais do IFBA.

O problema abordado envolve a dificuldade de localizar respostas específicas em arquivos PDF institucionais, como editais, regulamentos, calendários e portarias. Embora esses documentos estejam disponíveis digitalmente, sua consulta manual pode ser lenta, especialmente quando o usuário precisa navegar por diversos arquivos ou interpretar linguagem administrativa formal. Além disso, o uso direto de modelos de linguagem sem recuperação documental apresenta risco de alucinação, o que é inadequado em um contexto institucional.

A solução desenvolvida utiliza uma arquitetura RAG modular. O sistema processa documentos PDF por meio de um pipeline de ingestão automatizada, extrai texto estruturado, segmenta os documentos em chunks, gera embeddings, armazena os dados em PostgreSQL com PGVector e realiza recuperação híbrida combinando busca semântica e busca lexical. Sobre essa base, foram implementadas etapas adicionais de query rewriting, limiar de confiança, re-ranking, cache de perguntas e geração de respostas com citação de fonte documental.

A integração com o aplicativo Emile permitiu disponibilizar o chatbot em uma interface conversacional familiar ao usuário, com mensagem inicial explicativa, sugestões rápidas, avatar do assistente, resposta em streaming e tratamento explícito de erros. Essa integração demonstra a viabilidade de incorporar sistemas RAG a aplicações institucionais já existentes, sem exigir que o estudante acesse uma plataforma externa para consultar informações.

A avaliação do sistema mostrou que a recuperação documental não foi o principal gargalo de

desempenho. O maior custo de latência esteve associado à etapa de geração textual pelo modelo de linguagem. Essa constatação orientou otimizações como controle explícito do contexto, troca do modelo gerador, limitação da geração, uso de cache e interrupção antecipada em casos sem evidência documental suficiente.

Após as otimizações, o sistema apresentou os seguintes resultados principais no benchmark fixo:

- redução do p50 de tempo total de 43,80 segundos para 3,83 segundos;
- redução do p95 de tempo total de 70,23 segundos para 5,39 segundos;
- eliminação de respostas vazias no conjunto avaliado, passando de 3/9 para 0/9;
- eliminação de encerramentos por limite de geração, passando de 3/9 para 0/9;
- manutenção de zero casos de vazamento de raciocínio;
- aumento da fidelidade básica de 5/9 para 8/9;
- preservação de 39/40 perguntas respondíveis e bloqueio de 27/50 consultas negativas na validação adicional do limiar, com classificação idêntica em três repetições.

Esses resultados indicam que a escolha do modelo gerador foi decisiva para tornar o chatbot utilizável em contexto interativo. Na configuração inicial, o modelo baseado em *thinking* consumia grande parte do tempo de resposta na etapa de geração deliberativa, chegando a encerrar requisições sem produzir conteúdo visível. A configuração final com `qwen2.5:7b-instruct`, prompt curto e validação de fonte reduziu drasticamente a latência sem liberar respostas sem fonte recuperada ou raciocínio interno ao usuário.

Os resultados indicam que a abordagem RAG foi adequada no corpus e no conjunto de avaliação utilizados, pois combina atualização documental, rastreabilidade das fontes e geração de respostas em linguagem natural. A validação adicional mostrou que o limiar preserva alta sensibilidade e elimina consultas claramente fora do domínio, mas não distingue sozinho todos os fatos ausentes em documentos tematicamente próximos. Por isso, a redução do risco de alucinação resulta da atuação conjunta entre recuperação, gate, prompt restritivo e validação da fonte.

Apesar dos resultados obtidos, a solução possui limitações. A qualidade das respostas depende da atualização e cobertura dos documentos indexados, da qualidade da extração textual dos PDFs e do desempenho do modelo gerador utilizado. Documentos mal estruturados, escaneados ou desatualizados podem prejudicar a recuperação. Além disso, a avaliação realizada utiliza um conjunto controlado de perguntas, sendo necessária validação contínua em uso real para acompanhar novas formas de consulta e novos documentos institucionais.

Conclui-se que o chatbot desenvolvido cumpre o objetivo de automatizar a consulta a informações institucionais por meio de uma arquitetura RAG. A solução demonstra que é possível integrar recuperação documental, geração de linguagem natural, controle de confiabilidade e experiência conversacional em um sistema voltado ao atendimento de estudantes, preservando a rastreabilidade das fontes oficiais e reduzindo a dependência de busca manual em documentos PDF.

# Agradecimentos

Agradeço especialmente à minha família e aos meus amigos por todo o apoio recebido ao longo desta jornada, sobretudo nos momentos mais difíceis.

Dedico este trabalho, e tudo o que ele representa, às minhas avós, Neri e Fátima. Que eu possa continuar orgulhando vocês, hoje e sempre.

*"Levante-se e ande. Continue seguindo em frente. Você tem duas pernas boas. Então, levante-se e use-as. Você é forte o suficiente para criar seu próprio caminho."*

*— Edward Elric, Fullmetal Alchemist*

# Referências

1 LEWIS, P. et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. **Advances in Neural Information Processing Systems**, v. 33, p. 9459–9474, 2020.

2 GAO, Y. et al. Retrieval-augmented generation for large language models: A survey. **arXiv preprint arXiv:2312.10997**, 2023.

3 KARPUKHIN, V. et al. Dense passage retrieval for open-domain question answering. **Proceedings of EMNLP**, 2020.

4 REIMERS, N.; GUREVYCH, I. Sentence-bert: Sentence embeddings using siamese bert-networks. **Proceedings of EMNLP**, 2019.

5 ES, S. et al. Ragas: Automated evaluation of retrieval augmented generation. In: **Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics**. [S.l.: s.n.], 2024.

# Apêndices

## Apêndice A — Conjunto de Perguntas do Benchmark

| ID | Categoria | Pergunta | Tipo |
|---|---|---|---|
| Q01 | Bolsas e auxílios | Quais bolsas e auxílios estudantis estão disponíveis no IFBA? | Respondível |
| Q02 | Estágio | Quais são as regras do estágio curricular obrigatório no IFBA? | Respondível |
| Q03 | Requerimentos | Como faço um requerimento no IFBA? | Respondível |
| Q04 | CadÚnico | Quais informações os documentos do IFBA trazem sobre Cadastro Único (CadÚnico), critérios, cadastro e uso em auxílios da assistência estudantil? | Respondível |
| Q05 | Prazos de edital | Quais são os prazos do edital de assistência estudantil vigente? | Respondível |
| Q06 | Sem evidência | Qual é o número de tomadas elétricas da biblioteca do campus Salvador? | Sem cobertura |
| Q07 | Fora do escopo | Como preparar um bolo de chocolate? | Fora do domínio |
| Q08 | Sequencial | Sequência: Quem pode solicitar auxílio estudantil no IFBA? → Como funciona o estágio curricular obrigatório no IFBA? | Robustez |
| Q09 | Cache | Como faço um requerimento no IFBA? (pergunta repetida) | Cache |

## Apêndice B — Configuração Final do Sistema

| Parâmetro | Valor |
|---|---|
| Modelo de embedding | `qwen3-embedding:4b` |
| Modelo gerador | `qwen2.5:7b-instruct` |
| Modo de thinking | Desativado |
| Perfil de prompt | `guarded_short` |
| Chunk size | 400 |
| Chunk overlap | 50 |
| Top-k inicial | 5 |
| Re-ranking padrão | Desativado |
| Top-k final | 5 |
| Estratégia opcional de re-ranking | `embedding_hybrid` |
| Limiar de confiança | 0,5179 |
| Limite máximo de contexto | 12000 caracteres |
| Limite de geração | 450 tokens |
| Temperatura | 0,2 |
| Keep alive | 10 minutos |
| TTL do cache | 300 s |
| Banco vetorial | PostgreSQL + PGVector |
| Framework RAG | LlamaIndex |
| Parser | LlamaParse |
| Artefato de benchmark final | `20260710T021839Z-80d3b378.json` |

# Apêndice C — Prompt Final Utilizado

```
Siga estas regras na ordem. Primeiro classifique a pergunta.
1. Se não for sobre o IFBA, responda somente: Só posso ajudar com assuntos institucionais do
2. Se for sobre o IFBA mas o CONTEXTO não trouxer a resposta explícita,
   responda somente: Não encontrei essa informação nos documentos disponíveis
   no momento.
3. Se o CONTEXTO trouxer a resposta, use no máximo 4 frases ou 4 itens e termine
   com Fonte-ID: N, trocando N pelo número do cabeçalho que sustenta a resposta.
   Não escreva nem invente título de documento.
Responda em português do Brasil usando somente o CONTEXTO. Entregue apenas a
resposta final, sem análise ou raciocínio. Nunca invente.

CONTEXTO:
[Fonte 1: documento_1.pdf]
Trecho de teste.

PERGUNTA:
Pergunta teste
```

# Apêndice D — Exemplo de Log Estruturado


```
{
  "event": "chat_request_completed",
  "request_id": "bench-20260710T021839Z-80d3b378-03",
  "embedding_ms": 150.75,
  "retrieval_ms": 7.15,
  "reranking_ms": 0.0,
  "first_content_ms": 3832.02,
  "total_ms": 3832.45,
  "retrieved_nodes": 5,
  "context_chars": 2426,
  "cache_hit": false,
  "done_reason": "stop"
}
```


# Apêndice E — Resultados Consolidados do Benchmark

| Métrica | Antes | Depois |
|---|---|---|
| Modelo gerador | `qwen3:30b` | `qwen2.5:7b-instruct` |
| p50 | 43,80 s | 3,83 s |
| p95 | 70,23 s | 5,39 s |
| p50/p95 até primeiro conteúdo | 32,48 s / 55,42 s | 3,83 s / 5,39 s |
| Taxa de erro técnico | 0/9 | 0/9 |
| Taxa de resposta vazia | 3/9 | 0/9 |
| Encerramento por limite | 3/9 | 0/9 |
| Encerramento normal | 6/9 | 9/9 |
| Fidelidade básica | 5/9 | 8/9 |
| Vazamento de raciocínio | 0/9 | 0/9 |

| | | |
|---|---|---|
| Taxa de cache hit | Não aplicável ao benchmark fixo | 1/1 no teste de repetição exata |

# Apêndice F — Protocolo da Validação Adicional do Limiar

| **Elemento** | **Valor** |
|---|---|
| Threshold avaliado | `0,5179`, mantido fixo |
| Conjunto | 90 perguntas: 40 respondíveis, 30 negativas difíceis e 20 fora do domínio |
| Repetições | 3 execuções integrais; 270 recuperações |
| Geração textual | Não executada; coleta encerrada após o gate |
| Configuração | `top_k`=5, re-ranking desativado, tabela `tcc_documentos_ifba_v6`, chunking 400/50 |
| Hash SHA-256 do conjunto | `14887ba44eac8577b0764b560f6457a3 4a55c2cff2853e90bb04b6f1d0473389` |
| Matriz de confusão | TP=39, TN=27, FP=23 e FN=1 |
| Estabilidade | Nenhuma mudança de score ou classificação entre as três repetições |

Os scripts e dados brutos utilizados na revisão foram preservados no diretório:

`benchmarks/final_revision`

Os artefatos incluem o conjunto congelado, scores por repetição, métricas agregadas, análise de sensibilidade e os dados dos box plots. O código de produção, o corpus e o banco vetorial não foram modificados durante a coleta.

# Glossário, Siglas e Abreviações

| Termo | Definição |
|---|---|
| API | Interface de Programação de Aplicações, utilizada para comunicação entre sistemas. |
| ANN | Approximate Nearest Neighbor, técnica para busca aproximada de vizinhos mais próximos em espaços vetoriais. |
| BM25 | Algoritmo clássico de recuperação lexical baseado em frequência e relevância de termos. |
| Chunk | Trecho de texto extraído de um documento maior para indexação e recuperação. |
| Chunking | Processo de divisão de documentos em trechos menores. |
| C4 Model | Modelo de visualização de arquitetura que organiza software em níveis de contexto, contêineres, componentes e implantação. |
| Dense Retrieval | Recuperação baseada em embeddings densos e similaridade vetorial. |
| Embedding | Representação vetorial numérica de um texto, utilizada para medir similaridade semântica. |
| Faithfulness | Métrica que avalia se a resposta gerada é sustentada pelo contexto documental recuperado. |
| FastAPI | Framework Python utilizado para desenvolvimento da API backend. |
| Golden Dataset | Conjunto de perguntas e respostas de referência utilizado para avaliação do sistema. |
| Guardrails | Mecanismos de proteção que limitam o comportamento do modelo e reduzem respostas inadequadas. |
| Holdout | Conjunto de validação mantido separado dos dados usados para calibração. |
| Hybrid Retrieval | Estratégia de recuperação que combina busca vetorial densa e busca lexical. |
| LLM | Large Language Model, modelo de linguagem de grande escala utilizado para geração textual. |
| LlamaIndex | Framework utilizado para orquestração do pipeline RAG. |
| LlamaParse | Ferramenta utilizada para extração estruturada de conteúdo de documentos PDF. |
| NDJSON | Newline Delimited JSON, formato de streaming no qual cada linha contém um objeto JSON independente. |
| Ollama | Plataforma utilizada para hospedagem e execução local de modelos de linguagem e embeddings. |
| PGVector | Extensão do PostgreSQL que adiciona suporte a armazenamento e busca de vetores. |
| Prompt Injection | Tentativa de manipular o modelo por meio de instruções maliciosas inseridas na consulta. |
| Query Caching | Estratégia de cache baseada no armazenamento de respostas para perguntas repetidas. |
| Query Rewriting | Processo de reescrita ou expansão da pergunta do usuário para melhorar a recuperação documental. |
| RAG | Retrieval-Augmented Generation, arquitetura que combina recuperação de documentos com geração por modelo de linguagem. |

| | |
|---|---|
| RAGAS | Framework de avaliação de sistemas RAG. |
| Re-ranking | Reordenação dos trechos recuperados para priorizar os mais relevantes antes da geração. |
| Sparse Retrieval | Recuperação lexical baseada em correspondência de termos. |
| Streaming | Envio progressivo da resposta ao cliente durante sua geração. |
| Threshold | Valor de corte aplicado ao score de recuperação para decidir se o fluxo deve prosseguir até o modelo gerador. |
| Vector Store | Banco ou estrutura de armazenamento voltada à persistência e busca de vetores. |